چاپ چهارم

پیشرفته

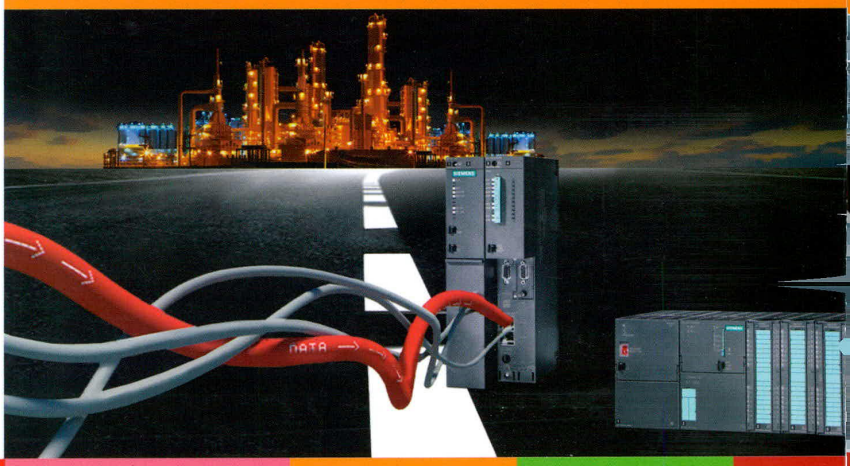
کاملترین مرجع کاربردی

PLC S7 SIEMENS

مهندس محمدرضا ماهر
مهندس احمد فرجی

- دارای تأییدیه گروه های برنامه ریزی درسی سازمان فنی و حرفه ای

FC / FB | Interrupts | PID Control | S7 40



DVD

...خت افزار S7 400 – کار با سیگنال های آنالوگ – استفاده از
D – برنامه نویسی ساختار یافته با FB و FC – آشنایی با SFB و SFC
...نامه نویسی راه اندازی PLC – برنامه نویسی وقفه ها – کنترل PID
PLC – نحوه عیب یابی در PLC و ...

کاملترین مرجع کاربردی

PLC S7

(سطح پیشرفته)

نوشتهٔ
مهندس محمدرضا ماهر
مهندس احمد فرجی

انتشارات نگارنده دانش

سرشناسه	: ماهر، محمدرضا، ۱۳۴۲-
عنوان و نام پدیدآور	: کاملترین مرجع کاربردی PLC S7 (سطح پیشرفته)/ محمدرضا ماهر، احمد فرجی.
مشخصات نشر	: تهران: نگارنده دانش، ۱۳۹۰.
مشخصات ظاهری	: ۶۵۶ ص. (چهارده + ۶۴۲): مصور، جدول، نمودار.
شابک	: ۹۷۸-۶۰۰-۶۱۹۰-۰۲-۰
وضعیت فهرست‌نویسی	: فیپا
یادداشت	: کتابنامه: ص.۶۳۴.
موضوع	: استپ (زبان برنامه‌نویسی کامپیوتر)
موضوع	: کنترل‌کننده‌های برنامه‌پذیر
شناسه افزوده	: فرجی، احمد، ۱۳۶۲-
رده‌بندی کنگره	: ۱۳۹۰ ۲۳۲م۹ک/TJ۲۲۳
رده‌بندی دیویی	: ۶۲۹/۸۹۵
شمارهٔ کتابشناسی ملی	: ۲۳۱۰۶۷۶

کاملترین مرجع کاربردی PLC S7 (سطح پیشرفته)

نوشتهٔ	مهندس محمدرضا ماهر
	مهندس احمد فرجی
حروفچینی	واحد تولید نشر نگارنده دانش
لیتوگرافی، چاپ، صحافی	گلپاگرافیک، فرشیوه، خیام
چاپ اول / چاپ دوم	پاییز ۱۳۹۰ / تابستان ۱۳۹۱
چاپ سوم / چاپ چهارم	بهار ۱۳۹۲ / زمستان ۱۳۹۲
تعداد صفحات، قطع	۶۵۶، وزیری
تعداد	۲۰۰۰ نسخه
بها به همراه DVD	۲۷۰۰۰ تومان

www.negarandedanesh.com

مراکز فروش: نگارنده دانش تلفن: ۶۶۹۶۲۰۵۳ – ۶۶۹۶۲۳۰۵

آزاده تلفن: ۶۶۴۱۵۷۵۳

سخن ناشر

با توسعهٔ روزافزون دانشگاه‌ها و مؤسسات آموزش عالی در کشور، اهمیّت انتشار کتاب‌ها و منابع علمی مناسب برای کسانی که به پژوهش، تدریس، یا تحصیل در سطوح مختلف آموزش عالی مشغول هستند، کاملاً روشن است. از طرفی از سال‌ها پیش، کمبود کتاب‌های علمی مناسب، چه از نظر محتوی و چه از نظر مواردی همچون نحوهٔ نگارش، ترجمه، و ویرایش در سطح دانشگاه کاملاً محسوس بوده است. انتشارات نگارندهٔ دانش با تکیه بر تجربهٔ مؤسسان خود و شناخت آسیب‌های موجود در صنعت نشر، برای ایفای بخشی از رسالت خود و جبران جزئی از این کمبود، اقدام به چاپ و نشر کتاب‌های دانشگاهی در زمینه‌های فنی مهندسی، علوم پایه، و کامپیوتر به صورت تألیف یا ترجمه کرده است.

در این راستا، این انتشارات همواره کوشیده است تا با بررسی کتاب‌های جدید منتشر شده توسط ناشرین معتبر بین‌المللی، دریافت پیشنهاد چاپ کتاب‌های مورد نظر اساتید محترم دانشگاه، تعامل با مراکز علمی و دانشگاهی، و همکاری با مؤلفین و مترجمین خبره و دارای تحصیلات مرتبط، همواره با نشرکتاب‌های برگزیده و مورد تأیید اساتید دانشگاه، پاسخگوی این نیاز جامعهٔ علمی کشور باشد.

بدیهی است پیشنهادها و انتقادهای سازنده جامعهٔ علمی کشور، ما را در بهبود کار و انجام هرچه بهتر وظیفه‌ای که بر عهده گرفته‌ایم یاری خواهد کرد.

مدیر مسئول
مهندس علی کلانتری
a-kalantari@hotmail.com

به‌نام آنکه هستی نام از او یافت

پیشگفتار مؤلفین

لطف پروردگار یکتا و استقبال بی‌نظیر علاقمندان عرصه کنترل و اتوماسیون، موجب شد تا کتاب حاضر در مدت‌زمانی کوتاه چندین بار به چاپ برسد.

این کتاب تکمیل کننده مطالب کتاب PLC S7 سطح مقدماتی است و مباحث اصلی آن سخت‌افزار S7-400، کار با سیگنال‌های آنالوگ، فانکشن‌نویسی، کار با وقفه‌ها و کنترل PID است. روش برنامه‌نویسی براساس زبان‌های LAD/FBD است. بدیهی است که مطالب کتاب سطح مقدماتی پیش‌نیاز استفاده مفید از محتویات این کتاب است.

کتاب سطح مقدماتی و این کتاب به‌گونه‌ای تدوین شده است که بتواند نیازهای اساسی کاربران PLC را به‌ویژه در کار با PLC زیمنس پوشش دهد. با این وجود کتاب دیگری با عنوان سطح تکمیلی در دست تهیه است که مطالب تکمیلی شامل برنامه‌نویسی به زبان STL و SCL و S7-Graph را همراه کار با SFC/SFBها توضیح داده و کاربران را به‌صورت حرفه‌ای با امکانات PLC S7 آشنا می‌سازد. این کتاب در مرحله نگارش است و امیدواریم بتوانیم در اولین فرصت آن را کامل کرده و به چاپ برسانیم.

برای عرضه مطلوب این اثر تلاش‌های فراوانی از جانب ما و ناشر محترم آقای مهندس علی کلانتری به عمل آمد؛ با این وجود ممکن است نواقص یا اشکالاتی از چشم ما دور مانده باشد. خوشوقت خواهیم شد مواردی از اینگونه یا هر پیشنهادی که برای بهبود و تکمیل مطالب این کتاب وجود دارد را از طریق پست‌های الکترونیکی reza.maher@gmail.com و ahmad_fa62@yahoo.com یا از طریق ناشر با ما در میان بگذارید.

محمدرضا ماهر احمد فرجی

فهرست مطالب

فصل ۱

نصب و پیکربندی سخت‌افزار S7-400

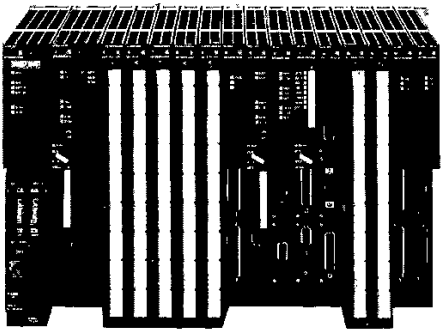
در این فصل با نحوه پیکربندی سخت‌افزار S7-400 آشنا می‌شوید. بسیاری ار نکاتی که در بحث پیکربندی S7-300 در کتاب مقدماتی ذکر شد، پیش نیاز فهم مطالب این بخش هستند.

چکیده مطالب

- رک در S7-400 نه تنها نقش نگهداری ماژول‌ها را دارد، بلکه ارتباط بین آنها را نیز برقرار می‌کند.
- انواع رک به UR و ER و CR تقسیم می‌شود که رک UR از بقیه متداول‌تر است.
- منبع تغذیه PS-400 دارای انواع مختلفی از نظر ولتاژ ورودی و جریان خروجی است. برخی از انواع آنها را می‌توان به‌صورت افزونه به‌کار برد.
- باتری پشتیبان که روی منبع تغذیه نصب می‌شود، برای حفاظت حافظه RAM در صورت قطع تغذیه است؛ بعلاوه امکان راه اندازی Hot را نیز فراهم می‌سازد.
- CPU-400 به سه دسته استاندارد، H و F تقسیم می‌شود که دو نوع آخر در این کتاب تشریح نمی‌شود.
- CPUهای S7-400 متنوع هستند. از نظر ارتباط با شبکه بعضی از مدل‌ها می‌توانند تا چهار پورت ارتباط شبکه را پشتیبانی کنند.
- پورت MPI/DP روی CPU را می‌توان توسط نرم‌افزار برای ارتباط با Profibus-DP تعریف کرد.
- پورت PN روی برخی CPUها برای ارتباط با شبکه Profinet است.
- همه CPUهای S7-400 دارای حافظه داخلی برای Load Memory هستند و بدون کارت حافظه نیز کار می‌کنند.
- در برخی از CPUها حافظه PII و PIQ به Work Memory منتقل شده و اندازه آنرا می‌توان توسط نرم افزار مشخص نمود.
- کارت‌های ورودی و خروجی SM-400 تنوع کمتری نسبت به SM-300 دارند.
- توسط کارت‌های IM می‌توان تا ۲۱ رک اضافی را به رک اصلی متصل نمود.

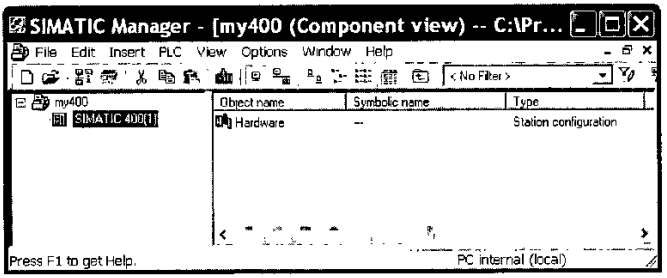
۱-۱ مقدمه

در کتاب مقدماتی با پیکربندی سخت‌افزار S7-300 آشنا شدید، در این فصل با ماژول‌های S7-400 و نحوه‌ی پیکربندی آن آشنا می‌شوید. پیکربندی S7-400 نسبت به S7-300 دارای برخی نکات مشترک و برخی تفاوت‌هاست که در ادامه‌ی مطالب این فصل بیان می‌گردد.



شکل ۱-۱ نمونه سیستم S7-400

به منظور انجام پیکربندی PLC S7-400 توسط نرم‌افزار Step7، لازم است مانند شکل ۱-۲ در محیط نرم‌افزار Simatic Manager یک ایستگاه کاری S7-400 را با کلیک راست موس یا با استفاده از منوی Insert به روشی که در کتاب مقدماتی ذکر شد، وارد کنید.



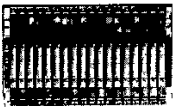
شکل ۱-۲ شروع پیکربندی S7-400 در Simatic Manager

با دابل‌کلیک بر روی ایستگاه ایجاد شده و دابل‌کلیک روی گزینه‌ی Hardware وارد محیط نرم‌افزار HW Config شوید. در ادامه ضمن توضیح اجزای S7-400 به‌ترتیب نحوه پیکربندی آنها در HWconfig ذکر می‌شود.

۱-۲ اجزای اصلی S7-400

در جدول ۱-۱ اجزای اصلی S7-400 را مشاهده می‌نمایید. اکثر این ماژول‌ها در S7-300 نیز وجود داشتند. باید توجه نمود که به لحاظ ساختار و عملکرد، برخی از این اجزا نسبت به اجزای مشابه در S7-300 دچار تغییراتی شده‌اند.

جدول ۱-۱ اجزای اصلی S7-400

عنوان	وظیفه	تصویر
Racks (UR: Universal Rack) (CR: Central Rack) (ER: Expansion Rack)	نگهداری ماژول‌ها و برقراری اتصال الکتریکی بین ماژول ها	
Power Supply Modules (PS = Power Supply) Accessories: Backup battery	تامین ولتاژ ۲۴ ولت و ۵ ولت مورد نیاز PLC	
CPUs Central Processing Units (CPUs)	اجرای برنامه کاربر، ارتباط با سایر CPUها از طریق شبکه-های صنعتی و ...	
Memory cards	ذخیره برنامه کاربر این کارت در اسلات CPU قرار می‌گیرد.	
IF 964-DP interface module	اتصال I/Oها به PLC از طریق شبکه پروفی باس، این کارت روی CPU نصب می‌شود.	

Signal Modules (SM)	دریافت سیگنال‌های پروسه و تبدیل آنها به دیتا (کارت ورودی)، تبدیل دیتا به سیگنال الکتریکی و ارسال آن به پروسه (کارت خروجی)	
Interface modules (IM)	ارتباط بین رک اصلی و رک های توسعه	
Function Module (FM)	برای انجام فانکشن‌های خاصی مانند: PID کنترل، کنترل موقعیت، شمارش سریع و ...	

۱-۳ تفاوت‌های مهم بین پیکربندی S7-400 و S7-300

برای پیکربندی S7-400 از نرم‌افزار STEP 7 استفاده می‌گردد. برخی از تفاوت‌های پیکربندی S7-400 و S7-300 به شرح زیر می‌باشد:

- در S7-300 تنها یک نوع رک وجود دارد، اما در S7-400 چندین نمونه رک وجود دارد.
- در S7-300 وظیفه‌ی رک فقط نگهداری ماژول‌ها می‌باشد، اما رک در S7-400 علاوه بر نگهداری ماژول‌ها، ارتباط بین آنها را به منظور تبادل دیتا نیز برقرار می‌سازد. در واقع آنچه در S7-300 به‌کار می‌رود یک ریل ساده است ولی در S7-400 رک به معنای واقعی خود استفاده می‌شود.
- برای ارتباط بین رک اصلی و رک های اضافی در هر دو سیستم از کارت IM استفاده می‌شود. در S7-300 می‌توان حداکثر ۳ رک توسعه قرار داد، در S7-400 می‌توان حداکثر ۲۱ رک توسعه قرار داد.

- در S7-300 فقط یک IM می‌توان در رک اصلی قرار داد، در S7-400 حداکثر می‌توان ۶ عدد IM در رک اصلی قرار داد.
- در S7-300 محل قرارگیری IM در اسلات ۳ می‌باشد، در S7-400 در رک اصلی IM در اسلات دلخواه (بعد از PS و CPU) قرار گرفته و در رک توسعه IM در اسلات آخر قرار می‌گیرد.
- در S7-300، وجود فضای خالی در بین ماژول‌های روی رک مجاز نمی‌باشد ولی در S7-400 وجود فضای خالی بین ماژول‌ها مشکلی ایجاد نمی‌نماید.
- ترتیب چیدمان ماژول‌ها روی رک در S7-300 مهم است ولی در S7-400 بعد از PS هر ماژول دلخواهی را می‌توان قرار داد. حتی می‌توان کارت‌های I/O را قبل از CPU نصب کرد.
- کارت‌های DI/DO و AI/AO که کارت‌های ترکیبی هستند و هم ورودی و هم خروجی به آنها متصل می‌گردد، در S7-400 وجود ندارد، به‌طور کلی تنوع کارت‌های S7-400 نسبت به S7-300 کمتر است.
- در S7-400 قابلیتی به‌نام Multicomputing وجود دارد که اجازه می‌دهد چند CPU (حداکثر چهار CPU) در کنار هم و روی یک رک قرار گرفته و برنامه‌های مختلفی را پردازش نمایند.
- توانایی‌های CPU از قبیل سرعت پردازش، حافظه و ... در S7-400 نسبت به S7-300 بیشتر است.

۱-۴ رک S7-400

با توجه به تنوع زیاد رک در S7-400، یکی از مهم‌ترین بخش‌ها در انجام پیکربندی S7-400 انتخاب صحیح رک می‌باشد. در ادامه‌ی بحث، ابتدا ساختار داخلی رک S7-400 بررسی و سپس انواع آن بیان می‌گردد.

۱-۴-۱ ساختار داخلی رک S7-400

رک S7-400 به‌طور کلی از دو بخش تشکیل شده‌است. این دو بخش عبارتند از:

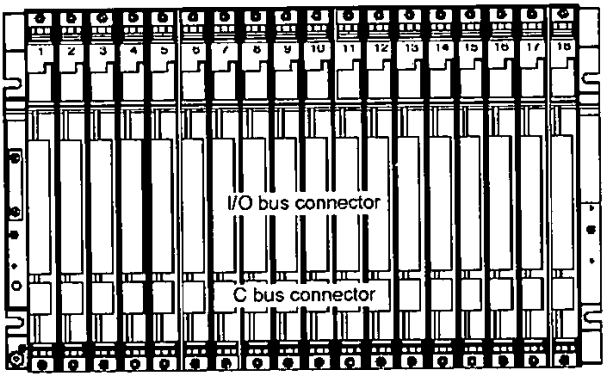
۱- I/O Bus (P Bus)

I/O Bus یک Backplane bus پارالل می‌باشد که برای تبادل سریع دیتا بین CPU و ماژول‌های ورودی/خروجی طراحی شده است (با سرعت تقریبی 5 Mbps). همه‌ی رک‌های S7-400 دارای I/O BUS می‌باشند.

۲- Communication Bus (C Bus)

C Bus که گاهی K-Bus نیز گفته می‌شود، یک bus سریال است و برای تبادل سریع دیتاهای زیاد به‌کار می‌رود (با سرعت 10.5 Mbps). وظیفه C Bus برقراری ارتباط بین CPU با FM و CP می‌باشد. رک‌های توسعه از نوع ER این قسمت را ندارند، از اینرو در آنها فقط می‌توان ماژول‌های ورودی و خروجی قرار داد. البته اگر از FMی استفاده شود که یک اسلات اشغال کند، می‌توان آنرا نیز در رک از نوع ER قرار داد.

در شکل ۱-۳ نمای شماتیک رک S7-400 نشان داده شده است.



شکل ۱-۳ ساختار رک S7-400

۱-۴-۲ انواع رک‌های S7-400

ساختار نشان داده شده در شکل ۱-۳ به‌صورت کلی است ولی عملاً همه رک‌ها این ساختار را ندارند و از اینرو به سه دسته اصلی تقسیم می‌شوند که عبارتند از:

- CR یا Central Rack که فقط می‌تواند به‌عنوان رک اصلی به‌کار رود.
- UR یا Universal Rack که می‌تواند به‌عنوان رک اصلی یا رک توسعه به‌کار رود.
- ER یا Expansion Rack که فقط می‌تواند به‌عنوان رک توسعه استفاده شود.

تفاوت بین رک اصلی و رک توسعه در کتاب مقدماتی تشریح شد. در اینجا همین اندازه اشاره می‌شود که در رک اصلی، CPU همراه با سایر کارت‌ها قرار می‌گیرند ولی در رک اضافی نمی‌توان CPU نصب کرد و معمولاً برای گسترش سیستم و نصب کارت‌های I/O اضافی به‌کار می‌رود. رک توسعه از طریق ماژول IM به رک اصلی متصل می‌گردد.
در شکل ۱-۴ انواع رک و ساختار آنها نشان داده شده است.

نوع رک و ساختار باس در آن	قابل استفاده به عنوان رک اصلی	قابل استفاده به عنوان رک توسعه
UR1 / UR2 (Universal Rack) — P bus / K bus	Yes	Yes
CR2 (Central Rack) — P bus, Segment 1 / P bus, Segment 2 / K bus	Yes	No
ER1 / ER2 (Expansion Rack) — P bus	No	Yes

شکل ۱-۴ انواع رک و ساختار

در جدول ۱-۲ انواع رک S7-400 و برخی از مشخصات آنها نشان داده شده است. توضیحات هر یک از آنها در ادامه خواهد آمد.

جدول ۱-۲ انواع رک S7-400

عنوان	وظیفه	کاربرد	تعداد اسلات	توضیحات
UR1	UNIVERSAL	به‌عنوان رک اصلی به‌عنوان رک توسعه	18	دارای دو نوع است که در یک نوع آن می‌توان دو منبع تغذیه از نوع REDUNDANT قرار داد.
UR2	UNIVERSAL	" "	9	" "
UR2H	UNIVERSAL	برای PLCهای REDUNDANT	2*9	شبیه دو رک اصلی UR2
CR2	SEGMENTED	به‌عنوان رک اصلی	10+8	دارای دو بخش است که هر بخش از آن می‌تواند یک رک اصلی مستقل باشد.
CR3	CENTRAL RACK	به‌عنوان رک اصلی	4	
ER1	EXPANSION	رک توسعه	18	دارای دو نوع است که در یک نوع آن می‌توان دو منبع تغذیه از نوع REDUNDANT قرار داد.
ER2	EXPANSION	رک توسعه	9	"

با این توضیح به معرفی انواع رک می‌پردازیم.

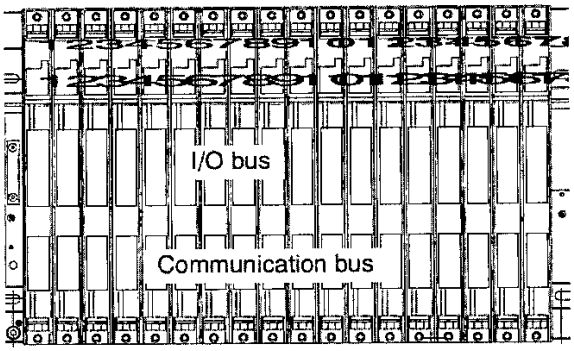
الف) رک UR

رک UR متداول‌ترین رک در سیستم‌های S7-400 است. این رک به سه نوع تقسیم می‌گردد:

- رک UR1
- رک UR2
- رک UR2H

رک UR1

این رک دارای ۱۸ اسلات است و می‌تواند به‌عنوان رک اصلی یا توسعه مورد استفاده قرار گیرد. در این رک هردو قسمت I/O Bus و C Bus به‌صورت پیوسته می‌باشد.

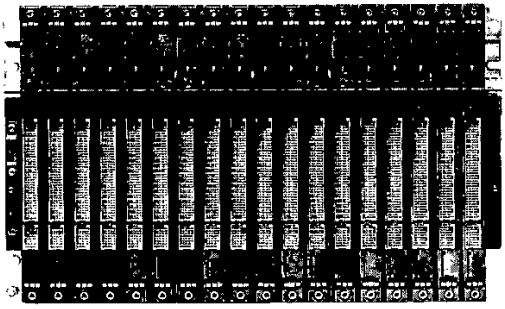


شکل۱-۵ نمای شماتیک رک از نوع UR1

رک UR1 به دو نوع تقسیم می‌شود. در نوع اول در مشخصات فنی آن ذکر شده که:

not suitable for redundant power supply

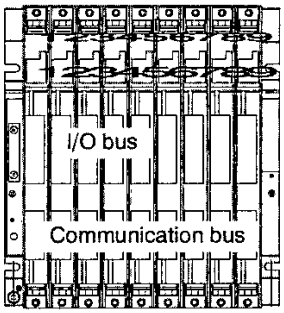
یعنی در این رک فقط یک منبع تغذیه می‌توان قرار داد. در نوع دوم که عبارت فوق برای آن ذکر نشده امکان نصب دو منبع تغذیه روی رک در کنار هم وجود دارد. این دو منبع تغذیه همزمان تغذیه باس را انجام می‌دهند و در صورت بروز مشکل روی یک تغذیه، PLC بی‌برق نخواهد شد. سخت افزار لازم برای جلوگیری از جریان گردشی بین آنها روی رک تعبیه شده است.

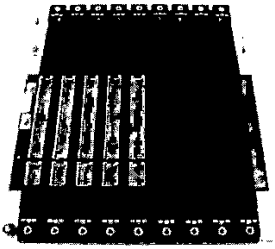


شکل ۱-۶ شکل واقعی رک UR1

رک UR2

دارای ۹ اسلات است و می‌تواند به‌عنوان رک اصلی یا توسعه مورد استفاده قرار گیرد. مشابه UR1 هر دو قسمت I/O Bus و C Bus در آن به‌صورت پیوسته می‌باشد.



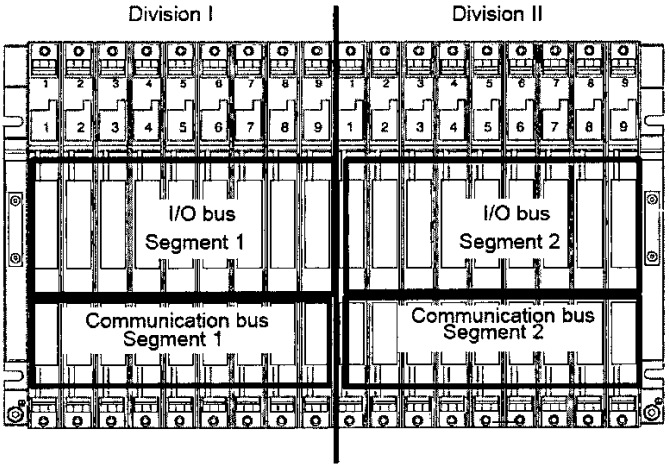


شکل ۱-۷ نمای واقعی و شماتیک رک از نوع UR2

رک UR2 نیز از نظر تغذیه به دو نوع تقسیم می‌شود؛ در یک نوع فقط یک منبع تغذیه می‌تواند به‌کار رود، ولی در نوع دوم می‌توان از دو منبع تغذیه استفاده کرد.

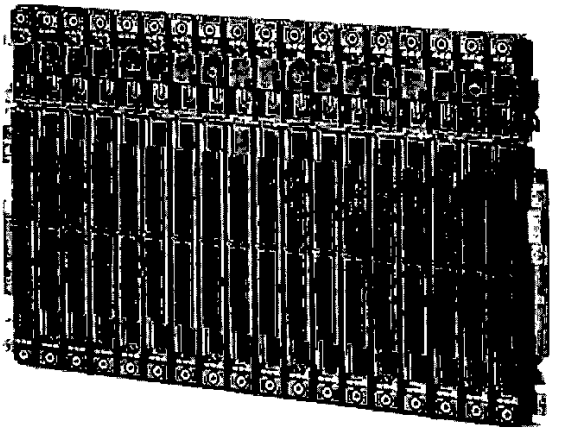
رک UR2-H

این نوع رک به دو بخش کاملاً مجزا از یکدیگر تقسیم شده است که هر کدام از آنها دارای I/O Bus و C Bus مربوط به خودشان می‌باشند. هر بخش از رک دارای ۹ اسلات می‌باشد. این رک در واقع مانند دو رک از نوع UR2 می‌باشد که به لحاظ الکتریکی کاملاً از یکدیگر مجزا شده‌اند.از این رک جهت پیکربندی سیستم‌های افزونه (S7-400 H) استفاده می‌شود.



شکل ۱-۸ رک S7-400 از نوع UR2-H

این رک از نظر ظاهر مشابه UR1 است ولی با کمی دقت تفاوت آنها روشن می‌شود. در رک UR1 شماره نوشته شده روی اسلات‌ها از 1 شروع شده و به 18 ختم می‌شود ولی در رک UR2H شماره از 1 شروع شده و پس از رسیدن به 9 مجدداً از 1 شروع می‌گردد.

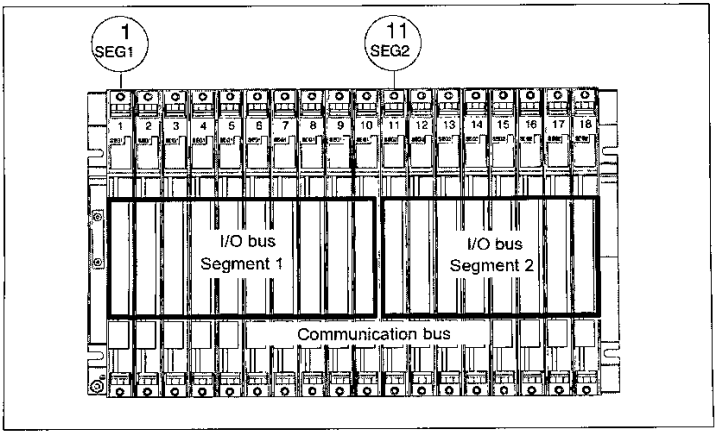


شکل ۱-۹ شکل واقعی رک UR2H

رک CR2

این رک دارای ۱۸ اسلات می‌باشد که در آن C Bus به‌صورت پیوسته است ولی I/O Bus به دو قسمت ۱۰ و ۸ اسلاته تقسیم شده است. در هر کدام از بخش‌های ۱۰ و ۸ اسلاته‌ی رک می‌توان یک CPU قرار داد که هرکدام برنامه‌ی مستقلی را اجرا نمایند. CPUها از طریق I/O Bus مربوطه با SMهای خود ارتباط داشته و از طریق C Bus مشترک می‌توانند با CP یا FM مربوط به خود نیز ارتباط برقرار نمایند. کاربرد این رک به‌عنوان رک اصلی می‌باشد.

در این نوع رک، هر دو قسمت رک از طریق یک PS تغذیه می‌شوند.



شکل۱-۱۰ رک از نوع CR2

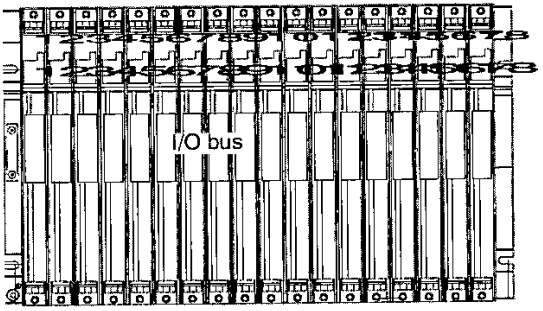
رک CR3

این رک دارای ۴ اسلات است و هر دو قسمت I/O BUS و C BUS را دارا می‌باشد. کاربرد آن به‌عنوان رک اصلی می‌باشد.

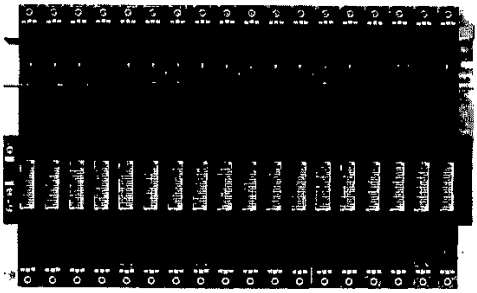
شکل ۱-۱۱ شکل واقعی رک CR3

رک ER1

از این رک فقط به‌عنوان رک توسعه می‌توان استفاده نمود. این رک دارای ۱۸ اسلات است و در آن فقط I/O Bus وجود دارد و فاقد C-Bus است. از اینرو در این نمونه رک نمی‌توان CP و FM قرار داد. البته در این رک برخی از نمونه‌های CP مربوط به اتصال Point to Point را می‌توان قرار داد ولی امکان قرار دادن CP مربوط به شبکه‌های پروفی‌باس و اترنت وجود ندارد. از FMها نیز آنهایی که تنها یک اسلات اشغال کنند را می‌توان روی این نمونه از رک قرار داد.



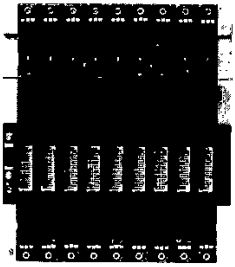
شکل ۱-۱۲ نمای شماتیک رک ER1



شکل ۱-۱۳ شکل واقعی رک ER1

رک ER2

خصوصیات این رک مشابه رک ER1 می‌باشد با این تفاوت‌که این نوع رک دارای ۹ اسلات است.



شکل ۱-۱۴ شکل واقعی رک ER2

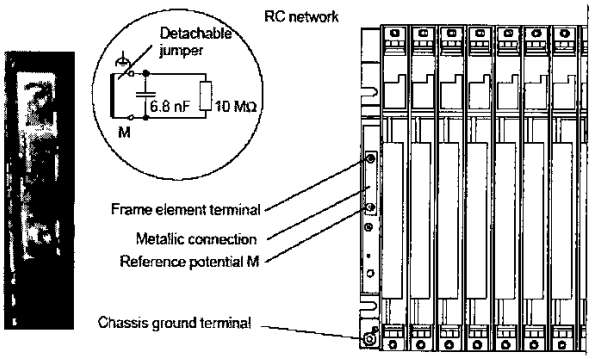
۱-۴-۳ نصب و اتصال زمین رک S7-400

در هنگام نصب، به نکات زیر توجه کنید:

- رک را به‌صورت افقی نصب کنید.
- بین رک و داکت‌ها حداقل ۵ سانتی‌متر فضا باشد (به‌منظور خنک‌کنندگی)
- بین دو رک حداقل ۱۰ سانتی‌متر فضا باشد.
- کاورهای روی باس‌های رک را وقتی استفاده نمی‌شوند باز نکنید (جلوگیری از نفوذ گرد و غبار)
- زمین‌کردن رک به‌طور صحیح انجام شود.

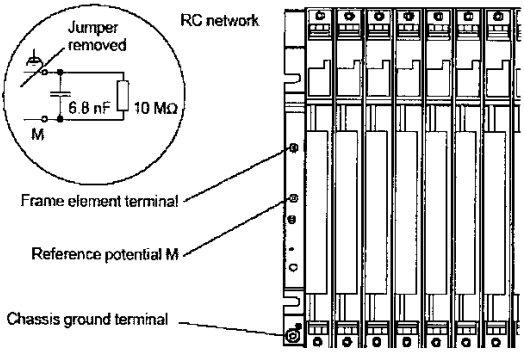
مشابه موارد بیان‌شده در مورد اتصال کابل زمین به رک در S7-300، رک S7-400 را نیز می‌توان به زمین متصل نمود. برای این‌منظور در قسمت پایین سمت چپ رک محل اتصال به زمین تعبیه شده است.

ماژول‌هایی که روی رک نصب می‌شوند از همین نقطه از طریق Rack با زمین ارتباط می‌یابند. اتصال به زمین مانند شکل ۱-۱۵ است. همانطور که در این شکل مشاهده می‌شود، به‌صورت پیش‌فرض یک Jumper بین M و زمین وجود دارد.



شکل۱-۱۵ اتصال زمین رک به‌طور مستقیم

پایین سمت چپ رک محل اتصال به زمین است. جامپر در قسمت بالا به‌صورت یک اتصال فلزی وجود دارد که پایه M را به زمین وصل کرده است. در برخی کاربردها مانند نیروگاهها که نیاز به آشکارسازی خطای اتصال زمین است یا امکان عبور جریان‌های زیاد وجود دارد، می‌توان نقطه M را از زمین جدا کرد. در این‌حالت حفاظت توسط یک مدار RC انجام می‌شود.



شکل ۱-۱۶ اتصال زمین رک از طریق مدار RC

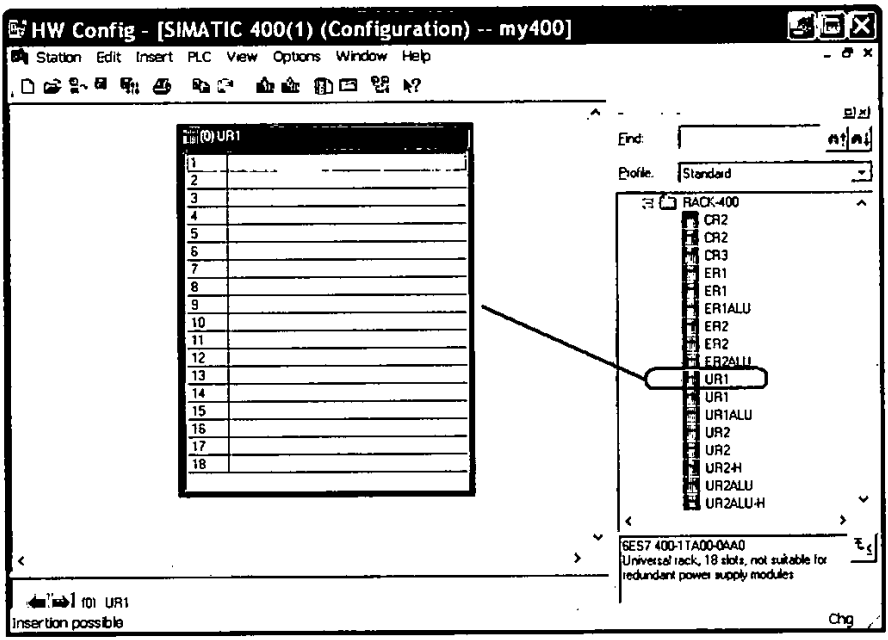
۱-۴-۴ پیکر بندی رک در Hwconfig

اولین قدم در پیکربندی سخت‌افزاری انتخاب نوع رک می‌باشد. با توجه به توضیحات ارائه شده می‌توان هر کدام از رک‌های موجود در کاتالوگ را از مسیر Simatic 400 - RACK-400 انتخاب نموده و آنرا در محیط برنامه قرار داد.

در شکل ۱-۱۷ نحوه‌ی وارد کردن رک از کاتالوگ موجود در برنامه‌ی HW Config نشان داده شده است.

همانطور که در این شکل مشاهده می‌شود و پیش از این نیز ذکر شد، رک S7-400 دارای تنوع زیادی می‌باشد که بسته به نیاز می‌توان یک رک اصلی و حداکثر ۲۱ رک توسعه انتخاب نمود.

تذکر: رک‌هایی که دارای علامت ALU می‌باشند از نوع آلومینیومی هستند.



شکل ۱-۱۷ نحوه انتخاب رک از کاتالوگ و وارد کردن آن به محیط Hwconfig

۱-۵ منبع تغذیه (PS-400)

۱-۵-۱ عملکرد

در S7-400 منابع تغذیه تنوع بیشتری نسبت به منابع موجود در S7-300 دارند. بعلاوه از نظر عملکرد با آنها تفاوت‌هایی دارند. این تفاوت‌ها عبارتند از:

- منابع تغذیه S7-300 همگی دارای ورودی 110/220 V AC هستند ولی منابع تغذیه S7-400 دو نوع دارند یک نوع مشابه S7-300 دارای ورودی AC ولی در نوع دیگر ورودی 24V DC است.
- در S7-400 منبع تغذیه خروجی 5VDC را برای تغذیه‌ی باس تولید می‌کند ولی در S7-300 ولتاژ 5 VDC توسط CPU ایجاد می‌شود.
- در S7-400 استفاده از منبع تغذیه روی رک الزامی است و در پیکر بندی نیز بایستی تعریف شود ولی در S7-300 این کار الزامی نیست و می‌توان از منبع تغذیه دلخواه بیرونی استفاده نمود.
- در S7-400 منبع تغذیه با CPU تبادل دیتا دارد و در صورت بروز مشکل روی تغذیه یا باتری یک سیگنال خطا به CPU گزارش می‌دهد. این امکان در S7-300 وجود ندارد.

- برای منابع تغذیه S7-400 محلی برای نصب باتری تعبیه شده است ولی در S7-300 باتری وجود ندارد. در نوع S7-300 قدیمی باتری روی خود CPU نصب می‌گردد.

اسلات‌های مجاز جهت قرارگیری PS

اسلات‌های مجاز برای قرار دادن منبع تغذیه اسلات ۱ الی ۴ است، بدین‌نحو که منبع تغذیه در اسلات ۱ قرار می‌گیرد. بعضی از مدل‌های منبع تغذیه ممکن است بیش از یک اسلات را اشغال نمایند که در اینصورت می‌توانند حداکثر تا اسلات ۴ را اشغال نمایند. در هر صورت اسلات شروع برای نصب منبع تغذیه اسلات اول می‌باشد.

اگر منبع تغذیه در اسلات غیر مجاز در رک قرار گیرد (از اسلات ۴ به بعد) روشن نخواهد شد. برای رفع این مشکل باید پس از خاموش نمودن منبع تغذیه آنرا از اسلات غیر مجاز در آورده و در اسلات شماره ۱ قرار داده و بعد از ۱ دقیقه آنرا روشن نمود. باید توجه شود که قرار دادن منبع تغذیه در اسلات غیر مجاز ممکن است سبب آسیب رسیدن به خود منبع تغذیه گردد.

منبع تغذیه Redundant در S7-400

در S7-400 می‌توان از دو منبع تغذیه به‌صورت Redundant استفاده نمود. برای این کار باید یک منبع تغذیه Redundant در اسلات ۱ و یک منبع تغذیه Redundant دیگر در اسلات ۳ قرار داد(منابع تغذیه افزونه فقط نوع 10A دارند که هر کدام دو اسلات را اشغال می‌کنند). باید دقت نمود ماژول‌هایی که روی رک نصب می‌شوند حداکثر به اندازه جریان یکی از منابع تغذیه، مجاز به مصرف جریان هستند که این مقدار در منابع تغذیه افزونه 10 A است.

برخی از خصوصیات منابع تغذیه‌ی افزونه عبارتست از:

- هرکدام از منابع تغذیه در صورتی‌که برای منبع دیگر اشکالی بوجود آید قادر به تغذیه همه‌ی رک می‌باشند.
- در حین کار PLC، هر کدام از منابع تغذیه را می‌توان تعویض نمود بدون آنکه خللی در کار سیستم بوجود آید.
- هر کدام از منابع تغذیه وظایف خود را نظارت نموده و در صورتی‌که اشکالی مشاهده نمایند آنرا به CPU گزارش می‌دهند.
- خطا روی هر یک از منابع تغذیه بر روی کار منبع تغذیه دیگر اثر نمی‌گذارد.
- با استفاده از دو باتری پشتیبان[1] بر روی هر کدام از منابع تغذیه‌ی افزونه، می‌توان حالت افزونگی[2] را در پشتیبانی از اطلاعات نیز بوجود آورد.

باتری پشتیبان

منابع تغذیه S7-400 دارای مکانی جهت قرار دادن یک یا دو باتری پشتیبان می‌باشند. استفاده از باتری پشتیبان اختیاری است. وظایف باتری عبارتست از:

۱- حفاظت حافظه RAM در صورت قطع تغذیه

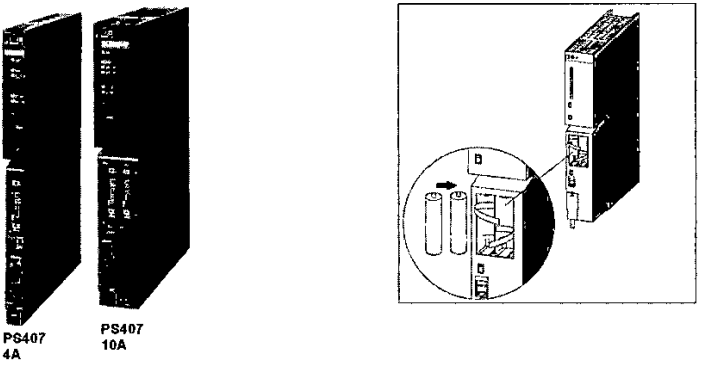
۲- امکان راه‌اندازی Hot

1. backup battery
2. redundancy

۳- کوتاه شدن زمان Self Test در سیستم‌های S7-400H

با استفاده از کارت حافظه Flash مورد اول یعنی حفاظت برنامه نیازی به باتری ندارد ولی برای موارد ۲ و ۳ وجود باتری الزامی است.

در صورتی‌که از دو باتری پشتیبان استفاده شود، اطمینان بیشتری وجود دارد که حتی اگر یک باتری ضعیف شود باتری سالم اطلاعات RAM را پشتیبانی می‌کند. در شکل ۱-۱۸ دو نمونه منبع تغذیه به همراه باتری پشتیبان نشان داده شده است.



شکل ۱-۱۸ منبع تغذیه S7-400 همراه با باتری پشتیبان

در جدول ۱-۳ مشخصات فنی باتری پشتیبان نشان داده شده است.

جدول ۱-۳ مشخصات فنی باتری پشتیبان

Backup Battery	
Order number	6ES7971-0BA00
Type	1 x lithium AA
Rated voltage	3.6 V
Rated capacity	2.3 Ah

با توجه به آمپر ساعت این باتری‌ها و جریان اندکی که برای حفاظت RAM لازم است، باتری‌ها ممکن است تا چندین ماه نیز بتوانند در صورت قطع تغذیه، RAM را پشتیبانی کنند. به‌طور معمول اگر ولتاژ باتری کمتر از ۳ ولت شود چراغ فالت باتری روشن شده و نیاز به تعویض دارد.

۱-۵-۲ انواع منبع تغذیه S7-400

به‌طور کلی منابع تغذیه‌ی S7-400 را می‌توان از دو جنبه مورد مقایسه قرار داد:

- از نظر ولتاژ ورودی که می‌تواند AC یا DC باشد.

- از نظر استاندارد بودن یا قابلیت افزونگی

ولتاژ خروجی همه منابع تغذیه 24 V DC و 5 V DC می‌باشد. جدول ۱-۴ مقایسه منبع تغذیه‌ی استاندارد و افزونه S7-400 را نشان می‌دهد.

جدول ۱-۴ انواع منبع تغذیه‌ی S7-400

نوع	ولتاژ خروجی	جریان
STANDARD	24V DC 5V DC	4A 10A 20A
REDUNDANT	24V DC 5V DC	10A

منابع تغذیه‌ی مختلف S7-400

در جدول ۱-۵ کلیه‌ی منابع تغذیه‌ی S7-400 به‌همراه خصوصیات آنها نشان داده شده است.

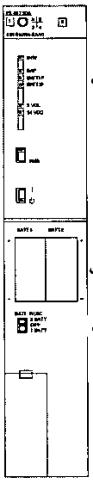
جدول ۱-۵ منابع تغذیه‌ی S7-400

منبع تغذیه	نوع منبع تغذیه	ولتاژ ورودی AC	ولتاژ ورودی DC	ولتاژ و جریان خروجی
PS 407 4A	استاندارد	۸۵ الی ۲۶۴ ولت	۸۸ الی ۳۰۰ ولت	5 VDC/4 A 24 VDC/0.5 A
PS 407 10A	استاندارد	۸۵ الی ۲۶۴ ولت	۸۸ الی ۳۰۰ ولت	5 VDC/10 A 24 VDC/1 A
PS 407 10A	افزونه	۸۵ الی ۲۶۴ ولت	۸۸ الی ۳۰۰ ولت	5 VDC/10 A 24 VDC/1 A
PS 407 20	استاندارد	۸۵ الی ۲۶۴ ولت	۸۸ الی ۳۰۰ ولت	5 VDC/20 A 24 VDC/1A
PS 405 4A	استاندارد	----------	۱۹,۲ الی ۷۲ ولت	5 VDC/4 A 24 VDC/0.5 A
PS 405 10A	استاندارد	----------	۱۹,۲ الی ۷۲ ولت	5 VDC/10 A 24 VDC/1 A
PS 405 10A	افزونه	----------	۱۹,۲ الی ۷۲ ولت	5 VDC/10 A 24 VDC/1 A
PS 405 20A	استاندارد	----------	۱۹,۲ الی ۷۲ ولت	5 VDC/20 A 24 VDC/1 A

۱-۵-۳ لامپ‌ها، کلیدها و علائم نشان‌دهنده PS

همانطور که در شکل ۱-۱۹ نشان داده شده است، روی منابع تغذیه‌ی S7-400 لامپ‌ها و علائم و کلیدهای مختلفی به‌کار رفته شده است.

به‌عنوان نمونه در این شکل PS 407 20A نشان داده شده است. سایر منابع تغذیه نیز تقریباً چنین شکلی داشته و اکثر علائم و کلیدهای به‌کار رفته شده در آنها مانند این منبع تغذیه می‌باشد.



شکل ۱-۱۹ PS 407 20A

تفاوت لامپ‌ها و علائم منابع تغذیه‌ی مختلف با یکدیگر

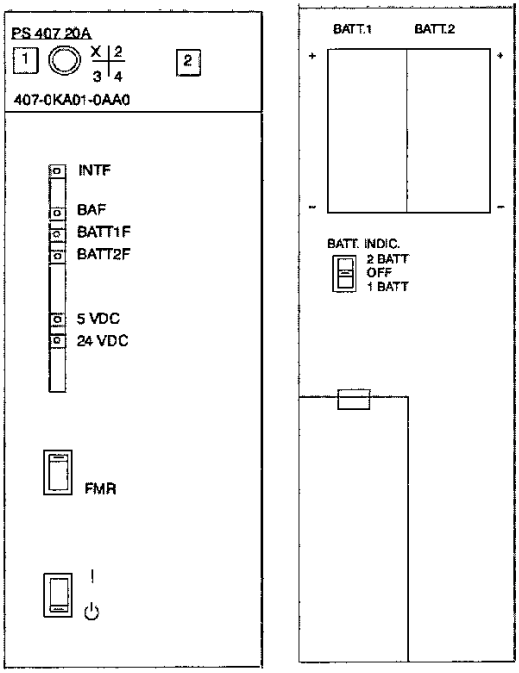
همه‌ی منابع تغذیه S7-400 دارای کلیدها و نمایشگرهای یکسانی هستند. تفاوت‌های عمده آنها به شرح زیر می‌باشد:

- منابع تغذیه که باتری پشتیبان قبول می‌کنند، دارای یک لامپ (BATTF) می‌باشند که خالی‌شدن یا غیرفعال بودن یا عدم وجود باتری پشتیبان را نشان می‌دهد.
- منابع تغذیه‌ای که دو باتری پشتیبان را قبول می‌کنند دارای دو لامپ (BATT1F , BATT2F) می‌باشند.

تذکر: در صورت بروز فالت روی یک باتری یا هر دو باتری چراغ BAF نیز روشن خواهد شد.

لامپ‌های نشان‌دهنده

همانطور که بیان شد و در شکل ۱-۲۰ مشاهده می‌گردد، روی PS تعدادی کلید، لامپ و علائم نشان‌دهنده وجود دارد که شرح آنها به‌صورت زیر است:



شکل ۱-۲۰ منبع تغذیه PS 407 20 A

الف) لامپ‌های مشترک

این لامپ‌ها در همه‌ی منابع تغذیه وجود دارند. در جدول ۱-۶ این لامپ‌ها توضیح داده شده‌اند.

جدول ۱-۶ مفاهیم لامپ‌ها

لامپ	رنگ	مفهوم
INTF	قرمز	وجود خطای داخلی
5 VDC	سبز	وجود ولتاژ خروجی 5V DC
24 VDC	سبز	وجود ولتاژ خروجی 24V DC

ب) لامپ‌های منابع تغذیه‌ی دارای باتری پشتیبان

منابع تغذیه‌ای که دارای باتری پشتیبان هستند، دارای چند لامپ نشان دهنده دیگر به شرح جدول۱-۷ می‌باشند.

جدول ۱-۷ مفاهیم لامپ‌های مربوط به باتری پشتیبان

مفهوم	رنگ	لامپ
ولتاژ باتری روی backplane bus خیلی ضعیف شده است.	قرمز	BAF
باتری کاملاً خالی شده است یا پلاریته باتری برعکس می‌باشد یا باتری وجود ندارد.	زرد	BATTF

ج) لامپ‌های منابع تغذیه‌ی دارای دو باتری پشتیبان

منابع تغذیه‌ای که دارای دو باتری پشتیبان باشند، دارای لامپ‌هایی هستند که شرح آنها در جدول ۱-۸ ارائه شده است.

جدول ۱-۸ لامپ‌های منابع تغذیه‌ی دارای دو باتری پشتیبان

مفهوم	رنگ	لامپ
ولتاژ باتری روی backplane bus خیلی ضعیف شده است.	قرمز	BAF
باتری ۱ کاملاً خالی شده است یا پلاریته باتری برعکس می‌باشد یا باتری وجود ندارد.	زرد	BATT1F
باتری ۲ کاملاً خالی شده است یا پلاریته باتری برعکس می‌باشد یا باتری وجود ندارد.	زرد	BATT2F

کلیدهای روی منبع تغذیه

روی منابع تغذیه‌ی S7-400 کلیدهایی قرار گرفته‌اند که شرح آنها در جدول ۱-۹ ارائه شده است.

جدول ۱-۹ شرح کلیدهای موجود در منبع تغذیه‌ی S7-400

نام کلید	عملکرد
pushbutton FMR	برای acknowledge (تأیید) نمودن خطا و ریست نمودن LEDها (لامپ‌های نشان دهنده) پس از برطرف شدن خطا به‌عنوان مثال وقتی با ضعیف شدن باتری چراغ فالت روشن شود، پس از تعویض باتری این چراغ خاموش نمی‌شود و نیاز است که با فشردن FMR برطرف شدن فالت را Acknowledge کنیم.

Standby switch	قطع و وصل نمودن ولتاژ 24 V DC یا 5 V DC خروجی دارای دو حالت است: ولتاژ ▮ خروجی منبع تغذیه برقرار است. ولتاژ ⏻ خروجی منبع تغذیه قطع است.
BATT.INDIC. switch	از این سوئیچ برای تنظیمات مربوط به وضعیت باتری‌ها استفاده می‌گردد. در منابع تغذیه‌ای که یک باتری پشتیبان دارند: Off: یعنی باتری غیر فعال است. BATT: یعنی باتری فعال می‌باشد. در منابع تغذیه‌ای که دو باتری پشتیبان دارند: Off: یعنی باتری غیر فعال است. 1BATT: فقط یک باتری فعال است. 2 BATT: هر دو باتری فعال هستند.

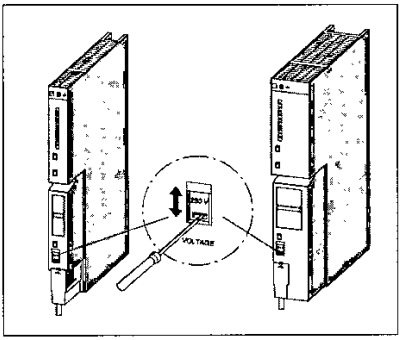
۱-۵-۴ نصب و اتصالات منبع تغذیه

در منابع تغذیه‌ای که ولتاژ ورودی AC قبول می‌نمایند، باید نوع ولتاژ ورودی را روی یکی از دو حالت زیر تنظیم نمود:

الف) 120 V AC

ب) 230 V AC

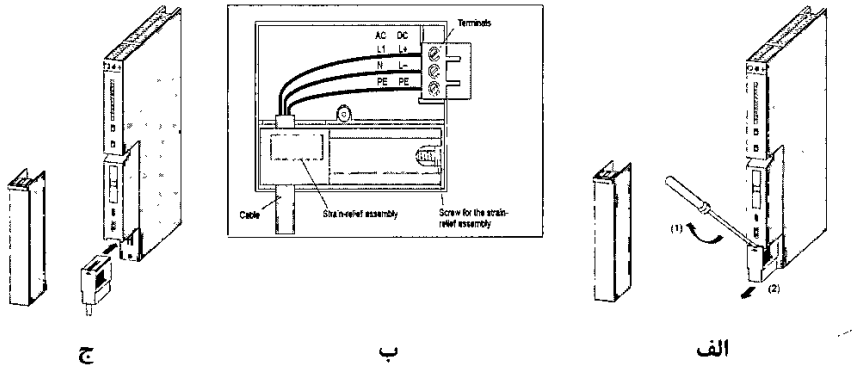
در شکل ۱-۲۱ چگونگی انجام این تنظیم نشان داده شده است.



شکل ۱-۲۱ تنظیم ولتاژ منبع تغذیه

اتصالات منبع تغذیه

برای انجام اتصالات مربوط به منبع تغذیه مطابق شکل ۱-۲۲ می‌توان به روش زیر عمل نمود:



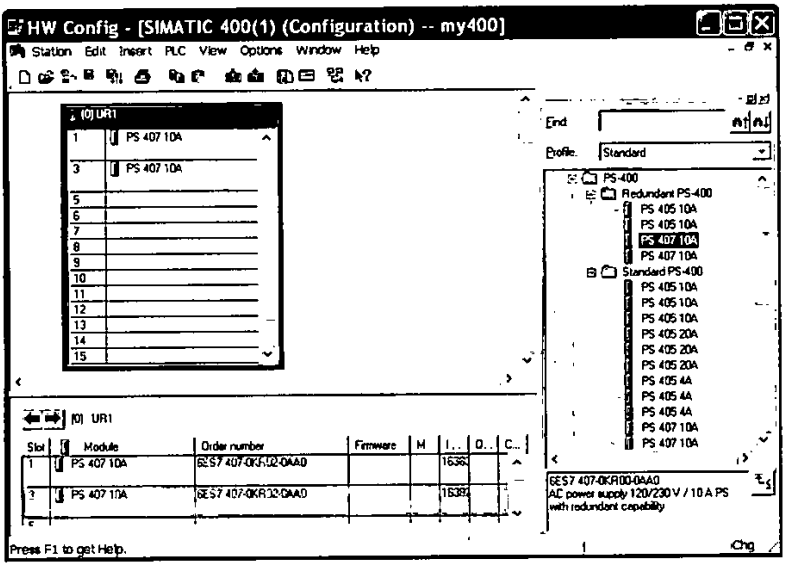
شکل ۱-۲۲ اتصال تغذیه ورودی به منبع تغذیه

۱- درپوش مخصوص منبع تغذیه را خارج نمایید.
۲- کانکتور مخصوص اتصال ولتاژ ورودی را خارج نمایید.
۳- مطابق شکل ۱-۲۲ اتصالات را انجام دهید.
۴- مطابق شکل ۱-۲۲ کانکتور مخصوص را در جای خود قرار دهید.
۵- در پوش مخصوص را در جای خود قرار دهید.

۱-۵-۵ پیکربندی PS در Hwconfig

جهت وارد نمودن PS در محیط برنامه‌ی HW Config و در قسمت کاتالوگ‌ها می‌توان مطابق شکل ۱-۲۳، PS مورد نظر را انتخاب نموده و آنرا در اسلات شماره‌ی ۱ قرار داد. همانطور که بیان شد و در این شکل ملاحظه می‌شود، می‌توان هر کدام از منابع تغذیه‌ی استاندارد یا افزونه را انتخاب نمود. منبع تغذیه انتخاب شده از نوع افزونه هستند.

تذکر: در نوع افزونه به منبع تغذیه یک آدرس Diagnostic داده می‌شود که در ستون I Address ظاهر می‌گردد. برای نوع استاندارد این آدرس وجود ندارد.



شکل ۱-۲۳ وارد کردن منبع تغذیه در Hwconfig

CPU-400 ۶-۱

۱-۶-۱ عملکرد و تفاوت‌ها نسبت به CPU-300

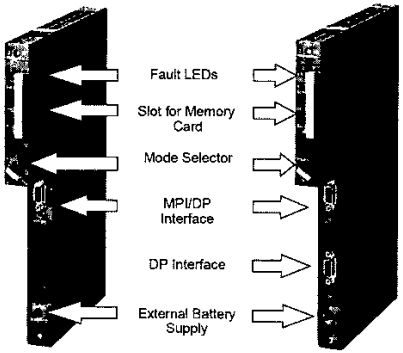
همانطور که قبلاً هم بیان شد، CPU اصلی‌ترین واحد یک PLC می‌باشد. CPUهای S7-400 نسبت به CPUهای S7-300 قدرتمندتر می‌باشند. CPUهای S7-400 از CPU 412-1 شروع شده و از آخرین مدل‌های ارائه شده‌ی آن می‌توان به CPU 417-4 اشاره نمود. از عمده تفاوت‌های CPUهای S7-400 نسبت به S7-300 به موارد زیر می‌توان اشاره نمود.

- در S7-400، CPU یکپارچه[1] وجود ندارد.
- در S7-400، CPUهای IFM وجود ندارد.
- در S7-400 اکثر CPUها دارای پورت شبکه DP می‌باشند.
- برخی CPUهای S7-400 دارای چهار Accumulator می‌باشند. در S7-300 در CPU318 این وضعیت وجود دارد.
- در S7-400، CPU را می‌توان در هر اسلات دلخواه بعد از منبع تغذیه قرار داد.
- در برخی CPUهای S7-400 قابلیت Multicomputing وجود دارد یعنی می‌توان تا چهار CPU را روی یک رک قرار داد. در S7-300 این قابلیت وجود ندارد.
- CPUهای S7-400 امکان Hot Restart دارند ولی در نوع S7-300 این توانایی وجود ندارد.

1. Compact

- در S7-400 برخلاف S7-300 کارت حافظه اختیاری است چون CPU دارای RAM داخلی است

شکل ۱-۲۴ دو نمونه از CPUهای S7-400 را مقایسه کرده است. همانطور که مشخص است تعداد پورت شبکه در آنها متفاوت است.



شکل ۱-۲۴ دو نمونه از CPUهای S7-400

وضعیت ظاهری CPUهای S7-400

با توجه به شکل فوق از نظر ظاهری موارد زیر را می‌توان مورد توجه قرار داد:

- همگی دارای لامپ‌های نشان دهنده هستند که بسیاری از آنها در تمام انواع CPUهای S7-400 مشابه می‌باشند.
- همگی دارای اسلاتی برای وارد کردن کارت حافظه هستند.
- همگی دارای سوئیچ تغییر مد کاری[1] هستند. در نوع قدیمی کلید دارای وضعیت RUN-P نیز بود که در نوع جدید به‌صورت شکل زیر در آمده و همانطور که در کتاب مقدماتی در مورد CPUهای 300 ذکر شد، وضعیت RUN در نوع جدید همان عملکرد RUN-P را دارد.

شکل ۱-۲۵ کلید وضعیت انتخاب وضعیت کاری CPU

- همه دارای پورت شبکه MPI هستند. سایر پورت‌های شبکه برای برخی وجود دارد.
- همه در قسمت پایین دارای محلی جهت اتصال تغذیه‌ی خارجی می‌باشند. کاربرد آن برای تعویض منبع تغذیه در شرایطی که از سیستم بهره‌برداری می‌شود می‌باشد.

لامپ‌های نشان دهنده

CPU دارای لامپ‌های نشان‌دهنده وضعیت کار CPU و خطاهای احتمالی پیش‌آمده می‌باشد.
لامپ‌ها و علائم موجود در این CPUها در جدول ۱-۱۰ نشان داده شده است.

1. RUN-STOP-MRES

جدول ۱-۱۰ لامپ‌های روی CPU های استاندارد و مفاهیم آنها

LED	Color	Meaning	Installed on CPU				
			412-1	412-2 414-2 416-2 416F-2	414-3 416-3	412-2 PN 414-3 PN/DP 414F-3 PN/DP 416-3 PN/DP 416F-3 PN/DP	417-4
INTF	red	Internal fault	X	X	X	X	X
EXTF	red	External fault	X	X	X	X	X
FRCE	yellow	Force command active	X	X	X	X	X
MAINT	yellow	Maintenance request pending	X	X	X	X	X
RUN	green	RUN mode	X	X	X	X	X
STOP	yellow	STOP mode	X	X	X	X	X
BUS1F	red	Bus fault at MPI/PROFIBUS DP interface 1	X	X	X	X	X
BUS2F	red	Bus fault at PROFIBUS DP interface 2	-	X	X	-	X
BUS5F	red	Bus fault at the PROFINET interface	-	-	-	X	-
IFM1F	red	Fault at interface module 1	-	-	X	X	X
IFM2F	red	Fault at interface module 2	-	-	-	-	X

نکات قابل توجه

- INTF به معنای Internal Fault است. اگر CPU در اجرای برنامه دچار مشکل شود یا به‌طور کلی مشکل داخلی داشته باشد، این چراغ با رنگ قرمز روشن می‌شود.
- EXTF به‌معنای External Fault است و بیانگر بروز اشکال در خارج از CPU است. اگر وقتی این چراغ روشن است INTF خاموش باشد، یعنی مشکل خارجی عملکرد CPU را در اجرای برنامه دچار اختلال نکرده است. ولی اگر هر دو چراغ فوق روشن باشند مشکل خارجی اجرای برنامه توسط CPU را نیز با اشکال مواجه ساخته است.
- دو چراغ فوق معادل چراغ SF در S7-300 هستند.
- در مورد چراغ‌های FRCE و RUN و STOP و BUSF به توضیحات کتاب مقدماتی مراجعه شود.
- چراغ‌های IFM1F و IFM2F معرف Interface Module Fualt هستند و در CPUهای خاصی که کارت IF در اسلات روی CPU قرار می‌گیرد کاربرد دارند.

۱-۶-۲ انواع CPUهای S7-400

CPU های S7-400 را به‌طور کلی می‌توان به سه گروه تقسیم‌بندی نمود:

الف) CPUهای استاندارد

ب) CPUهای سری H

ج) CPUهای سری F

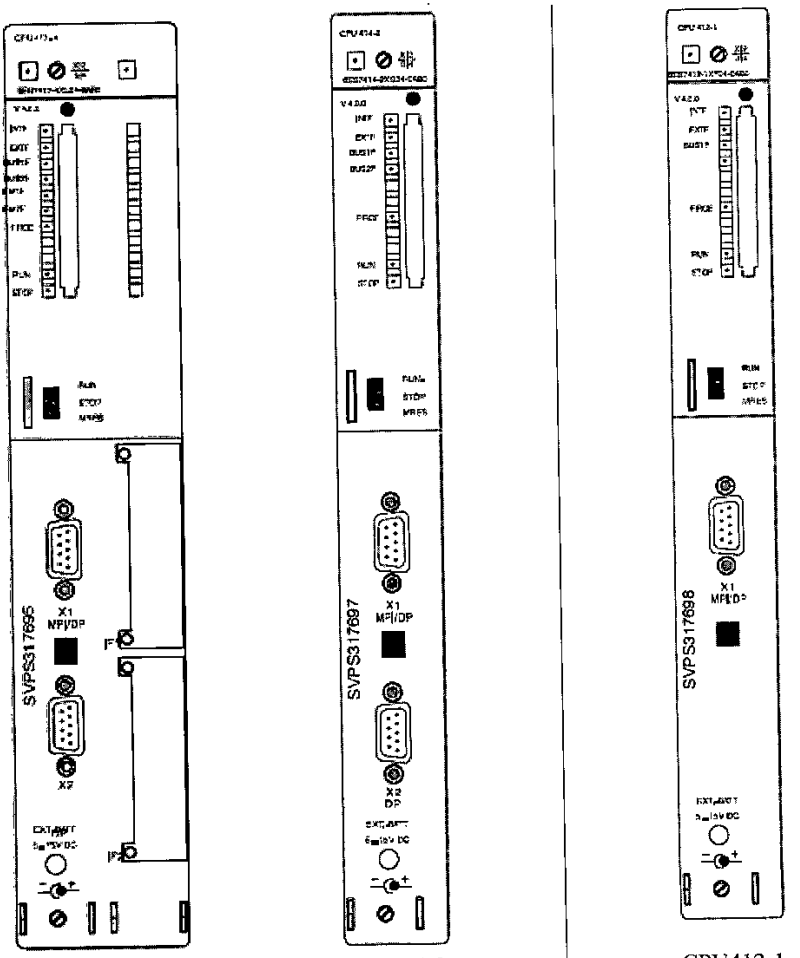
الف) CPU های استاندارد

این CPUها خود به انواع مختلفی تقسیم‌بندی می‌شوند. کد آنها از CPU412 شروع شده و تا CPU417 ادامه می‌یابند. این CPUها از جنبه‌های مختلف با یکدیگر تفاوت دارند از جمله:

- از نظر سرعت پردازش، تعداد ورودی و خروجی، تعداد تایمر و کانتر و حافظه داخلی

- ازنظر تعداد پورت اتصال به شبکه. کد انتهای CPU مانند عدد ۱ یا ۲ یا ۳ یا ۴ معرف تعداد پورت اتصال CPU به شبکه است.
- از نظر نوع پورت شبکه. در انتهای کد CPU حروف DP یا PN به‌ترتیب معرف شبکه Profibus و شبکه Profinet است.

تشریح تمام ویژگی‌های CPU در این کتاب ضرورتی ندارد. برای آشنایی ویژگی نمونه‌هایی از CPUهای استاندارد S7-400 را ذکر می‌کنیم.



CPU417-4 CPU414-2 CPU412-1

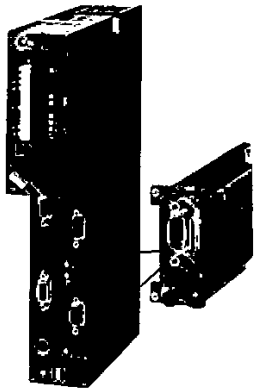
شکل ۱-۲۶ مقایسه ظاهر سه نوع CPU-400

جدول ۱-۱۱

CPU 417-4 6ES7417-4XL04-0AB0	CPU414-2 6ES7414-2XG04-0AB0	CPU412-1 6ES7 412-1XF04-0AB0	
0.03 ms/1000 instruction	0.06 ms/1000 instruction	0.1ms/1000 instruction	سرعت پردازش
16 KB Input 16 KB Output	8 KB Input 8 KB Output	4 KB Input 4 KB Output	تعداد I/O دیجیتال
8192	4096	2048	تعداد I/O آنالوگ
2048	2048	2048	تعداد Timer
2048	2048	2048	تعداد Counter
16 KB	8 KB	4 KB	تعداد Memory Bit
10 MB code 10MB data	256 KB code 256 KB data	72 KB code 72 KB data	حافظه Work Memory
256 KB RAM	256 KB RAM	256 KB RAM	حافظه Load Memory داخلی
64 MB RAM / Flash	64 MB RAM / Flash	64 MB RAM / Flash	ماکزیمم کارت حافظه قابل اضافه کردن
4	2	1	تعداد پورت شبکه

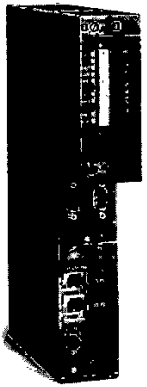
نکات قابل توجه

در CPU417-4 دو اسلات برای نصب کارت IF پیش بینی شده است. همانطور که در شکل دیده می‌شود با نصب کارت‌های IF امکان استفاده از دو پورت جدید برای شبکه Profibus-DP فراهم می‌گردد.



شکل ۱-۲۷ تصویر CPU 417-4

در برخی CPUها پورت Profinet روی CPU موجود است. شکل نمونه‌ای از آنها را در شکل ۱-۲۸می‌بینید. پورت Profinet مشابه LAN به کانکتور RJ45 وصل می‌شود. در کد CPUهایی که مجهز به این پورت هستند کلمه PN نوشته شده است.


CPU 414-3PN/DP

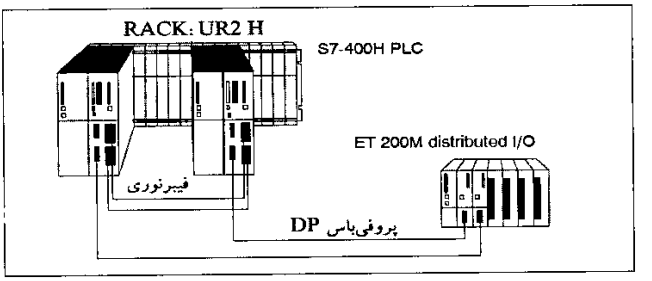

CPU 416-2DP


CPU414-2

شکل ۱-۲۸ تصاویر واقعی چند نمونه CPU-400

ب) CPUهای سری H

این CPU ها قابل استفاده در سیستم افزونه می‌باشند. در سیستم افزونه دو CPU روی رک از نوع UR2 H یا دو رک مجزا قرار می‌گیرند. یکی از CPUها به‌عنوان CPU اصلی و دیگری به عنوان رزرو[1] محسوب می‌گردد. در صورتی‌که CPU اصلی دچار اشکال شود، CPU رزرو به‌طور اتوماتیک و در کسری از ثانیه وارد عمل می‌شود. بدین ترتیب از توقف فرآیند جلوگیری بعمل می‌آید. نحوه‌ی ارتباط CPUهای H با یکدیگر و با I/Oها در شکل ۱-۲۹ نشان داده شده است. تشریح عملکرد سیستم‌های H و FH در کتاب جداگانه‌ای آورده می‌شود.



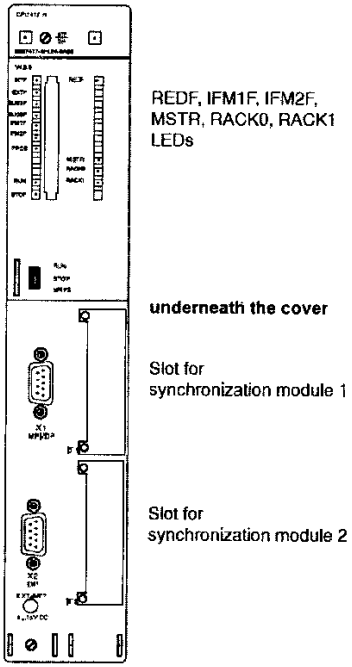
شکل ۱-۲۹ سیستم S7-400 H

CPUهای سری H که تا زمان تألیف این کتاب عرضه شده‌اند، سه مدل دارند:

- CPU 412-3H
- CPU 414-4H
- CPU 417-4H

1. Standby

شکل زیر نمای شماتیک CPU414-4H و CPU417-4H را نشان می‌دهد. نمای ظاهری CPU412-3H با این دو فقط در تعداد پورت‌های روی CPU است. در نوع CPU412-3H پورت DP وجود ندارد و فقط پورت MPI/DP موجود است.



شکل ۱-۳۰ نمای شماتیک CPU های 414-4H و 417-4H

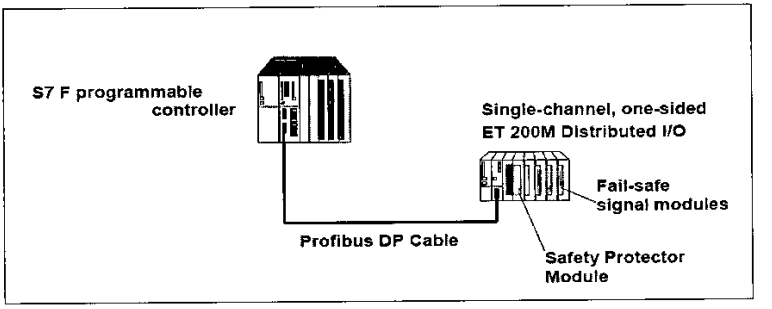
نکات قابل توجه

- در هر سه نوع CPU نوع H، دو پورت روی CPU برای اتصال به فیبر نوری وجود دارد و ماژول رابطی با عنوان Sync Module در آن قرار می‌گیرد.
- CPUهای H قابلیت Fail Safe را نیز دارند و با انجام تنظیمات و تست‌های خاص می‌توان آنها را برای سیستم‌های FH به‌کار برد.
- رونی CPUهای H در مقایسه با نوع استاندارد لامپ‌های اضافه‌تری وجود دارد. به‌عنوان مثال اگر چراغ MSTR روی یک CPU روشن باشد، معرف Master بودن این CPU است. چراغ REDF خطای افزونگی مانند قطع شدن فیبر نوری را نشان می‌دهد. چراغ‌های RACK0 و RACK1 نشان می‌دهند که CPU در کدام رک نصب شده است. تشریح جزئیات آنها در کتاب مربوطه آمده است.

ج) CPUهای سری F

این نوع CPU از CPUهای Fail Safe (ایمن در برابر خطا) محسوب می‌گردند. مشخصه‌ی آنها حرف F در نام CPU می‌باشد. این CPUها را نمی‌توان به‌طریق معمول پیکربندی و استفاده نمود. جزئیات کار با سیستم‌های F و سیستم‌های H در کتاب جداگانه ای آورده می‌شود.

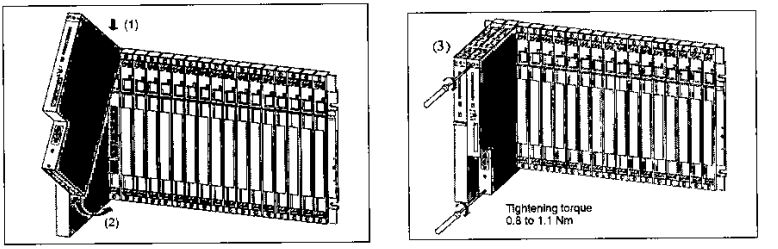
در شکل ۱-۳۱ نمونه‌ای از کاربرد این نوع CPU نشان داده شده است.



شکل ۱-۳۱ کاربرد CPU از نوع Fail safe

۱-۶-۳ نصب CPU در رک

جهت نصب CPU در رک می‌توان مطابق مراحل نشان داده شده در شکل ۱-۳۲ عمل نمود. پس از محکم شدن لبه در نقطه ۱ آنرا طبق مرحله ۲ روی رک قرار دهید تا محکم شود. سپس پیچ های بالا و پایین آنرا ببندید. ترجیحاً زمانی که CPU روی رک قرار می‌گیرد یا از روی رک برداشته می‌شود منبع تغذیه قطع باشد.



شکل ۱-۳۲ طریقه‌ی نصب CPU در رک

۱-۶-۴ نواحی حافظه درCPUهای S7-400

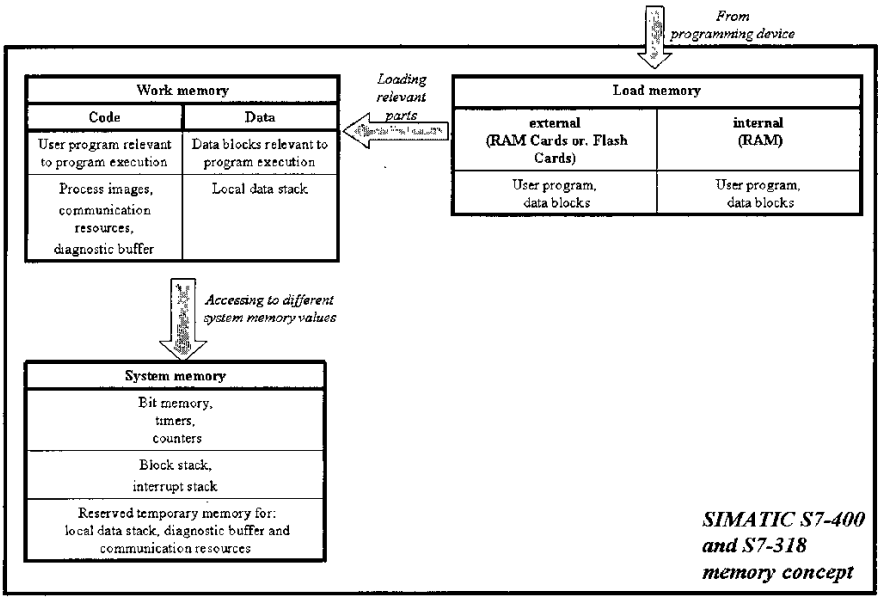
مشابه حافظه‌ی CPUهای S7-300، نواحی حافظه‌ی CPUهای S7-400 به سه قسمت زیر تقسیم‌بندی می‌شود:

۱- **Load Memory**: حافظه‌ی بارگذاری

۲- **Work Memory** : حافظه‌ی کاری

System memory-۳ : حافظه‌ی سیستمی

در شکل ۱-۳۳ نواحی حافظه در S7-400 نشان داده شده است. CPU318 از خانواده S7-300 ساختار حافظه مشابه دارد.



شکل ۱-۳۳ نواحی حافظه در S7-400

Load Memory ۱-۴-۶-۱

CPU های S7-400 دارای حافظه‌ی بارگذاری داخلی می‌باشند که برای ذخیره‌سازی بلاک‌های برنامه و بلاک‌های دیتا به‌کار می‌رود. در S7-400 به دو صورت می‌توان Load Memory را توسعه داد:

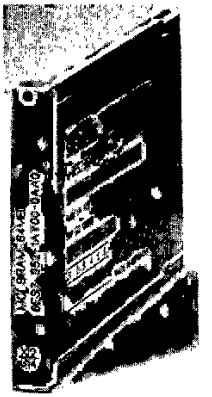
- کارت حافظه از نوع RAM
- کارت حافظه از نوع FLASH EPROM

کارت حافظه RAM

این نوع کارت حافظه به‌منظور توسعه‌ی حافظه‌ی بارگذاری CPUهای S7-400 استفاده می‌شود. در صورت قطع تغذیه‌ی الکتریکی، اطلاعات ذخیره شده در آن از بین می‌رود. از اینرو در صورت استفاده از کارت حافظه‌ی RAM، استفاده از باتری پشتیبان روی PS ضروری می‌باشد.

کارت حافظه FLASH EPROM

این نوع کارت حافظه جهت توسعه‌ی حافظه‌ی بارگذاری و ذخیره‌ی برنامه‌ی کاربر در CPUهای S7-400 استفاده می‌شود. در صورت قطع تغذیه‌ی الکتریکی کارت، اطلاعات موجود در آن از بین نمی‌رود. بدیهی است در صورت استفاده از این نوع کارت حافظه، وجود باتری پشتیبان با هدف حفظ برنامه ضروری نمی‌باشد. در صورت استفاده از این نوع کارت حافظه می‌توان توسط Programmer یا از طریق برنامه‌ی Step7، برنامه را به آن منتقل کرد. توضیحات لازم در کتاب مقدماتی آمده است.



شکل ۱-۳۴ کارت حافظه RAM برای CPU-400

جدول کارت‌های حافظه S7-400

کلیه‌ی کارت‌های موجود در S7-400 اعم از RAM یا Flash Eprom با ظرفیت‌های مختلفی ساخته شده‌اند. در هر پروژه‌ی صنعتی، طراح با توجه به حجم برنامه‌ی نوشته شده و در نظر گرفتن شرایط توسعه‌ی احتمالی سیستم در آینده، اقدام به انتخاب نوع کارت حافظه با میزان فضای حافظه‌ی مناسب می‌نماید. جدول ۱-۱۲ انواع کارت حافظه‌ی S7-400 و میزان حافظه‌ی آنها را نشان می‌دهد.

جدول ۱-۱۲ انواع کارت‌حافظه و ظرفیت آن

Name	Order Number	Current Consumption at 5 V	Backup Currents
MC 952 / 64 Kbytes / RAM	6ES7952-0AF00-0AA0	Typ. 20 mA Max. 50 mA	Typ. 0.5µA Max. 20 µA
MC 952 / 256 Kbytes / RAM	6ES7952-1AH00-0AA0	Typ. 35 mA Max. 80 mA	typ. 1µA Max. 40 µA
MC 952 / 1 Mbyte / RAM	6ES7952-1AK00-0AA0	Typ. 40 mA Max. 90 mA	Typ. 3 µA Max. 50 µA

MC 952 / 2 Mbytes / RAM	6ES7952-1AL00-0AA0	Typ. 45 mA Max. 100 mA / Typ. 5 µA Max. 60 µA
MC 952 / 4 MB / RAM	6ES7952-1AM00-0AA0	Typ. 45 mA Max. 100 mA / Typ. 5 µA Max. 60 µA
MC 952 / 8 MB / RAM	6ES7952-1AP00-0AA0	Typ. 45 mA Max. 100 mA / Typ. 5 µA Max. 60 µA
MC 952 / 16 MB / RAM	6ES7952-1AS00-0AA0	Typ. 100 mA Max. 150 mA / Typ. 50 µA Max. 125 µA
MC 952 / 64 MB / RAM	6ES7952-1AY00-0AA0	Typ. 100 mA Max. 150 mA / Typ. 100 µA Max. 500 µA
MC 952 / 64 Kbytes / 5V Flash	6ES7952-0KF00-0AA0	Typ. 15 mA Max. 35 mA / --
MC 952 / 256 Kbytes / 5V Flash	6ES7952-0KH00-0AA0	Typ. 20 mA Max. 45 mA / --
MC 952 / 1 Mbyte / 5V Flash	6ES7952-1KK00-0AA0	Typ. 40 mA Max. 90 mA / --
MC 952 / 2 Mbytes / 5V Flash	6ES7952-1KL00-0AA0	Typ. 50 mA Max. 100 mA / --
MC 952 / 4 Mbytes / 5V Flash	6ES7952-1KM00-0AA0	Typ. 40 mA Max. 90 mA / --
MC 952 / 8 Mbytes / 5V Flash	6ES7952-1KP00-0AA0	Typ. 50 mA Max. 100 mA / --
MC 952 / 16 Mbytes / 5V Flash	6ES7952-1KS00-0AA0	Typ. 55 mA Max. 110 mA / --
MC 952 / 32 Mbytes / 5V Flash	6ES7952-1KT00-0AA0	Typ. 55 mA Max. 110 mA / --
MC 952 / 64 Mbytes / 5V Flash	6ES7952-1KY00-0AA0	Typ. 55 mA Max. 110 mA / --
Dimensions W x H x D W¢H¢D (in mm)		7,5 x 57 x 87
Weight		Max. 35 g
EMC protection		Provided by construction

قراردادن کارت حافظه در اسلات CPU

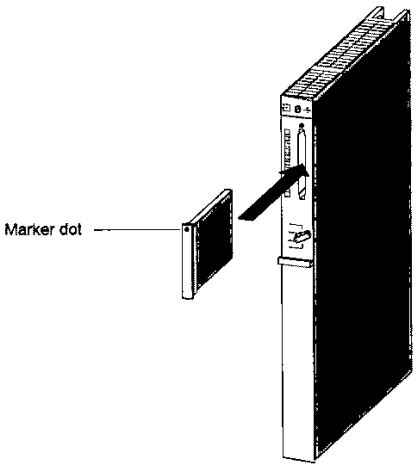
مراحل مختلف قراردادن کارت حافظه در CPU مطابق شکل ۱-۳۵ به صورت زیر می باشد:

۱- کلید انتخاب وضعیت کاری CPU را درحالت STOP قرار دهید.

۲- کارت حافظه را در اسلات CPU با توجه به نقطه‌ی نشانگر[1] موجود روی کارت وارد کنید، نقطه رو به بالا باشد.

۳- پس از وارد کردن کارت چراغ Stop به حالت چشمک زن درمی‌آید. کلید MRES را به مدت ۳ ثانیه نگه داشته رها کنید تا چراغ Stop ثابت شود.

1. Marker Dot



شکل ۱-۳۵ قرار دادن کارت حافظه در CPU

۱-۶-۴-۲ Work Memory

حافظه‌ی کاری از نوع RAM است و قسمت‌های اجرایی برنامه‌ی کاربر در آن قرار می‌گیرد. در صورت قطع تغذیه‌ی الکتریکی اطلاعات موجود در آن ری‌ست می‌شود.

یکی از تفاوت‌های مهم این حافظه با حافظه‌ی کاری S7-300 در این است که برخی از عناصر موجود در System memory مانند PII و PIQ و L-Stack به‌حافظه‌ی کاری‌منتقل‌شده‌اند.شکل‌ابتدای‌فصل‌این‌موضوع را بهترنشان می‌دهد.

در اکثر CPUهای S7-400 امکان مدیریت حافظه‌ی کاری وجود دارد. بدینصورت که می‌توان میزان حافظه‌ی اختصاص داده شده به هر بخش از حافظه‌ی کاری را در محدوده‌ی خاصی تغییر داد. مثلاً امکان تغییر اندازه‌ی PII و PIQ وجود دارد. تنظیم مربوطه در ادامه آورده شده است. در برخی CPUها این تنظیم امکان‌پذیر نیست.

قسمت‌هایی از حافظه‌ی Work memory که در اکثر CPUهای S7-400 قابل تغییر هستند عبارتند از:

- PII: جدول تصاویر ورودی‌ها
- PIQ: جدول تصاویر خروجی‌ها
- Communication resources
- diagnostic buffer: بافر تشخیص عیب
- میزان دیتاهای محلی مربوط به OBها

محاسبه‌ی ظرفیت اشغالی عناصر موجود در Work memory

جهت تغییر میزان ظرفیت عناصر موجود در حافظه‌ی کاری باید توجه نمود که افزایش اندازه‌ی ظرفیت این عناصر سبب کاهش حافظه‌ی مربوط به پردازش برنامه شده و سرعت کار CPU را کاهش می‌دهد. جدول ۱-۱۳ میزان فضای اشغالی عناصر موجود در حافظه‌ی کاری را نشان می‌دهد.

جدول ۱-۱۳ میزان فضای اشغالی عناصر موجود در حافظه‌ی کاری

Parameter	Required Working Memory	In Code/Data Memory
Size of the process inputs image	12 bytes per byte in the process input image	Code memory
Size of the process outputs image	12 bytes per byte in the process output image	Code memory
Communication resources (communication jobs)	72 bytes per communication job	Code memory
Size of diagnostic buffer	32 bytes per entry in the diagnostic buffer	Code memory
Volume of local data	1 byte per byte of local data	Data memory

تذکر: منظور از عبارت 12 byte per byte این است که هر بایت از ناحیه PII یا PIQ، ۱۲ بایت از بخش Code Memory در Work Memory فضا اشغال می‌نمایند.

موارد ذکر شده در مورد تغییر اندازه برخی از عناصر موجود در Work Memory در قسمت بعد توضیح داده می‌شود.

حافظه‌ی Work Memory در همه‌ی CPUهای S7-400 (به غیر از CPU 417-4 و CPU 417-4H) غیرقابل توسعه می‌باشد. در CPUهای 417-4 و 417-4H توسط استفاده از کارت حافظه می‌توان حافظه‌ی کاری را توسعه داد. این CPUها دارای دو اسلات در بدنه‌ی کناری خود است که مطابق جدول ۱-۱۴ می‌توان در هریک از آنها کارت حافظه مورد نظر را قرار داد.

جدول ۱-۱۴ کارت‌های حافظه جهت توسعه حافظه‌ی Work Memory

Slot 1	Slot 2
2 Mbytes	-
4 Mbytes	-
4 Mbytes	2 Mbytes
4 Mbytes	4 Mbytes

مراحل قرار دادن کارت حافظه جهت توسعه‌ی Work Memory

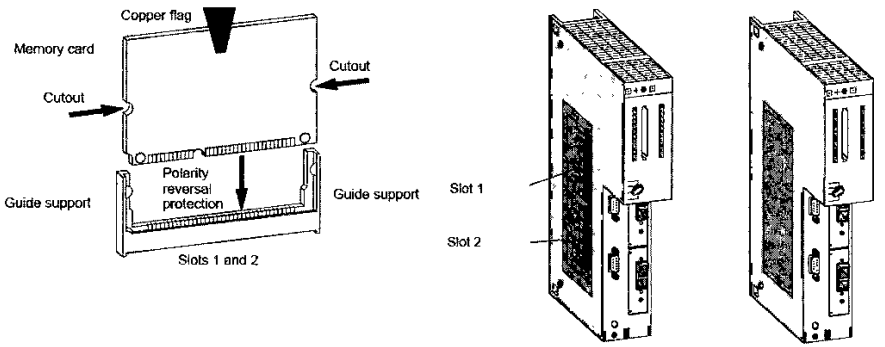
جهت قرار دادن کارت حافظه‌ی مربوط به حافظه‌ی کاری می‌توان مطابق شکل ۱-۳۶ مراحل زیر را انجام داد:

۱- کاور سمت چپ CPU را درآورید.

۲- کارت حافظه را در اسلات شماره یک قرار دهید.

۳- در صورت نیاز کارت حافظه‌ی دوم را در اسلات شماره‌ی دو قرار دهید.

۴- کاور را در سر جای خود محکم نمایید.



شکل ۱-۳۶ قرار دادن کارت حافظه جهت توسعه‌ی حافظه‌ی Work Memory در CPU417

۱-۶-۴-۳ System Memory

حافظه‌ی سیستمی از نوع RAM است و عناصری مانند تایمرها، کانترها، Bit Memory، Block Stack و Interrupt Stack در آن قرار دارد.

Retentive Memory

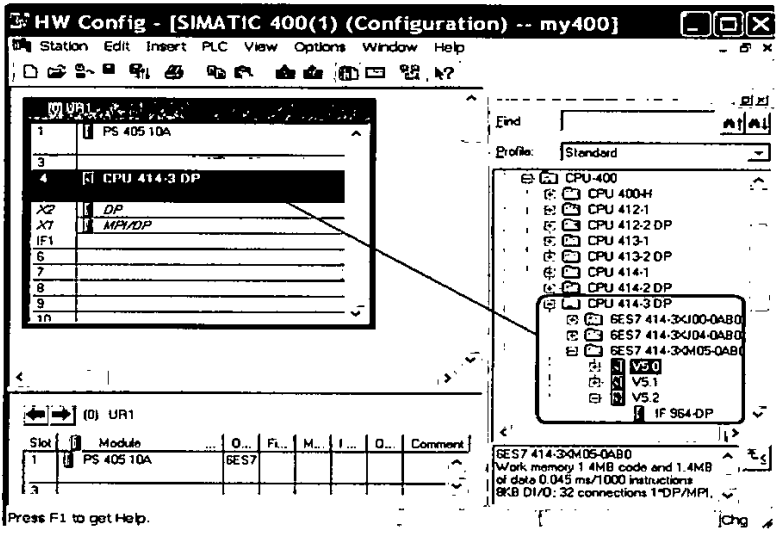
حافظه‌ی ماندگار است و برخی از عناصری که در System Memory قرار دارند را می‌توان در آن به‌صورت ماندگار تعریف نمود. این عناصر عبارتند از :Timers ، Counters و Bit Memory.

همچنین در برخی از CPUها که دیتا بلاک‌های غیر ماندگار دارند، در حافظه‌ی ماندگار امکان تعریف دیتا بلاک نیز وجود دارد.

۱-۶-۵ پیکربندی CPU از طریق نرم‌افزار HW Config

الف) CPUهای استاندارد S7-400

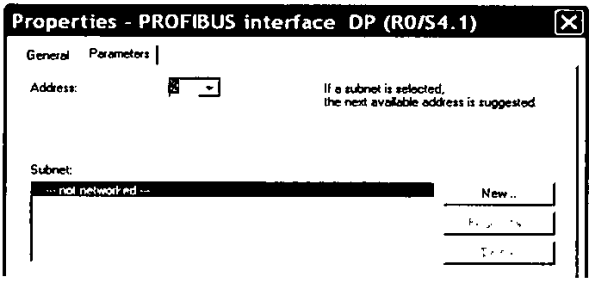
جهت پیکربندی این سیستم به روشی که ذکر شد، یک Station Simatic 400 ایجاد نموده و پس از انتخاب رک و منبع تغذیه، می‌توان CPU مورد نظر را در رک قرار داد. جهت پیکربندی این نوع CPUها علاوه بر STEP 7 نیاز به نرم‌افزار خاصی نمی‌باشد. در شکل ۱-۳۷ پیکربندی S7-400 استاندارد نشان داده شده است.



شکل ۱-۳۷ پیکربندی S7-400 استاندارد

نکته‌ای که در شکل ۱-۳۷ دیده می‌شود، این است که CPU بعد از منبع تغذیه در هر اسلاتی به‌جز اسلات آخر می‌تواند قرار گیرد. اسلات آخر برای کارت IM می‌باشد.

پس از وارد کردن CPU در اسلات فوق اگر CPU پورت DP داشته باشد، پنجره تعریف و تنظیم پروفی‌باس ظاهر می‌شود. با کلیک روی New شبکه پروفی باس برای پورت DP فعال خواهد شد.



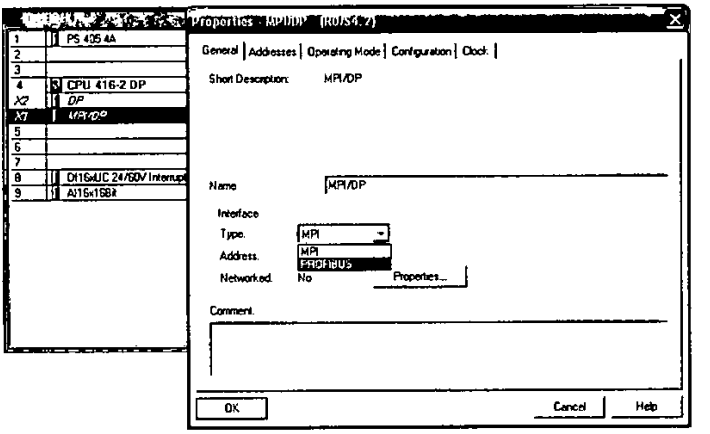
شکل ۱-۳۸ تعریف شبکه پروفی باس

تنظیم پورت MPI/DP

در CPUهایی که عدد 2 در انتهای کد آنها دیده می‌شود مانند CPU414-2، پورت دیگری با عنوان MPI/DP وجود دارد. به‌صورت پیش‌فرض تنظیم این پورت روی MPI است و در صورت لزوم می‌توان توسط نرم‌افزار آنرا به‌صورت DP تنظیم نمود.

به‌منظور تنظیم این پورت می‌توان مطابق شکل ۱-۳۹ بر روی گزینه‌ی MPI/DP زیر CPU قرار گرفته در رک دابل کلیک نموده و سپس از کادر باز شده و در قسمت Interface نوع پورت را انتخاب نمود. در این قسمت گزینه‌های MPI و

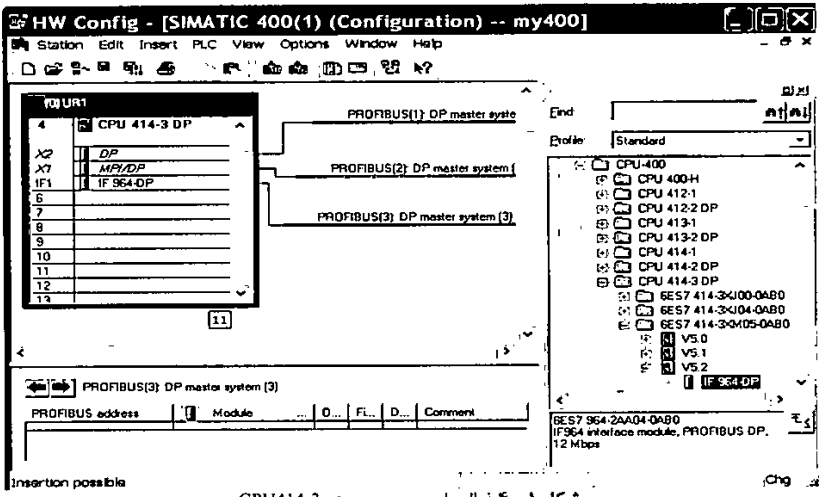
PROFIBUS وجود دارند. در صورتی‌که گزینه‌ی PROFIBUS انتخاب گردد پنجره ای مانند شکل ۱-۳۸ برای تعریف پروفی باس جدید باز می‌گردد.



شکل ۱-۳۹ تنظیم پورت MPI / DP

پیکربندی IF

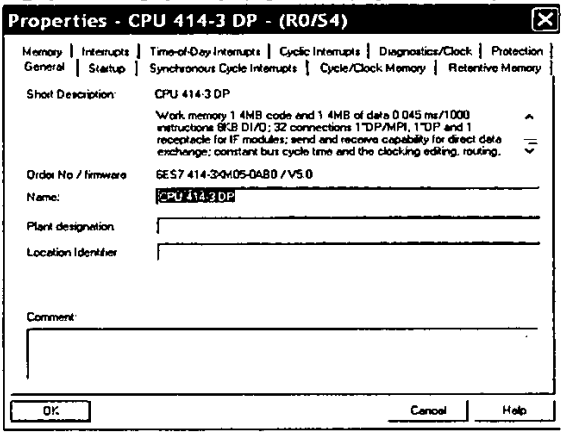
در CPUهایی که دارای اسلات برای کارت اینترفیس IF هستند مانند CPU414-3، همانطور که در شکل ۱-۳۹ مشاهده می‌شود یک اسلات خالی با عنوان IF در زیر CPU قرار دارد. کارت IF را می‌توان از کاتالوگ از زیر مجموعه CPU انتخاب و وارد کرد و شبیه حالت قبل شبکه پروفی‌باس را برای آن فعال نمود. شکل زیر CPU414-3 را با سه شبکه پروفی‌باس نشان می‌دهد.



شکل ۱-۴۰ فعال‌سازی هر سه پورت CPU414-3

تنظیم پارامترهای CPU

پس از وارد کردن CPU می‌توان پارامترهای آن را تنظیم نمود. برای این کار کافیست روی CPU دابل کلیک کنید. شکل زیر سربرگ‌های مختلف مربوط به پارامترهای CPU را نشان می‌دهد. موارد مربوط به وقفه‌ها[1] را در فصل‌های بعد تشریح خواهیم نمود. در اینجا فقط برخی موارد که تا این مرحله فهم آن بر خواننده روشن است را ذکر می‌نماییم.

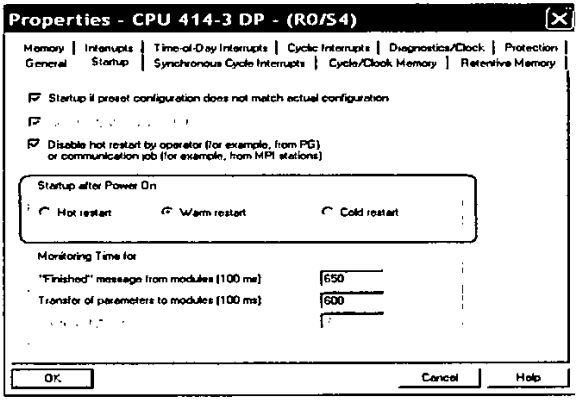


شکل ۱-۴۱ پارامترهای CPU-400

تنظیم راه‌اندازی مجدد CPU

در کتاب مقدماتی با انواع راه‌اندازی CPU آشنا شدید. در اینجا فقط این نکته ذکر می‌شود که در CPUهای S7-400 امکان انجام هر سه نوع راه‌اندازی **Cold**، **Warm** و **Hot** امکان‌پذیر است.

برای تنظیم نوع راه‌اندازی می‌توان در محیط برنامه‌ی HW Config روی CPU موجود در رک دابل‌کلیک نموده و از پنجره‌ی باز شده در قسمت Startup مطابق شکل ۱-۴۲ نوع راه‌اندازی را انتخاب نمود.

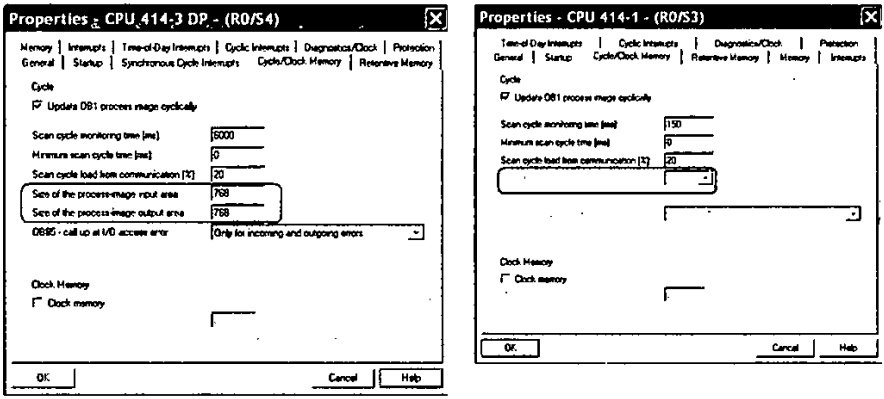


شکل ۱-۴۲ تنظیم نوع راه‌اندازی

1. Interrupts

تنظیم حافظه PII و PIQ

در بسیاری از CPUهای S7-400 برخلاف S7-300 می‌توان ناحیه PII و PIQ را در سربرگ Cycle/Clock Memory تنظیم کرد. همانطور که قبلاً ذکر شد این ناحیه در Work Memory قرار دارد. شکل ۱-۳۴ این سربرگ را برای دو CPU مختلف از خانواده S7-400 نشان می‌دهد. همانطور که دیده می‌شود تنظیم PII و PIQ فقط برای یکی از این دو امکان‌پذیر است.

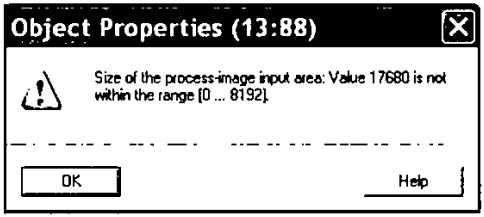


CPU414-3 CPU414-1

شکل ۱-۴۳ تنظیمات PII و PIQ

شکل ۱-۴۳ نشان می‌دهد که تعداد 768 Byte برای هر کدام از نواحی PII و PIQ در نظر گرفته شده است. از این ناحیه می‌توان برای سیگنال‌های دیجیتال و آنالوگ استفاده نمود. در عین حال می‌توان سیگنال‌های آنالوگ را در خارج از ناحیه فوق آدرس‌دهی کرد. توضیحات بیشتر در فصل بعد آمده است.

در صورت لزوم عدد 768 Byte را می‌توان افزایش داد، ولی حد ماکزیممی برای آن در نظر گرفته شده است. در صورت تجاوز از این حد با پیغام زیر مواجه می‌شویم که برای CPU414-3 این حد را 8192 بایت ذکر می‌کند.



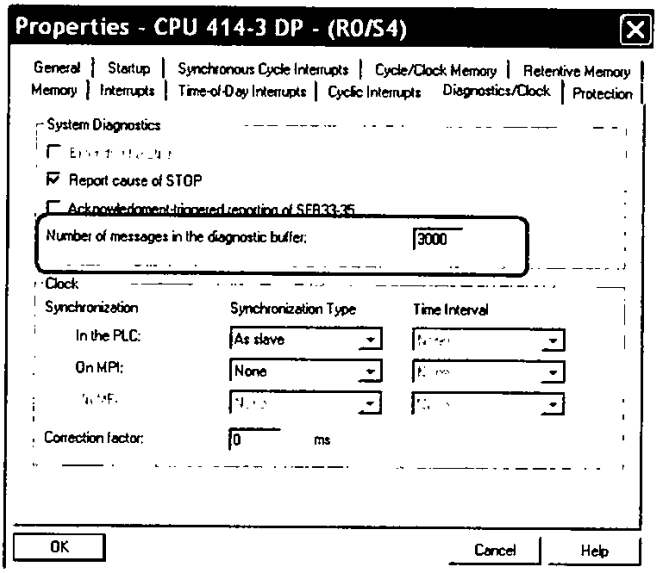
شکل ۱-۴۴ وارد کردن آدرس خارج از حد برای PII/PIQ

تنظیمات سیکل اسکن

در CPUهایی که حافظه PII و PIQ در Work Memory قرار دارد ماکزیمم زمان سیکل اسکن همانطور که در شکل۱-۴۳ دیده می‌شود 6000 ms است که بسیار بیشتر از زمان 150 ms سایر CPUها می‌باشد.

تنظیم تعداد پیغام در بافر

در سربرگ Diagnostics/Clock می‌توان تعداد پیغامی که در بافر ثبت می‌شود را تغییر داد. در شکل ۱-۴۵ این تعداد به‌صورت پیش‌فرض روی 3000 تنظیم شده است. در S7-300 تعداد پیغام بافر ماکزیمم 100 است.



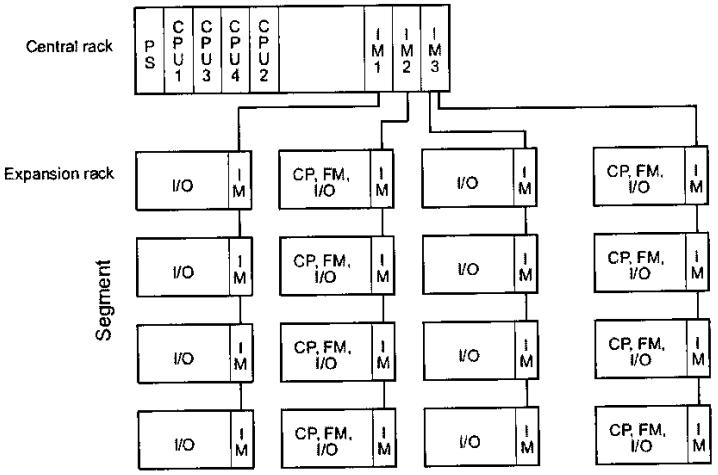
شکل ۱-۴۵ کلید وضعیت انتخاب وضعیت کاری CPU

قابلیت Multicomputing

منظور از این قابلیت، عملکرد همزمان چند CPU در یک رک می‌باشد (حداکثر ۴ عدد) که هر کدام برنامه مستقلی را اجرا می‌نماید به گونه‌ای که کارها به‌صورت موازی انجام می‌شود. همه‌ی CPUها با هم START شده و با هم به مد STOP می‌روند. از کاربردهای آن می‌توان به موارد زیر اشاره نمود:

- زمانی‌که برنامه‌ی کاربر بسیار بزرگ باشد و یک CPU به تنهایی قادر به پردازش آن نباشد.
- زمانی که نیاز باشد برنامه‌ی قسمتی از پروسه سریعتر از برنامه‌ی سایر قسمت‌ها پردازش شود.
- در مواردی که پروسه شامل چند بخش مستقل از هم باشد به نحوی که توسط هر CPU برنامه‌ی یک بخش از پروسه پردازش و اجرا شود.

هر CPU تنها به ماژول‌هایی دسترسی دارد که در موقع پیکربندی به آن اختصاص داده شده باشد. لازم به ذکر است که در همه‌ی CPU های S7-400 قابلیت Multicomputing وجود ندارد، بنابراین تنها می‌توان از CPUهایی استفاده نمود که این قابلیت را پشتیبانی نمایند. در شکل ۱-۴۶ یک سیستم با استفاده از چهار CPU با قابلیت Multicomputing نشان داده شده است.



شکل ۱-۴۶ چهار CPU با قابلیت Multicomputing

رک‌های مناسب جهت Multicomputing

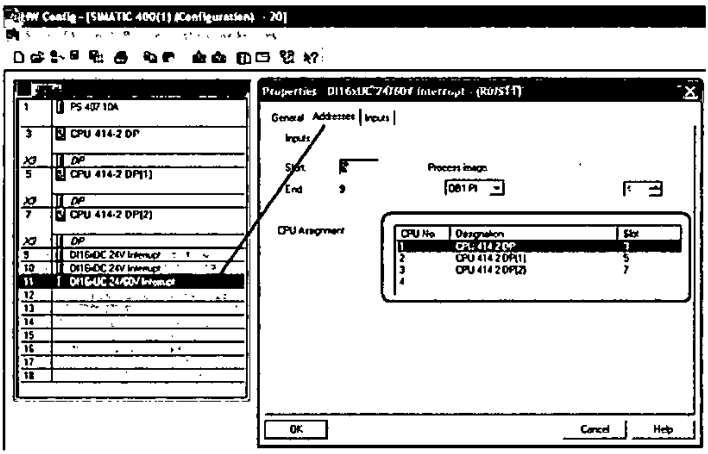
رک‌هایی که می‌توان CPUهای با قابلیت Multicomputing را روی آنها قرار داد عبارتند از:

- UR1 و UR2
- UR2 H
- CR3: در این رک به‌دلیل اینکه فقط چهار اسلات قرار دارد، تنها می‌توان از دو عدد CPU استفاده نمود.

نکته: در رک CR2 به‌دلیل اینکه رک قسمت‌بندی شده و در هر قسمت تنها یک عدد CPU می‌توان قرار داد، امکان استفاده از قابلیت Multicomputing وجود ندارد.

تنظیمات مورد نیاز جهت پیکربندی Multicomputing

پس از اینکه چند CPU با قابلیت Multicomputing روی رک قرار داده شد، باید هر کدام از ماژول‌های دیگر را به یکی از CPUها مرتبط نمود. برای این کار می‌توان روی ماژولی که در رک قرار داده شده است دابل‌کلیک نموده و در سر برگ آدرس، CPU مورد نظر را انتخاب نمود. در اینصورت CPU به آن کارت دسترسی خواهد داشت. همانطور که اشاره شد، CPU فقط به کارت‌هایی که به این روش برای آن تعریف شوند دسترسی داشته و به سایر کارت‌ها دسترسی ندارد. چگونگی پیکربندی و انجام تنظیمات مربوط به Multicomputing در شکل ۱-۴۷ نشان داده شده است.

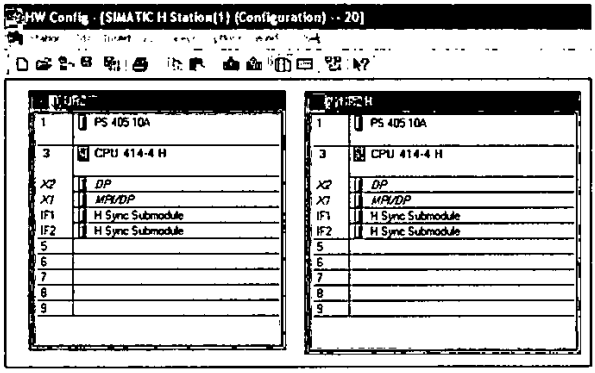


شکل ۱-۴۷ پیکربندی و انجام تنظیمات مربوط به Multicomputing

ب) پیکربندی CPUهای سری S7-400H از طریق نرم‌افزار HW Config

پیکربندی این سیستم‌ها خارج از بحث این کتاب است و جداگانه تشریح می‌شود. فقط به نکات زیر توجه نمایید:

- برای پیکربندی این CPUها در محیط Simatic Manager بایستی سیستم Simatic 400 H Station انتخاب شود.
- در محیط برنامه‌ی HWConfig نیاز به وارد کردن دو رک UR وجود دارد که معمولاً رک UR2H انتخاب می‌شود.
- در هر رک بایستی یک منبع تغذیه و یکCPU وارد شود.
- پس از وارد کردن CPU بایستی ماژول سنکرون ساز را در زیر آن معرفی نمود.
- توجه شود که نوع و مشخصات دو CPU و ماژول‌های سنکرون‌ساز کاملاً با یکدیگر یکسان باشد. در شکل ۱-۴۸ نمونه‌ای از پیکربندی S7-400 H نشان داده شده است.



شکل ۱-۴۸ پیکربندی S7-400 H

ج) پیکربندی CPUهای سری S7-400F از طریق نرم‌افزار HW Config

برای پیکربندی این سیستم‌ها نرم‌افزار Step7 به‌تنهایی کافی نیست و لازم است نرم‌افزارهای زیر نیز نصب گردند:

- S7 F Systems
- S7 F Configuration Pack

تشریح پیکربندی این سیستم‌ها خارج از بحث این کتاب است.

۱-۷ SM (کارت‌های ورودی/ خروجی)

۱-۷-۱ انواع و عملکرد

در S7-400 نیز به‌منظور تبدیل سیگنال الکتریکی ورودی به دیتا از کارت‌های ورودی و جهت تبدیل دیتا به سیگنال الکتریکی خروجی از کارت‌های خروجی استفاده می‌شود. انواع کارت‌های I/O در S7-400 را می‌توان به چهار گروه تقسیم نمود:

۱- **DI:** کارت دیجیتال ورودی

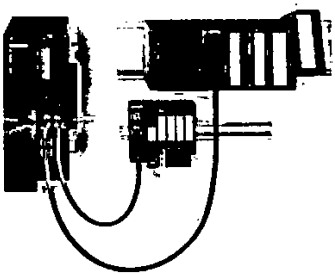
۲- **DO:** کارت دیجیتال خروجی

۳- **AI:** کارت آنالوگ ورودی

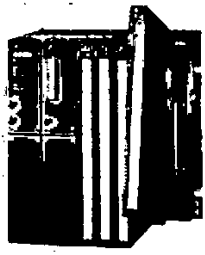
۴- **AO:** کارت آنالوگ خروجی

جزئیات عملکرد این کارت‌ها مشابه کارت‌های SM-300 است که در کتاب مقدماتی به تفصیل تشریح شد؛ بنابراین در اینجا از تکرار آن خودداری می‌کنیم.

باید توجه داشت که تنوع کارت‌های ورودی و خروجی در S7-400 نسبت به S7-300 کمتر است و امروزه معمولاً به‌جای استفاده از کارت‌های I/O در S7-400 از Remote I/Oها که روی آنها کارت‌های I/O مخصوص نصب می‌شود، استفاده می‌شود. این طرح نسبت به استفاده از کارت‌های SM400 اقتصادی‌تر است و مزایای دیگری از جمله وابسته نبودن به سازنده خاص را نیز به همراه دارد.



With Remote I/O



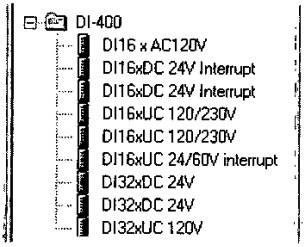
With Central I/O

شکل ۱-۴۹ حالت‌های مختلف اتصال I/O به S7-400

۱-۷-۲ کارت های DI-400

همانطور که در شکل ۱-۵۰ مشاهده می‌شود، تنوع کارت‌های DI نسبت به S7-300 کمتر می‌باشد. با توجه به ولتاژ ورودی، انواع کارت‌های DI را می‌توان به گروه‌های زیر تقسیم نمود:

- DI 24V DC
- DI 120V AC
- DI 120/230V UC
- DI 24/60 V UC



شکل ۱-۵۰

کارت‌های UC هم می‌توانند ورودی AC دریافت نمایند و هم ورودی DC.

مشخصات فنی

در جدول ۱-۱۵ انواع کارت‌های DI-400 و برخی از مشخصات آنها نشان داده شده است.

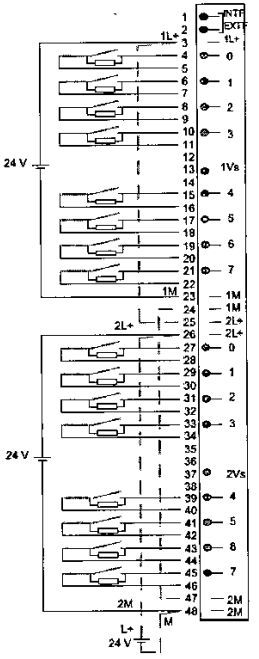
جدول ۱-۱۵ برخی از مشخصات کارت‌های DI-400

	SM 421; DI 32 x 24 VDC (-1BL0x-)	SM 421; DI 16 x 24 VDC (-7BH00-)	SM 421; DI 16__ 120 VAC (-5EH00-)	SM 421; DI 16x 24/60 VUC (-7DH00-)	SM 421; DI 16x 120/230 VUC (-1FH00-)	SM 421; DI 16x 120/230 VUC (-1FH20-)	SM 421; DI 32 x 120 VUC (-1EL00-)
Number of inputs	32 DI; isolated in groups of 32	16 DI; isolated in groups of 8	16 DI; isolated in groups of 1	16 DI; isolated in groups of 1	16 DI; isolated in groups of 4	16 DI; isolated in groups of 4	32 DI; isolated in groups of 8
Rated input voltage	24 VDC	24 VDC	120 VAC	24 VUC to 60 VUC	120 VAC/ 230 VDC	120/230 VUC	120 VAC/ VDC
Suitable for...	Switches , Two-wire proximity switches (BEROs)						
Programmable diagnostics	No	Yes	No	Yes	No	No	No
Diagnostic Interrupt	No	Yes	No	Yes	No	No	No
Hardware interrupt upon edge change	No	Yes	No	Yes	No	No	No
Adjustable input delays	No	Yes	No	Yes	No	No	No

Substitute value output	–	Yes	–	–	–	–	–
Special Features	High packaging density	Quick and with interrupt capability	Channel-specific isolation	Interrupt capability with low, variable voltages	For high, variable voltages	For high, variable voltages Input characte-ristic curve to IEC 61131-2	High packaging density

اتصالات کارت DI-400

جهت مشاهده‌ی نقشه‌ی داخلی کارت DI و اتصالات می‌توان به کاتالوگ کارت مراجعه نمود. در شکل ۱-۵۱ نمایی از کارت SM 421; DI 16 x DC 24 V Interrupt نشان داده شده است. در این کارت قابلیت تشخیص قطعی سیم و قطعی تغذیه وجود دارد. این کارت دارای دو منبع تغذیه‌ی داخلی (VS) می‌باشد که تغذیه‌ی سنسورها و کلیدها از آن انجام می‌پذیرد. همانطور که مشخص است، برای این که کارت بتواند قطعی سیم را تشخیص دهد باید یک مقاومت با سنسور به‌طور موازی اتصال داد. این کارت دارای دو لامپ نشان‌دهنده‌ی خطاهای پیش آمده نیز می‌باشد. جدول ۱-۱۶ لیست خطاهای پیش آمده و وضعیت لامپ‌ها را نشان می‌دهد.



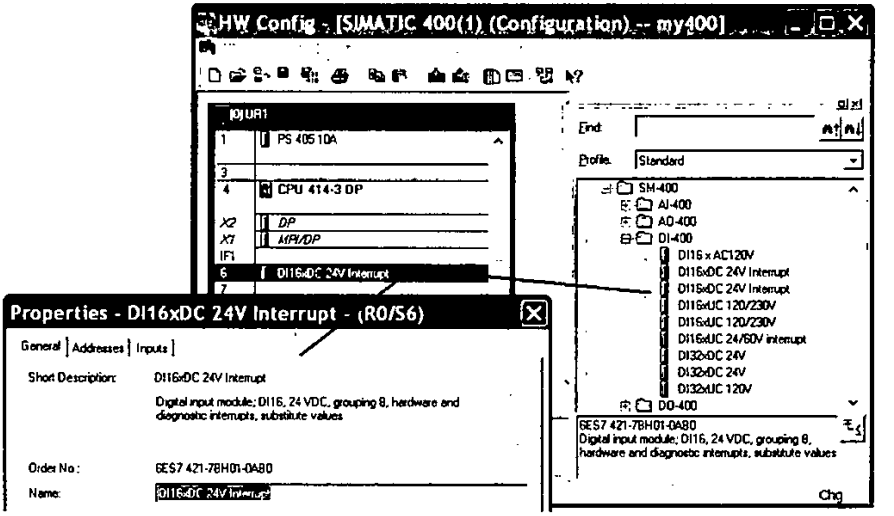
شکل ۱-۵۱

جدول ۱-۱۶

Diagnostic Message	LED	Scope of the Diagnosis	Parameters Can Be Assigned
Module problem	INTF/EXTF	Module	No
Internal malfunction	INTF	Module	No
External malfunction	EXTF	Module	No
Channel error present	INTF/EXTF	Module	No
External auxiliary supply missing	EXTF	Module	No
Front connector missing	EXTF	Module	No
Module not parameterized.	INTF	Module	No
Wrong parameters	INTF	Module	No
Channel information available	INTF/EXTF	Module	No
STOP mode	–	Module	No
Internal voltage failure	INTF	Module	No
EPROM error	INTF	Module	No
Hardware interrupt lost	INTF	Module	No
Parameter assignment error	INTF	Channel	No
Short-circuit to M	EXTF	Channel	Yes
Short-circuit to L+	EXTF	Channel	Yes
Wire break	EXTF	Channel	Yes
Fuse blown	INTF	Channel	Yes
Sensor supply missing	EXTF	Channel/channel group	Yes
No load voltage L+	EXTF	Channel/channel group	Yes

پیکربندی کارت‌های DI 400 در Hwconfig

پس از آنکه کارت DI مناسب انتخاب و در رک قرار گرفت، جهت تنظیم مشخصات آن می‌توان بر روی کارت مورد نظر دوبار کلیک نمود.



شکل ۱-۵۲

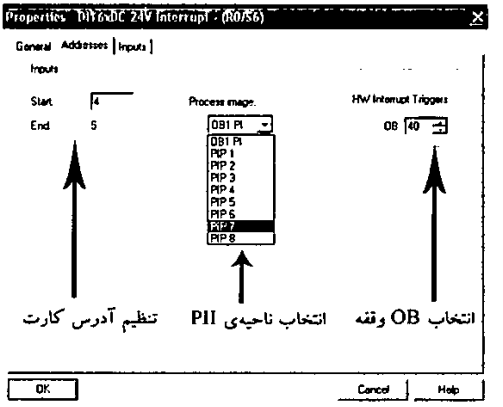
در اینصورت پنجره‌ی تنظیمات کارت باز می‌شود. در این پنجره ۲ یا ۳ سربرگ وجود دارد. این سربرگ‌ها عبارتند از: General , Address , Inputs.

الف) سربرگ General: شماره فنی و توضیحات مختصری در مورد کارت در این قسمت وجود دارد.

ب) سربرگ Addresses: در این‌سربرگ‌می‌توان آدرس اختصاص یافته به کارت را مشاهده و در صورت لزوم تغییر داد.

سر برگ Addresses

تنظیمات موجود در این سربرگ مشابه تنظیمات کارت‌های DI-300 می‌باشد.



شکل۱-۵۳ تنظیمات موجود در سربرگ Addresses

اکثر CPUهای S7-400 دارای PII به‌صورت پارتیشن‌بندی شده هستند، بنابراین امکان انتخاب ناحیه‌ای از PII که آدرس کارت در آن قرار دارد وجود دارد.

در کارت‌هایی که دارای قابلیتHardware Interrupt می‌باشند، امکان تعیین OB مورد نظر جهت این وقفه نیز وجود دارد.

بحث وقفه‌ها در ادامه آورده شده است.

ج) سربرگ intputs: در این سربرگ برخی از تنظیمات از جمله قابلیت‌های Diagnostic Interrupt و Hardware Interrupt قابل انجام می‌باشد. باید توجه نمود که فقط کارت‌هایی که از قابلیت‌های فوق پشتیبانی می‌نمایند، دارای این سربرگ می‌باشند.

همانطور که در شکل ۱-۵۴ ملاحظه می‌شود، اکثر تنظیمات موجود در این سربرگ مشابه تنظیمات کارت‌های DI-300 می‌باشد که در کتاب مقدماتی تشریح شد. در اینجا فقط به برخی تنظیمات جدید اشاره می‌گردد:

- **Destination CPU For Interrupt**

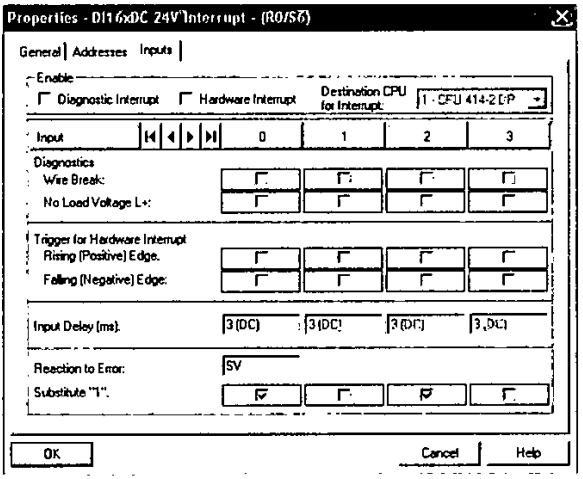
این گزینه زمانی که از چند CPU به‌صورت Multi computing استفاده شود، به منظور تعیین CPU مورد نظر جهت اعمال وقفه به آن مشخص می‌شود.

- **Reaction to Error**

توسط این تنظیم می‌توان نحوه‌ی عکس‌العمل CPU نسبت به بروز خطا در ماژول را مشخص نمود. گزینه‌های قابل انتخاب عبارتند از:

SV: با انتخاب این گزینه می‌توان مشخص نمود که در صورت بروز اشکال در کارت، چه مقداری در PII و در آدرس مربوط به کانال مورد نظر قرار گیرد. اگر گزینه‌ی Substitute 1 علامت زده شود، در صورت بروز اشکال در کارت مقدار یک منطقی در آدرس مربوط به کانال قرار می‌گیرد. اما اگر این گزینه علامت زده نشود، مقدار صفر در آدرس مربوط به کانال قرار می‌گیرد.

KLV: اگر این گزینه انتخاب گردد، آخرین مقدار معتبر قبل از بروز اشکال در کارت حفظ می‌شود.



شکل ۱-۵۴ تنظیمات سر برگ Inputs

۱-۷-۳ کارت‌های DO-400

تنوع کارت‌های DO-400 نسبت به کارت‌های DO-300 بسیار کمتر می‌باشد. مشابه کارت‌های DO-300، در اینجا نیز کارت رله‌ای با قابلیت جریان‌دهی زیاد وجود دارد.

مشخصات فنی

در جدول ۱-۱۷ انواع کارت‌های خروجیS7-400 نشان داده شده است

جدول ۱-۱۷ انواع کارت‌های DO و برخی از مشخصات آنها

Module Characte-ristics	SM 422; DO 16 x 24 VDC/2 A (-1BH1x)	SM 422; DO 16 x 20-125 VDC/1.5 A (-5EH10)	SM 422; DO 32 x 24 VDC/ 0.5 A (-1BL00)	SM 422; DO 32 x 24 VDC/0.5 A (-7BL00)	SM 422; DO 8 x 120/230 VAC/5 A (-1FF00)	SM 422; DO 16 x 120/230 VAC/2 A (-1FH00)	SM 422; DO 16x 20-120 VAC/2 A (-5EH00)
Number of outputs	16 DO; isolated in groups of 8	16 DO; isolated and reverse polarity protection in groups of 8	32 DO; isolated in groups of 32	32 DO; isolated in groups of 8	8 DO; isolated in groups of 1	16 DO; isolated in groups of 4	16 DO; isolated in groups of 1
Output current	2 A	1.5 A	0.5 A	0.5 A	5 A	2 A	2 A
Rated load voltage	24 VDC	20 to 125 VDC	24 VDC	24 VDC	120/ 230 VAC	120/ 230 VAC	20 to 120 VAC
Programmable diagnostics	No	Yes	No	Yes.	No	No	Yes
Diagnostic Interrupt	No	Yes	No	Yes	No	No	Yes
Substitute value output	No	Yes	No	Yes	No	No	Yes
Special Features	For high currents	For variable voltages	High packaging density	Particularly quick and with interrupt capability	For high currents with channel-specific isolation	–	For variable currents with channel-specific isolation

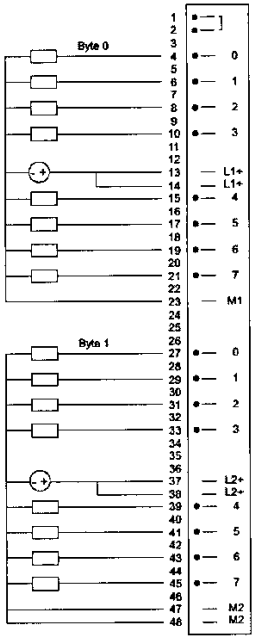
کارت رله‌ای

در جدول ۱-۱۸ برخی از مشخصات کارت رله‌ای نشان داده شده است.

جدول ۱-۱۸ مشخصات کارت رله‌ای

SM 422; DO 16 x 30/230 VUC/Rel. 5 A (-1HH00)	
Number of Outputs	16 outputs, isolated in groups of 8
Load Voltage	125 VDC 230 VAC
Special Features	–

اتصالات

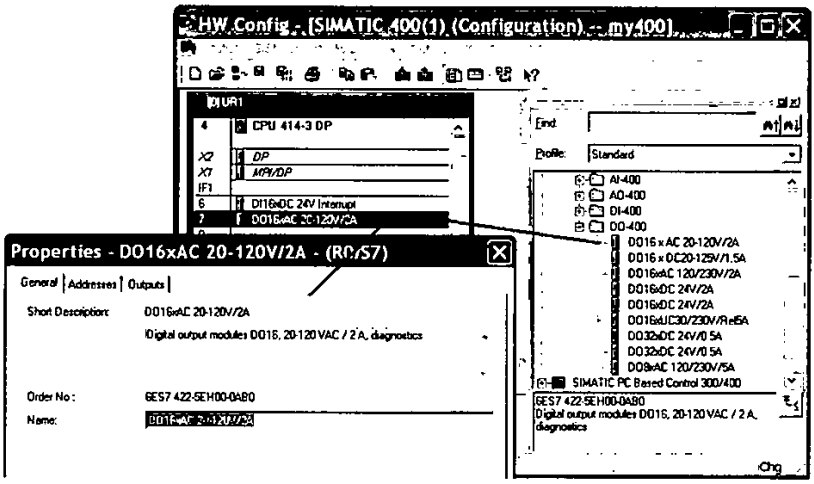


شکل ۱-۵۵

اتصالات سیم‌ها به Front Connector مشابه کارت DI می‌باشد. اما جهت انجام اتصال صحیح باید به کاتالوگ کارت مورد نظر مراجعه نمود. نحوه‌ی انجام اتصالات مربوط به کارت 16 x DC 20-125 V/1.5 A در شکل ۱-۵۵ نشان داده شده است.

پیکربندی کارت‌های DO 400 در Hwconfig

پس از آنکه کارت DO مناسب انتخاب و در رک قرار گرفت، جهت تنظیم مشخصات آن می‌توان بر روی کارت مورد نظر دابل‌کلیک نمود.



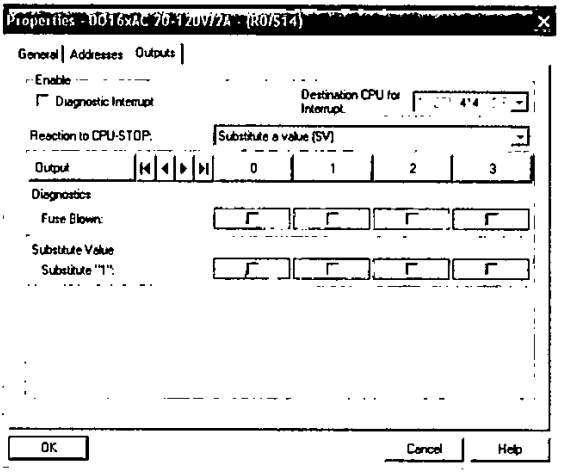
شکل ۱-۵۶

در اینصورت پنجره‌ی تنظیمات کارت باز می‌شود. در این پنجره ۲ یا ۳ سربرگ وجود دارد. این سربرگ‌ها عبارتند از: General , Addresses , Outputs.

الف) سربرگ General: مشابه کارت DI است.

ب) سربرگ Addresses: مشابه کارت DI است.

ج) سربرگ Outputs: در این سربرگ برخی از تنظیمات از جمله موارد مرتبط با Diagnostic Interrupt مثل تشخیص قطعی سیم، اتصال کوتاه و ... قابل انجام می‌باشد. باید توجه نمود که فقط کارت‌هایی که از قابلیت‌های فوق پشتیبانی می‌نمایند، دارای این سر برگ می‌باشند.



شکل ۱-۵۷ کارت DO16xAC 20-120V/2A

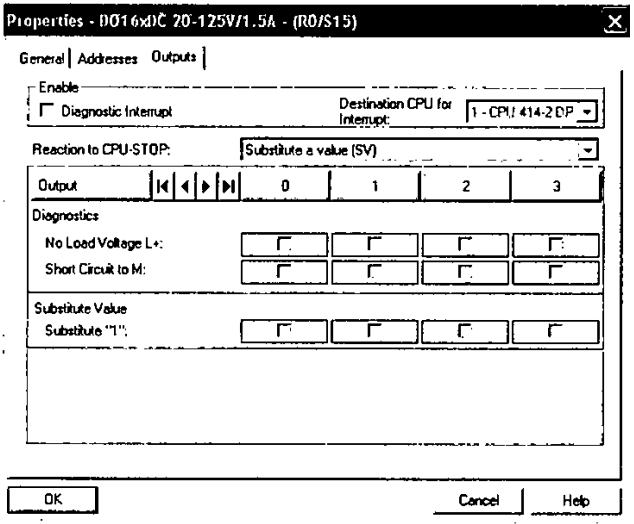
برخی تنظیمات در این پنجره را برای کارت‌های خاص ذکر می‌کنیم.

الف) DO16xAC 20-120V/2A

همانطور که در شکل ۱-۵۷ نشان داده است، این کارت قابلیت تشخیص سوختن فیوز مربوط به هر کانال خروجی را دارا می‌باشد. فعال کردن آن، توسط انتخاب گزینه‌ی Fuse Blown امکان‌پذیر است. همچنین در قسمت Reaction to CPU STOP می‌توان نوع واکنش کارت نسبت به توقف CPU را مشابه موارد گفته شده در کارت‌های DO-300 مشخص نمود.

ب) DO16xDC 20-125V/1.5A

این کارت قطعی تغذیه و اتصالی به زمین (یا ولتاژ منفی منبع تغذیه) را تشخیص می‌دهد. سر برگ Output این کارت در شکل ۱-۵۸ نشان داده شده است.



شکل ۱-۵۸ سربرگ Output کارت DO16xDC 20-125V/1.5A

در قسمت Diagnostic این کارت می‌توان موارد گفته شده را تنظیم نمود. این موارد عبارتند از:

No Load Voltage
با انتخاب این گزینه، قطعی تغذیه‌ی کارت تشخیص داده می‌شود.

Short Circuit to M
با انتخاب این گزینه، اتصال کوتاه سیگنال الکترکی خروجی به زمین تشخیص داده می‌شود.

Diagnostic Interrupt
در صورت وجود یکی از خطاهای فوق، یک OB وقفه راه‌اندازی می‌شود.(OB82)

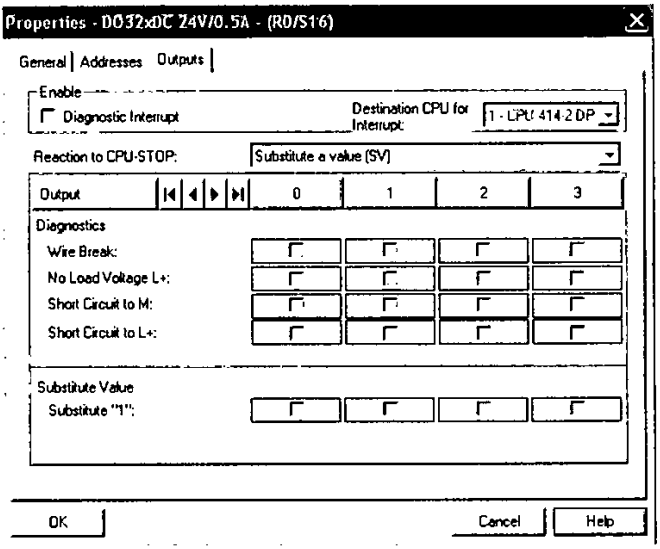
Reaction to CPU STOP
مشابه کارت DO16xAC 20-120V/2A

ج) DO32xDC 24V/0.5A

در این کارت موارد زیر قابل تنظیم است:

- **Wire Break**
  به‌منظور تشخیص قطعی سیم به‌کار می‌رود.
- **Missing Load Voltage L+**
  به‌منظور تشخیص قطعی تغذیه‌ی کارت به‌کار می‌رود.
- **Short Circuit to Ground**
  به‌منظور تشخیص اتصال کوتاه به زمین به‌کار می‌رود.
- **Short Circuit to L+**
  به‌منظور تشخیص اتصال کوتاه به ولتاژ مثبت منبع تغذیه به‌کار می‌رود.
- **Diagnostic Interrupt**
  در صورت وجود یکی از خطاهای فوق، یک OB وقفه راه‌اندازی می‌شود.(OB82)
- **Reaction to CPU STOP**
  مشابه کارت DO16xAC 20-120V/2A

در شکل ۱-۵۹ سر برگ Output در کارت DO32xDC 24V/0.5A نشان داده شده است.



شکل ۱-۵۹ سر برگ Output در کارت DO32xDC 24V/0.5A

۱-۷-۴ کارت‌های AI-400

در S7-400، کارت‌های آنالوگ خروجی نیز نسبت به S7-300 از تنوع کمتری برخوردار می‌باشند. در جدول ۱-۱۹ انواع این کارت‌ها و برخی از مشخصات آنها نشان داده شده است.

جدول۱-۱۹ انواع کارت‌های آنالوگ ورودی S7-400 و برخی از مشخصات آنها

Module / Characteristics	SM 431; AI 8 x 13 Bit (-1KF00-)	SM 431; AI 8 x 14 Bit (-1KF10-)	SM 431; AI 8 x 14 Bit (-1KF20-)	SM 431; AI 13 x 16 Bit (-0HH0-)	SM 431; AI 16 x 16 Bit (-7QH00-)	SM 431; AI 8 x RTD 16 Bit (-7KF10-)	SM 431; AI 8 x 16 Bit (-7KF00-)
Number of Inputs	8 AI U-/I measurem ent 4 AI for resistance measure-ment	8 AI for U/I measure-ment 4 AI for re-sistance/ tempera-ture measur	8 AI for U/I measure-ment 4 AI for re-sistance measure-ment	16 inputs	16 AI for U/ I/tempera-ture mea-surement 8 AI for re-sistance measure-	8 inputs	8 inputs
Resolution	13 bits	14 bits	14 bits	13 bits	16 bits	16 bits	16 bits
Measuring Method	Voltage Current Resistors	Voltage Current Resistors Temperature	Voltage Current Resistors	Voltage Current	Voltage Current Resistors Temperature	Resistors	Voltage Current Temperature
Measuring	Integrating	Integrating	Instantant. value encod	Integrating	Integrating	Integrating	Integrating
Programmable Diagnostics	No	No	No	No	Yes	Yes	Yes
Diagnostic Interrupt	No	No	No	No	Adjustable	Yes	Yes
Limit value Monitoring	No	No	No	No	Adjustable	Adjustable	Adjustable
Hardware Interrupt upon Limit Violation	No	No	No	No	Adjustable	Adjustable	Adjustable
Hardware Interrupt at End of Cycle	No	No	No	No	Adjustable	No	No
Potential Relationships	Analog section isolated from CPU			Non-isolated	Analog section isolated from CPU		
Max. Permissible Common Mode Voltage	Between the channels or between the refer-ence po-tential of the con-nected sensors	Between the chan-nels or between the channel and central ground point: 120 VAC	Between the chan-nels or be-tween the reference potential of the con-nected sensors and MANA:	Between the chan-nels or be-tween the reference potential of the con-nected sensor and central	Between the chan-nels or be-tween the channel and central ground point: 120 VAC	Between channel and central ground point: 120 VAC	Between the chan-nels or be-tween the channel and central ground point: 120 VAC

	and MANA: 30 VAC		8 VAC	ground point: 2 VDC/AC			
Ext. Power Supply Necessary	No	24 VDC (only with current, 2-DMU)	24 VDC (only with current, 2-DMU)	24 VDC (only with current, 2-DMU)	24 VDC (only with current, 2-DMU)	No	No

باید توجه نمود که اصول عملکرد این کارت‌ها با کارت‌های AI-300 یکسان می‌باشد و همه‌ی موارد بیان شده در مورد A/D، تبدیل سیگنال الکتریکی آنالوگ به دیتا، حد تفکیک، سیگنال‌های قابل دریافت توسط کارت AI و ... در مورد کارت‌های S7-400 نیز صدق می‌نماید. بنابراین از ارائه‌ی توضیحات مجدد خودداری می‌شود، خوانندگان گرامی جهت مطالعه‌ی مباحث بیان شده در مورد کارت‌های AI-300 می‌توانند به کتاب مقدماتی مراجعه نمایند.

لامپ‌ها و علائم نشان‌دهنده

کارت‌های آنالوگ ورودی به‌منظور نمایش حالات مختلف کاری و خطاهای پیش آمده جهت کارت دارای لامپ‌های INTF و EXTF می‌باشند. جدول ۱-۲۰ خطاهای پیش آمده و وضعیت این لامپ‌ها را نشان می‌دهد.

جدول ۱-۲۰

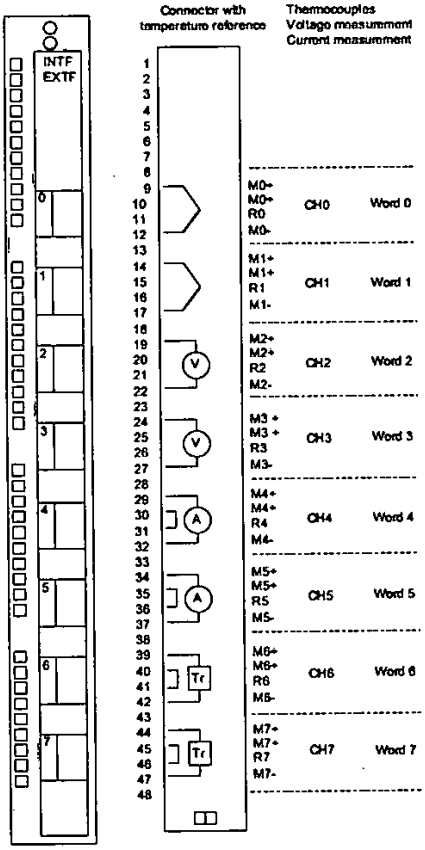
Diagnostic Message	LED	Diagnostics Effective for	Parameters Can Be Assigned
Module problem	INTF/EXTF	Module	No
Internal malfunction	INTF	Module	No
External malfunction	EXTF	Module	No
Channel error present	INTF/EXTF	Module	No
External auxiliary supply missing	EXTF	Module	No
Front connector missing	EXTF	Module	No
Module not configured.	INTF	Module	No
Wrong parameters	INTF	Module	No
Channel information available	INTF/EXTF	Module	No
Coding key incorrect or missing	INTF	Module	No
Thermocouple connection fault	EXTF	Module	No
STOP operating mode	–	Module	No
EPROM error	INTF	Module	No
RAM error	INTF	Module	No
ADC/DAC error	INTF	Module	No
Hardware interrupt lost	INTF	Module	No

Configuring/parameter assignment error	INTF	Channel	No
Short-circuit to M	EXTF	Channel	Yes
Wire break	EXTF	Channel	Yes
Reference channel error	EXTF	Channel	Yes
Underflow	EXTF	Channel	Yes
Overflow	EXTF	Channel	Yes
User connection not wired	EXTF	Channel	No
Open conductor in + direction	EXTF	Channel	No
Open conductor in – direction	EXTF	Channel	No
Run time calibration error	EXTF	Channel	No
Underrange or overrange	EXTF	Channel	No
Open conductor in the current source	EXTF	Channel	No
User calibration doesn't correspond to the parameter assignment	EXTF	Channel	No

اتصالات و مدار داخلی

در شکل ۱-۶۰ نمونه‌ای از نقشه‌ی اتصالات کارت‌های AI نشان داده شده است.

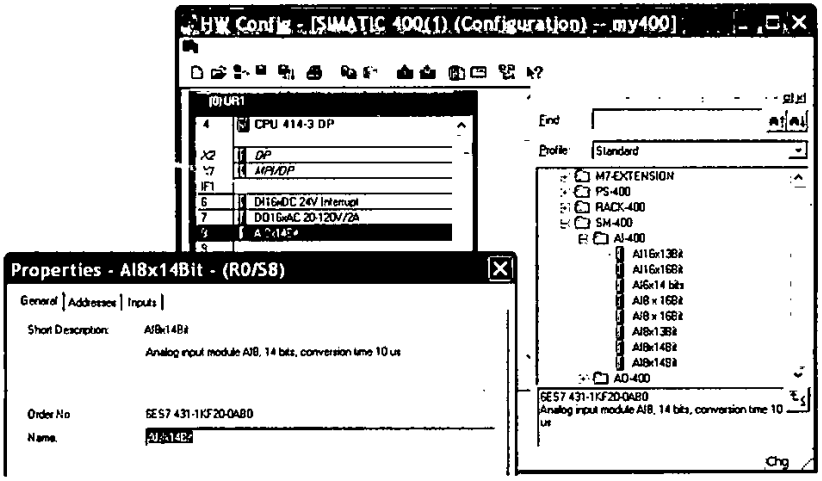
همانطور که مشخص است، در کانال‌های شماره‌ی 0 و 1 از ترموکوپل، کانال‌های شماره‌ی 2 و 3 از ورودی ولتاژی و کانال‌های شماره‌ی 4 و 5 از ورودی جریانی استفاده شده است. همچنین به‌دلیل استفاده از ترموکوپل و انجام جبران‌سازی مربوط به آن، در کانال‌های شماره‌ی 6 و 7 از ترمومتر مرجع استفاده شده است. همانطور که مشخص است و قبلاً نیز در مورد کارت‌های AI-300 بیان شد، به ازای هر کانال ورودی یک Word آدرس از حافظه اختصاص داده خواهد شد و دیتای تبدیل شده توسط A/D درون آن قرار می‌گیرد. کاربر می‌تواند دیتای فوق را پس از مقیاس نمودن به‌صورت یک عدد دیجیتال ۱۶ یا ۳۲ بیتی در برنامه‌ی خود استفاده نماید.



شکل ۱-۶۰ نقشه‌ی اتصالات کارتAI8x16Bit

پیکربندی کارت‌های AI 400 در Hwconfig

جهت وارد نمودن کارت AI-400 در رک، می‌توان از مسیر نشان داده شده در شکل ۱-۶۱ کارت مورد نظر انتخاب نموده و آنرا در اسلات مجاز در رک قرار داد.



شکل ۱-۶۱ انتخاب کارت AI-400 از کاتالوگ برنامه‌ی HW Config

تنظیم پارامترهای کارت AI

تقریباً اکثر موارد گفته شده در مورد تنظیم پارامترهای کارت‌های AI-300 در مورد کارت‌های AI-400 نیز صدق می‌نمایند. از اینرو فقط برخی از تفاوت‌های تنظیم پارامترها ذکر می‌گردد.

کارت AI16x16Bit با شماره‌ی فنی ES7 431-7QH00-0AB0۶

در این کارت تنظیمات جدید موجود در سربرگ Input Part1 عبارتست از:

- **Reference temperature**

در این کادر می‌توان دمای مرجع که اتصال کابل انتقال به ترموکوپل در آن انجام شده است را وارد نمود.

- **Reference channel error**

در مواردی که از ترموکوپل استفاده شود، وجود خطاهایی مثل قطعی سیم در ترمومتر مرجع را تشخیص می‌دهد.

- **Underflow**

اگر مقدار ورودی از عدد 8100 در کد هگز کمتر شود، خطای Underflow گزارش می‌شود.

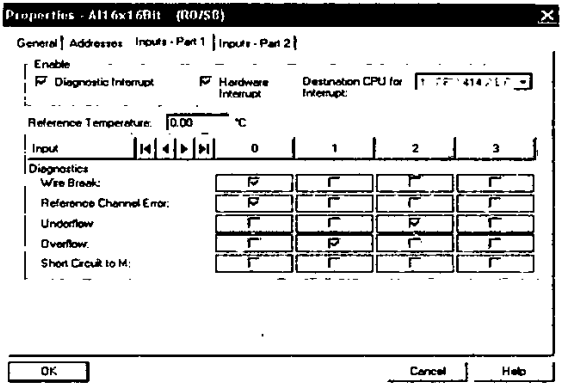
- **Overflow**

اگر مقدار ورودی از عدد 7FFF در کد هگز بیشتر شود، خطای Overflow گزارش می‌شود.

- **Short circuit to ground**

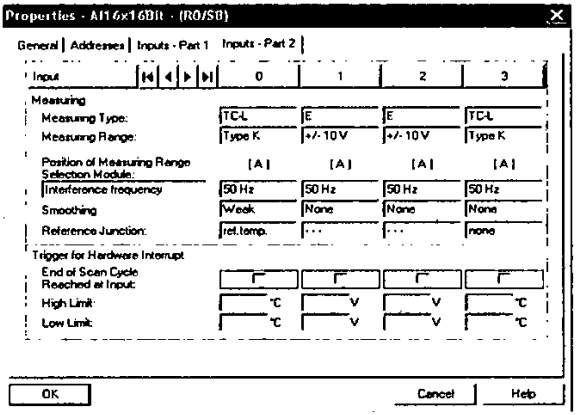
اگر اتصال سیم‌های ورودی به زمین اتفاق افتد خطای اتصال کوتاه به زمین گزارش داده می‌شود.

در شکل ۱-۶۲ سربرگ Input Part1 کارت فوق نشان داده شده است.



شکل ۱-۶۲ سربرگ Input Part 1 کارت AI 16x16Bit

در شکل ۱-۶۳ سربرگ Input Part 2 نشان داده شده است. اکثر گزینه‌های موجود در این سربرگ قبلاً توضیح داده شده‌اند. یکی از گزینه‌هایی که شکل آن عوض شده است، گزینه‌ی End of cycle reached on input می‌باشدکه با انتخاب آن در پایان سیکل تبدیل سیگنال به دیتا یک وقفه سخت‌افزاری به CPU اعمال می‌شود.



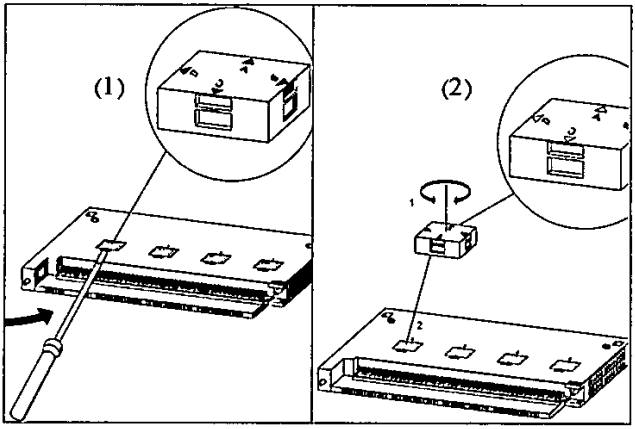
شکل ۱-۶۳ سر برگ Input Part 2 کارت AI 16x16Bit

تنظیمات سایر کارت‌ها

تنظیمات سایر کارت‌ها مشابه موارد بیان شده تاکنون است و تنظیم جدیدی در آنها وجود ندارد. در صورتی‌که خواننده در تنظیم آنها دچار اشکال شد، می‌تواند به موارد بیان شده در مورد AI-300 و موارد اخیر بیان شده در مورد AI-400 مراجعه نماید.

تنظیم سخت‌افزاری کارت AI

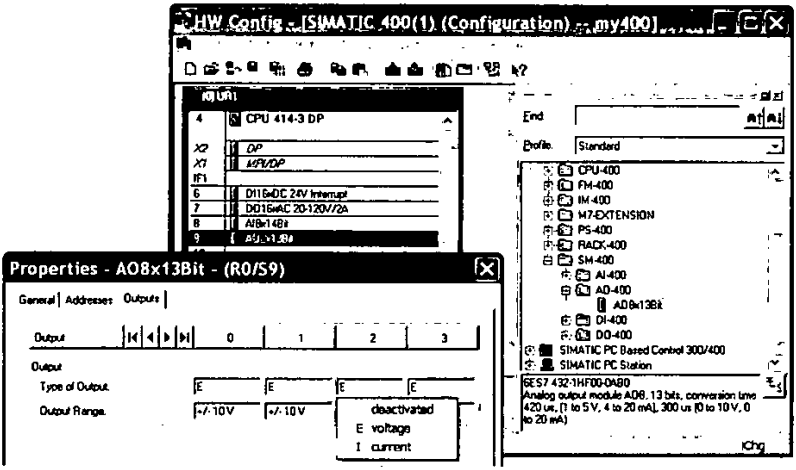
مشابه کارت‌های AI-300، در برخی از کارت‌های AI-400 علاوه بر تنظیم نرم‌افزاری که توسط برنامه‌ی HW Config انجام می‌پذیرد، نیاز به انجام تنظیم سخت‌افزاری نیز می‌باشد. این تنظیم با تعیین وضعیت Range Selection Module انجام می‌پذیرد. مراحل انجام این عمل در شکل ۱-۶۴ نشان داده شده است.



شکل ۱-۶۴ مراحل انجام تنظیم Range Selection Module

۱-۷-۵ کارت AO-400

دیتای ارسالی توسط CPU، در کارت‌های آنالوگ خروجی به سیگنال الکتریکی آنالوگ تبدیل شده و به تجهیزاتی آنالوگ مانند کنترل ولو، درایو و ... ارسال می‌گردد. در این کارت‌ها به‌منظور تبدیل دیتا به سیگنال الکتریکی، از واحدی به نام D/A استفاده می‌گردد. موارد بیان شده در مورد AO-300 در مورد AO-400 نیز صدق می‌نماید.



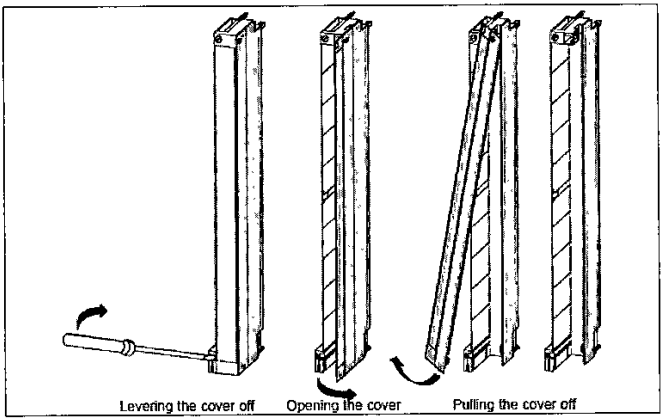
شکل ۱-۶۵

همانطور که در شکل ۱-۶۵ مشاهده می‌شود، در S7-400 فقط یک نوع کارت AO وجود دارد که می‌تواند خروجی ولتاژی و جریانی تحویل دهد. این کارت از نوع معمولی است و همانطور که در شکل ۱-۶۵ مشاهده می‌شود، هیچ تنظیم خاصی ندارد.

۱-۷-۶ نصب و سیم‌کشی کارت‌های SM

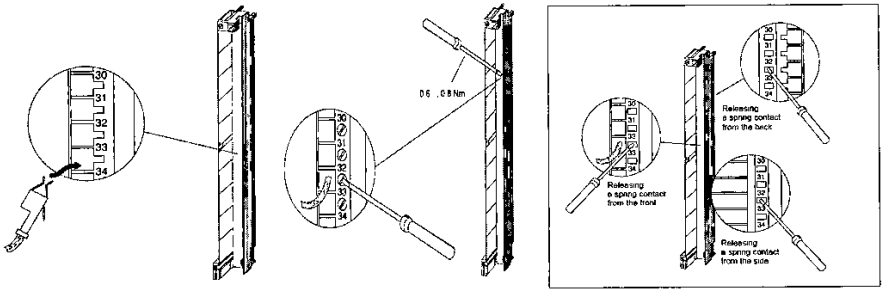
از آنجا که نصب و اتصالات کارت های SM400 مشابه است موضوع در همین قسمت تشریح می‌گردد. همه انواع کارت‌های ورودی و خروجی خانواده S7-400 فقط یک اسلات را روی رک اشغال می‌کنند. روش نصب همه آنها مشابه نصب CPU است که قبلاً ذکر شد.

به‌منظور اتصال سیم‌ها به کارت از Front Connector استفاده می‌شود. کانکتور جلویی دارای انواع سوکتی، پیچی و فنری است. در هر سه حالت برای شروع لازم است روکش آنرا مانند شکل ۱-۶۶ خارج نمود.



شکل ۱-۶۶ خارج نمودن درپوش Front Connector

پس از آن باید سیم را به اندازه کافی لخت کرده و مطابق شکل ۱-۶۷ به کانکتور متصل نمود.

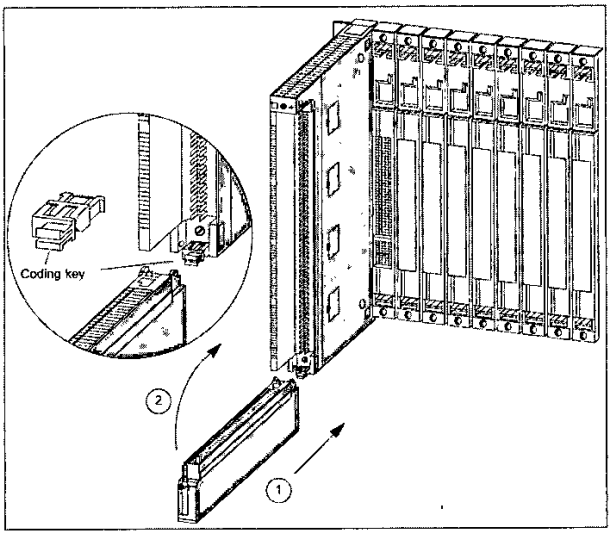


Front Connector سوکتی **Front Connector پیچی** **Front Connector فنری**

شکل۱-۶۷ نحوه اتصال به Front Connector

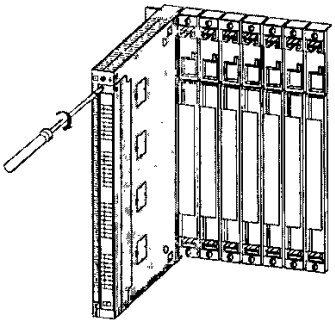
نصب Front Connector روی کارت

به‌منظور نصب صحیح Front Connector روی کارت‌های ورودی و خروجی، بر روی این کارت‌ها از یک سوکت به نام Coding Key استفاده می‌شود. لازم است که Front Connector به‌طور صحیح (مطابق شکل ۱-۶۸) به آن اتصال یابد.



شکل ۱-۶۸ اتصال Front Connector به کارت DI

پس از اتصال Front Connector به کارت، لازم است مطابق شکل ۱-۶۹ پیچ آنرا محکم نمود.



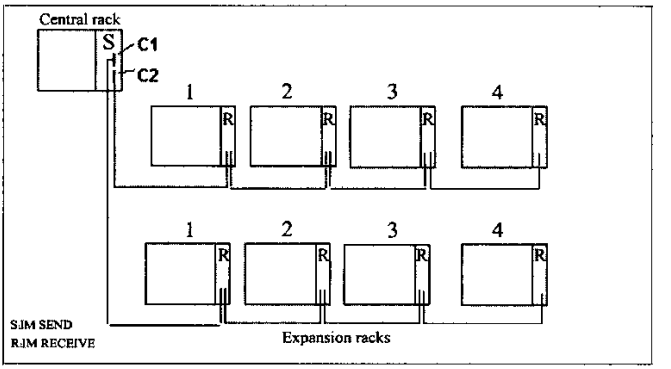
شکل ۱-۶۹

۱-۸ ماژول رابط بین رک‌ها[1]

۱-۸-۱ عملکرد

همانطور که در کتاب مقدماتی در پیکربندی S7-300 نیز بیان شد، به منظور ارتباط بین رک اصلی و رک‌های توسعه از ماژولی به‌نام IM استفاده می‌شود. در S7-400 چهار جفتIM وجود دارد.

هر جفتIM از دو IM تشکیل شده است که یک IM از نوع Send و یک IM از نوع Receive می‌باشد. در S7-400، هر IM از نوع Send دارای دو پورت ارتباطی به نام‌های C1 و C2 است. نحوه‌ی اتصال بدین صورت است که یک IM از نوع Send در رک اصلی قرار می‌گیرد. بسته به نوع IM Send حداکثر می‌توان به هر پورت آن چهار IM از نوع Receive که هر کدام در یک رک توسعه قرار گرفته است متصل نمود. همانطور که در شکل ۱-۷۰ملاحظه می‌شود، اتصال IMها به صورت زنجیری (سری) می‌باشد.



شکل ۱-۷۰ ارتباط زنجیری IMها

جدول ۱-۲۱، IMهای موجود در S7-400 و برخی از خصوصیات آنها را نشان می‌دهد.

جدول۱-۲۱ انواع جفتIM های S7-400

	Local connection		Remote connection	
Send IM	460-0	460-1	460-3	460-4
Receive IM	461-0	461-1	461-3	461-4
Max. number of connectable EMs per chain	4	1	4	4
Max. distance	5 m	1.5 m	102.25 m	605 m
5 V transfer	No	Yes	No	No

1. Interface Module

Max. current transfer per interface	–	5 A	–	–
Communication bus transmission	Yes	No	Yes	No

در جدول ۱-۲۱، هر ستون بیانگر یک جفتIM می‌باشد. همانطور که مشخص است، چهار ستون وجود دارد که بیانگر وجود چهار نوع جفتIM است. سطرهای جدول بیانگر خصوصیات IMها می‌باشند:

سطر اول: مشخص می‌کند که IM مورد نظر از نوع Send است.

سطر دوم: مشخص می‌کند که IM مورد نظر از نوع Receive است.

سطر سوم: حداکثر تعداد IM از نوع Receive به‌صورت اتصال زنجیری به هر پورت IM Send

سطر چهارم: حداکثر فاصله‌ی بین رک اصلی و توسعه (طول کابل IM)

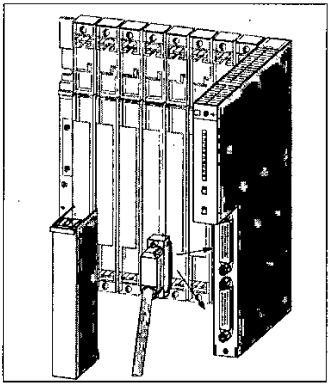
سطر پنجم: انتقال ولتاژ تغذیه‌ی توسط IM

سطر ششم: حداکثر جریان انتقالی توسط IM در صورت انتقال تغذیه

سطر هفتم: انتقال دیتاهای مربوط به C Bus

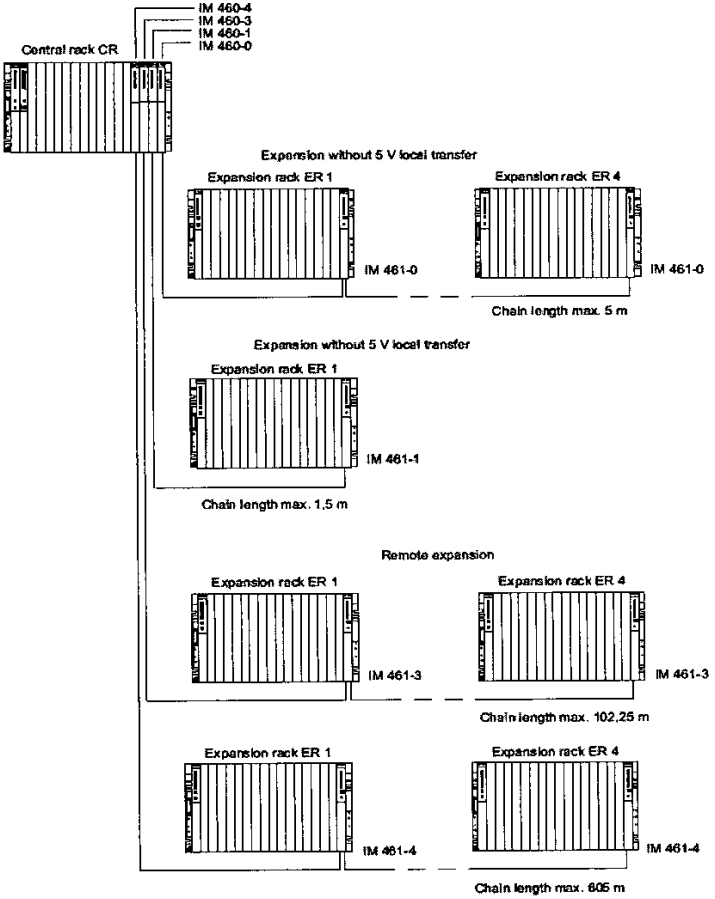
ارتباط IMها با یکدیگر

همانطور که اشاره شد، IM Send در رک اصلی قرار گرفته و ارتباط آن به‌صورت زنجیری با IMهای Receive قرار گرفته در رک توسعه برقرار می‌گردد. جهت اتصال کابل IM Receive به IM Send می‌توان مطابق شکل ۱-۷۱ درپوش IM را برداشته و کابل را به پورت مورد نظر اتصال داد.



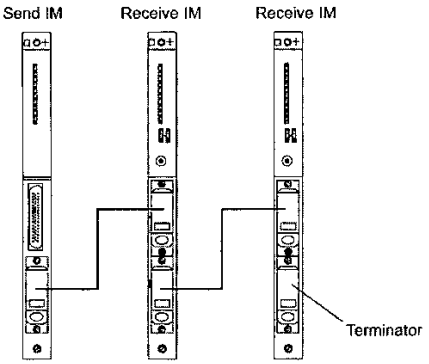
شکل ۱-۷۱ اتصال کابل IM

ممکن است در رک اصلی چند نوع IM از نوع Send قرار گیرند، در اینصورت هر IM Send فقط می‌تواند با جفت خود از نوع Receive ارتباط برقرار نماید. شکل ۱-۷۲ این موضوع را نشان می‌دهد.



شکل ۱-۷۲ ارتباط IMهای Send با جفتIM خود

هر IM از نوع Receive دارای یک ترمینیتور می‌باشد (به استثنای IM 461-1) که لازم است جهت آخرین IM در هر ارتباط زنجیری، ترمینیتور را در پایین‌ترین وضعیت آن قرار داد. در شکل ۱-۷۳ این موضوع نشان داده شده است.



شکل ۱-۷۳ تنظیم ترمینیتور آخرین IM Receive در ارتباط زنجیری

نکات استفاده از IM

برخی از نکات مهم استفاده از IM در پیکربندی S7-400عبارتست از:

- می‌توان حداکثر ۲۱ رک توسعه قرار داد.
- هر رک دارای یک شماره‌ی اختصاصی بین ۱ الی ۲۱ می‌باشد که توسط IM قرار گرفته در رک تنظیم می‌شود. شماره‌ی اختصاص داده شده به رک نباید تکراری باشد.
- در رک اصلی حداکثر می‌توان از شش عدد IM Send می‌توان استفاده نمود. البته فقط دو عدد IM از نوعی که تغذیه را منتقل می‌نمایند، می‌توان در رک اصلی قرار داد.
- ارتباط بین IMها زنجیری (سری) می‌باشد.
- تبادل دیتای C Bus فقط تا هفت رک امکان‌پذیر است. یعنی رک اصلی و رک‌های توسعه از شماره‌ی ۱ تا ۶
- طول کابل IM ازحداکثر طول مجازآن‌بیشتر نشود. حداکثر طول مجاز کابل IM در جدول ۱-۲۲ نشان داده شده است.

جدول ۱-۲۲ حداکثر طول مجاز کابل IM

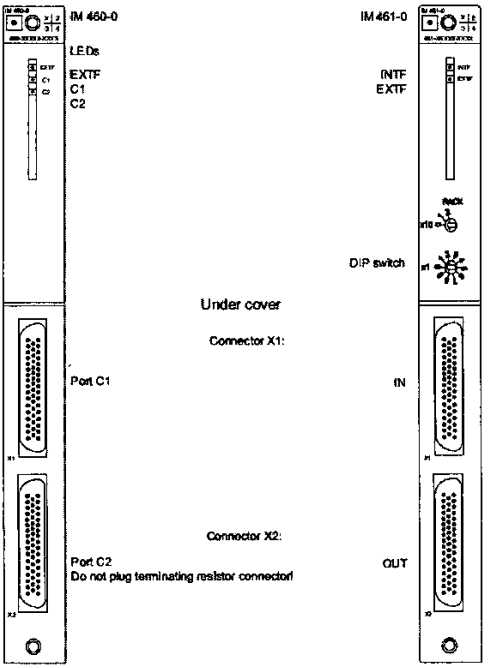
Connection type	Maximum (total) line length
Local connection with 5 V transfer via IM 460-1 and IM 461-1	1.5 m
Local connection without 5 V transfer via IM 460-0 and IM 461-0	5 m
Remote connection via IM 460-3 and IM 461-3	102.25 m
Remote connection via IM 460-4 and IM 461-4	605 m

۱-۸-۲ انواع جفت IMهای S7-400

الف) IM 460-0 و IM 461-0

این جفت IM جهت ارتباط محلی به‌کار می‌رود. همانطور که در شکل ۱-۷۴ ملاحظه می‌شود، IM 460-0 دارای دو پورت C1و C2 می‌باشد که به هر کدام از آنها می‌توان حداکثر چهار عدد IM 461-0 را متصل نمود. IM 461-0 نیز دارای دو

پورت به نام‌های IN و Out می‌باشد. در ارتباط زنجیری کابل رک اصلی به پورت IN وارد شده و از پورت Out جهت اتصال به IM بعدی استفاده می‌شود.



شکل ۱-۷۴ نمای شماتیک IM 460-0 و IM 461-0

لامپ‌ها و علائم

همانطور که در شکل ۱-۷۴ ملاحظه شد، هر کدام از IMهای 460-0 و 461-0 دارای لامپ‌های نشان دهنده‌ای می‌باشند. این لامپ‌ها نشان‌دهنده وضعیت کار IM و خطاهای احتمالی پیش‌آمده در زمان کار IM می‌باشند. شرح این لامپ‌ها و کانکتورهای X1 و X2 در جدول ۱-۲۳ ارائه شده است. باید توجه نمود که مشابه این لامپ‌ها و علائم در سایر IM ها نیز وجود دارد که تقریباً عملکرد مشابه‌ای دارند.

جدول ۱-۲۳

لامپ	مفهوم
INTF (قرمز)	در یکی از دو حالت زیر روشن می‌شود: • شماره‌ی رک تنظیم‌شده از ۲۱ بزرگتر باشد. • شماره‌ی رک در حالت کار عوض شود.
EXTF (قرمز)	در صورتی‌که یک خطای خارجی مثل قطعی سیم یا تنظیم نکردن ترمینیتور برای IMهای متصل به پورت C1 یا C2 پیش آید روشن می‌شود.

C1 (سبز)	اتصال کابل توسط پورت C1 به‌صورت صحیح انجام شده است.
C1 (سبز چشمک‌زن)	پورت C1 آماده‌ی عملکرد نمی‌باشد و راه اندازی خود را کامل نکرده است.
C2 (سبز)	اتصال کابل توسط پورت C2 به‌صورت صحیح انجام شده است.
C2 (سبز چشمک‌زن)	پورت C2 آماده‌ی عملکرد نمی‌باشد و راه اندازی خود را کامل نکرده است.
پورت‌ها	
X1	پورت C1
X2	پورت C2

DIP switch

به‌منظور تنظیم شماره‌ی رک، از دو عدد DIP switch که روی IM 461-0 قرار گرفته است، استفاده می‌شود. این DIP switch در IMهای Receive وجود دارد. سوئیچ X1 جهت تنظیم رقم یکان و سوئیچ X10 جهت تنظیم رقم دهگان از شماره‌ی رک استفاده می‌شود. مثلاً جهت ساخت شماره‌ی ۱۸ می‌توان سوئیچ X1 را روی عدد ۸ و سوئیچ X10 را روی عدد ۱ تنظیم نمود.

مشخصات فنی

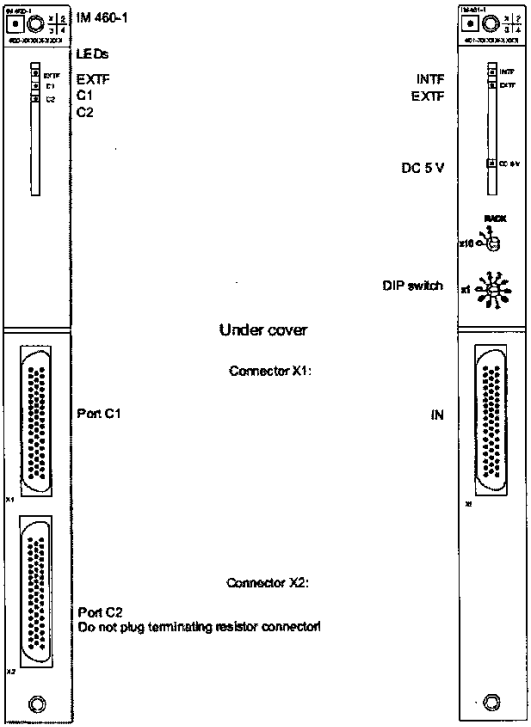
در جدول ۱-۲۴ مشخصات فنی IM 460-0 و IM 461-0 نشان داده شده است.

جدول ۱-۲۴ مشخصات فنی IM 460-0 و IM 461-0

Maximum line length (total)	5 m
Dimensions W x H x D (mm)	25 x 290 x 280
Weight • IM 460-0 • IM 461-0	600 g 610 g
Current consumption from the S7-400 bus 5 VDC • IM 460-0 • IM 461-0	Typ. 130 mA Max. 140 mA Typ. 260 mA Max. 290 mA
Power loss • IM 460-0 • IM 461-0	 Typ. 650 mW Max. 700 mW Typ. 1300 mW Max. 1450 mW
Terminator	6ES7461-0AA00-7AA0
Backup current	No

الف) IM 460-1 و IM 461-1

همانطور که در شکل ۱-۷۵ ملاحظه می‌شود، IM 460-1 دارای دو پورت به نام‌های C1 و C2 است ولی IM 461-1 دارای یک پورت می‌باشد. در صورت استفاده از این IMها به هرکدام از پورت‌های C1 و C2 فقط می‌توان یک IM متصل نمود. از جمله خصوصیات این IMها این است که تغذیه توسط خود IM منتقل شده و در رک توسعه نیازی به قرار دادن منبع تغذیه نمی‌باشد. از جمله نکات جدیدی که در این IMها به چشم می‌خورد، وجود یک لامپ به نام DC 5V روی IM 461-1 می‌باشد که نشان‌دهنده برقراری ولتاژ منتقل شده توسط IM است.



شکل ۱-۷۵ نمای شماتیک IM 460-1 و IM 461-1

لامپ‌ها و علائم

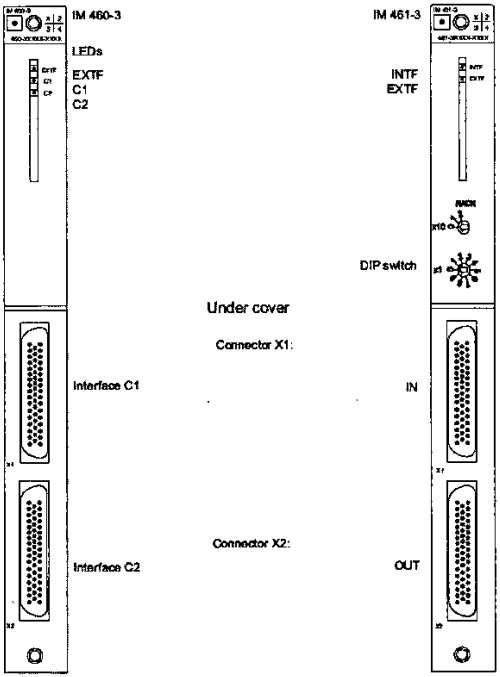
لامپ‌ها، علائم و پورت‌های موجود در IM460-1 مشابه لامپ‌ها و علائم به‌کار رفته در IM460-0 می‌باشد، لذا از توضیح مجدد آن خودداری می‌شود.

لامپ‌ها و پورت‌های IM461-1 مشابه IM460-1 است، با این تفاوت که:

- در IM 461-1 فقط یک پورت وجود دارد.
- در IM 461-1 یک لامپ به‌نام DC 5V وجود دارد که در صورت روشن شدن (رنگ سبز) نشان‌دهنده برقراری ولتاژ تغذیه در رک می‌باشد.

ج) IM 460-3 و IM 461-3

از این IMها جهت ارتباط دوردست تا حداکثر فاصله‌ی 102.25 m استفاده می‌شود. همانطور که در شکل ۱-۷۶ ملاحظه می‌شود، IM 460-3 دارای دو پورت ارتباطی می‌باشد که به هر کدام از پورت‌های آن می‌توان حداکثر چهار IM به‌صورت ارتباط زنجیری (سری) متصل نمود.



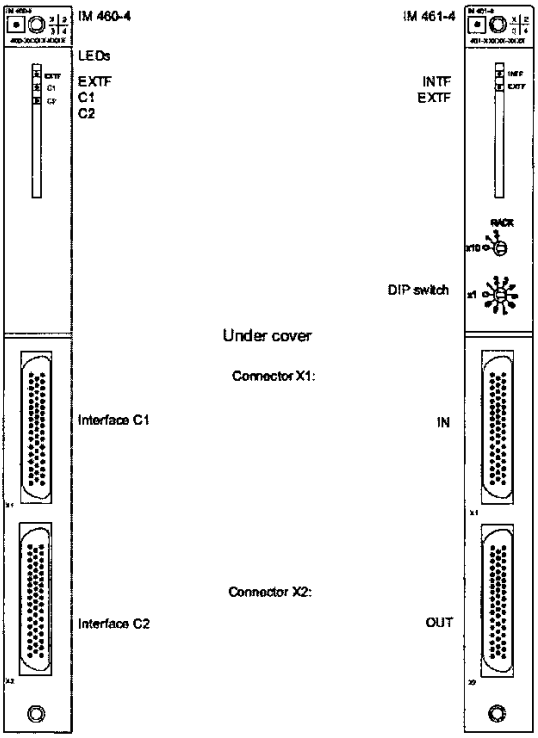
شکل۱-۷۶ IM 460-3 و IM 461-3

لامپ‌ها و علائم

به لحاظ ظاهری این IMها مشابه IM 460-0 و IM461-0 است و لامپ‌ها و علائم آنها نیز یکسان می‌باشد، لذا از ارائه‌ی توضیحات آن خودداری می‌شود.

د) IM 460-4 و IM 461-4

این جفتIM جهت ارتباط دور دست تا حداکثر فاصله‌ی 605 m به‌کار می‌روند. همانطور که در شکل ۱-۷۷ مشخص است، IM 461-4 دارای دو پورت به‌نام‌های C1 و C2 است که به هر کدام از آنها می‌توان حداکثر چهار IM به‌صورت زنجیری متصل نمود.



شکل ۱-۷۷ نمای شماتیک IM 460-4 و IM 461-4

لامپ‌ها و علائم

به لحاظ ظاهری این IMها نیز مشابه IM 460-0 و IM461-0 است و لامپ‌ها و علائم آنها نیز یکسان می‌باشد.

تذکر: IM460-4 و IM461-4 را با CPUهایی که کد آنها به‌صورت زیر است نمی‌توان به‌کار برد.

- 6ES7412-1XF00-0AB0
- 6ES7413-1XG00-0AB0
- 6ES7413-2XG00-0AB0
- 6ES7414-1XG00-0AB0
- 6ES7414-2XG00-0AB0
- 6ES7416-1XJ00-0AB0

سایر IM ها

الف) IM467

علاوه بر IM های گفته شده چند گروه IM دیگر نیز وجود دارند. موارد کاربرد این IMها عبارتست از:

- **IM 467:** جهت ارتباط با شبکه‌ی Profibus DP

- **IM 467 FO:** جهت ارتباط با شبکه‌ی Profibus DP

در شکل ۱-۷۸ یک IM 467 نشان داده شده است. IM 467 FO نیز به لحاظ ظاهری مشابه IM 467 است ولی می‌تواند از طریق فیبر نوری نیز با شبکه ارتباط برقرار نماید. این IM در حال حاضر تولید نمی‌شود.

شکل ۱-۷۸

ب) IM463-2

جهت ارتباط واحدهای توسعه S5 با S7-400 به‌کار می‌رود.

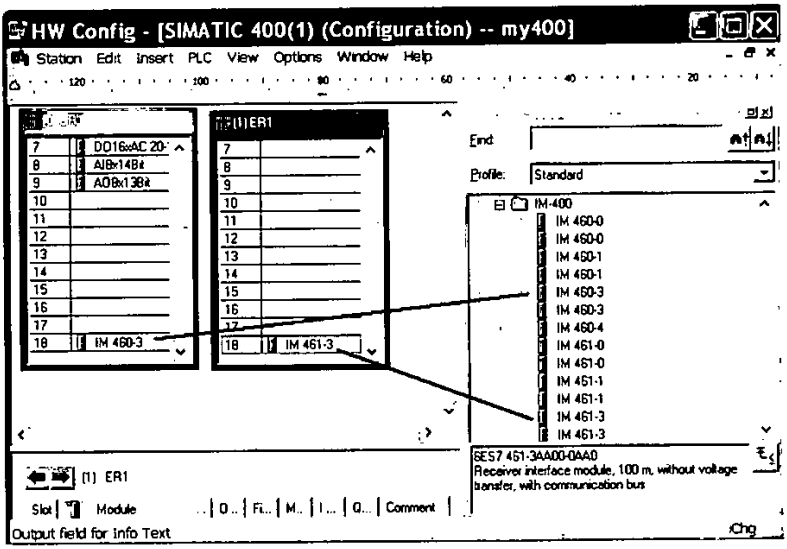
کارت‌های ورودی و خروجی S5 که روی رک S5 قرار دارند را می‌توان به S7-400 متصل کرد. در این حالت در سمت S7-400 کارت IM463-2 و در سمت S5 کارت IM314 نصب می‌گردد. طول کابل بین این دو IM ماکزیمم ۶۰۰ متر است.

شکل ۱-۷۹ IM463-2

۱-۸-۳ پیکربندی IM در Hwconfig

برای قرار دادن IM در رک می‌توان مطابق روش زیر عمل نمود:

۱- ابتدا رک اصلی از نوع UR1 و رک توسعه‌ی مورد نظر را وارد نمایید.
۲- پس از قرار دادن PS و CPU در رک اصلی، مطابق شکل ۱-۸۰ IM 460-3 را انتخاب نموده آنرا در رک اصلی وارد نمایید. IMهای Send را می‌توان بعد از سایر کارت‌ها در هر اسلات دلخواه قرار داد.

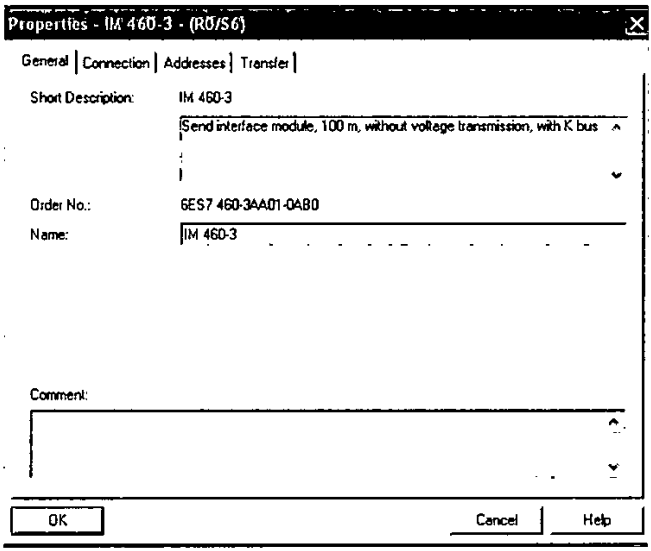


شکل ۱-۸۰ انتخاب IM در محیط برنامه‌ی HW Config

۳- در رک توسعه متناسب با IM که در رک اصلی قرار گرفته‌است، IM مناسب را انتخاب نمایید. در اینجا چون در رک اصلی از IM 460-3 استفاده شده است، در رک توسعه IM 461-3 انتخاب می‌گردد. باید توجه نمود که IM در رک توسعه در اسلات آخر قرار می‌گیرد.
۴- با قرار دادن IM ها ارتباط بین رک‌ها برقرار نمی‌شود و نیاز به تنظیم دیگری است که در ادامه توضیح داده خواهد شد.

تنظیمات IMها در برنامه‌ی HW Config

هنگامی‌که IM ها در رک اصلی و رک‌های توسعه قرار گرفتند، ارتباط بین آنها به‌طور اتوماتیک برقرار نمی‌شود. در این حالت لازم است کاربر به‌صورت دستی در محیط برنامه‌ی HW Config ارتباط بین IMها را معرفی نماید. برای این منظور می‌توان بر روی IMای که در رک اصلی قرار گرفته است دوبار کلیک نمود.
در اینصورت پنجره‌ی مشخصات IM باز می‌شود. این پنجره برای IM 460-3 در شکل ۱-۸۱ نشان داده شده است.



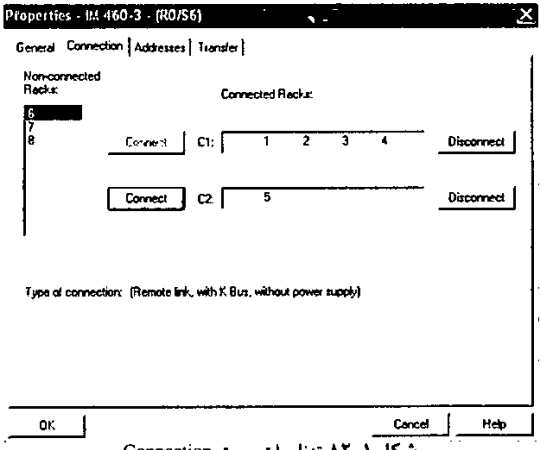
شکل ۱-۸۱ سربرگ General از پنجره‌ی مشخصات IM 460-3

سر برگ‌های موجود در این پنجره عبارتند از:

General: در این سر برگ، توضیحات کلی در مورد IM ارائه شده است.

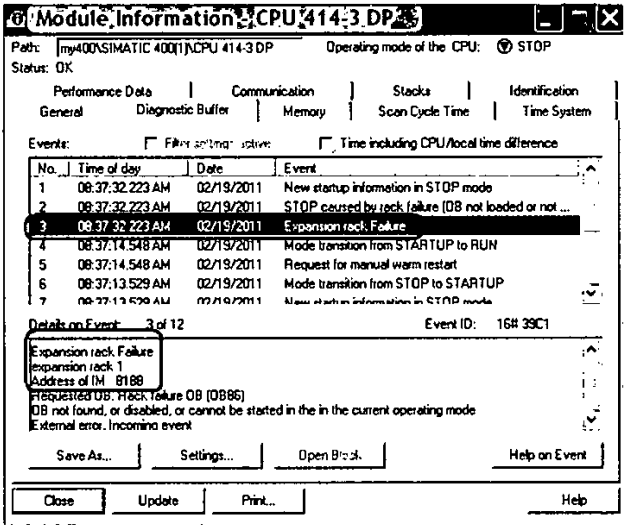
Connection: در این قسمت می‌توان اتصال مجازی بین IMها را برقرار نمود.

همانطور که در شکل ۱-۸۲ ملاحظه می‌شود، در قسمت Non-Connected Racks، شماره‌ی رک‌هایی که قابلیت اتصال به این IM را دارند نشان داده شده است. هر کدام از پورت‌های C1 و C2 داری دکمه‌ای به نام Connect جهت انتخاب رک مورد نظر جهت اتصال به این پورت و دکمه‌ی Disconnect جهت قطع اتصال IMهای متصل شده به هر پورت می‌باشند.



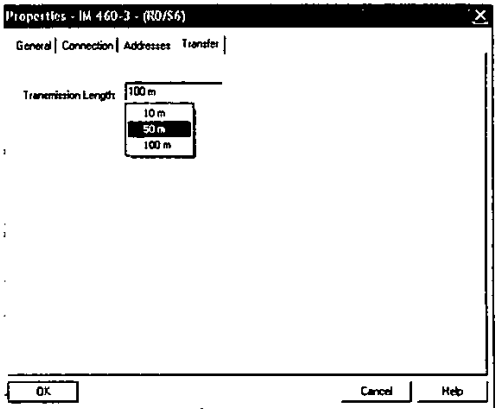
شکل ۱-۸۲ تنظیمات پورت Connection

Addresses: این آدرس به‌عنوان آدرس Diagnostics استفاده می‌شود. در صورت بروز مشکل ارتباطی اطلاعات خطا همراه با این آدرس به CPU اعلام می‌شود. شکل ۱-۸۳ بافر CPU را در شرایط بروز فالت روی IM موجود در رک شماره ۱ نشان می‌دهد.



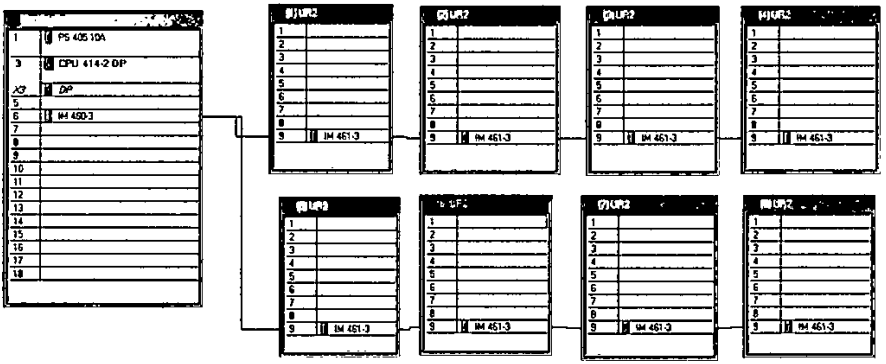
شکل ۱-۸۳

Transfer: در این قسمت می‌توان طول کابل IM را مشخص نمود.



شکل ۱-۸۴

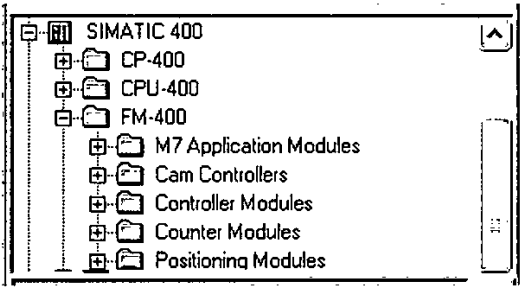
در شکل ۱-۸۵ نمایی از یک سیستم S7-400 را مشاهده می نمایید. در این سیستم به هر کدام از پورت‌های IM 460-3 چهار عدد IM 461-3 متصل شده است.



شکل ۱-۸۵

Function Module (FM-400) ۹-۱

مشابه FMهای S7-300، در S7-400 نیز تعدادی FM به‌منظور انجام وظایف خاص در نظر گرفته شده‌اند. این کارت‌ها در کاتالوگ به‌صورت شکل ۱-۸۶ دیده می‌شوند.



شکل ۱-۸۶ Function Module

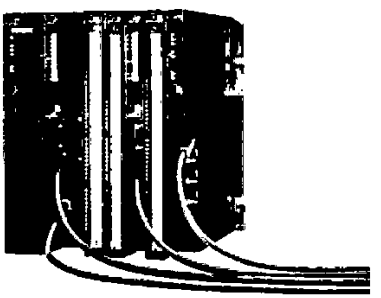
معرفی کلی این FMها در جدول ۱-۲۵ آمده است. تشریح عملکرد آنها در کتاب جداگانه‌ای آورده خواهد شد.

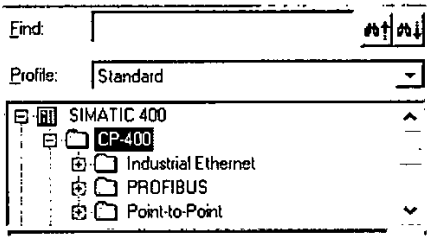
جدول ۱- ۲۵

	کاربرد	تعداد کانال
FM 450-1	Counting, measuring, proportioning position detection (incremental)	2
FM 452	Cam control	1
FM 455C	PID-Control (continuous)	16
FM 455S	PID-Control (step/impulse)	16
FM 451	Positioning (rapid traverse/creep feed)	3
FM 453	Positioning (with stepper and servo drives)	3
FM 458-1 DP	Counting, measuring, Cam control, Closed loop control, Motion Control	اختیاری

۱-۱۰ کارت شبکه (CP-400)

Communication Processors برای ارتباط PLC با شبکه استفاده می‌شوند.





شکل ۱-۸۷ کارت‌های شبکه S7-400

کارت‌های شبکه قابل استفاده عبارتند از:

- Industrial Ethernet
- Profibus
- Point to Point (مدباس)

برخلاف S7-300 در اینجا کارتی برای اتصال به شبکه ASI وجود ندارد.

بررسی و تشریح شبکه‌ها در کتاب‌های جداگانه‌ای آورده شده است.

۱-۱۱ پرسش و تحقیق

با توجه به توقف تولید IM467، چه ماژولی جایگزین آن شده و چه تفاوت‌های عملکردی با ماژول قبلی دارد؟

۱-۱۲ تست‌های خود آزمایی

۱- در مورد پیکربندی S7-400 کدام گزینه صحیح نمی‌باشد؟

الف) در S7-400 رک نقش نگهداری ماژول‌ها را دارد.

ب) در S7-400 رک نقش ارتباط ماژول‌ها را برعهده دارد.

ج) در S7-400 چندین تنوع رک وجود دارد.

د) در S7-400 فضای خالی روی رک بین ماژول‌ها مجاز نمی‌باشد.

۲- کدام‌یک از باس‌های زیر جهت ارتباط I/O ها با CPU استفاده می‌شود؟

الف) P Bus ب) C Bus ج) K bus د) هیچ‌کدام

۳- در رک‌های ER1 و ER2 کدام باس وجود ندارد؟

الف) P Bus ب) I/O Bus ج) C Bus د) هر سه مورد

۴- کدام‌یک از رک‌های زیر می‌تواند به‌عنوان رک اصلی و توسعه مورد استفاده قرار گیرد؟

الف) UR2 ب) CR2 ج) ER1 د) CR3

۵- کدام‌یک از رک‌های زیر فقط به‌عنوان رک توسعه می‌تواند مورد استفاده قرار گیرد؟

الف) UR2 ب) CR2 ج) ER1 د) CR3

۶- در S7-400 اسلات‌های مجاز جهت قرار گیری PS کدام‌یک می‌باشد؟

الف) فقط اسلات ۱ ب) فقط اسلات ۲ ج) فقط اسلات ۳ د) اسلات ۱ الی ۴

۷- در منبع تغذیه PS 407 10A حداکثر قابلیت جریان‌دهی چقدر است؟

الف) 4A ب) 10A ج) 407 A د) هیچ‌کدام

۸- اگر دو منبع تغذیه 10A به‌صورت افزونه استفاده شوند، حداکثر جریان تحویلی آنها چقدر است؟

الف) 10A ب) 15A ج) 20A د) نامشخص

۹- نقش باتری پشتیبان که روی PS نصب می‌شود چیست؟

الف) تأمین ولتاژ منبع تغذیه ب) تأمین ولتاژ مورد نیاز CPU

ج)جلوگیری از پاک شدن RAM د) هیچ‌کدام

۱۰- در مورد CPUهای S7-400 کدام مورد صحیح است؟

الف) مدل یکپارچه ندارند. ب) مدل IFM ندارد.

ج) اکثر آنها پورت شبکه دارند. د) هر سه مورد

۱۱- در CPUهای دارای پورت قابل تنظیم، چگونه می‌توان نوع پورت را تنظیم نمود؟

الف) به‌صورت نرم‌افزاری ب) به صورت سخت‌افزاری ج) موارد الف و ب د) هیچ‌کدام

۱۲- در کدام نوع از CPUهای زیر وجود ماژول سنکرون‌ساز ضروری می‌باشد؟

الف) CPUهای استاندارد ب)CPUهای سری F ج) CPUهای سری H د) هیچ‌کدام

۱۳- Redundancy را در کدامیک از تجهیزات زیر می‌توان به‌کار برد؟

الف) CPU ب) ET ج) I/Oها د) هر سه مورد

۱۴- کد H در شماره‌ی CPU بیانگر کدام مورد زیر است؟

الف) افزونه ب) ایمن در برابر خطا ج)یک‌پارچه د)هیچ‌کدام

۱۵- در CPUهای S7-400، ..... روی CPU نصب شده و توسط آن می‌توان پورت شبکه به پورت‌های CPU اضافه نمود.

الف) ماژول IF ب) ماژول Sync ج) کارت شبکه د) هیچ‌کدام

۱۶- کدام گزینه بیانگر عملکرد Multicomputing می‌باشد؟

الف) عملکرد دو CPU که یکی از آنها به‌عنوان رزرو دیگری می‌باشد.

ب) عمکرد چهار CPU که دو تا از آنها رزرو می‌باشند.

ج) عمکرد همزمان چند CPU در یک رک که همه یک برنامه را اجرا نمایند.

د) عملکرد همزمان چند CPU در یک رک که هر کدام برنامه‌ی مستقلی را اجرا نمایند.

۱۷- CPUهای Multicomputing به کدامیک از کارت‌های ورودی و خروجی موجود در رک دسترسی دارند؟

الف) همه‌ی کارت‌ها ب) هیچکدام از کارت‌ها

ج) کارت‌های تعریف شده برای CPU د) هر سه مورد

۱۸- در S7-400 کدامیک از راه‌اندازی‌های مجدد قابل اجرا می‌باشد؟

الف) Warm ب) Cold ج) Hot د) هر سه مورد

۱۹- در این نوع راه‌اندازی وجود باتری پشتیبان الزامی می‌باشد.

الف) Warm ب) Cold ج) Hot د) هر سه مورد

۲۰- کارت حافظه به‌منظور توسعه کدام ناحیه از حافظه‌ی CPU به‌کار می‌رود؟

الف) Load Memory ب) Work Memory

ج) System Memory د) هر سه مورد

۲۱- در مورد کارت حافظه از نوع RAM کدام گزینه صحیح نمی‌باشد؟

الف) جهت ذخیره‌سازی اطلاعات آن نیاز به باتری پشتیبان می‌باشد.

ب) در صورت قطع تغذیه اطلاعات آن پاک می‌شود.

ج) امکان برنامه‌نویسی آن در بیرون از CPU و سپس انتقال آن به CPU وجود دارد.

د) هر سه مورد

۲۲- در مورد کارت حافظه از نوع Flash Eprom کدام گزینه صحیح است؟

الف) جهت ذخیره‌سازی اطلاعات آن نیاز به باتری پشتیبان نمی‌باشد.

ب) در صورت قطع تغذیه اطلاعات آن پاک نمی‌شود.

ج) امکان برنامه‌نویسی آن در بیرون از CPU و سپس انتقال آن به CPU وجود دارد.

د) هر سه مورد

۲۳- در S7-400 نواحی PII و PIQ در کدام قسمت ناحیه‌ی حافظه قرار دارند؟

الف) Load Memory ب) Work Memory ج) System Memory د) Retentive Memory

۲۴- Work Memory در کدام‌یک از CPUهای زیر قابل توسعه است؟

الف) CPU 412-1 ب) CPU 414-3DP ج) CPU 416-3DP د) CPU 417-4

۲۵- نوع حافظه‌ی Work Memory از کدام است؟

الف) RAM ب) ROM ج) Flash Eprom د) هیچ‌کدام

۲۶- کدام مورد را نمی‌توان در Retentive Memory تعریف نمود؟

الف) Timer ب) Counter ج) Bit Memory د) PII

۲۷- چرا در پروژه‌های صنعتی نوین به‌جای استفاده از کارت‌های I/O در S7-400، از شبکه و واسط ET که کارت‌های ورودی و خروجی روی آن قرار می‌گیرد استفاده می‌شود؟

الف) این روش اقتصادی‌تر است.

ب) قابلیت‌های کارت‌های I/O که روی ET نصب می‌شود بالاتر است.

ج) موارد الف و ب

د) هیچ‌کدام

۲۸- روی یک کارت کد DI 16xDC 24V درج شده است. این کد بیانگر چیست؟

الف) کارت از نوع دیجیتال خروجی 24V است و ۱۶ ترمینال دارد.

ب) کارت از نوع دیجیتال ورودی 24v AC می‌باشد و ۱۶ ترمینال دارد.

ج) کارت از نوع دیجیتال ورودی 24 v DC می‌باشد و ۱۶ ترمینال دارد.

د) کارت از نوع دیجیتال خروجی 24 v DC می‌باشد و ۱۶ ترمینال دارد.

۲۹- کدامیک از کارت‌های زیر دارای قابلیت‌های ویژه می‌باشد؟

الف) DI 16xDC 24V

ب) DI 32xDC 24V

ج) DI 16xDC 24V Interrupt

۳۰- منظور از قابلیت Multicomputing چیست؟

الف) عملکرد همزمان چند PLC در چند رک ب) عملکرد دو CPU که یکی پشتیبان دیگری است

ج) عملکرد همزمان چند CPU در یک رک د) هیچ‌کدام

۳۱- کدامیک از جفت IMهای زیر تغذیه را نیز منتقل می‌کند؟

الف) IM 460-0 و IM461-0 ب) IM460-1 و IM461-1

ج) IM460-3 و IM461-3 د) IM460-4 و IM461-4

۳۲- IM467 برای چه‌منظور استفاده می‌شود؟

الف) ارتباط رک اصلی و توسعه ب) ارتباط S7-400 و S5

ج) ارتباط S7-400 و شبکه پروفی‌باس د) ارتباط S7-400 و شبکه اترنت صنعتی

فصل ۲

برنامه‌نویسی و کار با سیگنال‌های آنالوگ

در این فصل سعی شده است تا سیگنال‌های آنالوگ از جنبه‌های مختلف مورد بررسی قرار گرفته و نحوه کار و برنامه‌نویسی آنها در PLC تشریح گردد.

۲-۱ مقدمه

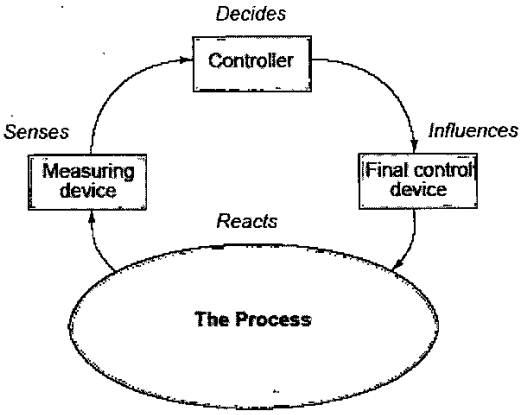
در کتاب مقدماتی توضیحاتی در مورد سیگنال‌های آنالوگ داده شد. در آنجا انواع سنسورها و عملگرها معرفی شد و تنظیمات کارت‌های ورودی و خروجی بیان گردید، ولی مطالب در حد آشنایی با آنالوگ بود. در این فصل سعی داریم که به جزئیات کار با سیگنال‌های آنالوگ بپردازیم وآنها را از جنبه‌های مختلف مورد بررسی قرار دهیم.

همانگونه که می‌دانید، از دیدگاه PLC سیگنال‌های آنالوگ به دو دسته تقسیم می‌شوند:

۱- آنالوگ ورودی: Analog Input که به‌اختصار AI خوانده می‌شود.

۲- آنالوگ خروجی: Analog Output که به‌اختصار AO خوانده می‌شود.

سنسورهای آنالوگ ورودی در سمت فرآیند پارامترهایی مانند دما، فشار، فلو، سرعت، وزن، سطح و ... را اندازه‌گیری کرده و به کنترلر ارسال می‌کنند. کنترلر براساس وضعیت سیگنال‌ها تصمیم می‌گیرد چه فرمانی را به عملگرهای موجود در فرآیند بفرستد تا فرآیند به نحو مطلوب کنترل و هدایت شود.



شکل ۲-۱ حلقه کنترل یک پارامتر فرآیندی

سنسورها و عملگرها می‌توانند دیجیتال یا آنالوگ باشند. نوع دیجیتال در کتاب مقدماتی تشریح گردید. در این فصل سیگنال‌های ورودی و خروجی آنالوگ مورد بررسی قرار می‌گیرند.

۲-۲ سیگنال‌های آنالوگ ورودی

در اتوماسیون صنعتی سیگنال‌های آنالوگ ورودی از جمله سیگنال‌های پر کاربرد محسوب می‌شوند. اهم سیگنال‌های آنالوگ ورودی در صنعت از ادوات زیر دریافت می‌شوند:

- ترانسمیترهای فشار و اختلاف فشار
- ترانسمیترهای فلو

- ترانسمیترهای سطح
- ترانسمیترهای دما
- ترموکوپل‌ها
- ترمورزیستانس‌ها
- لود سل‌ها
- مقاومت‌های متغیر

اهمیت کاربرد این سیگنال‌ها در صنایعی که اتوماسیون فرآیند[1] دارند و در آنها لوپ‌های کنترلی زیاد وجود دارد بیشتر محسوس است.

به‌طور کلی، سیگنال‌های آنالوگ ورودی برای دو کاربرد اصلی زیر در PLC استفاده می‌شوند:

- **مانیتورینگ شرایط فرآیند:** در این حالت PLC پس از دریافت سیگنال آنالوگ ورودی با پردازش مختصر آنرا در اختیار سیستم مانیتورینگ قرار می‌دهد.
- **کنترل فرآیند:** در این حالت PLC پس از دریافت سیگنال آنالوگ ورودی آنرا در برنامه کنترل استفاده می‌کند و بر اساس وضعیت آنالوگ فرامین عملگرها را صادر می‌نماید. این کاربرد تحت عنوان لوپ کنترل یا PID کنترل نیز شناخته می‌شود.

آنچه در این فصل ذکر می‌شود برای هر دو کاربرد فوق مورد نیاز خواهد بود. در این فصل ابتدا به تشریح سنسورها و وسایل ورودی آنالوگ می‌پردازیم. سپس تنظیمات و اتصالات کارت‌های AI را شرح داده و سرانجام برنامه‌نویسی سیگنال‌های آنالوگ ورودی را توضیح می‌دهیم.

۲-۲-۱ مروری بر عملکرد کارت آنالوگ ورودی

در کتاب سطح مقدماتی مدار داخلی و نحوه تبدیل سیگنال به دیتا در کارت آنالوگ ورودی تشریح گردید. همانطور که در آنجا ذکر شد، سیگنال آنالوگ توسط مبدل A/D کارت به یک رشته دیتای ۱۶ بیتی تبدیل می‌شود. ممکن است تمام ۱۶ بیت برای تبدیل استفاده نشود، که این به میزان دقت و حد تفکیک کارت آنالوگ ورودی بستگی دارد.

منظور از حد تفکیک، قابلیت جداسازی است و بیانگر تشخیص کوچکترین تغییر در سیگنال الکتریکی دریافتی توسط کارت می‌باشد.

در برخی کارت‌ها تغییرات سیگنال توسط مبدل A/D به ۱۶ بیت تبدیل می‌شود. این نوع کارت‌ها دقت زیاد دارند. در برخی دیگر تغییرات سیگنال فقط با ۱۲ بیت تبدیل می‌شود. این نوع کارت‌ها دقت کمتری دارند. دقت کارت می‌تواند از ۸ بیت تا ۱۶ بیت باشد.

حد تفکیک یک کارت بر حسب Bit بیان شده و هر چه عدد آن بزرگتر باشد، تغییرات کوچکتر توسط کارت قابل شناسایی خواهد بود و دقت کارت بیشتر است. بیشترین حد تفکیک 16Bit می‌باشد که در آن یک Bit به‌منظور علامت در نظر گرفته می‌شود. به منظور درک حد تفکیک به ذکر یک مثال می‌پردازیم.

[1] Process Automation

مثال: فرض کنید که توسط یک کارت آنالوگ ورودی، سیگنال الکتریکی ولتاژی 0 الی 10V دریافت می‌گردد. مطلوبست محاسبه‌ی حد تفکیک کارت بر حسب ولتاژ.

الف) در حالت حد تفکیک 12Bit

حل: برای محاسبه‌ی حد تفکیک کارت به روش تقریبی از روش زیر استفاده می‌شود:

$$\frac{\text{حد پایین مقدار ورودی به کارت} - \text{حد بالای مقدار ورودی به کارت}}{\text{2 به توان عدد حد تفکیک کارت}}$$

پس داریم:

$$\frac{10-0}{2^{12}} = \frac{10}{4096} = 2.44\ \text{mV}$$

یعنی حداقل تغییر قابل تشخیص توسط کارت 2.44 mV می‌باشد و کارت تغییرات کمتر از آنرا تشخیص نمی‌دهد.

ب) در حالت حد تفکیک 16Bit

حل: در حالت 16Bit، یک Bit به منظور علامت در نظر گرفته می‌شود. لذا 15Bit باقیمانده و داریم:

$$\frac{10-0}{2^{15}} = \frac{10}{32768} = 0.3\ \text{mV}$$

یعنی حداقل تغییر قابل تشخیص توسط کارت 0.3 mV می‌باشد. همانطور که مشخص است، در این حالت نسبت به حالت (الف) تغییرات کم‌تر نیز توسط کارت قابل تشخیص است، لذا دقت کارت بیشتر می‌باشد. جدول ۲-۱ نشان می‌دهد که در هر حد تفکیک[1] مقادیر با چه پله‌ای به عدد دیجیتال تبدیل می‌شوند. به‌عنوان مثال در حالت حد تفکیک 16Bit مقدار آنالوگ با فاصله‌ی یک عدد یک عدد به مقدار دیجیتال تبدیل می‌شود. در حالت حد تفکیک 14Bit مقدار آنالوگ با فاصله‌ی چهار عدد چهار عدد به مقدار دیجیتال تبدیل می‌شود.

اگر بیت علامت را در دیتای تبدیل شده در نظر نگیریم ماکزیمم قدرت تفکیک، همانطور که در جدول ۲-۱ نشان داده شده، ۱۶ بیت و حداقل ۹ بیتی است. ولی اگر بیت علامت را مانند جدول ۲-۲ در نظر بگیریم که با توجه به علامت

1. Resolution

سیگنال در یک بازه ثابت است، بنابراین تعداد بیت‌های باقیمانده که قابل تغییر هستند ماکزیمم ۱۵ بیت و حداقل ۸ بیتی خواهد بود.

جدول ۲-۱

Resolution in Bits	Units		Analog Value	
	Decimal	Hexadecimal	High-Order Byte	Low-Order Byte
9	128	80_H	00000000	1xxxxxxx
10	64	40_H	00000000	01xxxxxx
11	32	20_H	00000000	001xxxxx
12	16	10_H	00000000	0001xxxx
13	8	8_H	00000000	00001xxx
14	4	4_H	00000000	000001xx
15	2	2_H	00000000	0000001x
16	1	1_H	00000000	00000001

جدول ۲-۲

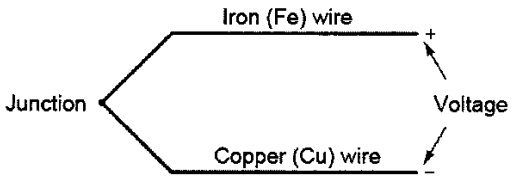
Resolution in bits (+ sign)	Units		Analog value	
	dec	hex	high byte	low byte
8	128	80_H	Sign 0000000	1xxxxxxx
9	64	40_H	Sign 0000000	01xxxxxx
10	32	20_H	Sign 0000000	001xxxxx
11	16	10_H	Sign 0000000	0001xxxx
12	8	8_H	Sign 0000000	00001xxx
13	4	4_H	Sign 0000000	000001xx
14	2	2_H	Sign 0000000	0000001x
15	1	1_H	Sign 0000000	00000001

تذکر: کارت‌های با دقت بالا تغییرات کوچک سیگنال را نیز تبدیل می‌کنند؛ بنابراین نویزپذیر هستند.

۲-۲-۲ کار با ترموکوپل

عملکرد

همانطور که در کتاب سطح مقدماتی ذکر شد، ترموکوپل از دو فلز غیر هم‌جنس تشکیل شده که در یک نقطه به هم متصل شده‌اند. وقتی این نقطه در محیط گرم قرار گیرد در دو سر آزاد ترموکوپل ولتاژ میلی ولت ظاهر خواهد شد.



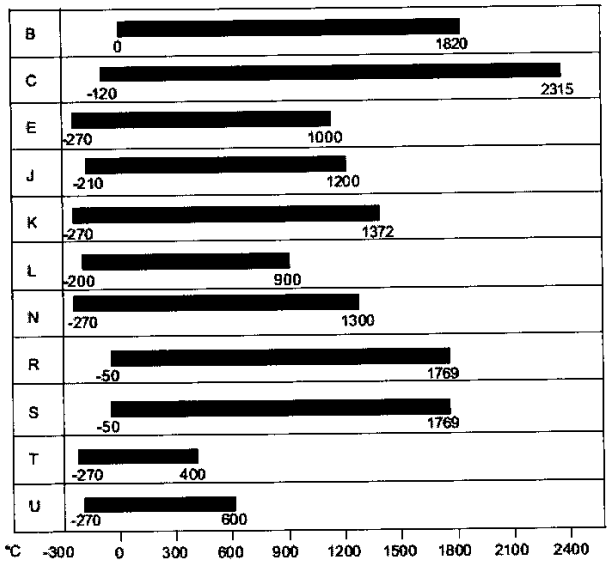
شکل ۲-۲ عملکرد نمونه ترموکوپل

ترموکوپل‌ها انواع مختلف دارند و تفاوت آنها در جنس دو فلزی است که به هم متصل گردیده‌اند. جدول ۲-۳ انواع ترموکوپل و جنس آلیاژهای به‌کار رفته در آنها را نشان می‌دهد.

جدول ۲-۳ انواع ترموکوپل و بازه دمای کاری آنها

Thermocouple	Conductor		Temperature	Voltage Range
Type	Positive	Negative	Range (°C)	(mV)
E	Chromel	Constantan	-270° to 1.000°	-9.835 to 76.358
J	Iron	Constantan	-210° to 1.200°	-8.096 to 69.536
K	Chromel	Alumel	-270° to 1,372°	-6.548 to 54.874
T	Copper	Constantan	-270° to 400°	-6.258 to 20.869
S	Platinum-10% Rhodium	Platinum	-50° to 1,768°	-0.236 to 18.698
R	Platinum-13% Rhodium	Platinum	-50° to 1,768°	-0.226 to 21.108

شکل ۲-۳ بازه اندازه‌گیری دما را برای ترموکوپل‌های مختلف نشان می‌دهد.



شکل ۲-۳ بازه اندازه‌گیری دما برای ترموکوپل‌های مختلف

نکات قابل توجه

- ترموکوپل برخلاف سایر ورودی‌های آنالوگ که تغذیه بیرونی دارند هیچ تغذیه‌ای لازم ندارد و در واقع خود ترموکوپل به‌عنوان یک منبع جریان محسوب می‌گردد.

- سیگنال میلی‌ولت ترموکوپل به‌شدت نویزپذیر است و اگر انتخاب کابل و کابل‌کشی به‌طور صحیح انجام نگیرد ممکن است سیستم کنترل در شرایط بروز نویز با مشکل مواجه شود.
- ترموکوپل برخلاف ترمورزیستانس‌ها برای اندازه‌گیری دماهای بالا به‌کار می‌رود.
- در ترمورزیستانس در اثر عبور جریان خود سنسور گرم شده و خطا در اندازه‌گیری ایجاد می‌کند. این مشکل در ترموکوپل وجود ندارد.
- ساختار ترموکوپل محکم‌تر و سخت از ترمورزیستانس است؛ بنابراین به‌راحتی آنرا در شکل‌های مختلف برای کاربردهای متنوع عرضه می‌کنند.
- در اتصال ترموکوپل به کارت آنالوگ ورودی یا در اتصال سیم ترموکوپل به سیم مسی در محل اتصال ترموکوپل جدیدی ایجاد می‌شود که منجر به خطا در اندازه‌گیری می‌شود؛ از اینرو باید به روش‌های مختلف این خطا را برطرف کرد. این موضوع در ادامه در بحث جبران‌سازی تشریح می‌شود.

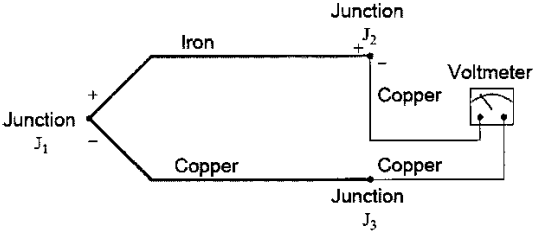
جبران‌سازی خطا در ترموکوپل

در اتصال ترموکوپل به سیستم کنترل با دو اصطلاح مواجه هستیم:

- Hot Junction یا محل اتصال گرم که در داخل محیط گرم قرار می‌گیرد.
- Cold Junction یا محل اتصال سرد که در بیرون محیط گرم قرار دارد و محلی است که سیم ترموکوپل به سیم یا فلز دیگری متصل می‌شود.

در ترموکوپل‌ها در محل اتصال سرد خطا ایجاد می‌شود. منشاء این خطا ایجاد ترموکوپل جدید در محل اتصال فلز ترموکوپل به فلز دیگر است. اگر سیم ترموکوپل کوتاه باشد محل اتصال سرد، نقطه‌ای است که سیم ترموکوپل به سیم مسی متصل گردیده است. اگر سیم ترموکوپل بلند باشد و به‌طور مستقیم به ترمینال ورودی کنترلر متصل گردد محل اتصال سرد، ترمینال ورودی است که جنس آن با جنس سیم ترموکوپل متفاوت است.

برای سادگی فرض کنید که ترموکوپلی مانند شکل ۲-۴ داریم که یکی از دو فلز آهن و مس تشکیل شده است و در محل اتصال سرد نیز هر دو سیم از جنس مس هستند؛ بنابراین دو ترموکوپل خواهیم داشت که ترموکوپل نقطه گرم J1 در دمای بالا قرار دارد و ترموکوپل نقطه سرد J2 در دمای محیط است. بر این اساس ولتاژ دریافت شده که توسط ولتمتر اندازه‌گیری می‌شود تفاضل J2-J1 خواهد بود یعنی به اندازه J1 خطا داریم.



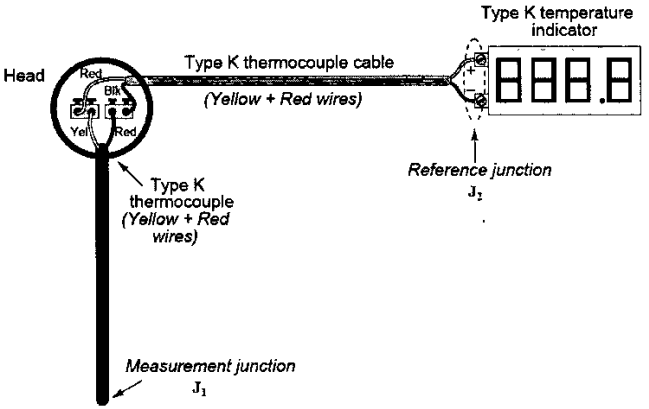
$$V_{meter} = V_{J1} - V_{J2}$$

شکل ۲-۴ خطا در محل اتصال سرد ترموکوپل

به‌صورت تئوری می‌توان گفت که اگر دما در محل اتصال سرد را روی صفر درجه ثابت نگه داریم، Vj2 صفر خواهد شد. این کار در عمل امکان‌پذیر نیست و به روش‌های دیگری عمل می‌شود.

استفاده از سیم هم‌جنس ترموکوپل

اگر از ترموکوپل با سیم بلند استفاده کنیم یا اگر در محل اتصال از سیم همجنس ترموکوپل استفاده کنیم خطای فوق برطرف خواهد شد ولی در محل اتصال به کنترلر باز ترموکوپل ضعیفی خواهیم داشت. بدیهی است دما در محل نصب سیستم کنترل معمولاً به‌صورت ثابت و کمتر از ۳۰ درجه سانتی‌گراد کنترل می‌شود ولی در فیلد در Junction Box دما متغیر است و ممکن است در طول روز و شب یا فصل گرم و سرد بیشتر یا کمتر از دمای محیط کنترلر باشد.
با این روش یک خطای متغیر را به یک خطای کوچکتر و نسبتاً ثابت تغییر داده‌ایم.

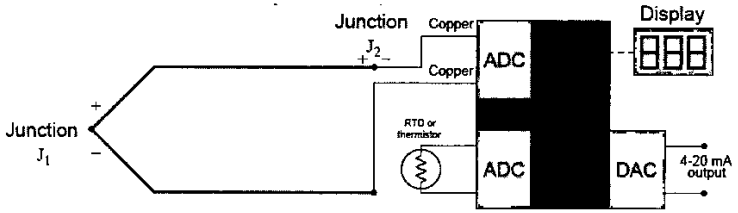


شکل ۲-۵ استفاده از سیم هم‌جنس ترموکوپل برای جبران خطا

روش فوق در فواصل زیاد هزینه زیادی را در بر خواهد داشت. روش اقتصادی‌تری که در صنعت مورد استفاده قرار می‌گیرد استفاده از سیم‌هایی است که جنس آن نزدیک به جنس سیم‌های ترموکوپل باشد. این سیم‌ها Extension Grade خوانده می‌شوند.

استفاده از جبران‌ساز بیرونی

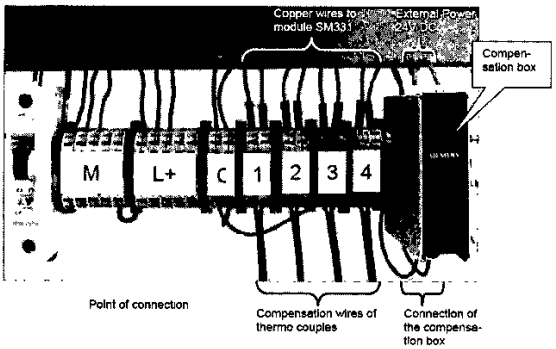
روش دیگر که نسبت به روش قبلی اقتصادی‌تر است این است که ترموکوپل را به همان سیم‌های مسی معمولی متصل کنیم. این روش برای مسافت‌های زیاد از نظر اقتصادی کم هزینه‌تر از روش قبل است. برای جبران خطا ناشی از تشکیل ترموکوپل در محل Cold Junction با استفاده از سنسور جداگانه‌ای دمای محل اتصال سرد را اندازه‌گیری کرده و به کنترلر انتقال دهیم. کنترلر به‌صورت داخلی این دمای اندازه‌گیری شده اصلی را با این دما اصلاح می‌کند. به‌عنوان مثال در شکل ۲-۴ دیدیم که دمای اندازه‌گیری شده کمتر از دمای واقعی است. با روش فوق کنترلر دمای محل اتصال سرد را به دمای دریافتی از ترموکوپل اضافه می‌کند و خطا جبران می‌شود.



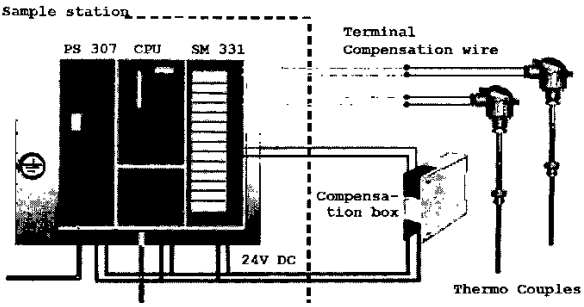
شکل ۲-۶ استفاده از جبران‌ساز در محل اتصال سرد ترموکوپل

فصل ۲

برای اینکه روش فوق قابل استفاده باشد، معمولاً کارت‌های آنالوگ ورودی که سیگنال ترموکوپل دریافت می‌کنند دارای یک ورودی مجزا برای جبران‌ساز هستند.



شکل ۲-۷ ماژول جبران‌ساز ساخت زیمنس



شکل ۲-۸ نحوه اتصال جبران‌ساز به کارت ورودی آنالوگ

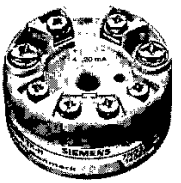
تذکر: در روش فوق اگر کارت به‌عنوان مثال ۸ ورودی باشد، جبران‌ساز برای همه آنها یکسان عمل می‌کند. امکان اتصال چند جبران‌ساز بیرونی به کارت وجود ندارد پس بهتر است سیم همه ترموکوپل‌ها درهمان Cold Junction که جبران‌ساز در آن نصب شده است به سیم مسی متصل شده و همه ترموکوپل‌ها نیز از یک نوع باشند.

تذکر: در کاربردهایی که سیم ترموکوپل مستقیماً به ترمینال ورودی کنترلر وصل می‌شود نیاز به جبران‌ساز بیرونی نیست. جبران‌ساز داخلی در خود کارت تعبیه شده است.

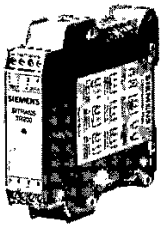
تذکر: در روش فوق اگر دو ترمینالی که سیم‌های ترموکوپل به آن متصل است اتصال کوتاه شود، دمایی که توسط کنترلر دیده می‌شود دمای محیط است (که از RTD دریافت می‌شود).

استفاده از مبدل mV/mA

روش دیگر استفاده از مبدل است که سیگنال دریافتی از ترموکوپل را به میلی‌آمپر تبدیل کرده و انتقال می‌دهد.



الف) مبدل نوع Head mount

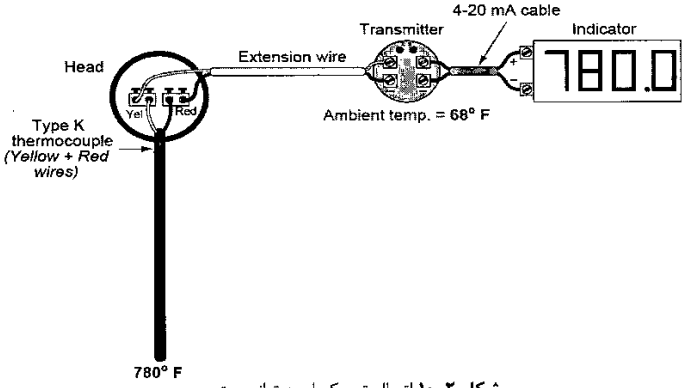


ب) مبدل نوع Field mount

شکل ۲-۹ مبدل‌های mV/mA

این مبدل که Temperature Transmitter نام دارد به دو صورت قابل نصب است:

- روی خود سنسور در محل اندازه‌گیری مانند شکل الف که به آن head mount می‌گویند.
- در محل Junction box که سیم ترموکوپل تا آن نقطه کشیده شده است که به آن panel mount می‌گویند.



شکل ۲-۱۰ اتصال ترموکوپل به ترانسمیتر

ترانسمیترهای فوق دارای جبران‌ساز داخلی هستند و خطای ناشی از محل اتصال سیم ترموکوپل به ترمینال ترانسمیتر به‌صورت داخلی در آنها جبران می‌شود. در برخی مدل‌ها امکان اتصال جبران‌ساز بیرونی به ترانسمیتر نیز وجود دارد که

ترانسمیتر خطا را براساس سیگنال آنها تصحیح می‌کند. یکی از ویژگی‌های بزرگ استفاده از این روش کاهش تأثیر نویز است سیگنال 4-20 mA نسبت به mV نویزپذیری بسیار کمتری دارد.

این مبدل‌ها را نمی‌توان در محیطهای IS به‌کار برد. بنابراین در محیطهای خطرناک استفاده از سیم هم‌جنس ترموکوپل اگر چه پر هزینه‌تر است ولی به‌دلیل پایین بودن سطح ولتاژ و توان روی کابل شرایط ایمن‌تری را فراهم می‌سازد.

فصل ۲

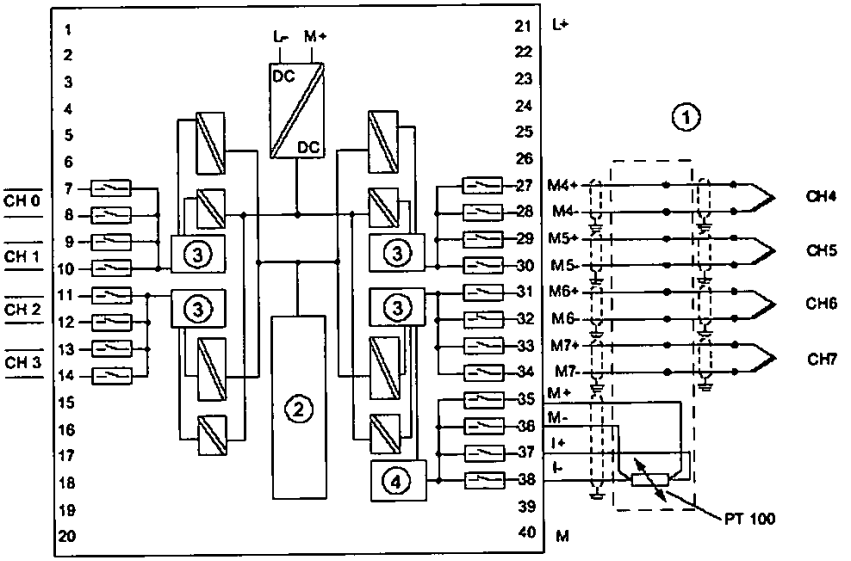
مقایسه روش‌های اتصال ترموکوپل

جدول ۲-۴

	اتصال با سیم هم‌جنس ترموکوپل	استفاده از جبران‌ساز بیرونی	استفاده از مبدل mv/mA
نویز پذیری	زیاد	زیاد	کم
کاربرد در محیط خطرناک	بله	خیر	خیر
هزینه	زیاد	کم	زیاد

اتصال ترموکوپل به کارت آنالوگ

برای اتصال هر ترموکوپل دو ترمینال تعبیه شده است (اتصال همیشه دو سیمه انجام می‌شود). شکل ۲-۱۱ نقشه اتصال ترموکوپل به کارت آنالوگAI8xTC را که مخصوص ترموکوپل است نشان می‌دهد.



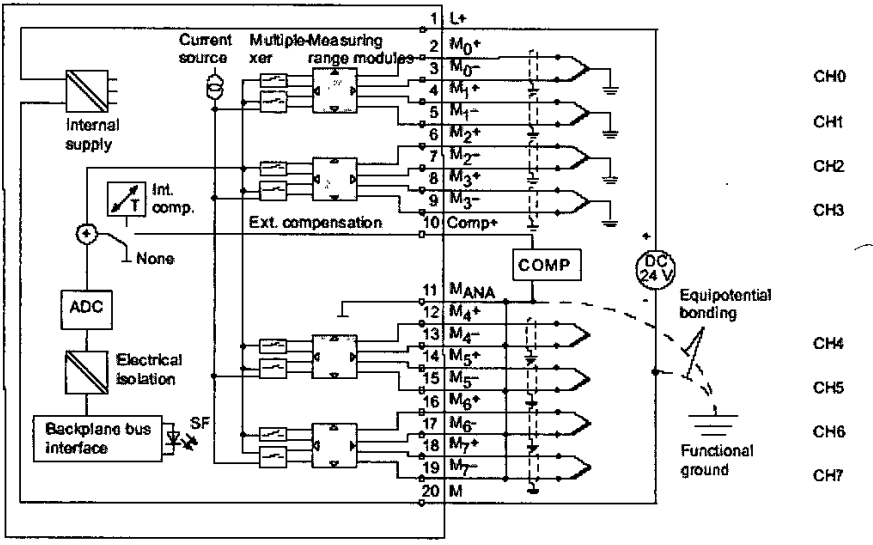
شکل ۲-۱۱ اتصالات کارت AI8xTC

همانطور که در شکل دیده می‌شود، این کارت دارای هشت کانال CH0 تا CH7 است که بر هر کانال می‌توان یک ترموکوپل را متصل نمود.

جبران‌ساز بیرونی در صورت نیاز در محل Cold Junction نصب می‌شود. این جبران‌ساز می‌تواند یک PT100 باشد که به‌صورت ۴ سیمه مانند شکل فوق به کارت متصل می‌گردد.

اگر از سیم هم‌جنس ترموکوپل استفاده شده باشد. در این حالت اتصال PT100 فوق لازم نیست. برای جبران‌سازی خطای ناشی از محل اتصال سیم ترموکوپل به ترمینال کارت یک جبران‌ساز داخلی استفاده شده است که دمای کارت را اندازه‌گیری کرده و سیگنال را تصحیح می‌کند. این جبرانساز داخلی برای 50-0 درجه سانتیگراد تنظیم شده است.

شکل ۲-۱۲ نمونه دیگری از کارت آنالوگ ورودی را نشان می‌دهد که قادر است سیگنال‌های متنوع جریانی و ولتاژی و RTD و ترموکوپل را دریافت کند.

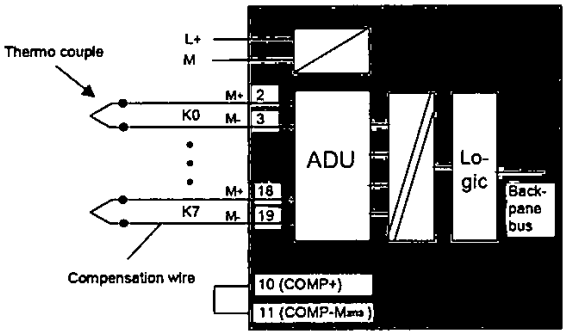


شکل ۲-۱۲ اتصالات کارت AI8x12 bit

این کارت دارای یک Selection Module است که با تنظیم آن هر دو کانال می‌توانند یک نوع سیگنال را دریافت کنند. بنابراین اگر تنظیم اولین Selection Module روی ترموکوپل باشد به کانال‌های CH0 و CH1 فقط می‌توان ترموکوپل متصل نمود.

همانطور که دیده می‌شود، دو ترمینال برای جبران‌ساز COMP در نظر گرفته شده است.

اگر سیم هم‌جنس ترموکوپل باشد برای استفاده از جبران‌سازی داخلی ورودی Comp اتصال کوتاه می‌شود.



شکل ۲-۱۳ اتصال ترموکوپل به کارت AI8x12 bit

تذکر: در صورتی‌که از مبدل mv/mA یعنی ترانسمیتر دما استفاده شود، نحوه اتصال آن مطابق توضیحاتی است که در بخش سیگنال‌های جریانی آورده شده است.

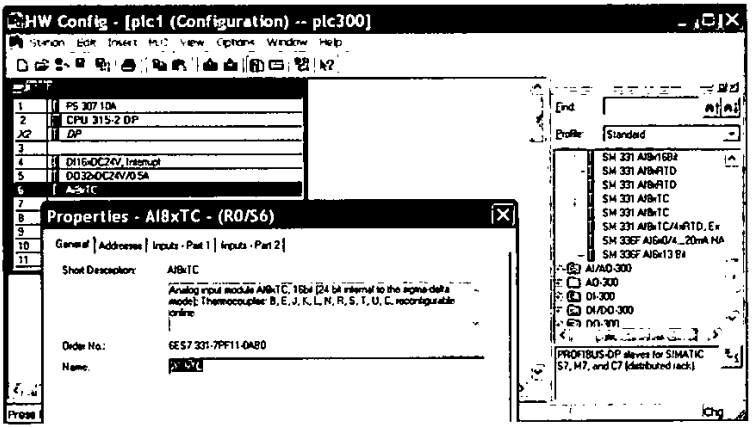
تنظیمات ترموکوپل در Hwconfig

با توضیحاتی که در کتاب مقدماتی و فصل اول این کتاب داده شد با نحوه پیکربندی کارت‌های آنالوگ در Hwconfig آشنا شده‌اید. در اینجا به مرور برخی نکات خاص ترموکوپل در تنظیمات کارت می‌پردازیم.

کارت AI8xTC را بررسی می‌کنیم که به‌طور خاص فقط برای ترموکوپل استفاده می‌شود. تنظیمات این کارت بیش از تنظیمات کارت‌های چند منظوره است که ترموکوپل را نیز پشتیبانی می‌کنند بنابراین با معرفی این تنظیمات شناخت پارامترهای ترموکوپل در سایر کارت‌ها نیز آسان خواهد بود.

کارت AI 8 x TC در زیر مجوعه SM300 قرار دارد و می‌توان آنرا روی رک 300 یا روی ET200M (که به شبکه پروفی‌باس متصل است) قرار داد.

پس از وارد کردن این کارت با دوبار کلیک روی آن سه سربرگ زیر را می‌بینیم.



شکل ۲-۱۴ وارد کردن کارت AI8xTC در محیط hwconfig

بخش‌های General و Address قبلا توضیح داده شده‌اند. همانطور که در توضیحات بخش General دیده می‌شود، میزان دقت این کارت 16 بیتی است.

در این پنجره دو سربرگ دیگر با عناوین زیر وجود دارد:

- Inputs Part 1
- Inputs Part2
- Input Part 1

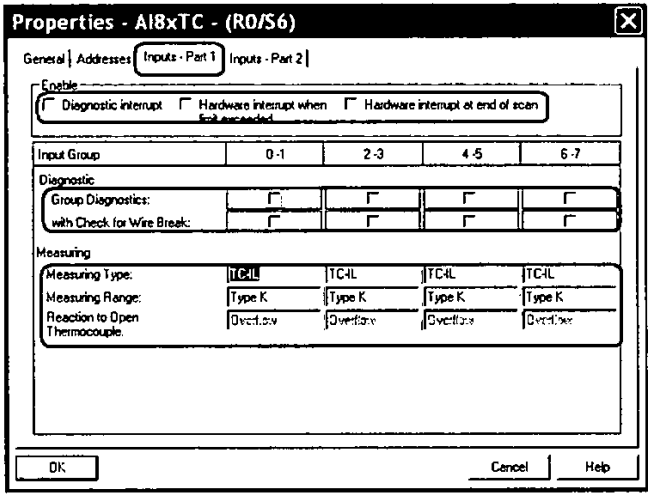
همانطور که در شکل ۲-۱۵ دیده می‌شود، این سربرگ شامل سه بخش Enable، Diagnostic و Measuring است.

در بخش Enable می‌توان تنظیمات مربوط به وقفه کارت را فعال نمود. در اینجا دو نوع وقفه دیده می‌شود :

Diagnostics Interrupt: وقفه ای است که در صورت بروز اشکال در کارت فعال می‌شود این وقفه OB82 را فراخوان می‌کند.

Hardware Interrupt: وقفه‌ای است که در صورت بروز شرایط خاص مانند افزایش دما از حد معین یا کاهش دما از حد تعیین شده فعال می‌گردد. وقتی این وقفه فعال شود در پنجره تنظیمات حدود آن تعیین می‌شود. برای کارت فوق این حدود در سربرگ Input Part 2 نمایش داده می‌شوند. فعال شدن این وقفه OB40 را فراخوان می‌کند.

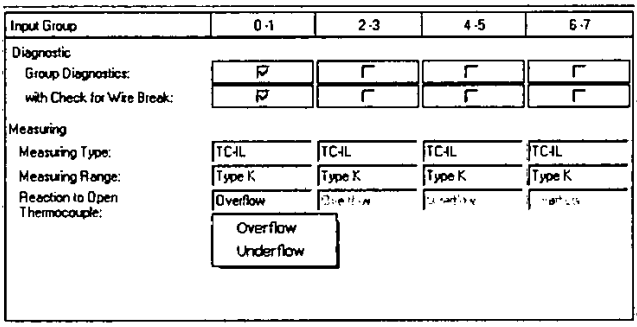
وقفه‌ها در فصل‌های بعدی همین کتاب تشریح شده‌اند.



شکل ۲-۱۵ تنظیمات مربوط به سربرگ Input part 1

بخش Diagnostic: در این بخش می‌توان وقفه مربوط به اشکال را برای کانال‌های مورد نظر فعال نمود. وقتی این گزینه را برای یک زوج کانال فعال کنیم در زیر آن گزینه Check for wire break فعال می‌شود و می‌توان Reaction to Open thermocouple را انتخاب نمود. یعنی تعیین کرد که اگر سیم ترموکوپل باز شد چه مقداری را برای سیگنال

آن جایگزین نماید. Overflow مقدار 7FFF هگز معادل 32767 را جایگزین می‌کند و underflow مقدار 8000 هگز معادل 32768- را به‌جای سیگنال قرار می‌دهد.

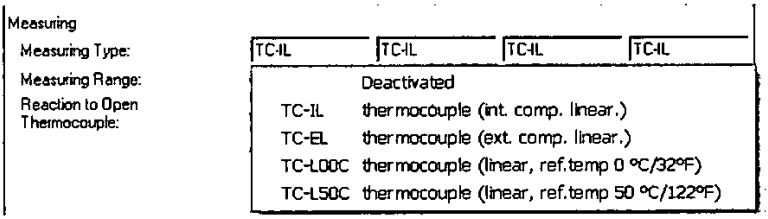


شکل ۲-۱۶ تنظیمات کارت AI8xTC

- **Overflow:** در پروسه‌های گرمایشی چنانچه این گزینه انتخاب شود، در صورت قطع شدن سیم ترموکوپل، ماژول مقدار 7FFF را به فرم Hex برمی‌گرداند و حلقه‌ی کنترلی به‌طور اتوماتیک گرما را کاهش می‌دهد.
- **Underflow:** در پروسه‌های سرمایشی چنانچه این گزینه انتخاب شود، در صورت قطع شدن سیم ترموکوپل، ماژول مقدار 8000 را برمی‌گرداند و حلقه‌ی کنترلی به‌طور اتوماتیک سرما را کاهش می‌دهد.

بخش Measuring: در بخش Measuring نوع ترموکوپل و نوع جبران‌سازی را می‌توان دقیقاً مشخص نمود.

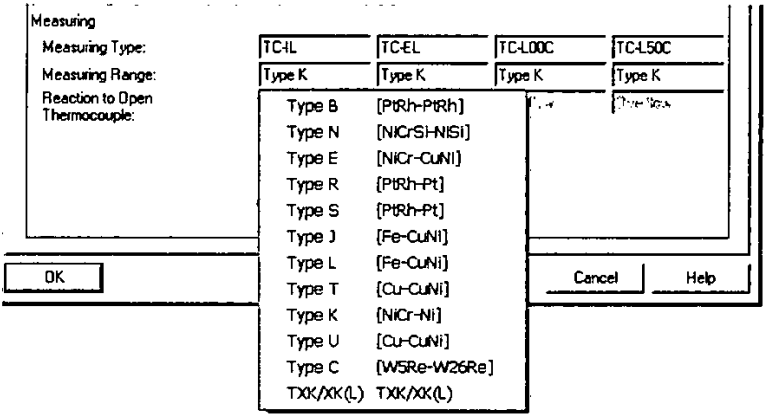
در قسمت Measuring type مانند شکل ۲-۱۷ گزینه‌های مختلفی دیده می‌شود.



شکل ۲-۱۷ انتخاب نوع جبران‌سازی ترموکوپل در تنظیمات کارت

- **TC-IL:** جبران‌سازی به‌صورت داخلی و تبدیل به‌صورت خطی است.
- **TC-EL:** جبران‌سازی به‌صورت خارجی است ولی تبدیل خطی است.
- **TC-L00C:** به‌صورت خطی و بر اساس این است که جبران‌ساز خارجی در دمای مبنای 0 درجه سانتی‌گراد است.
- **TC-L50C:** به‌صورت خطی و بر اساس این است که جبران‌ساز خارجی در دمای مبنای 50 درجه سانتی‌گراد است.
- گزینه Deactivated عمل تبدیل A/D کانال مربوطه را غیرفعال می‌کند.

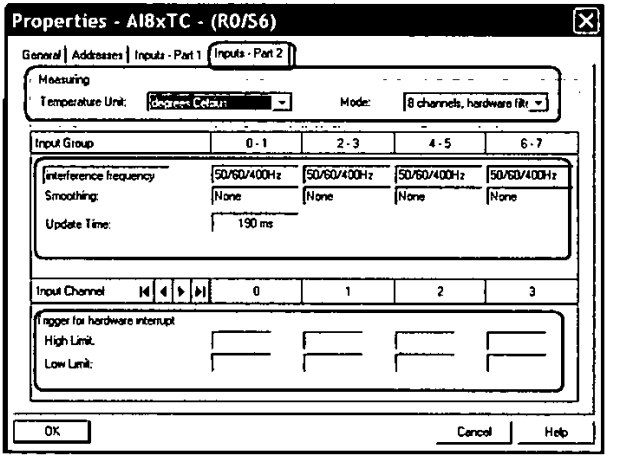
به‌جز Deactivated هر کدام از گزینه‌های دیگر انتخاب شود، در قسمت Measuring Type می‌توان نوع ترموکوپل را دقیقاً تعیین کرد.



شکل ۲-۱۸ انتخاب نوع ترموکوپل در تنظیمات کارت

Input Part 2

در این قسمت، مطابق شکل ۲-۱۹ گزینه‌های جدید زیر وجود دارد:



شکل ۲-۱۹ سربرگ Input- Part2 در تنظیمات کارت AI8xTC

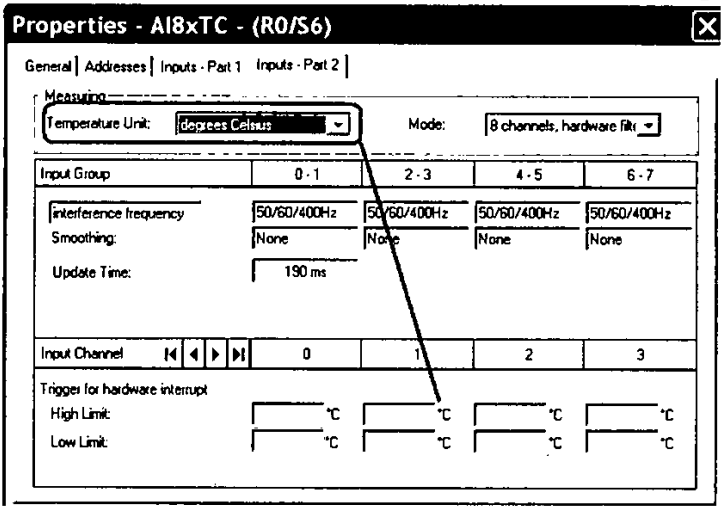
Temperature Unit

در این قسمت واحد اندازه‌گیری دما که مربوط به وقفه سخت‌افزاری است تعیین می‌شود. گزینه‌های قابل انتخاب عبارتند از:

- درجه Celsius

- درجه Fahrenheit

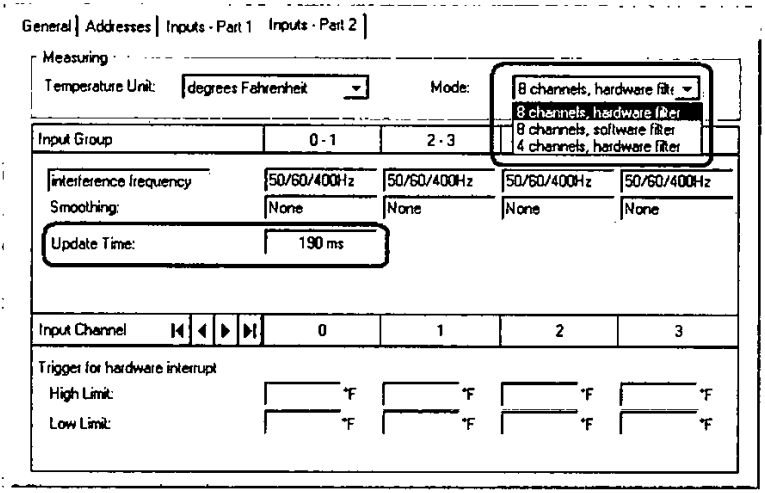
اگر قبلاً در سربرگ Input Part 1 گزینه Hardware Interrupt فعال باشد، واحد اندازه‌گیری دما را در پایین پنجره فوق می‌بینیم.



شکل ۲-۲۰ تنظیم واحد دما برای وقفه در کارت AI8xTC

Mode

در این قسمت می‌توان تعیین نمود که چند کانال کارت فعال باشند. این تنظیم بر روند زمان تبدیل آنالوگ به دیجیتال در کارت تأثیرگذار می‌باشد.

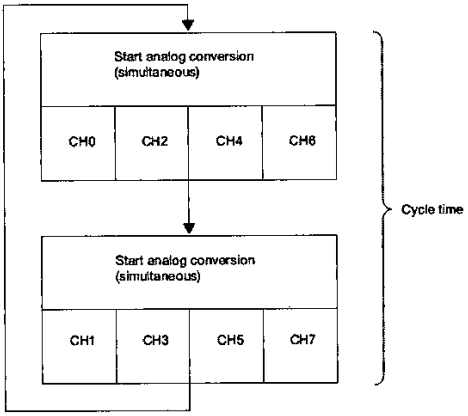


شکل ۲-۲۱ تنظیمات Mode در کارت AI8xTC

اگر دقت کنیم با انتخاب هر کدام از گزینه‌های این قسمت زمان سیکل تبدیل که روبروی update time نوشته شده است تغییر می‌کند. توضیحات بیشتر در ادامه آورده شده است.

گزینه‌های قابل انتخاب عبارتند از:

- **8 channels, hardware filter**: با انتخاب این گزینه، همه‌ی ۸ کانال کارت فعال بوده و نویزها تا 100 دسی‌بل را فیلتر می‌کند. در این حالت زمان سیکل تبدیل آنالوگ به دیجیتال که در شکل ۲-۲۲ نشان داده شده است زیاد و در حد 200 ms خواهد بود. همانطور که در شکل۲-۱۱ مشخص است این کارت دارای ۴ مبدل A/D است که در نقشه فوق با شماره ۳ نشان داده شده است. در این مد روش کار به این صورت است که ابتدا کانال‌های زوج به مبدل A/D ارسال می‌شود سپس نوبت به کانال‌های فرد می‌رسد. بنابراین سیکل تبدیل به‌صورت زیر خواهد بود:



شکل ۲-۲۲ سیکل تبدیل آنالوگ به دیجیتال در حالت ۸ کانال

کارت برای تبدیل سیگنال هر کانال 91 ms زمان نیاز دارد. از آنجا که برای هر دو کانال مجاور یک A/D وجود دارد بنابراین باید به‌نوبت روی مبدل سوئیچ شوند. برای سوئیچ از یک کانال به کانال دیگر از همان گروه از رله OptoMOS استفاده شده که زمان سوئیچینگ آن 7 ms است. بنابراین زمان سیکل تبدیل به‌صورت زیر و نزدیک 200ms خواهد بود:

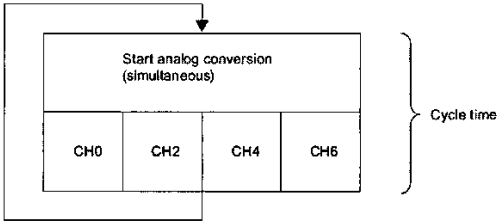
$$\text{Cycle time} = (91\ \text{ms} + 7\ \text{ms}) \times 2 = 196\ \text{ms}$$

- **8 channels, software filter**: با انتخاب این گزینه همه‌ی ۸ کانال کارت فعال بوده و نویزهای تا 90dB فیلتر می‌گردد. در این حالت زمان تبدیل آنالوگ به دیجیتال در کارت کمتر از حالت قبل است. سیکل تبدیل در این روش مشابه شکل قبل است یعنی ابتدا کانال‌های زوج سپس کانال‌های فرد تبدیل می‌شوند. تفاوتی که با حالت قبل دارد در این است که بسته به فرکانس نمونه برداری، سیکل تبدیل متغیر است. جدول ۲-۵ این موضوع را نشان می‌دهد.

جدول ۲-۵

Programmed noise suppression	Channel cycle time*	Module cycle time (all channels)
50 Hz	83 ms	166 ms
60 Hz	72 ms	144 ms
400 Hz	23 ms	46 ms
* Channel cycle time = channel conversion time +7 ms channel changeover time within the group		

- **4 channels, hardware filter:** با انتخاب این گزینه تنها 4 کانال از 8 کانال کارت فعال می‌باشد. در این حالت زمان تبدیل آنالوگ به دیجیتال در کارت به طرز قابل ملاحظه‌ای تا 10 ms کاهش می‌یابد.



شکل ۲-۲۳ سیکل تبدیل آنالوگ به دیجیتال در حالت ۴ کاناله

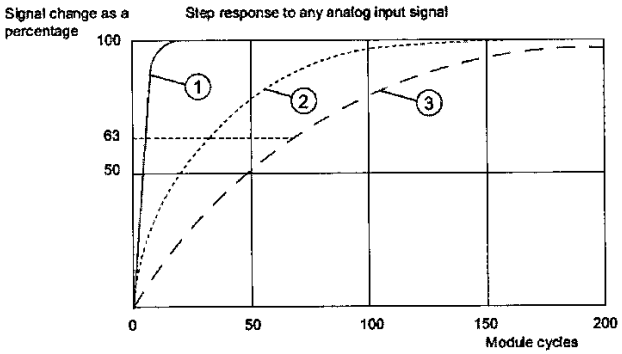
تذکر: اگر گزینه Check for wire break فعال باشد یعنی قطع شدن سیم نیاز به بررسی داشته باشد، در دو حالت اول یعنی ۸ کاناله این کار سیکل تبدیل را به اندازه 4ms افزایش می‌دهد. در حالت سوم یعنی ۴ کاناله سیکل تبدیل همان 10ms است و هر 170 ms یکبار قطع شدن سیم بررسی می‌گردد.

تذکر: کانال‌های استفاده نشده را اولاً Deactivate کنید تا در سیکل تبدیل وارد نشوند و سیکل کاهش یابد. ثانیاً ترمینال مثبت و منفی آنها را اتصال کوتاه کنید تا در بررسی Diagnostic دیده نشوند و خطاهای احتمالی آن کانال‌ها فیلتر شود.

شکل ۲-۲۴ اتصال کوتاه کانال‌های استفاده نشده

Smoothing

توسط این گزینه می‌توان تغییرات سیگنال ورودی را با شیب مورد نظر به پردازشگر منتقل نمود. اگر این گزینه روی Weak قرار گیرد هر گونه تغییر به‌سرعت منتقل می‌شود و اگر گزینه Strong انتخاب شود، تغییرات به آرامی منتقل می‌گردد. منحنی‌های شکل ۲-۲۵ این موضوع را بهتر نشان می‌دهند. توجه شود که محور افقی سیکل تبدیل و محور عمودی سیگنال آنالوگ است. انتخاب smoothing برای همه سیگنال‌های آنالوگ وجود دارد. بسته به ماهیت سیگنال و رفتار آن می‌توان smoothing مناسب را انتخاب نمود. به‌عنوان مثال اگر تغییرات سیگنال آنالوگ به‌صورتی است که فقط در برخی حالات ممکن است یک پیک گذرا پیش بیاید با انتخاب حالت Strong می‌توان شرایط گذرا را فیلتر کرد.



شکل ۲-۲۵ انواع Smoothing در کارت آنالوگ

اگر سیگنال ترموکوپل توسط نویز دچار اغتشاش شود، می‌توان با انتخاب منحنی ۳ موج دریافتی را صاف نمود.

جدول‌های تبدیل A/D ترموکوپل‌ها

همانطور که در ابتدای این فصل توضیح داده شد، سیگنال‌های آنالوگ توسط A/D به عدد ۱۶ بیتی تبدیل می‌شوند جدول‌های ۲-۶ نتیجه تبدیل دمای هر ترموکوپل را به عدد ۱۶ بیتی نشان می‌دهد. عدد ۱۶ بیتی به فرمت دسیمال و هگز ارائه شده است.

تذکر: اتصال ترموکوپل با پلاریته معکوس یا استفاده از ترموکوپل نادرست منجر به سیگنال Underflow خواهد شد.

جدول‌های ۲-۶

Type B			Type C in			
°C	dec	hex	°C	dec	hex	
> 2070,0	32767	7FFFH	> 2500,0	32767	7FFFH	Overflow
2070,0 : 1820,1	20700 : 18201	50DCH : 4719H	2500,0 : 2315,1	25000 : 23151	61A8H : 5A6FH	Overshoot range
1820,0 : 0,0	18200 : 0	4718H : 0000H	2315,0 : 0,0	23150 : 0	5A6EH : 0000H	Rated range
: -120,0	: -1200	: FB50H	-0,1 : -120,0	-1 : -1200	FFFFH : FB50H	Undershoot range
< -120,0	-32768	8000H	< -120,0	-32768	8000H	Underflow

Type E			Type J			
> 1200,0	32767	7FFFH	> 1450,0	32767	7FFFH	Overflow
1200,0 : 1000,1	12000 : 10001	2EE0H : 2711H	1450,0 : 1200,1	14500 : 12001	38A4H : 2EE1H	Overshoot range
1000,0 : -270,0	10000 : -2700	2710H : F574H	1200,0 : -210,0	12000 : -2100	2EE0H : F7CCH	Rated range
-	-	-	-	-	-	Undershoot range
< -270,0	< -2700	< F574H	< -210,0	< -2100	<F7CCH	Underflow

Type K			Type L			
> 1622,0	32767	7FFFH	> 1150,0	32767	7FFFH	Overflow
1622,0 : 1372,1	16220 : 13721	3F5CH : 3599H	1150,0 : 900,1	11500 : 9001	2CECH : 2329H	Overshoot range
1372,0 : -270,0	13720 : -2700	3598H : F574H	900,0 : -200,0	9000 : -2000	2328H : F830H	Rated range
-	-	-	-	-	-	Undershoot range
< -270,0	< -2700	< F574H	< -200,0	< -2000	< F830H	Underflow

Type N			Type R ,S			
°C	dec	hex	°C	dec	hex	
> 1550,0	32767	7FFFH	> 2019,0	32767	7FFFH	Overflow
1550,0 : 1300,1	15500 : 13001	3C8CH : 32C9H	2019,0 : 1769,1	20190 : 17691	4EDEH : 451BH	Overshoot range
1300,0 : -270,0	13000 : -2700	32C8H : F574H	1769,0 : -50,0	17690 : -500	451AH : FE0CH	Rated range
-	-	-	-50,1 : -170,0	-501 : -1700	FE0BH : F95CH	Undershoot range
< -270,0	< -2700	< F574H	< -170,0	-32768	8000H	Underflow

Type T			Type U			
°C	dec	hex	°C	dec	hex	
> 540,0	32767	7FFFH	> 850,0	32767	7FFFH	Overflow
540,0 : 400,1	5400 : 4001	1518H : 0FA1H	850,0 : 600,1	8500 : 6001	2134H : 1771H	Overshoot range
400,0 : -270,0	4000 : -2700	0FA0H : F574H	600,0 : -200,0	6000 : -2000	1770H : F830H	Rated range
-	-	-	-	-	-	Undershoot range
< -270,0	< -2700	< F574H	< -200,0	< -2000	< F830H	Underflow

۲-۲-۳ کار با سنسورهای مقاومتی

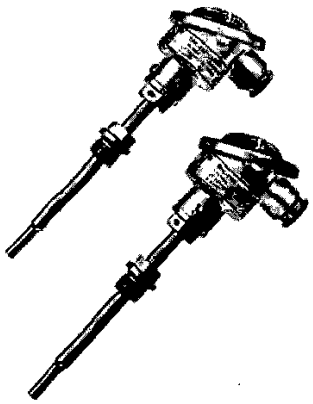
سنسورهای مقاومتی که به‌صورت ورودی آنالوگ به PLC متصل می‌شوند عبارتند از:

- ترموزیستانس یا RTD
- PTC
- مقاومت متغیر

فصل ۲

الف) بررسی RTD

RTD یا Resistance Temperature Detector یکی از سنسورهای پر کاربرد در اندازه‌گیری دما هستند. اصول اندازه‌گیری در آنها براساس تغییرات مقاومت ناشی از تغییرات دماست.



شکل ۲-۲۶ نمونه RTD صنعتی

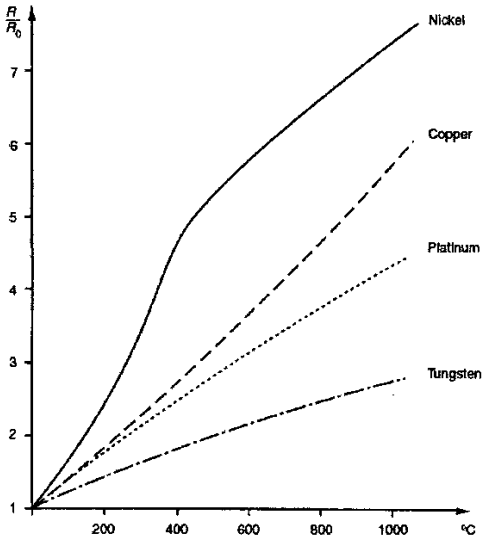
در واقع اصول عملکرد بر اساس رابطه زیر است که نشان می‌دهد میزان تغییر مقاومت تقریباً نسبت مستقیم با مقدار دما دارد. این رابطه ساده شده و تقریبی است:

$$R = R_0 (1+aT)$$

در این رابطه R0 مقدار مقاومت در صفر درجه سانتی‌گراد و a ضریبی است که به جنس مقاومت بستگی دارد جنس مقاومت می‌تواند مس یا پلاتین یا تنگستن یا نیکل باشد که هر کدام در بازه خاصی کاربرد دارند؛ اگر چه در دماهای زیاد به‌کار نمی‌روند.

- Platinum: -270°C to +1000°C
- Copper: -200°C to +260°C
- Nickel: -200°C to +430°C
- Tungsten: -270°C to +1100°C

شکل ۲-۲۷ منحنی تغییرات R/R0 را بر حسب تغییرات دما در هر کدام از انواع فوق نشان می‌دهد.

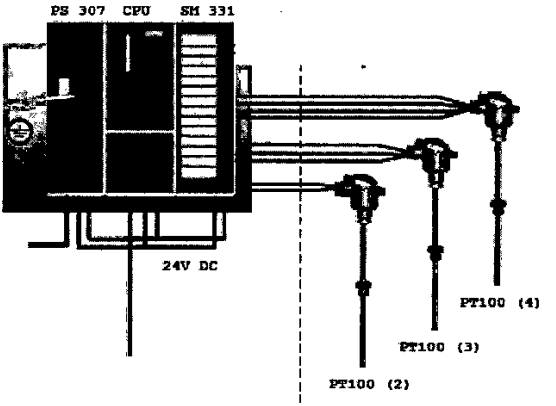


شکل ۲-۲۷ تغییر مقاومت انواع RTD نسبت به دما

Pt100 یکی از اعضای خانواده RTD است که پرکاربرد و معروف است. این سنسور در دمای صفر درجه مقاومتی برابر با 100 اهم دارد و تغییرات آن نسبتاً خطی است.

خطای RTD و نحوه اصلاح آن

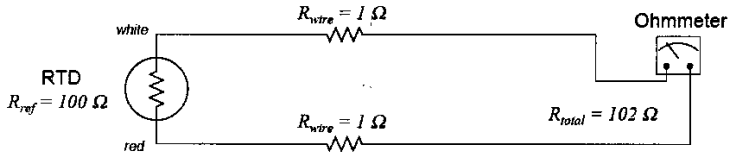
اتصال RTD می‌تواند دو سیمه یا سه سیمه یا چهار سیمه باشد.



شکل ۲-۲۸ اتصال RTD به کارت آنالوگ

دو سیمه

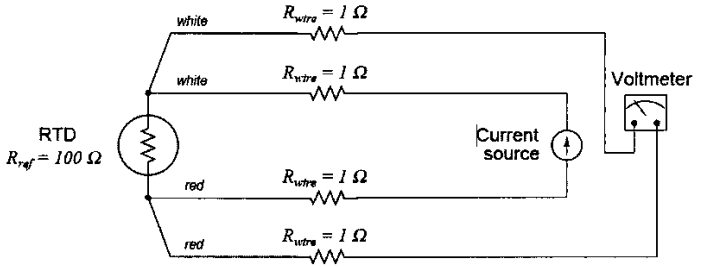
دو سیمه فقط برای مسافت‌های خیلی کم ممکن است به‌کار رود زیرا مقاومت سیم نیز در اندازه‌گیری دیده می‌شود.



شکل ۲-۲۹ اتصال RTD به‌صورت دو سیمه

چهار سیمه

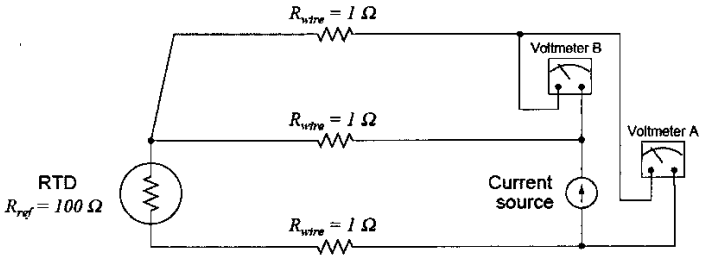
برای اندازه‌گیری دقیق به‌کار می‌رود در این حالت مانند شکل ۲-۳۰ یک منبع جریان ثابت بین دوسر قرار می‌گیرد و اندازه-گیری ولتاژ در دو سر دیگر صورت می‌گیرد. به‌علت اینکه از مسیر اندازه‌گیری ولتاژ جریان نمی‌گذرد، ولتاژ اندازه‌گیری شده ولتاژ دو سر مقاومت RTD است؛ پس در اندازه‌گیری مقاومت سیم‌ها دیده نمی‌شوند.



شکل ۲-۳۰ اتصال RTD به‌صورت چهار سیمه

سه سیمه

در سه سیمه خطا وجود دارد ولی از دو سیمه کمتر است زیرا اهم یک مسیر در اندازه‌گیری دیده می‌شود. اگر این خطا از نظر کاربر قابل قبول باشد می‌توان آنرا اجرا نمود.

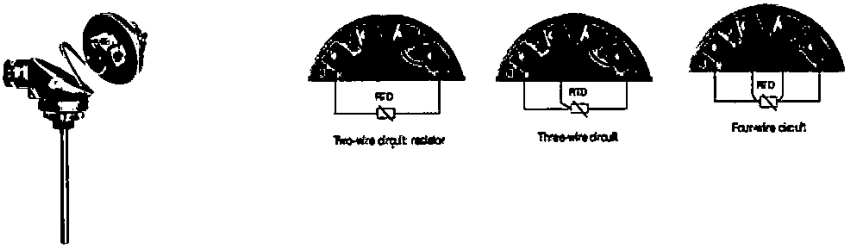


شکل ۲-۳۱ اتصال RTD به‌صورت سه سیمه

یکی از موارد دیگر که در کار اندازه‌گیری RTD خطا ایجاد می‌کند، گرمای ایجاد شده در خود مقاومت RTD به‌دلیل عبور جریان از آن است. این گرمای داخلی نیز روی میزان مقاومت اندازه‌گیری شده تأثیر می‌گذارد و خطا ایجاد می‌کند. به این پدیده Self Heating گفته می‌شود. یک روش برای حل این مشکل فوق استفاده از پالس جریانی است.

با توجه به اینکه کارت آنالوگ طبق زمان‌بندی خاصی اقدام به نمونه‌برداری می‌کند، لازم نیست که همیشه جریان در مقاومت تزریق شود بلکه فقط در لحظات نمونه‌برداری این کار انجام می‌گیرد. بنابراین جریان به‌صورت پالس است و میزان گرمای تولید شده کمتر از حالت عبور جریان مداوم می‌باشد.

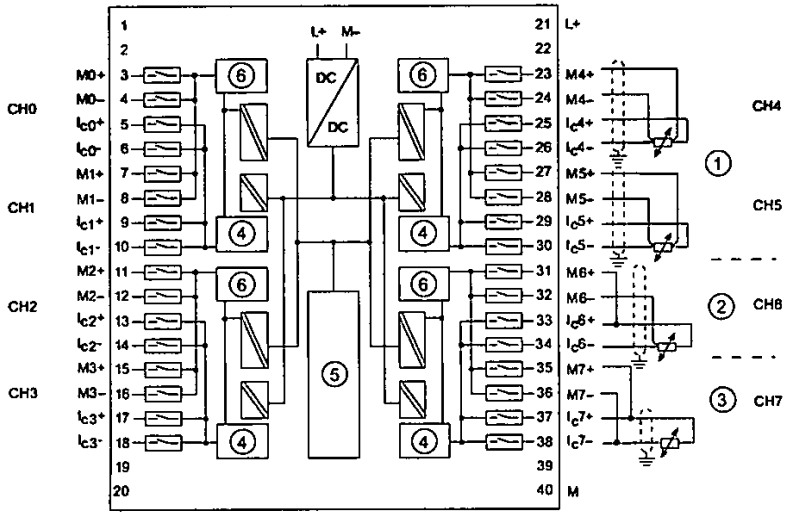
در برخی کاربردها سیگنال RTD را توسط ترانسمیتر به 4-20 mA تبدیل کرده و با دو سیم انتقال می‌دهند.



شکل ۲-۳۲ اتصال RTD به ترانسمیتر Head Mount

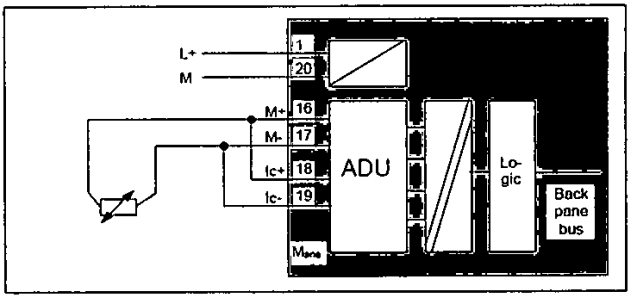
اتصال

شکل ۲-۳۳ نقشه اتصالات کارت AI 8 x RTD را نشان می‌دهد. اتصال چهار سیمه با شماره ۱، اتصال سه سیمه با شماره ۲ و اتصال دو سیمه با شماره ۴ مشخص شده است.



شکل ۲-۳۳ اتصال کارت AI 8 x RTD

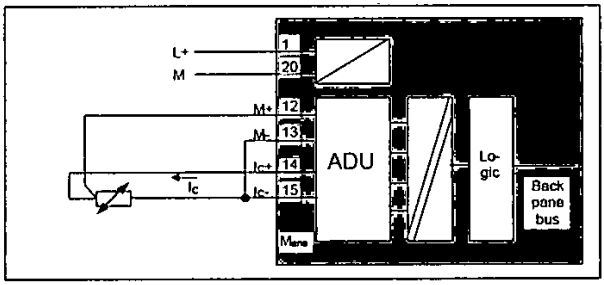
شکل ۲-۳۴ نحوه اتصال یک RTD دو سیمه به کارت AI را نشان می‌دهد.



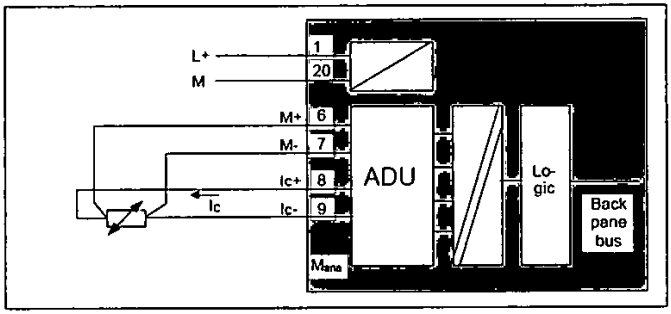
شکل ۲-۳۴ نحوه اتصال Pt100 دو سیمه به کارت AI

همانطور که در شکل ۲-۳۴ مشخص است، ابتدا توسط کارت جریان مشخصی به Pt100 اعمال شده و سپس افت ولتاژ بوجود آمده در Pt100 اندازه‌گیری می‌شود. با تقسیم ولتاژ اندازه‌گیری شده بر جریان اعمالی به Pt100، میزان مقاومت آن به‌دست می‌آید.

شکل۲-۳۵ اتصال یک Pt100 سه سیمه و در شکل ۲-۳۶ اتصال یک Pt100 چهار سیمه به کارت AI مشاهده می‌شود.



شکل۲-۳۵ اتصال Pt100 سه سیمه به کارت AI

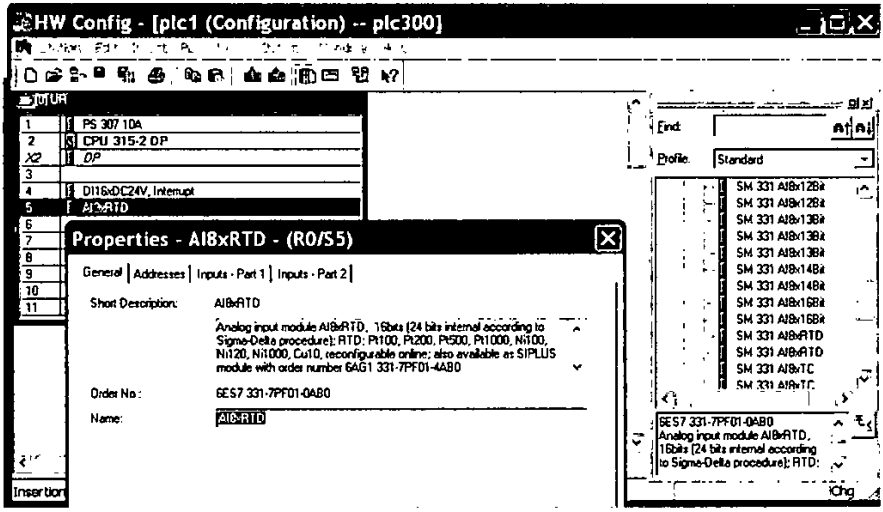


شکل ۲-۳۶ اتصال Pt100 سه سیمه به کارت AI

تنظیمات RTD در Hwconfig

کارت‌های متنوعی برای دریافت سیگنال‌های RTD وجود دارند. برخی از آنها چند منظوره هستند و سیگنال‌های دیگری را نیز قبول می‌کنند ولی برخی مانند کارت AI 8 x RTD به‌صورت خاص برای RTD طراحی شده‌اند.

در این بخش کارت خاص فوق مورد بررسی قرار می‌گیرد که پارامترهای بیشتری نسبت به کارت‌های معمولی دارد. با بیان این پارامترها فهم پارامترهای RTD در کارت‌های چند منظوره ساده خواهد بود.



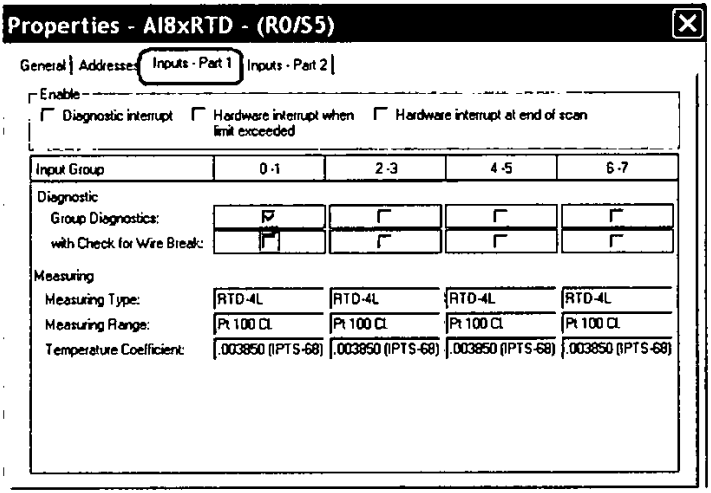
شکل ۲-۳۷ وارد کردن کارت AI8xRTD در Hwconfig

با دوبار کلیک نمودن روی این کارت، پنجره ای مانند شکل ۲-۳۷ ظاهر می‌شود که دارای چند سربرگ است. در سربرگ General توضیحاتی در مورد کارت دیده می‌شود. همانطور که می‌بینیم میزان دقت این کارت 16 bit می‌باشد.

در سربرگ‌های Input Part 1 و Input Part2 تنظیم پارامترهای اصلی این کارت انجام می‌شود:

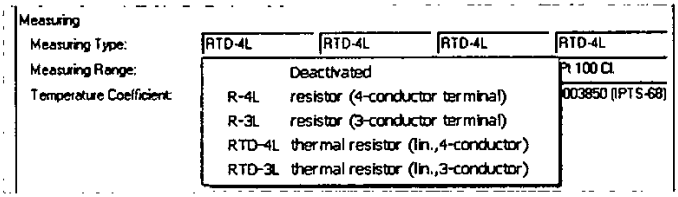
سربرگ Input Part 1

در این سربرگ پارامترهای مختلفی دیده می‌شود. بخش مربوط به وقفه‌ها و بخش مربوط به Diagnostic مشابه کارت ترموکوپل هستند که قبلاً توضیح داده شد.



شکل ۲-۳۸ تنظیمات سربرگ Input-Part1 کارت AI8xRTD

در بخش Measuring می‌توان نوع سنسور را انتخاب نمود.
این کارت علاوه بر RTD می‌تواند مقاومت نیز قبول کند و مشابه RTD به‌صورت سه سیمه یا چهار سیمه بسته شود.



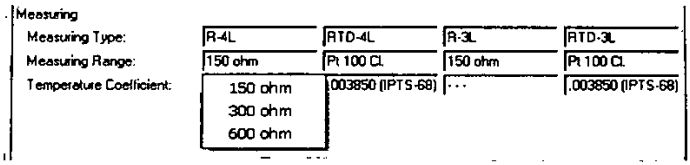
شکل ۲-۳۹ تنظیمات نوع سنسور در کارت AI8xRTD

پس از انتخاب RTD، اگر بخش Measuring Range را ملاحظه کنیم می‌بینیم که انواع بسیار متنوعی وجود دارد.

Pt 100 Cl.	Pt 100 climatic range
Pt 200 Cl.	Pt 200 climatic range
Pt 500 Cl.	Pt 500 climatic range
Pt 1000 Cl.	Pt 1000 climatic range
Pt 100 Std.	Pt 100 standard range
Pt 200 Std.	Pt 200 standard range
Pt 500 Std.	Pt 500 standard range
Pt 1000 Std.	Pt 1000 standard range
Ni 100 Cl.	Ni 100 climatic range
Ni 120 Cl.	Ni 120 climatic range
Ni 200 Cl.	Ni 200 climatic range
Ni 500 Cl.	Ni 500 climatic range
Ni 100 Std.	Ni 100 standard range
Ni 120 Std.	Ni 120 standard range
Ni 200 Std.	Ni 200 standard range
Ni 500 Std.	Ni 500 standard range
Cu 10 Cl.	Cu 10 climatic range
Cu 10 Std.	Cu 10 standard range
LG-Ni 1000K	LG-NI 1000 climatic range
LG-Ni 1000S	LG-Ni 1000 standard range
GOST-Pt 10K	Pt 10 GOST climatic range
GOST-Pt 50K	Pt 50 GOST climatic range
GOST-Pt 100K	Pt 100 GOST climatic range
GOST-Pt 500K	Pt 500 GOST climatic range
GOST-Pt 10S	Pt 10 GOST standard range
GOST-Pt 50S	Pt 50 GOST standard range
GOST-Pt 100S	Pt 100 GOST standard range
GOST-Pt 500S	Pt 500 GOST standard range
GOST-Cu 10K	Cu 10 GOST climatic range
GOST-Cu 50K	Cu 50 GOST climatic range
GOST-Cu 100K	Cu 100 GOST climatic range

شکل ۲-۴۰ انتخاب نوع RTD در تنظیمات کارت AI8xRTD

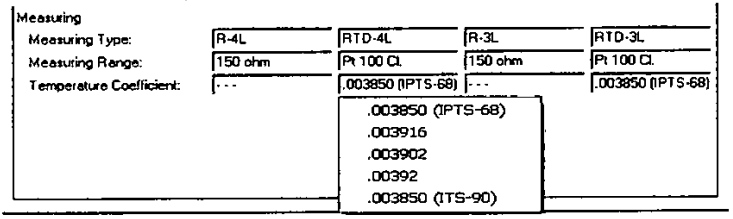
اگر مقاومت را انتخاب کنیم، این بخش فقط دارای سه انتخاب 600 , 300 , 150 اهم خواهد بود.



شکل ۲-۴۱ انتخاب نوع مقاومت در کارت AI8xRTD

در قسمت پایین پنجره گزینه Temeperature Coefficient وجود دارد که فقط برای RTD قابل استفاده است و ضریب تصحیح دما می‌باشد. این ضریب با توجه به نوع RTD دارای مقادیر و انتخاب‌های خاصی است. به‌عنوان نمونه:

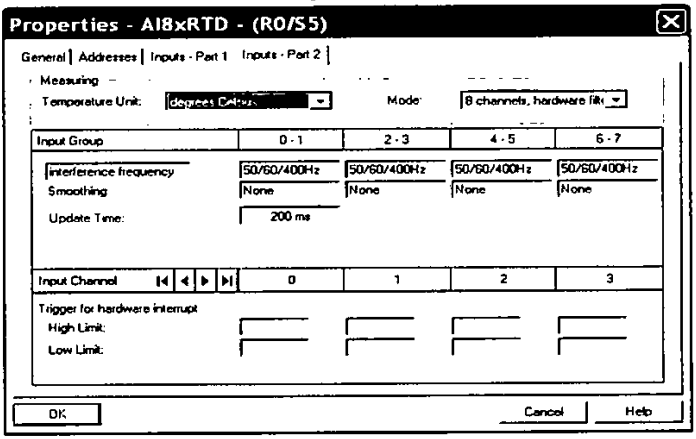
- برای PT 100 با توجه به استاندارد IEC 751، ضریب ۰,۰۰۳۸۵۰ انتخاب می‌گردد.
- برای NI 100 با توجه به استاندارد DIN 43760، ضریب ۰,۰۰۶۱۸ انتخاب می‌گردد.



شکل ۲-۴۲ ضریب تصحیح دما در کارت AI8xRTD

سربرگ Input Part 2

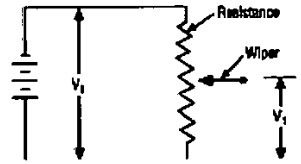
این سربرگ کاملاً مشابه آنچه برای ترموکوپل توضیح داده شد می‌باشد.



شکل ۲-۴۳ تنظیمات سربرگ Input-Part2 کارت AI8xRTD

ب) مقاومت متغیر

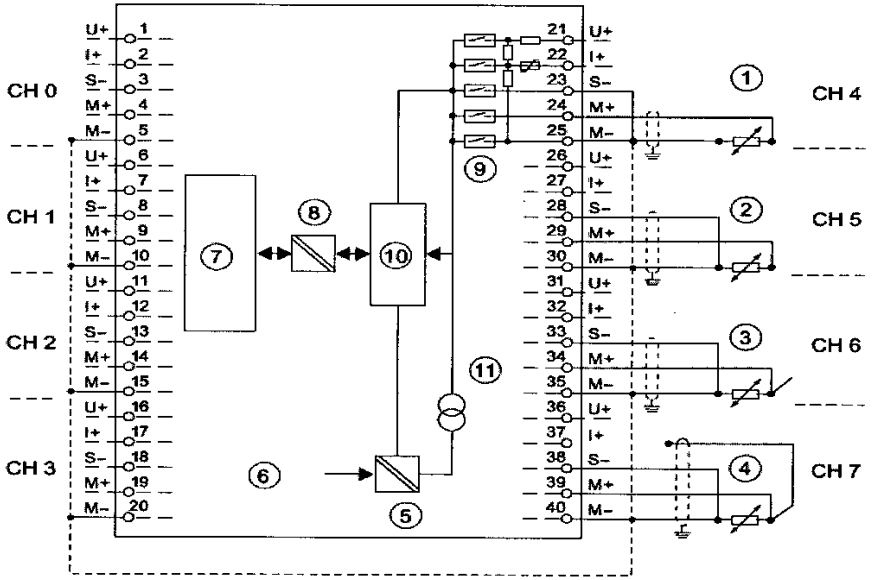
مقاومت متغیر برای کاربردهای دیگری به‌جز اندازه‌گیری دما به‌کار می‌رود. یکی از کاربردهای آن در اندازه‌گیری موقعیت وسایل متحرک است. به‌عنوان مثال می‌توان مشخص نمود که یک جک هیدرولیکی تا چه حد باز شده است.



شکل ۲-۴۴ کاربرد مقاومت متغیر بعنوان آنالوگ ورودی

مقاومت متغیر نیز می‌تواند به‌صورت ۲ سیمه یا ۳ سیمه یا ۴ سیمه بسته شود و رفتار آن و اتصالات آن مشابه مواردی است که برای RTD توضیح داده شد. کارت AI 8 x RTD که قبلاً تشریح شد، مقاومت متغیر را نیز قبول می‌کرد و این موضوع را در قسمت تنظیمات سخت افزار دیدیم. در اینجا کارت دیگری را برای تکمیل مطلب معرفی می‌کنیم.

شکل ۲-۴۵ کارت چند منظوره آنالوگ AI 8 x 13 bit را نشان می‌دهد که مقاومت متغیر به صورت‌های ۲ سیمه، ۳ سیمه و ۴ سیمه به آن متصل شده است.



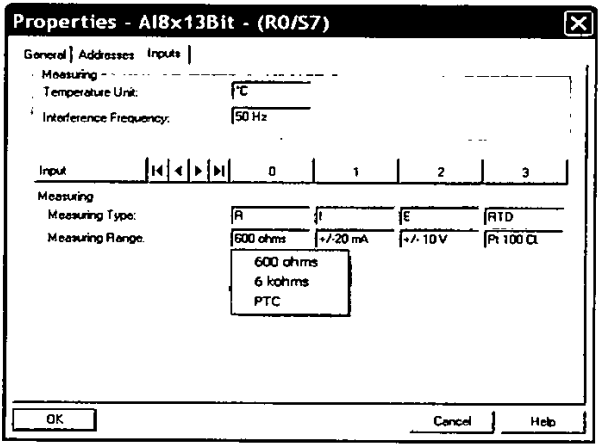
شکل ۲-۴۵ اتصالات کارت آنالوگ AI 8 x 13 bit

این کارت اگر چه می‌تواند انواع مختلف سیگنال‌های اهمی، جریانی، ولتاژی را پشتیبانی کند ولی از نظر برخی ویژگی‌ها ضعیف‌تر از کارت AI 8 x RTD است.

این کارت همانطور که در شکل مشخص است، فقط دارای یک مبدل A/D است در حالی که کارت قبلی دارای ۴ مبدل بود.

میزان دقت این کارت 13 bit است در حالی که کارت قبلی 16 bit بود.

اگر این کارت را به محیط Hwconfig وارد نموده و روی آن دوبار کلیک کنیم، شکل ۲-۴۶ را خواهیم داشت. همانطور که دیده می‌شود برخلاف کارت قبلی، این کارت فقط دارای یک سربرگ input است و تنظیمات کمتری نسبت به کارت قبلی دارد.



شکل ۲-۴۶ تنظیمات کارت آنالوگ AI 8 x 13 bit

این کارت همانطور که دیده می‌شود سیگنال جریانی، ولتاژی و RTD و اهمی را پشتیبانی می‌کند. اگر نوع اهمی را انتخاب کنیم مانند شکل ۲-۴۶ سه گزینه می‌بینیم که یکی از آنها مربوط به PTC است و دوگزینه دیگر مقاومت متغیر را تعیین می‌کند.

ج) PTC

PTC همانطور که در کتاب سطح مقدماتی تشریح شد، سنسورهای سیلیکونی هستند که برای تغییرات جزئی دما به‌کار می‌روند و مقاومت آنها با افزایش دما بالا می‌رود. یکی از کاربردهای مهم آنها در حفظ دمای موتورهای الکتریکی است.

PTC را می‌توان به کارت آنالوگ فوق متصل نموده و در تنظیم Hwconfig همانطور که در شکل ۲-۴۶ مشخص است، PTC را انتخاب نمود.

جدول‌های تبدیل A/D سنسورهای مقاومتی

جدول‌های ۲-۷

Resistive transducer range							
dec	hex	6kΩ	10 kΩ	150 Ω	300 Ω	600 Ω	
32767	7FFF	7.111 kΩ	11.852 kΩ	177.77 Ω	355.54 Ω	711.09 Ω	Overflow
32512	7F00			176.39 Ω	352.78 Ω	705.55 Ω	
32511	7EFF	7.055 kΩ	11.759 kΩ	176.38 Ω	352.77 Ω	705.53 Ω	Overshoot range
27649	6C01						
27648	6C00	6.0 kΩ	10 kΩ	150 Ω	300 Ω	600 Ω	
20736	5100	4.5 kΩ	7.5 kΩ	112.5 Ω	225 Ω	450 Ω	
1	1	217.0 mΩ	361.7 mΩ	5.43 mΩ	10.85 mΩ	21.70 mΩ	Rated range
0	0	0 Ω	0 Ω	0 Ω	0 Ω	0 Ω	
		مقادیر منفی بطور فیزیکی عملا غیر ممکن است					Undershoot range

Pt x00 Standard /GOST in °C (1 digit =0.1°C)			Pt x00 GOST Standard in °C (1 digit =0.1°C)			Pt x00 climatic/ GOST in °C (1 digit =0.01°C)			
°C	dec	hex	°C	Dec	hex	°C	Dec	hex	
> 1000,0	32767	7FFFH	> 1295,0	32767	7FFFH	> 155,00	32767	7FFFH	Overflow
1000,0 : 850,1	10000 : 8501	2710H : 2135H	1295,0 : 1100,1	12950 : 11001	3296H : 2AF9H	155,00 : 130,01	15500 : 13001	3C8CH : 32C9H	Overshoot range
850,0 : -200,0	8500 : -2000	2134H : F830H	1100,0 : -260,0	11000 : -2600	2AF8H : F5D8H	130,00 : -120,00	13000 : -12000	32C8H : D120H	Rated range
-200,1 : -243,0	-2001 : -2430	F82FH : F682H	-260,1 : -273,2	-2601 : -2732	F5D7H : F554H	-120,01 : -145,00	-12001 : -14500	D11FH : C75CH	Undershoot range
< - 243,0	-32768	8000H	< - 273,2	-32768	8000H	< - 145,00	-32768	8000H	Underflow

Ni x00 Standard in °C (1 digit =0.1°C)			Ni 100 GOST Standard in °C (1 digit =0.1°C)			Ni x00 climatic in °C (1 digit =0.01°C)			
°C	dec	hex	°C	Dec	hex	°C	Dec	hex	
> 295,0	32767	7FFFH	> 212,4	32767	7FFFH	> 295,00	32767	7FFFH	Overflow
295,0 : 250,1	2950 : 2501	B86H : 9C5H	212,4 : 180,1	2124 : 1801	084CH : 0709H	295,00 : 250,01	29500 : 25001	733CH : 61A9H	Overshoot range
250,0 : -60,0	2500 : -600	9C4H : FDA8H	180,0 : -60,0	1800 : -600	0708H : FDA8H	250,00 : -60,00	25000 : -6000	61A8H : E890H	Rated range
-60,1 : -105,0	-601 : -1050	FDA7H : FBE6H	-60,1 : -105,0	-601 : -1050	FDA7H : FBE6H	-60,01 : -105,00	-6001 : -10500	E88FH : D6FCH	Undershoot range
< -105,0	-32768	8000H	< - 105,0	-32768	8000H	< - 105,00	-32768	8000H	Underflow

PTC KTY83/110 in °C (1 digit =0.1 °C)			PTC KTY84/130 in °C (1 digit =0.1 °C)			
°C	dec	hex	°C	dec	hex	
> 206,3	32767	7FFFH	> 352,8	32767	7FFFH	Overflow
206,3 : 175,1	2063 : 1751	080FH : 06D7H	352,8 : 300,1	3528 : 3001	0DC8H : 0BB9H	Overshoot range
175 : -55	1750 : -550	06D6H : FDDAH	300 : -40	3000 : -400	0BB8H : FE70H	Rated range
-55,1 : -64,7	-551 : -647	FDD9H : FD79H	-40,1 : -47,0	-401 : -470	FE6FH : FE2AH	Undershoot range
< -64,7	-32768	8000H	< -47,0	-32768	8000H	Underflow

۲-۲-۴ کار با سیگنال‌های جریانی

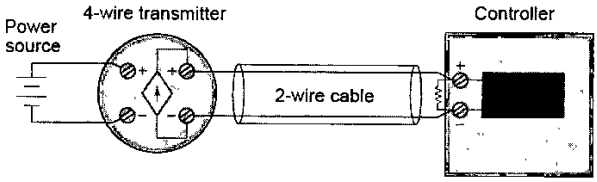
سیگنال‌های جریانی معمولاً از ترانسمیترها دریافت می‌شوند و دارای رنج‌های مختلف استاندارد هستند از جمله:

- 0-20 mA
- +/- 20 mA
- 4-20mA
- +/- 3.2 mA
- +/- 10 mA

در بین این سیگنال‌ها 4-20 mA از همه مرسوم‌تر است زیرا امکان تشخیص قطعی سیم در آن وجود دارد. ترانسمیترهای جریانی می‌توانند به دو طریق چهار سیمه و دوسیمه به کارت AI متصل شوند.

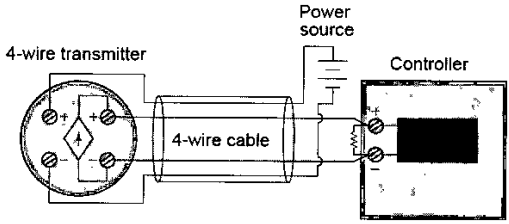
الف) ترانسمیترهای جریانی چهار سیمه Self Powered

اگر ترانسمیتر دارای تغذیه مجزا باشد، مانند شکل ۲-۴۷ چهار سیمه بسته می‌شود و به آن Self Powered نیز می‌گویند.



شکل ۲-۴۷ اتصال ترانسمیتر جریانی ۴ سیمه با تغذیه در محل

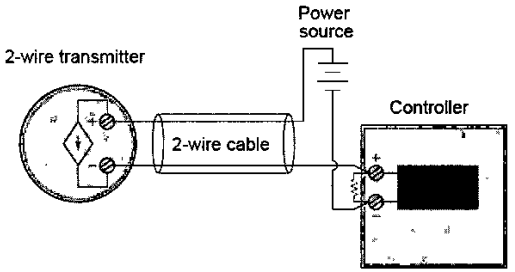
در برخی کاربردها تغذیه از سمت کنترلر برقرار می‌شود، بنابراین مانند شکل ۲-۴۸ کابل‌کشی بیشتری لازم است.



شکل ۲-۴۸ اتصال ترانسمیتر جریانی ۴ سیمه با تغذیه از سمت کنترلر

ب) جریانی ۲ سیمه (Loop Powered)

در این حالت تغذیه و سیگنال روی یک زوج سیم قرار دارند. به جهت فلش روی ترانسمیتر دقت کنید و آنرا با حالت ۴ سیمه مقایسه کنید. در ۴ سیمه ترانسمیتر یک منبع جریان است ولی در این حالت مشابه یک بار مصرف کننده است.



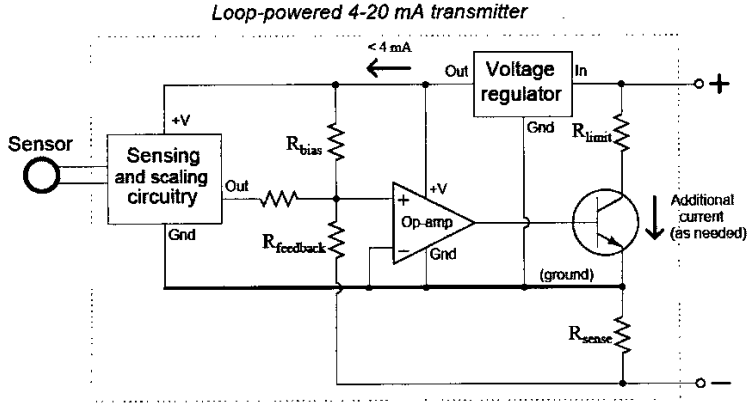
شکل ۲-۴۹ اتصال ترانسمیتر جریانی ۲ سیمه

معمولاً کنترلر جریان 4-20 mA را از مقاومتی عبور می‌دهند تا ولتاژ استاندارد 1-5V در ورودی ظاهر شود. این مقاومت ۲۵۰ اهم است.

با توجه به شکل و با در نظر گرفتن منبع تغذیه ۲۴ ولتی و افت ۵ ولت روی ترمینال کارت ورودی کنترلر ۱۹ ولت برای ترانسمیتر قابل استفاده است. بنابراین حتی در حداقل مقدار جریان که ۴ میلی‌آمپر است توان کافی برای عملکرد ترانسمیتر در ترمینال‌های آن وجود دارد.

مدار داخلی ترانسمیتر به‌صورت شکل ۲-۵۰ طراحی می‌شود تا بتواند در کمتر از ۴ میلی‌آمپر کار کند. بدیهی است برای ارسال جریان ناشی از اندازه‌گیری سیگنال از یک ترانزیستور شانت استفاده شده است که جریان اضافی مورد نیاز (معرف سیگنال اندازه گیری شده) را بر می‌گرداند.

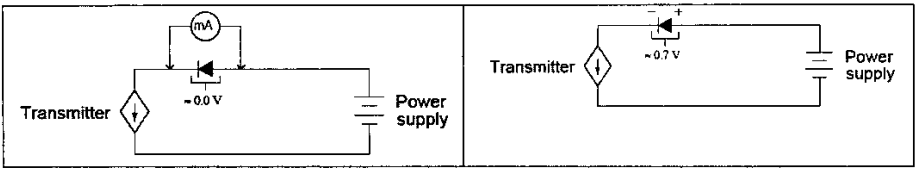
به این ترتیب جریان داخلی ترانسمیتر 4 mA و جریان ناشی از تغییرات اندازه‌گیری بین 0-16 mA خواهد بود. مجموع این جریان از منبع تغذیه سمت کنترلر کشیده می‌شود بنابراین تغییرات بین 4-20 mA خواهد بود.



شکل ۲-۵۰ مدار داخلی ترانسمیتر جریانی

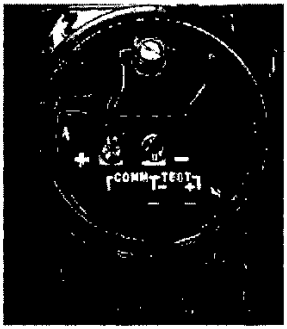
وقتی ترانسمیتر جریانی ۲ سیمه به کنترلر متصل و سیستم در حال کار است، در صورت نیاز به اندازه‌گیری جریان نباید مدار قطع شود. برای این منظور می‌توان از آمپرمتر کلمپی استفاده کرد.

روش دیگر این است که دیودی را به‌صورت سری در مدار قرار می‌دهند. این دیود در هنگام اندازه‌گیری آمپرمتر مسیر را اتصال کوتاه[1] می‌کند. اهم مقاومت کمتر از یک اهم باید باشد.



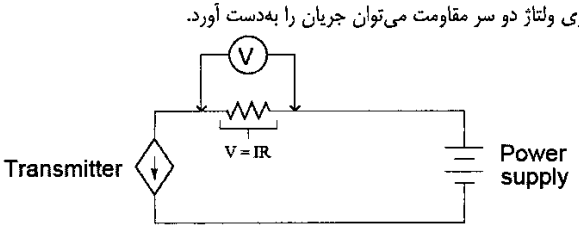
شکل ۲-۵۱ اندازه گیری جریان بدون قطع مدار ترانسمیتر

برخی سازندگان دیود فوق را به‌صورت داخلی در ترانسمیتر قرار داده‌اند و دو سر آن را با عنوان ترمینال‌های Test در دسترس گذاشته‌اند؛ مانند شکل ۲-۵۲ که ترانسمیتر اختلاف فشار مدل 3051ساخت Rosemount را نشان می‌دهد.



شکل ۲-۵۲ ترمینال‌های TEST برای اندازه‌گیری جریان ترانسمیتر در حین کار

ممکن است به‌جای دیود فوق یک مقاومت با اهم بسیار کم (کمتر از یک اهم) قرار گیرد که به آن مقاومت شنت می‌گویند. با اندازه‌گیری ولتاژ دو سر مقاومت می‌توان جریان را به‌دست آورد.

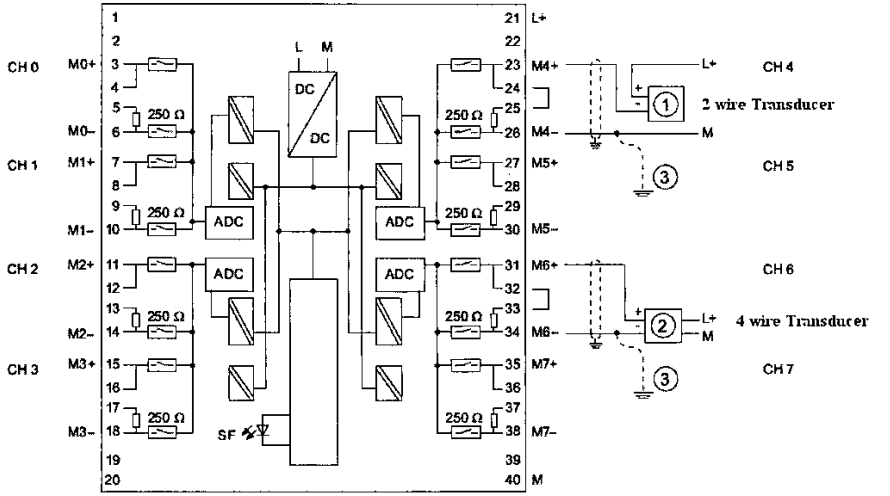


شکل ۲-۵۳ استفاده از مقاومت شنت برای اندازه‌گیری جریان ترانسمیتر در حین کار

1. Bypass

اتصالات

شکل ۲-۴۳ اتصال ترانسمیتر دو سیمه و چهار سیمه را برای کارت AI 8 x 12 bit که یک کارت چند منظوره بوده و سیگنال‌های ولتاژی و ترموکوپل و اهمی را نیز قبول می‌کند نشان می‌دهد.



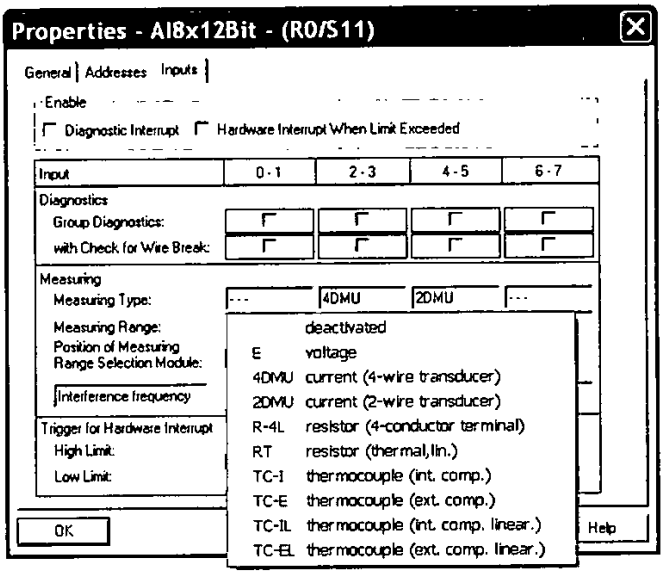
شکل ۲-۵۴ اتصالات کارت AI 8 x 12 bit

در شکل ۲-۵۴ در تمام کانال‌ها یک مقاومت 250 اهم دیده می‌شود. با عبور جریان ولتاژی در دو سر این مقاومت ظاهر می‌شود که کارت از آن استفاده می‌کند. به‌عنوان مثال با عبور جریان 4-20mA ولتاژ 1-5VDC در دوسر مقاومت خواهیم داشت.

اکنون کارت AI 8 x 16 bit را در نظر بگیرید. این کارت برخلاف کارت قبلی نیاز به تغذیه بیرونی ندارد. همانطور که در شکل ۲-۵۵ مشاهده می‌شود، کانال‌های روبرو به هم متصل می‌شوند و تغذیه مدار داخلی کارت از همین طریق برقرار می‌شود.

تنظیمات در HWconfig

اگر کارت AI 8 x 12 bit را در Hwconfig وارد کنیم و روی آن دوبار کلیک نماییم پنجره ای مانند شکل ۲-۵۴ باز می‌شود.

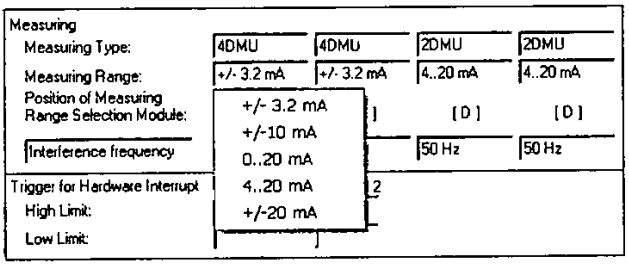


شکل ۲-۵۵ تنظیمات کارت AI 8 x 12 bit

تنظیمات جریانی دوسیمه 2DMU یا چهار سیمه 4DMU در قسمت Measuring Type انجام می‌شود.

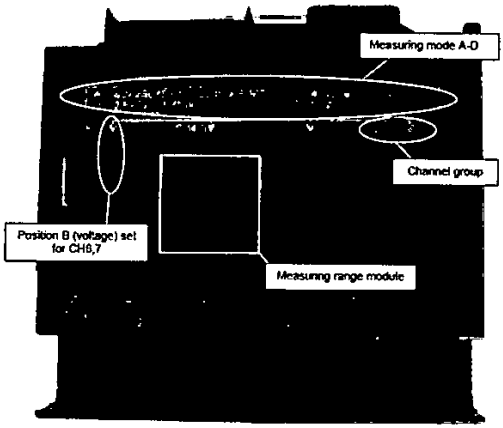
اگر تنظیم را روی دو سیمه قرار دهیم Measuring Range فقط 4-20 mA خواهد بود.

اگر تنظیم را روی چهار سیمه قرار دهیم Measuring Range گزینه‌های مختلف مانند شکل زیر قابل انتخاب خواهد بود.



شکل ۲-۵۶ تنظیمات جریانی کارت AI 8 x 12 bit

نکته‌ای که در مورد این کارت و برخی دیگر از کارت‌های چند منظوره وجود دارد گزینه Selection Module است. در کتاب مقدماتی تشریح شد که برخی کارت‌ها دارای ماژول کوچکی هستند که بسته به نوع سنسور بایستی در موقعیت A یا B یا C یا D تنظیم شود. این موقعیت در تنظیمات Hwconfig نیز مشخص است و در شکل ۲-۵۶ برای کانال‌های آخر حرف D ذکر شده است.



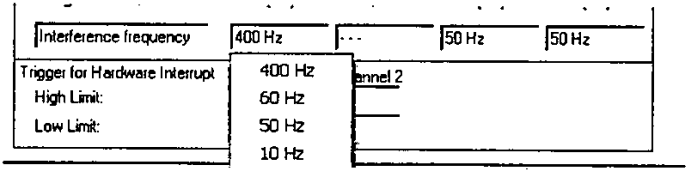
شکل۲-۵۷ ماژول تنظیم رنج در کارت AI

با توجه به نوع و رنج سیگنال ورودی کارت، باید ماژول تنظیم رنج را در یکی از حالت‌های چهارگانه فوق تنظیم نمود. جدول ۲-۸ نحوه انتخاب کد صحیح را نشان می‌دهد.

جدول ۲-۸ انتخاب کد صحیح

Position	Measurement type
A	Thermo couple / Resistance measurement
B	Voltage (default setting)
C	Current (4 wire transducer)
D	Current (2 wire transducer)

پارامتر دیگری که در تنظیمات این کارت دیده می‌شود Interference Frequency است.



شکل ۲-۵۸ تنظیمات حد تفکیک کارت

این گزینه با توجه به حد تفکیک کارت انتخاب می‌گردد. جدول ۲-۹ چگونگی انتخاب InterferenceFrequency را با توجه به حد تفکیک[1] کارت نشان می‌دهد. همانطور که دیده می‌شود، امکان Overrange یعنی 14 bit نیز برای این کارت وجود دارد.

جدول ۲-۹

Interference Frequency Suppression (Hz)	Integration Time (ms)	Resolution (bits)
400	2.5	9+sign bit
60	16.6	12+sign bit
50	20	12+sign bit
10	100	14+sign bit

جدول‌های تبدیل A/D سیگنال‌های آنالوگ ورودی جریانی

جدول‌های ۲-۱۰

Current measuring range					
dec	hex	±20 mA	±10 mA	±3.2 mA	
32767	7FFF	23.70 mA	11.85 mA	3.79 mA	Overflow
32512	7F00				
32511	7EFF	23.52 mA	11.76 mA	3.76 mA	Overshoot range
27649	6C01				
27648	6C00	20 mA	10 mA	3.2 mA	
20736	5100	15 mA	7.5 mA	2.4 mA	
1	1	723.4 nA	361.7 nA	115.7 nA	
0	0	0 mA	0 mA	0 mA	Rated range
-1	FFFF				
-20736	AF00	-15 mA	-7.5 mA	-2.4 mA	
-27648	9400	-20 mA	-10 mA	-3.2 mA	
-27649	93FF				Undershoot range
-32512	8100	-23.52 mA	-11.76 mA	-3.76 mA	
-32513	80FF				Underflow
-32768	8000	-23.70 mA	-11.85 mA	-3.79 mA	

1. Resolution

Current measuring range				
dec	hex	0 mA to 20 mA	4 mA to 20 mA	
32767 32512	7FFF 7F00	23.70 mA	22.96 mA	Overflow
32511 27649	7EFF 6C01	23.52 mA	22.81 mA	Overshoot range
27648 20736 1 0	6C00 5100 1 0	20 mA 15 mA 723.4 nA 0 mA	20 mA 16 mA 4 mA + 578.7 nA 4 mA	Rated range
-1 -4864 -4865 -32768	FFFF ED00 ECFF 8000	-3.52 mA	1.185 mA	Undershoot range Underflow

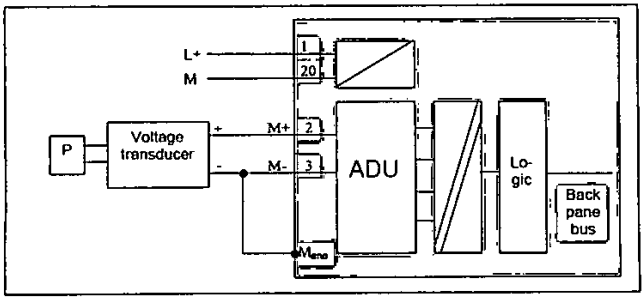
۲-۲-۵ کار با سیگنال‌های ولتاژی

برخی ترانسمیترها دارای خروجی ولتاژی هستند. سیگنال‌های آنالوگ ولتاژی استاندارد عبارتند از:

- 0-10V
- +/- 10V
- 1-5V
- +/- 5V
- +/- 500mV
- +/- 2.5V

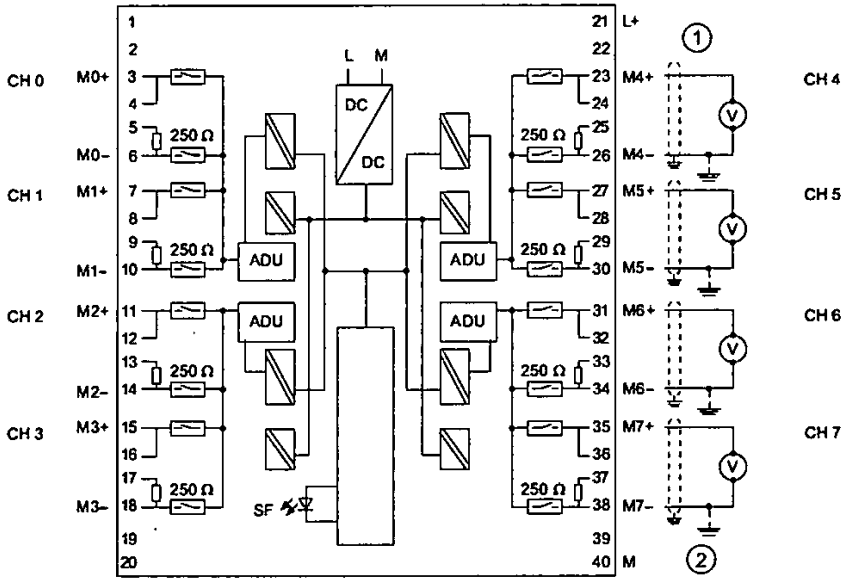
سیگنال‌های ولتاژی در مقایسه با سیگنال‌های جریانی معایبی دارند از جمله:

- در نوع ولتاژی اهم مسیر منجر به افت می‌شود ولی جریانی افت ندارد.
- در نوع ولتاژی تأثیر نویز بیشتر از جریانی است.
- در نوع 4-20 mA امکان تشخیص قطعی سیم هست ولی در نوع ولتاژی خیر. به همین علت استفاده از جریانی 4-20 mA متداول‌تر از نوع ولتاژی است.



شکل ۲-۵۹ اتصال ترانسمیتر ولتاژی به کارت آنالوگ ورودی

اگر کارت AI 8 x 12 bit که قبلاً تشریح شد را بخواهیم به‌صورت ولتاژی استفاده کنیم اتصالات به‌صورت شکل ۲-۶۰ است.

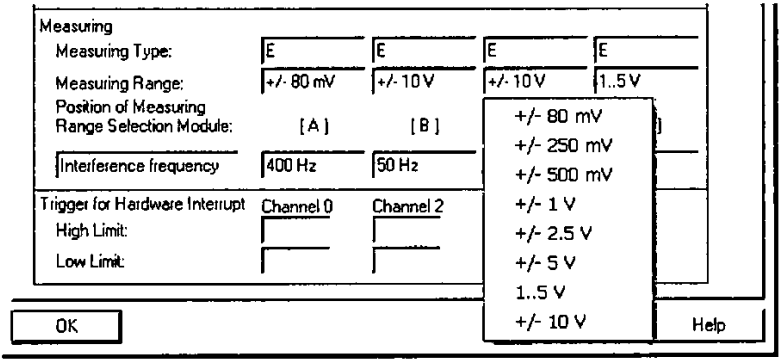


شکل ۲-۶۰ اتصالات کارت AI 8x12 bit

همانطور که دیده می‌شود ولتاژی همیشه به‌صورت دو سیمه بسته می‌شود.

تنظیم در HWconfig

اگرهمان کارت‌قبلی‌یعنی AI 8 x 12 bit را مدنظرقرار دهیم، با انتخاب ولتاژ گزینه‌هایی مانند شکل ۲-۶۱ ظاهر می‌گردد.



شکل ۲-۶۱ تنظیمات ولتاژ در کارت آنالوگ

توجه شود که Selection Module برای ولتاژ نیز بایستی در موقعیت صحیح قرار گیرد.

جدول‌های تبدیل A/D سیگنال‌های آنالوگ ورودی ولتاژی

این جدول‌های در ادامه آورده شده‌اند. نکته‌ای که در این جدول‌ها بایستی به آن دقت کرد این است که برخی مقادیر معادل برای تمام کارت‌ها با Resolutionهای مختلف یکسان هستند. به‌عنوان مثال معادل سیگنال 20 mA عدد 6C00 هگز است یعنی ۸ بیت با ارزش کمتر صفر است؛ بنابراین اگر قدرت تفکیک کارت حتی ۸ بیتی باشد باز به ازای 20 mA همین عدد را به CPU می‌فرستد و برای کارت‌های 12bit , 13 bit , 14 bit , 16 bit نیز معادل 20 mA همین عدد خواهد بود. این نکته برای سیگنال‌های ولتاژی نیز صادق است.

جدول‌های ۲-۱۱

Voltage measuring range						
dec	hex	±10 V	±5 V	±2.5 V	±1 V	
32767	7FFF	11.851 V	5.926 V	2.963 V	1.185 V	Overflow
32512	7F00					
32511	7EFF	11.759 V	5.879 V	2.940 V	1.176 V	Overshoot range
27649	6C01					
27648	6C00	10 V	5 V	2.5 V	1 V	
20736	5100	7.5 V	3.75 V	1.875 V	0.75 V	
1	1	361.7 µV	180.8 µV	90.4 µV	36.17 µV	
0	0	0 V	0 V	0 V	0 V	Rated range
-1	FFFF					
-20736	AF00	-7.5 V	-3.75 V	-1.875 V	-0.75 V	
-27648	9400	-10 V	-5 V	-2.5 V	-1 V	
-27649	93FF					Undershoot range
-32512	8100	-11.759 V	-5.879 V	-2.940 V	-1.176 V	
-32513	80FF					Underflow
-32768	8000	-11.851 V	-5.926 V	-2.963 V	-1.185 V	

Voltage measuring range					
dec	hex	±500 mV	±250 mV	±80 mV	
32767	7FFF	592.6 mV	296.3 mV	94.8 mV	Overflow
32512	7F00				
32511	7EFF	587.9 mV	294.0 mV	94.1 mV	Overshoot range
27649	6C01				
27648	6C00	500 mV	250 mV	80 mV	Rated range
20736	5100	375 mV	187.5 mV	60 mV	

1 0 -1 -20736 -27648	1 0 FFFF AF00 9400	18.08 µV 0 mV -375 mV -500 mV	9.04 µV 0 mV -187.5 mV -250 mV	2.89 µV 0 mV -60 mV -80 mV	
-27649 -32512	93FF 8100	 -587.9 mV	 -294.0 mV	 -94.1 mV	Undershoot range
-32513 -32768	80FF 8000	 -592.6 mV	 -296.3 mV	 -94.8 mV	Underflow

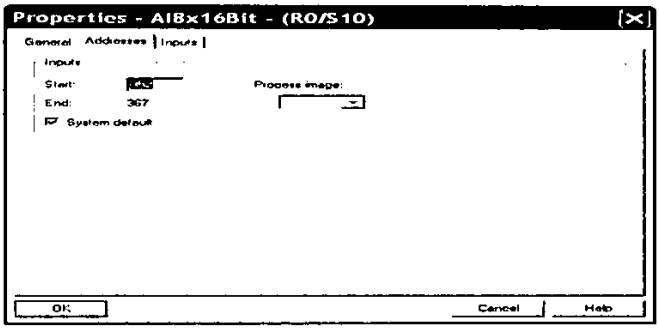
Voltage measuring range				
dec	hex	1 V to 5 V	0 V to 10 V	
32767 32512	7FFF 7F00	5.741 V	11.852 V	Overflow
32511 27649	7EFF 6C01	5.704 V	11.759 V	Overshoot range
27648 20736 1 0	6C00 5100 1 0	5 V 4 V 1 V + 144.7 µV 1 V	10 V 7.5 V 0 V + 361.7 µV 0 V	Rated range
-1 -4864	FFFF ED00	0.296 V	Negative values are not supported	Undershoot range
-4865 -32768	ECFF 8000			Underflow

۲-۳ آدرس‌دهی آنالوگ ورودی

۲-۳-۱ کلیات آدرس‌دهی آنالوگ ورودی

همانطور که قبلاً توضیح داده شد، سیگنال آنالوگ ورودی در کارت آنالوگ از طریق یک مبدل A/D به ۱۶ بیت یا یک Word تبدیل می‌گردد. بنابراین آنالوگ ورودی همواره به‌صورت Word آدرس‌دهی می‌شود.

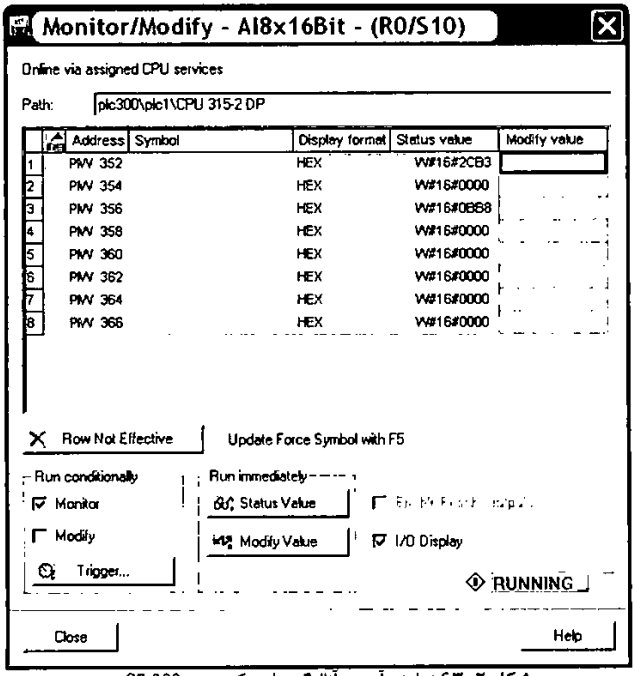
آدرس شروع در تنظیمات کارت آنالوگ قابل انتخاب است. اگر در Hwconfig روی هر کارت آنالوگ ورودی دو بار کلیک کنیم سربرگ Address در آن وجود دارد. شکل ۲-۶۲ این سربرگ را برای کارت AI 8 x 16 bit نشان می‌دهد. همانطور که مشاهده می‌شود آدرس از ۳۵۲ شروع و به ۳۶۷ ختم شده و فاصله بین این دو عدد ۱۶ بایت معادل 8 Word است.



شکل ۲-۶۲ سربرگ Address در کارت آنالوگ ورودی

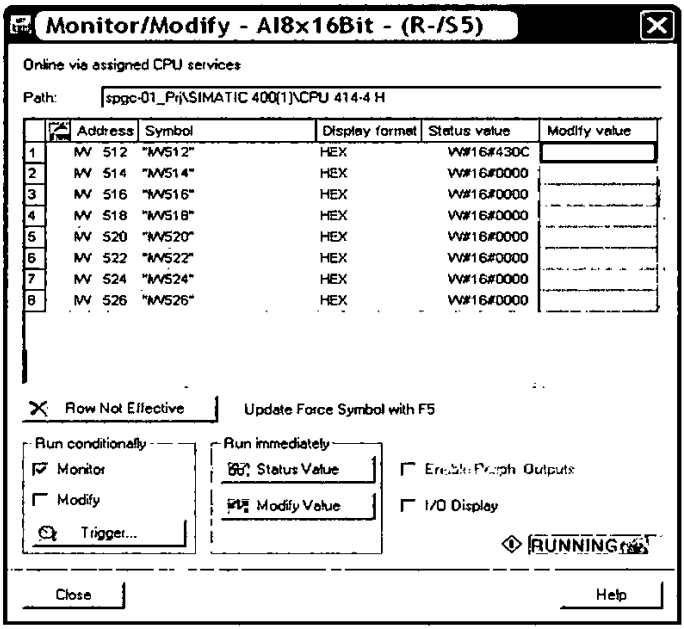
به‌طور معمول آدرس‌ها توسط سیستم انتخاب می‌شود و همانطور که در شکل ۲-۶۲ دیده می‌شود، گزینه System default فعال است. با غیر فعال کردن این گزینه کاربر می‌تواند هر آدرس دلخواهی را به‌عنوان آدرس شروع وارد کند به‌شرط اینکه این آدرس قبلاً استفاده نشده باشد.

بهترین روش برای پیدا کردن آدرس آنالوگ آن است که وقتی کامپیوتر به PLC متصل است در محیط HWConfig روی کارت آنالوگ با کلیک راست پنجره Monitor / Modify را فعال کنیم. این پنجره نه تنها آدرس آنالوگ را نشان می‌دهد بلکه با فعال کردن چک‌باکس Monitor، مقدار لحظه‌ای سیگنال را به‌صورت عدد هگز نیز نمایش می‌دهد.



شکل ۲-۶۳ نمایش آدرس آنالوگ برای یک نمونه S7-300

شکل ۲-۶۴ پنجره Monitor / Modify را برای کارت آنالوگ یک سیستم S7-400 نشان می‌دهد.
در هر دو شکل ۲-۶۳ و ۲-۶۴ دیده می‌شود که آدرس‌ها به‌صورت زوج هستند و فاصله بین آنها دو بایت می‌باشد.



شکل ۲-۶۴ نمایش آدرس آنالوگ برای یک نمونه S7-400

اما نکات سوال برانگیزی که در مورد آدرس آنالوگ وجود دارد این است که:

- چرا در برخی موارد آدرس به‌صورت IW و در برخی موارد به‌صورت PIW ظاهر می‌شود؟ این موضوع در دو شکل بالا دیده می‌شود.
- آیا می‌توان کاری کرد که آدرس PIW به فرم IW ظاهر شود؟ برعکس چطور؟

برای پاسخ به سوالات بالا لازم است به موارد زیر توجه شود:

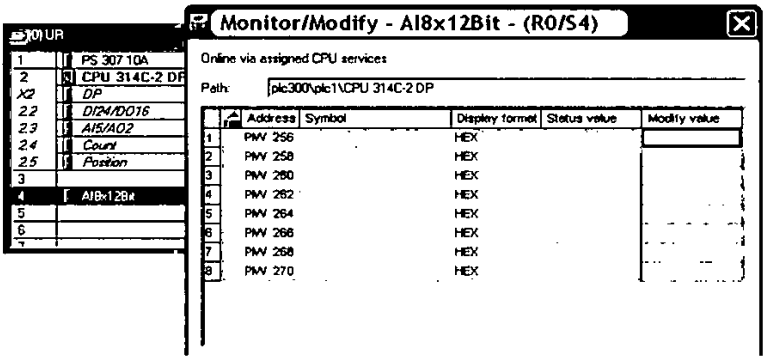
- حرف P در ابتدای PIW معرف Peripheral است.
- آدرس‌هایی که در ناحیه حافظه ورودی PII قرار می‌گیرند توسط سیستم به‌صورت IW نشان داده می‌شوند.
- آدرس‌هایی که در خارج از ناحیه PII قرار می‌گیرند توسط سیستم به‌صورت PIW نمایش داده می‌شوند.
- آدرس‌های IW از کارت خوانده شده و تصویر آنها در PII ذخیره می‌شود. در برنامه‌نویسی هرجا به آدرس IW اشاره شود مقدار آن از حافظه PII گرفته می‌شود.
- آدرس‌های PIW چون خارج از بازه حافظه هستند، بنابراین تصویری از آنها در PII قرار نمی‌گیرد. در برنامه‌نویسی هرجا به PIW اشاره شود مستقیماً و به‌صورت Peripheral از کارت خوانده می‌شود.
- آدرس‌هایی که به‌صورت IW هستند را در برنامه‌نویسی می‌توان به‌صورت PIW آدرس‌دهی کرد.

- آدرس‌هایی که به‌صورت PIW هستند را در برنامه نویسی نمی‌توان به‌صورت IW آدرس‌دهی کرد.
- اگر کارتی که آدرس آن به‌صورت IW خوانده شده دچار مشکل شود، از آنجا که نظیر آدرس در حافظه وجود دارد برای CPU مشکلی پیش نمی‌آید و فقط مقدار سیگنال تغییر نمی‌کند.
- اگر کارتی که آدرس آن به‌صورت PIW خوانده شده دچار مشکل شود، CPU را با مشکل مواجه می‌کند. چراغ‌های فالت روشن و پیغام I/O Acces Error در بافر ثبت می‌شود. در این شرایط اگر وقفه مربوط به این خطا وجود نداشته باشد CPU به مد Stop می‌رود.

برای فهم بهتر نکات کلیدی فوق موضوع را با چند حالت مختلف تشریح می‌کنیم.

۲-۳-۲ آدرس‌دهی آنالوگ ورودی در S7-300

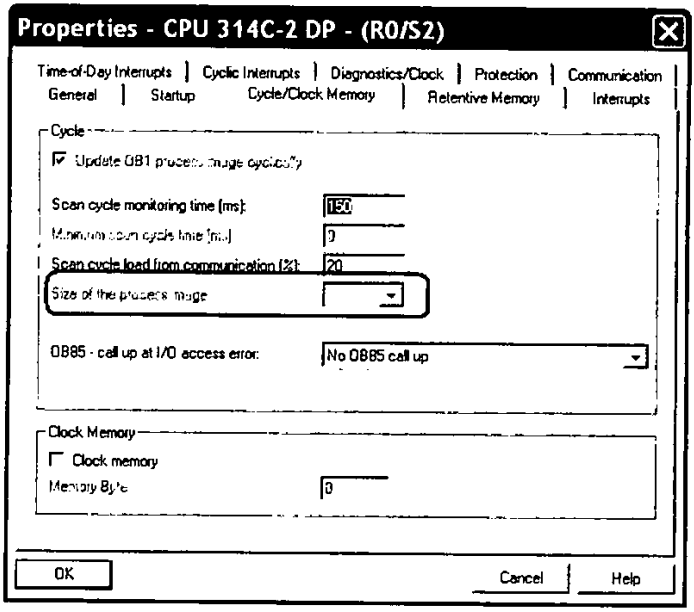
فرض کنید در پروژه ای در کنار CPU314C-2DP یک کارت آنالوگ AI 8x 12 bit قرار گرفته است. پس از پیکر-بندی این کارت در Hwconfig و مشاهده آدرس توسط Monitor/Modify مشاهده می‌کنیم که همه آدرس‌ها به-صورت PIW هستند.



شکل ۲-۶۵ آدرس آنالوگ برای یک نمونه S7-300

علت این است که CPU314C-2DP ماکزیمم ۱۲۸ بایت فضا برای ناحیه PII دارد و آدرس ۲۵۶ که به‌عنوان آدرس شروع به آنالوگ ورودی داده شده خارج از بازه PII قرار می‌گیرد.

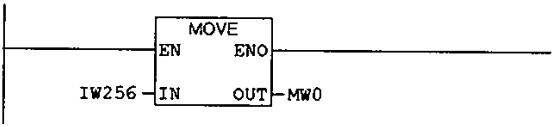
در این CPU ناحیه PII دارای فضای ثابتی است و قابل تغییر نیست یعنی نمی‌توان آنرا به‌صورت نرم‌افزاری افزایش داد تا آنالوگ ورودی در داخل فضای PII قرار گیرد. غیر قابل تغییر بودن PII را می‌توان با دوبار کلیک روی CPU در بخش Cycle / Clock مانند شکل ۲-۶۶ مشاهده کرد.



شکل ۲-۶۶ عدم امکان افزایش PII برای CPU314C-2DP

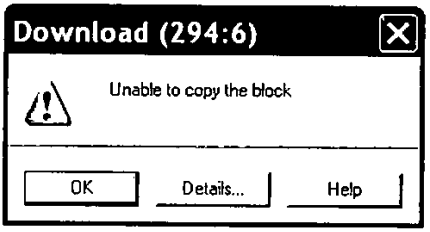
در این شرایط می‌توانیم تست‌های زیر را انجام دهیم.

تست ۱: در برنامه OB1 آدرس آنالوگ فوق را بدون حرف P به‌صورت IW مانند مثال زیر به‌کار می‌بریم.



شکل ۲-۶۷ خطا در برنامه‌نویسی به‌دلیل حذف P در آدرس‌دهی

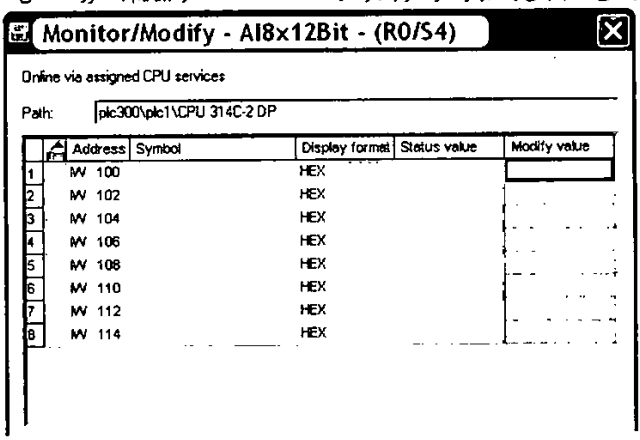
وقتی این برنامه را ذخیره و دانلود می‌کنیم با پیغام خطای زیر مواجه شده و چراغ فالت SF روی CPU روشن می‌گردد. در واقع CPU نتوانسته است این برنامه را بپذیرد.



شکل ۲-۶۸ پیغام خطای آدرس غیر مجاز برای CPU 300

این پیغام خطا در موارد دیگر که شماره تایمر یا کانتر یا M یا Q نیز غیر مجاز باشد ظاهر می‌گردد.

تست ۲: روی کارت آنالوگ دوبار کلیک کرده و در سربرگ Address با غیر فعال کردن System default آدرس شروع را 100 وارد می‌کنیم. پس‌ازذخیره‌ودانلوداگر پنجره Monitor/Modify را ببینیم به‌صورت شکل ۲-۶۹ خواهد بود.



شکل ۲-۶۹ نمایش آدرس آنالوگ برای یک نمونه S7-400

همانطور که دیده می‌شود، حرف P حذف شده و آدرس به‌صورت IW است. علت این است که آدرس در داخل بازه PII (که ۱۲۸ بایت فضا دارد) قرار گرفته است.

در این حالت در برنامه‌نویسی می‌توان آدرس را به‌صورت IW100 یا حتی PIW 100 به‌کار برد؛ ولی در حالت PIW همانطور که ذکر شد دیتا مستقیماً از کارت گرفته می‌شود.

نکته ۱: در S7-300 به‌دلیل کوچک بودن حافظه PII آنالوگ‌ها به‌طور اتوماتیک در خارج از PII آدرس‌دهی می‌شوند. در واقع فضای PII برای دیجیتال ورودی استفاده می‌گردد.

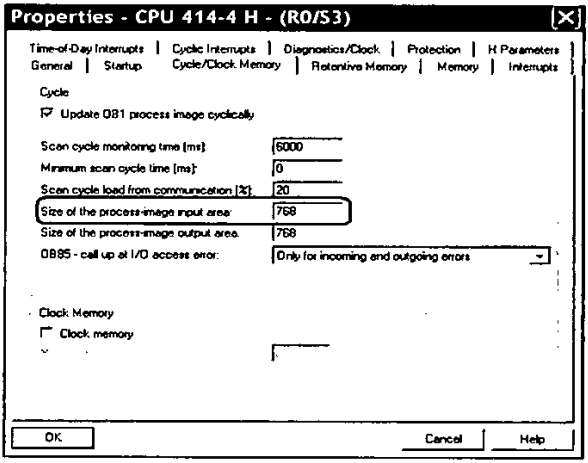
نکته ۲: در S7-300 آدرس PIW را نمی‌توان تا هر حد دلخواهی افزایش داد. اگر خارج از فضایی که در مشخصات CPU برای آنالوگ نوشته شده آدرس‌دهی کنیم با پیغام خطا مواجه می‌شویم.

۲-۳-۳ آدرس‌دهی آنالوگ ورودی در S7-400

در برخی از CPUهای 400 مانند 3-414 یا بالاتر آنالوگ از ابتدا در ناحیه PII قرار می‌گیرد و به‌صورت IW آدرس‌دهی می‌شود. در برخی دیگر آنالوگ شبیه S7-300 با PIW آدرس‌دهی می‌شود ولی دو تفاوت اصلی با S7-300 وجود دارد:

۱- اگر آنالوگ در S7-400 به‌صورت PIW باشد و در برنامه‌نویسی با IW آدرس‌دهی شود منجر به روشن شدن فالت INTF شده و اگر وقفه مربوطه وجود نداشته باشد، منجر به توقف CPU شده و در بافر پیغام Area Length Error ثبت می‌گردد. در حالی که در S7-300 این اشکال منجر به Stop شدن CPU نمی‌شود. این موضوع برای تایمر و کانتر نیز صادق است.

۲- ناحیه PII توسط کاربر قابل تغییر است و می‌توان با بالا بردن اندازه آن آنالوگ‌های بیشتری را به حافظه PII منتقل نمود. برای این کار در Hwconfig روی CPU دوبار کلیک کرده و در بخش Cycle Clock Memory مانند شکل ۲-۷۰ مقدار PII را تغییر می‌دهیم.



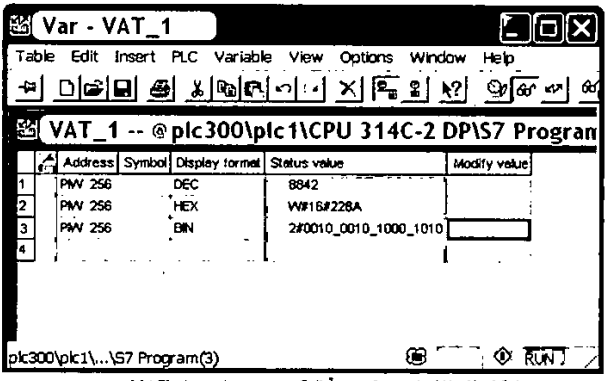
شکل ۲-۷۰ امکان تغییر ناحیه PII در CPU414-4H

۲-۳-۴ Monitor/Modify/Force کردن آنالوگ ورودی

مانیتور کردن آنالوگ ورودی

برای مشاهده مقدار آنالوگ می‌توان همانطور که ذکر شد، از پنجره Monitor / Modify در محیط Hwconfig استفاده کرد؛ ولی این پنجره مقدار را فقط به‌صورت هگزادسیمال نشان می‌دهد.

راه بهتر استفاده از جدول Varriable Table است که در کتاب سطح مقدماتی توضیح داده شد. با استفاده از این جدول می‌توان سیگنال آنالوگ ورودی را به فرمت دلخواه Hex یا Decimal یا Binary مشاهده کرد. شکل ۲-۷۱ این موضوع را بهتر نشان می‌دهد.

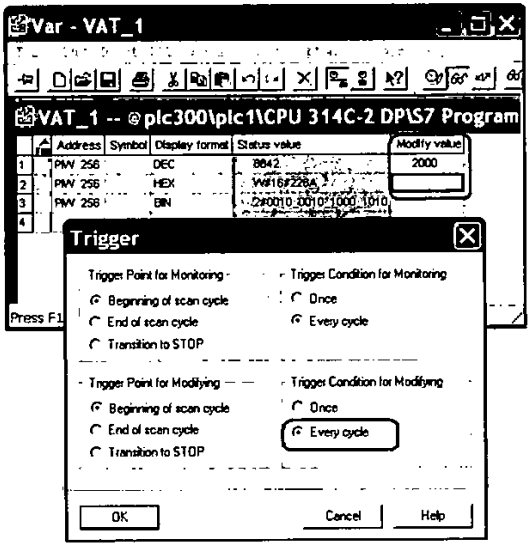


شکل ۲-۷۱ مانیتور کردن آنالوگ ورودی با جدول VAT

در نمایش باینری تمام ۱۶ بیت که خروجی A/D هستند نمایش داده می‌شوند. در شرایطی که Resolution کارت پایین باشد برخی بیت‌های با ارزش کمتر صفر نمایش داده می‌شوند. به‌عنوان مثال برای کارت AI 8 x 12 bit چهار بیت سمت راست که ارزش کمتر دارند همواره صفر خواهند بود.

Modify کردن آنالوگ ورودی

توسط پنجره Monitor /Modify یا از طریق جدول VAT می‌توان مطابق توضیحاتی که در کتاب سطح مقدماتی داده شد، آنالوگ ورودی را Modify نمود. این کار برای PIW و IW هر دو امکان‌پذیر است. فقط بایستی توجه داشت که تنظیمات پنجره‌های فوق فقط برای اعمال مقدار در یک سیکل اسکن است. در صورتی‌که لازم است به‌طور مداوم مقداری Modify شود بایستی در تنظیمات Triger گزینه every Cycle انتخاب گردد. توضیحات کامل در کتاب سطح مقدماتی داده شده است.



شکل ۲-۷۲ Modify کردن آنالوگ ورودی و تنظیمات آن

Force کردن آنالوگ ورودی

آدرس‌های PIW که به‌صورت Peripheral خوانده می‌شوند قابل Force نیستند؛ ولی آدرس‌های IW را می‌توان Force نمود. روش Force کردن در کتاب سطح مقدماتی تشریح شده است.

۲-۳-۵ برنامه‌نویسی آنالوگ ورودی

الف) برنامه‌نویسی با آنالوگ خام

سیگنال‌های آنالوگ ورودی معرف یک کمیت فرآیندی مانند فشار، دما، فلو و... هستند. استفاده از آنها در برنامه‌نویسی برای دو منظور است:

۱- برای کنترل کمیت فرآیندی: به‌عنوان مثال برای ایجاد آلارم حد بالا یا حد پایین و ایجاد فرمان‌های لازم برای کنترل فرآیند

۲- برای مانیتور کردن لحظه به لحظه کمیت فرآیندی: در این حالت سیگنال دریافتی توسط PLC به سیستم مانیتورینگ ارسال می‌گردد.

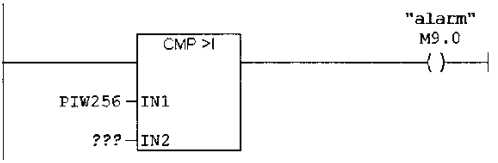
فصل ۲

مثال ۲-۱

در نظر بگیرید که یک ترانسمیتر فشار روی مخزنی نصب شده و به‌صورتی کالیبره شده است که به ازای فشار 0 bar سیگنال 4 mA و به ازای فشار 25 bar سیگنال 20 mA را به کانالی از کارت آنالوگ که آدرس آن آدرس PIW256 می‌باشد می‌فرستد. در این شرایط لازم است:

- اگر فشار از 19 bar بیشتر شد، آلارم High Pressure تولید شود.
- اگر فشار از 21 bar بیشتر شد، فرمان تریپ تولید شود.
- مقدار فشار لحظه به لحظه در سیستم مانیتورینگ نمایش داده شود.

آنچه که از ترانسمیتر به کارت ارسال می‌شود یک سیگنال جریانی است و آنچه از کارت به CPU ارسال می‌شود یک دیتای ۱۶ بیتی است که اصطلاحاً به آن آنالوگ خام گفته می‌شود. همانطور که در جدول ۲-۱۲ دیده می‌شود مقدار نرمال دیتای A/D بین 0 تا 27648 تغییر می‌کند. اگر بخواهیم آنالوگ خام را به یک مقایسه‌گر بدهیم تا آلارم تولید کند نمی‌توانیم مقدار PIW256 را با عدد 19 مقایسه کنیم. زیرا PIW256 آنالوگ خام است و مفهوم فشار ندارد.



شکل ۲-۷۳ برنامه‌نویسی با آنالوگ خام

پس باید معادل A/D شده 19 bar را محاسبه و به پایه دوم مقایسه‌گر بدهیم. برای محاسبه معادل 19bar از جدول ۲-۱۲ استفاده می‌کنیم.

جدول ۲-۱۲

Pressure (0-25 bar)	mA	A/D decimal
25	20	27648
.	.	.
.	.	.
0	4	0

اگر دو ستون Pressure و A/D را در نظر بگیریم، می‌توانیم معادله خط بین این‌دو را به‌دست آورده سپس به ازای هر نقطه فشار دلخواه مقدار A/D را محاسبه و در برنامه نویسی استفاده کنیم.

از آنجا که به‌ازای 0 bar مقدار دسیمال نیز 0 است، معادله خط از مبدا مختصات می‌گذرد و می‌توان تناسب گرفت یا از رابطه زیر استفاده نمود:

$$y = m * x$$

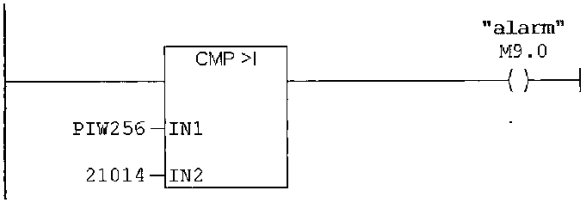
که در آن:

- y مقدار سیگنال دسیمال A/D است.
- m ضریب زاویه خط است و برابر است با $m=(27648-0)/(25-0)=1106$.
- x مقدار فشار است.

با توجه به رابطه فوق معادل فشار 19 bar برابراست با

$$y = 1106 * 19 = 21014$$

بنابراین برنامه تولید آلارم به‌صورت زیر خواهد بود:



شکل ۲-۷۴ برنامه نویسی با آنالوگ خام

تمرین ۲-۱: اگر بازه کاری ترانسمیتر فشار بین دو عدد x1 و x2 باشد (مثلاً 1 – 25 bar) رابطه فوق تغییر کرده، معادله خط به‌صورت زیر تبدیل می‌شود. چگونگی به‌دست آوردن این معادله خط را بررسی کنید.

$$y = m * (x - x1)$$

که در آن:

$$m = 27648 / (x2 - x1)$$

همانطور که دیده می‌شود، استفاده از این روش برای کاربرد عملی مناسب نیست. با فرض اینکه به ازای نقاط مورد نظر برای آلارم و تریپ نیز معادل دسیمال محاسبه شود سوالی که باقی می‌ماند این است که برای نمایش سیگنال آنالوگ در محیط مانیتورینگ چه باید کرد. بدیهی است که اگر مقادیر A/D به سیستم مانیتورینگ ارسال و در آنجا نمایش داده شوند هیچ مفهومی برای اپراتور ندارند زیرا اپراتور فرآیند نیاز به مشاهده مقدار کمیت فرآیندی مانند فشار و ... دارد.

با توجه به نکات فوق نتیجه‌ای که به‌دست می‌آید این است که:

سیگنال آنالوگ خام فاقد مفهوم است و معرف کمیت فرآیندی نیست؛ بنابراین به‌طور مستقیم در برنامه-نویسی استفاده نمی‌شود و نیاز به Scale کردن دارد.

تذکر: برای سیگنال‌های ترموکوپل و RTD، روش ساده‌تر این است که با توجه به جدول‌های ذکر شده قبلی، با تقسیم مقدار A/D بر عدد 10 مقدار دما را به‌دست آوریم.

فصل ۲

ب) برنامه‌نویسی با آنالوگ Scale شده

برای استفاده از سیگنال آنالوگ به‌صورتی که مفهوم کمیت فرآیندی داشته باشد، بایستی فرمول به‌دست آمده را تغییر داد به‌صورتی که مقدار PIW را به‌عنوان متغیر y بگیرد و مقدار فرآیندی را به‌عنوان متغیر x برگرداند. این عدد مفهوم فرایندی (مانند فشار) خواهد داشت و مقایسه می‌تواند براساس آن انجام شود.

این رابطه به‌صورت زیر خواهد بود:

$$x = a * y + x1$$

که در آن:

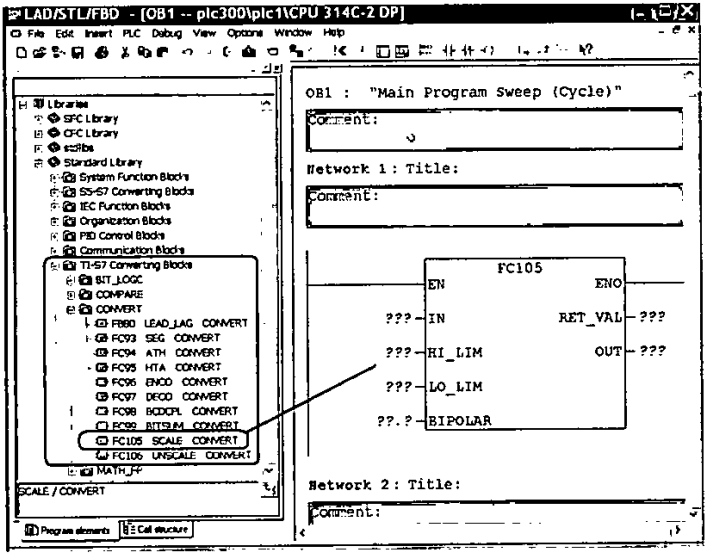
$$a = (x2 - x1)/27648$$

بهترین روش آن است که رابطه فوق در یک فانکشن پیاده‌سازی گردد و پس از صدا زدن فانکشن مورد نظر در برنامه آنالوگ به‌عنوان ورودی به آن داده شده و مقدار کمیتی فرآیندی از خروجی آن گرفته شود.

در عمل نیازی نیست که کاربر این فانکشن را ایجاد کند، زیرا در کتابخانه نرم افزار Step7 فانکشن ذکر شده که قبلاً توسط زیمنس تهیه شده است وجود دارد.

فانکشن Scale زیمنس FC105

این‌فانکشن در کتابخانه step7 در زیرمجموعه Standard Library درقسمت TI-S7 مانندشکل۲-۷۵دردسترس است.



شکل ۲-۷۵ فراخوانی FC105 از کتابخانه نرم‌افزار

عملکرد

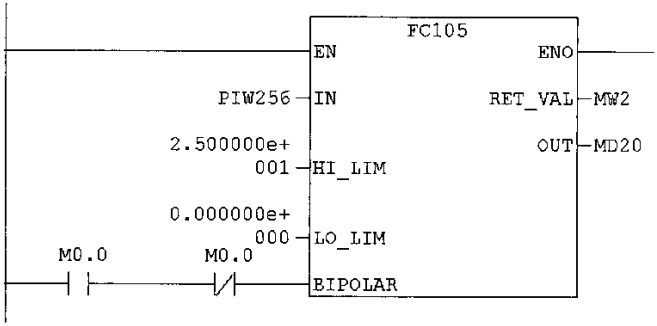
این فانکشن آنالوگ خام را به‌عنوان ورودی گرفته و آنرا بین حدود مورد نظر Scale می‌کند. خروجی Scale شده معرف کمیت فرآیندی است که در برنامه برای تولید فرمان یا آلارم و ... می‌توان استفاده نمود.

ورودی‌ها

IN : ازجنس Integer است و آدرس آنالوگ ورودی به آن داده می‌شود.

HI_LIM و LO_LIM: به‌صورت عدد Real است و حدود کار نرمال ترانسمیتر به آن داده می‌شود. به‌عنوان مثال در شکل ۲-۷۶ برای ترانسمیتر فشار حد پایین 0.0 و حد بالا 25.0 داده شده است.

BIPOLAR: از جنس Bool است. اگر سیگنال دارای علامت است یعنی مثبت و منفی می‌شود به آن یک اختصاص می‌دهیم و اگر سیگنال فقط مثبت است به این ورودی صفر اختصاص می‌دهیم. چون این ورودی از جنس Bool است نمی‌توان 0 یا 1 را مستقیماً به آن اختصاص داد (این محدودیت زبان LAD/FBD است ولی در STL امکان پذیر است) بنابراین صفر و یک با کنتاکت‌ها ساخته می‌شود و به این ورودی داده می‌شود.



شکل ۲-۷۶ ورودی و خروجی‌های FC105

خروجی‌ها

RET_VAL: از جنس Word است و کد وضعیت فانکشن را بر می‌گرداند. وقتی سیگنال در بازه نرمال است این خروجی مقدار صفر و وقتی سیگنال آنالوگ از بازه نرمال خارج می‌شود این خروجی مقدار 8 را برمی‌گرداند. در واقع این خروجی برای شرایطی که سیگنال جریانی 4-20 mA از 4 کمتر یا از 20 بیشتر شود مفید است.

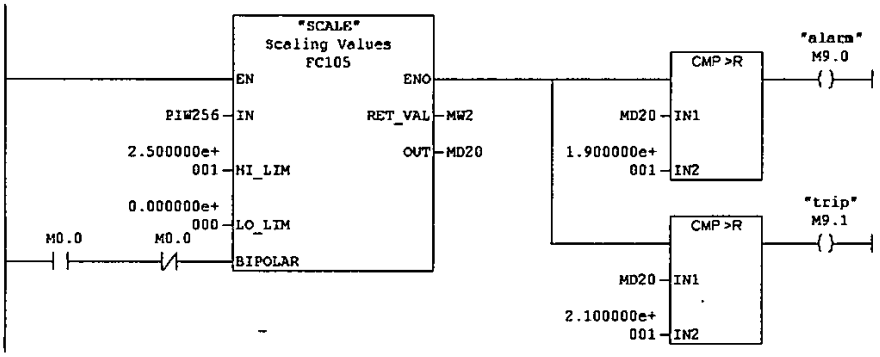
OUT: آنالوگ Scale شده به‌صورت عدد Real از این خروجی قابل دریافت است و می‌توان آنرا به یک متغیر ۳۲ بیتی (Double Word) اختصاص داد.

تذکر: اگر ورودی in از بازه نرمال بالاتر رود خروجی Out از Hi_Lim بالاتر نخواهد رفت.

اگر ورودی in از بازه نرمال کمتر شود خروجی Out از Lo_Lim کمتر نخواهد شد.

مثال ۲-۲

برنامه تولید آلارم و تریپ مثال ۲-۱ با استفاده از فانکشن Scale به‌صورت زیر خواهد بود. همانطور که مشاهده می‌شود، به‌سادگی خروجی Scale شده را که معرف فشار است، با اعداد 19.0 و 21.0 مقایسه کرده‌ایم.

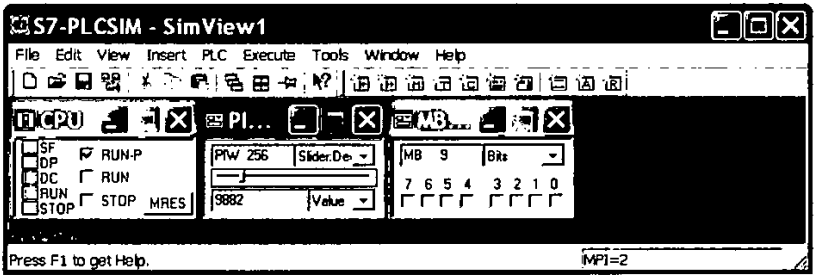


شکل ۲-۷۷ برنامه مثال ۲-۲

تذکر: اگر برنامه فوق که در OB1 نوشته شده است را به PLC دانلود کنیم، با فالت و stop شدن CPU مواجه می‌شویم. خواننده می‌تواند این موضوع را با استفاده از سیمولاتور نیز چک کند.

علت بروز مشکل فوق این است که در برنامه‌نویسی ساختار یافته که از فانکشن استفاده می‌شود بایستی ابتدا فانکشن به PLC و سپس بلاک OB1 دانلود گردد؛ یا هر دو بلاک با یکدیگر دانلود گردند تا مشکلی پیش نیاید.

تمرین ۲-۱: پس از رفع مشکل فوق سیمولاتور را به‌صورت شکل زیر اجرا کرده و آنالوگ را به‌صورت slider شبیه شکل ۲-۷۸ تغییر دهید و نتیجه را در برنامه ببینید. اکنون آنالوگ را از 27648 بالاتر ببرید و خروجی Ret_Val را مانیتور کنید.



شکل ۲-۷۸ شبیه‌سازی آنالوگ با سیمولاتور

تمرین ۲-۲: در برنامه قبل به‌جای PIW256 از آدرس PIW512 استفاده کنید. این آدرس در سخت افزار وجود ندارد. پس از دانلود برنامه چه اتفاقی می‌افتد؟ آیا در سیمولاتور می‌توان PIW512 را وارد کرد؟

مثال ۲-۳

دریک مخزن سیال توسط سه پمپ وارد می‌شود که بسته به سطح مخزن یک، دو و یا سه پمپ کار می‌کنند. روی مخزن یک ترانسمیتر سطح نصب شده که بین 5-0 متر کالیبره شده است و به ازای ارتفاع 5 متر 20mA و به ازای ارتفاع 0 متر 4mA ارسال می‌کند.

برنامه‌ای بنویسید که:

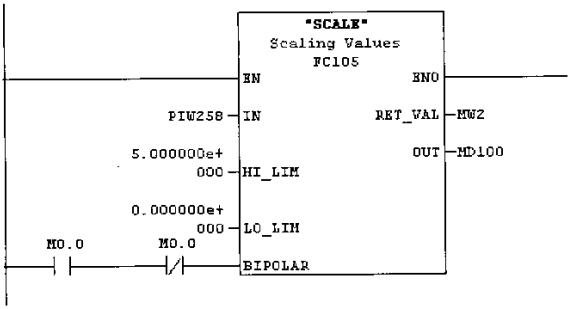
وقتی سطح از 1m کمتر است هر سه پمپ کار کنند.

وقتی سطح بین 1-3 m است فقط دو پمپ کار کنند.

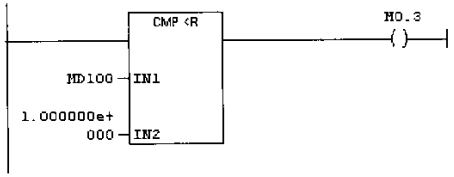
وقتی سطح بین 3-4 m است فقط یک پمپ کار کند.

وقتی سطح از 4m بیشتر شد هر سه پمپ خاموش شوند.

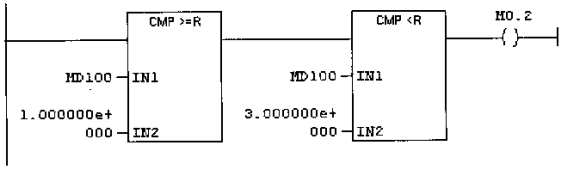
Network 1 : Title:



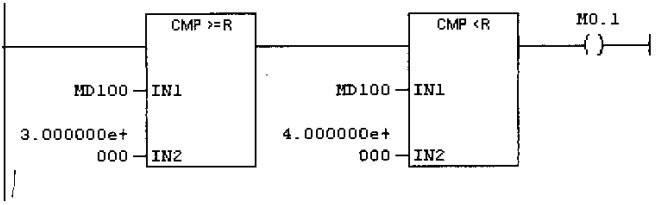
Network 2 : Title:



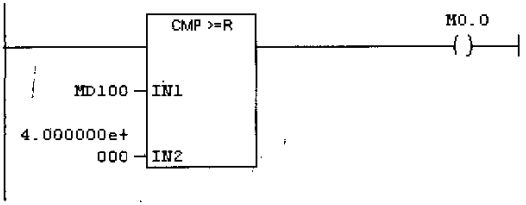
Network 3 : Title:



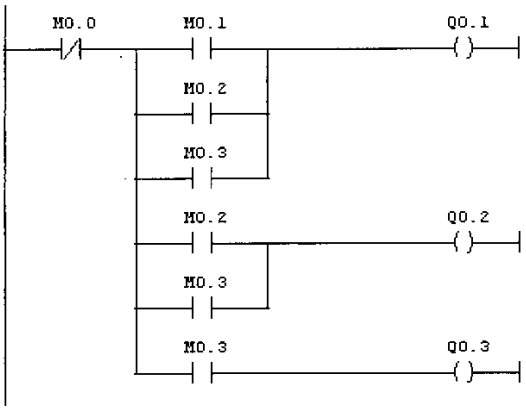
Network 4 : Title:



Network 5 : Title:



Network 6 : Title:



شکل ۲-۷۹ برنامه مثال ۲-۳

تمرین ۲-۳: برنامه مثال ۲-۳ را طوری کامل کنید که وقتی یک پمپ کار می‌کند چراغ آلارم Q1.0 به‌صورت زیر چشمک زن شود:

- وقتی یک پمپ کار می‌کند با فرکانس کم
- وقتی دو پمپ کار می‌کند با فرکانس متوسط
- وقتی هر سه پمپ کار می‌کنند با فرکانس زیاد

تمرین ۲-۴: تمرین ۲-۳ را طوری کامل کنید که در هر یک از سه حالت ذکر شده فقط تعداد ۳۰ پالس به خروجی Q1.0 منتقل شده سپس Q1.0 خاموش شود. برنامه این تمرین را با استفاده از اختصاص زمان متغیر به دو تایمر پالس بنویسید.

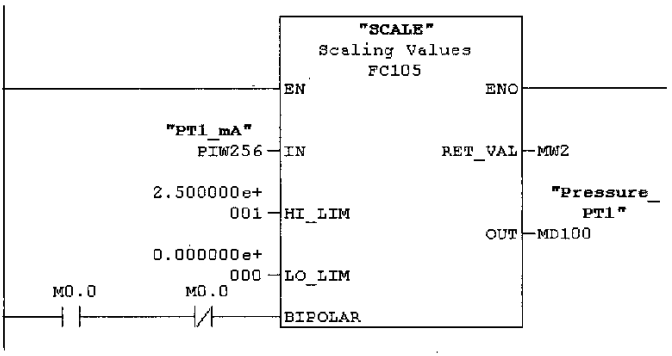
مثال ۲-۴

روی مخزنی دو ترانسمیتر فشار مشابه وجود داردکه هر دو بین 0-25 bar کالیبره شده‌اند. برنامه‌ای بنویسید که:

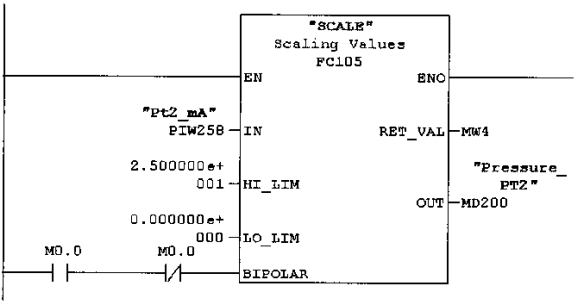
- اگر هر دو سیگنال دریافتی از ترانسمیترها در بازه نرمال است و اختلاف بین آنها از 1bar کمتر است میانگین آنها به‌عنوان مبنا استفاده شود.
- اگر هر دو سیگنال دریافتی از ترانسمیترها در بازه نرمال است ولی اختلاف بین آنها بیش از 1bar است آنکه ماکزیمم است به‌عنوان مبنا استفاده شود. در این حالت آلارم Difference_Fault تولید شود.
- اگر سیگنال یکی از ترانسمیترها از بازه نرمال خارج شود، دیگری که فاقد مشکل است به‌عنوان مبنا قرار گیرد. در این حالت آلارم PT1_Fault برای ترانسمیتر اول و آلارم PT2_Fault برای ترانسمیتر دوم تولید شود.
- اگر سیگنال هر دو ترانسمیتر از بازه نرمال خارج شود، مقدار ثابت 20 bar مبنا قرار گیرد و آلارم 2PT_Fault ظاهر شود. اگر پس از گذشت ۳۰ دقیقه این وضعیت باقی بماند سیگنال Trip فعال گردد.

برنامه به‌صورت زیر است:

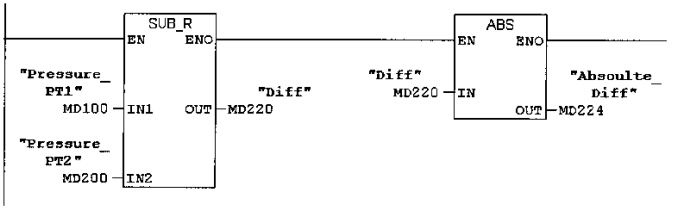
```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
```



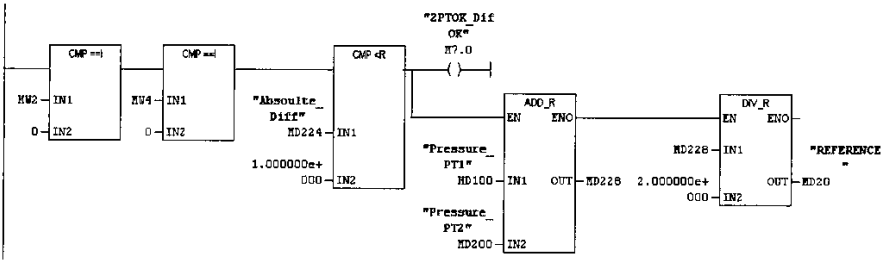
Network 2 : Title:



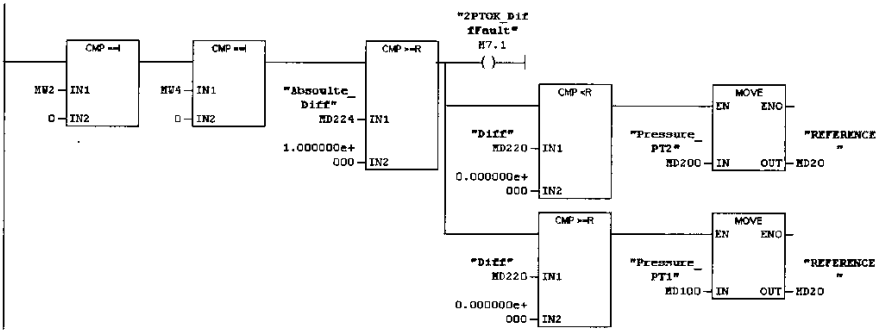
Network 3 : Title:



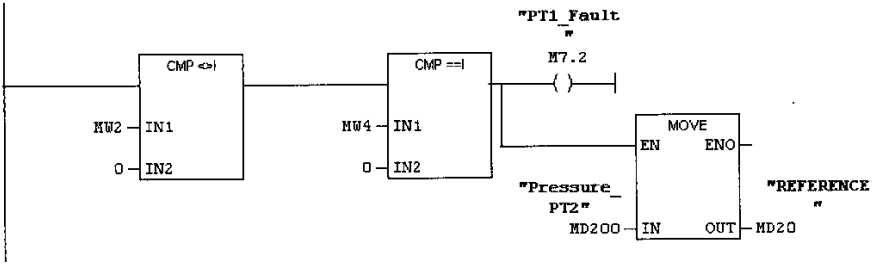
Network 4 : Title:



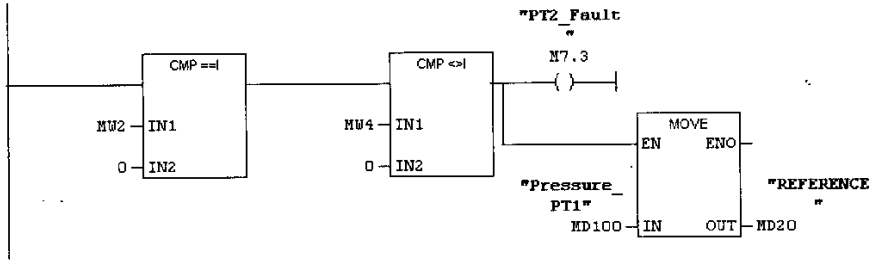
Network 5 : Title:



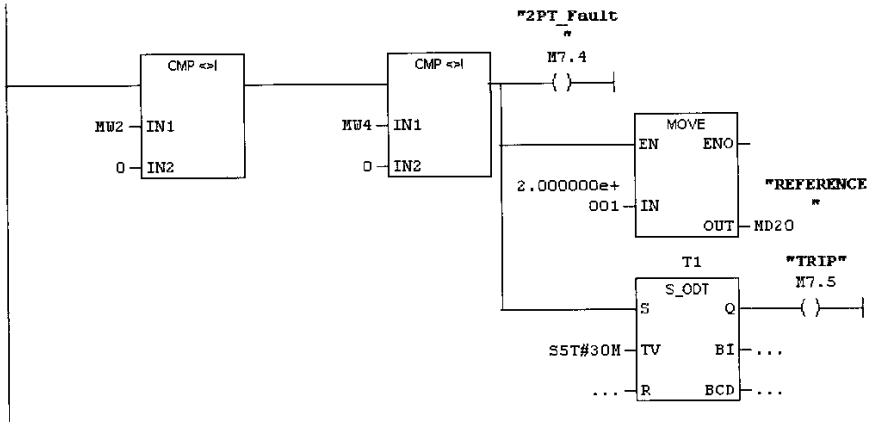
Network 6 : Title:



Network 7: Title:



Network 8 : Title:



شکل ۲-۸۰ برنامه مثال ۲-۴

فصل ۲

۲-۴ سیگنال‌های آنالوگ خروجی

۲-۴-۱ انواع آنالوگ خروجی

سیگنال‌های آنالوگ خروجی یکی از سیگنال‌های پر کاربرد به‌ویژه در کنترل فرآیند که لوپ‌های کنترلی زیاد وجود دارند هستند. این سیگنال‌ها به‌صورت ولتاژی یا جریانی هستند. از مهمترین وسایلی که به‌عنوان آنالوگ خروجی استفاده می‌شود می‌توان موارد زیر را نام برد:

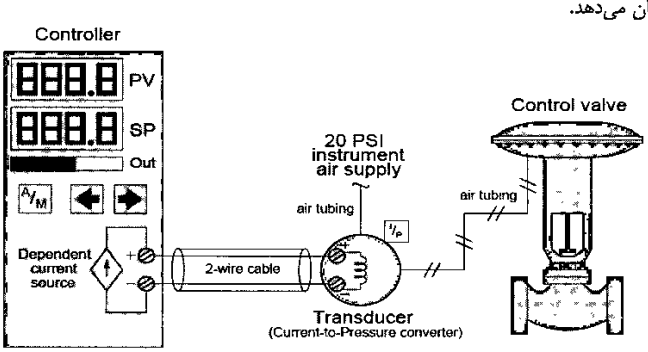
- کنترل ولو
- درایوهای کنترل دور موتور

علاوه بر دو مورد فوق انواع نشان دهنده‌ها و Meterها که روی درب پنل یا میز اپراتوری نصب می‌شوند نیز به‌عنوان آنالوگ خروجی محسوب می‌شوند.

کنترل ولو

برخلاف سلونوئیدولوها که مسیر سیال را کاملاً باز یا کاملاً بسته می‌کنند، کنترل ولو می‌تواند به‌صورت پیوسته میزان باز وبسته نمودن مسیر را تغییر دهد از اینرو به آن ولو تناسبی[1] نیز می‌گویند.

کنترل ولوها به‌عنوان Final Element در لوپ کنترلی نقش حیاتی دارند و کنترل فلو، فشار یا در برخی موارد کنترل دما توسط تغییر موقعیت آنها انجام می‌شود. ولو کنترلی دارای یک قسمت محرک است که فرمان خود را از کنترلر می‌گیرد. این محرک Actuator دارای انواع مختلف الکتریکی، پنوماتیکی و هیدرولیکی است. شکل ۲-۸۱ شماتیک عملگرهای نیوماتیکی را نشان می‌دهد.



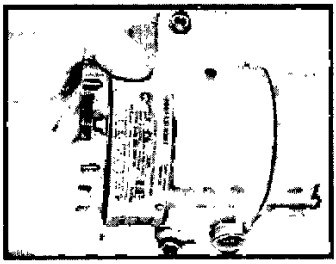
شکل ۲-۸۱ کنترل ولو با محرک پنوماتیکی

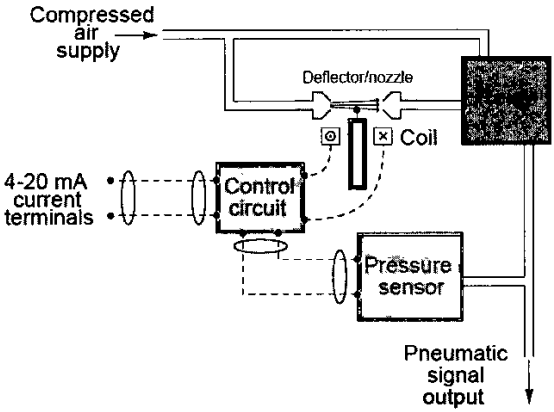
کنترل‌ولوها به‌منظور باز نمودن تدریجی دریچه، نیاز به سیگنال الکتریکی پیوسته (آنالوگ) دارند. اکثر کنترل‌ولوهای صنعتی با سیگنال 4 mA الی 20 mA کار می‌کنند.

1. Proportional Valve

همانطور که در شکل قبل ۲-۸۱ می‌شود، سیگنال 4-20 mA از سمت کنترلر به یک مبدل جریان به فشار که در اصطلاح I/P خوانده می‌شود داده می‌شود. این مبدل سیگنال الکتریکی را به فشار هوا (معمولاً 3-15 psi) تبدیل می‌نماید. موقعیت ولو با تغییر فشار هوا تغییر می‌کند.

شماتیک یک مبدل I/P در شکل ۲-۸۲ نشان داده شده است.

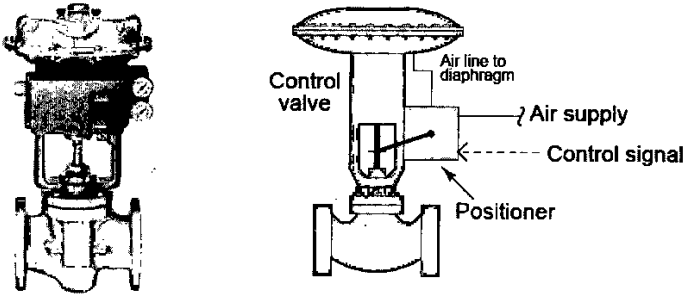




شکل ۲-۸۲ مبدل جریان به فشار I/P

برای کنترل سریع و دقیق، برخی ولوهای کنترلی مجهز به پوزیشنر هستند. پوزیشنر دارای یک ورودی الکتریکی، و یک ورودی هوای فشرده است. پوزیشنر با توجه به فرمان الکتریکی دریافتی از کنترلر سیگنال فشار را تغییر می‌دهد و موقعیت ولو را اصلاح می‌کند. در واقع پوزیشنر یک کنترلر است که مقدار مبنا[1] را از PLC گرفته و به‌صورت داخلی موقعیت ولو را تغییر می‌دهد تا با مقدار مبنا اختلافی نداشته باشد.

1. Set Point



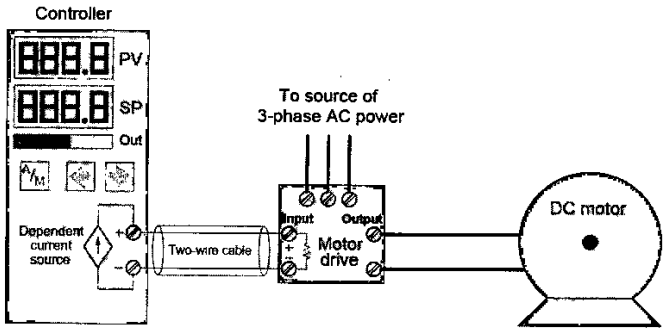
شکل ۲-۸۲ ولو با پوزیشنر

درایوهای کنترل دور

درایوها برای کنترل دور موتورهای الکتریکی AC یا DC به‌کار می‌روند. موتور الکتریکی می‌تواند محرک یک پمپ یا گیربکس یا جابه‌جا کننده موقعیت یک سیستم و کاربردهایی از این نوع باشد.

از جمله درایوهای پرکاربرد در صنعت می‌توان به درایو کنترل دور موتور سه‌فاز القایی (AC Drive) اشاره نمود. این درایوها توسط کنترل همزمان ولتاژ و فرکانس، اقدام به کنترل همزمان سرعت و گشتاور موتور می‌نمایند. اکثر این درایوها دارای ترمینال‌های مخصوصی جهت اعمال سیگنال الکتریکی آنالوگ به‌منظور تنظیم فرکانس توسط درایو می‌باشند؛ که این سیگنال معمولاً یک سیگنال ولتاژی مانند 0 V الی 10 V می‌باشد و از خروجی آنالوگPLC گرفته می‌شود.

درایو مقدار مبنای سرعت را به‌صورت یک سیگنال ولتاژی یا جریانی از کنترلر می‌گیرد و بر اساس آن دور موتور را تنظیم می‌کند.



شکل ۲-۸۴ درایو محرک موتور

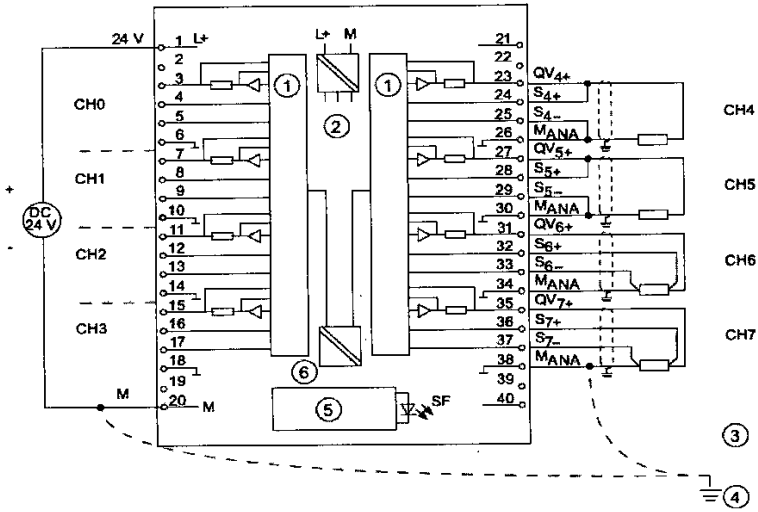
۲-۴-۲ عملکرد کارت آنالوگ خروجی

در کتاب سطح مقدماتی به نحوه تبدیل سیگنال در کارت آنالوگ خروجی اشاره شد. درآنجا ذکر شد که دیتا بصورت یک رشته ۱۶ بیتی وارد مبدل D/A کارت می‌گردد و از طریق آن به ولتاژ یا جریان تبدیل می‌گردد. کارت‌های آنالوگ خروجی نیز شبیه آنالوگ ورودی دارای قدرت تفکیک مختلف هستند که از ۸ بیت تا ۱۶ بیت می‌باشد.

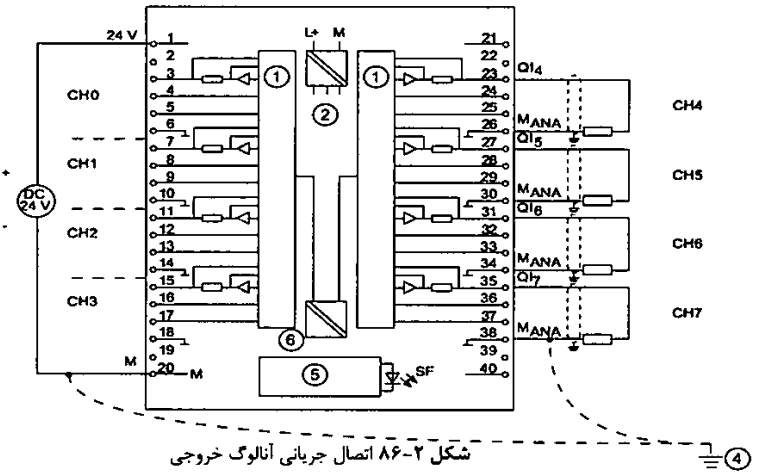
۲-۴-۳ اتصالات

شکل ۲-۸۵ شماتیک اتصالات کارت آنالوگ ۸ خروجی AO 8 x 12 bit را نشان می‌دهد. نکات قابل توجه عبارتند از:

- کارت دارای دو مبدل D/A می‌باشد که در شکل با شماره ۱ نشان داده شده است. در واقع دو دسته کانال چهار تایی وجود دارند که برای هر دسته A/D مجزایی وجود دارد.
- اتصال ولتاژی می‌تواند دو سیمه یا چهار سیمه باشد. در نوع چهار سیمه ولتاژ دو سر بار به‌عنوان فیدبک برمی‌گردد و کارت افت ولتاژ مسیر را جبران می‌کند. اتصال دو یا چهار سیمه ولتاژی به‌صورت فیزیکی است و هیچ تنظیم خاصی در نرم‌افزار ندارد.
- نوع جریانی فقط دو سیمه بسته می‌شود.



شکل ۲-۸۵ اتصال ولتاژی آنالوگ خروجی

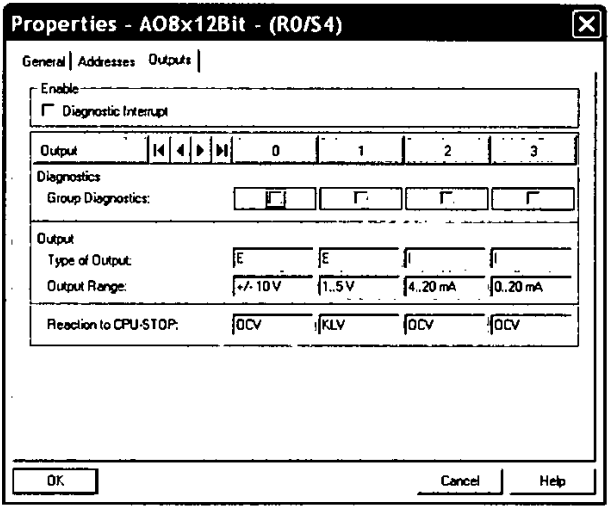


شکل ۲-۸۶ اتصال جریانی آنالوگ خروجی

اتصال سایر کارت‌های آنالوگ خروجی که دو یا چهار خروجی هستند نیز مشابه کارت فوق می‌باشد.

۲-۴-۴ تنظیمات در HWconfig

به‌عنوان نمونه تنظیمات کارت AO 8 x 12 bit را بررسی می‌کنیم. اگر این کارت را در Hwconfig وارد کرده و روی آن دوبار کلیک نماییم پنجره‌ای مانند شکل ۲-۸۷ باز می‌شود.



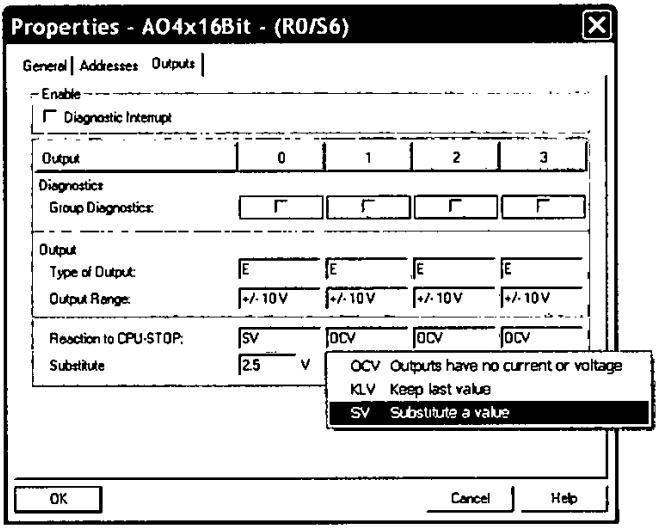
شکل ۲-۸۷ تنظیم کارت آنالوگ خروجی

همانطور که در شکل۲-۸۷ دیده می‌شود:

- این کارت سیگنال‌های ولتاژی و جریانی را پشتیبانی می‌کند که برای هر کانال آن به‌طور جداگانه قابل تعریف است.
- امکانات وقفه[1] برای تشخیص عیب برای کل کارت و برای تک تک کانال‌ها وجود دارد.
- در صورت Stop شدن CPU می‌توان تعیین کرد که کارت چگونه خروجی را کنترل کند یعنی چه ولتاژ یا جریانی را روی آن قرار دهد. در این کارت فقط دو گزینه KLV و OCV وجود دارد که انتخاب آن برای هر کانال می‌تواند متفاوت باشد و به استراتژی ایمنی فرآیند بستگی دارد.

مفهوم گزینه‌های KLV و OCV و SV

- **Keep last valid value (KLV)** : با انتخاب این گزینه، در صورت توقف CPU آخرین مقدار معتبر قبلی روی خروجی حفظ می‌گردد.
- **Output have no Current or Voltage (OCV)** : به این مفهوم است که در صورت توقف CPU ولتاژ یا جریان خروجی صفر می‌گردد.
- برخی کارت‌های خروجی مانند کارت AO 4 x 16 bit دارای گزینه SV مانند شکل۲-۸۸ است که به مفهوم Substitute Value است و می‌توان تعیین کرد که در صورت توقف CPU چه ولتاژ یا جریانی روی خروجی قرار گیرد.



شکل ۲-۸۸ تنظیم نحوه واکنش به توقف CPU در آنالوگ خروجی

1. Interrupt

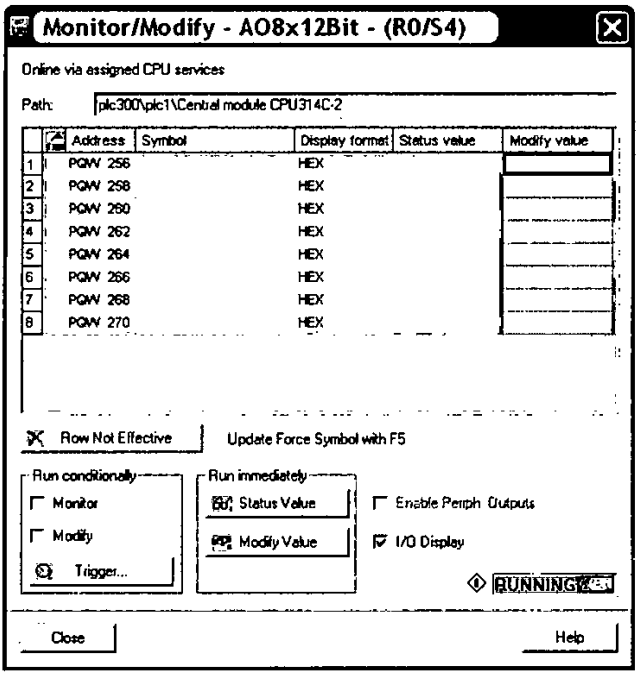
۲-۴-۵ آدرس‌دهی آنالوگ خروجی

تمامی نکاتی که در مورد آدرس‌دهی آنالوگ ورودی گفته شد برای آدرس‌دهی آنالوگ خروجی نیز صادق است. از اینرو فقط نکات مهم را مرور می‌کنیم.

فصل ۲

مشاهده آدرس‌ها در پنجره Monitor /Modify

وقتی ارتباط کامپیوتر با PLC برقرار است، در محیط Hwconfig می‌توان با کلیک راست روی کارت AO و انتخاب گزینه Monitor/Modify آدرس‌های آنالوگ را در پنجره‌ای مانند شکل ۲-۸۹ مشاهده نمود.



شکل ۲-۸۹ نمایش آدرس آنالوگ خروجی در نرم‌افزار

تفاوت آدرس‌دهی QW و PQW

ممکن است آدرس آنالوگ خروجی با یا بدون حرف P که معرف Peripheral است تعیین شود. اگر آدرس خارج از ناحیه حافظه خروجی (PIQ) باشد به‌صورت خودکار به‌صورت PQW ظاهر می‌شود در غیر اینصورت یعنی اگر آدرس در بازه PIQ باشد، به‌صورت QW خواهد بود. برای توضیحات بیشتر به بحث ارائه شده برای آنالوگ ورودی مراجعه نمایید.

۲-۴-۶ Monitor/Modify/Force کردن آنالوگ خروجی

Monitor کردن

آنالوگ‌هایی که به‌صورت QW هستند را می‌توان مانیتور کرد و وضعیت آنها را مشاهده نمود، ولی آدرس‌های PQW قابل مانیتور شدن نیستند. شکل ۲-۹۰ این موضوع را نشان می‌دهد. آنچه برای QW نمایش داده می‌شود مقدار دیجیتال است که پس از اعمال به D/A به ولتاژ یا جریان تبدیل می‌گردد.

	Address	Symbol	Display format	Status value
1	PQW 256		HEX	
2	PQW 258		HEX	
3	PQW 260		HEX	
4	PQW 262		HEX	
5	PQW 264		HEX	
6	PQW 266		HEX	
7	PQW 268		HEX	
8	PQW 270		HEX	

	Address	Symbol	Display format	Status value
1	QW 256		HEX	W#16#2EBB
2	QW 258		HEX	W#16#0000
3	QW 260		HEX	W#16#02FD
4	QW 262		HEX	W#16#0000
5	QW 264		HEX	W#16#0000
6	QW 266		HEX	W#16#0000
7	QW 268		HEX	W#16#0000
8	QW 270		HEX	W#16#0000

شکل ۲-۹۰ دو نوع آدرس‌دهی آنالوگ خروجی

Modify کردن

هر دو نوع آدرس QW و PQW قابل Modify کردن هستند. به‌عنوان مثال اگر مقدار هگز 5000 را به آدرس مورد نظر اعمال کنیم ولتاژی حدود ۷/۴ ولت روی کانال مشاهده خواهیم کرد.

تنظیمات پنجره Modify در بحث آنالوگ ورودی توضیح داده شد.

Force کردن

آدرس‌های PQW قابل Force کردن نیستند ولی QW را می‌توان Force نمود.

۲-۴-۷ برنامه‌نویسی آنالوگ خروجی

همانطور که توضیح داده شد در کارت آنالوگ خروجی مقداردیجیتال ۱۶ بیتی به مدار D/A داده شده و به سیگنال الکتریکی تبدیل می‌گردد.

تصور کنید که یک ولو کنترلی به کانال PQW256 متصل است. این ولو به ازای 4mA بسته و به ازای 20mA کاملاً باز است؛ ولی سوال این است که چه مقدار دیجیتالی بایستی به این آدرس فرستاد تا به مقدار mA مورد نظر تبدیل گردد.

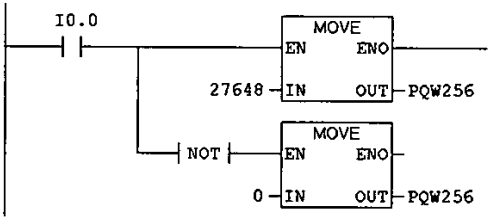
جدول‌های تبدیل برای آنالوگ خروجی برعکس آنالوگ ورودی بوده و به‌صورت جدول‌های ۲-۱۳ است.

جدول‌های ۲-۱۲

Voltage Output Ranges					
Output Range 0 to 10 V	Output Range 1 to 5 V	Output Range ±10 V	Units Decimal	Units Hex	Range
0	0	0	>32511	>7EFFH	Overflow
11.7589 : 10.0004	5.8794 : 5.0002	11.7589 : 10.0004	32511 : 27649	7EFFH : 6C01H	Overrange
10.0000 : 0	5.0000 : 1.0000	10.0000 : 0 : – 10.0000	27648 : 0 : – 6912 – 6913 : – 27648	6C00H : 0H : E500H E4FFH : 9400H	Nomial range
0	:0.9999 0	10.0004 : – 11.7589	– 27649 : – 32512	93FFH : 8100H	Underrange
		0	≤ 32512	<8100H	Underflow

Current Output Ranges					
Output Range 0 to 20 mA	Output Range 4 to 20 mA	Output Range ±20 mA	Units Decimal	Units Hex	Range
0	0	0	>32511	>7EFFH	Overflow
23.515 : 20.0007	22.81 : 20.005	23.515 : 20.0007	32511 : 27649	7EFFH : 6C01H	Overrange
20.000 : 0	20.000 : 4.000	20.000 : 0 : – 20.000	27648 : 0 : – 6912 – 6913 : – 27648	6C00H : 0H : E500H E4FFH : 9400H	Nominal range
0	3.9995 0	: – 23.515	– 27649 : – 32512	93FFH : 8100H	Underrange
	0	0	≤ 32512	<8100H	Underflow

با توجه به جدول ۲-۱۳، اگر مقدار دیجیتال 0 به کارت خروجی تبدیل شود به 4mA تبدیل می‌گردد و اگر مقدار دیجیتال 27648 اعمال شود خروجی کارت 20 mA خواهد بود. به برنامه زیر توجه کنید.



شکل ۲-۹۱ اعداد معادل ۴ و ۲۰ میلی‌آمپر در برنامه

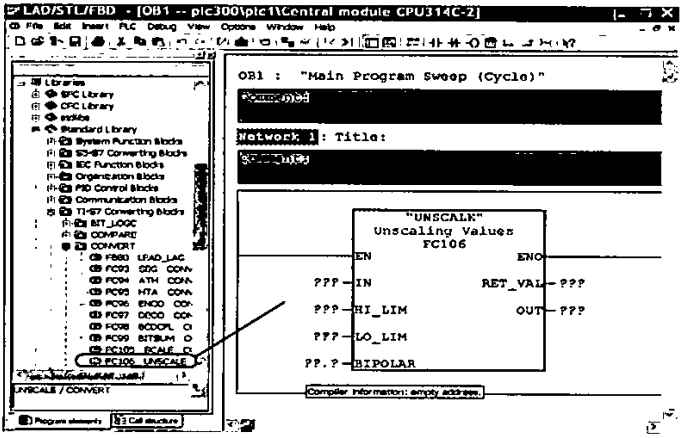
در عمل از روش اعمال مستقیم مقدار به آدرس آنالوگ استفاده نمی‌شود بلکه فرمان به‌صورتی که برای برنامه‌نویس یا اپراتور فرآیند با مفهوم است به خروجی داده می‌شود. به‌عنوان مثال اپراتور با اعمال فرمان 100% ولو کنترلی را کاملا باز می‌کند و با اعمال فرمان 50% آن را به حالت نیمه باز کنترل می‌کند پس:

در برنامه‌نویسی معمولاً سیگنال آنالوگ خروجی به‌صورت خام استفاده نمی‌شود. فرمان مورد نظر ابتدا Unscale شده سپس به خروجی اعمال می‌شود.

علت نام‌گذاری unscale این است که در اینجا مقدار فرمان فرآیندی توسط فانکشن به زبان D/A ترجمه می‌شود که کاملاً برعکس فانکشن Scale است. در Scale مقدار A/D به مقدار فرآیندی ترجمه می‌شد.

فانکشن FC106 Unscale

برنامه Unscale می‌تواند توسط کاربر مشابه رابطه‌ای که قبلاً برای آنالوگ ورودی ذکر شد نوشته و به‌کار گرفته شود ولی عملاً نیازی به این کار نیست و می‌توان از کتابخانه نرم‌افزار از زیر مجموعه TI-S7 این فانکشن را در کنار فانکشن Scale پیدا نمود.



شکل ۲-۹۲ فراخوانی FC106 از کتابخانه برنامه

عملکرد

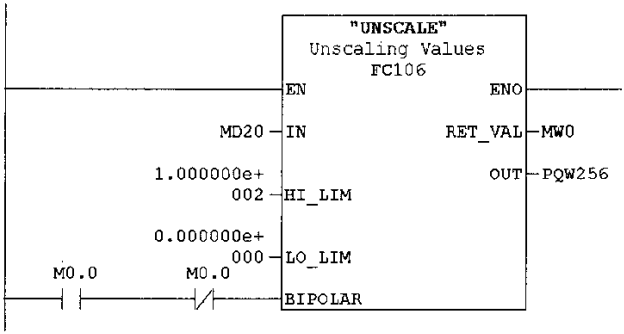
این فانکشن دارای دو حد بالا و پایین است که متناسب با سیگنال الکتریکی خروجی است. به‌عنوان مثال حد بالا معادل 20 mA و حد پایین معادل 4 mA است. براساس حدود فوق این فانکشن فرمانی که بایستی به خروجی آنالوگ ارسال شود را به‌صورت یک عدد اعشاری می‌گیرد و آنرا در حدود فوق Unscale کرده و به خروجی ارسال می‌نماید.

ورودی‌ها

IN: ازجنس Real است و مقدار فرمان به آن داده می‌شود. به‌عنوان مثال می‌توان به آن یک متغیر ۳۲ بیتی اختصاص داد. مقدار این متغیر از سمت سیستم مانیتورینگ داده شود.

HI_LIM و LO_LIM: به‌صورت عدد Real است و حدود فرمان را مشخص می‌کند. به‌عنوان مثال اگر خروجی ولتاژی و بین 0-10V باشد، در اینصورت LO_LIM معادل صفر ولت و HI_LIM معادل ده ولت است.

BIPOLAR: از جنس Bool است و مشابه آنچه قبلاً برای Scale ذکر شد می‌باشد. اگر سیگنال دارای علامت است یعنی مثبت و منفی می‌شود، به آن یک اختصاص می‌دهیم و اگر سیگنال فقط مثبت است به این ورودی صفر اختصاص می‌دهیم.



شکل ۲-۹۳ ورودی و خروجی‌های FC106

خروجی‌ها

RET_VAL: از جنس Word است و مشابه فانکشن Scale کد وضعیت فانکشن را بر می‌گرداند. وقتی سیگنال در بازه نرمال است این خروجی مقدار صفر و وقتی سیگنال آنالوگ از بازه نرمال خارج می‌شود، این خروجی مقدار 8 را بر می‌گرداند. به‌عنوان مثال اگر فرمانی بالاتر از HI_LIM به ورودی IN داده شود این خروجی 8 خواهد بود.

OUT: آدرس آنالوگ خروجی به این خروجی داده می‌شود.

در شکل ۲-۹۳ اگر تنظیم PQW256 روی ولتاژ 0-10V باشد، همانطور که در ورودی‌ها دیده می‌شود ۱۰ ولت معادل 100 درصد و صفر ولت معادل صفر درصد است. بنابراین اگر MD20 عدد 50.0 وارد شود خروجی به اندازه ۵ ولت ولتاژ خواهد داشت.

سوال: اگر در شکل فوق تنظیم کانال روی 10 V -/+ باشد، به ازای MD20 = 50.0 چه ولتاژی در خروجی ظاهر خواهد شد ؟

اکنون برای آشنایی با نحوه برنامه‌نویسی FC106 چند مثال ساده مطرح می‌شود.

مثال ۲-۵

یک پمپ سیال را به داخل مخزن می‌فرستد و درایو پمپ با سیگنال 0-10V کار می‌کند که به خروجی PQW272 متصل است. روی مخزن ۴ سوئیچ سطح به‌صورت زیر تعبیه شده است:

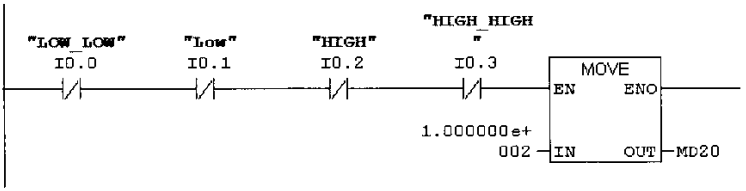
عنوان سوئیچ	سطح آشکار سازی	آدرس
LL	Low Low	I124.0
L	Low	I124.1
H	High	I124.2
HH	High High	I124.3

برنامه‌ای بنویسید که:

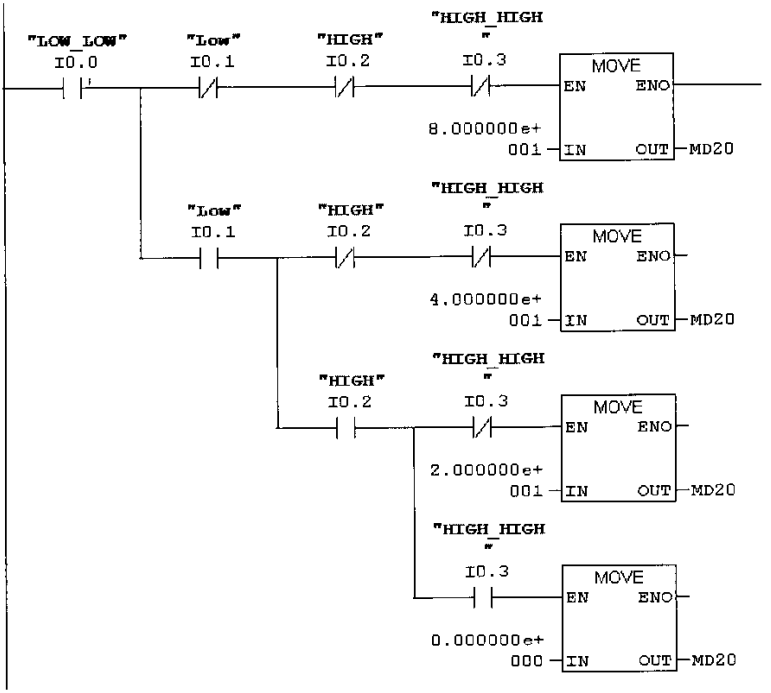
- اگر سطح کمتر از LL است، پمپ با ظرفیت ۱۰۰ درصد کار کند.
- اگر سطح بالاتر از LL و کمتر از L باشد، پمپ با ظرفیت ۸۰ درصد کار کند.
- اگر سطح بالاتر از L و کمتر از H باشد، پمپ با ظرفیت ۴۰ درصد کار کند.
- اگر سطح بالاتر از H و کمتر از HH باشد، پمپ با ظرفیت ۲۰ درصد کار کند.
- اگر سطح بالاتر از HH باشد، پمپ غیر فعال شود.

برنامه به‌صورت زیر است:

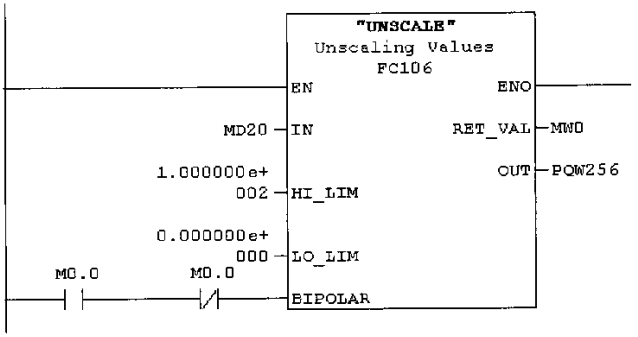
```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
```



Network 2: Title:



Network 3: Title:



شکل ۲-۹۴ برنامه مثال ۲-۵

تمرین ۲-۵: در مثال قبل اگر پمپ در خروجی مخزن باشد، منطق کنترل برعکس خواهد بود یعنی وقتی سطح کمتر از L است پمپ غیر فعال و وقتی بالاتر از HH است با حداکثر ظرفیت کار می‌کند. برنامه را اصلاح و با سیمولاتور یا PLC تست کنید.

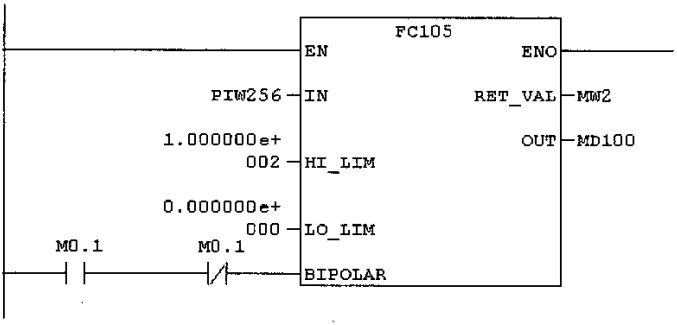
مثال ۲-۶

اگر ترانسمیتری بین 0 تا Scale 100 شده باشد، برنامه‌ای بنویسید که هر چقدر آنالوگ ورودی زیاد شود ولو کنترلی برعکس فرمان بگیرد، یعنی:

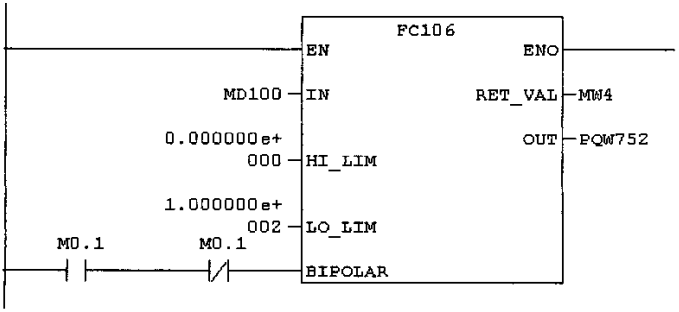
- اگر آنالوگ ورودی 0 باشد، فرمان ولو 100% باشد.
- اگر آنالوگ ورودی 100 باشد، فرمان ولو 0% باشد.

برنامه به‌صورت زیر است. همانطور که دیده می‌شود پایه‌های Hi و Lo در دو فانکشن برعکس یکدیگر داده شده است.

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
```




```
Network 2: Title:
```



شکل ۲-۹۵ برنامه مثال ۲-۶

تذکر: در عمل فرمان کنترلی پمپ یا ولو توسط یک کنترلر PID که فیدبک سطح را از یک ترانسمیتر می‌گیرد داده می‌شود. بحث کنترلر PID در فصل‌های بعدی خواهد آمد.

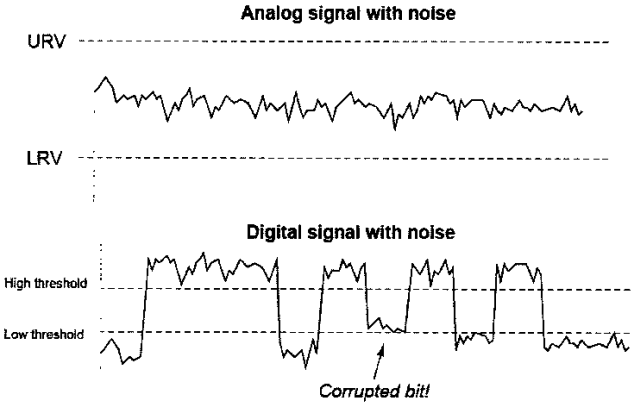
تمرین ۲-۶: در مثال ۲-۶ اگر دو فانکشن را به‌صورت سری در یک Network قرار دهیم چه تفاوت عملکردی نسبت به حالت قبل می‌بینیم؟

فصل ۲

۲-۵ بررسی تأثیر نویز روی سیگنال‌های آنالوگ

۲-۵-۱ شناخت تأثیر نویز

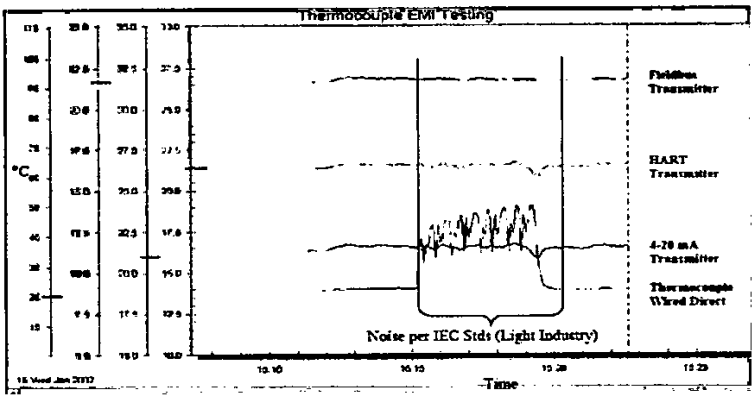
همانطور که در کتاب سطح مقدماتی اشاره شد، تأثیر مخرب نویز بر سیگنال‌های آنالوگ بیش از سیگنال‌های دیجیتال است زیرا در سیگنال‌های دیجیتال اگر سطح ولتاژ صفر یا یک به‌دلیل تأثیر نویز تغییر کند و این تغییرات از حد تعریف شده بیشتر نباشد، باز سیگنال صفر یا یک معتبر و قابل استفاده است.



شکل ۲-۹۶ مقایسه تأثیر نویز روی سیگنال آنالوگ و دیجیتال

این در حالی است که در سیگنال‌های آنالوگ اگر نویز منجر به تغییر سطح ولتاژ یا جریان شود، تفکیک آن نسبت به سیگنال واقعی امکان پذیر نیست. به‌عنوان مثال اگر ترانسمیتر فشار سیگنالی معادل ۵ ولت را به کنترلر ارسال کند و این سیگنال در طول مسیر به‌دلیل بروز نویز به 5.5 ولت افزایش یابد، کنترلر ۵٫۵ ولت را به‌عنوان فشار می‌شناسد و نمی‌تواند ۰/۵ ولت نویز را از ۵ ولت سیگنال تفکیک کند. برای نویزهایی که دامنه ضعیف دارند یا به‌صورت AC روی موج DC سوار می‌شوند می‌توان با استفاده از فیلترهای سخت افزاری یا نرم‌افزاری آنها را حذف نمود، ولی روش اصولی این است که با استفاده از کابل مناسب و کابل‌کشی مناسب میزان تأثیر نویز را به حداقل برسانیم.

در بین سیگنال‌های آنالوگ، سیگنال‌های میلی‌ولت نسبت به نویز، آسیب‌پذیرتر از سایرین هستند وسیگنال‌های جریانی کمتر تأثیر می‌گیرند. شکل ۲-۹۷ تأثیر نویز را روی سیگنال‌های آنالوگ مختلف مورد مقایسه قرار داده است. همانطور که در این شکل دیده می‌شود، سیگنال ترموکوپل بیش از جریانی تحت تأثیر قرار گرفته است.

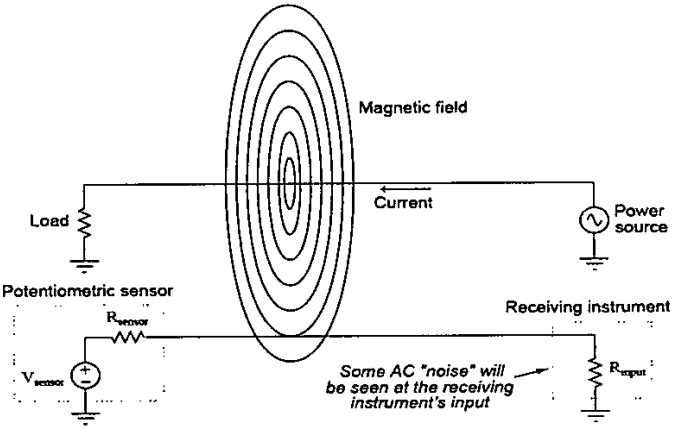


شکل ۲-۹۷ مقایسه تأثیر نویز روی سیگنال‌های آنالوگ مختلف در شرایط آزمایشی یکسان

۲-۵-۲ روش‌های کاهش تأثیر نویز

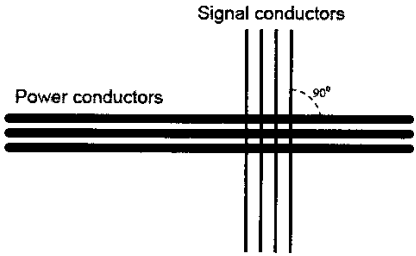
الف) جداسازی کابل‌های قدرت از کابل‌های سیگنال

میدان مغناطیسی ایجاد شده توسط کابل‌های قدرت می‌تواند منجر به اختلال در سیگنال آنالوگ شود. شکل ۲-۹۸ تأثیر این میدان را روی سیگنال ابزاردقیق نشان می‌دهد.



شکل ۲-۹۸ تأثیر میدان مغناطیسی کابل قدرت روی کابل سیگنال

برای کاهش این تأثیر، توصیه می‌شود کابل‌های سیگنال را تا حد امکان عمود بر کابل‌های قدرت قرار دهید. این روش تأثیر نویز مغناطیسی را به حداقل می‌رساند.

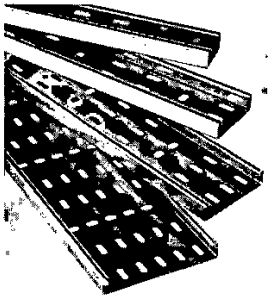


شکل ۲-۹۹ کابل‌کشی متقاطع به‌منظور کاهش تأثیر میدان مغناطیسی

فصل ۲

اگر لازم است کابل‌های سیگنال به موازات کابل‌های قدرت کشیده شوند هیچ‌گاه آنها را در مجاورت یکدیگر قرار ندهید. لازم است آنها را در سینی‌های فلزی کابل[1] مجزا قرار دهید که حدود ۳۰ سانتی‌متر با یکدیگر فاصله دارند.

به‌طور کلی هر جا تعداد زیادی کابل با ولتاژها و جریان‌های مختلف وجود داشته باشد، به‌منظور کاهش تأثیرات الکترومغناطیسی لازم است آنها را دسته بندی نموده و هر کدام را در سطح و فاصله خاصی نسبت به دیگری قرار داد. به‌عنوان مثال نباید یک کابل انتقال دیتا را با یک‌کابل ۱۰۰۰ ولت AC با جریان ۲۰۰ آمپری کنار هم در یک سطح قرار داد.



شکل ۲-۱۰۰ سینی کابل

در استاندارد IEC 61000-5 کلاس‌هایی که کابل‌های مختلف را طبقه‌بندی می‌کند ارائه شده است. این استاندارد ۵ سطح برای کابل‌ها تعریف می‌کند که سطح ۵ مربوط به ولتاژهای فشار قوی و فشار متوسط است. سایر سطوح به‌صورت زیر طبقه‌بندی شده‌اند:

Calss 1: کابل‌های حامل سیگنال‌های خیلی حساس مانند آنالوگ‌های ضعیف (میلی‌ولت) و سیگنال‌های شبکه

Calss 2: سیگنال‌های با حساسیت کمتر مانند آنالوگ‌های معمولی 4-20 mA یا 0-10 V و سیگنال‌های دیجیتال On/OFF و پالس‌های انکودرها

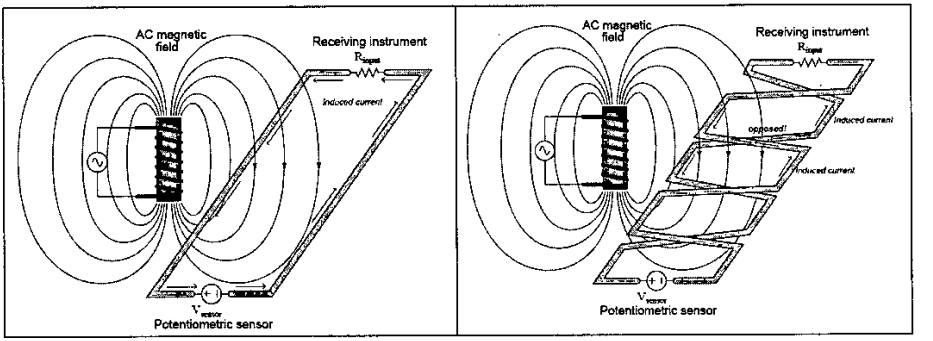
Calss 3: ولتاژهای توزیع AC کمتر از ۱۰۰۰ ولت و DC مانند ۴۸ ولت که تداخل اندکی ایجاد می‌کنند.

1. Cable Tray

Calss 4: سیگنال‌های قدرتی که به‌شدت تداخل ایجاد می‌کنند؛ مانند خروجی درایوها که برای موتورهای دور متغیر به‌کار می‌رود و یا تغذیه ماشین‌های جوشکاری.

ب) استفاده از کابل با سیم به هم تابیده

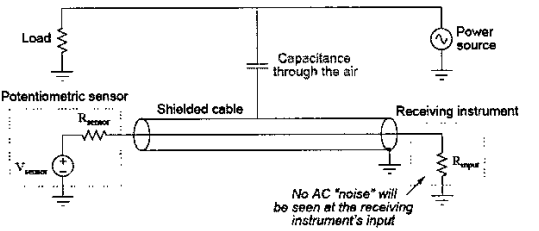
اگر دو سیم در کابل به موازات هم باشند، فلوی ایجاد شده توسط میدان مغناطیسی بیرونی جریانی را در مسیر بسته شده مدار الکتریکی القا می‌کند. اگر دو رشته سیم به‌صورت متقاطع باشند فلوی ایجاد شده در هر دو لوپ مجاور همدیگر را تضعیف می‌کنند و تأثیر به حداقل می‌رسد. این موضوع در شکل ۲-۱۰۱ نشان داده شده است.



شکل ۲-۱۰۱ تأثیر به هم تابیدگی سیم در حذف القای مغناطیسی

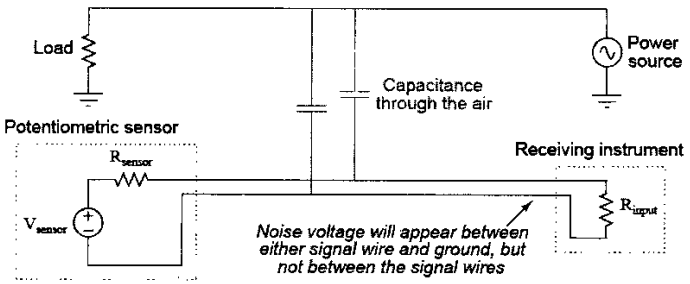
ج) استفاده از کابل شیلد دار

شیلد، کابل را در مقابل نویزهای خازنی مصون نگه می‌دارد. اگر کابل سیگنال در فاصله‌ای نسبت به یک هادی تحت ولتاژ مانند کابل قدرت قرار گیرد بین هادی‌های دو کابل، خازنی تشکیل می‌گردد که دی‌الکتریک آن عایق کابل و هواست. اگر از کابل شیلد دار استفاده شود، خازن بین سطح ولتاژ بیرونی و شیلد بسته می‌شود و منجر به تداخل سیگنال نمی‌شود.



شکل ۲-۱۰۲ تأثیر شیلد در حذف نویز خازنی

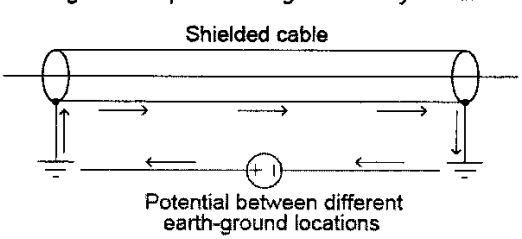
اگر سیگنال آنالوگ به‌صورت تفاضلی باشد، یعنی هیچ‌یک از دو سیم به زمین متصل نباشند در اینصورت تأثیر نویز خازنی روی هر دو سیم خواهد بود که تفاضل آنها حذف می‌گردد حتی اگر شیلد وجود نداشته باشد.



شکل ۲-۱۰۳ تأثیر سیگنال تفاضلی در حذف نویز خازنی

د) زمین کردن مناسب

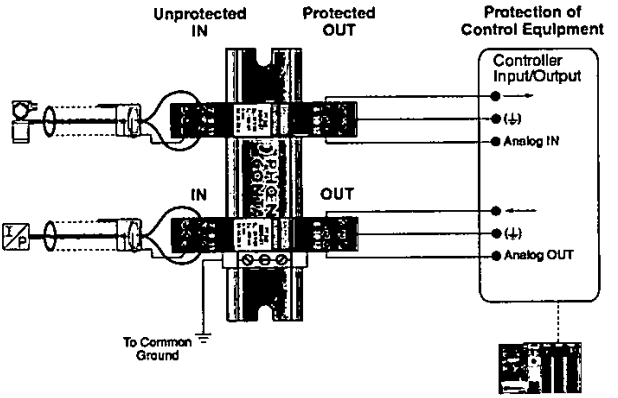
شیلد کابل لازم است به زمین متصل گردد. اگر سیستم زمین در سمت کنترلر و در سمت ترانسمیتر یا کنترل ولو یکسان نباشد، ضروری است اتصال زمین فقط در یک سمت انجام گیرد. در غیر اینصورت به‌دلیل هم پتانسیل نبودن دو طرف، جریان گردشی از شیلد عبور خواهد کرد که سیگنال آنالوگ را تحت تأثیر قرار می‌دهد.



شکل ۲-۱۰۴ عبور جریان گردشی از شیلد به‌دلیل اتصال نامناسب به زمین

ه) ایزوله‌سازی

با استفاده از ماژول‌های حفاظتی از عبور موج‌های گذرا و تأثیر روی کارت ورودی /خروجی کنترلر جلوگیری به‌عمل می‌آید. به شکل ۲-۱۰۵توجه کنید.



شکل ۲-۱۰۵ استفاده از مدارهای حفاظتی در مسیر سیگنال آنالوگ

۲-۶ پرسش و تحقیق

فانکشن‌های Scale نظیر FC105 سیگنال آنالوگ را به‌صورت خطی تبدیل می‌کنند. در صورتی‌که رفتار سیگنال آنالوگ به‌صورت غیر خطی باشد، چه باید کرد؟

۲-۷ تمرین

تمرین ۲-۳ را به‌صورتی کامل کنید که وقتی سطح مخزن از ۳ متر بیشتر است پمپی که کمتر کار کرده، روشن بماند.

فصل ۳

کار با Data Block و UDT

در این فصل ضمن معرفی انواع دیتابلاک، روش کار با آن و چگونگی استفاده از آن در برنامه‌نویسی تشریح می‌شود.

چکیده مطالب

- جایگاه دیتا بلاک در حافظه Load Memory است.
- از مزایای دیتا بلاک می‌توان به فضای زیاد، دسته بندی دیتا، امکان تعریف متغیرهای Complex اشاره کرد.
- دیتا بلاک به دو دسته اشتراکی و اختصاص تقسیم می‌شود.
- نوع اشتراکی برای همه بلاک‌های برنامه‌نویسی قابل دسترس است.
- نوع اختصاصی برای Function Block استفاده می‌شود.
- در نوع اشتراکی تعریف متغیرهای دیتا بلاک به عهده کاربر است ولی در نوع اختصاص متناسب با ساختار FB به‌طور خودکار ایجاد می‌شود.
- اختصاص مقدار اولیه به متغیرهای دیتا بلاک نکات خاصی دارد که بایستی با دقت انجام شود.
- اگر از آدرس DB در برنامه استفاده شود ولی DB به PLC دانلود نشده باشد منجر به فالت و توقف CPU می‌گردد.
- باز کردن دیتا بلاک می‌تواند با دستور OPN انجام شود.
- UDT برای تعریف یک ساختار مشخص از متغیرها استفاده می‌شود. با تعریف UDT می‌توان آنرا در DBهای مختلف به‌کار برد.

فصل ۳

۳-۱ مقدمه

همانطور که در کتاب سطح مقدماتی اشاره شد، دیتابلاک[1] که به اختصار DB نامیده می‌شود، بلاکی است که امکان برنامه‌نویسی در آن وجود نداشته و صرفاً برای کار با دیتا استفاده می‌شود. در برنامه می‌توان دیتاهایی را در DB ذخیره کرد یا دیتاهایی را از DB خواند. در این فصل ضمن معرفی انواع دیتابلاک به چگونگی ایجاد متغیر در آن و استفاده در برنامه‌نویسی پرداخته شده و مزایای آن نسبت به متغیرهای حافظه تشریح می‌گردد.

۳-۲ جایگاه Data Block در حافظه

در کتاب سطح مقدماتی نواحی حافظه CPU تشریح گردید و در آنجا دیدیم که حافظه CPU دارای سه بخش اصلی است. این بخش‌ها در شکل ۳-۱ نشان داده شده‌اند.

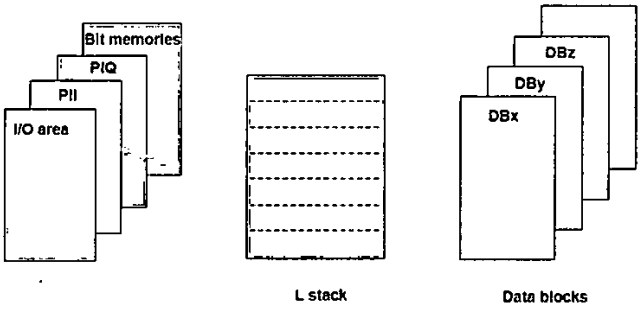
Load Memory	*Work Memory*	*System Memory*

شکل ۳-۱ بخش‌های حافظه CPU

ناحیه‌ای که مربوط به حافظه سیستم بوده و متغیرهای حافظه در آن ذخیره می‌شوند دارای بخش‌های مختلفی است. این نواحی عبارتند از:

- **نواحی مربوط به دیتاهای ورودی و خروجی (PII، PIQ، ناحیه I/O):** این نواحی به‌منظور ذخیره‌سازی دیتاهای مربوط به ورودی‌ها و خروجی‌ها استفاده می‌شوند. محل قرارگیری این نواحی در System Memory می‌باشد. در برخی از CPUهای جدید این نواحی در Work Memory قرار گرفته‌اند.
- **Bit Memories:** به‌منظور ذخیره‌سازی نتایج میان‌برنامه و اطلاعاتی که قرار است در جاهای مختلف برنامه از آن استفاده شود به‌کار می‌رود. محل قرارگیری این ناحیه در System Memory می‌باشد.
- **L stack (Local Data Stack):** به‌منظور ذخیره‌سازی دیتاهای مربوط به متغیرهای محلی از نوع Temp به‌کار می‌رود که در فصل بعد تشریح خواهد شد. محل قرارگیری آن در System Memory می‌باشد.

1. Data Block



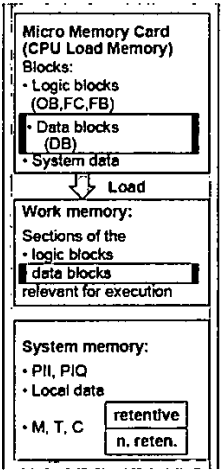
شکل ۳-۲ نواحی مختلف ذخیره‌سازی دیتا در CPU

فصل ۳

اکنون باید دید جایگاه دیتابلاک در کدام بخش حافظه است. دیتابلاک برخلاف متغیرهای حافظه در Load Memory قرار می‌گیرد و این یک مزیت بزرگ برای DB نسبت به متغیرهای حافظه که در System Memory قرار می‌گیرند محسوب می‌شود. زیرا System Memory قابل افزایش نیست ولی Load Memory با اضافه کردن کارت حافظه یا ارتقاء کارت حافظه قابل افزایش است. سایر مزایای دیتا بلاک در ادامه مورد بحث قرار می‌گیرد.

جایگاه دیتابلاک در S7-300

همانطور که اشاره شد، محل قرارگیری Data Block در Load Memory می‌باشد. شکل ۳-۳ محل قرارگیری دیتابلاک در حافظه CPUهای S7-300 را نشان می‌دهد. همانطور که مشخص است، دیتا بلاک در Load Memory قرار گرفته و درصورتی‌که از آن در برنامه استفاده شود، به حافظه‌ی کاری[1] منتقل می‌شود.



شکل ۳-۳ محل قرارگیری دیتابلاک در حافظه CPU در S7-300

1. Work Memory

در S7-300 با CPUهای جدید، حافظه Load Memory کارت فلش است که در اسلات CPU قرار می‌گیرد از اینرو دیتا بلاکی که در این حافظه قرار می‌گیرد دارای ویژگی Retentivity یعنی ماندگاری است؛ یعنی با قطع تغذیه، اطلاعات آن حفظ می‌شود.

نکته ای که بایستی به آن توجه نمود این است که ایجاد دیتا بلاک به دو طریق می‌تواند انجام شود:

۱- ایجاد دیتا بلاک به‌صورت دستی توسط کاربر

۲- ایجاد دیتا بلاک توسط SFC در برنامه

در روش اول، دیتابلاک پس از دانلود به حافظه EPROM منتقل می‌شود ولی در روش دوم در حافظه RAM قرار می‌گیرد که خاصیت ماندگاری ندارد. روش دوم در کتاب سطح تکمیلی مورد بحث قرار خواهد گرفت.

برخی از CPUهای S7-300 مانند CPU315 از دیتابلاک‌های غیرماندگار نیز پشتیبانی می‌نمایند. دراینصورت می‌توان دیتابلاک مربوطه را در قسمت Retentive Memory از تنظیمات CPU به‌صورت ماندگار تعریف نمود.

حجم Data Block

تعداد و اندازه دیتابلاک‌های هر CPU، به مشخصات فنی آن CPU وابسته است. در کاتالوگ فنی هر CPU معمولاً تعداد دیتابلاک‌هایی که CPU ساپورت می‌کند و نیز حداکثر اندازه آن ذکر می‌شود.

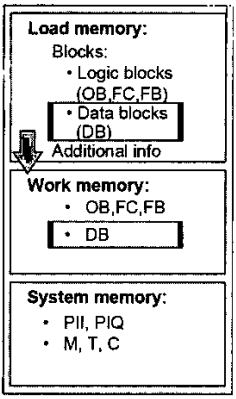
در جدول ۳-۱ اندازه و تعداد دیتابلاک‌های برخی از CPUهای S7-300 نشان داده شده است.

جدول ۳-۱ اندازه فضای دیتابلاک در برخی CPUهای S7-300

	CPU 312	CPU 315-2 DP	CPU 317-2 PN/DP
DB • Number, max.	1 024; Number range: 1 to 16000	1 024; Number range: 1 to 16000	2 048; Number range: 1 to 16000
DB • Size, max.	32 Kibyte	64 Kibyte	64 Kibyte

جایگاه دیتابلاک در S7-400

در S7-400 حافظه داخلی Load Memory از جنس RAM است. وقتی دیتا بلاک دانلود می‌شود، به این حافظه منتقل می‌گردد، بنابراین در صورتی ماندگار خواهد بود که باتری پشتیبان وجود داشته باشد.



شکل ۳-۴ محل قرارگیری دیتابلاک در حافظه CPU در S7-400

اگر CPU400 دارای کارت حافظه از نوع Flash باشد و دانلود دیتا بلاک از طریق گزینه Download user Program to Memory Card که در منوی PLC در دسترس است انجام شود، در اینصورت دیتابلاک به کارت حافظه Flash منتقل شده و ویژگی ماندگار بودن را دارا خواهد بود.

۳-۳ مزایا و معایب دیتابلاک Data Block

از آنجا که دیتابلاک برای خواندن و نوشتن دیتا استفاده می‌شود، اگر بخواهیم مزایا و معایب آنرا ذکر کنیم بایستی آنرا با متغیرهای حافظه[1] مقایسه نماییم. در اینصورت مزیت‌های زیادی را برای دیتابلاک می‌توان ذکر نمود. به همین علت در برنامه‌نویسی فرآیندهای صنعتی ترجیح می‌دهند از DB به‌جای Memory Bits استفاده کنند.

۳-۳-۱ مزایای دیتابلاک

این مزیت‌ها عبارتند از:

۱- حجم فضای زیاد و قابل توسعه

یکی از مواردی که به‌عنوان یک مزیت برای دیتابلاک محسوب می‌شود، محل قرارگیری آن در حافظه است. همانطور که اشاره شد، محل قرارگیری دیتابلاک در حافظه بارگذاری[2] می‌باشد. می‌دانیم که Load Memory را می‌توان با استفاده از کارت حافظه توسعه داد، بنابراین در صورتی‌که دیتاهای موجود در دیتابلاک زیاد شوند به‌نحوی که با کمبود حافظه Load Memory مواجه شویم، می‌توان با استفاده ازکارت حافظه حجم حافظه Load Memory را افزایش داد. از

1. Bit Memory
2. Load Memory

طرفی تعداد دیتابلاک‌ها در هر CPU تعداد قابل توجهی است و حجمی که هر دیتابلاک می‌تواند به‌خود اختصاص دهد نیز حجم بالایی است، در حالی که در مواردی که از Bit Memoryها استفاده شود به‌دلیل حجم پایین آنها ممکن است به‌سرعت حافظه آنها تمام شود. از طرفی محل قرارگیری Bit Memoryها در System Memory است که حافظه غیر قابل توسعه CPU محسوب می‌شود، بنابراین امکان افزایش آن وجود ندارد.

۲- دسته‌بندی و مدیریت دیتا

حافظه Memory دارای دسته‌بندی خاصی نیست. اگر چه برنامه‌نویس می‌تواند با دقت ناحیه Memory Bits را به چند بخش تقسیم کرده و هر بخش را برای کار خاصی استفاده کند، ولی باز ساختار به‌خوبی دیتابلاک دارای نظم و دسته‌بندی نخواهد بود. در کار با دیتابلاک کاربر می‌تواند برای هر دسته از نیازهای کاری خود یک DB مجزا اختصاص دهد. به‌عنوان مثال می‌تواند برای هر ناحیه از فرآیند یک DB مجزا تعریف کند یا سیگنال‌های آنالوگ و دیجیتال را در DBهای مستقلی قرار دهد. همچنین برای ارتباط با سیستم مانیتورینگ یک DB خاص تعریف کند و برای تبادل دیتا بین کنترلرها روی شبکه DBهای جداگانه‌ای اختصاص دهد. این موارد با Bit Memories امکان پذیر نیست یا به‌دشواری قابل انجام است.

۳- امکان تعریف متغیرهای Complex

برخی از دیتاها نیاز به حافظه‌ای بیشتر از ۴ بایت دارند. ماکزیمم حافظه‌ای که برای Bit Memoryها می‌توان تعریف نمود ۴ بایتی است که به‌صورت MD معرفی می‌شود. پس اگر دیتایی مانند Date and Time که نیاز به حافظه‌ای بیش از ۴ بایت دارد مورد نیاز باشد نمی‌توان آنرا در Memory ذخیره کرد. در حالی که در دیتا بلاک امکان معرفی دیتاهایی از این نوع به‌سادگی وجود دارد.

استفاده از Array و Struct نیز که از دیتاهای Complex محسوب می‌شوند در DB امکان‌پذیر است ولی در Memory Bits این کار امکان‌پذیر نیست. متغیرهای Complex در فصل‌های بعدی مورد بحث قرار می‌گیرند.

۴- آدرس‌دهی خودکار

یکی دیگر از مزایای دیتابلاک آدرس‌دهی خودکار است، به‌نحوی که امکان استفاده از آدرس تکراری که منجر به ایجاد خطای Over Lap می‌شود را از بین می‌برد. هر سطر دیتا بلاک دارای نام مشخصی است که برنامه‌نویس با استفاده از آن دیگر نگران تلاقی آدرس نخواهد بود ولی در Bit Memory حتی اگر آدرس سمبلیک استفاده شود باز احتمال خطا وجود دارد.

۵- امکان ماندگاری

اگر DB روی کارت فلش ذخیره شده باشد، دیتاهای آن با قطع و وصل تغذیه حفظ می‌شوند. این ویژگی برای Bit Memory بایستی به‌صورت دستی در ناحیه Retentive Memory از مشخصات CPU تعریف شود. به کتاب سطح مقدماتی مراجعه کنید.

۶- امکان اختصاص مقدار اولیه به متغیرها

در هنگام تعریف متغیرها در دیتا بلاک امکان اختصاص مقدار اولیه[1] به آنها وجود دارد ولی این ویژگی در Bit Memories موجود نیست.

۷- امکان محافظت در برابر نوشتن

یک دیتابلاک را می‌توان در برابر نوشتن محافظت نمود، یعنی آنرا به‌صورت Read Only تعریف کرد. این کار برای برخی دیتاها مانند مقادیر مبنای فرآیند مفید است. این ویژگی در Bit Memories وجود ندارد.

۳-۳-۲ معایب دیتا بلاک

دیتا بلاک‌پس‌ازایجادشدن باید به CPU دانلود شود. اگر قبل از دانلود در برنامه به آدرسی از دیتا بلاک ارجاع شود CPU دچار مشکل شده و درصورت عدم وجود وقفه‌های خاص CPU متوقف خواهد شد. این مشکل در Bit Memories وجود ندارد، زیرا در حافظه سیستم موجود هستند و نیاز به دانلود ندارند.

فصل ۳

۳-۴ انواع Data Block

همانطور که در کتاب سطح مقدماتی اشاره شد، دیتا بلاک به دو نوع زیر تقسیم می‌شود:

- Shared DB
- Instance DB

الف) نوع Shared (اشتراکی)

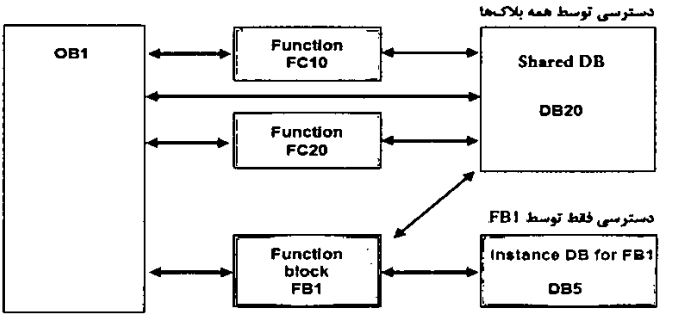
این نوع دیتا بلاک به‌صورت اشتراکی بوده و همه‌ی بلاک‌های برنامه‌نویسی می‌توانند از آن استفاده نمایند. ساختار این DB می‌تواند توسط کاربر تغییر داده شود.

ب) نوع Instance (اختصاصی)

این نوع دیتابلاک مخصوص FB یا SFBها می‌باشد و به‌عنوان حافظه‌ای برای آنها در نظر گرفته می‌شود. در واقع هر FB یا SFB دارای یک دیتابلاک اختصاصی می‌باشد که برخی از متغیرها و پارامترهای آن در این دیتابلاک ذخیره می‌شود. ساختار این DB وابسته به پارامترهای FB/SFB است و به‌طور خودکار ایجاد می‌شود و کاربر نمی‌تواند این ساختار را تغییر دهد.

در شکل ۳-۵ نحوه دسترسی به دیتابلاک اشتراکی و اختصاصی نشان داده شده است.

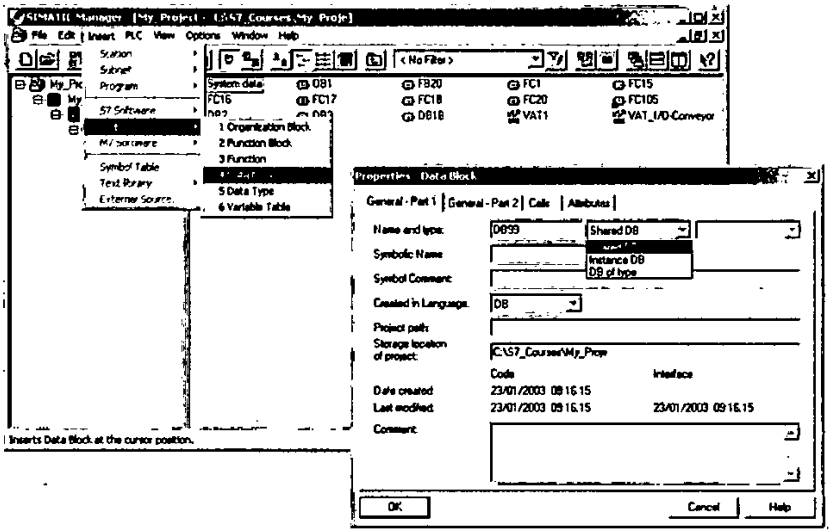
1. Initial Value



شکل ۳-۵ دیتابلاک اشتراکی و اختصاصی

۳-۵ ایجاد دیتابلاک در محیط Simatic Manager

برای ایجاد دیتابلاک روش‌های مختلفی وجود دارد. یکی از ساده‌ترین روش‌ها در شکل ۳-۶ نشان داده شده است.



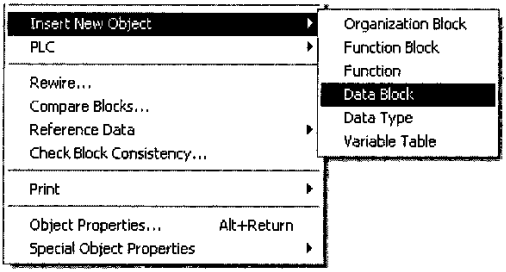
شکل ۳-۶ ایجاد دیتابلاک

همانطور که در شکل ۳-۶ مشخص است، لازم است ابتدا شماره‌ای به DB اختصاص دهیم. این شماره نمی‌تواند صفر باشد و سایر اعداد بزرگتر از صفر مجاز هستند.

در هنگام ایجاد دیتابلاک، باید نوع آنرا نیز مشخص نمود. گزینه‌های قابل انتخاب عبارتند از:

- **Shared:** در اینصورت دیتابلاک از نوع اشتراکی ایجاد می‌شود.
- **Instance:** در اینصورت دیتابلاک از نوع اختصاصی بوده و باید در کادر مجاور، بلاکی که قرار است این دیتابلاک به‌عنوان دیتابلاک اختصاصی آن باشد را مشخص نمود.
- **DB of type:** در اینصورت دیتابلاک مطابق با الگوی موجود در UDT که قبلاً ایجاد شده باشد، ساخته می‌شود. برای انجام این روش قبلاً باید UDT مربوطه ساخته شده باشد. UDT در فصل بعد توضیح داده می‌شود.

علاوه بر روش فوق می‌توان مشابه ایجاد سایر بلاک‌ها در محیط کاری Simatic Manager کلیک راست نموده و مشابه شکل ۳-۷، با انتخاب گزینه Insert New Object، گزینه Data Block را انتخاب نمود.



شکل ۳-۷ روش دیگر ایجاد دیتابلاک

ادامه مراحل، مشابه موارد ذکر شده در روش قبل است.

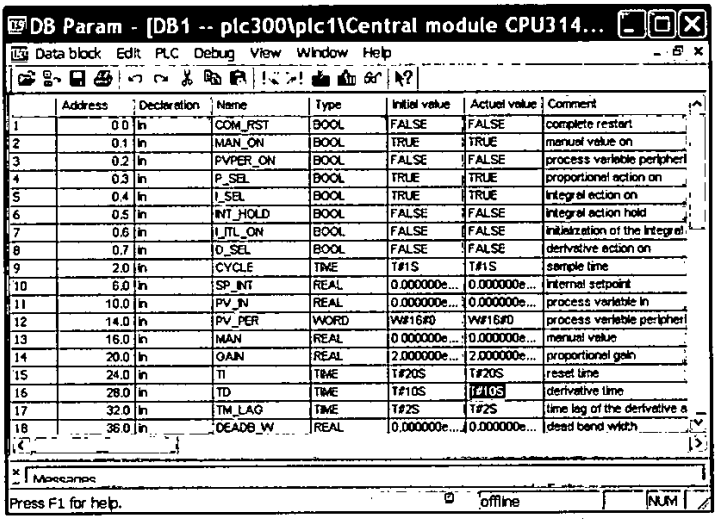
۳-۶ ساختار Data Block

۳-۶-۱ ساختار Instance DB

ساختار دیتابلاک‌های اختصاص[1] به‌طور خودکار براساس ورودی و خروجی و سایر متغیرهای تعریف شده در FB یا SFB شکل می‌گیرد.

به‌عنوان مثال DB زیر به‌صورت Instance به FB41 زیمنس که از فانکشن بلاک‌های کتابخانه زیمنس است اختصاص داده شده است. پس از باز کردن DB ساختار آنرا به‌صورت شکل زیر می‌بینیم. همانطور که مشاهده می‌شود امکان تغییر یا اضافه کردن در آن وجود ندارد.

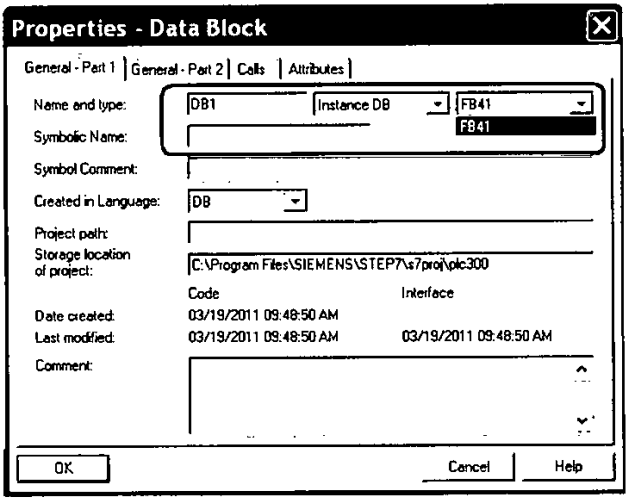
1. Instance DB



شکل ۳-۸ DB اختصاصی FB41

اگر کاربر بخواهد DBهای نوع Instance را ایجاد کند به دو روش می‌تواند عمل کند:

روش اول: ابتدا FB یا SFB مورد نظر را به پوشه بلاک برنامه وارد کرده سپس با ایجاد DB جدید در پنجره زیر، FB مورد نظر را به آن اختصاص می‌دهد.

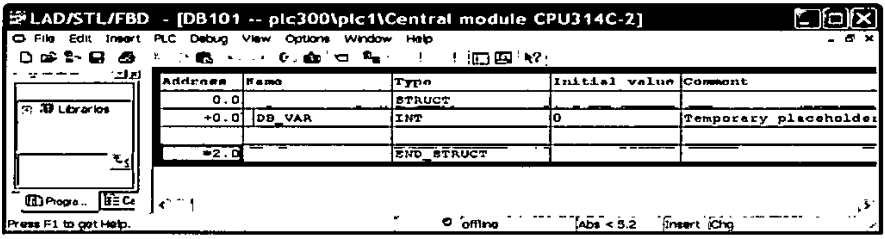


شکل ۳-۹ اختصاص DB به FB

روش دوم: در این روش که نسبت به روش اول متداول‌تر است، نیازی به ایجاد Instance DB توسط کاربر نیست، بلکه به محض صدا زدن FB یا SFB در برنامه DB را به آن اختصاص می‌دهیم و برنامه به‌طور خودکار آنرا ایجاد می‌کند. این روش در فصل بعد تشریح شده است.

۳-۶-۲ ساختار Shared DB

ساختار این نوع DB توسط‌کاربر تعریف می‌شود. وقتی DB از نوع اشتراکی را ایجاد و باز کنیم، محیطی مانند شکل ۳-۱۰ خواهیم دید. همانطور که دیده می‌شود یک سطر دیتا با نام DB_VAR به‌صورت پیش‌فرض از نوع Integer در آن وجود دارد. سطرهای دیگر می‌تواند توسط کاربر ایجاد گردد.



شکل ۳-۱۰ محیط دیتابلاک اشتراکی

همانطور که در شکل ۳-۱۰ دیده می‌شود، در دیتا بلاک محیطی برای برنامه‌نویسی وجود ندارد. محیط دیتابلاک به لحاظ ظاهری شبیه محیط Symbol Table است که برای تعریف سمبل‌ها استفاده می‌شود. در دیتابلاک هر متغیر در یک سطر تعریف می‌شود. برای هر متغیر تعدادی گزینه قابل تنظیم است که عبارتند از:

Address: در این ستون آدرس اختصاص داده شده به متغیر نشان داده می‌شود. این آدرس توسط سیستم انتخاب شده و غیر قابل تغییر است، مگر اینکه سطری را که متغیر در آن ایجاد شده جابه‌جا نماییم که در اینصورت از طرف سیستم آدرس جدیدی اختصاص داده می‌شود. در هر صورت امکان اینکه کاربر بتواند آدرس اختصاص داده شده را Modify نماید وجود ندارد. این مسئله باعث جلوگیری از تداخل آدرس‌های می‌شود.

Name: در این ستون می‌توان نام متغیر را به دلخواه تعریف نمود. نام متغیر مانند یک سمبل در نظر گرفته می‌شود، یعنی در برنامه هم می‌توان آدرس متغیر را به‌نحوی که در ادامه ذکر می‌شود وارد نمود و هم می‌توان نام متغیر را وارد نمود. این نام در جدول سمبل‌ها ظاهر نمی‌شود.

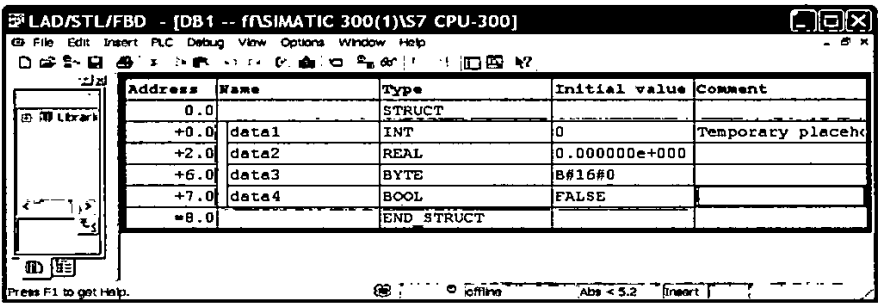
Type: در این قسمت می‌توان نوع متغیر را تعیین نمود. در واقع در این ستون Data Type متغیر تعیین می‌شود و همه دیتاهای نوع Elementary و Complex قابل انتخاب هستند.

Initial Value: در این ستون مقدار اولیه دیتابلاک تنظیم می‌شود.

Comment: در این ستون می‌توان یک توضیح دلخواه در مورد متغیر ایجاد شده وارد نمود.

وقتی سطر جدیدی در دیتا بلاک تعریف می‌کنیم با توجه به نوع دیتای آن آدرس سطر بعد و حجم کل DB که در انتها نشان داده می‌شود تغییر می‌کند. به‌عنوان مثال در شکل زیر چون سطر اول از جنس integer است دو بایت اشغال می‌کند بنابراین سطر دوم از آدرس 2.0 شروع شده است. در سطر دوم، دیتا از نوع Real است که چهار بایت را اشغال می‌کند بنابراین سطر سوم از آدرس 6.0 شروع می‌شود.

در سطر چهارم با وجود اینکه دیتا از نوع Bool که یک بیت فضا نیاز دارد می‌باشد ولی یک بایت فضا به آن اختصاص می‌یابد. از این فضا بعداً نیز می‌توان استفاده کرد، یعنی اگر سطر پنجم از نوع Bool اضافه شود سایز کل دیتا بلاک که تا اینجا ۸ بایت است تغییر نخواهد کرد.



شکل ۳-۱۱ اندازه و حجم آدرس‌های DB

تذکر: در برخی موارد که نوع دیتای یک سطر را تغییر می‌دهیم. سطر به حالت قرمز رنگ در می‌آید که نشان دهنده خطاست. علت خطا عددی است که به‌صورت پیش‌فرض در ستون Initial Value نوشته شده و مربوط به دیتای قبلی است. با حذف این عدد و ذخیره‌سازی DB مشکل برطرف می‌شود. به‌طور کلی توصیه می‌گردد که در هنگام تعریف سطرهای DB ستون Initial Value را خالی بگذارید تا خود برنامه آنرا پر کند.

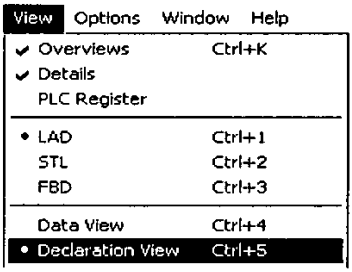
۳-۷ انواع نمایش محیط دیتابلاک

محیط یک دیتابلاک را به سه صورت می‌توان نمایش داد:

- Declaration View
- Data View
- Online View

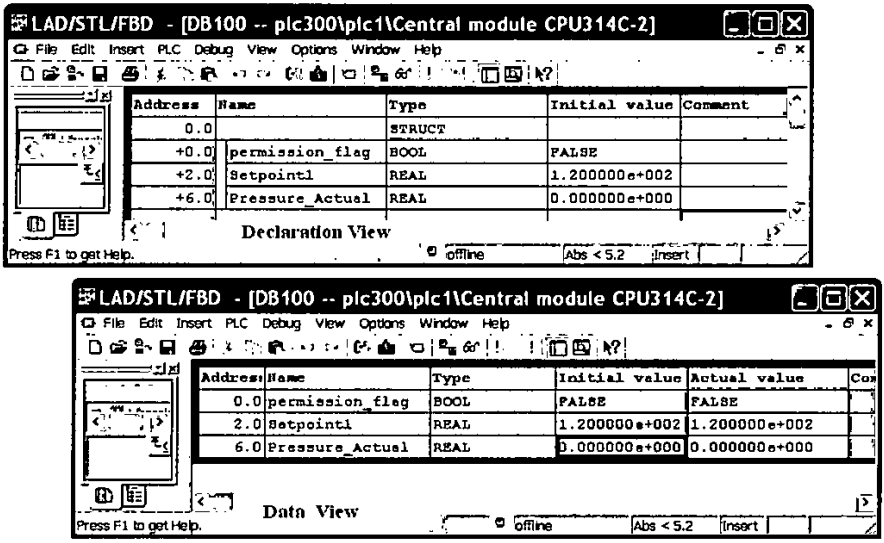
الف) Declaration View: در این نمایش که در قسمت قبل توضیح داده شد، امکان ایجاد متغیرها و تعیین نام سمبولیک و نوع آنها وجود دارد.

ب) Data View: این نمایش به منظور Monitor نمودن دیتاهای در حالت Online به‌کار می‌رود. در این نمایش ستون Actual Value نیز نمایش داده شده که در آن مقدار واقعی متغیر را می‌توان دید.



شکل ۳-۱۲ انتخاب نوع نمایش متغیرهای ایجاد شده در دیتابلاک

شکل ۳-۱۳ یک دیتابلاک را در دو نمایش Declaration View و Data View نشان می‌دهد.



شکل ۳-۱۳ دیتابلاک در دو نمایش Declaration View و Data View

ج) DB در حالت Online

در حالت Online می‌توان محتویات دیتابلاک موجود در حافظه CPU را مشاهده نمود. برای اینکه بتوان یک DB را در حالت Online مشاهده نمود می‌توان به دو روش عمل کرد:

۱- با استفاده از آیکن مانیتور در نوارابزار

۲- با استفاده از منوی File > open online

در روش اول روی آیکن Monitor که به شکل عینک است کلیک کنیم. در این حالت دیتاهای واقعی لحظه به لحظه در این محیط نمایش داده می‌شوند.

@DB1 -- "Motor Data" -- plc 1_12\SIMATIC 300(1)\CPU 314C-2 DP\...\DB1 ONLINE

Address	Name	Type	Initial value	Actual value	Comment
0.0	motor_data.speed	INT	0	0	
2.0	motor_data.temp	REAL	0.000000e+000	30.0	
6.0	motor_data.current	REAL	0.000000e+000	0.0	

شکل ۳-۱۴ Monitor نمودن DB1

در روش دوم از منوی File گزینه Open Online را انتخاب می‌کنیم. تفاوت این روش با روش اول در این است که در اینجا DB از داخل حافظه PLC باز می‌شود ولی در روش اول DB از پروژه Offline باز شده و دیتاها را از PLC می‌گیرد. شکل ۳-۱۵ باز شدن DB به روش دوم را نشان می‌دهد.

DB1 -- "Motor Data" -- plc 1_12\SIMATIC 300(1)\CPU 314C-2 DP\...\DB1 ONLI...

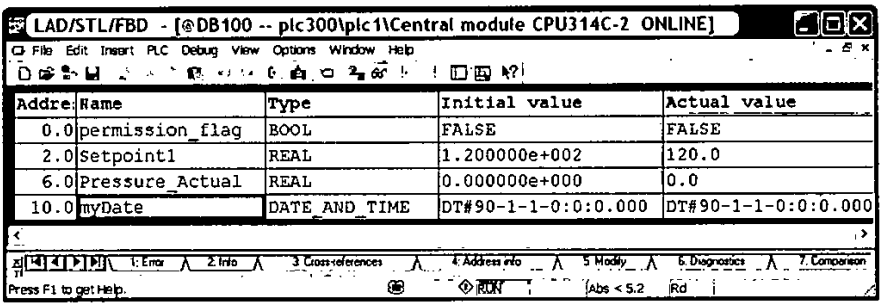
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	motor_data	UDT1		
=10.0		END_STRUCT		

شکل ۳-۱۵ نمایش DB1 در حالت Online

نکات قابل توجه

- در مورد نمایش Online باید توجه داشت که محتویات درون DB که در PLC قرار دارد نمایش داده می‌شود.
- در صورتی‌که در حالت Online تغییری در متغیرهای دیتابلاک بوجود آوریم، این تغییرات فقط روی دیتابلاک موجود در حافظه PLC اعمال شده و به دیتابلاکی که درون نرم‌افزارSimatic Manager است (Offline) اعمال نمی‌شود.
- دیتابلاک‌هایی که با دستورات برنامه‌نویسی ایجاد می‌شوند را فقط می‌توان در حالت Online مشاهده نمود و در حالت Offline امکان مشاهده و مانیتور آنها وجود ندارد.

نکته: در ادامه خواهیم دید که در دیتا بلاک می‌توان برخی متغیرهایی که از ۳۲ بیت بیشتر هستند نیز تعریف نمود. به‌عنوان نمونه می‌توان یک سطر از جنس Date_and_Time که فضایی بیش از ۴ بایت اشغال می‌کند را تعریف کرد. اگر متغیرهایی از این نوع تعریف شده باشند توسط مانیتور کردن یا در پنجره online نمی‌توان مقدار آنها را دید و مقدار Actual Value آنها مانند شکل زیر به رنگ خاکستری نمایش داده می‌شود.

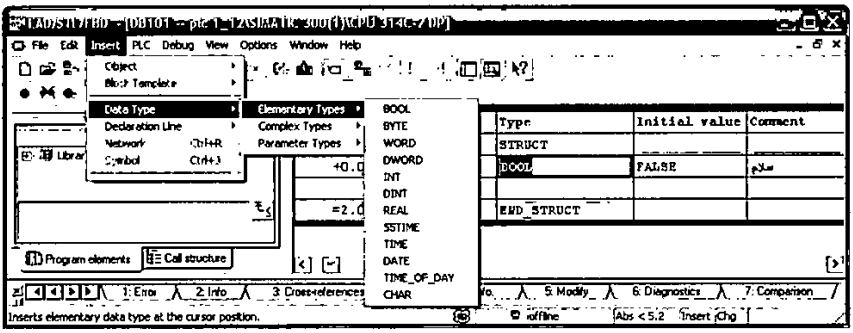


شکل ۳-۱۶ عدم امکان مانیتورینگ برای متغیرهای بیش از ۳۲ بیت

فصل ۳

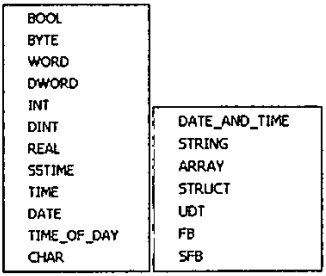
۳-۸ انواع متغیرهای قابل تعریف در Data Block

همانطور که اشاره شد، برای ایجاد متغیر در دیتابلاک کافی است که در ستون Name نام متغیر را وارد نموده و در ستون Type نوع آنرا مشخص کرد. شکل ۳-۱۷ نحوه تعیین نوع دیتا را برای متغیر ایجاد شده نشان می‌دهد.



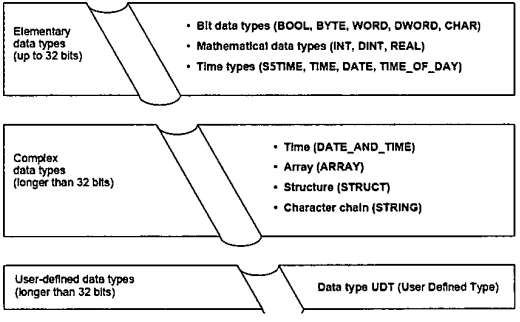
شکل ۳-۱۷ ایجاد متغیر در دیتابلاک

به‌طور کلی انواع دیتاهای قابل تعریف در دیتابلاک عبارتند از:



شکل ۳-۱۸ انواع دیتاها

از بین انواع فوق، گزینه‌های FB و SFB غیرقابل انتخاب می‌باشند. شکل ۳-۱۹ مقایسه کلی بین انواع دیتاها را ارئه می‌کند. برای شناخت جزئیات انواع متغیرهای فوق به کتاب سطح مقدماتی مراجعه نمایید.



شکل ۳-۱۹ گروه‌های دیتا

۳-۹ آدرس‌دهی متغیرهای ایجاد شده در دیتابلاک

پس از اینکه متغیرهای مورد نظر در دیتابلاک ایجاد شد، لازم است در برنامه از آدرس آنها استفاده نمود. برای این کار باید بتوان متغیرهای موجود در دیتابلاک را آدرس‌دهی نمود. آدرس‌دهی را می‌توان به دو روش انجام داد، به‌صورت مطلق یا بصورت سمبولیک. در آدرس‌دهی مطلق، آدرس به فرمت اصلی نوشته شده ولی در آدرس‌دهی سمبولیک نام متغیر وارد می‌شود.

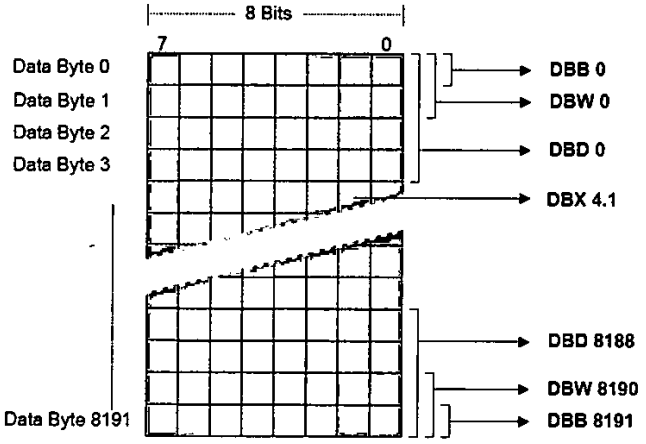
آدرس دهی مطلق[1]

در حالت آدرس‌دهی مطلق، هر چهار حالت آدرس‌دهی زیر قابل انجام است:

- آدرس‌دهی بیتی
- آدرس‌دهی بایتی
- آدرس‌دهی Word
- آدرس‌دهی Dword

شکل ۳-۲۰ فرمت‌های مختلف آدرس‌دهی دیتابلاک را نشان می‌دهد.

1. Absolute



شکل ۳-۲۰ انواع آدرس‌دهی مطلق متغیرهای دیتابلاک

فصل ۳

مثال ۳-۱

با استفاده از دیتابلاک مقدار دما، سرعت و حالت خاموش یا روشن بودن یک موتور را ذخیره نمایید.

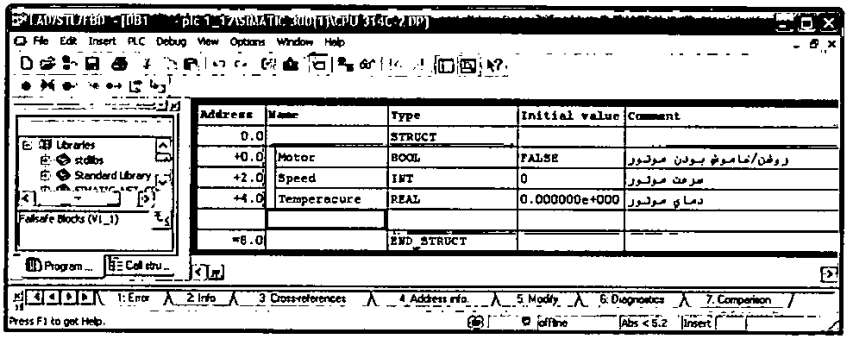
- مقدار دما به فرم Real از حافظه MD0 خوانده می‌شود.
- میزان سرعت به فرم INT از حافظه MW4 خوانده می‌شود.
- روشن بودن موتور توسط M6.0 مشخص می‌شود.

حل: ابتدا یک دیتابلاک با شماره دلخواه (مثلاً DB1) ایجاد نموده و سپس سه متغیر زیر را در آن ایجاد می‌نماییم:

۱- Motor از نوع Bool

۲- Speed از نوع Integer

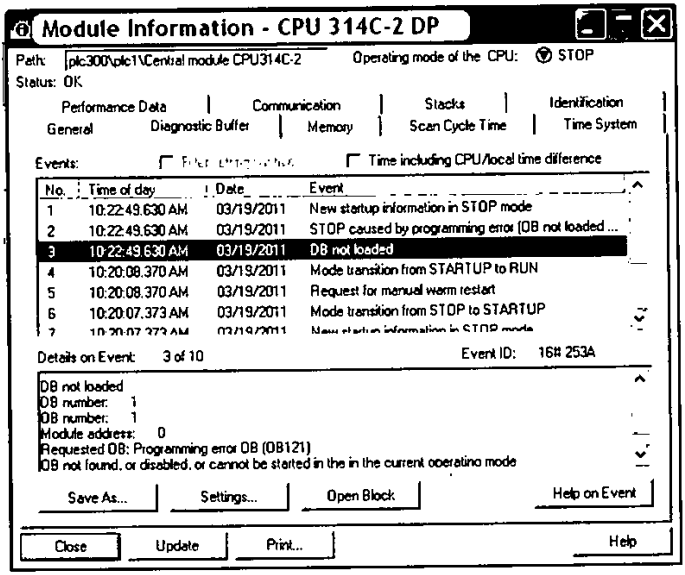
۳- Temperature از نوع Real



شکل ۳-۲۱ ایجاد متغیرهای لازم برای مثال ۳-۱ در DB1

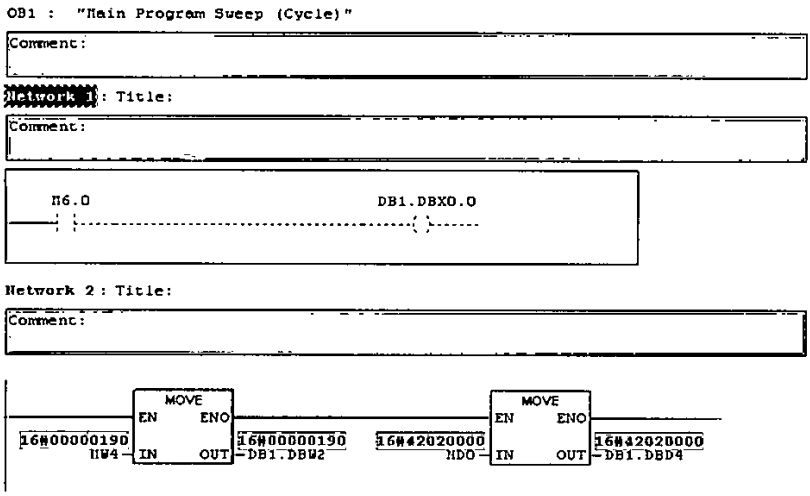
برای انتقال مقادیر دما،سرعت و خاموش یا روشن بودن موتور را از آدرس‌های مربوطه در Memoryها به آدرس‌های دیتابلاک می‌توان مراحل زیر را دنبال نمود:

- برای دما می‌توان توسط دستور Move، محتویات MD0 را به آدرس DB1.DBD4 منتقل نمود.
- برای سرعت نیز می‌توان توسط دستور Move، MW4 را به آدرس DB1.DBW2 منتقل نمود.
- برای وضعیت موتور می‌توان از یک تیغه باز با آدرس M6.0 و یک Coil با آدرس DB1.DBx0.0 استفاده نمود.
- پس از اینکه متغیرها در دیتابلاک ایجاد شدند، باید دیتابلاک را Save و Download نمود. بدیهی است که عدم دانلود دیتابلاک و استفاده از آدرس‌های آن در برنامه می‌تواند منجر به توقف CPU گردد و در بافر CPU مانند شکل ۳-۲۲ پیغام DB not Loaded ثبت می‌گردد.



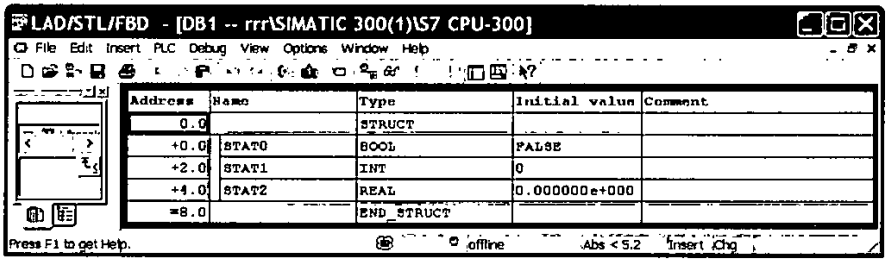
شکل ۳-۲۲ وضعیت بافر CPU در شرایط فالت DB

پس ار دانلود DB برنامه را در حالت Online مانند شکل ۳-۲۳ خواهیم دید.



شکل ۳-۲۳ برنامه مورد نظر جهت مثال ۳-۱

تذکر: اسامی وارد شده در سطرهای دیتا بلاک به PLC دانلود نمی‌شوند و در پروژه روی کامپیوتر باقی می‌مانند، بنابراین اگر DB را از PLC آپلود کنیم اسامی وارد شده در سطرها در دیتا بلاک آپلود شده ظاهر نمی‌شوند و به‌جای آنها اسامی STAT مانند شکل زیر نمایش داده می‌شود. همانطور که مشخص است Commentها نیز پس از آپلود خالی هستند.



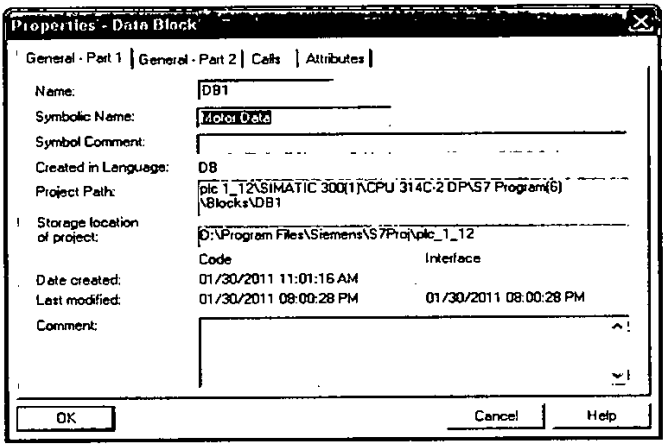
شکل ۳-۲۴ نمونه دیتا بلاک آپلود شده

آدرس‌دهی با سمبل[1]

در آدرس‌دهی سمبلیک به‌جای استفاده از آدرس دیتاهای DB به نام آنها اشاره می‌شود. برای این منظور بهتر است ابتدا به خود DB نیز نام سمبلیک اختصاص داده شود تا نام متغیرها در برنامه به‌صورت کامل دیده شوند.

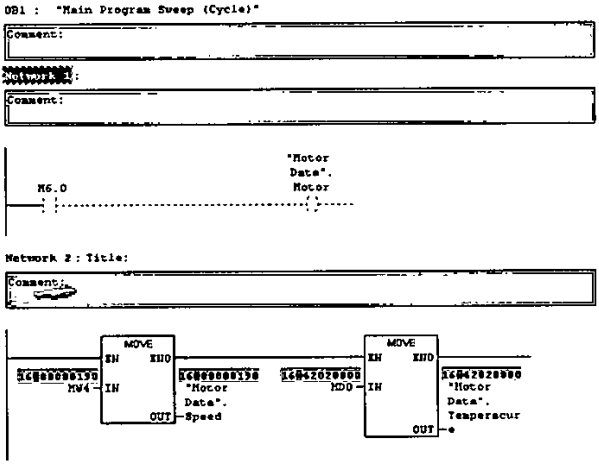
برای این کار می‌توان در شرایطی که دیتابلاک بسته است، با کلیک راست روی آن، گزینه Object Properties را انتخاب نمود. در اینصورت کادری مانند شکل۳-۲۵ نمایان می‌شود که در قسمت Symbolic Name می‌توان نام مورد نظر را اختصاص داد. به‌عنوان نمونه برای مثال ۳-۱ ما نام سمبولیک Motor Data را وارد می‌کنیم.

1. **Symbolic**



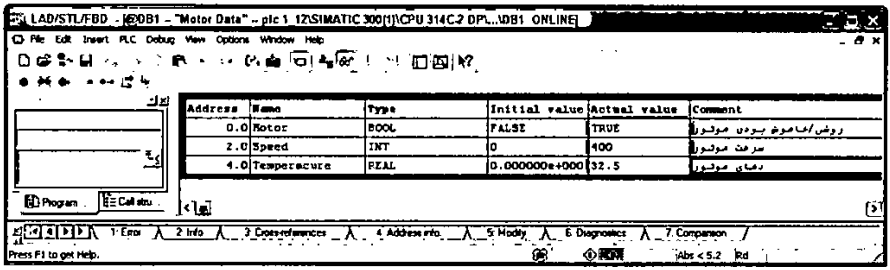
شکل ۳-۲۵ تعیین نام سمبولیک برای DB1

در اینصورت برنامه نوشته شده در OB1 به‌صورت زیر خواهد بود. همانطور که دیده می‌شود اسامی متغیرها به‌طور کامل سمبلیک است که در آن ابتدا نام دیتابلاک، سپس نام متغیر دیتابلاک که با نقطه از نام دیتابلاک جدا شده، ظاهر می‌شود.



شکل ۳-۲۶ برنامه نوشته شده در OB1 با نمایش سمبولیک آدرس‌های DB1

حال اگر به DB1 رفته و View را روی حالت Data View تنظیم نماییم، به‌شرط فعال نمودن ابزار Monitor (عینک) مقادیر واقعی متغیرها را می‌توانیم در ستون Actual Value مشاهده نماییم. به شکل۳-۲۷ توجه کنید.



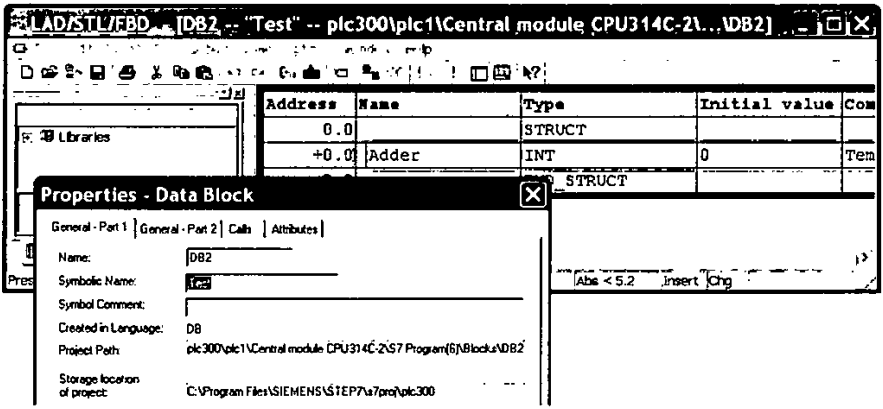
شکل ۳-۲۷ محیط DB1 در حالت Online

البته باید توجه نمود که در هر دو حالت آدرس‌دهی مطلق یا سمبولیک محتویات دیتابلاک به‌یک صورت بوده و نمایش متغیرها در خود دیتابلاک تغییر نخواهد کرد. همچنین در هر دو حالت می‌توان از طریق ابزار Monitor در حالت Data View مقادیر واقعی متغیرها را مشاهده نمود.

فصل ۳

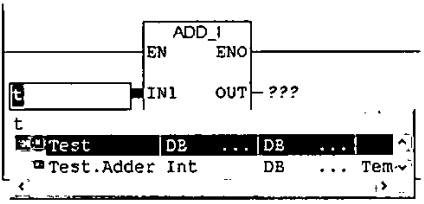
مثال ۳-۲

DB2 با نام سمبلیک Test ایجاد شده و سطر اول آن که از جنس INT است به Adder تغییر نام یافته است.



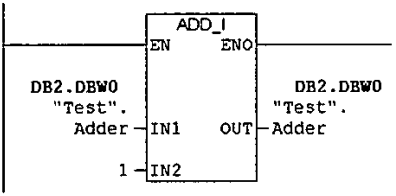
شکل ۳-۲۸ تعریف DB برای مثال ۳-۲

در این حالت در برنامه ساده زیر وقتی کلمه Test را در پایه ورودی یا خروجی ADD_I تایپ می‌کنیم، کادری باز می‌شود که سطرهای DB را در زیر مجموعه‌اش نشان می‌دهد و می‌توان نام مورد نظر را مانند شکل ۳-۲۹ انتخاب نمود.



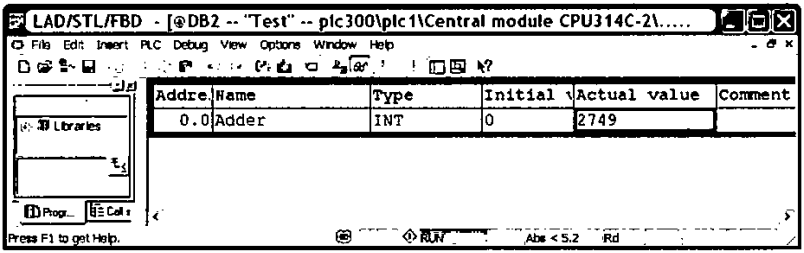
شکل ۳-۲۹ انتخاب متغیر از DB مربوط به مثال ۳-۲

برنامه کامل شده به‌صورت زیر است. اگر از منوی View در قسمت Display with گزینه‌های نمایش را فعال کنیم، برنامه را با آدرس‌های مطلق و سمبلیک خواهیم دید.



شکل ۳-۳۰ برنامه مثال ۳-۲

پس از دانلود این برنامه اگر DB2 را مانیتور کنیم، می‌بینیم که مقدار سطر Adder مدام در حال تغییر است زیرا برنامه در هر سیکل مقدار قبلی را از DB می‌خواند، آنرا با یک جمع می‌کند و نتیجه را در DB می‌نویسد.

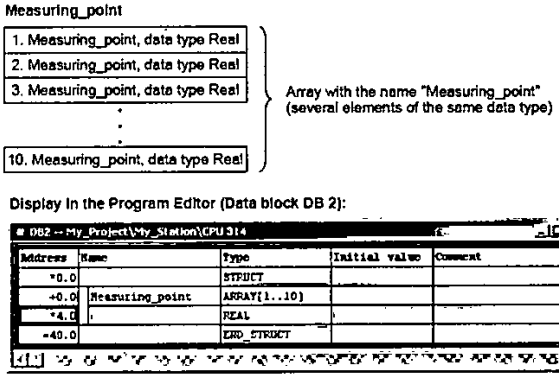


شکل ۳-۳۱ دیتابلاک در وضعیت Online مربوط به مثال ۳-۲

۳-۱۰ استفاده از آرایه در دیتابلاک

آرایه از چندین عنصر با تایپ یکسان تشکیل می‌شود. فرض کنید در یک پروژه صنعتی 10 موتور وجود دارد که دمای هرکدام از آنها به فرم Real اندازه‌گیری شده و باید در یک دیتابلاک ذخیره شود. برای این موضوع باید 10 متغیر از نوع

Real با نام‌های Measuring Point 1 تا Measuring Point 10 در دیتابلاک مورد نظر ایجاد شوند. به‌جای اینکه تک‌تک این متغیرها را جداگانه ایجاد نماییم، می‌توان یک آرایه با ده عضو ایجاد کنیم. شکل ۳-۳۲ چگونگی این عمل را نشان می‌دهد.



شکل ۳-۳۲ ایجاد آرایه در DB2

همانطور که در این شکل دیده می‌شود، در ستون Type بایستی تعداد عنصر آرایه را مشخص کرد سپس در زیر آن نوع متغیرها را تعیین نمود، که در این مثال نوع متغیر Real انتخاب شده است. با توجه به اینکه هر عنصر Real چهار بایت فضا اشغال می‌کند و آرایه ده عنصر دارد، بنابراین همانطور که دیده می‌شود، حجم دیتابلاک ۴۰ بایت است.

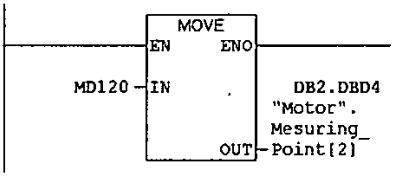
در مثال فوق اگر View را روی حالت Data View قرار دهیم شکلی مانند شکل ۳-۳۳ نشان داده می‌شود.

DB2 -- My_Project\My_Station\CPU 314

Address	Name	Type	Initial value	Actual value	Comment
0.0	Measuring_point[1]	REAL	0.000000e+000	1.000000e+002	
4.0	Measuring_point[2]	REAL	0.000000e+000	1.023000e+002	
8.0	Measuring_point[3]	REAL	0.000000e+000	1.035000e+002	
12.0	Measuring_point[4]	REAL	0.000000e+000	1.067000e+002	
16.0	Measuring_point[5]	REAL	0.000000e+000	1.052000e+002	
20.0	Measuring_point[6]	REAL	0.000000e+000	1.050000e+002	
24.0	Measuring_point[7]	REAL	0.000000e+000	1.055000e+002	
28.0	Measuring_point[8]	REAL	0.000000e+000	1.055000e+002	
32.0	Measuring_point[9]	REAL	0.000000e+000	1.079000e+002	
36.0	Measuring_point[10]	REAL	0.000000e+000	1.079900e+002	

شکل ۳-۳۳ نمایش DB2 در حالت Data View

همانطور که در این شکل دیده می‌شود، هر عنصر آرایه دارای شماره ایندکس منحصر به فرد بین ۱ تا ۱۰ است. وقتی در برنامه نویسی از آدرس آرایه استفاده می‌کنیم، لازم است به این شماره اشاره نماییم. شکل ۳-۳۴ نحوه انتقال دیتا به عضو دوم از آرایه فوق را نشان می‌دهد.



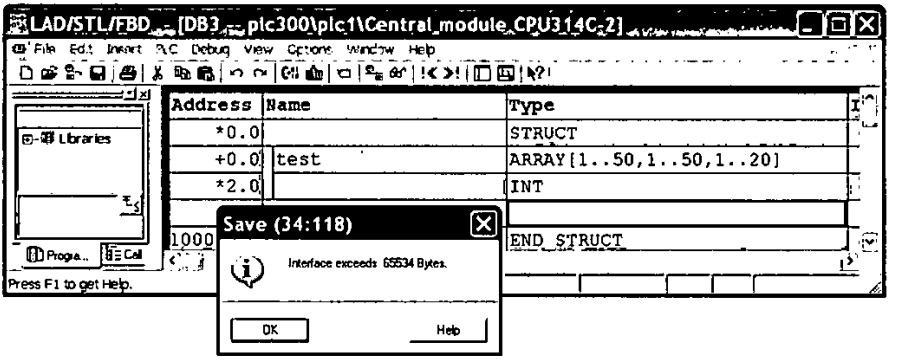
شکل ۳-۳۴ استفاده از آدرس آرایه

نکات قابل توجه

- آرایه می‌تواند مانند مثال‌های زیر چند بعدی باشد.

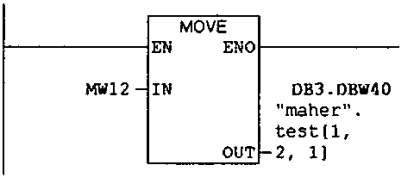
Test	Array[1..100]	۱۰۰ عنصر	آرایه یک بعدی
Test	Array[1..100 ,1..200]	۲۰۰۰۰ عنصر	آرایه دو بعدی
Test	Array[1..100 , 1..200 , 1..5]	۱۰۰۰۰۰عنصر	آرایه سه بعدی

- ماکزیمم می‌توان آرایه ۵ بعدی تعریف کرد.
- ماکزیمم تعداد عناصر آرایه با توجه به نوع آنها می‌تواند تا حدی باشد که تعداد بایت‌های دیتا بلاک از حد تعیین شده فراتر نرود. شکل ۳-۳۵ پیغام خطای ناشی از زیاد بودن تعداد بایت تعریف شده توسط یک آرایه سه بعدی که از 64 kb فراتر رفته را در هنگام ذخیره‌کردن DB نشان می‌دهد.



شکل ۳-۳۵ افزایش تعداد بایت DB از حد ماکزیمم

در برنامه‌نویسی اشاره به آرایه‌های چند بعدی مانند مثال زیر است. در این مثال که آدرس‌دهی عنصر [1,2,1] Test را از آرایه سه بعدی نشان می‌دهد.

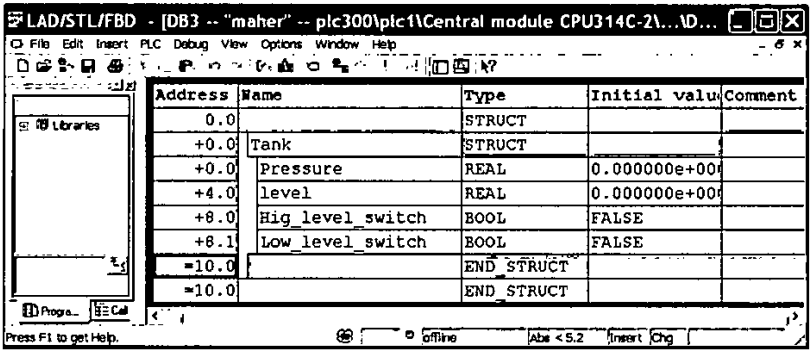


شکل ۳-۳۶ آدرس‌دهی آرایه دو بعدی در برنامه

- همه عناصر آرایه فقط می‌توانند از یک جنس تعریف شوند. در صورت نیاز به تعریف متغیرهای مختلف تحت عنوان یک مجموعه بایستی از Structre استفاده نمود.

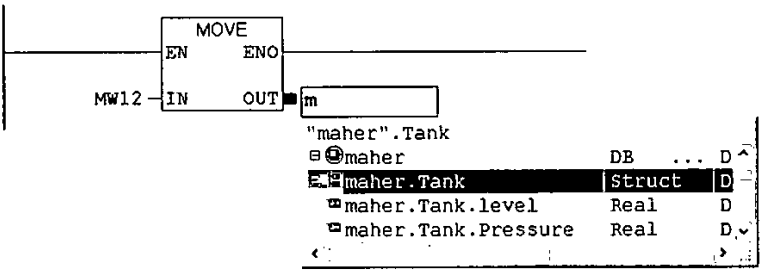
۳-۱۱ استفاده از Structure در دیتابلاک

در مواقعی که چندین متغیر با دیتاتایپ‌های مختلف وجود داشته باشد، می‌توان مجموعه آنها را به‌صورت یک Structure تعریف نمود. به‌عنوان مثال اگر یک مخزن دارای سیگنال‌های مختلفی از ترانسمیترها و سوئیچ‌های مختلف باشد می‌توانیم مجموعه آنها را تحت عنوان یک استراکچر تعریف کنیم. مانند شکل ۳-۳۷ ابتدا Structure را با نام Tank تعریف کرده سپس متغیرهای آنرا تعریف می‌نماییم.



شکل ۳-۳۷ متغیر از نوع Struct در DB

در این‌صورت در برنامه‌نویسی در هنگام اختصاص آدرس، پنجره‌ای مانند شکل ۳-۳۸ باز می‌شود که ساختار را نشان می‌دهد. همانطور که در این شکل دیده می‌شود بین اسامی علامت نقطه به‌کار رفته است.

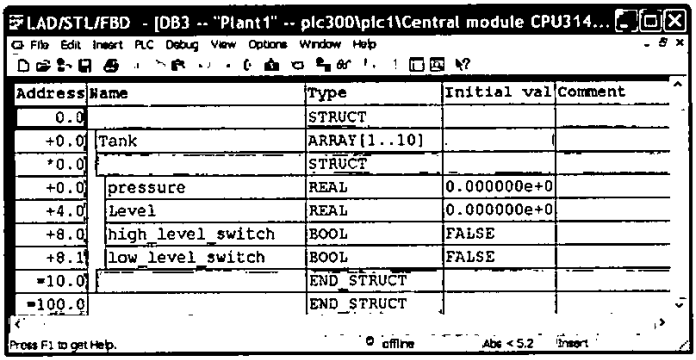


شکل ۳- ۳۸ آدرس‌دهی Struct در برنامه

ترکیب Array با Struct

اگر تعداد زیادی سیستم که هر کدام دیتاهای مختلفی که هم نوع نیستند وجود داشته باشد، می‌توان Array را با Struct به‌کار گرفت. به‌عنوان مثال اگر ده مخزن با سیگنال‌های ذکر شده قبل وجود داشته باشد. می‌توان مخازن را به‌صورت Array تعریف کرد ولی نوع عناصر Array را Struct انتخاب نمود.

شکل ۳-۳۹ تعریف ده مخزن به‌صورت Array که همگی دارای ساختار Struct یکسان هستند را نشان می‌دهد.



شکل ۳-۳۹ ترکیب آرایه و Struct در DB

همانطور که در این شکل دیده می‌شود، Struct به اندازه ۱۰ بایت فضا اشغال کرده است، پس ۱۰ عنصر آرایه جمعاً ۱۰۰ بایت فضا خواهند داشت.

اگر DB فوق را در حالت Data View درآوریم شکل ۳-۴۰ را خواهیم دید.

LAD/STL/FBD - [DB3 -- "Plant1" -- plc300\plc1\Central module CPU314C-2\...\DB3]

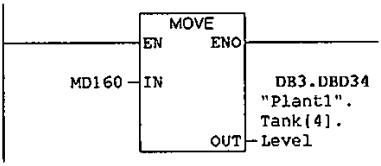
File Edit Insert PLC Debug View Options Window Help

Address	Name	Type	Initial value	Actual value	Comm
0.0	Tank[1].pressure	REAL	0.000000e+000	0.000000e+000	
4.0	Tank[1].Level	REAL	0.000000e+000	0.000000e+000	
8.0	Tank[1].high_level_switch	BOOL	FALSE	FALSE	
8.1	Tank[1].low_level_switch	BOOL	FALSE	FALSE	
10.0	Tank[2].pressure	REAL	0.000000e+000	0.000000e+000	
14.0	Tank[2].Level	REAL	0.000000e+000	0.000000e+000	
18.0	Tank[2].high_level_switch	BOOL	FALSE	FALSE	
18.1	Tank[2].low_level_switch	BOOL	FALSE	FALSE	
20.0	Tank[3].pressure	REAL	0.000000e+000	0.000000e+000	

Press F1 to get Help. Offline Abs < 5.2 Insert

شکل ۳-۴۰ ترکیب آرایه و Structure در حالت Data View

در این حالت در برنامه آدرس‌دهی مانند مثال زیر خواهد بود.



شکل ۳-۴۱ استفاده از آدرس ترکیبی آرایه و Struct

فصل ۳

۳-۱۲ استفاده از UDT

UDT که مخفف User Data Type است، ابزاری مشابه Struct است که توسط آن می‌توان دیتاهای مختلف را به‌عنوان یک مجموعه تعریف کرده، سپس آنها را در دیتا بلاک فراخوان نمود. پس به‌طور خلاصه:

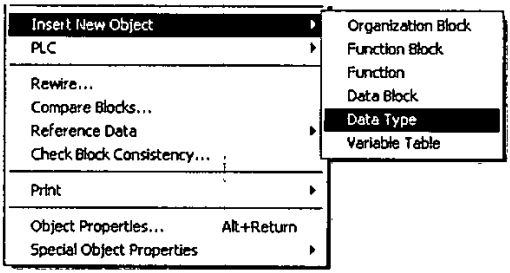
- UDT در بیرون دیتا بلاک تعریف می‌شود و می‌توان از آن در دیتا بلاک‌های مختلف استفاده نمود در حالی که Struct در داخل DB تعریف می‌شود و فقط همانجا کاربرد دارد.
- UDT علاوه بر دیتا بلاک در پارامترهای محلی بلاک‌های برنامه‌نویسی نیز قابل استفاده است. این موضوع در فصل بعد مورد بحث قرار می‌گیرد.
- UDT همانند VAT ابزاری برای سهولت تعریف دیتاست و قابل دانلود به PLC نیست. بنابراین پس از آپلود از PLC آن را در پوشه بلاک نخواهیم دید.

برای استفاده از UDT مراحل زیر دنبال شود:

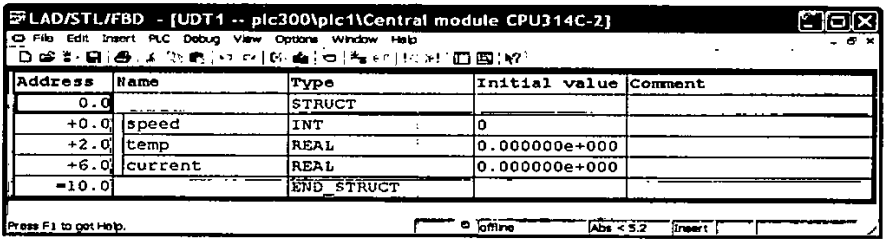
۱- ایجاد UDT جدید در محیط Simatic Manager با کلیک راست و انتخاب گزینه Data Type (شکل ۳-۴۲). بار اول به‌طور خودکار UDT1 ساخته می‌شود.

۲- ورود به محیط UDT1 و تعریف متغیرهای لازم (شکل ۳-۴۳)

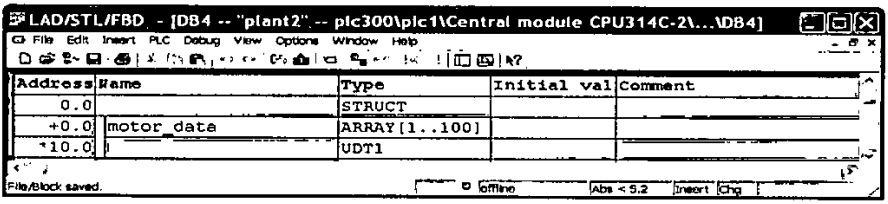
۳- ورود به دیتابلاک و استفاده از UDT (شکل ۳-۴۴)



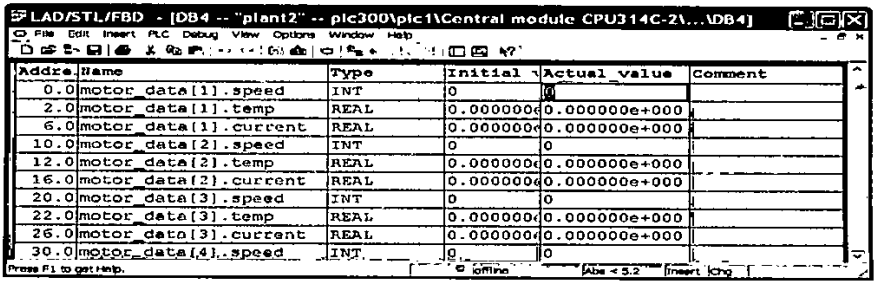
شکل ۳-۴۲ ایجاد UDT1



شکل ۳-۴۳ تعریف متغیرها در UDT1

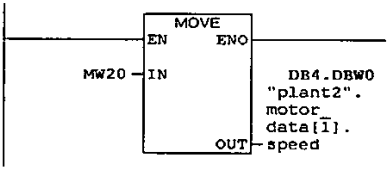


شکل ۳-۴۴ استفاده از UDT1 در DB4 در نمایش Declaration View



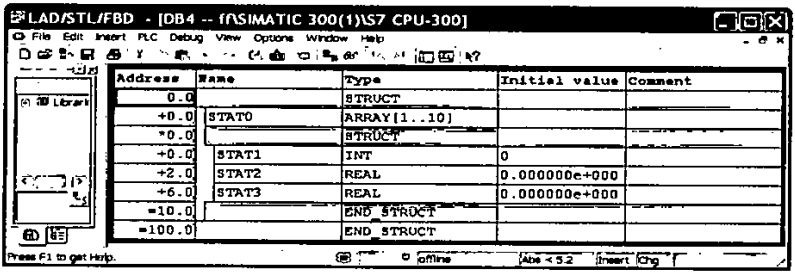
شکل ۳-۴۵ DB4 در حالت Data View

آدرس دیتای تعریف شده توسط UDT در برنامه مانند مثال زیر مشابه آدرس Struct است.



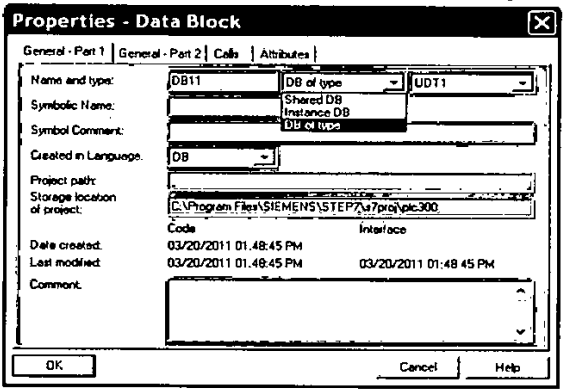
شکل ۳-۴۶ استفاده از آدرس UDT در برنامه

اگر دیتا بلاکی که در آن از UDT استفاده شده را از PLC آپلود کنیم، آنچه به پروژه وارد می‌شود ساختار Struct مانند شکل ۳-۴۷ دارد و UDT را در آن نمی‌بینیم. شکل مربوط ۳-۴۷ به مثال فوق است.



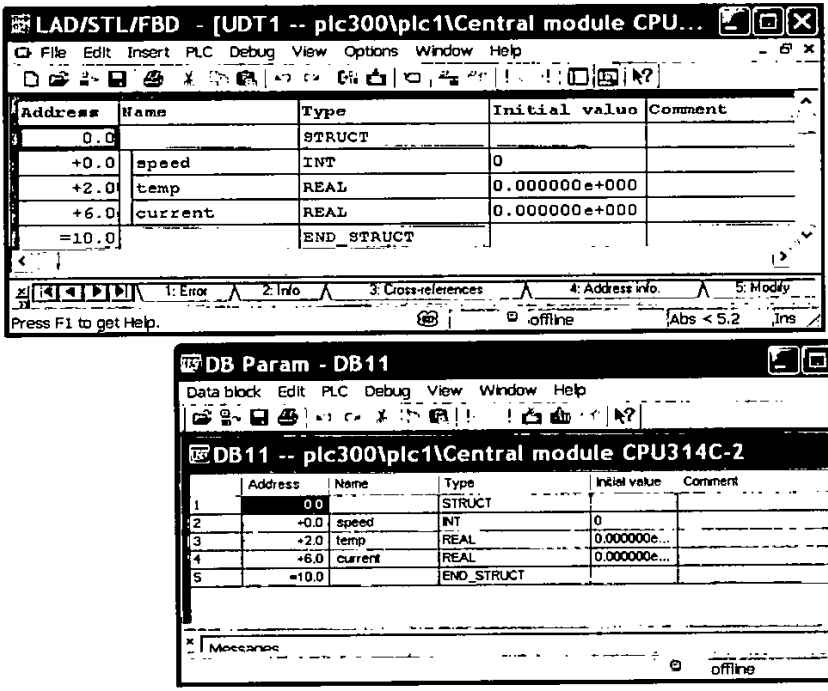
شکل ۳-۴۷ آپلود DB که با UDT ایجاد شده

نکته: اگر قبلاً یک UDT با ساختار مشخصی تعریف شده باشد و دیتا بلاک جدیدی در پوشه بلاک ساخته شود، در پنجره مانند شکل ۳-۴۸ گزینه دیگری با عنوان DB of type ظاهر خواهد شد. در صورت انتخاب این گزینه DB به‌طور خودکار طبق ساختار UDT ساخته خواهد شد.



شکل ۳-۴۸ اختصاص UDT به DB

مثال شکل ۳-۴۹ ساختار UDT و DB ساخته شده به روش فوق را نشان می‌دهد. دقت کنید که DB ایجاد شده قابل تغییر نیست و اگر لازم است سطر جدید اضافه شود بایستی این کار در UDT انجام گرفته و DB از نو ساخته شود تا تغییرات را بگیرد.



شکل ۳-۴۹ DB ساخته شده بر اساس UDT

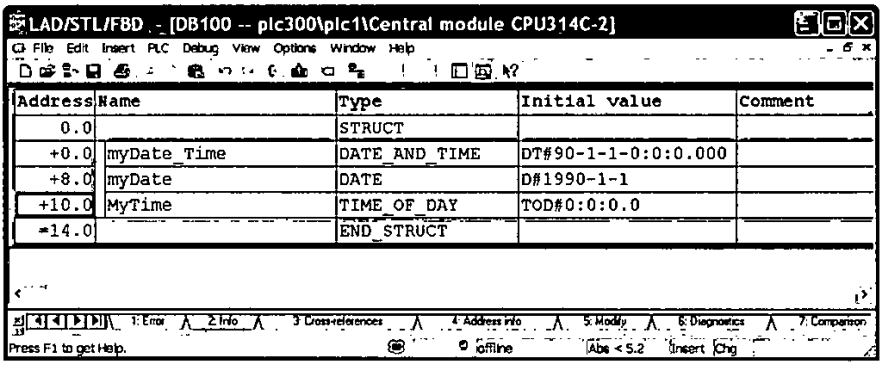
۳-۱۳ استفاده از Date_And_Time در DB

متغیری که از نوع Date_And_Time باشد، تاریخ و زمان را به‌صورت ترکیبی نشان می‌دهد. به‌عنوان مثال ممکن است مقدار این متغیر به‌صورت زیر باشد که بخش اول معرف تاریخ و بخش دوم معرف زمان است.

DT#98-1-1-17:20:11.345

زمان در واقع به تنهایی متغیری از نوع TOD است که نبایستی با Time اشتباه گرفته شود. توضیحات کافی در این زمینه در کتاب سطح مقدماتی آورده شده است.

متغیرهای Date_And_Time فضایی بیش از ۴ بایت اشغال می‌کند. به شکل ۳-۵۰ دقت کنید. همانطور که در این شکل دیده می‌شود، متغیر Date_And_Time به اندازه ۸ بایت فضا اشغال کرده است زیرا Date به تنهایی ۲ بایت و Time به تنهایی ۴ بایت فضا نیاز دارد.



شکل ۳-۵۰ استفاده از Date And Time در DB

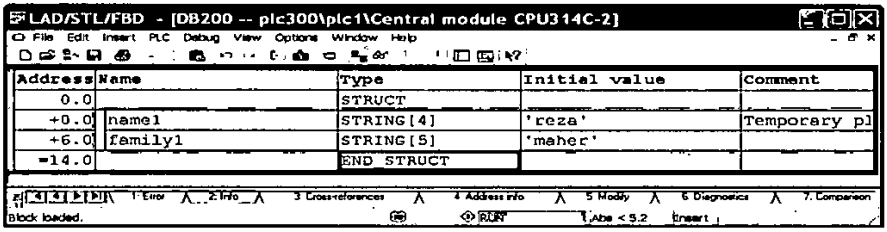
فصل ۳

نکاتی که در ارتباط با متغیرDate_And_Time می‌توان ذکر کرد:

- این متغیر را می‌توان در DB تعریف کرد ولی خواندن یا نوشتن در آن نمی‌تواند توسط دستورات معمولی مانند Move انجام شود. برای این منظور فانکشن‌های خاصی به نام فانکشن‌های IEC به‌کار می‌رود که در کتاب سطح تکمیلی تشریح خواهند شد.
- این متغیر را نمی‌توان در دیتا بلاک مانیتور نمود و با رنگ خاکستری ظاهر می‌شود.

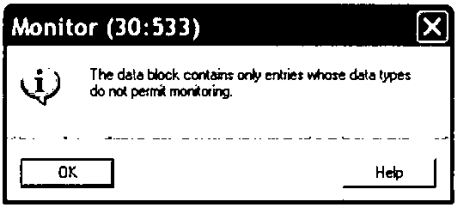
۳-۱۴ استفاده از String در DB

یکی دیگر از انواع متغیرهای Complex که در دیتابلاک قابل تعریف است استرینگ می‌باشد. استرینگ به‌صورت رشته‌ای از کاراکترهاست. شکل ۳-۵۱ نمونه ای از تعریف استرینگ در دیتا بلاک را نشان می‌دهد.



شکل ۳-۵۱ استفاده از String در DB

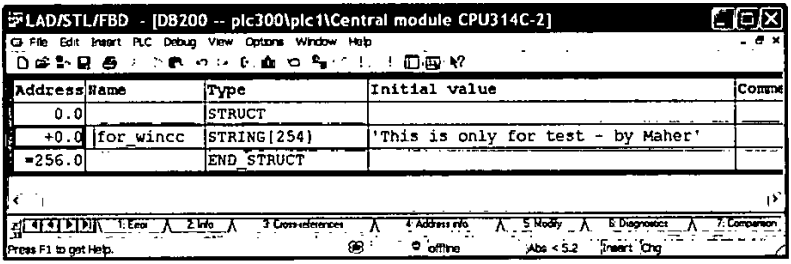
یک استرینگ می‌تواند ماکزیمم ۲۵۴ کاراکتر باشد. می‌توان در استرینگ پیغام دلخواهی را برای چاپ روی پرینتر یا ارسال به سیستم مانیتورینگ ذخیره نمود. ولی این کاراکترها در محیط دیتا بلاک قابل مانیتور نیستند و در صورت فعال سازی مانیتور با پیغام خطای زیر مواجه می‌شویم.



شکل ۳-۵۲ خطای مانیتور کردن استرینگ در DB

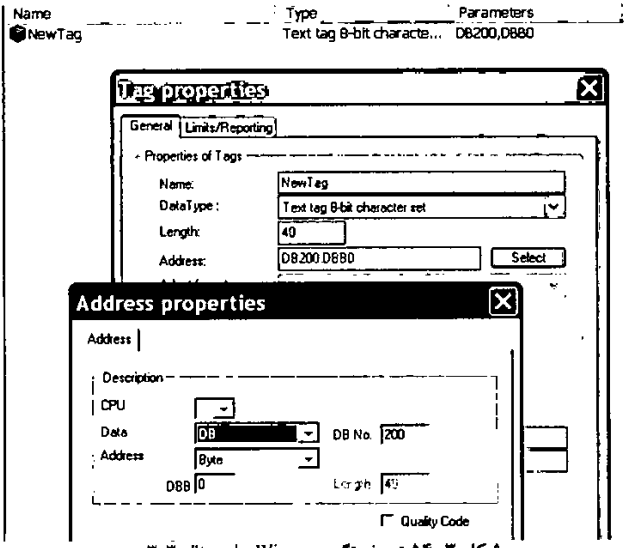
مثال ۳-۳

در این مثال پیغامی که به‌صورت استرینگ در DB200 مانند شکل ۳-۵۳ تعریف شده است در winCC نمایش داده می‌شود. خوانندگانی که با WinCC آشنایی دارند می‌توانند آنرا با سیمولاتور یا PLC تست کنند.



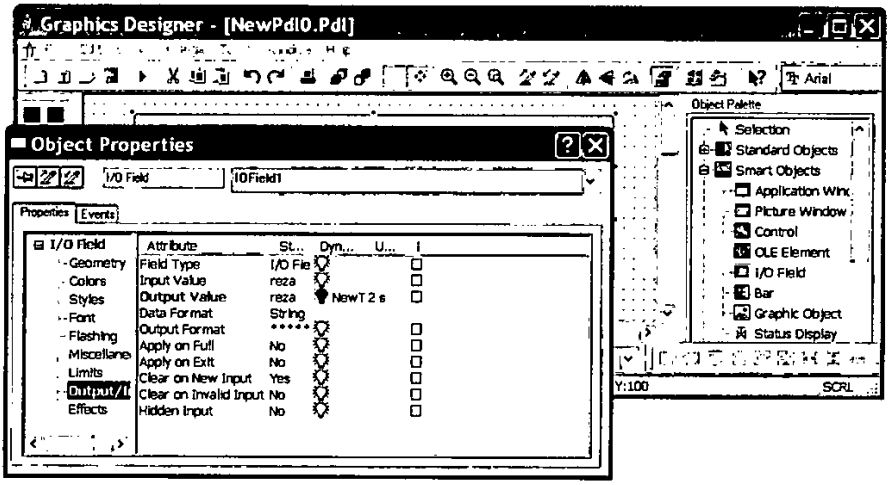
شکل ۳-۵۳ DB مربوط به مثال ۳-۳

پس از ساختن DB200 به‌صورت فوق آنرا دانلود کنید. در محیط Wincc یک Tag مانند شکل۳-۵۴ تعریف نمایید.



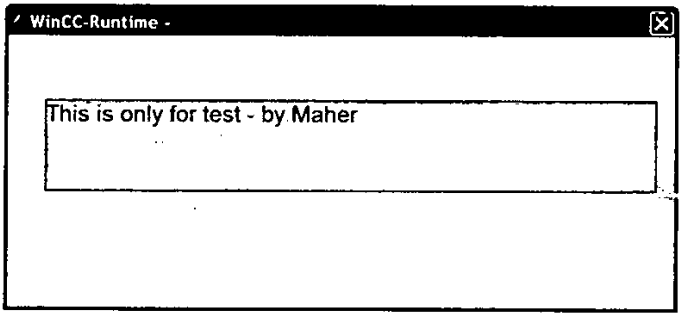
شکل ۳-۵۴ تعریف تگ در Wincc برای مثال ۳-۳

در صفحه گرافیک با استفاده از I/O Field تگ فوق را متصل نموده و نوع نمایش را به‌صورت String انتخاب کنید



شکل ۳-۵۵ تنظیمات گرافیک winCC در مثال ۳-۳

پس از فعال‌سازی WinCC متن پیغام مانند شکل ۳-۵۶ در محیط Runtime نمایش داده خواهد شد.



شکل ۳-۵۶ wincc runtime برای مثال ۳-۳

نکات قابل توجه

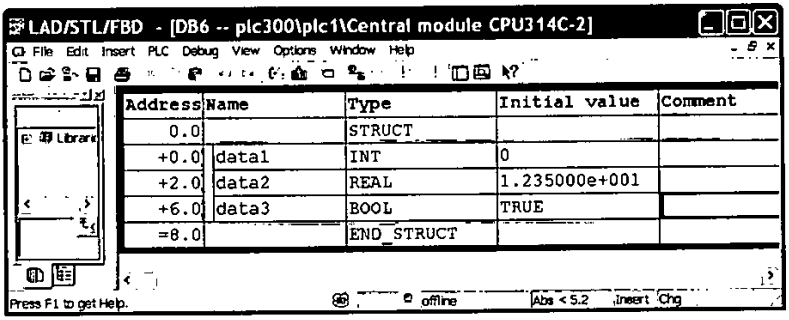
- در کار با استرینگ معمولاً به فانکشن‌های خاصی که در زیرمجموعه IEC Function در کتابخانه نرم‌افزار قرار دارند نیاز خواهید داشت.
- فانکشن‌های IEC می‌توانند دو استرینگ را با هم ترکیب کنند یا بخشی از استرینگ را جدا کنند.
- تشریح فانکشن‌های IEC در کتاب سطح تکمیلی آورده شده است.

۳-۱۵ اختصاص مقدار اولیه به Data Block

در هر دو نوع دیتابلاک اختصاصی و اشتراکی، ستونی با عنوان Initial Value وجود دارد که برای اختصاص مقدار اولیه به متغیر به‌کار می‌رود. منظور از مقدار اولیه این است که اگر این متغیر در برنامه خوانده شود، مقداری را برگرداند بدون اینکه در جایی از برنامه این مقدار در دیتا بلاک نوشته شده باشد.

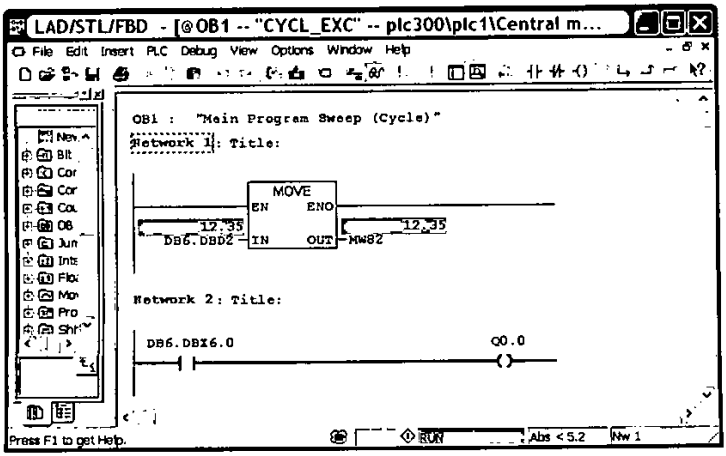
در Instance DB مقدار اولیه از تنظیمات سمت FB یا SFB به‌طور خودکار وارد می‌شود و قابل تغییر نیست. ولی در Shared DB کاربر می‌تواند با توجه به نکاتی که در ادامه تشریح می‌شود به متغیرهای دیتابلاک مقدار اولیه اختصاص دهد.

دیتا بلاک اشتراکی مثال زیر را در نظر بگیرید. برای اختصاص مقدار اولیه در همان زمان که نام سطر را تعریف می‌کنیم بلافاصله در جلوی آن مقدار اولیه را وارد می‌نماییم. در شکل ۳-۵۷ برای سطر دوم مقدار اولیه 12.35 و برای سطر سوم مقدار اولیه True را وارد می‌کنیم.



شکل ۳-۵۷ اختصاص مقدار اولیه به DB

پس از دانلود کردن دیتابلاک، توسط برنامه زیر مقادیر سطرهای دوم و سوم را می‌خوانیم. مشاهده می‌شود که مقدار اولیه به برنامه وارد شده است.



شکل ۳-۵۸ خواندن مقادیر اولیه DB در برنامه

با استفاده از این روش می‌توان مقادیر مبنای مورد نیاز را در دیتابلاک وارد نموده و در ضمن آنرا به روشی که در ادامه تشریح می‌گردد Protect نمود تا فقط برای خواندن استفاده شود.

ولی نکته‌ای که در اختصاص مقدار اولیه وجود دارد این است که اگر پس از یک‌بار تعریف مقدار اولیه آنرا تغییر دهیم، در برنامه مقدار جدید وارد نمی‌شود و همان مقدار قبلی را می‌خواند. پس در این روش نباید پس از تعریف، مقدار اولیه عوض شود. برای تغییر لازم است دیتابلاک قبلی حذف شده و از نو ساخته شود.

در عمل برای رفع مشکل فوق و اینکه به‌سهولت امکان تغییر مقادیر اولیه در دیتا بلاک وجود داشته باشد به روش‌های دیگری دیتابلاک اشتراکی را ایجاد می‌کنند. این روش‌ها عبارتند از:

- ایجاد دیتا بلاک به روش STL Source
- ایجاد دیتا بلاک به روش SCL Source

در این دو روش دیتا بلاک به‌صورت Source File مانند شکل زیر ایجاد شده و پس از کامپایل دیتابلاک در پوشه Blocks ساخته می‌شود و متغیرها با مقادیر اولیه جدید تنظیم می‌شوند.

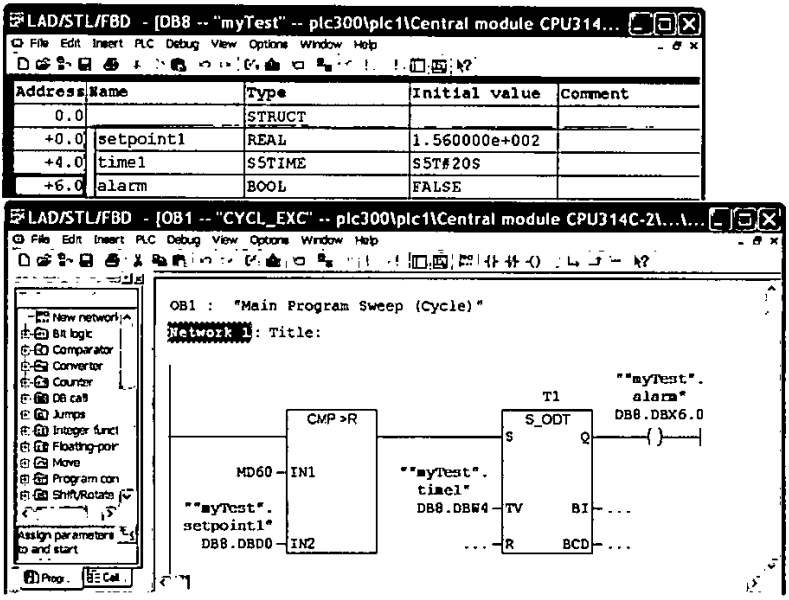
فصل ۳

```
DATA_BLOCK DB 6

  STRUCT
   data1 : INT ; :=100
   data2 : REAL  := 4.001000e+002;
   data3 : BOOL ;:= false;
  END_STRUCT;
BEGIN
   data1 := 0 ;
   data2 := 1.235000e+001 ;
   data3 := TRUE ;
END_DATA_BLOCK
```

تشریح روش Source File در کتاب سطح تکمیلی آورده شده است

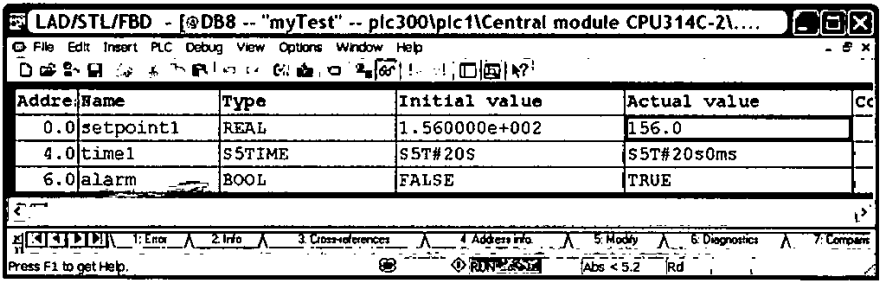
مثال ۳-۴

در برنامه زیر زمان تایمر و مقدار مبنا در دیتابلاک تعریف شده و دارای مقدار اولیه هستند. در صورتی‌که مقدار MD60 از مقدار مبنا بیشتر شد تایمر با زمان تعیین شده به‌کار می‌افتد و آلارم تولید می‌کند. این آلارم در دیتا بلاک ذخیره می‌شود.



شکل ۳-۵۹ برنامه مثال ۳-۴

دیتابلاک فوق در حالت Online پس از تولید آلارم به‌صورت شکل ۳-۶۰ خواهد بود.

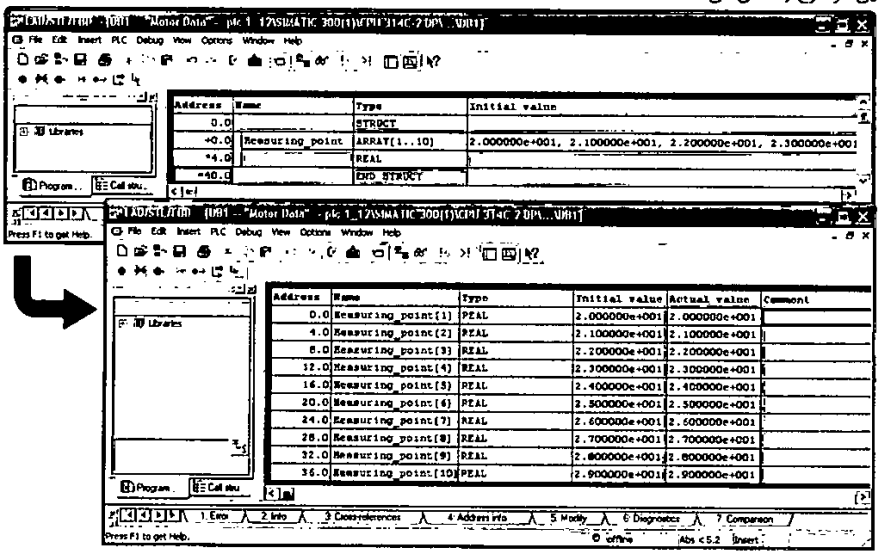


شکل ۳-۶۰ DB مثال ۳-۴ در حالت Online

تنظیم Initial Value برای آرایه

سوالی که مطرح می‌شود این است که اگر بخواهیم برای همه عضوهای یک آرایه مقدار اولیه اختصاص دهیم چه باید کرد؟ به‌عنوان مثال می‌خواهیم به‌ترتیب مقدار ۲۰ تا ۲۹ را برای متغیرهای شماره ۱ تا ۱۰ یک آرایه ۱۰ عضوی وارد کنیم. اگر در ستون Initial Value عددی را وارد کنیم، این عدد به‌عنوان مقدار اولیه متغیر شماره ۱ در نظر گرفته می‌شود. برای سایر

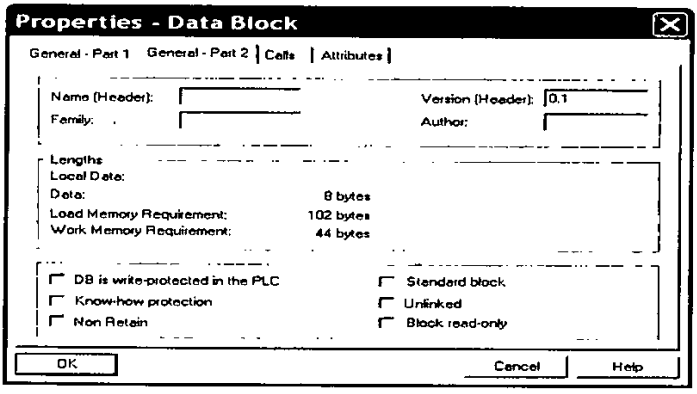
متغیرها کافیست تا پس از مقدار اولیه وارد شده علامت (,) را به‌کار برده و مقدار اولیه بعدی را وارد نماییم. شکل ۳-۶۱ این موضوع را نشان می‌دهد.



شکل ۳-۶۱ اختصاص مقدار اولیه به آرایه

۳-۱۶ بررسی ویژگی‌های دیتابلاک

در Properties دیتابلاک سربرگ‌هایی مانند شکل ۳-۶۲ می‌بینیم که ویژگی‌های DB را از طریق آن می‌توان تغییر داد.



شکل ۳-۶۲ سربرگ‌های Properties در DB

در سربرگ General-Part2 همانطور که در شکل ۳-۶۲ مشاهده می‌شود، بخش‌های مختلفی مشاهده می‌شود. این بخش‌ها عبارتند از:

در قسمت بالا می‌توان مشخصاتی از تهیه کننده دیتابلاک، ورژن و ... را برای اطلاعات بیشتر وارد کرد. این اطلاعات بعداً در نمایش Detail در پوشه بلاک قابل مشاهده است.

در قسمت وسط، اطلاعاتی در مورد تعداد بایت DB و فضایی که در Load Memory و Work Memory مورد نیاز است آورده شده است.

در قسمت پایین، چند گزینه وجود دارد که به‌صورت پیش‌فرض غیر فعال هستند ولی در صورت لزوم می‌توان از آنها استفاده نمود این گزینه‌ها عبارتند از:

DB is Write Protected in the PLC: با فعال‌سازی این گزینه DB در مقابل نوشتن حفاظت می‌شود.

Non Retain: با فعال‌سازی این گزینه دیتاها در DB ماندگار نخواهند بود و با قطع و وصل تغذیه پاک می‌شوند.

Unlinked: با فعال‌سازی این گزینه دیتا بلاک در Load Memory باقی مانده و به Work Memory لود نمی‌شود.

تشریح موارد فوق در ادامه آمده است.

Protect کردن DB در برابر نوشتن در آن

می‌توان یک DB را به‌نحوی محافظت نمود که برنامه کاربر نتواند چیزی در آن بنویسد. در مواردی ممکن است بخواهیم برخی از Setpointهایی که لازم است بدون تغییر باشند را در یک DB ایجاد نماییم. در این شرایط ممکن است کاربر سهواً یا عمداً در ضمن برنامه خود مقداری را به آن دیتابلاک منتقل نماید و مقدار جدید روی مقادیر قبلی متغیرها قرار گیرد. برای رفع این مشکل می‌توان دیتابلاک را دربرابر نوشتن محافظت نمود. اگر در دیتابلاکی که در برابر نوشتن محافظت شده است مقداری قرار داده شود، CPU دچار فالت شده و ممکن است متوقف شود.

برای Protect کردن یک دیتابلاک لازم است مراحل زیر اجرا شود:

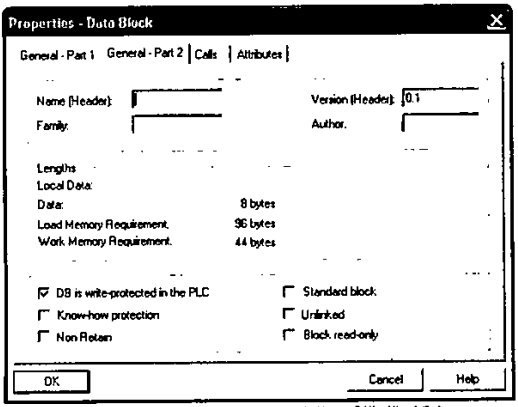
۱- ابتدا اگر محیط دیتابلاک باز است، آنرا ببندید.

۲- روی دیتابلاک کلیک راست نموده و گزینه Object Properties را انتخاب نمایید.

۳- در سربرگ General - Part 2 علامت مربوط به گزینه DB is write-protected in the PLC را فعال نمایید.

۴- روی گزینه OK کلیک نمایید تا پنجره بسته شود.

۵- DB را دانلود نمایید.



شکل ۳-۶۳ محافظت نمودن دیتابلاک در برابر نوشتن

از Protect خارج نمودن DB

برای اینکه یک دیتابلاک را از حالت محافظت در برابر نوشتن خارج نماییم می‌توان مراحل بالا را طی نموده و علامت مربوط به گزینه DB is write-protected in the PLC را غیرفعال نمود؛ سپس تغییرات بوجود آمده را دانلود نمود.

نکته: مواردی که استفاده از DB می‌تواند منجر به توقف CPU گردد:

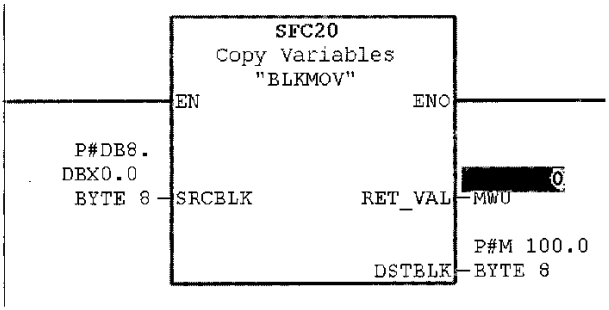
- آدرسی از متغیر موجود در دیتابلاکی فراخوانی شود که آن دیتابلاک قبلاً دانلود نشده باشد.
- استفاده از آدرسی از دیتابلاک که عملاً آن آدرس در دیتابلاک وجود ندارد.
- نوشتن در دیتابلاکی که در برابر نوشتن محافظت شده است.

Unlink کردن DB

اگر گزینه Unlink که در شکل ۳-۶۲ نشان داده شد فعال و به PLC دانلود شود. دیتا بلاک به Load Memory وارد می‌شود ولی در صورتی‌که از آدرس‌های DB در برنامه استفاده گردد منجر به توقف CPU می‌گردد زیرا فعال‌سازی گزینه فوق موجب می‌شود که دیتا بلاک به Work Memory وارد نشود.

در این شرایط فقط می‌توان با SFCهای خاصی از DB استفاده کرد. SFC20 یکی از این موارد است که در برنامه زیر نشان داده شده است. با استفاده از آن آدرس‌های مورد نظر از DB به Work Memory منتقل شده و در حافظه دلخواه برای استفاده قرار می‌گیرد.

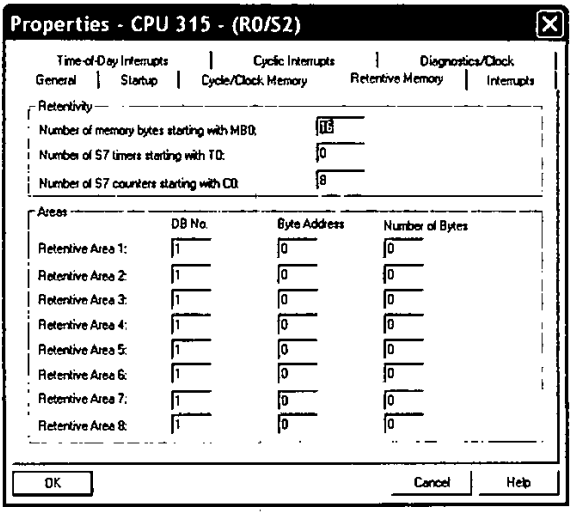
تشریح این SFC در کتاب سطح تکمیلی آمده است.



شکل ۳-۶۴ استفاده از SFC20

Non Retain کردن DB

در برخی CPUها در بخش Retentive Memory می‌توان علاوه بر حافظه Timer و Counter و Memory، دیتا بلاک را نیز به‌عنوان ماندگار تعریف کرد. شکل ۳-۶۵ این موضوع را برای CPU315 نشان می‌دهد.



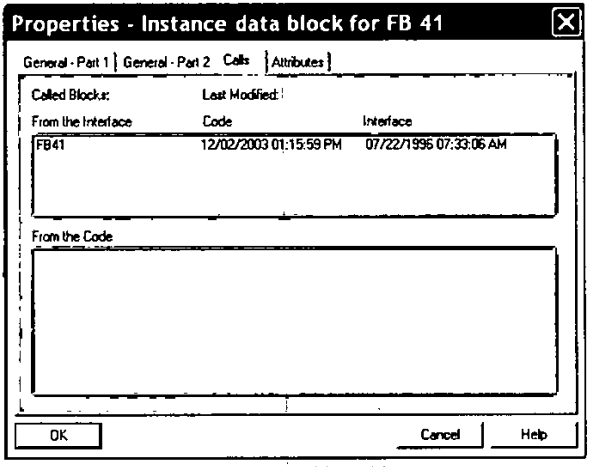
شکل ۳-۶۵ تنظیمات ماندگاری DB در CPU315

DBها و آدرس‌های تعیین شده در آنها با قطع تغذیه پاک نمی‌شوند.

اگر در پنجره ویژگی‌های دیتا بلاک، گزینه Non Retain را فعال کنیم DBهای فوق ماندگار نخواهند بود.

سربرگ Call

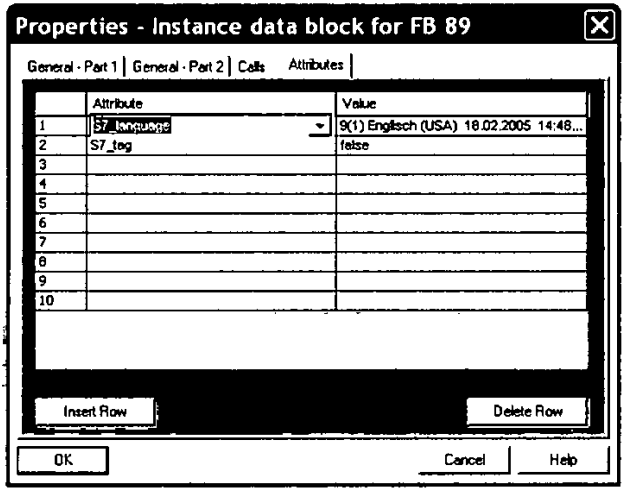
برای Instance DB یا DBهای که به UDT لینک شده باشند در این بخش می‌توان بلاک لینک شده را مشاهده نمود. شکل ۳-۶۶ DB اختصاص را برای FB41 نشان می‌دهد.



شکل ۳-۶۶ سربرگ Calls

سربرگ Attribute

در این قسمت می‌توان خصوصیات ویژه‌ای را برای دیتابلاک تعریف کرد. این ویژگی‌ها به‌صورت داخلی توسط برنامه مورد استفاده قرار می‌گیرد. برخی از این ویژگی‌ها با سیستم مانیتورینگ WinCC ارتباط پیدا کرده و در برنامه‌های PCS7 مورد استفاده قرار می‌گیرند.



شکل ۳-۶۷ سربرگ Attribute

۳-۱۷ دستور DB Call

در محیط برنامه‌نویسی LAD/STL/FBD برای استفاده از دیتا بلاک ابتدا بایستی آنرا باز نمود سپس از آدرس‌های آن استفاده کرد. این کار در برنامه‌نویسی به دو روش انجام می‌شود:

روش اول: با به‌کار بردن نام یا شماره دیتابلاک قبل از آدرس مورد نظر مانند:

DB1.DBW2

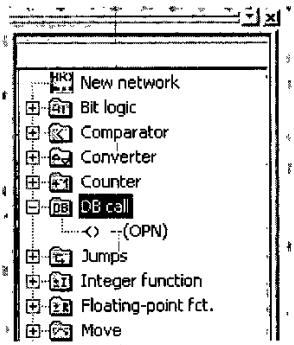
یا

Plant1.Pressure

در این دو روش به‌طور خودکار ابتدا برنامه، DB را باز کرده سپس آدرس مورد نظر را از آن می‌خواند. این روش در سیستم‌های قدیمی زیمنس مانند S5 قابل استفاده نبود.

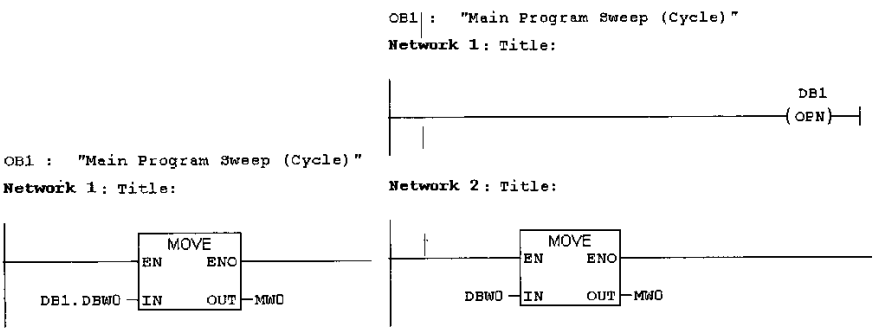
روش دوم: روش قدیمی است که در S5 استفاده می‌شد و در S7 نیز قابل استفاده است. در این روش ابتدا DB با دستور Open باز شده سپس آدرس مورد نظر بدون ذکر شماره DB از آن خوانده می‌شود.

شکل ۳-۶۸ محل قرارگیری دستور DB Call در پنجره دستورات را به زبان LAD نشان می‌دهد.



شکل ۳-۶۸ محل قرارگیری دستور DB Call در پنجره دستورات

شکل ۳-۶۹ روش اول و دوم را برای فراخوانی دیتا بلاک با یکدیگر مقایسه کرده است.



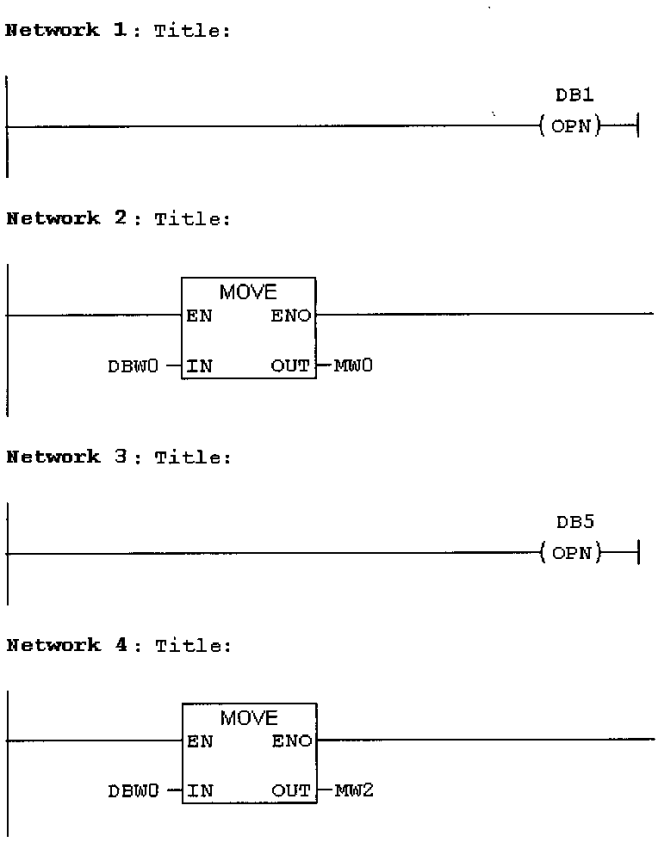
روش اول **روش دوم**

شکل ۳-۶۹ مقایسه دو روش فراخوانی DB

همانطور که در این شکل دیده می‌شود، در روش دوم برای خواندن از دیتابلاکی که باز شده مستقیماً آدرس DBW0 به‌کار رفته است و پیشوند نام DB که در روش اول به‌کار می‌رفت را ندارد.

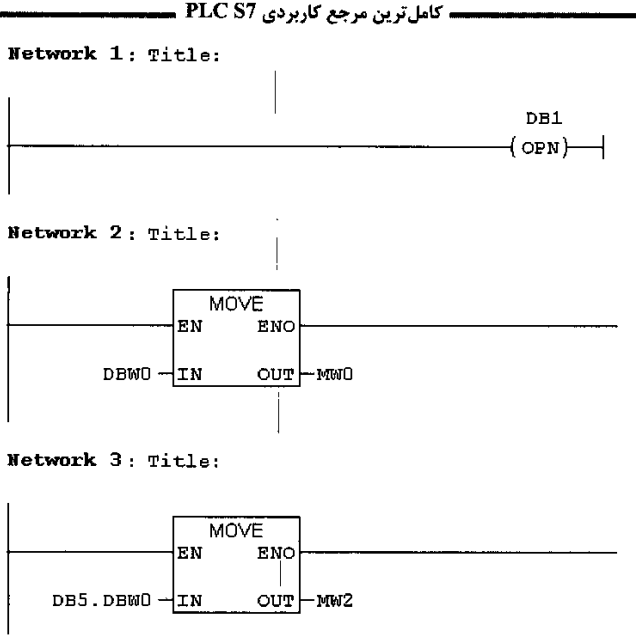
نکات قابل توجه

- در روش دوم اگر دستور OPN به‌کار نرود و مستقیماً به آدرس DB در برنامه اشاره شود، منجر به فالت و توقف CPU می‌گردد.
- در روش دوم نمی‌توان قبل از دستور OPN شرط خاصی به‌کار برد.
- در روش دوم اگر دستور OPN جدیدی که دیتابلاک دیگری را باز می‌کند به‌کار رود، به‌طور خودکار DB قبلی بسته می‌شود. در برنامه زیر MW0 از DB1 و Mw2 از DB5 مقدار می‌گیرد.



شکل ۳-۷۰ استفاده از دو دستور OPN

اگر در برنامه روش‌های اول و دوم به‌صورت ترکیبی استفاده شوند، با هر بار استفاده از یک روش، DB باز شده قبلی بسته شده و DB جدید باز می‌شود. در شکل ۳-۷۱ در Network3 دیتابلاک باز شده قبلی یعنی DB1 بسته شده و DB5 باز می‌گردد.



شکل ۳-۷۱ استفاده همزمان از دو روش فراخوانی DB

- در برخی برنامه‌ها استفاده از روش دوم الزامی است. نمونه آن برنامه‌نویسی با آدرس‌دهی غیر مستقیم است که در کتاب سطح تکمیلی مورد بحث قرار گرفته است.

۳-۱۸ ایجاد DB با فانکشن‌های سیستمی

دیتا بلاک را می‌توان توسط فانکشن‌های SFC خاصی ایجاد یا حذف نمود. این SFCها عبارتند از:

- SFC22: برای ایجاد دیتا بلاک
- SFC23 :برای حذف کردن دیتا بلاک

این فانکشن‌ها در کتاب سطح تکمیلی بحث می‌شوند. در اینجا فقط به برخی نکات اشاره می‌کنیم:

- وقتی دیتا بلاک با SFC ایجاد شود، آنرا در پوشه Offline نمی‌بینیم ولی در پنجره Online قابل مشاهده است و با آپلود کردن به پوشه Offline منتقل می‌گردند.
- دیتا بلاک‌هایی که با SFC ایجاد می‌شوند در RAM قرار می‌گیرند.
- شماره DB و تعداد بایت آن در پارامترهای SFC مشخص می‌شود.

۳-۱۹پرسش و تحقیق

ساختار دیتابلاک، انواع متغیرهای قابل تعریف در آن و سایر ویژگی‌های توضیح داده شده برای S7 در مورد PLC S5 چگونه است؟

۳-۲۰ تمرین

مثال ۲- ۴ را به‌صورتی باز نویسی کنید که به‌جای تمام متغیرهایی که از نوع M در برنامه به‌کار رفته است از دیتا بلاک استفاده گردد.

فصل ۳

فصل ۴

برنامه‌نویسی و کار با FC و FB

در این فصل ضمن تشریح ساختار FC و FB، نحوه برنامه‌نویسی با آن به همراه مثال‌های برنامه‌نویسی متعدد تشریح می‌شود.

چکیده مطالب

- برنامه‌نویسی می‌تواند به روش خطی یا ساختار یافته انجام شود.
- در فرآیندهای بزرگ استفاده از برنامه‌نویسی ساختار یافته اجتناب‌ناپذیر است.
- فانکشن (FC) و فانکشن بلاک (FB) در برنامه‌نویسی ساختار یافته به‌کار می‌روند.
- در برنامه‌نویسی ساختار یافته، ابتدا باید آخرین FC یا FB را ایجاد و برنامه‌نویسی کرد سپس مرحله به مرحله بلاک‌های قبلی را ایجاد نمود.
- بهترین کاربرد FB برای کنترل نهایی و بهترین کاربرد FC برای برنامه‌نویسی تقسیم شده است.
- تعداد فراخوانی تو در توی بلاک‌ها به ویژگی Nesting Depth هر CPU بستگی دارد.
- درFC و FB می‌توان‌از متغیرهای موقتTemp برای نتایج میان برنامه استفاده کرد. این متغیرها در حافظه L-Stack که بخشی از حافظه System Memory می‌باشد ذخیره می‌گردد.
- FB در هنگام فراخوانی نیاز به DB دارد.
- FC و FB دارای پارامترهای ورودی و خروجی باشند. ورودی و خروجی‌های FC ذخیره نمی‌شوند ولی ورودی و خروجی‌های FB در دیتا بلاک اختصاصی آن ذخیره می‌گردند.
- در FB می‌توان برای ذخیره‌سازی نتایج میان برنامه از متغیر Stat استفاده کرد. این متغیرها برخلاف نوع Temp در L-Stack ذخیره نمی‌شوند بلکه در دیتا بلاک می‌نشینند.
- برای دسترسی به آدرس‌های DB اختصاصی در سایر بلاک‌های برنامه‌نویسی، لازم است ابتدا DB با دستور OPN باز گردد.
- وقتی چند FB تو در تو صدا زده می‌شوند، می‌توان یک DB را به همه آنها اختصاص داد. به این قابلیت Multi Instance گفته می‌شود.
- تغییر در ساختار FB یا FC در حین کار و دانلود آن می‌تواند CPU را با مشکل مواجه کند.

فصل ۴

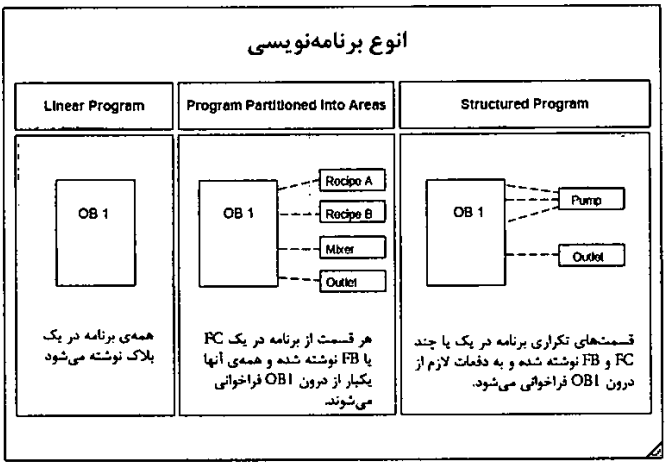
۴-۱ مقدمه

همانطور که در کتاب سطح مقدماتی اشاره شد، برنامه‌نویسی در Step7 به‌روش‌های مختلفی امکان‌پذیر می‌باشد. یکی از روش‌های برنامه‌نویسی که در پروژه‌های بزرگ از آن استفاده می‌شود، روش برنامه‌نویسی ساختاریافته است. در این روش از FC و FB به‌عنوان بلاک‌های برنامه‌نویسی استفاده شده و منطق اصلی کنترل در این بلاک‌ها نوشته می‌شود، سپس این بلاک‌ها را در OB فرا می‌خوانیم. در این فصل ضمن مقایسه روش برنامه‌نویسی ساختاریافته و خطی، به بررسی FC و FB و نکات مربوط به آنها خواهیم پرداخت.

۴-۲ مقایسه برنامه‌نویسی خطی و ساختار یافته

همانطور که قبلا اشاره شد، به‌طور کلی سه روش برنامه‌نویسی در Step7 وجود دارد که عبارتند از:

- برنامه‌نویسی خطی[1]
- برنامه‌نویسی تقسیم‌شده[2]
- برنامه‌نویسی ساختاریافته[3]



شکل ۴-۱ انواع روش‌های برنامه‌نویسی در Step7

در این بخش به بررسی و مقایسه برنامه‌نویسی خطی و ساختاریافته می‌پردازیم.

- در برنامه‌نویسی خطی، همه‌ی برنامه در یک بلاک و در Networkهای متوالی نوشته می‌شود. در واقع در این روش، برنامه مربوط به بخش‌های مختلف پروسه زیر هم در OB1 نوشته می‌شود. فرض نمایید در یک پروسه صنعتی ۱۰ ژنراتور وجود دارد که توسط یک PLC کنترل می‌شوند. در روش خطی برنامه مربوط به کنترل هر ژنراتور در تعدادی Network نوشته شده و همه آنها زیر یکدیگر در OB1 قرار می‌گیرند.

1. Linear
2. Partioned
3. Structured

- در روش برنامه‌نویسی ساختاریافته می‌توان برنامه کنترل را یک بار در یک FC و FB نوشته و آنرا به دفعات لازم در برنامه استفاده نمود. در واقع با این کار از نوشتن بخش‌های تکراری برنامه جلوگیری می‌شود. به‌عنوان مثال یک بلاک برای کنترل ژنراتور بطور عام تهیه می‌شود، سپس در OB1 به تعداد ژنراتورها آن را صدا می‌زنند و در هر بار صدا زدن پارامترهای همان ژنراتور را به آن اختصاص می‌دهند.
- در روش برنامه‌نویسی تقسیم شده برنامه هر بخش فرآیند در یک بلاک مجزا قرار می‌گیرد. این گونه برنامه‌نویسی برای کارهای تکراری به‌کار نمی‌رود، بلکه همانطور که از نامش پیداست، برای تقسیم‌بندی برنامه بخش‌های مختلف یا کارهای مختلف استفاده می‌شود. اگر بخواهیم ساده‌تر بحث کنیم، باید بگوییم که برنامه‌نویسی تقسیم شده نیز در زیر مجموعه برنامه‌نویسی ساختاریافته قرار می‌گیرد ولی در واقع با آن تفاوت دارد. در مثال ژنراتور، اگر به‌صورت تقسیم شده برنامه نوشته شود بایستی ۱۰ فانکشن مختلف ایجاد شود که هر فانکشن برای کنترل ژنراتور خاصی می‌باشد و پارامترها و آدرس‌های ژنراتور در داخل فانکشن به‌کار رفته است، به‌عبارت دیگر اگر چه منطق فانکشن‌ها یکسان است ولی آدرس به‌کار رفته در آنها یکسان نیست. در حالی که در برنامه‌نویسی ساختار یافته فانکشن برای تمام ژنراتورها دقیقاً یکی است ولی در هنگام فراخوانی آدرس‌های هر ژنراتور به‌صورت ورودی به آن داده می‌شود و نتیجه را در خروجی‌های فانکشن برمی‌گرداند. در برنامه‌نویسی تقسیم شده معمولاً فانکشن ورودی و خروجی ندارد.

۴-۲-۱ برنامه‌نویسی خطی

موارد کاربرد برنامه‌نویسی خطی

زمانی که پروسه کوچک بوده و تعداد I/Oها کم باشد، می‌توان از این روش استفاده نمود (اگرچه توصیه نمی‌شود)، ولی اگر پروسه بزرگ بوده و تعداد I/Oها زیاد باشد یا اینکه بخش‌های تکراری در برنامه وجود داشته باشد، استفاده از این روش راه‌حل مناسبی نیست.

فصل ۴

مزایای برنامه‌نویسی خطی

اگرچه برنامه‌نویسی خطی دارای معایب زیادی است، ولی چند مورد در آن وجود دارد که شاید بتوان از آنها به‌عنوان مزایای این روش برنامه‌نویسی یاد نمود. این موارد عبارتند از:

- **کاهش زمان سیکل‌اسکن**

در روش خطی از آنجایی که سیستم‌عامل CPU فقط OB1 را اجرا کرده و بلاک‌های دیگر را فراخوان نمی‌کند، لذا تا حدی زمان سیکل‌اسکن کاهش پیدا می‌نماید؛ ولی این زمان در مقایسه با کل زمان سیکل، قابل ملاحظه نیست.

- **سهولت برنامه‌نویسی (در پروسه‌های کوچک)**

از آنجایی که در این روش، لازم نیست کاربر بلاک‌های مختلف را ایجاد و فراخوانی نماید، در برنامه‌های کوچک برنامه‌نویسی به این روش آسان است.

معایب برنامه‌نویسی خطی

- **افزایش زمان برنامه‌نویسی**

از آنجایی که در این روش برنامه‌های تکراری نیز نوشته می‌شوند، لذا برنامه‌نویس باید زمان بیشتری را صرف برنامه‌نویسی نماید.

- **عیب‌یابی مشکل**

عیب‌یابی برنامه نوشته شده به روش خطی کاری مشکل و زمان‌بر است. چون در این روش، پیدا نمودن بخشی از برنامه که مربوط به کنترل سیستمی است که دچار خطا شده، کاری مشکل است و استفاده از ابزار عیب‌یابی نیز در آن به‌آسانی امکان‌پذیر نیست.

- **پیچیدگی برنامه‌نویسی (در پروسه‌های بزرگ)**

در پروسه‌های بزرگ که منطق کنترل نیز پیچیده است، این روش می‌تواند برنامه‌نویس را دچار سردرگمی نموده به‌نحوی که عملا امکان برنامه‌نویسی وجود نداشته باشد.

- **امکان ایجاد خطا در برنامه‌نویسی**

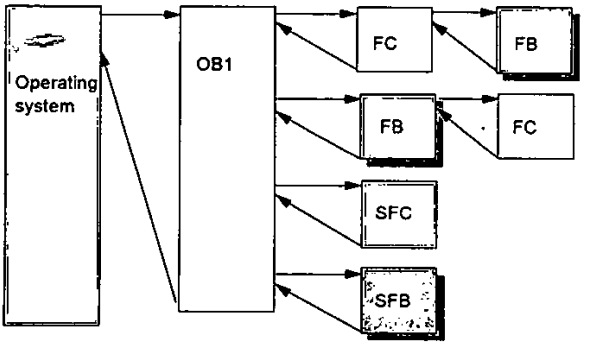
در این روش از آنجایی که برنامه کنترل همه بخش‌های فرآیند در یک بلاک نوشته می‌شود، امکان اشتباه توسط برنامه‌نویس را بالا می‌برد. تلاقی آدرس‌ها و کاربرد آدرس‌های به‌جای یکدیگر از جمله این اشتباهات است.

۴-۲-۲ برنامه‌نویسی ساختاریافته

موارد کاربرد برنامه‌نویسی ساختاریافته

به‌دلیل مشکلاتی که در روش برنامه‌نویسی خطی ذکر شد، معمولاً از برنامه‌نویسی ساختاریافته (و تقسیم‌شده) استفاده می‌شود. در این روش، بخش‌های مختلف برنامه در بلاک‌های FC و FB نوشته شده و در برنامه از آنها استفاده می‌شود. از جمله موارد کاربرد این روش برنامه‌نویسی می‌توان به موارد زیر اشاره نمود:

- زمانی که در برنامه بخش‌های تکراری وجود داشته باشد، با این روش می‌توان یک بار برنامه مورد نظر را در یک FC یا FB نوشته و به دفعات از آن در برنامه‌ی اصلی استفاده نمود.
- زمانی که نیاز به طراحی فانکشن خاصی باشد می‌توان این فانکشن را در یک FC یا FB نوشته و از آن در برنامه استفاده نمود. مثلاً ممکن است کاربر بخواهد یک برنامه جهت Scale نمودن دیتاهای آنالوگ طراحی نماید. در این حالت می‌توان برنامه مورد نظر را در یک FC یا FB پیاده‌سازی نموده و در برنامه از آن استفاده نمود.



شکل ۴-۲ برنامه‌نویسی ساختار یافته

مزایای برنامه‌نویسی ساختاریافته

از جمله مزایای برنامه‌نویسی ساختاریافته می‌توان به موارد زیر اشاره نمود:

- **کاهش زمان برنامه‌نویسی**

در این روش چون از نوشتن بخش‌های تکراری برنامه اجتناب می‌شود، لذا زمانی که برنامه‌نویس صرف نوشتن برنامه می‌نماید کاهش می‌یابد.

- **سهولت برنامه‌نویسی**

در این روش چون برنامه‌ها در بلاک‌های مختلف نوشته می‌شوند، لذا امکان مدیریت روی نوشتن برنامه وجود داشته و برنامه‌نویس دچار سردرگمی نخواهد شد.

- **امکان عیب‌یابی سریع**

در این روش چون منطق کنترل هر بخش در بلاکی جداگانه نوشته می‌شود، لذا امکان دسترسی به برنامه بخشی که دچار اشکال شده است سریعتر امکان‌پذیر می‌باشد. مثلاً فرض نمایید که در یک سیستم صنعتی که دارای بخش‌های مختلف است، برنامه کنترل مربوط به پمپ‌های آب در یک FC نوشته شده باشد. در صورتی‌که منطق کنترل به‌طور صحیح پیاده‌سازی نشود، می‌توان به‌راحتی به FC مذکور مراجعه نموده و اشکال را بررسی نمود.

معایب برنامه‌نویسی ساختاریافته

از مواردی که شاید بتوان از آن به‌عنوان یک عیب برای این روش برنامه‌نویسی یاد نمود، افزایش زمان سیکل‌اسکن به‌دلیل دستورات فراخوانی بلاک‌هاست. البته باید توجه داشت که امروزه سرعت پردازش CPUهای S7-300 و 400 آنقدر بالا است که این افزایش زمان، محسوس نیست. بنابراین افزایش زمان‌سیکل‌اسکن آنقدر ناچیزاست‌که‌می‌توان آنرا نادیده گرفت.

فصل ۴

۴-۳ انواع متغیرها و پارامترها در فانکشن و فانکشن بلاک

قبل از تشریح بحث FC و FB لازم است با انواع متغیرها و پارامترهایی که در آنها استفاده می‌شود آشنا شویم. قبل از تشریح مطلب به نکات زیر توجه نمایید:

- متغیرهای FC و FB مواردی هستند که برای ذخیره‌سازی نتایج میان برنامه از آنها استفاده می‌شود. به‌عبارت دیگر این متغیرها می‌توانند جایگزین bit Memory ها شوند. این متغیرها در حافظه‌ای با عنوان L-Stack مخفف Local Stack ذخیره می‌شوند. متغیرهای فوق در برنامه‌نویسی ساختار یافته و برنامه‌نویسی تقسیم شده می‌توانند به‌کار روند.
- پارامترهای FC و FB مواردی هستند که به‌عنوان ورودی یا خروجی استفاده می‌شوند. ورودی‌ها مقادیر را از بیرون FC یا FB به داخل آنها منتقل می‌کنند. خروجی‌ها نتایج به‌دست آمده از پردازش برنامه FC یا FB را به بیرون آنها منتقل می‌کنند. پارامترهای فوق برای برنامه‌نویسی ساختار یافته کاربرد دارند.
- به مجموعه متغیرها و پارامتر فوق Local Variable یا متغیرهای محلی گفته می‌شود که در قسمت Interface ابتدای FC و FB تعریف می‌شوند.

۴-۳-۱ انواع متغیرها در Step7

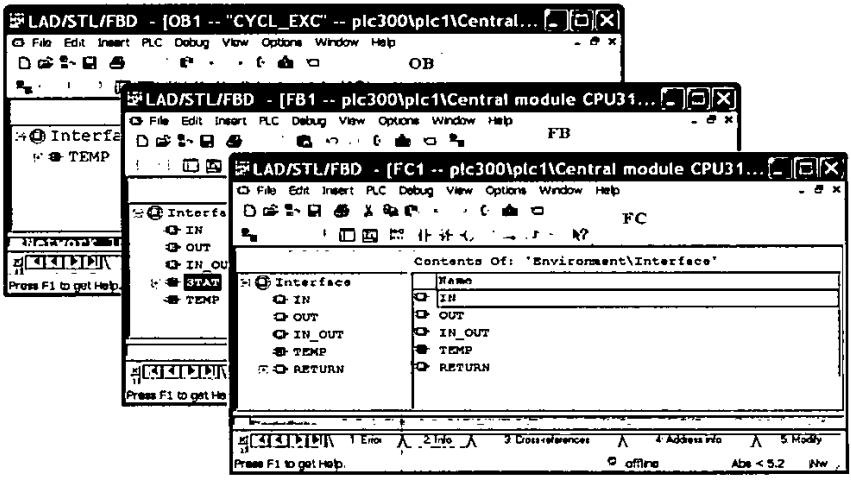
انواع متغیرهای موجود در Step7 را می‌توان به‌طور کلی به دو گروه سراسری و محلی تقسیم نمود:

الف) متغیرهای سراسری[1]

این متغیرها در همه‌ی بلاک‌های برنامه‌نویسی معتبر بوده و از آنها می‌توان در همه‌ی بلاک‌ها استفاده نمود. از جمله‌ی این متغیرها می‌توان به I, Q, M, T, C, DB اشاره نمود. این متغیرها می‌توانند به‌صورت Absolute یا Symbolic آدرس‌دهی شوند. اگر این متغیرها به‌صورت سمبلیک تعریف شوند مشخصه آنها در برنامه با " است. به‌عنوان مثال متغیر "Pressure" در برنامه معرف یک متغیر سراسری است. اگر مقدار این متغیرها در یک بلاک برنامه‌نویسی اعم از OB یا FB یا FC تغییر کند در سایر بلاک‌ها نیز مقدار جدید در دسترس است. نام متغیرهای سمبلیک در کل برنامه بایستی منحصر بفرد باشد.

ب) متغیرهای محلی[2]

متغیرهای محلی، فقط در بلاکی که تعریف شده‌اند معتبر می‌باشند و از آنها نمی‌توان به‌عنوان واسط، جهت ذخیره داده در بلاک‌های مختلف استفاده نمود. این متغیرها در قسمت Declaration Section بلاک تعریف می‌شوند و مشخصه‌ی آنها در آدرس‌دهی، علامت # در قبل از شروع نام متغیر می‌باشد. نام متغیر محلی در همان بلاک بایستی منحصر بفرد باشد ولی می‌توان در دو بلاک مختلف متغیرهای محلی با اسم مشابه تعریف نمود.



شکل ۴-۳ ناحیه Declaration برای FC , FB , OB

1. Global Variable
2. Local Variable

به‌عنوان مثال Pressure# یک متغیر محلی است که فقط در همان FB یا FC یا OB معتبر است و در بلاک‌های دیگر قابل دسترسی نیست. این متغیرها را می‌توان با اسم یا با آدرس‌دهی که با حرف L شروع می‌شود به‌کار برد مانند L0.0 یا LW2 حرف L معرف Local بودن متغیر است.

شکل ۴-۴ انواع متغیرهای موجود در Step7 را نشان می دهد.

Global Variables / Data (valid in the entire program)	Local Variables / Data (only valid in one block)	
• PII / PIQ • I/ O • M / T / C • DB areas	Temporary Variables • temporary storage in L stack • usable in OBs / FCs / FBs	Static Variables • are retained even after the block is executed • permanent storage in DBs • can be used in FBs only

شکل ۴-۴ انواع متغیرها در Step7

همانطور که در این شکل مشاهده می‌شود، متغیرهای محلی به دو نوع تقسیم می‌شوند:

- متغیرهای نوع Temp یا Temporary: این متغیرها برای همه بلاک‌های برنامه‌نویسی یعنی FC و FB و OB وجود دارند. مقدار این متغیرها با بسته شدن بلاک اعتبار نخواهد داشت.
- متغیرهای نوع Stat یا Static: این متغیرها فقط برای FB وجود دارند. مقدار این متغیرها با بسته شدن بلاک از بین نمی‌رود زیرا در حافظه‌ای از جنس DB ذخیره می‌گردد. (این متغیرها از حافظه L-Stack استفاده نمی‌کنند.)

دو نوع متغیر فوق در ادامه تشریح شده‌اند.

فصل ۴

۱- متغیرهای نوع Temp (Temporary Variables): این متغیرها از نوع موقتی بوده و در همه‌ی بلاک‌های برنامه‌نویسی (OB,FC,FB) می‌توانند ایجاد شوند. محل ذخیره‌سازی این متغیرها ناحیه‌ی L stack (Local Data Stack) در حافظه‌ی سیستمی CPU می‌باشد. در مورد L Stack در ادامه بحث توضیحات لازم ارائه می‌شود. متغیرهای نوع Temp اگرچه به‌عنوان متغیرهای موقتی شناخته می‌شوند، اما در عمل با پایان اجرای بلاکی که این متغیرها در آن ایجاد شده‌اند، مقادیرشان پاک نمی‌شود. دلیل این موضوع ذخیره‌سازی آنها در L Stack می‌باشد. البته باید توجه نمود در زمانی که یک بلاک که متغیر موقتی در آن تعریف شده در حال اجرا نمی‌باشد، دسترسی به مقادیر موجود در متغیرهای موقتی امکان‌پذیر نیست.

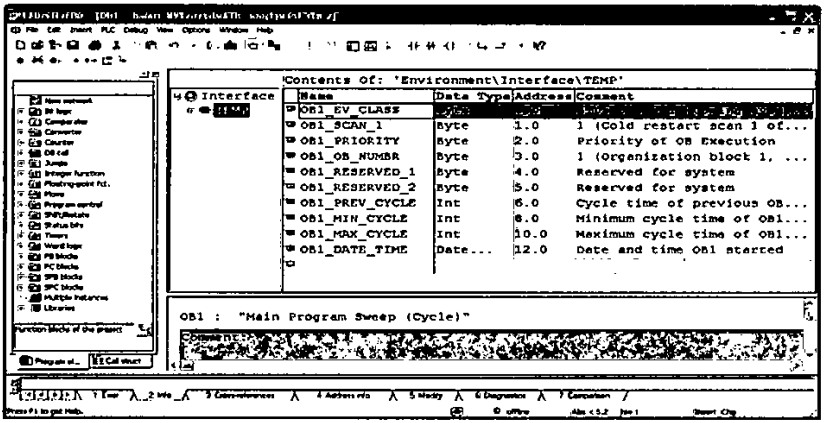
توجه: در صورت راه‌اندازی مجدد CPU، محتویات L Stack پاک می‌شود.

وقتی یک OB را باز می‌کنیم، در قسمت بالای آن که به Declation Table موسوم است در زیر مجموعه Temp متغیرهای از قبل تعریف شده‌ای را می‌بینیم. به‌دنبال این متغیرها کاربر می‌تواند متغیرهای دلخواه خود را تعریف کند. ولی وقتی FC یا FB را باز می‌کنیم هیچ متغیر از قبل تعریف شده‌ای در آن نمی‌بینیم و کاربر در صورت لزوم بایستی آنها را تعریف نماید. با این توضیح، متغیرهای نوع Temp را می‌توان به دو نوع زیر تقسیم نمود:

الف) متغیرهای محلی ایجاد شده‌توسط سیستم

این متغیرها در OBها وجود داشته و هر کدام اطلاعات خاصی، مطابق با عملکرد OB مربوطه در اختیار کاربر قرار می‌دهد. این اطلاعات فقط خواندنی هستند و کاربر نمی‌تواند در آنها چیزی بنویسد.

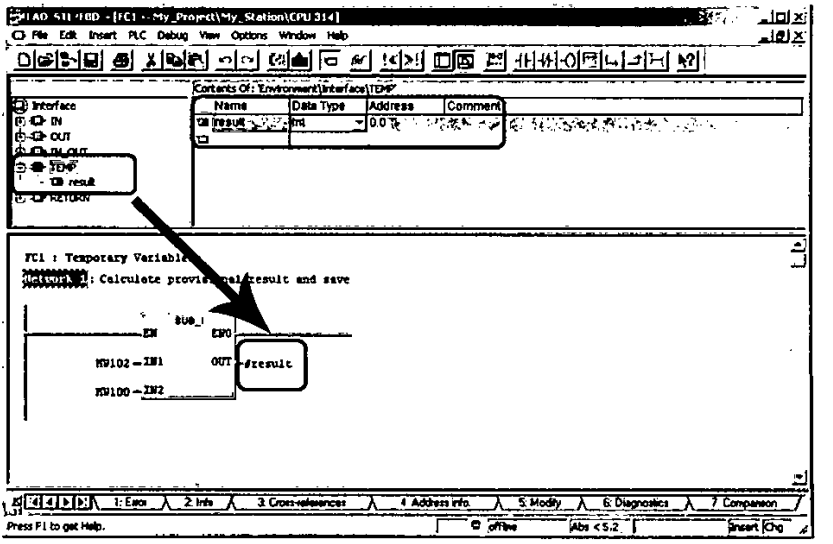
به‌عنوان مثال در OB1 متغیر محلی به نام #OB1_MAX_CYCLE وجود دارد که بیشترین زمان اجرای OB1 را به‌صورت عدد صحیح بر حسب میلی‌ثانیه نشان می‌دهد. شکل ۴-۵ متغیرهای Temp موجود در OB1 را نشان می‌دهد.



شکل ۴-۵ متغیرهای محلی از نوع Temp موجود در OB1

ب) متغیرهای محلی ایجاد شده‌توسط کاربر

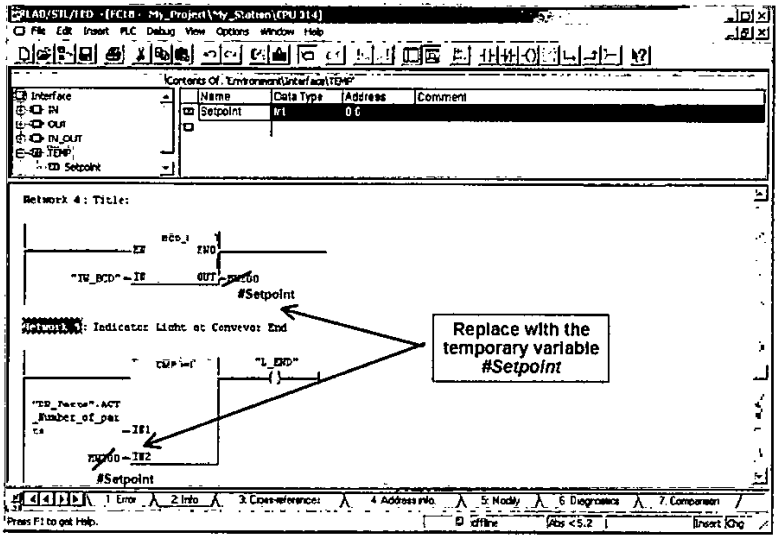
این متغیرها توسط کاربر ایجاد شده و نقشی شبیه Bit Memoryها ایفا می‌نمایند. یعنی می‌توان از این متغیرها جهت ذخیره اطلاعات میان‌برنامه استفاده نمود. این نوع متغیرها را در هر بلاک برنامه‌نویسی (OB,FC,FB) در قسمت Interface و زیرشاخه Temp می‌توان ایجاد و در برنامه‌ی همان بلاک به‌کار برد.



شکل ۴-۶ متغیرهای Temp ایجاد شده توسط کابر

همانطور که در شکل ۴-۶ دیده می‌شود، یک متغیر Temp به‌نام result از نوع Integer در FC1 ایجاد شده و نتیجه عمل تفریق MW100 و MW102 در آن قرار گرفته است.

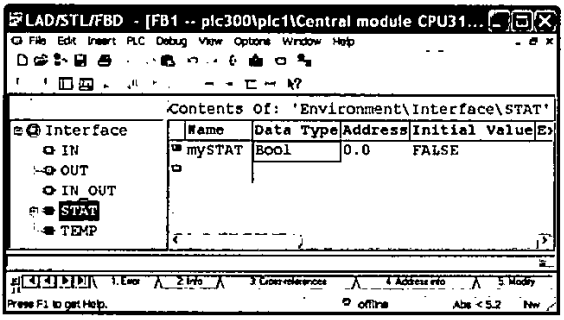
شکل ۴-۷ نشان می‌دهد که چگونه به‌جای استفاده از Memory جهت ذخیره‌سازی نتایج میان‌برنامه، از متغیرهای محلی استفاده شده است.



شکل ۴-۷ جایگزینی متغیر محلی به‌جای متغیرهای سراسری

۲- متغیرهای نوع Static (Static Variables): این متغیرها را فقط در FB می‌توان تعریف نمود. متغیرهایی که در این قسمت تعریف شوند، در دیتابلاک اختصاصی FB ذخیره شده و حتی پس از پایان اجرای بلاک، قابل دسترسی می‌باشند. یعنی در زمانی که بلاک مورد نظر (که متغیر Static در آن ایجاد شده است) اجرا نمی‌شود، باز امکان دسترسی به متغیرهای Static وجود دارد. متغیرهای Static در حافظه L-Stack که بخشی از System Memory است قرار نمی‌گیرند و محل ذخیره‌سازی آنها DB خاص FB است که در Load Memory قرار دارد.

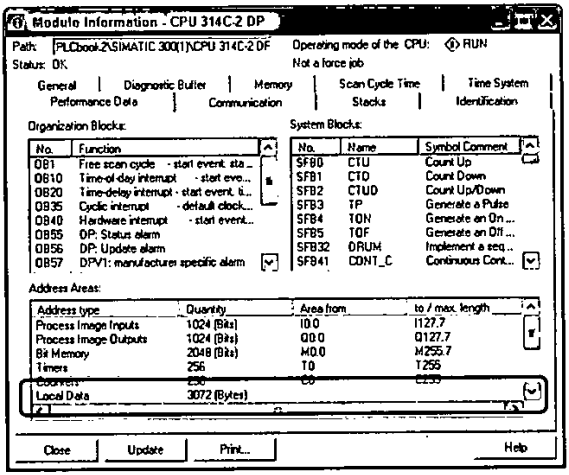
شکل ۴-۸ متغیر STAT را برای FB نشان می‌دهد.



شکل ۴-۸ متغیر STATIC

(L Stack) Local Data Stack

L- Stack بخشی از حافظه است که در System Memory قرار دارد و صرفاً برای متغیرهای Temp مورد استفاده قرار می‌گیرد. میزان اندازه L stack بستگی به مشخصات CPU داشته و در هر CPU ممکن است با CPU دیگر متفاوت باشد. این اندازه معمولاً در کاتالوگ فنی CPU درج می‌شود. علاوه بر آن در حالت Online می‌توان در پنجره Properties مربوط به CPU و در سربرگ Performance Data آنرا مشاهده نمود.



شکل ۴-۹ میزان حافظه Local Data

اندازه L Stack در CPUهای جدید، نسبت به CPUهای قدیمی بسیار افزایش پیدا نموده است. جدول ۴-۱ اندازه L Stack را برای برخی از CPUهای جدید S7-300 نشان می‌دهد.

جدول ۴-۱

	CPU 312	CPU 314	CPU 315-2 DP	CPU 317-2 PN/DP
Local data per priority class, max.	32 Kibyte; Max. 2 KB per block	32 Kibyte; Max. 2 KB per block	32 Kibyte; Max. 2 KB per block	32 Kibyte; Max. 2 KB per block

همانطور که در این جدول دیده می‌شود، حافظه Local به‌صورت کلی مطرح شده است و نمی‌توان گفت که چقدر برای FC یا FB منظور شده است. فاکتور اصلی بر اساس کلاس اولویت Priority Class است. برای فهم Priority Class بایستی ذکر کنیم که در CPU تعداد متنوعی OB وجود دارد که هر کدام در یک کلاس اولویت قرار می‌گیرند. اگر توسط سیستم عامل CPU دو OB فراخوان شود ابتدا آنکه کلاس اولویت بالاتر دارد اجرا می‌گردد.

به‌عنوان مثال OB1 در کلاس اولویت ۱ قرار دارد، ولی OBهای دیگر مانند OB10 یا OB35 و ... که در فصل بعد تشریح می‌شوند کلاس‌های اولویت بالاتر دارند.

Local Data Stack Size

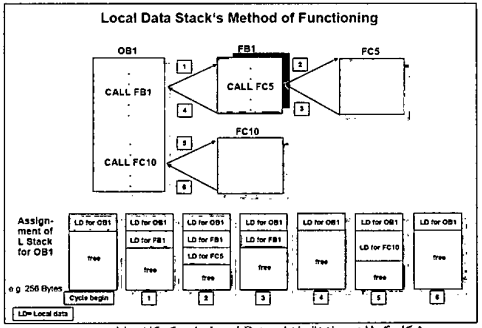
Entire size: 1.5 Kbyte (CPU 313..316)

Execution		For S7-300™: Priority class	L stack size
Startup (one-time execution)		27	256 bytes
Cyclic execution		1	
Time-controlled execution	Time-of-Day Interrupt	2	256 bytes
	Time-Delay Interrupt	3	256 bytes
	Cyclic Interrupt	12	256 bytes
Event-driven execution	Hardware Interrupt	16	256 bytes
	Error handling in startup	28	256 bytes
	Error handling in scan cycle	26	

شکل ۴-۱۰ تقسیم‌بندی فضای L Stack برای کلاس‌های مختلف در CPUهای قدیمی S7-300

برای هر کلاس اولویت حجم حافظه معینی Local Data قابل استفاده است. به‌عنوان مثال، با توجه به مشخصات ذکر شده در شکل بالا اگر FC و FBهای مختلفی در OB1 صدا زده شوند، می‌توان آنقدر پارامترهای محلی Temp و Stat در آنها تعریف کرد تا مجموع حافظه آنها از ۲۵۶ بایت که مربوط به این CPU می‌باشد فراتر نرود. ولی حافظه محلی مربوط به OBهای سیکلی[1] که با OB3x معرفی می‌شوند جداست و در آنها و FC و FB به‌کار رفته در آنها نیز می‌توان ۲۵۶ بایت دیتای محلی تعریف نمود.

شکل ۴-۱۱ به‌طور دقیق‌تر این موضوع را برای کلاس اولویت ۱ نشان می‌دهد.



شکل ۴-۱۱ نحوه اشغال فضای Local Data برای یک کلاس اولویت

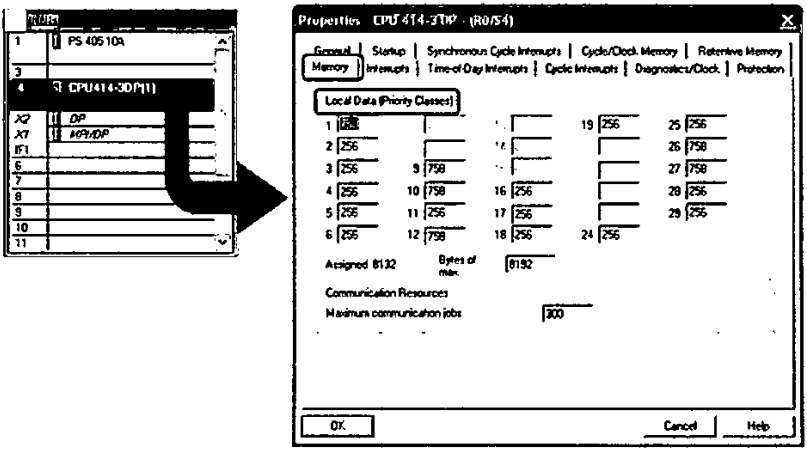
1. Cyclic Interupt

همانطور که در شکل ۴-۱۱ مشخص است:

- در سیکل آغازین فقط دیتاهای محلی OB1 فضای مربوط به کلاس اولیت OB1 را اشغال می‌کنند. یعنی تا قبل از فراخوانی FC/FB فقط متغیرهای Temp مربوط به OB1 از ناحیه حافظه Local استفاده می‌کنند و بقیه فضای ۲۵۶ بایت خالیست.
- وقتی FB1 صدا زده می‌شود پارامترهای Temp مربوط به FB1 نیز فضای دیگری را از Local Memory اشغال می‌کنند. توجه شود که در این حالت مجموع متغیرهای OB1 و FB1 از این ناحیه استفاده می‌کنند.
- وقتی از داخل FB1 فانکشن FC5 فراخوان می‌شود پارامترهای Temp مربوط به FC5 نیز علاوه بر موارد قبلی حافظه جدیدی را به‌خود اختصاص می‌دهند.
- تا زمانی که بلاک FC یا FB باز است و اجرای آن تمام نشده حافظه فوق را اشغال می‌کند ولی پس از پایان اجرا و برگشت به بلاک قبلی، حافظه مربوطه آزاد می‌شود ولی OB1 همواره به اندازه پارامترهای Temp خودش فضایی را اشغال می‌کند.
- با توجه به شکل در ادامه برنامه فانکشن FC10 صدا زده شده است. در این حالت حافظه Local توسط متغیرهای Temp مربوط به OB1 و FC10 اشغال شده است.
- اگر چه OB1 و FC و FBهای فراخوان شده در آن از یک فضای حافظه استفاده می‌کنند ولی متغیر Temp که در یکی از این بلاک‌ها تعریف می‌شود در دیگری قابل استفاده نیست زیرا ناحیه حافظه هر کدام مجزاست.
- برای اینکه ببینیم آیا حافظه اشغال شده از Local Memory از حد مجاز بیشتر شده یا نه، بایستی با توجه به شکل فوق در هر بار فراخوانی از ابتدای بیرون رفتن از OB1 تا برگشت به OB1 حجم دیتاهای Temp را محاسبه و با میزان ماکزیمم مربوط به CPU مقایسه کنیم. نحوه مشاهده میزان Local Data استفاده شده توسط هر بلاک در Propeties بلاک قابل مشاهده است که در ادامه تشریح می‌شود.

نکته: مدیریت فضای L-Stack در S7-400

در S7-400 امکان مدیریت فضای اختصاص یافته به هر کلاس اولویت وجود دارد. برای انجام این عمل می‌توان وارد محیط HW Config شده و در پنجره مشخصات CPU و در سربرگ Memory این تنظیم را انجام داد. به شکل ۴-۱۲ توجه کنید.



شکل ۴-۱۲ تعیین فضای اختصاص یافته به هر کلاس اولویت

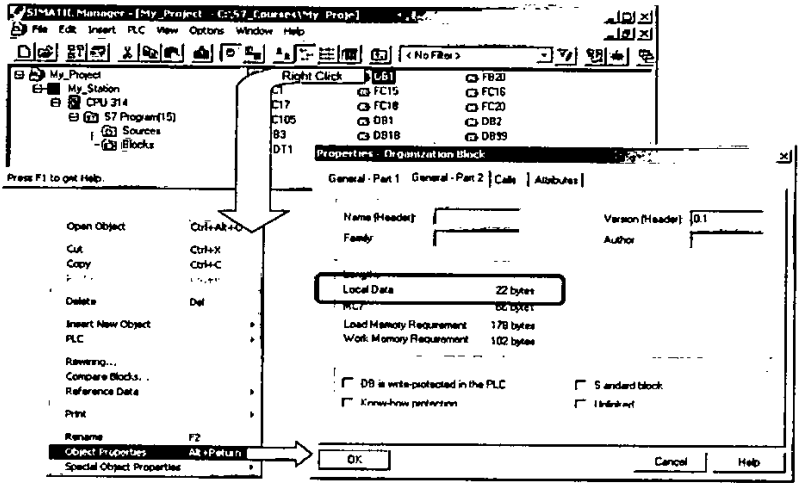
همانطورکه در شکل ۴-۱۲دیده می‌شود، کلاس‌های اولویت مختلف که می‌توان میزان فضای اختصاص یافته L Stack به آنها را تغییر داد در این سربرگ وجود دارند. در جلوی هر کلاس اولویت، می‌توان عددی را (بر حسب بایت) وارد نمود. مثلاً این CPU دارای فضای L Stack به میزان 8 Kbyte می‌باشد که می‌توان آنرا بین کلاس‌های اولویت مختلف تقسیم‌بندی نمود. البته در این CPU باید توجه نمود که جمع کل فضاهای اختصاص داده شده از 8 Kbyte بیشتر نشود.

نکته: حفظ یا پاک شدن مقادیر L Stack

در واقع برای هر کلاس اولویت یک میزان حافظه مشخصی از ناحیه L Stack به‌صورت عمومی در اختیار بلاک‌هایی که در آن کلاس استفاده شده‌اند قرار می‌گیرد. در هر سیکل اسکن بلاک‌هایی که در آن کلاس اولویت در حال اجرا هستند از طریق متغیرهای محلی خودشان اجازه نوشتن در این ناحیه را دارا می‌باشند. سوالی که طرح می‌شود این است که آیا با پایان اجرای بلاکی که متغیر محلی خاصی در آن تعریف شده است، مقدار این متغیر در L Stack از بین می‌رود؟ پاسخ این است که اگر توسط سایر بلاک‌هایی که در این اولویت فراخوانی می‌شوند، در آدرس مذکور از Lstack (که بلاک قبلی مقدار متغیر محلی را در آنجا قرار داده) مقدار جدیدی قرار نگیرد، مقدار قبلی حفظ می‌شود. در غیر اینصورت مقدار جدید جایگزین مقدار قبلی می‌شود. به‌همین دلیل مشاهده می‌شود که دیتاهای موجود در L Stack گاهی حفظ و گاهی پاک می‌شوند.

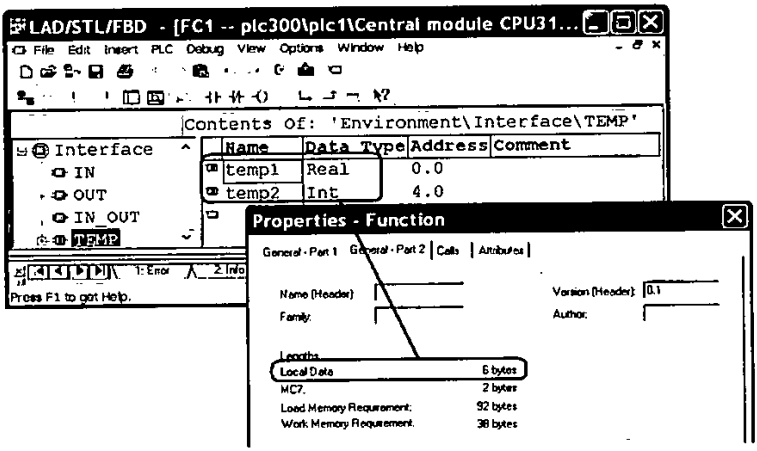
مشاهده اندازه دیتاهای محلی (Temp) یک بلاک

برای مشاهده اندازه دیتاهای محلی موجود در یک بلاک می‌توان روی نام بلاک در محیط Simatic Manager کلیک راست نموده و مطابق شکل ۴-۱۳ عمل نمود. همانطور که در این شکل مشخص است، جلوی گزینه Local Data مقدار 22 Byte نوشته شده است یعنی این بلاک دارای 22 Byte متغیر محلی می‌باشد.



شکل ۴-۱۳ مشاهده میزان متغیرهای محلی بلاک OB1

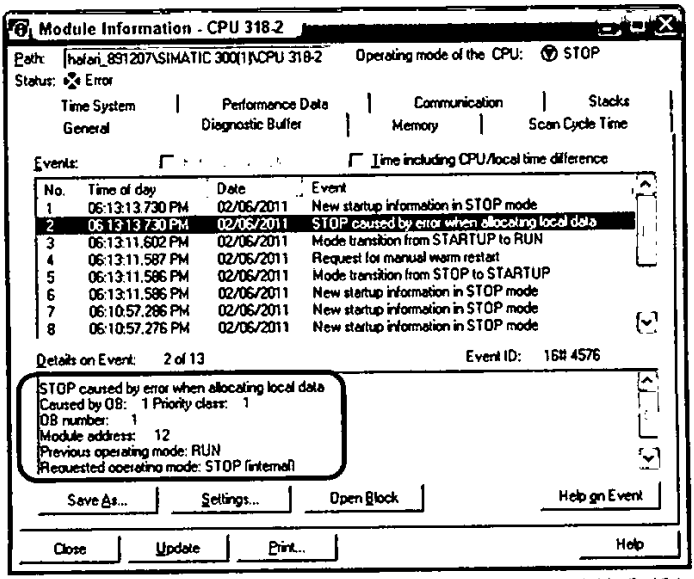
اگر به‌عنوان مثال یک FC ایجاد کرده و در قسمت متغیرهای محلی آن دو متغیر به‌صورت زیر یکی از جنس Integer و دیگری از جنس Real تعریف کنیم، پس از ذخیره سازی اگر Properties این فانکشن را ببینیم مشاهده می‌کنیم که ۶ بایت فضا به Local Data اختصاص داده است.



شکل ۴-۱۴ مشاهده فضای اشغال شده از Local Dta توسط یک FC

تمرین ۴-۱: با ایجاد یک FB مورد فوق را تست کنید. آیا تعریف متغیرهای Static تأثیری روی میزان Local Data خواهد داشت؟

تذکر: در صورتی‌که در ضمن اجرای برنامه، میزان دیتاهای محلی در یک کلاس اولویت از حد مجاز بیشتر شود، CPU متوقف شده و پیام *STOP caused by error when allocating local data* در بافر CPU ثبت می‌شود. به شکل ۴-۱۵ توجه کنید.

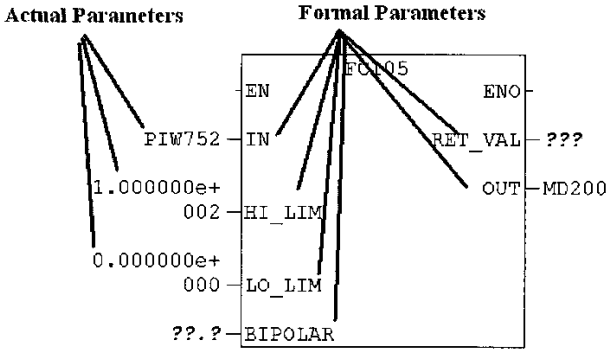


شکل ۴-۱۵ بافر CPU در حالت خطای افزایش بیش از حد متغیرهای محلی در یک کلاس اولویت

۴-۳-۲ انواع پارامترهای قراردادی Formal Parameters

منظور از پارامترهای قراردادی مواردی هستند که واسط بین ورودی و خروجی FC وFB محیط بیرون آن قرار می‌گیرند. به‌عبارت دقیقتر، Formal Parameter رابط بین Actual Parameter و برنامه یک FC یا FB هستند.

به‌عنوان مثال FC105 که فانکشن Scale زیمنس است و در فصل قبل تشریح شد را در نظر بگیرید. اسامی که در ورودی و خروجی‌های آن ظاهر می‌شود Formal Parameters است. به شکل ۴-۱۶ توجه کنید.

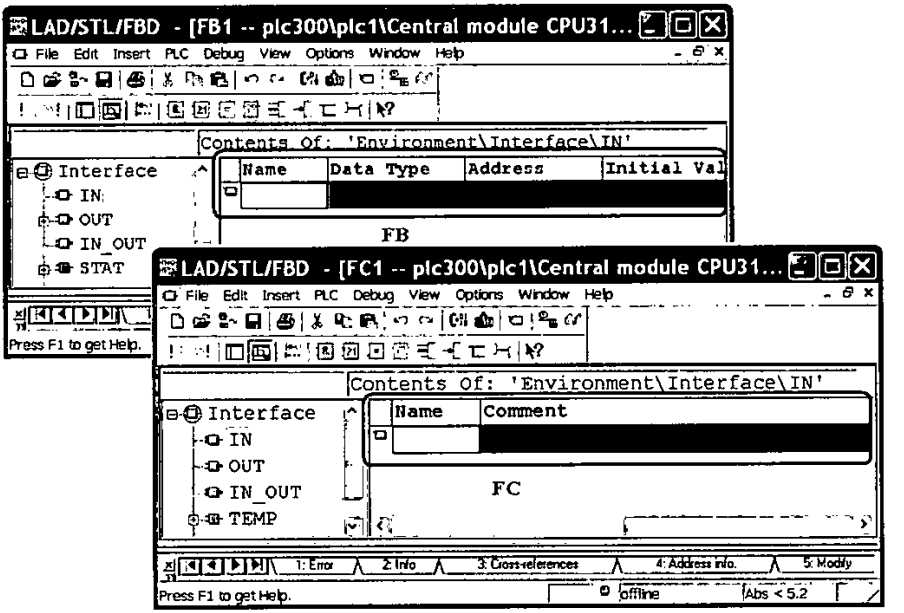


شکل ۴-۱۶ پارامترهای Actual و Formal

فصل ۴

نکات قابل توجه

- Formal Parameter فقط برای FC و FB قابل تعریف است. OB دارای هیچ ورودی و خروجی نیست.
- Formal Parameter به سه دسته IN و OUT و IN_OUT تقسیم می‌شوند.
- IN برای گرفتن مقادیر Actual در ورودی FC / FB به‌کار می‌رود.
- OUT برای ارسال نتایج برنامه به پارامترهای Actual درج شده در خروجی FC/FB به‌کار می‌رود.
- IN_OUT: می‌تواند به‌عنوان IN یا OUT هر دو به‌کار رود.
- مهمترین کاربرد پارامترهای قراردادی در برنامه‌نویسی ساختار یافته است. کاربر می‌تواند برنامه‌ی یکسانی را نوشته و در هر فراخوانی بلاک، آدرس‌های Actual آنرا عوض نماید.
- Formal Parameter حافظه ای در L-Stack ندارد.
- در FB مقادیر واقعی که به Formal Parameter داده می‌شود یا از آنها گرفته می‌شود در دیتابلاک اختصاصی آن ذخیره می‌گردد از اینرو دارای آدرس است. در FC این مقادیر در هیچ حافظه‌ای ذخیره نمی‌شود و دارای هیچ آدرسی نیست. به شکل ۴-۱۷ توجه کنید.



شکل ۴-۱۷ مقایسه پارامترهای قراردادی FC و FB از نظر وجود آدرس

با استفاده از پارامترهای قراردادی می‌توان در FC یا FB مورد نظر برنامه‌ای نوشت که در هر بار فراخوانی امکان اختصاص آدرس‌های متفاوت وجود داشته باشد. فرض کنید در یک پروسه ۱۰ مخزن با منطق کنترل یکسان ولی آدرس‌های ورودی و خروجی متفاوت وجود داشته باشند. در اینصورت با استفاده از پارامترهای قراردادی موجود در FC یا FB می‌توان برنامه را به‌گونه‌ای طراحی نمود که در هربار فراخوانی آن FC یا FB، آدرس‌های مربوط به یکی از مخازن وارد شود. بدین ترتیب می‌توان FC یا FB مورد نظر را ده بار فراخوانی نموده و هربار آدرس‌های یکی از مخازن را وارد نمود.

در شکل ۴-۱۸ انواع پارامترهای قراردادی قابل تعریف در FC و FB با ذکر خصوصیات آنها نشان داده شده است.

Formal parameters

Type of parameter	Declaration	Use	Graphic Display
Input parameter	in	Read only	To the left of the block
Output parameter	out	Write only	To the right of the block
In/out parameter	In_out	Read / write	To the left of the block

شکل ۴-۱۸ انواع پارامترهای قراردادی قابل تعریف در FC و FB

۴-۴ مقایسه FC و FB

همانطور که اشاره شد، در برنامه‌نویسی ساختاریافته می‌توان از FC و FB استفاده نمود. این دو بلاک از برخی جنبه‌ها مشابه و از برخی جنبه‌های دیگر متفاوت هستند. این موارد را می‌توان به‌صورت زیر خلاصه نمود:

تشابه FC و FB

- امکان استفاده از پارامترهای قراردادی در هر دو (IN , OUT , IN_OUT) وجود دارد.
- در روش‌های برنامه‌نویسی تقسیم‌شده و ساختاریافته از هر دو می‌توان استفاده کرد.
- در FC و FB هر دو می‌توان متغیر TEMP تعریف کرد. این متغیر در L-Stack ذخیره شده و در صورت قطع و وصل تغذیه CPU یا خاموش روشن شدن PLC یا راه‌اندازی مجدد، مقادیر موجود در L Stack پاک می‌شوند. از طرفی با پایان اجرای بلاکی که متغیرهای محلی از نوع TEMP در آن تعریف شده‌اند دسترسی به این متغیرها امکان‌پذیر نمی‌باشد.
- می‌توان فراخوانی FC یا FB را مشروط به انجام عمل خاصی نمود. در زبان LAD/FBD هر دو بلاک FC و FB دارای یک ورودی EN مخفف Enable هستند که می‌توان آنرا به یک کنتاکت یا یک لاجیک متصل کرد. در این شرایط بلاک در صورتی اجرا می‌شود که ورودی EN یک باشد.
- FC و FB در زبان LAD/FBD دارای خروجی ENO هستند که نشان‌دهنده صحت اجرای بلاک است. اگر بلاک با موفقیت اجرا شود، این پارامتر دارای مقدار یک منطقی خواهد شد.
- در فراخوانی چند بلاک از درون یکدیگر، به Nesting Depth یا عمقِ تودرتویی توجه نمایید. تعداد بلاک FC یا FB که به‌صورت تو در تو می‌توان از داخل هم صدا زد به این پارامتر بستگی دارد که مقدار آن در CPUهای مختلف متفاوت است. جدول ۴-۲ این پارامتر را برای برخی CPUهای 300 نشان می‌دهد.

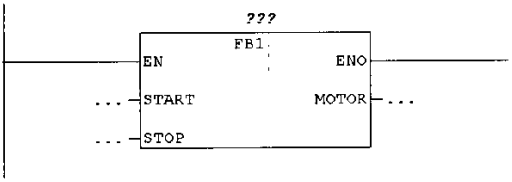
فصل ۴

جدول ۴-۲

Nesting depth	CPU 312	CPU 314	CPU 315-2 DP	CPU 317-2 PN/DP
• per priority class	16	16	16	16
• additional within an error OB	4	4	4	4

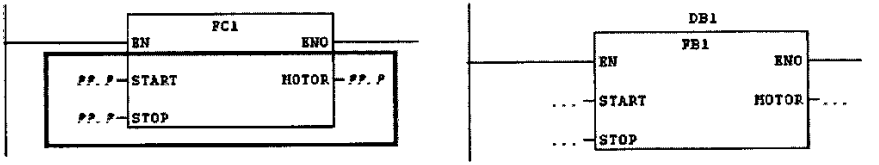
تفاوت‌های FC و FB

- **حافظه:** FC فاقد حافظه است ولی FB دارای حافظه‌ای از جنس DB است که تمام پارامترهای IN و OUT و IN_OUT و متغیرهای STATIC در آن ذخیره می‌شود و حتی با پایان اجرای بلاک باز در دسترس هستند. در ضمن اگر DB روی حافظه فلش ذخیره شود در صورت قطع و وصل تغذیه پارامترها و متغیرهای فوق پاک نمی‌شوند. این امکانات برای FC وجود ندارد.
- **فراخوانی:** در هنگام فراخوانی FB باید شماره DB اختصاصی آنرا نیز ذکر نمود، در غیر اینصورت علامت؟ مانند شکل ۴-۱۹ در بالای FB مورد نظر ظاهر شده و امکان ذخیره‌سازی یا دانلود وجود ندارد، ولی در FC این موضوع وجود ندارد.



شکل ۴-۱۹ عدم اختصاص دیتابلاک به FB

- **اختصاص مقدار اولیه:** در هنگام فراخوانی FC نمی‌توان پایه‌ای را بدون آدرس رها نمود. دلیل این موضوع این است که پارامترهای قراردادی تعریف شده در FC در هیچ حافظه‌ای ذخیره نشده و مقدار اولیه‌ای به‌آنها اختصاص داده نمی‌شود؛ اما از آنجایی که در FB، پارامترهای قراردادی در دیتابلاک اختصاصی FB ذخیره شده، به آنها مقدار اولیه اختصاص داده می‌شود لذا می‌توان در هنگام فراخوانی FB برخی از پارامترها را آدرس‌دهی نکرد.



شکل ۴-۲۰ عدم اختصاص آدرس به پایه‌های FC و FB

همانطور که در شکل ۴-۲۰ مشخص است، در صورتی‌که به پارامترهای FC مقداری اختصاص پیدا نکند علامت ? در پایه مربوطه ظاهر شده و امکان ذخیره‌سازی و دانلود برنامه وجود ندارد، اما عدم اختصاص آدرس به پایه‌های FB اشکالی بوجود نمی‌آورد. در مورد نحوه اختصاص مقدار اولیه به پارامترهای FB در ادامه به تفصیل بحث خواهیم کرد.

- **زبان‌های برنامه‌نویسی:** در FC فقط امکان استفاده از زبان‌های برنامه‌نویسی پایه مثل LAD,FBD و STL وجود دارد، اما در FB علاوه بر زبان‌های پایه امکان برنامه‌نویسی به زبان GRAPH نیز وجود دارد. علاقه‌مندان می‌توانند جهت توضیحات بیشتر به کتاب برنامه‌نویسی PLC به زبان S7-GRAPH تألیف احمد فرجی – علیرضا سینا مراجعه نمایند.

۴-۵ نحوه کار با FC

برای طراحی یک FC و کار با آن مراحل زیر بایستی انجام شود:

۱- ایجاد FC

۲- ایجاد متغیرهای محلی و پارامترهای قراردادی مورد نیاز در FC

۳- نوشتن برنامه در FC

۴- ذخیره‌سازی و دانلود FC

۵- فراخوانی در بلاک ماقبل

نکات قابل توجه

- اگر FCبرای برنامه‌نویسی تقسیم شده به‌کار می‌رود، ممکن است نیازی به تعریف پارامترهای قراردادی (in , out) نداشته باشد. تعریف متغیرهای نوع Temp در آن برحسب نیاز صورت می‌گیرد.
- در برنامه‌نویسی ساختار یافته معمولاً FC دارای ورودی و خروجی (پارامترهای قراردادی) است که قبل از برنامه‌نویسی بایستی تعریف شوند.
- در برنامه‌نویسی FC برای ذخیره‌سازی نتایج میان برنامه، اگر از این نتایج در بیرون از FC استفاده نمی‌شود بهتر است به‌جای M و DB از متغیرهای محلی Temp استفاده شود. در صورت استفاده از M و DB امکان دارد که در این آدرس‌ها بیرون از FC نیز مقداری نوشته شود و موجب گردد که نتایج حاصل از برنامه FC به‌صورت غلط ارائه شود. از اینرو در نوشتن برنامه FC به‌ویژه در برنامه‌نویسی ساختار یافته سعی شود از متغیرهای Global استفاده نشود.

برای تفهیم نکات و مراحل کار با FC موضوع را با ذکر چند مثال دنبال می‌کنیم.

مثال ۴-۱ برنامه‌نویسی تقسیم شده

شرایط فرآیند

- در فرآیند دو ناحیه وجود دارد Area_01 و Area_02.
- در هر ناحیه یک مخزن وجود دارد که سطح سیال موجود درآن بایستی کنترل شود.
- نوع سیال هر مخزن با دیگری متفاوت است، ولی هر دو مخزن ارتفاع یکسان دارند.
- روی هر مخزن یک ترانسمیتر سطح، دو سوئیچ سطح، یک ولو ورودی و یک ولو خروجی وجود دارد که به‌صورت On/Off هستند.
- هر دو ترانسمیتر سیگنال 4-20 mA می‌فرستند و بین 0 تا 5 متر کالیبره شده‌اند.

با شرایط فوق برنامه‌ای بنویسید که:

- وقتی سطح کمتر از ۱ متر است یا سوئیچ سطح پایین فعال است، ولو خروجی بسته و ولو ورودی باز باشد.
- وقتی سطح از ۱متر بیشتر و از ۴ متر کمتر است هر دو ولو باز باشند.
- وقتی سطح از ۴ متر بیشتر یا سوئیچ سطح بالا فعال است، ولو ورودی بسته و ولو خروجی باز باشد.

لیست آدرس‌ها

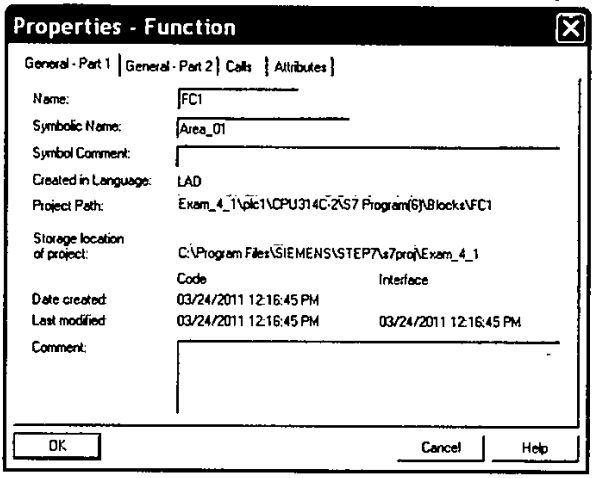
PIW752	Tank1_LT
I124.0	Tank1_LSL
I124.1	Tank1_LSH
Q124.0	Tank1_Inlet_Valve
Q124.1	Tank1_Outlet_Valve
PIW754	Tank2_LT
I125.0	Tank2_LSL
I125.1	Tank2_LSH
Q125.0	Tank2_Inlet_Valve
Q125.1	Tank2_Outlet_Valve

فصل ۴

حل: ابتدا پیکربندی سخت افزار و تعریف آدرس‌ها به‌صورت سمبلیک را انجام داده و پیکربندی را ذخیره و دانلود می‌کنیم. سپس مراحل زیر را به‌ترتیب انجام می‌دهیم.

۱- ایجاد FC1 برای Area_1

مانند شکل ۴-۲۱ فانکشن FC1 را ایجاد و نام سمبلیک Area_1 را به آن اختصاص می‌دهیم.

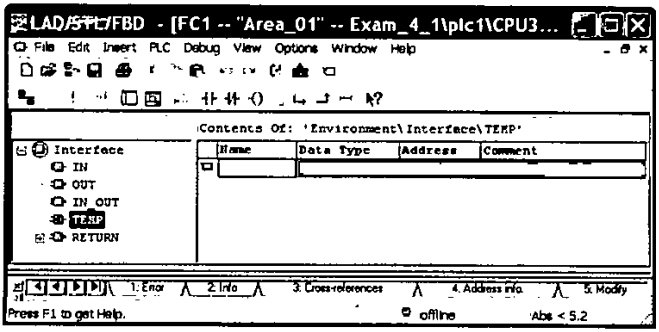


شکل ۴-۲۱ ایجاد FC1 برای مثال ۴-۱

۲- باز کردن FC1 و تعریف متغیرهای Temp مورد نیاز

در برنامه‌نویسی تقسیم شده از آنجا که در خود FC از آدرس‌های فیزیکی واقعی استفاده می‌کنیم، نیاز به تعریف پارامترهای IN و OUT نداریم، ولی پارامترهای Temp را متناسب با نیاز تعریف می‌کنیم.

چون در ابتدا نیازها دقیقاً مشخص نیست در حین برنامه‌نویسی هرجا لازم شد به‌جای M و DB از Temp استفاده کنیم در همانجا آنرا تعریف خواهیم کرد.

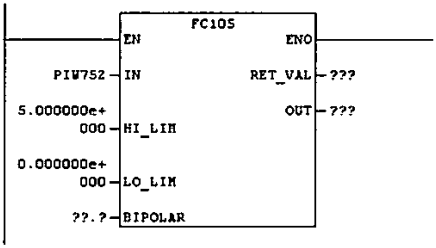


شکل ۴-۲۲ متغیر TEMP در FC1

۳- برنامه‌نویسی

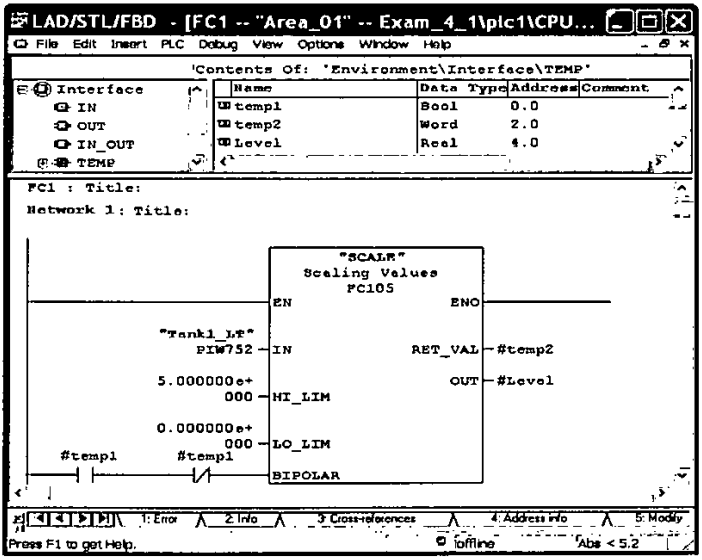
ابتدا لازم است سیگنال ترانسمیتر سطح را Scale کرده پس به FC105 نیاز داریم. پس از صدا زدن FC105 مانند شکل ۴-۲۳ آدرس‌ها و مقادیری که ثابت هستند را وارد می‌کنیم.

```
FC1 : Title:
Network 1 : Title:
```



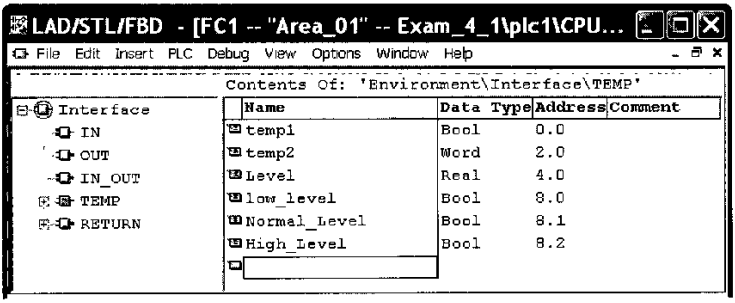
شکل ۴-۲۳ صدا زدن فانکشن Scale در FC1

سایر آدرس‌هایی که با علامت سوال ظاهر شوند لازم است به متغیرهایی اختصاص داده شوند. در اینجا به‌جای استفاده از متغیرهای سراسری از متغیرهای محلی استفاده می‌کنیم. همانطور که می‌بینیم نیاز به سه متغیر که به‌ترتیب از جنس Bool و Word و Real باشند داریم. این متغیرها را مانند شکل ۴-۲۴ به‌صورت Temp تعریف کرده و به پایه‌های FC105 اختصاص می‌دهیم.



شکل ۴-۲۴ اختصاص متغیرهای Temp به FC105

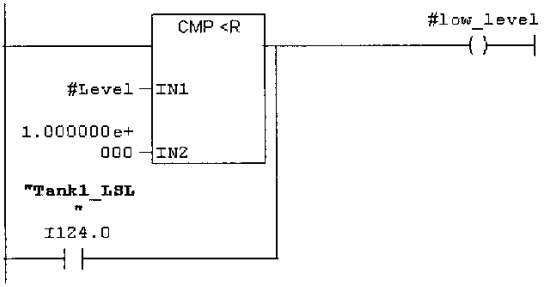
اکنون می‌توانیم با استفاده از مقدار متغیر Level ادامه برنامه کنترل را به‌صورت زیر بنویسیم. ابتدا براساس مقدار Level و سوئیچ‌های سطح سیگنال‌های سطح پایین، سطح نرمال و سطح بالا را ایجاد می‌کنیم. این سیگنال‌ها نیز به‌صورت Temp از جنس Bool تعریف می‌شوند، بنابراین جدول متغیرهای محلی به‌صورت زیر کامل می‌شود.



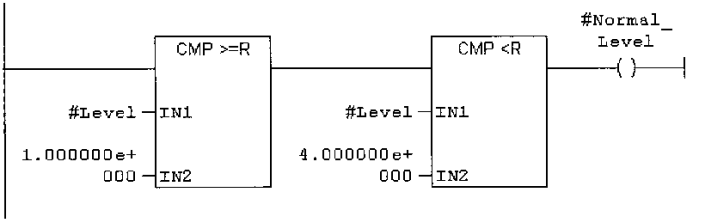
شکل ۴-۲۵ تعریف کامل متغیرهای TEMP مورد نیاز

تا این مرحله برنامه به‌صورت زیر است:

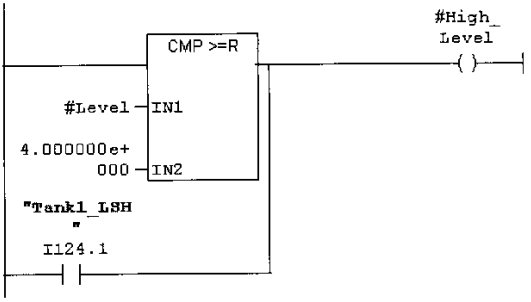
Network 2: Title:



Network 3: Title:



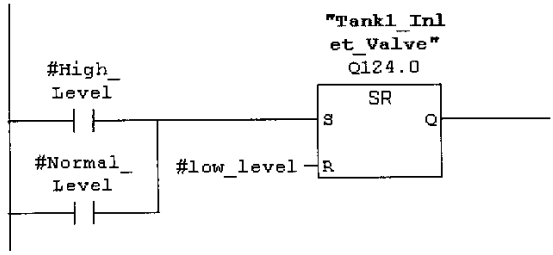
Network 4 : Title:



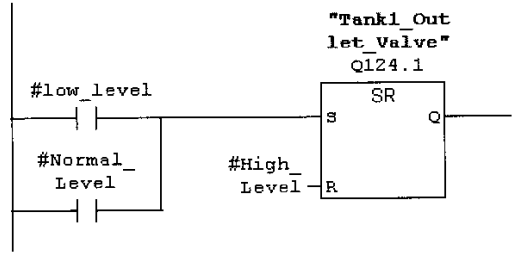
شکل ۴-۲۶ بخش اول برنامه فانکشن FC1 مثال ۴-۱

اکنون با استفاده از سیگنال‌های ساخته شده، ولو ورودی و خروجی را با برنامه زیر کنترل می‌کنیم.

Network 5 : Title:



Network 6 : Title:



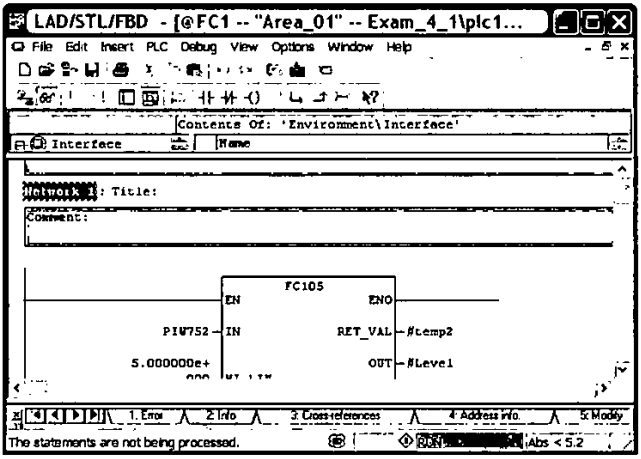
شکل ۴-۲۷ ادامه برنامه فانکشن FC1 مثال ۴-۱

۴- ذخیره سازی FC و دانلود

در این مرحله برنامه FC1 کامل شده است؛ آنرا ذخیره و دانلود می‌کنیم.

نکات قابل توجه

- اگر در این مرحله دانلود انجام نشود، می‌توان این کار را بعداً پس از تکمیل OB1 انجام داد یعنی FC1 و OB1 را همزمان دانلود نمود. در هر صورت بایستی توجه داشت که اگر FC دانلود نشود، پس از فراخوانی آن در OB1 در پردازش CPU مشکل پیش می‌آید که می‌تواند منجر به توقف CPU گردد.
- اگر FC دانلود شود ولی در OB صدا زده نشود، هیچگاه CPU آن را اجرا نخواهد کرد زیرا فقط بلاک‌های OB توسط سیستم عامل CPU اجرا می‌گردند. بنابراین در این شرایط اگر FC را باز کرده و Monitor کنیم، نوار سبز رنگ پایین پنجره ثابت است که نشان دهنده اجرا نشدن این بلاک می‌باشد.



شکل ۴-۲۸ فانکشن در حالت Online قبل از صدا زدن در OB

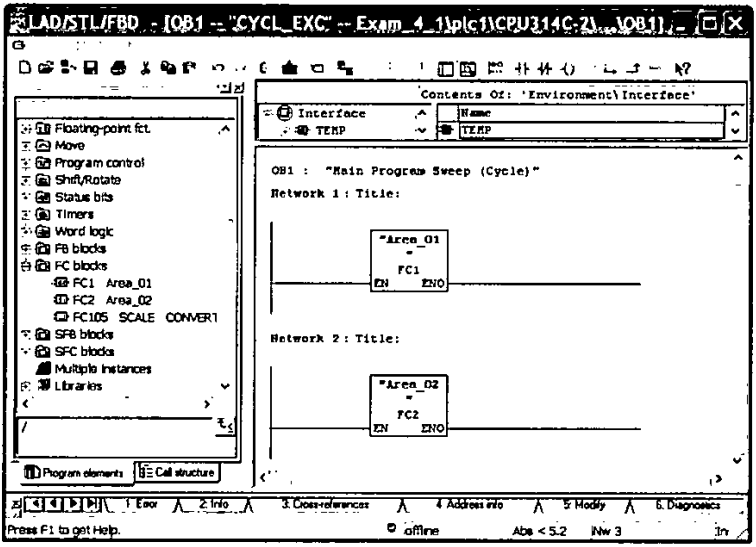
۵- تکرار مراحل ۱ تا ۴ برای FC2

FC2 را با نام سمبلیک Area_02 ایجاد کرده و مشابه FC1 نسبت به برنامه‌نویسی آن اقدام می‌کنیم. لازم است توجه شود که:

- منطق برنامه FC2 مشابه FC1 می‌باشد.
- آدرس‌های فیزیکی با توجه به سیگنال‌های مخزن ۲ وارد می‌شود.
- متغیرهای Temp مشابه FC1 است حتی می‌تواند نام آنها نیز مشابه FC1 باشد.

۶- فراخوانی FC1 و FC2 از بلاک ماقبل

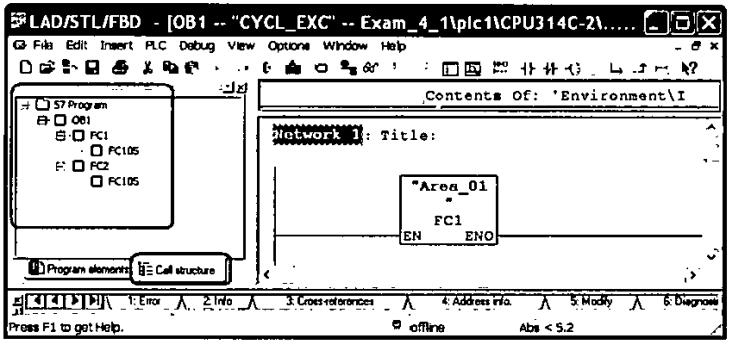
پس از اینکه FC1 و FC2 ذخیره و دانلود شدند آنها را می‌بندیم. OB1 را باز کرده و مانند شکل ۴-۲۹ FC1 و FC2 را از زیر مجموعه FC Blocks صدا می‌زنیم.



شکل ۴-۲۹ صدا زدن فانکشن‌ها از OB1

همانطور که دیده می‌شود، فانکشن‌ها دارای ورودی و خروجی نیستند و فقط برای تفکیک‌سازی دو ناحیه فرآیند از آنها استفاده شده است.

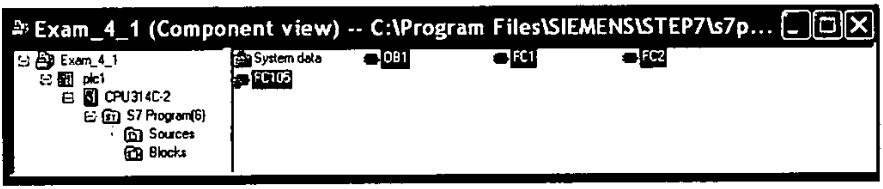
پس از ذخیره‌سازی OB1 بایستی آنرا دانلود کنیم. اگر PLC در مد RUN است، دانلود با دقت و احتیاط صورت گیرد. اگر دانلود برنامه به‌طور صحیح انجام شود، منجر به توقف نخواهد شد. در واقع سیستم عامل سیکل جدید را با OB1 جدید شروع خواهد کرد. برای دانلود OB1 اگر از محیط برنامه‌نویسی اقدام کنیم بایستی اطمینان داشته باشیم که بلاک‌های فراخوان شده قبلاً دانلود شده باشند. برای مشاهده بلاک‌های فراخوان شده روی Call Structure در پایین پنجره مانند شکل ۴-۳۰ کلیک می‌کنیم. همانطور که دیده می‌شود FC1 و FC2 و FC105 بلاک‌هایی هستند که فراخوان شده و بایستی قبلا دانلود شده باشند.



شکل ۴-۳۰ سلسه مراتب فراخوانی بلاک‌های مثال ۴-۱

راه بهتر برای دانلود کردن این است که OB1 و سایر بلاک‌ها را ببندیم و به Simatic Manager باز گردیم و از پوشه Blocks همه بلاک‌ها را بجز System data انتخاب کرده و همه را یکجا دانلود کنیم. در این روش مشکل قبلی پیش نمی‌آید ولی بایستی توجه داشت که:

- System data انتخاب و دانلود نشود زیرا این پوشه حاوی اطلاعات سخت‌افزاری PLC است و دانلود آن در حین کار منجر به توقف CPU خواهد شد.
- بلاک‌های برنامه‌نویسی باز نباشند. اگر بلاکی باز و هنوز ذخیره نشده باشد، در این روش بلاک قدیمی به PLC دانلود می‌گردد.
- بلاک‌های غیر مجاز (مانند OB2 و ...) در پوشه Blocks وجود نداشته باشد.

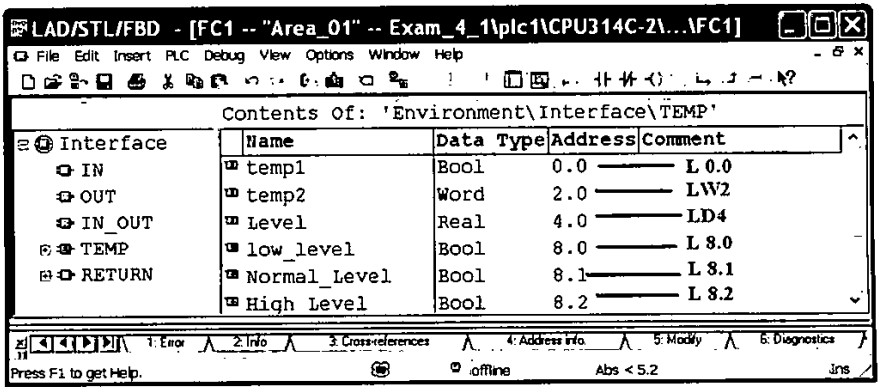


شکل ۴-۳۱ دانلود یکباره همه بلاک‌ها

پس از دانلود، پروژه فوق را با PLC یا توسط سیمولاتور تست کنید.

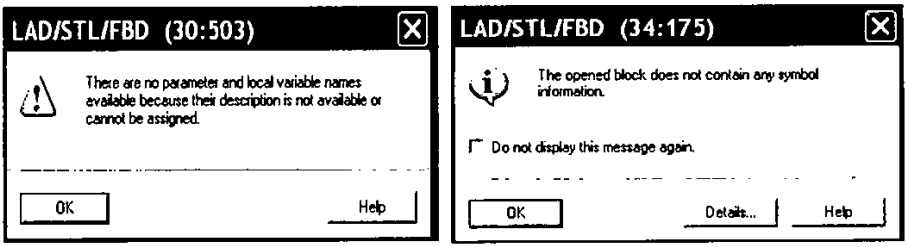
نکات تکمیلی مثال ۴-۱

نکته ۱: متغیرهای محلی در فانکشن‌ها همگی دارای یک آدرس هستند. این آدرس‌ها با حرف L شروع می‌شوند و می‌توان در برنامه به‌جای استفاده از اسامی از آدرس‌ها استفاده نمود. به شکل ۴-۳۲ دقت شود.



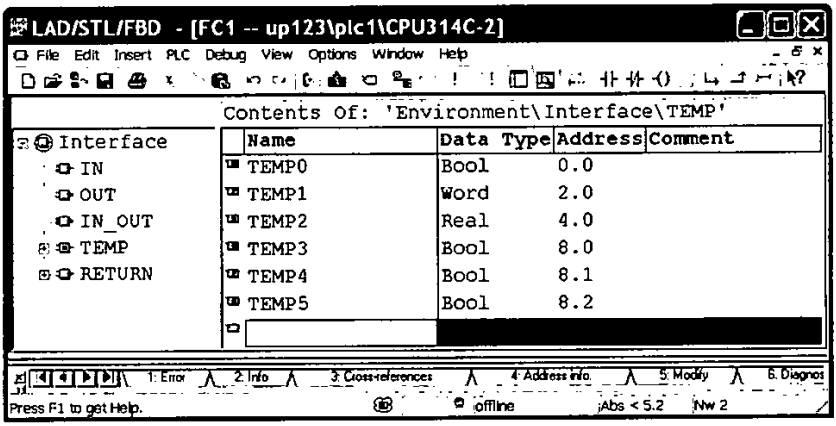
شکل ۴-۳۲ آدرس متغیرهای محلی

نکته ۲: پس از دانلود و تست، پروژه جدیدی ایجاد کنید و از PLC کل اطلاعات را آپلود کنید. اگر پس از آپلود برنامه یکی از فانکشن‌ها را باز کنیم با دو اخطار متوالی زیر مواجه می‌شویم.



شکل ۴-۳۳ پیغام‌های سیستم در زمان آپلود کردن فانکشن

این اخطارها می‌گوید که اسامی متغیرهای محلی و متغیرهای سراسری در دسترس نیستند. از اینرو اگر فانکشن‌ها را باز کنیم مانند شکل ۴-۳۴ می‌بینیم که همه اسامی متغیرهای محلی با اسامی دیگری که با حروف TEMP شروع می‌شوند جایگزین شده‌اند.



شکل ۴-۳۴ اسامی متغیرهای TEMP پس از آپلود

تمرین ۴-۲: در فانکشن‌های به‌کار رفته در برنامه مثال ۴-۱ خروجی فیلیپ فلاپ‌ها را به آدرس‌های Q متصل کنید و برای آدرس‌های بالای فیلیپ‌فلاپ‌ها از متغیرهای محلی Temp استفاده کنید. آیا در نتایج پردازش برنامه مشکلی وجود دارد؟

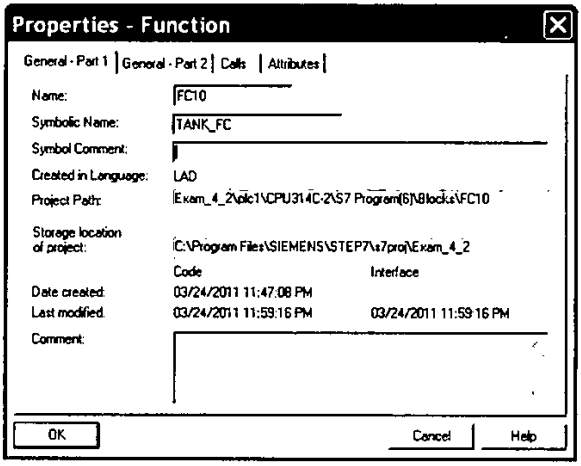
مثال ۴- ۲: برنامه‌نویسی ساختار یافته با FC

اگر بخواهیم برنامه مثال ۴-۱ را به‌صورت ساختار یافته طراحی کنیم، لازم است تغییراتی به‌صورت زیر اعمال نماییم.

ایجاد یک فانکشن که بتواند برای هر مخزن، ورودی‌ها را بگیرد و نتایج خروجی را برگرداند. ورودی‌ها عبارتند از آدرس ترانسمیتر، آدرس سوئیچ‌های سطح و خروجی‌ها عبارتند از آدرس ولوهای ورودی و خروجی.

مراحل کار

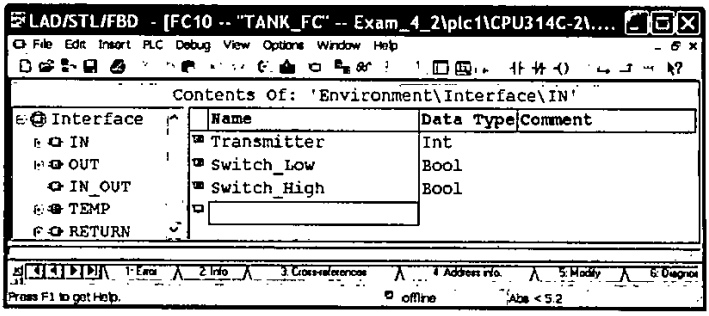
۱- ایجاد FC10 با نام سمبلیک TANK_FC



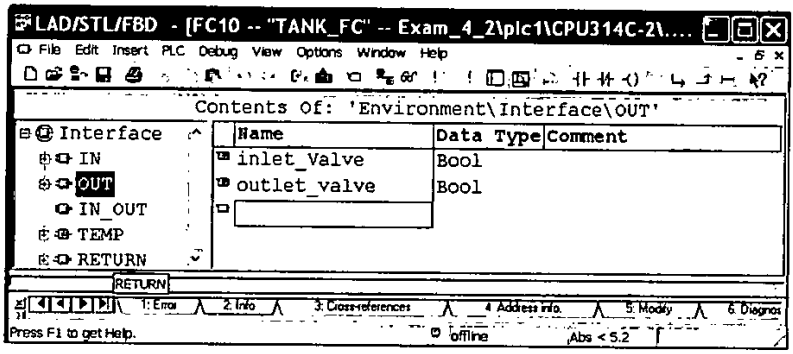
شکل ۴-۳۵ ایجاد فانکشن مثال ۴-۲

۲- تعریف پارامترهای ورودی خروجی FC10

شکل‌های زیر پارامترهای ورودی و خروجی را در قسمت Declation Section نشان می‌دهد. این پارامترها برخلاف نوع TEMP دارای آدرس نیستند و فضایی را در L-Stack اشغال نمی‌کنند.



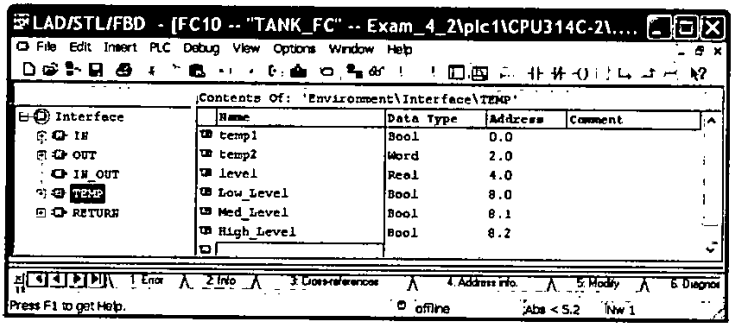
شکل ۴-۳۶ پارامترهای ورودی فانکشن مثال ۴-۲



شکل ۴-۳۷ پارامترهای خروجی فانکشن مثال ۴-۲

۳- تعریف پارامترهای TEMP

پارامترهای TEMP مثال ۴-۱ در اینجا نیز استفاده می‌شوند، پس به همان روش تعریف می‌گردند.



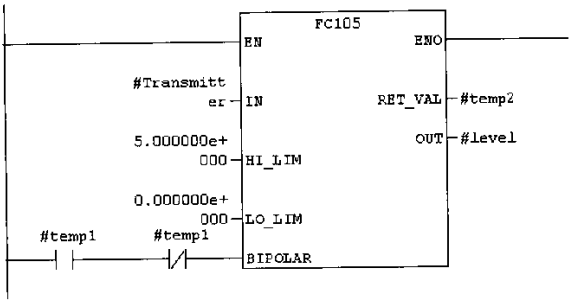
شکل ۴-۳۸ متغیرهای TEMP فانکشن مثال ۴-۲

۴- برنامه‌نویسی

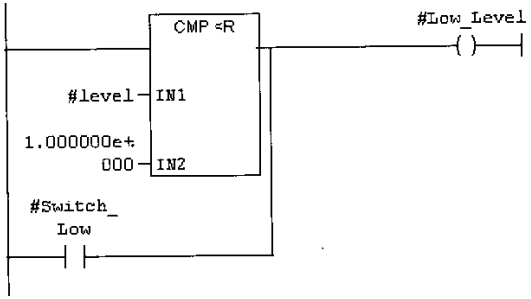
برنامه قبلی به‌صورت زیر اصلاح می‌گردد. تفاوت فقط در این است که به‌جای آدرس‌های واقعی در برنامه از متغیرهای IN و OUT استفاده شده است.

FC10 : Title:

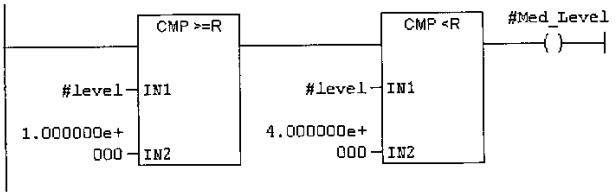
Network 1 : Title:



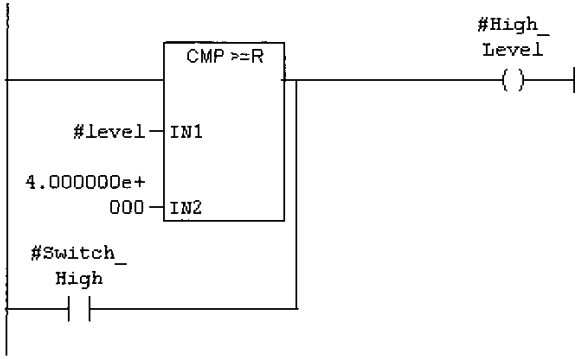
Network 2 : Title:



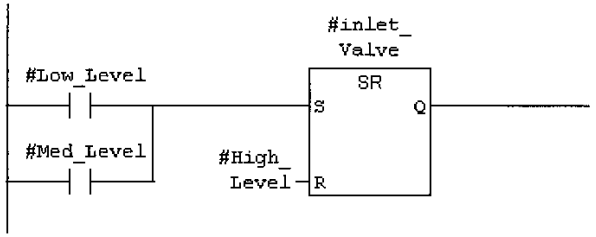
Network 3 : Title:



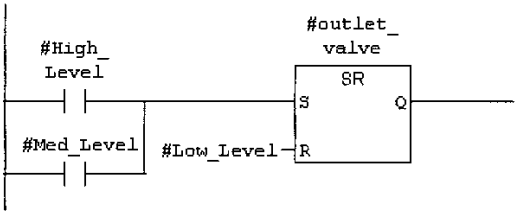
Network 4 : Title:



Network 5 : Title:



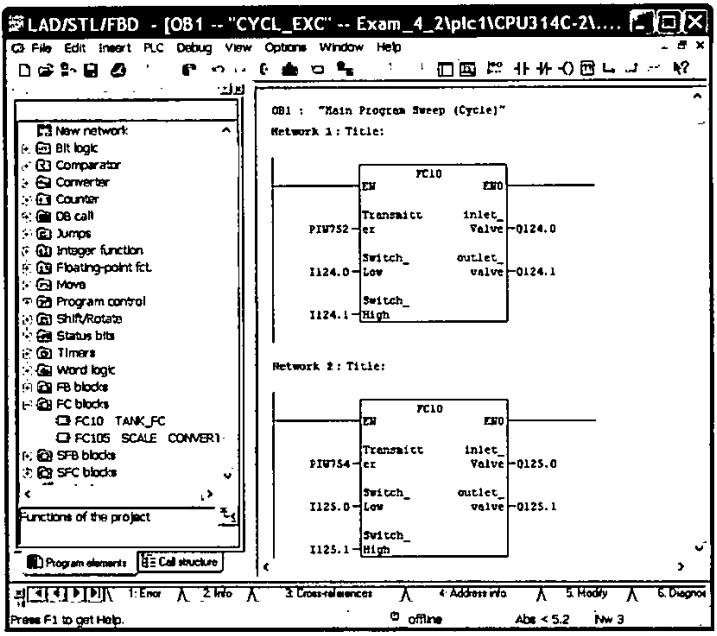
Network 6 : Title:



شکل ۴-۳۹ برنامه فانکشن مثال ۴-۲

۵- صدا زدن از بلاک ماقبل

OB1 را ایجاد کرده و دوبار فانکشن FC10 را صدا می‌زنیم. در مثال ۴-۱ برای هر ناحیه یک فانکشن جداگانه تعریف شده بود ولی در اینجا یک فانکشن وجود دارد که برای هر ناحیه یک بار صدا زده می‌شود و آدرس‌های مربوط به آن ناحیه داده می‌شود. به شکل ۴-۴۰ توجه کنید.



شکل ۴-۴۰ فراخوانی فانکشن در OB1 مربوط به مثال ۴-۲

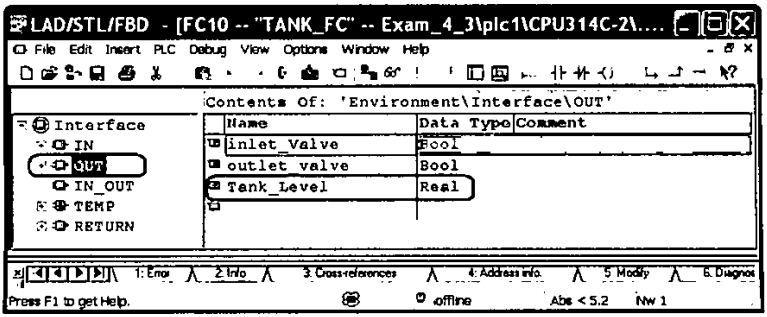
۶- ذخیره‌سازی و دانلود بلاک‌های OB1 , FC10 , FC105 و تست برنامه

مثال ۴-۳: تغییر در ساختار FC در حین کار

برنامه ۴-۲ را به‌گونه ای تغییر دهید که مقدار سطح هر مخزن نیز در خروجی فانکشن FC10 نشان داده شود.

در این مثال هدف آن است که با انجام تغییر در ساختار FC در حین کار آشنا شوید. بنابراین برنامه مثال ۴-۲ را باز کرده و تغییرات لازم را اعمال می‌کنیم.

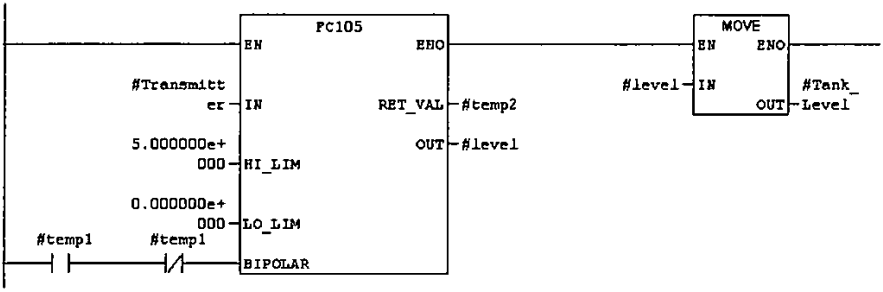
با باز کردن FC10 خروجی جدیدی در بخش Output با نام Tank_Level از جنس Real تعریف می‌نماییم.



شکل ۴-۴۱ پارامترهای خروجی فانکشن مثال ۴-۳

پس از تعریف خروجی فوق در برنامه FC10 مقدار سطح را به این خروجی انتقال می‌دهیم. به شکل ۴-۴۲ توجه کنید.

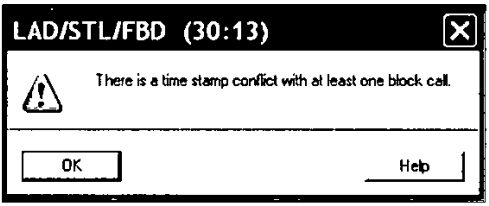
```
FC10 : Title:
Network 1 : Title:
```



شکل ۴-۴۲ برنامه فانکشن مثال ۴-۳

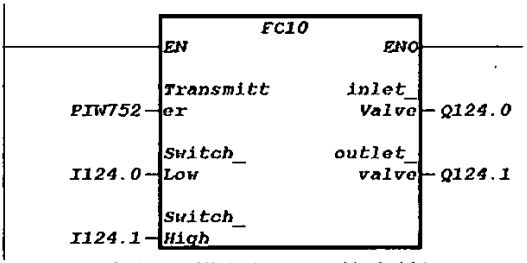
نکته مهم: پس از ذخیره سازی FC10 اگر آن را دانلود کنیم مشکلی برای PLC پیش نخواهد آمد. لازم است خواننده این مطلب را به‌خاطر بسپارد تا در بحث FB بتواند این مقایسه را انجام دهد.

با ذخیره‌سازی FC10 اگر بلاک OB1 را باز کنیم با اخطار زیر مواجه می‌شویم.



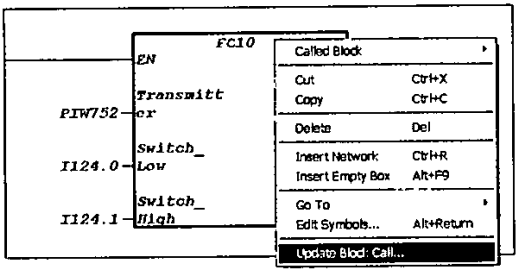
شکل ۴-۴۴ اخطار مربوط به تغییر در ساختار فانکشن

این اخطار معرف آن است که بلاک FC10 فراخوان شده در OB1 با بلاک FC10 فعلی متفاوت است. با کلیک روی OK برنامه به‌صورت زیر ظاهر می‌شود. همانطور که دیده می‌شود بلاک‌های FC10 هر دو به رنگ قرمز نمایش داده شده‌اند.



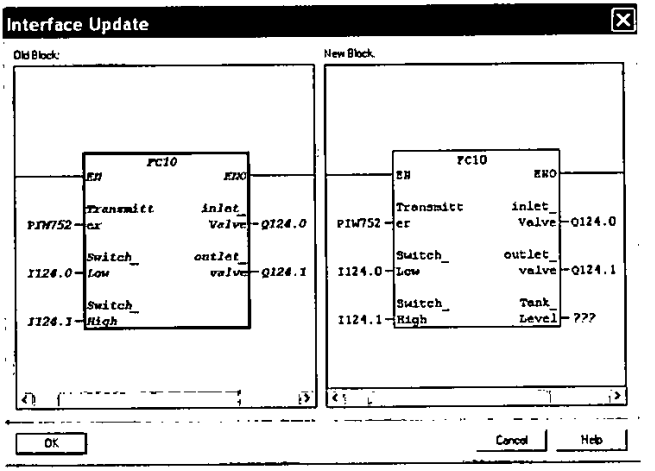
شکل ۴-۴۵ تغییر در ساختار فانکشن مثال ۴-۳

در این شرایط دیگر امکان ذخیره‌سازی و دانلود OB1 را نداریم و لازم است ابتدا هر دو FC10 موجود در آن را به FC10 جدید Update کنیم. کافیست روی FC10 کلیک راست کرده و گزینه Update Block Call را انتخاب کنیم.



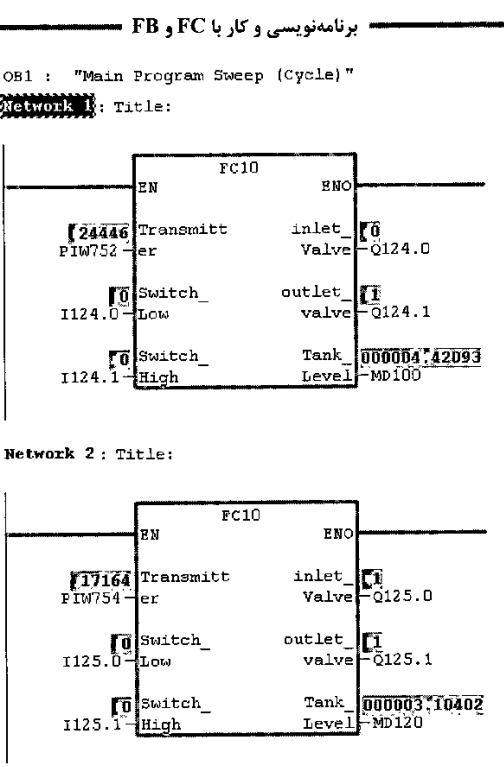
شکل ۴-۴۶ نحوه رفع اشکال فانکشن مثال ۴-۳

با کلیک روی Update Block Call پنجره زیر ظاهر می‌شود که در سمت چپ بلاک قدیمی و در سمت راست بلاک جدید دیده می‌شود. همانطور که مشخص است بلاک جدید خروجی Tank_Level را داراست.



شکل ۴-۴۷ Update کردن فانکشن مثال ۴-۳

با اتمام Update خروجی‌های Tank_Level را به دو آدرس حافظه متصل می‌کنیم. برنامه را ذخیره و دانلود و تست می‌نماییم.



شکل ۴-۴۸ فانکشن‌های مثال ۴-۳ در حالت Online

فصل ۴

مثال ۴-۴: استفاده از پارامتر IN_OUT

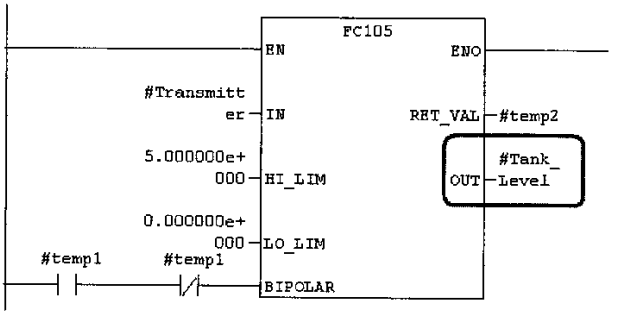
برنامه ۴-۳ را به‌گونه‌ای تغییر دهید که در FC10 خروجی FC105 مستقیماً به Tank_Level متصل گردد و نیازی به استفاده از متغیر Level که از جنس TEMP نباشد.

این برنامه اگر چه ساده به‌نظر می‌رسد ولی در صورت حذف Temp با مشکلی مواجه می‌شویم.

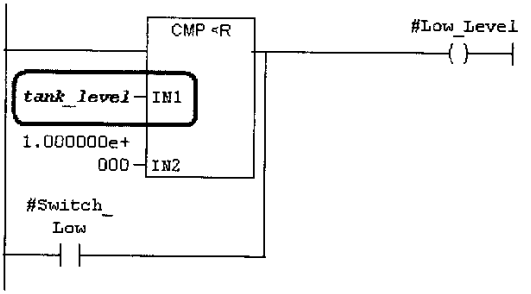
در Network1 مانند شکل ۴-۴۹ مشکلی نیست و خروجی TANK_Level به OUT متصل می‌شود ولی در Network2 و بعدی که لازم است مقدار سطح به مقایسه‌گر داده شود، مقایسه‌گر TANK_LEVEL را به‌عنوان ورودی قبول نمی‌کند و مانند شکل ۴-۴۹ به رنگ قرمز در می‌آید.

FC10 : Title:

Network 1 : Title:



Network 2 : Title:

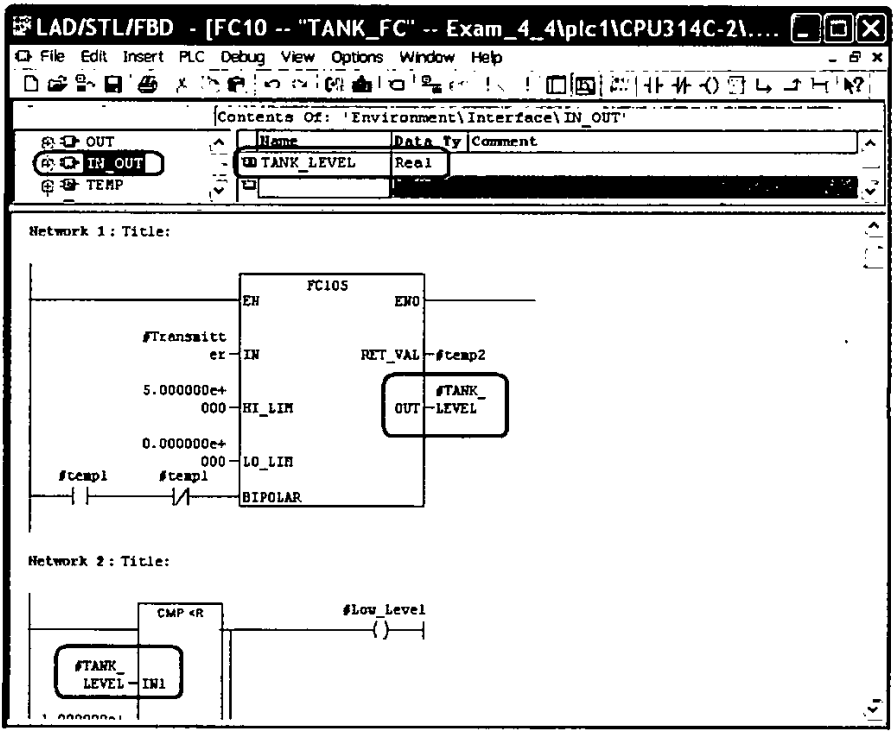


شکل ۴-۴۹ برنامه فانکشن مثال ۴-۴

علت این است که پارامترهای تعریف شده به‌عنوان OUT را نمی‌توان به ورودی IN المان‌های برنامه‌نویسی اختصاص داد. این مشکل برای متغیرهای BOOL وجود ندارد. در ضمن در برنامه‌نویسی به زبان STL این مشکل به‌طور کلی وجود ندارد و پارامترهای OUT همگی قابل خواندن هستند.

برای رفع مشکل فوق لازم است متغیر TANK_LEVEL را به‌صورت IN_OUT تعریف کنیم.

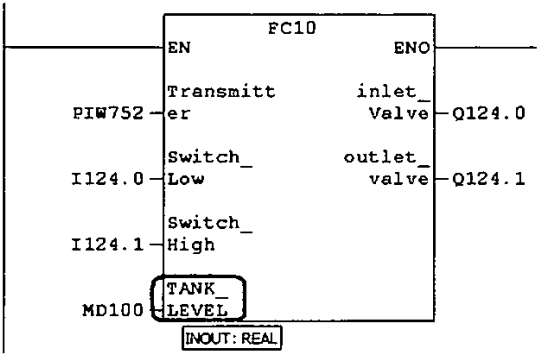
برنامه FC10 به‌صورت زیر در می‌آید. همانطور که دیده می‌شود این پارامتر را هم در خروجی و هم در ورودی المان‌های برنامه می‌توان استفاده کرد. بدین ترتیب با حذف یک متغیر TEMP (منظور متغیر Level است که از جنس Real تعریف شده بود) به اندازه ۴ بایت از فضای L-Stack آزاد می‌شود.



شکل ۴-۵۰ استفاده از متغیر IN_OUT در فانکشن مثال ۴-۴

فصل ۴

در OB1 پس از Update کردن FC10 برنامه به‌صورت زیر خواهد بود. همانطور که دیده می‌شود، پارامترهای IN_OUT شبیه IN در ورودی ظاهر می‌شوند.



شکل ۴-۵۱ فراخوانی فانکشن مثال ۴-۴

نکته: در ورودی فانکشن‌ها هر دو نوع پارامتر IN و IN_OUT ظاهر می‌شوند. اگر ماوس را کمی روی ورودی مورد نظر نگه داریم مانند شکل فوق مشخص می‌شود که پارامتر از چه نوع و از چه جنسی است.

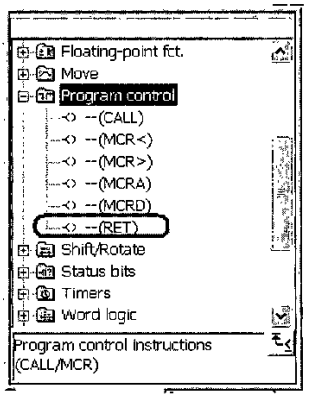
تمرین ۴-۳: اگر در برنامه قبلی دو مخزن با یکدیگر متفاوت و به‌صورت زیر باشند، اصلاحات لازم را در فانکشن اعمال کرده و برنامه را کامل کنید:

مخزن ۱	مخزن ۲
ارتفاع: ۳ متر	ارتفاع: ۵ متر
حد پایین ارتفاع: ۰٫۵ متر	حدپایین ارتفاع: ۱ متر
حد بالای ارتفاع: ۳٫۵ متر	حد بالای ارتفاع: ۴ متر

مثال ۴-۵: استفاده از دستور RET در برنامه

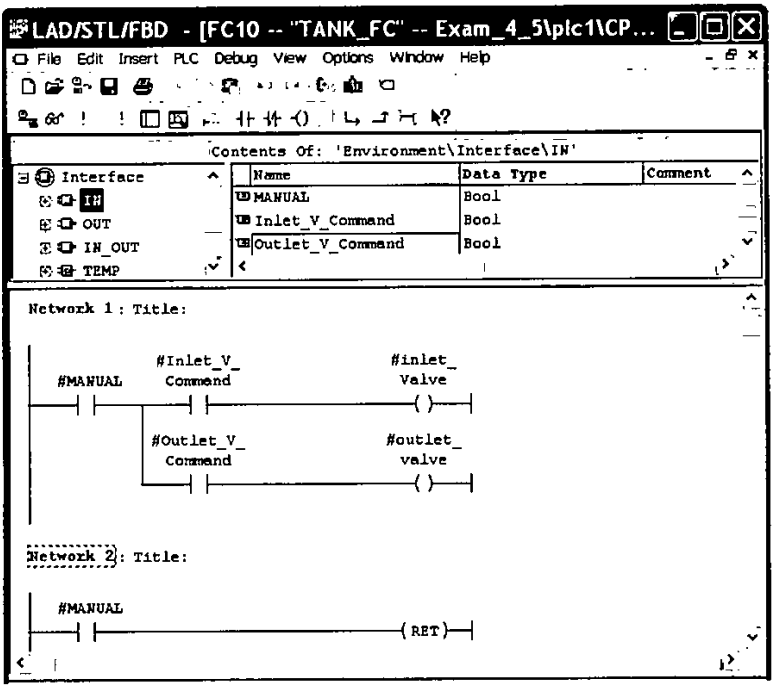
فرض کنید برای هر ناحیه از فرآیند یک کلید دو حالته AUTO/MANUAL وجود دارد. در حالت AUTO ولوهای ورودی و خروجی از منطق برنامه فرمان می‌گیرند و در حالت Manual توسط کلیدهای Open/Close باز و بسته می‌شوند. این موارد نیز به‌عنوان ورودی به فانکشن داده می‌شوند. برنامه لازم را بنویسید.

در اینجا هدف آن است که خواننده با دستور RET از پنجره Program Element آشنا شود. این دستور که در زیر مجموعه Program Control مانند شکل ۴-۵۲ قرار دارد، اگر در فانکشنی استفاده شود و شرط قبل از آن برآورده شود موجب می‌شود تا پردازش فانکشن قطع شده و به بلاک ماقبل برگردد.



شکل ۴-۵۲ دستور RET در کاتالوگ برنامه

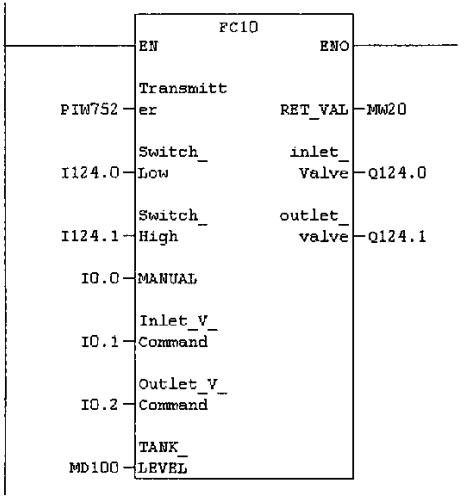
برای حل این برنامه، مانند شکل ۴-۵۳ سه ورودی جدید به FC10 قبلی اضافه می‌کنیم. سپس برنامه‌ای که در Network1 و Network2 مشاهده می‌کنید به ابتدای فانکشن اضافه می‌کنیم و برنامه-های قبلی را از Network3 به بعد قرار می‌دهیم. در صورت یک شده کلید Manual فرمان به‌صورت دستی به ولوها اعمال می‌شود و Network2 باعث می‌شود که پردازش فانکشن قطع شده و به OB1 بازگشت کند.



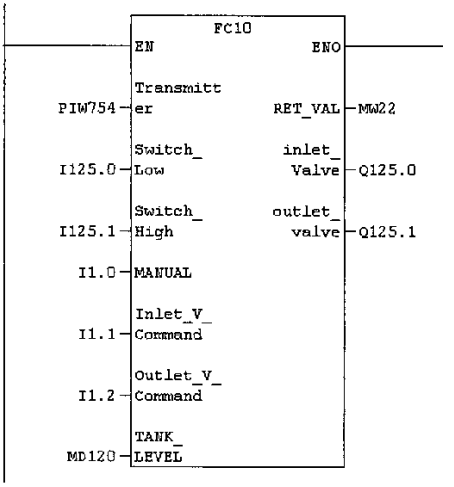
شکل ۴-۵۳ برنامه مثال ۴-۵

در OB1 بلاک FC10 به‌صورت زیر در می‌آید که کلیدهای دستی و فرمان ولوها را به ورودی‌های مربوطه اختصاص می‌دهیم. برنامه را به PLC یا سیمولاتور دانلود و تست کنید.

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1 : Title:
```




```
Network 2 : Title:
```

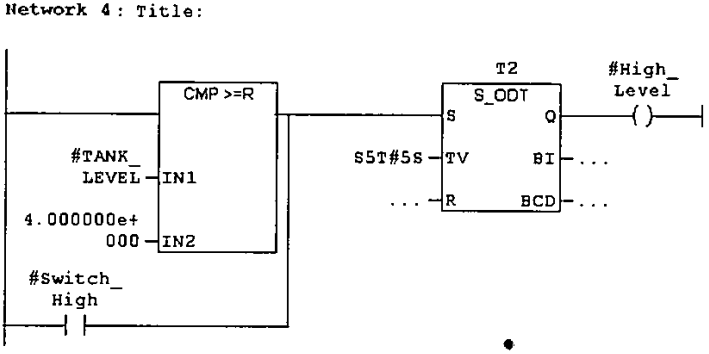


شکل ۴-۵۴ صدا زدن فانکشن FC10 در OB1 مربوط به مثال ۴-۵

مثال ۴-۶: استفاده از تایمر در فانکشن

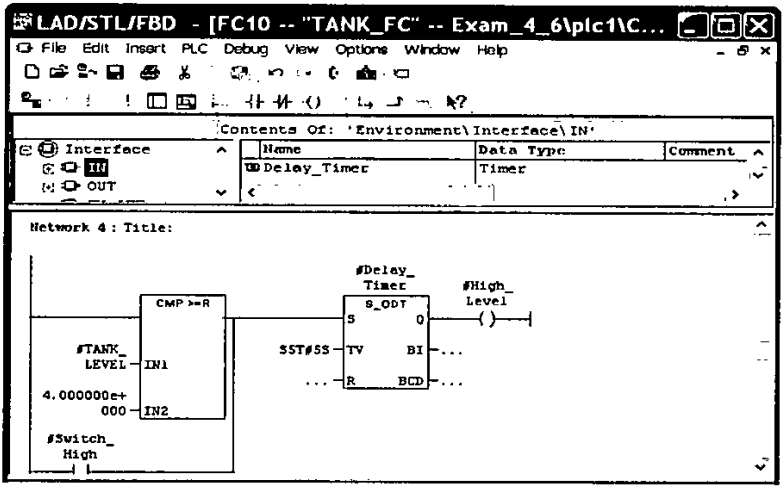
فرض کنید در مثال ۴-۴ لازم است که فرمان به ولو ورودی با ۵ ثانیه تأخیر اعمال شود تا نوسانات سطح سیال منجر به فرمان‌های قطع و وصل ولو نشود. برای این منظور اگر مانند شکل ۴- ۵۵ در برنامه‌ی FC10 تایمر به‌کار ببریم، درست

عمل نمی‌کند. به‌عنوان مثال وقتی سطح مخزن اول از High بالاتر ولی سطح مخزن دوم در ناحیه نرمال قرار دارد، در هر دو مخزن ولوهای ورودی وصل هستند در حالی که برای مخزن اول ولو ورودی بایستی بسته شود. علت این است که در هر بار که OB1 اجرا می‌شود تایمر T2 دوبار فرمان می‌گیرد. یک بار برای مخزن ۱ که شرط برقرار است و تایمر به‌کار می‌افتد ولی وقتی FC10 برای مخزن ۲ فراخوان می‌شود شرط برقرار نیست و ورودی تایمر صفر می‌شود و تایمر نمی‌تواند به‌کار خود ادامه دهد.



شکل ۴-۵۵ برنامه مثال ۴-۶

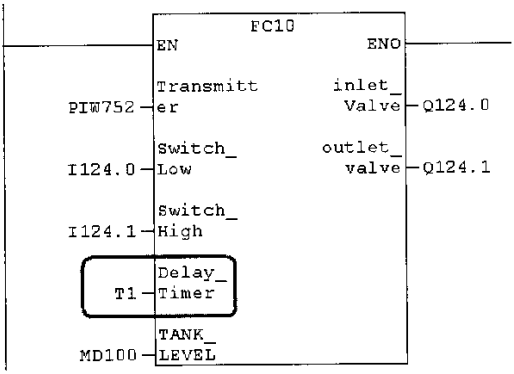
به‌نظر می‌رسد که برای رفع این مشکل می‌توان در ورودی FC10 پارامتر ورودی از نوع تایمر تعریف کرد که از بیرون شماره تایمر دلخواه و متفاوت به آن داده شود. پس در FC10 یک ورودی از جنس Timer مانند شکل ۴-۵۶ تعریف و استفاده می‌کنیم.



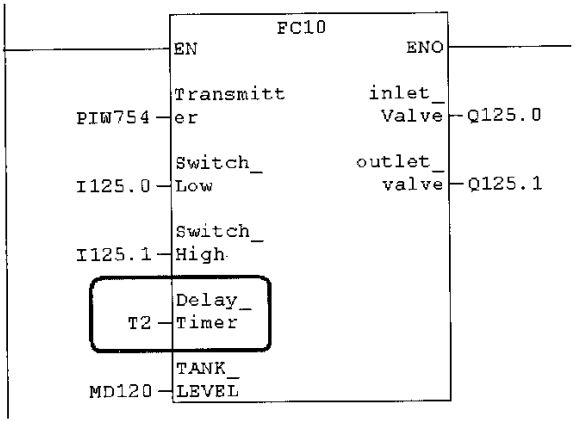
شکل ۴-۵۶ تعریف پارامتر ورودی از جنس تایمر در مثال ۴-۶

با ذخیره سازی FC10 و صدا زدن آن در OB1 می‌توان برای هر FC شماره تایمر متفاوتی را اختصاص داد.

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
```




```
Network 2: Title:
```



شکل ۴-۵۷ فراخوانی فانکشن در برنامه مثال ۴-۶

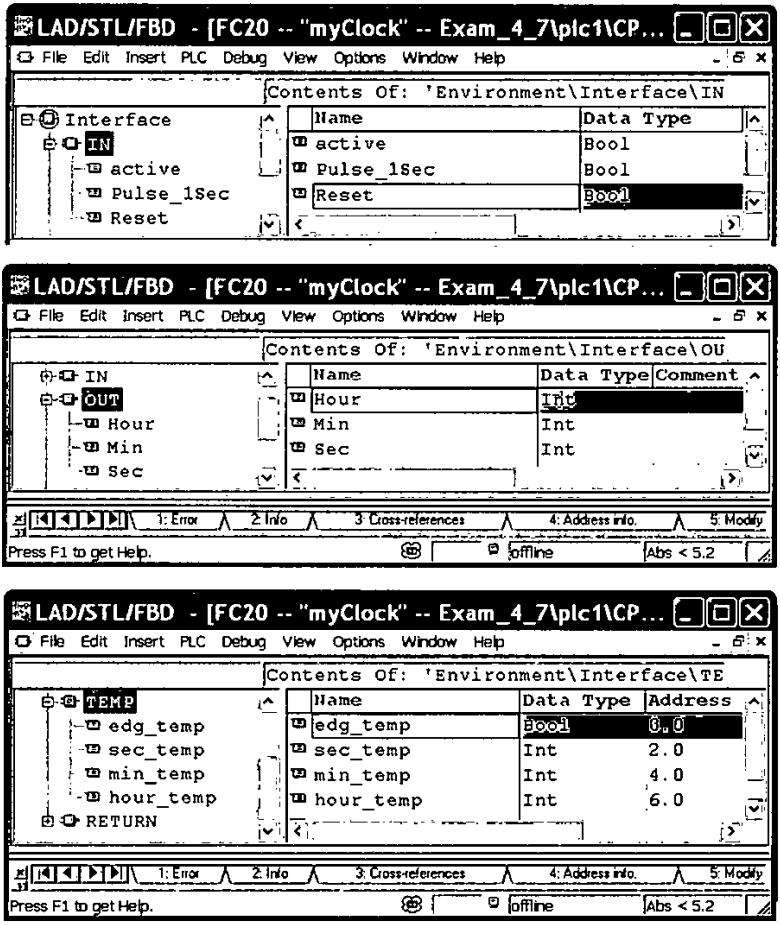
برنامه را ذخیره، دانلود و تست کنید. خواهید دید که باز مشکل قبلی برطرف نشده است و در برخی شرایط پاسخ درست نیست. علت اصلی این مشکل این است که FC فاقد حافظه است و نمی‌تواند وضعیت تایمر را برای ارسال فرمان در مدت تعیین شده در حافظه‌ای نگه دارد. در بحث FB خواهیم دید که به‌دلیل وجود حافظه این مشکل وجود ندارد.

مثال ۴-۷: فانکشن زمان سنج

فانکشنی طراحی کنید که با استفاده از پالس CPU پس از فعال شدن ورودی آن زمان سپری شده را بر حسب ساعت و دقیقه و ثانیه به‌صورت تفکیک شده در خروجی نشان دهد.

حل: ابتدا CPU Clock را در پارامترهای CPU مطابق توضیحاتی که در کتاب مقدماتی ذکر شد فعال کرده و به MB5 اختصاص می‌دهیم. توجه شود که بیت M5.5 پالس یک ثانیه خواهد بود که در برنامه از آن استفاده می‌کنیم.

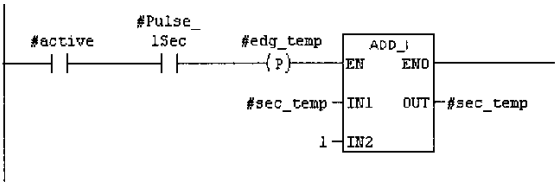
FC20 را با نام myClock ایجاد کرده و ورودی و خروجی و متغیرهای Temp را مانند شکل ۴-۵۸ تعریف می‌نماییم.



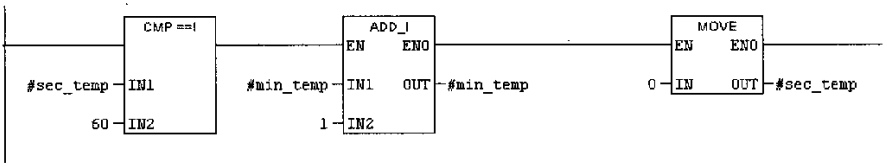
شکل ۴-۵۸ تعریف متغیرها و پارامترهای مثال ۴-۷

برنامه نوشته شده در فانکشن به‌صورت زیر است. آنالیز این برنامه به عهده خواننده واگذار می‌شود.

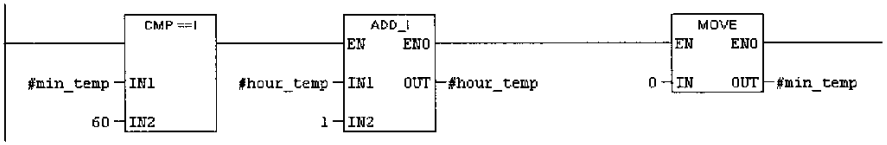
```
FC20 : Title:
Network 1 : Title:
```



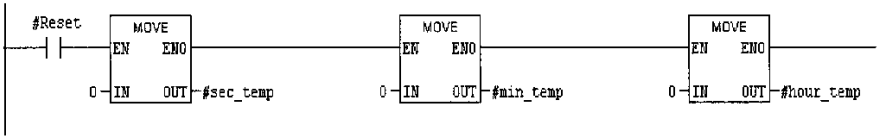

```
Network 2 : Title:
```



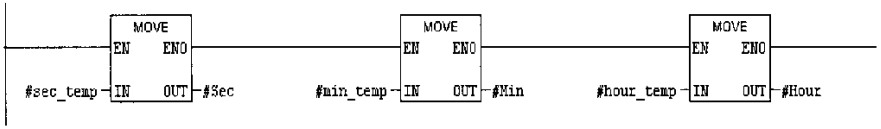

```
Network 3 : Title:
```




```
Network 4 : Title:
```



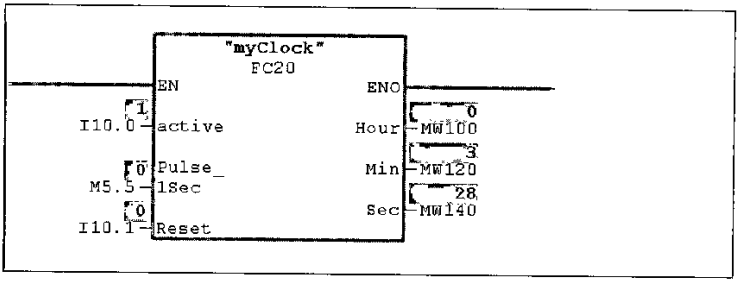

```
Network 5 : Title:
```



شکل ۴-۵۹ برنامه مثال ۴-۷

پس از ذخیره‌سازی و دانلود OB1 را باز کرده و فانکشن را در آن صدا می‌زنیم. ورودی و خروجی‌ها را به آدرس‌های مورد نظر اختصاص داده، دانلود و تست می‌کنیم.

```
OB1 :   "Main Program Sweep (Cycle)"
Network 1: Title:
```



شکل ۴-۶۰ برنامه مثال ۴-۷ در حالت Online

تمرین ۴-۴: مثال ۴-۷ را به‌صورتی اصلاح کنید که میلی‌ثانیه را نیز در خروجی فانکشن نشان دهد.

مثال ۴-۸: فانکشن کنترل چپگرد راستگرد موتور

تعداد پنج موتور M1 تا M5 وجود دارد که می‌خواهیم همه را توسط یک فانکشن کنترل کنیم. در کتاب سطح مقدماتی با برنامه کنترل چپگرد-راستگرد آشنا شدید. در این مثال می‌خواهیم این برنامه را در یک FC و با استفاده از پارامترهای قراردادی بازنویسی نماییم.

مراحل کار عبارتند از:

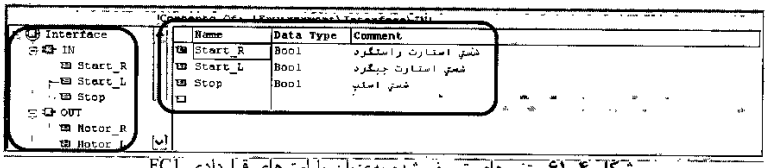
۱- ابتدا یک بلاک جدید با نام FC1 ایجاد می‌نماییم.

۲- پارامترهای مختلفی که باید در هنگام فراخوانی FC1 نشان داده شوند را در قسمت پارامترهای قراردادی ایجاد می‌نماییم.

جدول ۴-۳

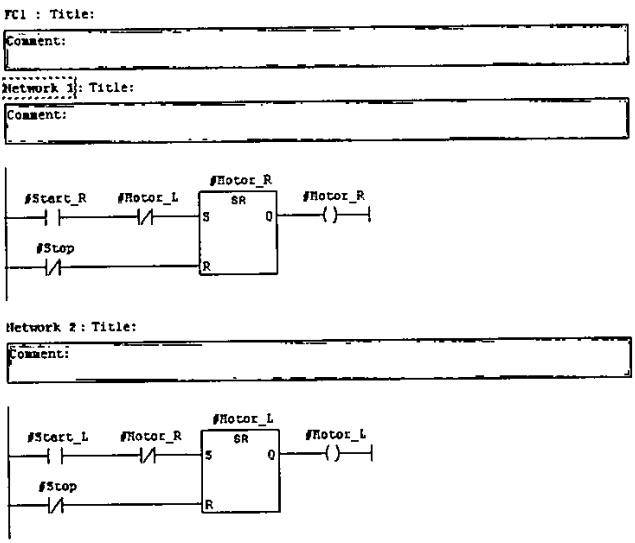
توضیح	Data Type	Name	
شستی استارت راستگرد (تیغه باز)	BOOL	Start_R	IN
شستی استارت چپگرد (تیغه باز)	BOOL	Start_L	IN
شستی استپ (تیغه بسته)	BOOL	Stop	IN
فرمان روشن شدن موتور به‌صورت راستگرد	BOOL	Motor_R	OUT
فرمان روشن شدن موتور به‌صورت چپگرد	BOOL	Motor_L	OUT

شکل ۴-۶۱ Declaration Section مربوط به FC1 را نشان می‌دهد که پارامترهای قراردادی مورد نظر در آن ایجاد شده است.



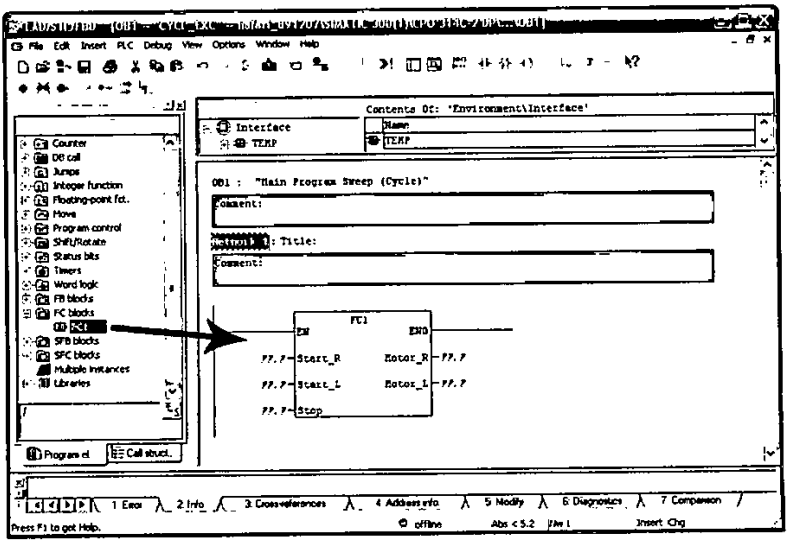
شکل ۴-۶۱ متغیرهای تعریف شده به‌عنوان پارامترهای قراردادی FC1

۳ – برنامه‌نویسی

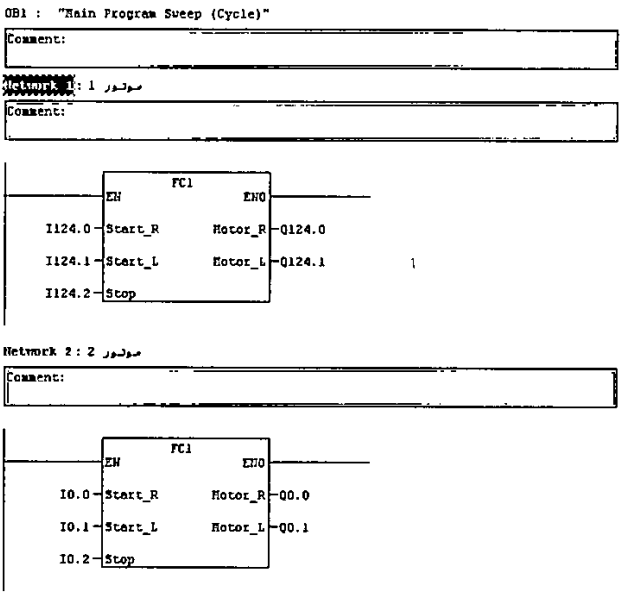


شکل ۴-۶۲ برنامه نوشته شده در FC1 جهت مثال ۴-۸

۴- صدا زدن فانکشن در OB1



شکل۴-۶۳ فراخوانی FC1 در OB1



شکل ۴-۶۴

همانطور که در شکل ۴-۶۴ مشخص است، FC1 دوبار در OB1 فراخوانی شده و در هربار فراخوانی آدرس‌های مختلفی به‌آن اختصاص داده شده است. در مرحله اول فراخوانی آدرس‌های مربوط به موتور۱ و در بار دوم آدرس‌های مربوط به موتور ۲ به آن اختصاص یافته است. به‌همین روش می‌توان FC1 را فراخوانی نموده و آدرس‌های موتورهای ۳ تا ۵ را نیز به آن اختصاص داد.

فصل ۴

۴-۶ نحوه کار با FB

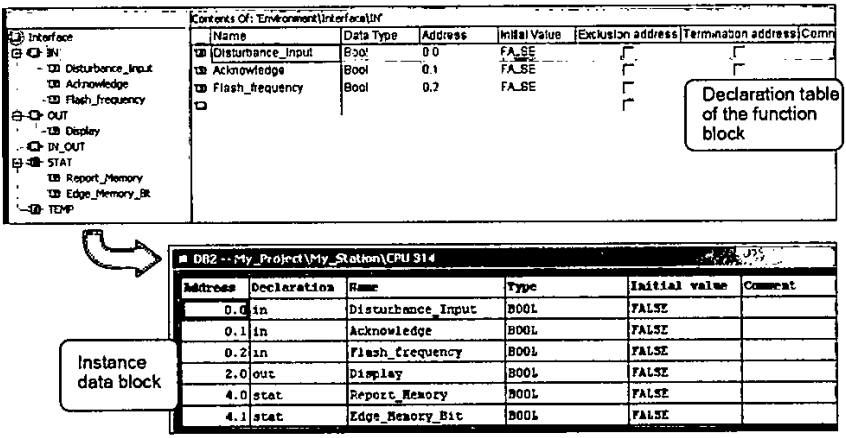
ساختار FB و برنامه‌نویسی آن از جهات بسیاری شبیه FC است به همین دلیل مراحل کار با آن عمدتاً مشابه مراحل کار با FC است. ابتدا باید FB را ایجاد کرد، پارامترها و متغیرهای آن را تعریف نمود، برنامه‌نویسی آن را کامل کرد و نهایتاً در بلاک ماقبل آن را صدا زد. تمام این مراحل در کار با FC نیز انجام می‌شد. ولی در بین این مراحل، تفاوت‌هایی را نیز مشاهده می‌کنیم. یکی از دلایل بروز تفاوت ناشی از تفاوت عمده بین FC و FB است که قبلاً نیز اشاره شد. برخلاف FC که حافظه ندارد FB دارای حافظه ای از جنس DB است.

با مقایسه FB با FC می‌توان به نکات زیر اشاره نمود:

- هر FB نیاز به یک DB دارد.
- FB شبیه FC دارای پارامترهای IN و OUT و IN_OUT است.
- برخلاف FC پارامترهای IN و OUT و IN_OUT در FB دارای آدرس هستند، زیرا این پارامترها در DB ذخیره می‌گردند.

- برخلاف FC می‌توان به تمام پارامترهای IN و OUT و IN_OUT در FB مقدار اولیه اختصاص داد.
- در FB شبیه FC می‌توان متغیر TEMP تعریف کرد. این متغیر در L-Stack ذخیره می‌گردد و در DB آن را نمی‌بینیم.
- در FB برخلاف FC می‌توان متغیر STAT تعریف کرد، این متغیر که برای ذخیره نتایج میان برنامه استفاده می‌شود در DB ذخیره می‌گردد.
- در FB برخلاف FC پارامتر RETURN وجود ندارد.
- FC برخلاف FB معمولاً برای برنامه‌نویسی تقسیم شده استفاده نمی‌شود.
- در برنامه‌نویسی ساختار یافته اگر چه می‌توان از FC یا FB استفاده کرد، ولی FB را معمولاً برای کنترل المان نهایی[1] به‌کار می‌برند و FC را برای تقسیم فرآیند به نواحی یا Taskهای مختلف استفاده می‌نمایند.
- دستور CALL که برای FC در LAD استفاده می‌شد، برای FB قابل استفاده نیست.
- دستور RET که برای FC در LAD استفاده می‌شد، برای FB نیز قابل استفاده است.
- نحوه Protect کردن FB مشابه توضیحات داده شده برای FC است.

پس از ایجاد FB و باز کردن آن، در قسمت Declaration محیطی مانند شکل ۴-۶۵ مشاهده می‌شود. کلیه متغیرهایی که در این بخش ایجاد شوند، به‌طور اتوماتیک در دیتابلاک اختصاصی FB ایجاد و ذخیره می‌شوند. به شکل ۴-۶۵ توجه کنید.



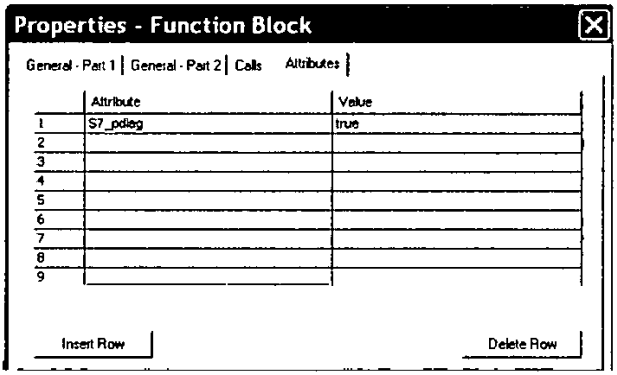
شکل ۴-۶۵ متغیرهای ایجاد شده در دیتابلاک اختصاصی FB

همانطور که در شکل ۴-۶۵ مشاهده می‌گردد، می‌توان در یک FB متغیر محلی از نوع Static تعریف نمود. این نوع متغیر پس از ایجاد در FB، به‌طور اتوماتیک در دیتابلاک اختصاصی FB نیز ایجاد شده و ذخیره می‌گردد. کاربر می‌تواند ضمن برنامه خود از این نوع متغیر محلی که نقشی شبیه متغیر محلی از نوع Temp بازی می‌کند استفاده نماید. در رابطه با

1. Final Element

ایجاد و استفاده از متغیرهای محلی Static باید توجه نمود که میزان اندازه مجاز این متغیرها بستگی به دیتابلاک اختصاصی FB داشته و ربطی به ناحیه L Stack ندارد.

نکته دیگری که در شکل قبل در بخش Interface مشاهده می‌شود، ستون‌هایی است که در جلوی پارامترها با عناوین Exclusion Address و Termination Address وجود دارد. از اینجا می‌توان ارتباط بین پارامترها و نرم‌افزار Pdiag را تعریف کرد. در صورت فعال‌سازی این گزینه‌ها اگر Attribute فانکشن بلاک را ببینیم مانند شکل ۴-۶۶ برای آن فعال شده است.



شکل ۴-۶۶

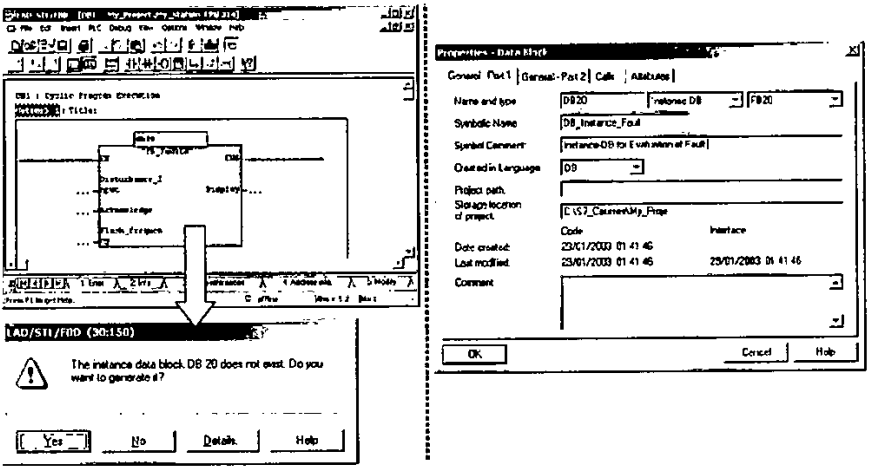
فصل ۴

روش ایجاد دیتابلاک اختصاصی برای یک FB

به دو روش می‌توان دیتابلاک اختصاصی یک FB را ایجاد و به‌آن FB اختصاص داد. این دو روش عبارتند از:

۱- مشابه شکل ۴-۶۷ (الف) یک دیتابلاک ایجاد نموده و نوع آنرا روی حالت Instance DB گذاشته و FB مورد نظر جهت اختصاص دیتابلاک به آنرا مشخص می‌کنیم.

۲- مشابه شکل ۴-۶۷ (ب) می‌توان در زبان FBD یا LAD، FB مربوطه را فراخوانی نموده و در بالای FB نام دیتابلاک مورد نظر را وارد نمود. در صورتی‌که دیتابلاک مورد نظر موجود نباشد، به‌طور اتوماتیک ایجاد می‌گردد. این روش مرسوم‌تر از روش قبل است.



الف ب

شکل ۴-۶۷ روش‌های ایجاد دیتابلاک اختصاصی یک FB

نکات مهم در باره ایجاد دیتابلاک اختصاصی

- هر دیتابلاک را می‌توان به یک FB اختصاص داد.
- هر FB می‌تواند دارای چندین دیتابلاک اختصاصی باشد، بدین معنی که هربار FB فراخوانی می‌شود به آن یک دیتابلاک اختصاص داده شود.
- اگر متغیرهای موجود در یک FB تغییر داده شود، باید DB اختصاصی از نو ایجاد شود.

مراحل کار با FB

برای کار با FB به‌طور کلی می‌توان مراحل زیر را طی نمود:

- ایجاد FB
- ایجاد متغیرهای محلی و پارامترهای قراردادی لازم در FB
- نوشتن برنامه در FB
- ایجاد دیتابلاک اختصاصی
- ذخیره‌سازی و دانلود FB و دیتابلاک اختصاصی
- فراخوانی FB و DB اختصاصی در بلاک ماقبل

به منظور تفهیم مطلب، مراحل کار با FB را با ذکر چند مثال دنبال می‌کنیم.

مثال ۴-۹: استفاده از FB و FC برای کنترل دو ناحیه فرآیند (تکمیل مثال ۴- ۳)

مثال ۴-۳ که در آن کنترل دو ناحیه از فرآیند توسط FC10 انجام می‌شد را به‌صورت کامل‌تر و با تغییرات زیر توسط FB می‌نویسیم.

- فرآیند دارای دو ناحیه مختلف است Area_01 و Area_02.
- در هر ناحیه تعداد ۱۰ مخزن وجود دارد که ارتفاع آنها متفاوت است.
- هر مخزن یک ترانسمیتر سطح دارد که بین حدود نرمال مربوط به همان مخزن کالیبره شده است. حد پایین که به ازای آن ترانسمیتر 4mA می‌فرستد 0 متر وحدبالاهمان‌ارتفاع‌مخزن‌است‌که ترانسمیتر به ازای آن 20 mA می‌فرستد.
- هر مخزن یک سوئیچ سطح پایین LSL و یک سوئیچ سطح بالا LSH دارد که در ارتفاع مناسب نصب شده است.
- هر مخزن دارای یک ولو ورودی و یک ولو خروجی است.

منطق برنامه

- اگر سطح کمتر از ۲۰ درصد ارتفاع مخزن باشد یا سوئیچ LSL فعال شود، در اینصورت ولو ورودی باز و ولو خروجی بسته شود.
- اگر سطح بین ۲۰ تا ۸۰ درصد ارتفاع مخزن باشد، هر دو ولو باز هستند.
- اگر سطح بیش از ۸۰ درصد ارتفاع مخزن باشد یا سوئیچ LSH فعال شود، در اینصورت ولو ورودی بسته و ولو خروجی بسته شود.
- برای جلوگیری از فرمان‌های قطع و وصل ولو ورودی که ممکن است در اثر نوسان سطح ایجاد شود فرمان به ولو با تأخیر ۵ ثانیه اعمال گردد.

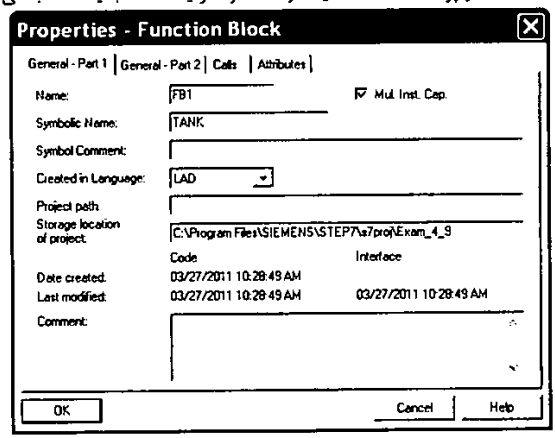
حل: در این برنامه از دو FC برای تقسیم‌بندی دو ناحیه فرآیند استفاده می‌کنیم. برای کنترل مخازن از یک FB استفاده می‌کنیم. در هر FC به تعداد مخازن FB را صدا می‌زنیم.

با توجه به ساختار ذکر شده لازم است ابتدا FB را ایجاد و کامل کنیم. سپس FCها را ایجاد کرده و در آنها FB را صدا بزنیم و نهایتاً OB1 را ایجاد کرده و FCها را در آن فراخوان کنیم.

فصل ۴

ایجاد FB1 با نام TANK

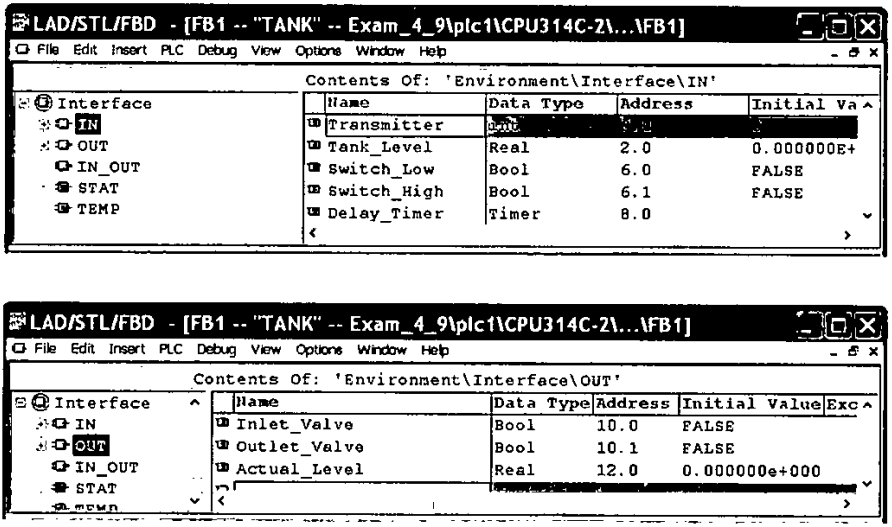
در محیط Simatic Manager در پوشه Blocks کلیک راست کرده و یک FB جدید انتخاب می‌کنیم.



شکل ۴-۶۸ تعریف FB در مثال ۴-۹

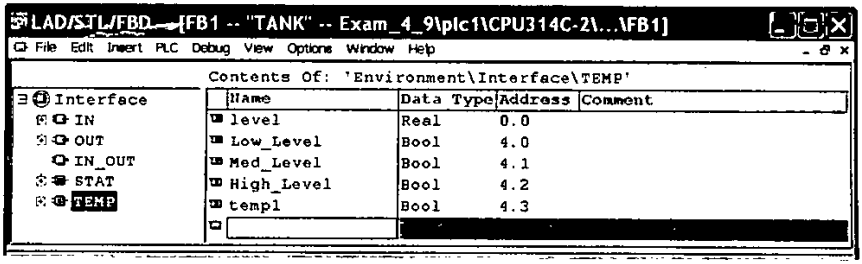
تعریف ورودی و خروجی‌های FB1

با باز کردن FB قدم اول تعریف پارامترهای ورودی و خروجی است. همانطور که در شکل ۴-۶۹ دیده می‌شود، برخلاف FC در اینجا تمام ورودی و خروجی‌ها دارای آدرس و مقدار اولیه هستند. این آدرس‌ها و مقادیر در دیتابلاک مربوط به FB وارد خواهد شد.



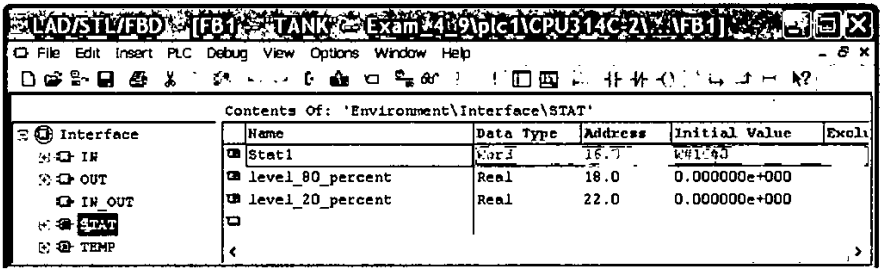
شکل ۴-۶۹ ورودی و خروجی‌های FB در مثال ۴-۹

متغیرهای میان برنامه می‌توانند Temp یا Stat باشند، با توجه به اینکه نیازی به ذخیره‌سازی مقادیر میان برنامه در دیتابلاک نیست بنابراین می‌توانیم آنها را از نوع TEMP انتخاب کنیم. این متغیرها شبیه مثال ۴-۳ خواهند بود.



شکل ۴-۷۰ متغیرهای TEMP در مثال ۴-۹

برای ذخیره‌سازی خروجی RET_VAL مربوط به FC105 از متغیر نوع STAT استفاده می‌کنیم تا بتوان از آن در بلاک‌های دیگر نیز استفاده کرد. همینطور مقادیر ۸۰ درصد و ۲۰ درصد سطح را که در محاسبات میان برنامه مورد نیاز هستند به‌صورت Stat تعریف می‌کنیم.



شکل ۴-۷۱ متغیرهای STAT در مثال ۴-۹

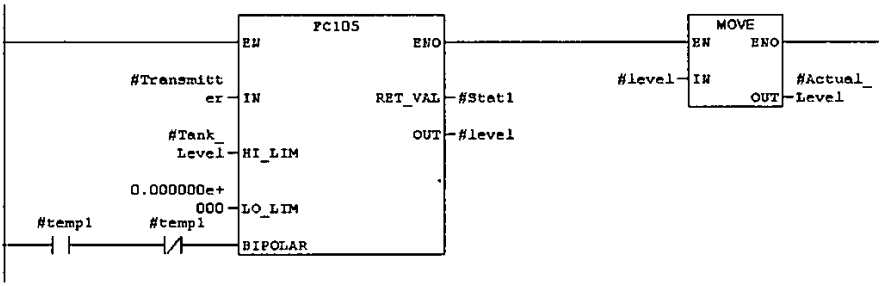
همانطور که در شکل‌های بالا دیده می‌شود:

- متغیرهای TEMP دارای مقدار اولیه نیستند، زیرا در L-Stack ذخیره می‌شوند.
- آدرس‌هایی که برای IN و OUT و IN_OUT و STAT که در دیتا بلاک ذخیره می‌شوند از صفر شروع شده و به دنبال هم ادامه می‌یابد و هیچ آدرس تکراری در این ناحیه نداریم؛ ولی آدرس‌های TEMP که در L-Stack ذخیره می‌شوند به‌طور جداگانه از صفر شروع شده و ادامه می‌یابند.

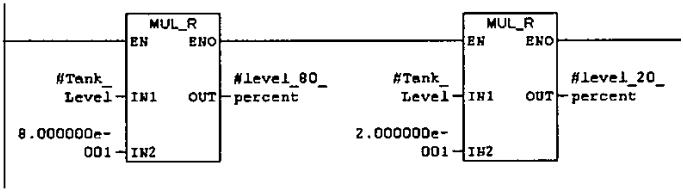
برنامه‌نویسی

فصل ۴

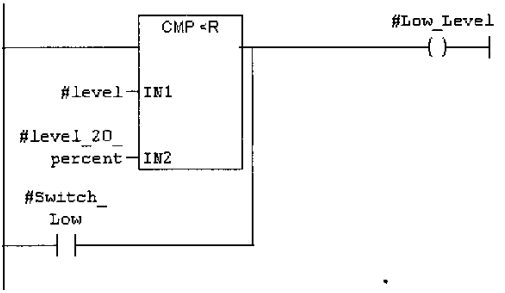
```
FB1 : Title:
Network 1 : Title:
```



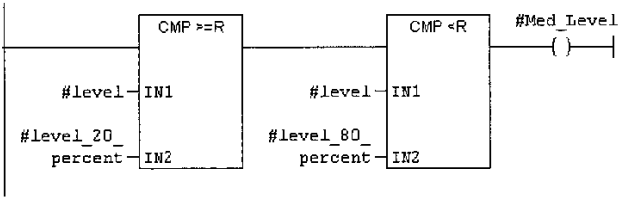

```
Network 2 : Title:
```



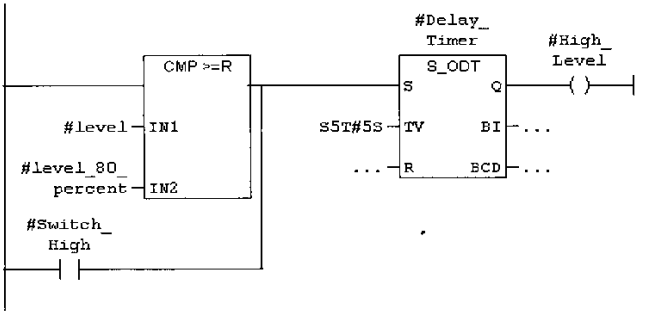
Network 3 : Title:



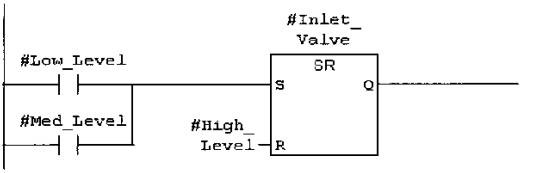
Network 4 : Title:

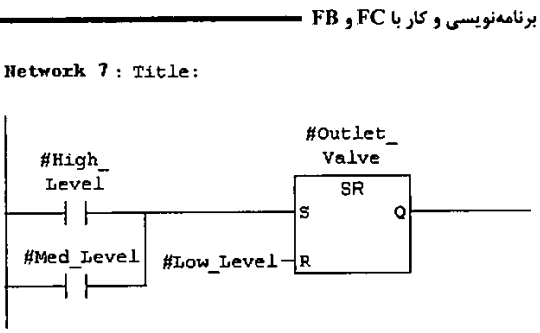


Network 5 : Title:



Network 6 : Title:



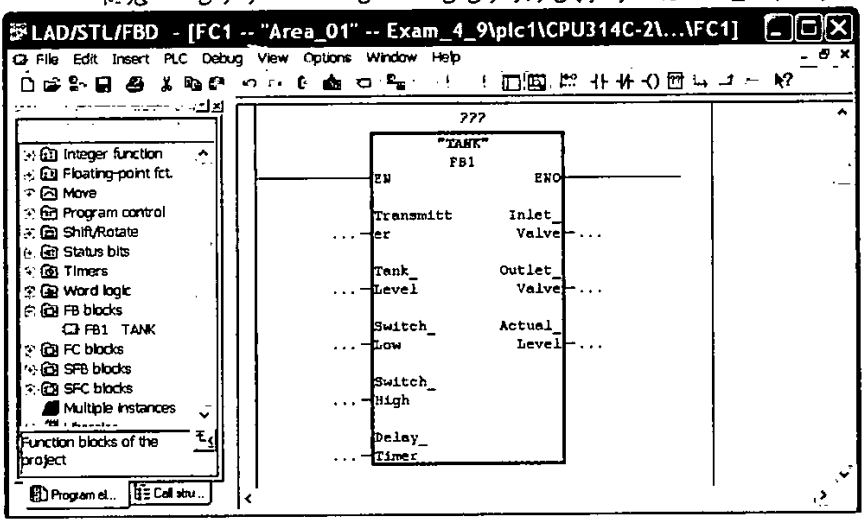


شکل ۴-۷۲ برنامه FB در مثال ۴-۹

اکنون برنامه FB را ذخیره کرده و می‌بندیم.

ایجاد FC1 برای مخازن ناحیه ۱ و صدا زدن FB1 در آن

FC1 را با نام Area_01 ایجاد کرده و پس از باز کردن آن مانند شکل ۴-۷۳ FB1 را در آن صدا می‌زنیم.



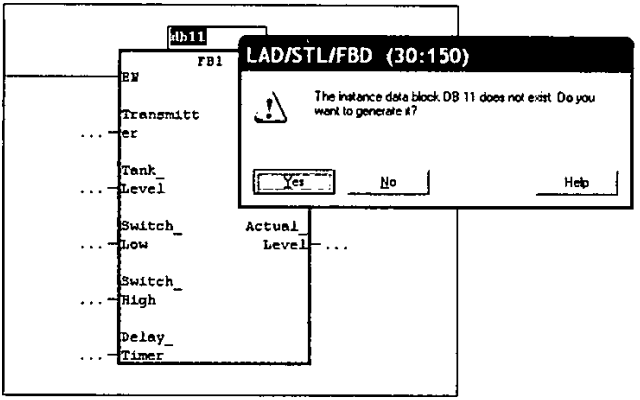
شکل ۴-۷۳ صدا زدن FB در FC مربوط به مثال ۴-۹

با توجه به شکل، دو تفاوت عمده بین صدا زدن FB و FC مشاهده می‌شود:

۱- ورودی و خروجی‌های FB با نقطه‌چین و بدون علامت قرمز ظاهر می‌شوند. مفهوم این امر این است که حتی اگر برخی ورودی و خروجی‌ها را خالی بگذاریم و آدرس یا مقداری به آنها اختصاص ندهیم باز امکان ذخیره و دانلود وجود دارد زیرا ورودی و خروجی‌هایی هم که خالی هستند دارای مقدار اولیه می‌باشند. در FC این امکان وجود نداشت و همه ورودی و خروجی‌ها با رنگ قرمز ظاهر می‌شد. در FC قبل از اختصاص مقدار و آدرس به همه ورودی و خروجی‌ها امکان ذخیره‌سازی و دانلود وجود نداشت.

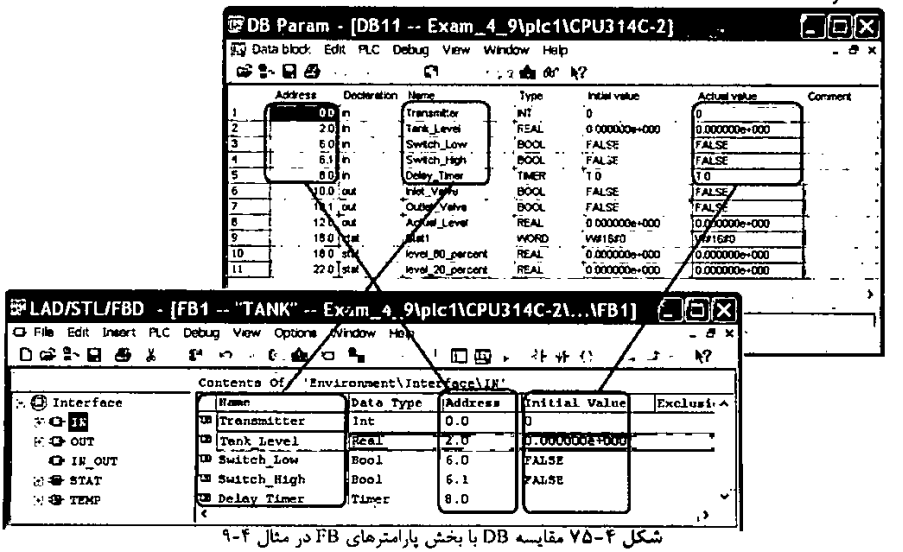
۲- در بالای FB علامت سوال قرمز رنگ ظاهر می‌شود که در این قسمت بایستی یک دیتا بلاک معرفی شود. این نیاز در FC وجود نداشت.

برای اختصاص دیتابلاک به FB به یکی از دو روشی که قبلاً تشریح شد می‌توان عمل نمود. راه ساده‌تر این است که در هنگام فراخوانی FB در بالای آن DB را اختصاص دهیم. توجه شود که DB با این شماره نبایستی قبلاً وجود داشته باشد در اینصورت نرم‌افزار آنرا به‌طور خودکار ایجاد می‌نماید.



شکل ۴-۷۴ اختصاص DB به FB در مثال ۴-۹

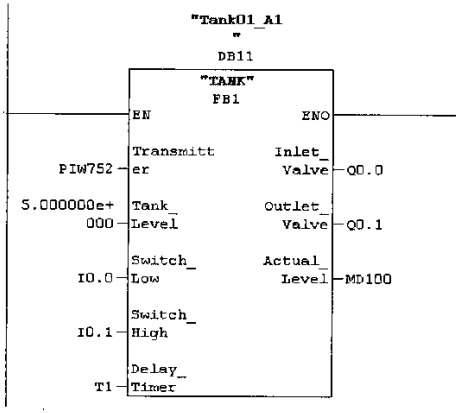
اگر به پوشه Blocks برگردیم و دیتا بلاک ایجاد شده را باز کنیم شکل ۴-۷۵ را می‌بینیم. همانطور که مشخص است بجز متغیرهای Temp سایر موارد در دیتا بلاک دیده می‌شوند. مقدار اولیه و آدرس آنها از بخش Declaration بالای FB گرفته شده است.



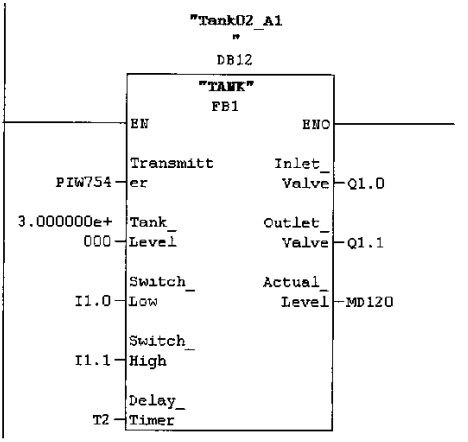
شکل ۴-۷۵ مقایسه DB با بخش پارامترهای FB در مثال ۴-۹

به این ترتیب به تعداد مخازن موجود در ناحیه ۱ فانکشن بلاک FB1 را صدا می‌زنیم و هر بار شماره دیتابلاک جدیدی را به آن اختصاص می‌دهیم. بهتر است با کلیک راست روی DBها به آنها اسم سمبلیک نیز اختصاص دهیم.
بخشی از برنامه FC1 به‌صورت زیر است.

```
FC1 : Title:
Networzk 1 : Title:
```




```
Network 2 : Title:
```



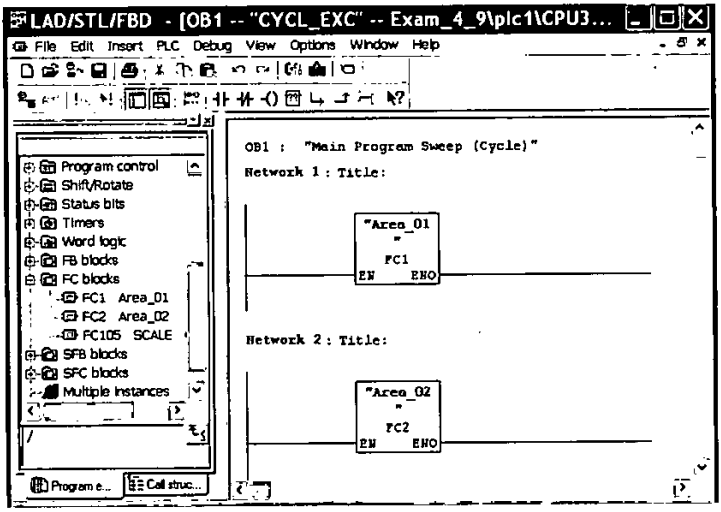
شکل ۴-۷۶ بخشی از برنامه FC1 در مثال ۴-۹

ایجاد FC2 برای مخازن ناحیه ۲ و صدا زدن FB1 در آن

FC2 را با نام سمبلیک Area_02 ایجاد و باز کرده سپس مشابه کاری که در FC1 انجام شد در FC2 نیز به تعداد مخازن ناحیه ۲ فانکشن‌بلاک FB1 را صدا زده و هر بار DB جدیدی را به آن اختصاص می‌دهیم.
FC2 را ذخیره کرده و می‌بندیم.

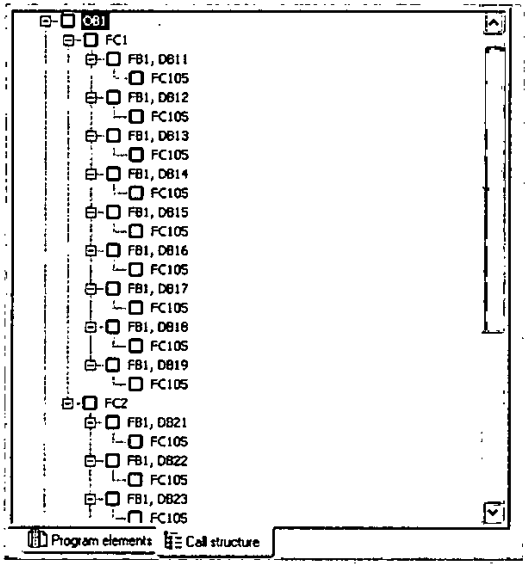
صدا زدن FC1 و FC2 در OB1

FC1 و FC2 داری هیچ ورودی و خروجی خاصی نیستند. در واقع این دو برای تقسیم‌بندی فرآیند به‌کار رفته‌اند. مطابق شکل ۴-۷۷ آنها را در OB1 صدا می‌زنیم.



شکل ۴-۷۷ صدا زدن فانکشن‌ها در OB1 مربوط به مثال ۴-۹

OB1 را ذخیره کرده و می‌بندیم. سلسله مراتب فراخوانی بلاک‌ها را با کلیک روی Call Structure مانند شکل ۴-۷۸ خواهیم دید. همانطور که دیده می‌شود هر FB با یک DB صدا زده شده و در آن FC105 نیز فراخوان شده است.



شکل ۴-۷۸ سلسه مراتب فراخوانی در مثال ۴-۹

دانلود بلاک‌ها و تست برنامه

در دانلود بلاک‌ها بایستی توجه داشت که علاوه بر FC و FB و OB بایستی همه DBها نیز دانلود شوند در غیر اینصورت CPU با مشکل مواجه خواهد شد.

پس از دانلود اگر هر کدام از DBها را باز و مانیتور کنیم وضعیت سیگنال‌های همان مخزن خاص را در آن خواهیم دید. شکل ۴-۷۹ DB11 را برای مخزن شماره یک نشان می‌دهد. ستون Actual Value@ مقادیر را در حالت Online نشان می‌دهد.

DB Param - [@DB11 -- Exam_4_9\plc1\CPU314C-2 ONLINE]

Data block Edit PLC Debug View Window Help

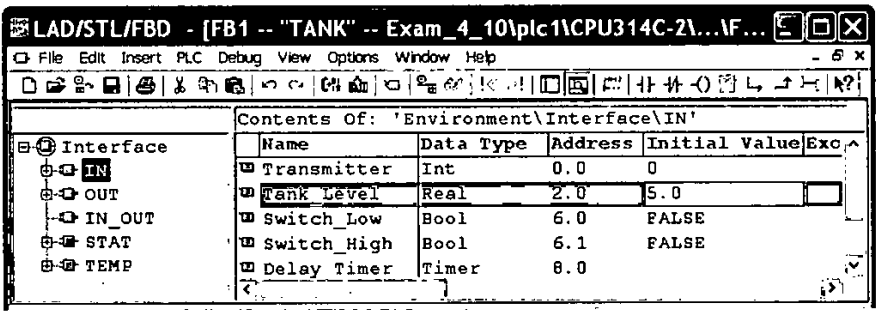
	Address	Declaration	Name	Type	Initial value	@Actual value	Actual value	Comment
1	0.0	in	Transmitter	INT	0	15083	0	
2	2.0	in	Tank_Level	REAL	0.000000e+000	5.0	0.000000e+000	
3	6.0	in	Switch_Low	BOOL	FALSE	FALSE	FALSE	
4	6.1	in	Switch_High	BOOL	FALSE	FALSE	FALSE	
5	8.0	in	Delay_Timer	TIMER	T 0		T 0	
6	10.0	out	Inlet_Valve	BOOL	FALSE	TRUE	FALSE	
7	10.1	out	Outlet_Valve	BOOL	FALSE	TRUE	FALSE	
8	12.0	out	Actual_Level	REAL	0.000000e+000	2.727684	0.000000e+000	
9	16.0	stat	Stat1	WORD	W#16#0	W#16#0000	W#16#0	
10	18.0	stat	level_80_percent	REAL	0.000000e+000	4.0	0.000000e+000	
11	22.0	stat	level_20_percent	REAL	0.000000e+000	1.0	0.000000e+000	

شکل ۴-۷۹ DB11 در حالت Online مربوط به مثال ۴-۹

تمرین ۴-۵: در مثال قبل، با قطع و وصل تغذیه PLC وضعیت مقادیر ذخیره شده در دیتابلاک را بررسی کنید. آیا پاک می‌شوند ؟

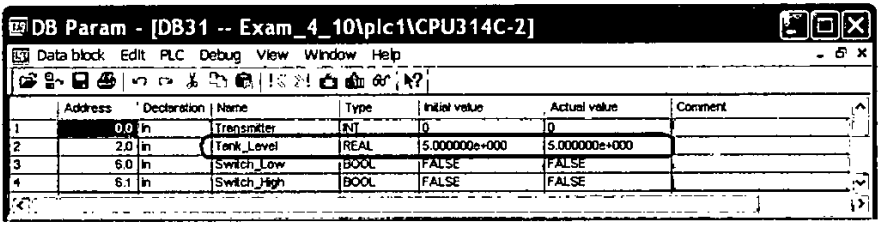
مثال ۴-۱۰:تنظیمات مقدار اولیه در FB

در برنامه مثال ۴-۹ اگر در هنگام فراخوانی FB1 ورودی Tank_Level مقدار داده نشود آن را صفر در نظر می‌گیرد. می‌توان مقدار اولیه 5.0 را در بخش Intial Value مانند شکل ۴-۸۰ وارد کرد. در اینصورت اگر کاربر در برنامه‌نویسی به این ورودی مقدار بدهد مقدار وارد شده مبنای کار است و اگر آنرا خالی رها کند مقدار پیش‌فرض 5.0 مبنای کار خواهد بود. این کار وقتی مفید خواهد بود که اکثراً مخازن دارای ارتفاع 5.0 متر باشند. در این شرایط برای مخازن فوق نیازی به وارد کردن مقدار در این ورودی نخواهیم داشت.



شکل ۴-۸۰ اختصاص مقدار اولیه به ورودی

در این شرایط مقدار 5.0 در دیتا بلاک در ستون InitialValue و ستون Actual Value ظاهر خواهد شد.

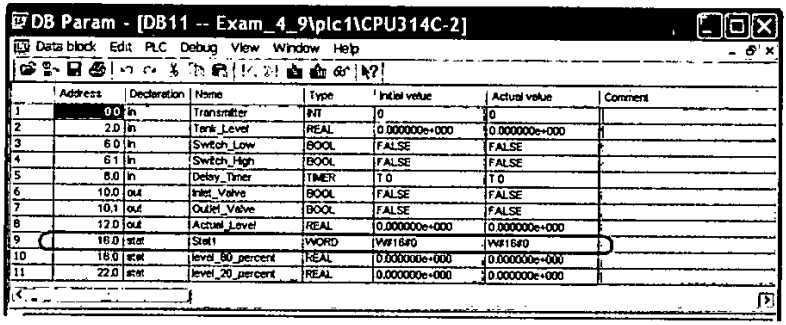


شکل ۴-۸۱ مشاهده مقدار اولیه در دیتا بلاک

مثال ۴-۱۱: استفاده از آدرس متغیرهای Stat در خارج از FB

برنامه مثال ۴-۹ را به‌صورتی کامل کنید که اگر سیگنال هر کدام از ترانسمیترهای مخازن هر دو ناحیه از بازه نرمال خارج شد آلارم روی Q100.0 نمایش داده شود.

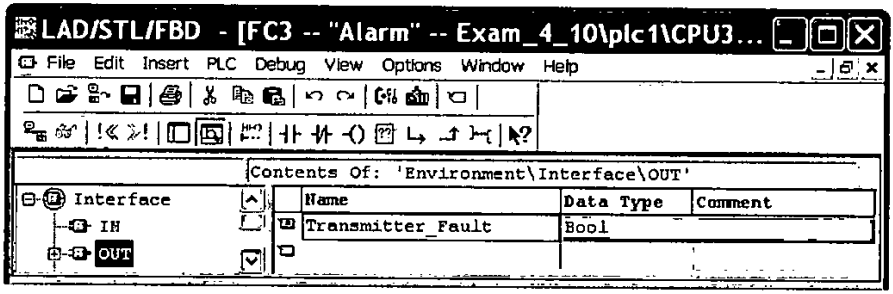
حل: وقتی سیگنال ترانسمیتر از بازه نرمال خارج می‌شود، خروجی RET_VAL فانکشن FC105 مقدار 8 خواهد داشت. از آنجا که این خروجی در متغیر STAT1 ذخیره شده در DBها دارای آدرس است که می‌توان از آدرس مربوطه در برنامه استفاده کرد. شکل ۴-۸۲ STAT1 را در DB11 نشان می‌دهد. همانطور که دیده می‌شود آدرس DB11.DBW16 می‌باشد. در سایر DBها نیز آدرس سطر مربوط به STAT1 مشابه است، یعنی DBW16 که اگر عدد 8 (که معادل 1000 باینری است) در آن قرار گیرد بیت DBX17.3 آن یک خواهد شد.



	Address	Declaration	Name	Type	Initial value	Actual value	Comment
1	0.0	in	Transmitter	INT	0	0	
2	2.0	in	Tank_Level	REAL	0.000000e+000	0.000000e+000	
3	6.0	in	Switch_Low	BOOL	FALSE	FALSE	
4	6.1	in	Switch_High	BOOL	FALSE	FALSE	
5	8.0	in	Delay_Timer	TIMER	T 0	T 0	
6	10.0	out	Inlet_Valve	BOOL	FALSE	FALSE	
7	10.1	out	Outlet_Valve	BOOL	FALSE	FALSE	
8	12.0	out	Actual_Level	REAL	0.000000e+000	0.000000e+000	
9	16.0	stat	Stat1	WORD	W#16#0	W#16#0	
10	18.0	stat	level_80_percent	REAL	0.000000e+000	0.000000e+000	
11	22.0	stat	level_20_percent	REAL	0.000000e+000	0.000000e+000	

شکل ۴-۸۲ آدرس متغیر STAT در دیتا بلاک

برای تولید آلارم، برنامه مربوطه را در یک فانکشن می‌نویسیم. FC3 را با نام Alarm ایجاد نموده و یک خروجی با نام Transmitter_Fault از جنس Bool تعریف می‌کنیم.

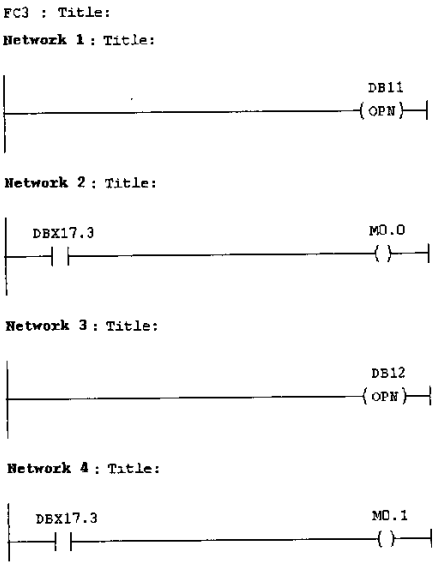


شکل ۴-۸۳ فانکشن FC3 در مثال ۴-۱۱

در استفاده از آدرس‌های DBهای Instance بایستی توجه شود که نمی‌توان مشابه Global DB آنها را با دستورات ترکیبی مانند DB11.DBW16 خواند. شکل ۴-۸۴ نشان می‌دهد که نمی‌توان بیت مربوط به خطای ترانسمیتر را با دستور ترکیبی از DB11 خواند.

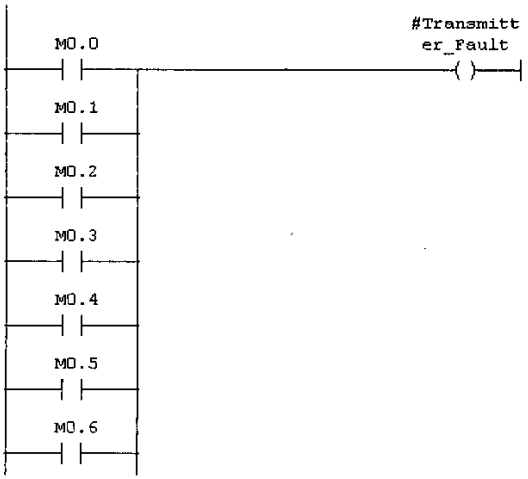
شکل ۴-۸۴ روش نادرست خواندن دیتا از Instance DB

روش درست آن است که ابتدا با دستور OPN دیتا بلاک را باز کرده سپس از بیت مربوطه استفاده کنیم. در برنامه زیر تمام بیت‌های مربوطه از DB را در Memory Bits ذخیره می‌کنیم.



شکل ۴-۸۵ روش درست خواندن دیتا از Instance DB

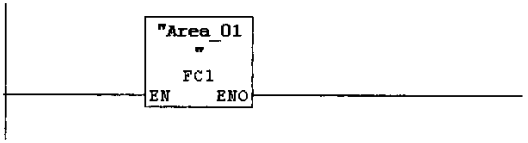
در انتهای FC3 بیت‌های فوق را با یکدیگر OR کرده و روی خروجی فانکشن می‌ریزیم.



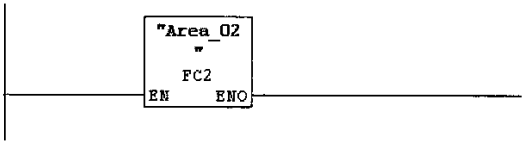
شکل ۴-۸۶ بخشی از برنامه FC3 در مثال ۴-۱۱

با صدا زدن FC3 در OB1 خروجی آن را به Q100.0 اختصاص می‌دهیم.

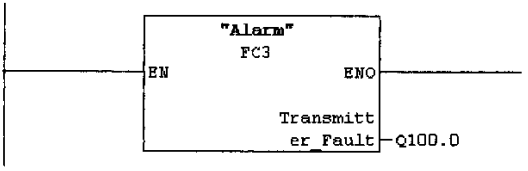
```
OB1 :  "Main Program Sweep (Cycle)"
Network 1 : Title:
```




```
Network 2 : Title:
```




```
Network 3 : Title:
```



شکل ۴-۸۷ صدا زدن فانکشن‌ها در مثال ۴-۱۱

مثال ۴-۱۲: کنترل نوارنقاله در کارخانه بطری پرکنی

در این مثال به بررسی کنترل یک نوارنقاله در کارخانه بطری پرکنی خواهیم پرداخت و در حل مثال از FB استفاده می‌نماییم. همانطور که در شکل ۴-۸۸ نشان داده شده است، در این پروسه از اداوت زیر استفاده شده است:

- **نوارنقاله**

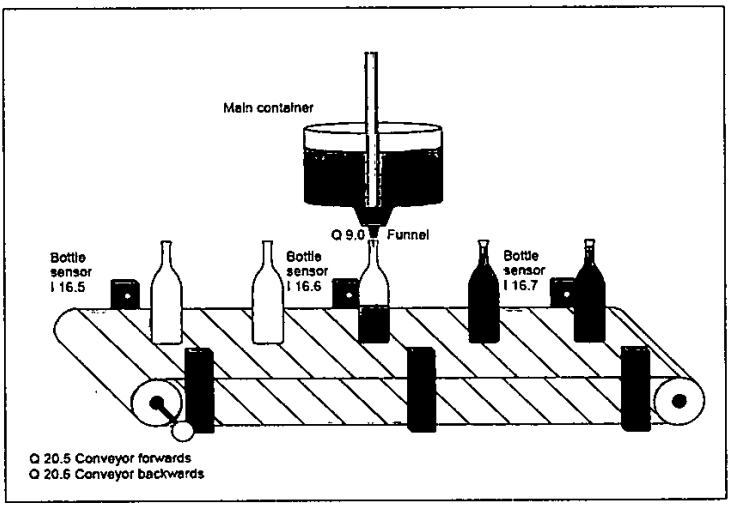
این نوارنقاله دارای یک موتور راستگرد، چپگرد می‌باشد که حرکت راستگرد آن باعث حرکت نوارنقاله به جلو و حرکت چپگرد آن باعث حرکت نوارنقاله به عقب می‌شود.

- **سنسورهای تشخیص بطری**

به‌منظور تعیین و تشخیص موقعیت بطری از سه سنسور که در ابتدا، وسط و انتهای خط قرار گرفته‌اند استفاده می‌شود. همه این سنسورها به‌صورت NO (نرمال باز) در نظر گرفته شده‌اند.

- **مخزن اصلی**

در این مخزن ماده‌ای که قرار است به درون بطری‌ها ریخته شود قرار گرفته است. این مخزن دارای یک شیر خروجی بوده که در حالت عادی بسته است و در صورتی‌که برق‌دار شود باز شده و مایع درون آن خارج می‌شود.



شکل ۴-۸۸ پروسه مورد نظر جهت مثال ۴-۱۲

منطق کنترل

کنترل این پروسه، خود از بخش‌های مختلفی تشکیل شده است:

- **روشن و خاموش شدن کل سیستم:** با فشردن شستی I0.0 (تیغه باز) کل سیستم روشن شده و با فشردن I0.1 (تیغه بسته) کل سیستم خاموش شود.
- **انتخاب مد کنترل (دستی-اتوماتیک):** این سیستم دارای دو وضعیت کنترلی دستی و اتوماتیک است. اگر I0.4 مقدار 0 داشته باشد، مد دستی و اگر مقدار 1 داشته باشد مد اتوماتیک انتخاب می‌شود. با فشردن شستی I0.5 (تیغه باز) مد انتخابی تایید می‌شود. مد انتخاب شده توسط دو لامپ نمایش داده می‌شود. لامپ Q8.2 نشان‌دهنده حالت دستی و لامپ Q8.3 نشان‌دهنده حالت اتوماتیک است.

توجه: هنگامی که مد تغییر کند یا سیستم Off شود، مد انتخابی قبلی غیرفعال می‌شود.

- **منطق کنترل در مد دستی:** در حالت دستی با فشردن و نگه‌داشتن شستی I0.2 (تیغه باز)، نوارنقاله به جلو حرکت می‌کند (خروجی Q 20.5 فعال می‌شود) و با فشردن و نگه‌داشتن شستی I0.3 (تیغه باز) نوارنقاله به عقب حرکت می‌کند (خروجی Q 20.6 فعال می‌شود).
- **منطق کنترل در مد اتوماتیک:** در مد اتوماتیک، موتور به‌صورت راستگرد روشن شده و نوارنقاله به جلو حرکت می‌کند. در این مد، موتور در دو حالت خاموش می‌شود:

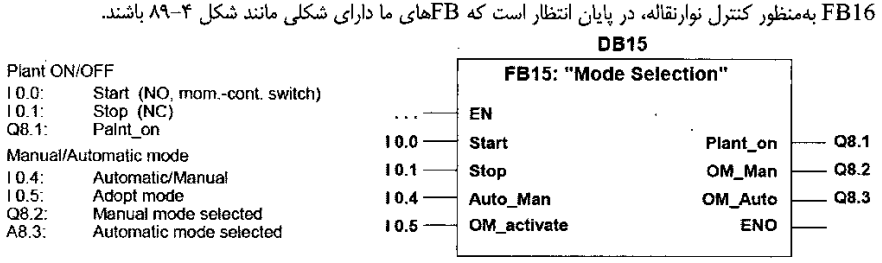
۱- شستی استپ I0.1 (تیغه بسته) فشرده شود.

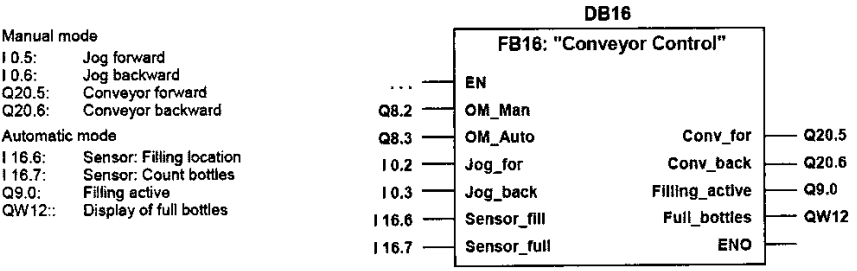
۲- سنسور I16.6 وجود بطری را در زیر مخزن حس نماید. این سنسور در شرایط عادی 0 و در حالت تشخیص بطری 1 می‌فرستد.

- **عملیات مایع‌گیری:** هنگامی که یک بطری زیر مخزن قرار گرفت (I6.6=1) مرحله تخلیه مایع به درون بطری آغاز می‌شود. در این مرحله خروجی Q9.0 که مربوط به ولو مخزن است روشن می‌شود. این عملیات ۳ ثانیه به طول می‌انجامد. پس از آن موتور به‌طور اتوماتیک روشن شده و بطری بعدی مقابل سنسور قرار می‌گیرد، این روال به‌طور سیکلی تکرار می‌شود.
- **شمارش بطری‌ها:** سنسور I16.7 که در انتهای نوارنقاله قرار گرفته است، بطری‌های پرشده را تشخیص می‌دهد. لازم است تعداد این بطری‌ها در زمانی که سیستم فعال است شمارش شده و در حافظه QW12 به‌فرم BCD قرار گیرد.

فصل ۴

در این مثال برای پیاده‌سازی منطق کنترلی فوق از دو FB استفاده می‌نماییم. FB15 به‌منظور انتخاب مد کنترلی و FB16 به‌منظور کنترل نوارنقاله، در پایان انتظار است که FBهای ما دارای شکلی مانند شکل ۴-۸۹ باشند.





شکل ۴-۸۹

مراحل حل مثال به شرح زیر می‌باشد:

۱- ایجاد FB15 و FB16

۲- تعریف پارامترهای قراردادی مطابق شکل ۴-۸۹

۳- برنامه‌نویسی FB15 و FB16

۴- ذخیره‌سازی و دانلود FB15 و FB16

۵- تعریف سمبل‌های لازم در جدول سمبل‌ها

۶- فراخوانی و آدرس‌دهی FB15 و FB16 در OB1

۷- اختصاص DB15 به‌عنوان دیتابلاک اختصاصی FB15 و DB16 به‌عنوان دیتابلاک اختصاصی FB16

۸- دانلود DB15 و DB16

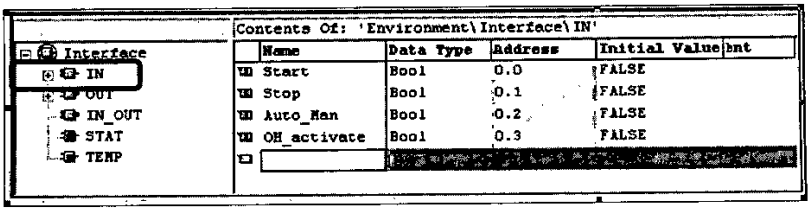
۹- دانلود OB1

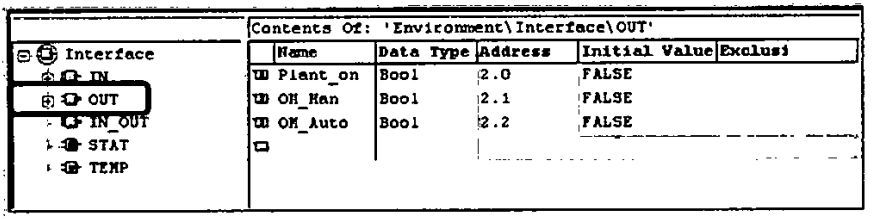
ایجاد FB15 و FB16: لازم است تا FBهای زیر ایجاد شوند:

- FB15 با نام سمبولیک Mode Selection
- FB16 با نام سمبولیک Conveyor Control

تعریف پارامترهای قراردادی در FB15 و FB16:

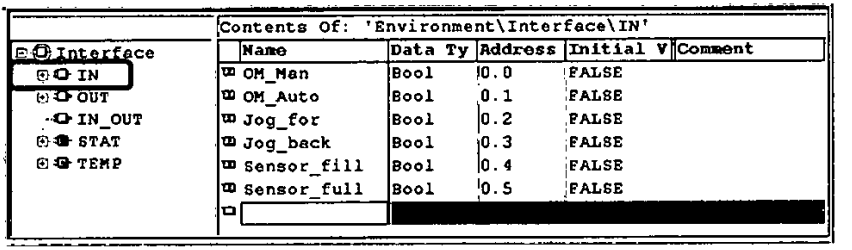
در این مثال لازم است تا پارامترهای قراردادی را مطابق شکل‌های ۴-۹۰ و ۴-۹۱ ایجاد نمایید.

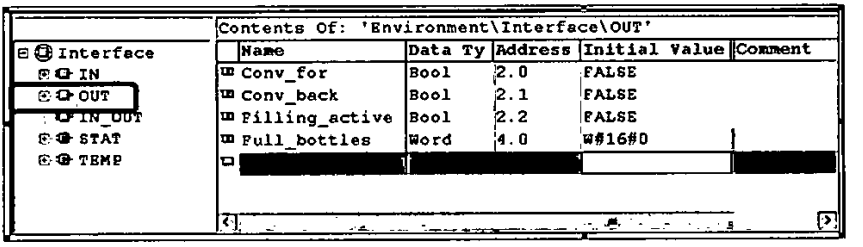




شکل ۴-۹۰ تعریف پارامترهای IN و Out در FB15

تعریف پارامترهای قراردادی در FB16





شکل ۴-۹۱ تعریف پارامترهای In و Out در FB16

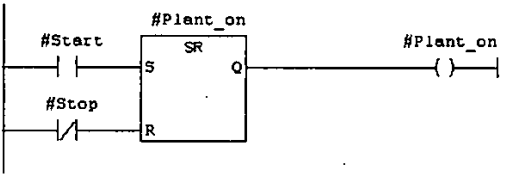
برنامه‌نویسی FB15: پس از تعریف پارامترهای قراردادی در FB15، نوبت به برنامه‌نویسی آن با استفاده از این پارامترها می‌رسد. لازم است برنامه زیر در FB15 پیاده‌سازی شود.

FB15 : Mode_Selection

Comment:

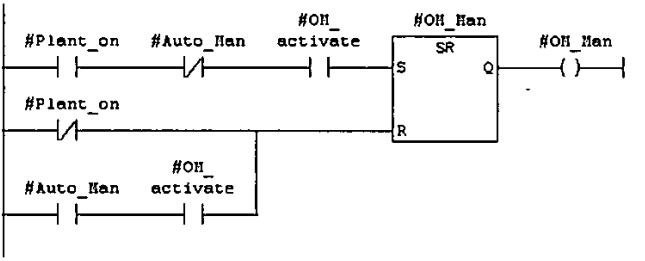
Network 1 : Plant on/off

Comment:



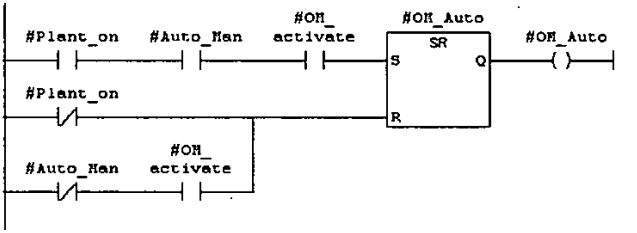
Network 2 : Operating Mode (OM): Manual

Comment:



Network 3 : Operating Mode (OM): Automatic

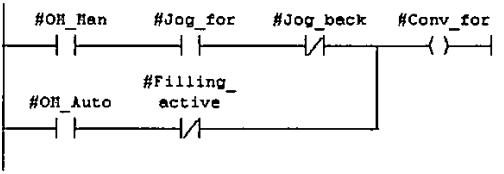
Comment:



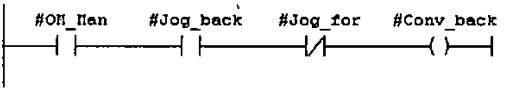
شکل ۴-۹۲

برنامه‌نویسی FB16: پس از تعریف پارامترهای قراردادی، نوبت به برنامه‌نویسی FB16 با استفاده از این پارامترها می‌رسد. لازم است برنامه زیر در FB16 پیاده‌سازی شود.

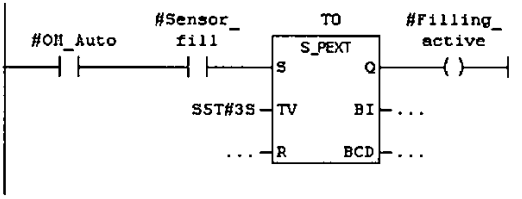
```
FB16 : Title:
Comment:

Network 1: Conveyor Forward
Comment:
```

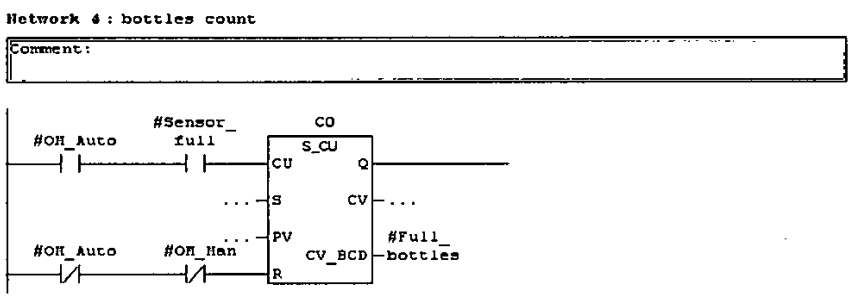



```
Network 2 : Conveyor Backward
Comment:
```




```
Network 3: Start Filling_time
Comment:
```





شکل ۴-۹۳

ذخیره‌سازی و دانلود FB15 و FB16

پس از اینکه برنامه مورد نظر در FB15 و FB16 نوشته شد، باید برنامه نوشته شده را Save و سپس FBها را دانلود نمود. بدیهی است درصورت عدم دانلود FBها و فراخوانی آنها CPU متوقف می‌شود.

تعریف جدول سمبل‌ها

قبل از فراخوانی FBها و آدرس‌دهی آنها لازم است که سمبل‌های مورد نظر را مطابق آدرس‌های حقیقی موجود در پروسه تعریف نمود. شکل ۴-۹۴ سمبول‌های به‌کار رفته در پروسه و آدرس مطلق آنها را نشان می‌دهد.

Symbol Editor - [S7 Program(6) (Symbols) -- PLCbook2_exap3-1\SIMATIC 300(1)\CPU 314C-2 DP]

Symbol Table Edit Insert View Options Window Help

All Symbols

	Statu	Symbol	Address	Data type	Comment
1		Mode_Selection_DB	DB 15	FB 15	
2		Conveyor_Control_DB	DB 16	FB 16	
3		Mode_Selection	FB 15	FB 15	
4		Conveyor_Control	FB 16	FB 16	
5		Start	I 0.0	BOOL	
6		Stop	I 0.1	BOOL	
7		Jog_forward	I 0.2	BOOL	
8		Jog_backward	I 0.3	BOOL	
9		Man/Auto	I 0.4	BOOL	
10		Enter_Mode	I 0.5	BOOL	
11		INI1	I 16.5	BOOL	
12		Filling_Position	I 16.6	BOOL	
13		Counting_Bottles	I 16.7	BOOL	
14		Plant_on	Q 8.1	BOOL	
15		Manual_Mode	Q 8.2	BOOL	
16		Automatic_Mode	Q 8.3	BOOL	
17		Filling_In_progress	Q 9.0	BOOL	
18		Battery Error	Q 9.7	BOOL	
19		Conveyor_forward	Q 20.5	BOOL	
20		Conveyor_backward	Q 20.6	BOOL	
21		Display	QW 12	WORD	
22					

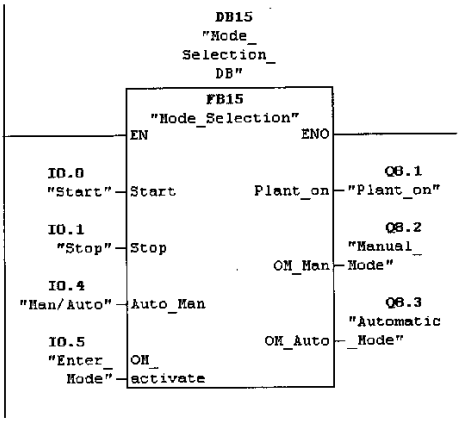
Press F1 to get Help.

شکل ۴-۹۴ آدرس‌های به‌کار رفته در مثال ۴-۱۲

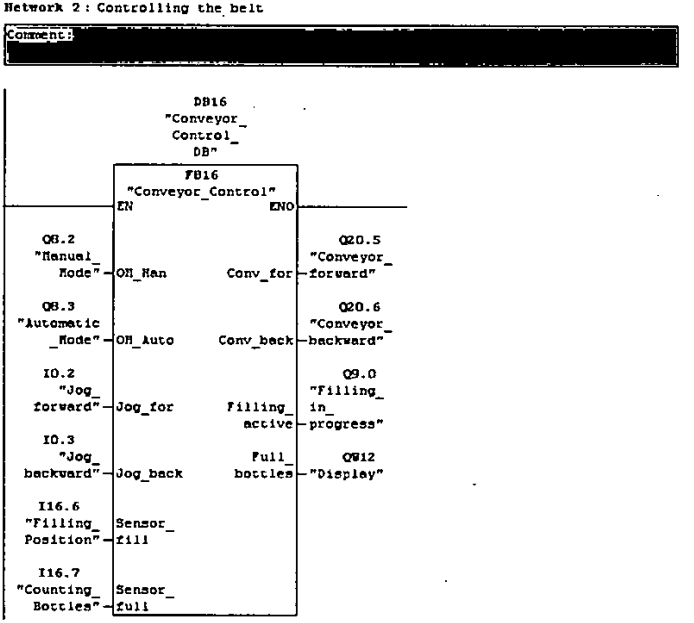
فراخوانی FB15 و FB16 در OB1

پس از دانلود FBهای 15 و 16 باید آنها را در OB1 فراخوانی نموده، به FB15 دیتابلاک۱۵ و به FB16 دیتابلاک ۱۶ را اختصاص داد. همچنین باید آدرس‌های مطلق مورد نظر را به پارامترهای FBها اختصاص داد. برنامه زیر این موضوع را نشان می‌دهد.

```
OB1 : Title:
Comment:

Network 1: Operating mode
Controlling the Operating Mode
```



فصل ۴



شکل ۴-۹۵

توجه: نباید در این مرحله OB1 را دانلود نمود زیرا به‌دلیل اینکه DBها دانلود نشده‌اند CPU متوقف می‌گردد.

دانلود DB15 و DB16

پس از اینکه به FBها دیتابلاک مورد نظر را اختصاص دادیم این دیتابلاک‌ها به‌طور اتوماتیک ایجاد می‌گردند. در این مرحله باید برنامه‌نویس این DBها را نیز به CPU دانلود کند در غیر اینصورت همانطور که اشاره شد CPU متوقف می‌گردد.

دانلود OB1

مرحله آخر انجام این پروژه دانلود OB1 به CPU است. پس از دانلود OB1 می‌توان برنامه را تست و اجرا نمود.

مثال ۴-۱۳:کنترل موتور و ژنراتور در دو بخش فرآیند

در یک فرایند دو بخش آبرسانی و برق رسانی وجود دارد. در سیستم آبرسانی از ۵ عدد پمپ آب استفاده می‌شود که محرک آنها موتور الکتریکی است. این موتورها دارای مشخصات یکسان و منطق کنترل یکسانی می‌باشند.

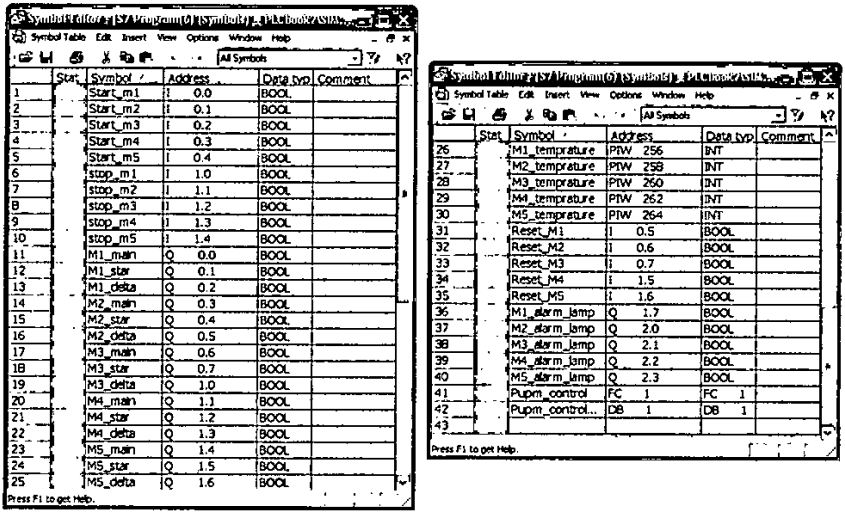
- در سیستم برق‌رسانی از پنج عدد ژنراتور با مشخصات یکسان و منطق کنترل یکسان استفاده می‌شوند.

بخش آبرسانی

در این بخش ۵ موتور سه‌فاز با راه‌اندازی ستاره/مثلث وجود دارد که باید مطابق منطق کنترلی زیر کنترل شوند:

- شستی I0.0 تا I0.4 به‌عنوان استارت موتورهای M1 تا M5 در نظر گرفته شده است. این شستی‌ها از نوع NO (تیغه باز) می‌باشند. در صورتی‌که هرکدام از این شستی فشرده شود، لازم است موتور مربوطه روشن شود.
- شستی I1.0 تا I1.4 به‌عنوان استپ موتورهای M1 تا M5 درنظر گرفته شده است. این شستی‌ها از نوع NC (تیغه بسته) بوده و لازم است در در صورت فعال‌شدن هرکدام از آنها موتور مربوطه متوقف شود.
- هر کدام از موتورها دارای سه خروجی Main، Star و Delta می‌باشند و لازم است در شروع کار موتور خروجی‌های Main و Star روشن شده و پس از ۶ ثانیه خروجی Star خاموش و خروجی Delta روشن شود.
- دمای هر کدام از موتورها توسط یک pt100 که مستقیماً به PLC متصل است اندازه‌گیری شده و به PLC گزارش می‌شود.
- در صورتی‌که دمای هر کدام از موتورها از حد مجاز فراتر رفت (بیش‌تر از ۶۵ درجه) موتور خاموش شده و تا زمانی‌که شستی ریست فشرده نشده است، موتور روشن نشود.
- هنگامی که به‌دلیل افزایش دما از حد مجاز، هر کدام از موتورها خاموش شوند لامپ آلارمی روشن شده و تا زمانی‌که شستی ریست فشرده نشده است، این لامپ خاموش نشود.
- لازم است دمای موتورها در یک خانه از دیتابلاک با فرمت Real ذخیره شود.
- برای برنامه‌نویسی این پروسه می‌توان از FC و DB به‌صورت ترکیبی استفاده نمود.

ابتدا لازم است سمبل‌های به‌کار رفته در این مثال را مطابق شکل ۴-۹۶ ایجاد نمود.

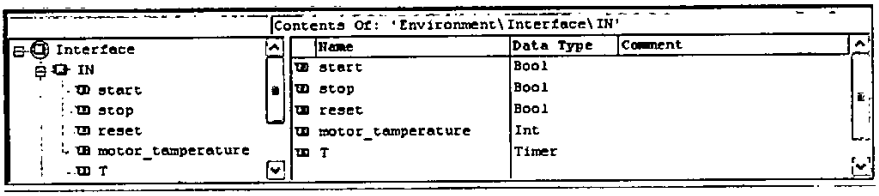


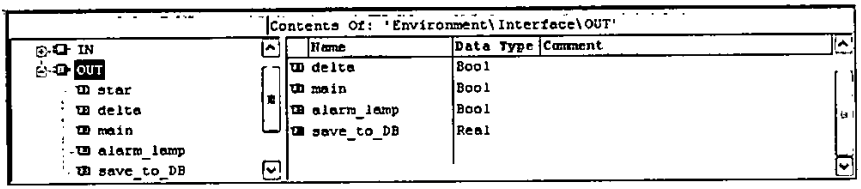
	Stat	Symbol	Address	Data typ	Comment
1		Start_m1	I 0.0	BOOL	
2		Start_m2	I 0.1	BOOL	
3		Start_m3	I 0.2	BOOL	
4		Start_m4	I 0.3	BOOL	
5		Start_m5	I 0.4	BOOL	
6		stop_m1	I 1.0	BOOL	
7		stop_m2	I 1.1	BOOL	
8		stop_m3	I 1.2	BOOL	
9		stop_m4	I 1.3	BOOL	
10		stop_m5	I 1.4	BOOL	
11		M1_main	Q 0.0	BOOL	
12		M1_star	Q 0.1	BOOL	
13		M1_delta	Q 0.2	BOOL	
14		M2_main	Q 0.3	BOOL	
15		M2_star	Q 0.4	BOOL	
16		M2_delta	Q 0.5	BOOL	
17		M3_main	Q 0.6	BOOL	
18		M3_star	Q 0.7	BOOL	
19		M3_delta	Q 1.0	BOOL	
20		M4_main	Q 1.1	BOOL	
21		M4_star	Q 1.2	BOOL	
22		M4_delta	Q 1.3	BOOL	
23		M5_main	Q 1.4	BOOL	
24		M5_star	Q 1.5	BOOL	
25		M5_delta	Q 1.6	BOOL	

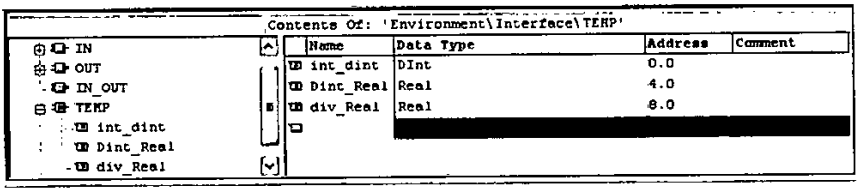
	Stat	Symbol	Address	Data typ	Comment
26		M1_temprature	PIW 256	INT	
27		M2_temprature	PIW 258	INT	
28		M3_temprature	PIW 260	INT	
29		M4_temprature	PIW 262	INT	
30		M5_temprature	PIW 264	INT	
31		Reset_M1	I 0.5	BOOL	
32		Reset_M2	I 0.6	BOOL	
33		Reset_M3	I 0.7	BOOL	
34		Reset_M4	I 1.5	BOOL	
35		Reset_M5	I 1.6	BOOL	
36		M1_alarm_lamp	Q 1.7	BOOL	
37		M2_alarm_lamp	Q 2.0	BOOL	
38		M3_alarm_lamp	Q 2.1	BOOL	
39		M4_alarm_lamp	Q 2.2	BOOL	
40		M5_alarm_lamp	Q 2.3	BOOL	
41		Pupm_control	FC 1	FC 1	
42		Pupm_control...	DB 1	DB 1	
43					

شکل ۴-۹۶ سمبل‌های به‌کار رفته در مثال ۴-۱۳

پس از تعریف سمبل‌ها لازم است ابتدا FC1 را با نام سمبولیک Pump_control ایجاد نموده و سپس پارامترهای قراردادی و متغیرهای محلی Temp را مطابق شکل ۴-۹۷ در آن ایجاد نمود. توجه داشته باشید که در اینجا شماره تایمر به‌عنوان ورودی فانکشن است پس هر بار که فانکشن را صدا می‌زنیم شماره جدیدی به تایمر می‌دهیم بنابراین مشکلی که قبلا در مورد صدا زدن تایمر در فانکشن بیان کردیم در اینجا پیش نخواهد آمد.

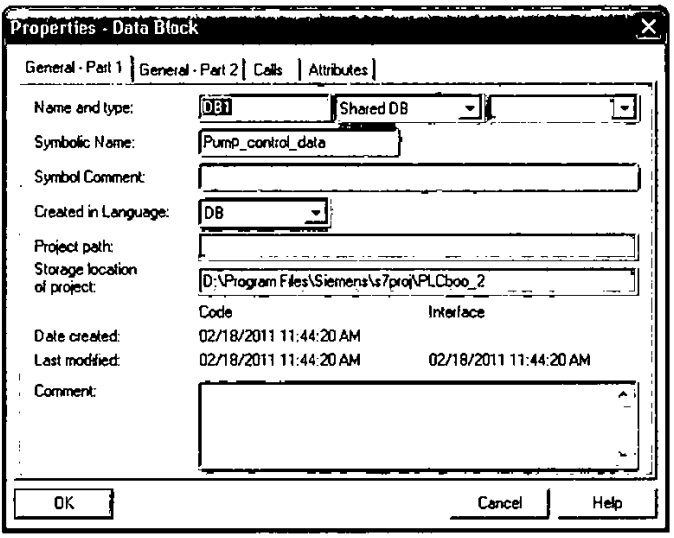






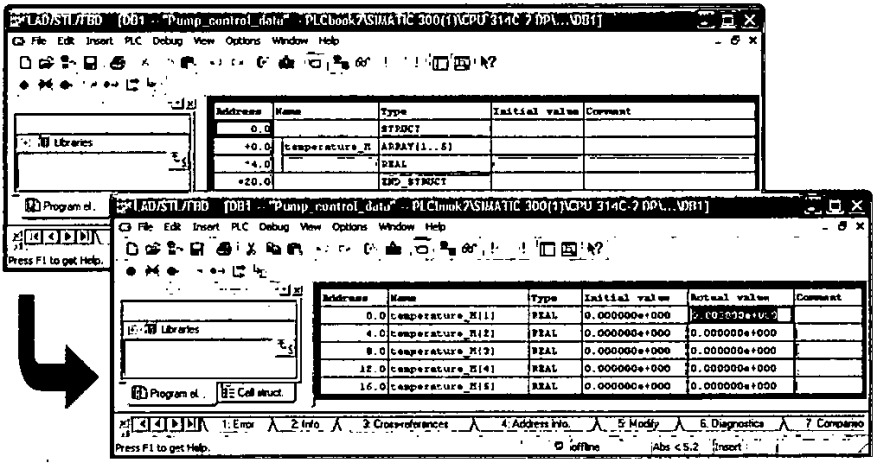
شکل ۴-۹۷ پارامترها و متغیرهای محلی مثال ۴-۱۳

پس از ایجاد FC1 و تعریف متغیرهای محلی و پارامترهای قراردادی در آن، باید DB1 را نیز با نام سمبولیک Pump_Control_data ایجاد نموده و متغیرهای لازم را در آن تعریف کرد. در DB1 نیاز به تعریف ۵ متغیر از نوع Real مربوط به ذخیره‌سازی دمای موتورهای پمپ وجود دارد. از آنجایی که باید ۵ متغیر با تایپ یکسان ایجاد نمود، می‌توان از آرایه استفاده کرد. شکل ۴-۹۸ ایجاد DB1 را نشان می‌دهد.



شکل ۴-۹۸ ایجاد DB1 با نام سمبولیک Pump_Control_data

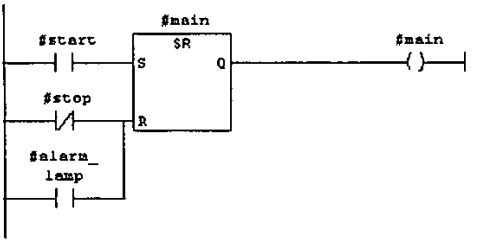
شکل ۴-۹۹ متغیرهای ایجاد شده در DB1 را نشان می‌دهد.



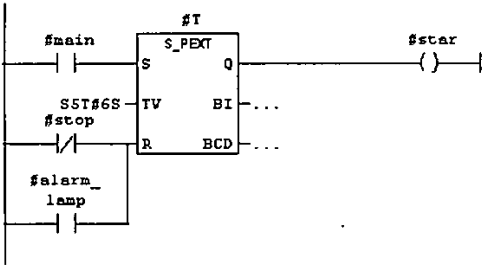
شکل ۴-۹۹ متغیرهای ایجاد شده در DB1

پس از تعریف متغیرها در DB1 نوبت به برنامه‌نویسی FC1 می‌رسد. در این ارتباط باید توجه داشت که منطق کنترل یک بار در FC1 نوشته شده و این بلاک ۵ بار (به تعداد موتورهای پمپ) در OB1 فراخوانی می‌شود. برنامه FC1 را می‌توان مانند برنامه زیر پیاده‌سازی نمود.

```
FC1 : Title:
Comment:

Network 1 : Title:
Comment:
```

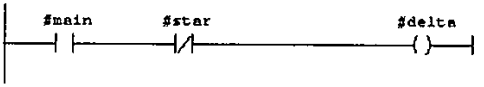



```
Network 2 : Title:
Comment:
```



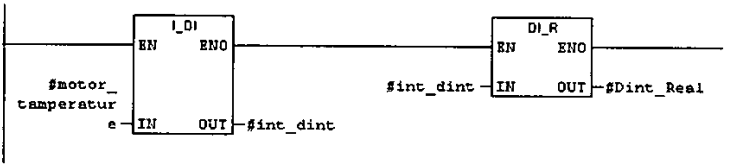
Network 3 : Title:

Comment:



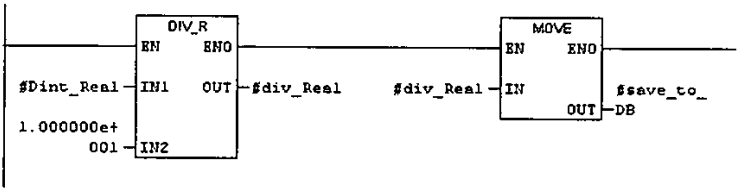
Network 4 : Title:

Comment:



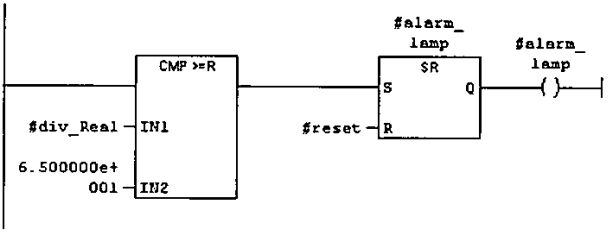
Network 5 : Title:

Comment:



Network 6 : Title:

Comment:



شکل ۴-۱۰۰

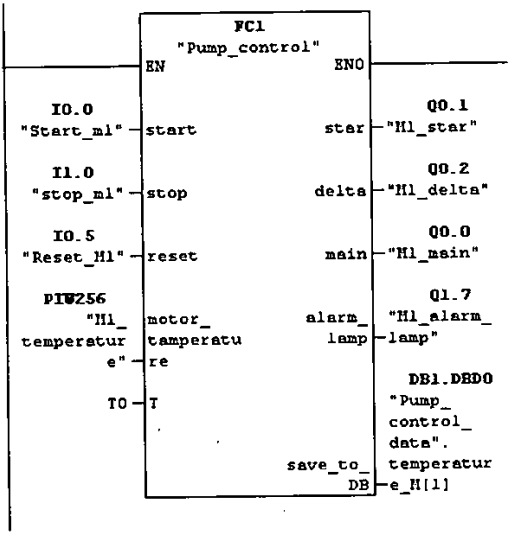
پس از اینکه برنامه فوق در FC1 نوشته شد، باید ابتدا این بلاک را ذخیره‌سازی و دانلود نموده و سپس آنرا در OB1 فراخوانی نمود. در هنگام فراخوانی FC1، در هر فراخوانی آدرس‌های مربوط به یکی از موتورها را به آن اختصاص می‌دهیم.

OB1 : "Main Program Sweep (Cycle)"

Comment:

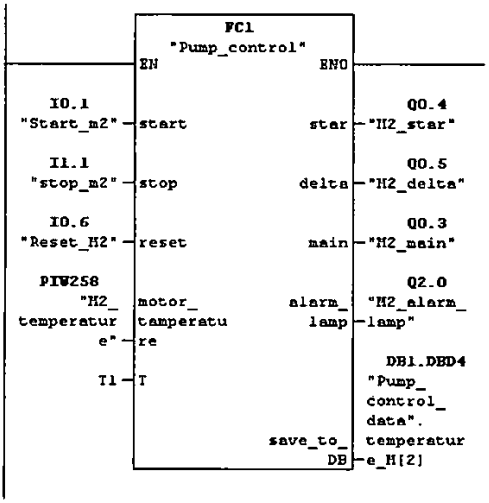
Network 1: M1 pump Control

Comment:



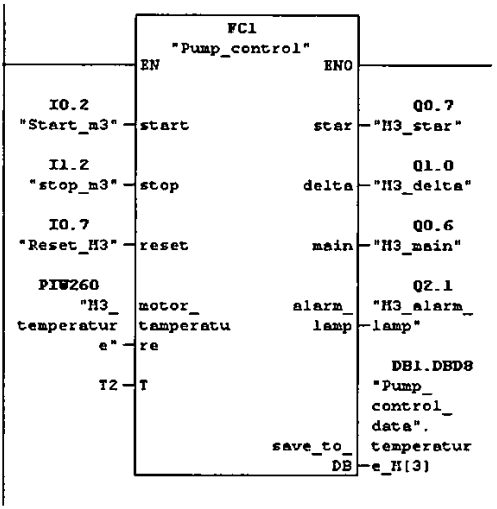
Network 2 : H2 pump Control

Comment:



Network 3 : H3 pump Control

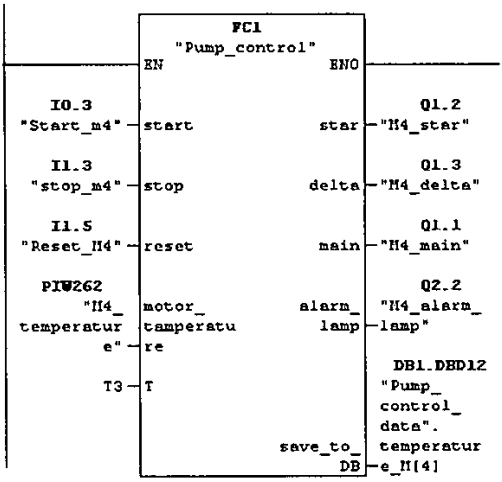
Comment:



فصل ۴

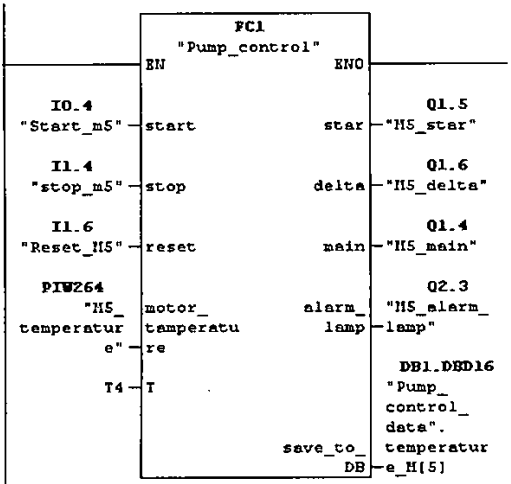
Network 4 : M4 pump Control

Comment:



Network 5 : M5 pump Control

Comment:



شکل ۴-۱۰۱

بخش برق‌رسانی (کنترل ژنراتورها)

در ارتباط با کنترل ژنراتورها باید به نکات زیر توجه نمود:

- هر ژنراتور با یک شستی استارت (NO) روشن و با یک شستی استپ (NC) خاموش شود.
- برای روشن شدن ژنراتور باید دما کمتر از ۷۰ درجه باشد.
- دما و سرعت هر ژنراتور باید در حافظه (DB) ذخیره شوند. دما از آدرس‌های PIW 752 الی PIW 760 دریافت شده و سرعت از MW0 الی MW8 دریافت می‌گردد.
- اگر دمای هر کدام از ژنراتورها بیشتر از ۷۰ درجه شد، ژنراتور خاموش و لامپ آلارم مجزایی روشن شود.
- اگر دما از حد خطا کاهش پیدا نمود، تا زمانی‌که شستی ریست مربوط به ژنراتور مورد نظر فشرده نشده است لامپ خاموش نشده و ژنراتور استارت نشود.

برای این قسمت از پروسه می‌توان از FB استفاده نمود. ابتدا سمبل‌های مورد نیاز این پروژه را به‌شرح زیر ایجاد می‌نماییم.

	Status	Symbol	Address	Data type	Comment
1		Generator_Control	FB 1	FB 1	
2		Start_G1	I 124.0	BOOL	
3		Start_G2	I 124.1	BOOL	
4		Start_G3	I 124.2	BOOL	
5		Start_G4	I 124.3	BOOL	
6		Start_G5	I 124.4	BOOL	
7		Reset_G1	I 124.5	BOOL	
8		Reset_G2	I 124.6	BOOL	
9		Reset_G3	I 124.7	BOOL	
10		stop_G1	I 125.0	BOOL	
11		stop_G2	I 125.1	BOOL	
12		stop_G3	I 125.2	BOOL	
13		stop_G4	I 125.3	BOOL	
14		stop_G5	I 125.4	BOOL	
15		Reset_G4	I 125.5	BOOL	
16		Reset_G5	I 125.6	BOOL	
17		G1_Speed	MW 0	INT	
18		G2_Speed	MW 2	INT	

	Status	Symbol	Address	Data type	Comment
19		G3_Speed	MW 4	INT	
20		G4_Speed	MW 6	INT	
21		G5_Speed	MW 8	INT	
22		G1_temperature	PIW 752	INT	
23		G2_temperature	PIW 754	INT	
24		G3_temperature	PIW 756	INT	
25		G4_temperature	PIW 758	INT	
26		G5_temperature	PIW 760	INT	
27		G1_ON	Q 124.0	BOOL	
28		G2_ON	Q 124.1	BOOL	
29		G3_ON	Q 124.2	BOOL	
30		G4_ON	Q 124.3	BOOL	
31		G5_ON	Q 124.4	BOOL	
32		G1_alarm_lamp	Q 124.7	BOOL	
33		G2_alarm_lamp	Q 125.0	BOOL	
34		G3_alarm_lamp	Q 125.1	BOOL	
35		G4_alarm_lamp	Q 125.2	BOOL	
36		G5_alarm_lamp	Q 125.3	BOOL	

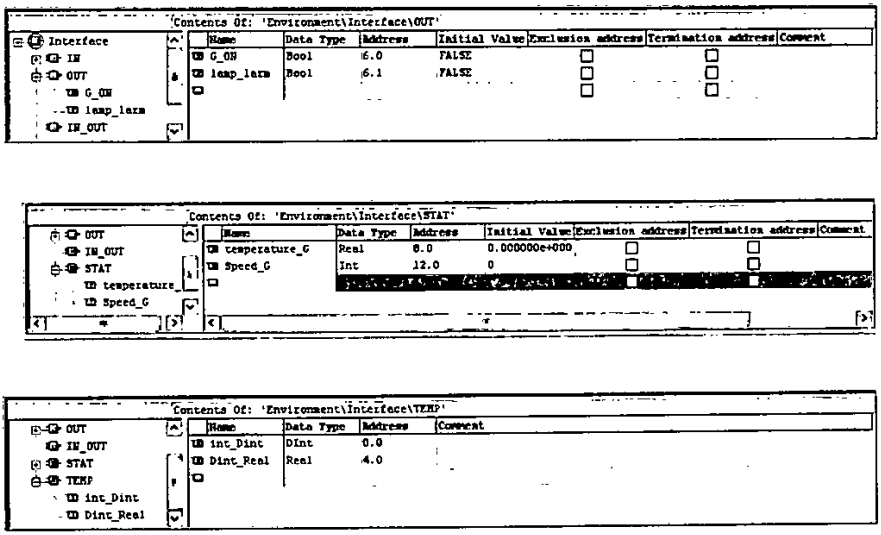
شکل ۴-۱۰۲ سمبل‌های مورد استفاده در بخش کنترل ژنراتور

پس از تعریف سمبل‌های سراسری، باید پارامترهای قراردادی و متغیرهای محلی مورد نیاز را در FB1 ایجاد نمود. شکل ۴-۱۰۳ متغیرهای لازم ایجاد شده در FB1 را نشان می‌دهد.

Contents Of: 'Environment\Interface\IN'

Interface
- IN
  - Start
  - Stop
  - Reset
  - Temperature

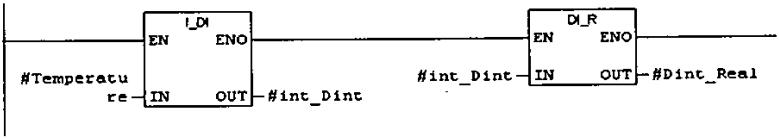
Name	Data Type	Address	Initial Value	Exclusion address	Termination address	Comment
Start	Bool	0.0	FALSE			
Stop	Bool	0.1	FALSE			
Reset	Bool	0.2	FALSE			
Temperature	Int	2.0	0			
Speed	Int	4.0	0			



شکل ۴-۱۰۳ پارامترها و متغیرهای محلی در FB1

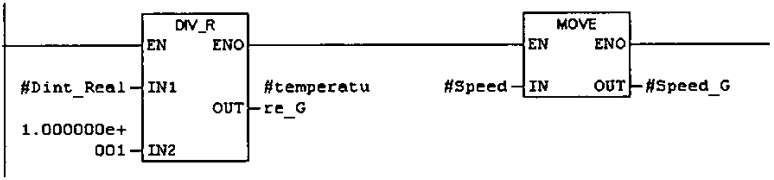
پس از اینکه متغیرهای لازم در FB1 ایجاد شدند، باید برنامه مورد نظر را در FB1 پیاده‌سازی نمود. بنابراین مطابق منطق کنترلی بیان شده، برنامه مورد نظر را در FB1 می‌نویسیم. تحلیل این برنامه بر عهده خواننده می‌باشد.

```
FB1 : Generator Control
Comment:

Network 1: Title:
Comment:
```



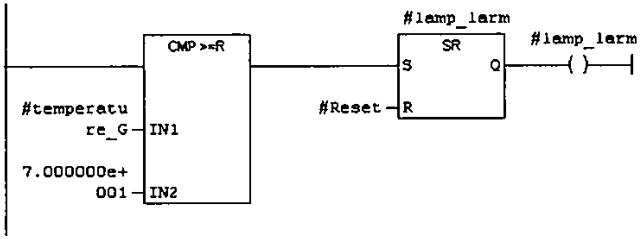
Network 2 : Title:

Comment:



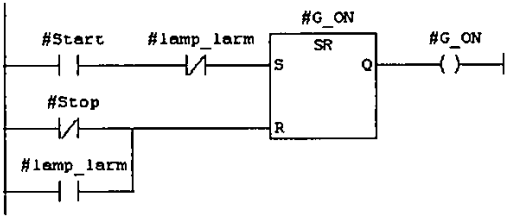
Network 3 : Title:

Comment:



Network 4 : Title:

Comment:



شکل ۴-۱۰۴

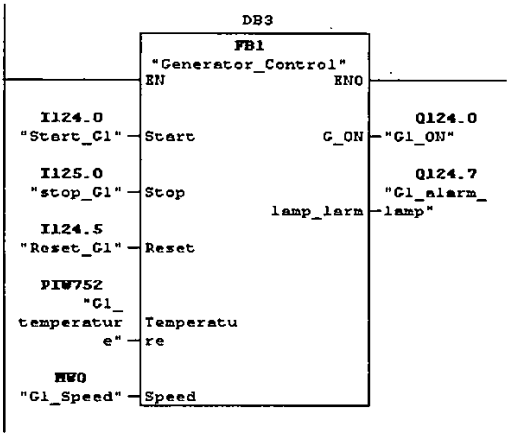
پس از اینکه برنامه مورد نظر در FB1 نوشته شد، باید برنامه را ذخیره و دانلود نمود. سپس به OB1 رفته و FB1 را پنج بار از درون OB1 فراخوانی می نماییم. در هر فراخوانی یک دیتابلاک مجزا به FB1 اختصاص داده و آدرس‌های مربوط به یکی از ژنراتورها را وارد می نماییم.

OB1 : "Main Program Sweep (Cycle)"

Comment:

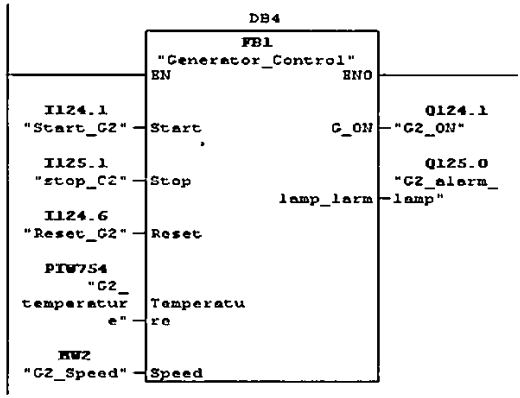
Network 1 : G1 control

Comment:



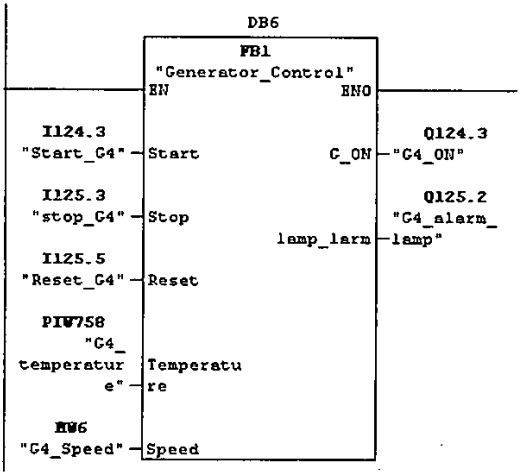
Network 2 : G2 control

Comment:



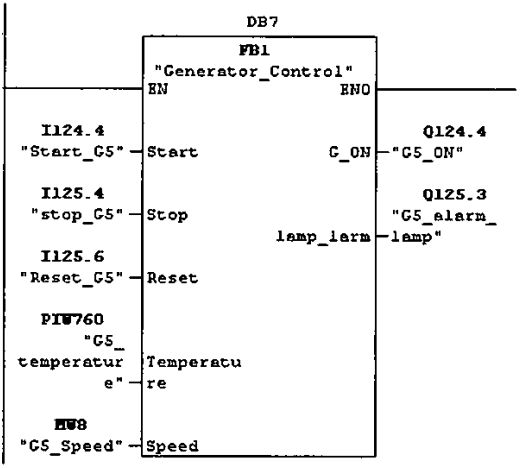
Network 4 : G4 control

Comment:



Network 5 : G5 control

Comment:



شکل ۴-۱۰۵

پس از فراخوانی FBها، لازم است تا ابتدا همه دیتابلاک‌ها به CPU دانلود شده و سپس OB1 نیز دانلود شود.

۴-۷ استفاده از مدل Multi Instance

انواع مدل‌های Instance DB

به‌طور کلی دو روش برای اختصاص دیتابلاک اختصاصی یک FB وجود دارد:

۱- استفاده از حالت Instance Model

۲- استفاده از حالت Multi Instance

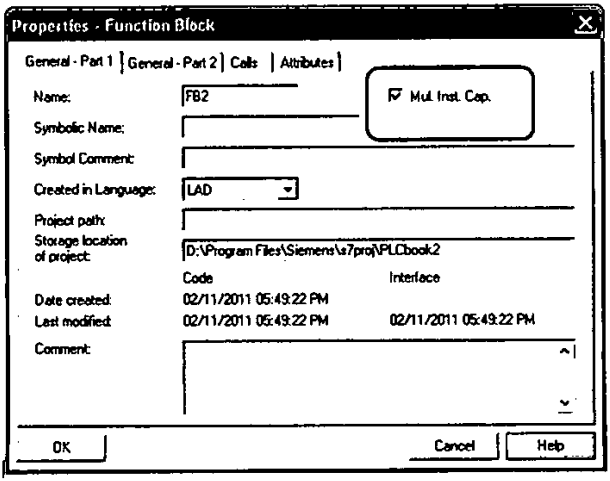
Instance Model

در این روش، در هر بار فراخوانی FB یک دیتابلاک خاص به آن اختصاص داده می‌شود. همچنین در صورتی‌که چندین FB فراخوانی شوند، به هر کدام یک دیتابلاک خاص اختصاص داده می‌شود. این همان روشی است که تا اینجای کار ما مطابق آن عمل می‌کردیم.

Multi-instance Model

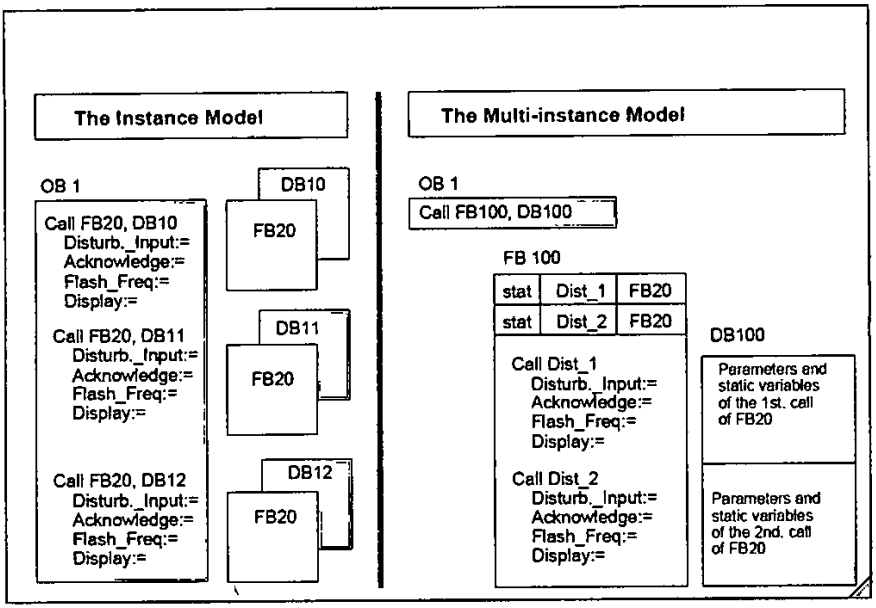
با استفاده از قابلیت Multi Instance، می‌توان از یک دیتابلاک به‌عنوان دیتابلاک اختصاصی چندین FB استفاده نمود. یعنی برای چندین FB یک دیتابلاک اختصاصی مشترک قرار داد که دیتاهای همه FBها در آن ذخیره شوند.

برای این منظور لازم است تا در هنگام ایجاد FB، مطابق شکل ۴-۱۰۶گزینه Mul . Ins. Cap علامت زده شود.



شکل ۴-۱۰۶ انتخاب حالت Multi Instance برای یک FB

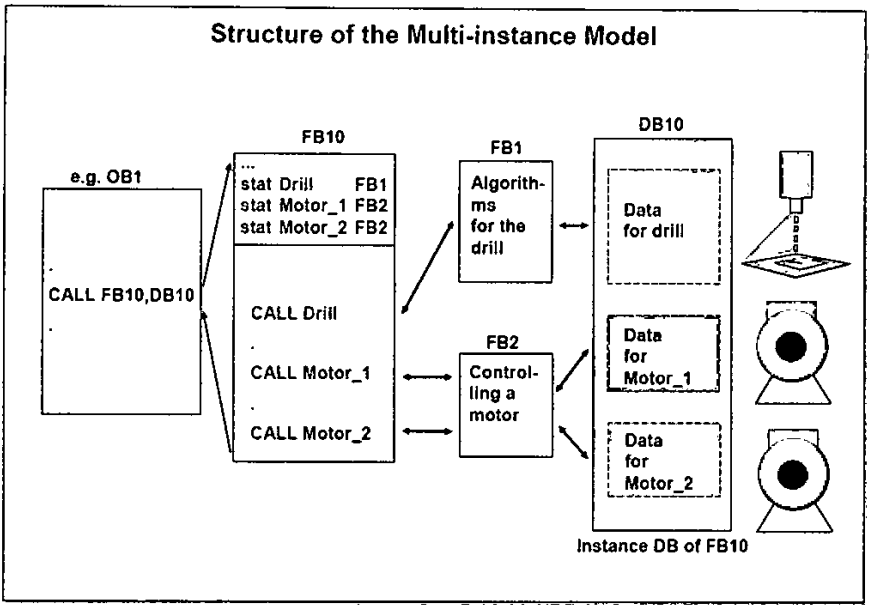
شکل ۴-۱۰۷ دو مدل Instance و Multi Instance را نشان می‌دهد.



شکل ۴-۱۰۷ مدل Instance و Multi Instance

همانطور که در شکل ۴-۱۰۷ مشخص است، در مدل Multi Instance نیاز به یک FB جهت مدیریت فراخوانی سایر FBها وجود دارد. مثلاً در شکل ۴-۱۰۷، FB100 جهت مدیریت فراخوانی FB20 استفاده شده است. برای هر بار فراخوانی FB20، باید یک متغیر Static از نوع FB20 در قسمت Declaration در FB100 تعریف نمود. سپس می‌توان در برنامه FB100، این متغیرهای Static را فراخوانی نمود. در اینصورت دیتاهای مربوط به همه فراخوانی‌ها در DB100 ذخیره می‌شوند.

شکل ۴-۱۰۸ نمونه‌ای دیگر از مدل Multi Instance را نشان می‌دهد که در آن، دیتاهای مربوط به سه تجهیز در یک دیتابلاک مشترک ذخیره می‌شوند. همانطور که در شکل مشخص است، از FB10 برای مدیریت فراخوانی‌ها استفاده شده است و سپس دوبار FB1 و یک بار FB2 از درون آن فراخوانی شده و دیتاهای آنها در DB10 قرار گرفته است.



شکل ۴-۱۰۸ نمونه‌ای از مدل Multi Instance

مزایای مدل Multi Instance

استفاده از مدل Multi Instance دارای مزایای زیر می‌باشد:

- با استفاده از این روش نیاز به استفاده از دیتابلاک‌های اختصاصی کمتری در برنامه وجود دارد.
- برنامه‌نویسی نیاز به مدیریت دیتابلاک‌های اختصاصی برای FBهای مختلف ندارد و این‌کار به‌طور اتوماتیک توسط FB مدیریت، کنترل می‌شود.
- برای چندین FB می‌توان از یک دیتابلاک اختصاصی مشترک استفاده نمود.

نکات مهم

- در این روش نمی‌توان درون FB از آدرس‌های مستقیم (I,Q) استفاده نمود.
- استفاده از سیگنال‌های مربوط به پروسه فقط از طریق پارامترهای FB امکان‌پذیر می‌باشد.

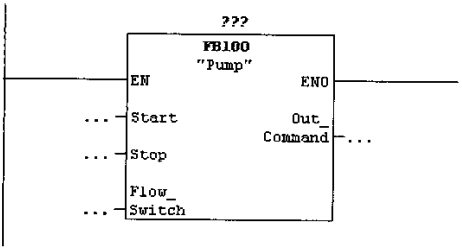
مثال ۴-۱۴: صدا زدن FB در FB با استفاده از ویژگی Multi Instance

در مثال‌هایی که تا اینجا ارائه گردید، صدا زدن FC در FB و صدا زدن FB در FC تست گردد. در اینجا به بررسی صدا زدن FB در FB می‌پردازیم و به نکات مربوط به آن اشاره می‌کنیم.

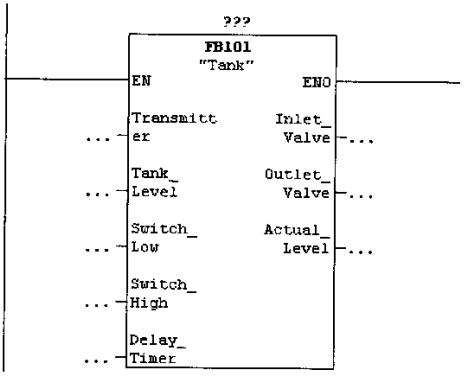
فرض کنید FB100 برای کنترل موتور پمپ و FB 101 برای کنترل مخزن طراحی شده است. اکنون می‌خواهیم FB200 را برای کنترل یک مجموعه که شامل مخزن و پمپ است به‌کار ببریم. اگر پس از ایجاد کردن FB200 آن را باز کنیم و FB100 و FB101 را مانند شکل ۴-۱۰۹ در آن صدا بزنیم همانطور که می‌بینیم هر FB نیاز به یک DB دارد. این موضوع ممکن است در عمل کار را پیچیده کند زیرا اگر کاربر پس از کپی کردن FB200 و FB100 و FB101 به پروژه دیتابلاک‌هایی مانند DBهای داخل FB200 داشته باشد بایستی یا آنها را تغییر دهد یا دیتا بلاک‌های به‌کار رفته در برنامه FB200 را عوض کند.

FB200 : Title:

Network 1: Title:



Network 2: Title:



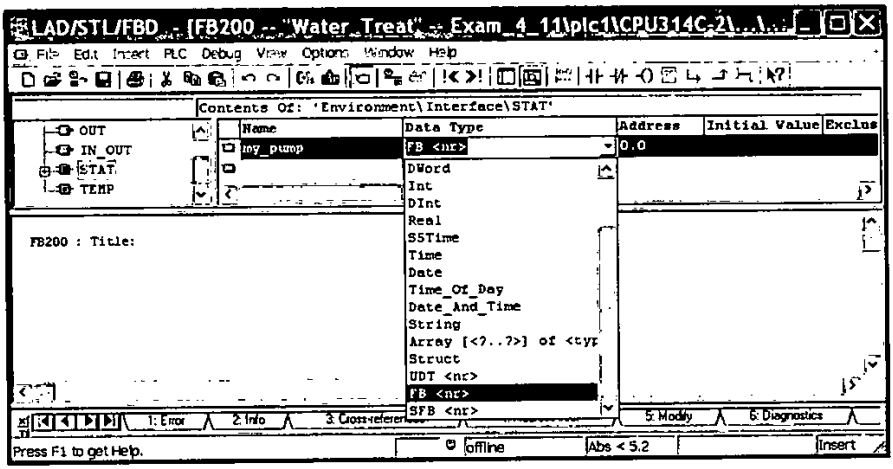
شکل ۴-۱۰۹ روش استفاده از Instance DB در مثال ۴-۱۴

برای رفع این مشکل می‌توان کاری کرد که FB100 و FB101 در هنگام فراخوانی نیاز به دیتابلاک خاص نداشته باشند و هر دو از دیتا بلاک مربوط به FB200 استفاده کنند. به این ترتیب کاربر فقط کافیست با فراخوانی FB200 یک شماره DB به آن اختصاص دهد و نیازی به اختصاص DB به FBهای صدا زده شده در داخل FB200 ندارد.

برای انجام این کار مراحل زیر بایستی دنبال شود:

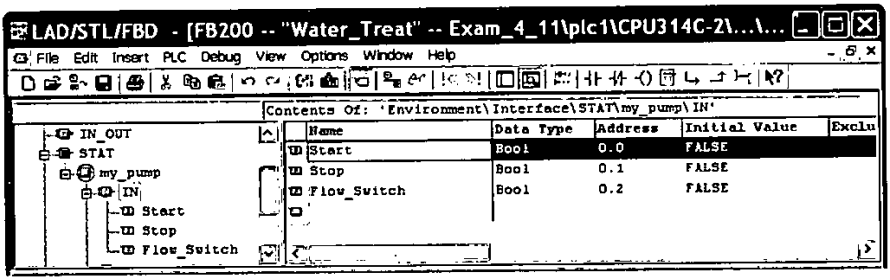
- ابتدا FB100 و FB101 را کامل کرده و ذخیره می‌کنیم.

- FB200 را ایجاد کرده و در بخش پارامترهای STAT یک پارمتر مانند شکل ۴-۱۱۰ با نام دلخواه تعریف کرده و نوع دیتا را FB <nr> انتخاب می‌کنیم.



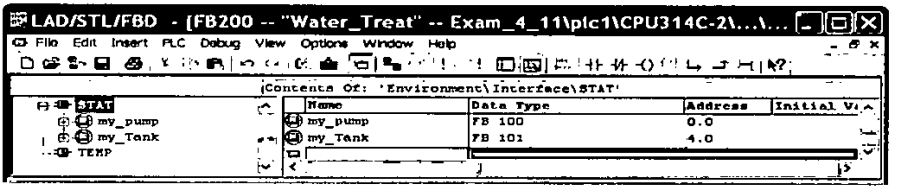
شکل ۴-۱۱۰ روش تعریف FB100 به‌عنوان متغیر استاتیک در مثال ۴-۱۴

- به‌جای nr شماره 100 را می‌نویسیم پس از انجام این کار مشاهده خواهیم کرد که متغیر تعریف شده در زیر مجموعه STAT به‌صورت یک Struct ظاهر می‌شود. ساختار این Struct دقیقا از بخش Interface مربوط به FB100 گرفته شده است.



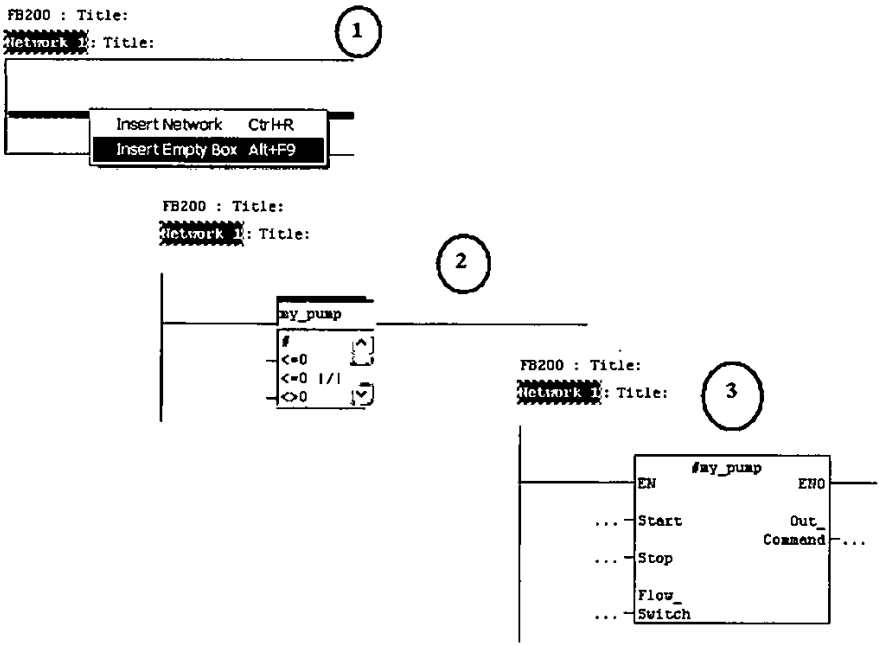
شکل ۴-۱۱۱ استراکچر FB پس از تعریف به‌صورت متغیر استاتیک

- مرحله قبل را برای FB101 تکرار می‌کنیم.



شکل ۴-۱۱۲ تعریف FB101 به‌عنوان متغیر استاتیک در مثال ۴-۱۴

- پس از تعریف FB100 و FB101 به‌صورت یک متغیر استاتیک، برای صدا زدن آنها در برنامه‌ی FB200 نبایستی به‌صورت معمولی آنها را از پنجره Program Element صدا زد بلکه بایستی در Network مورد نظر یک Empty Box وارد نمود و نام تعیین شده برای متغیر استاتیک را در بالای آن تایپ کرد. مراحل کار در شکل ۴-۱۱۳ نشان داده شده است.

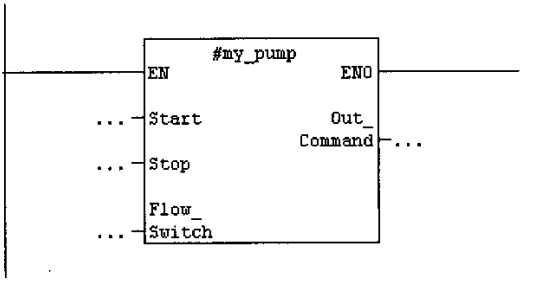


شکل ۴-۱۱۳ مراحل فراخوانی FB تعریف شده بعنوان متغیر استاتیک

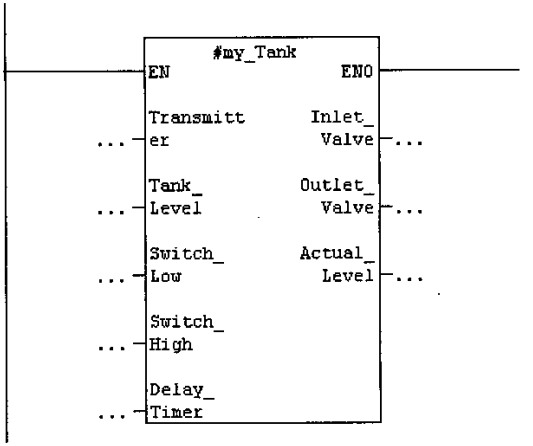
همانطور که دیده می‌شود در مرحله سه فانکشن‌بلاک FB100 بدون اینکه در بالای آن رنگ قرمز ظاهر شود وارد شده است.

مراحل سه‌گانه قبل را برای بلاک FB101 نیز تکرار می‌کنیم. نهایتاً برنامه زیر را در FB200 خواهیم داشت.

```
FB200 : Title:
Network 1: Title:
```

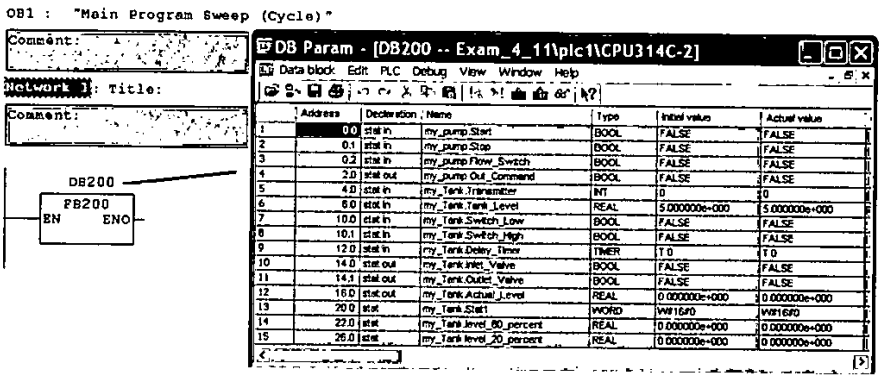



```
Network 2: Title:
```



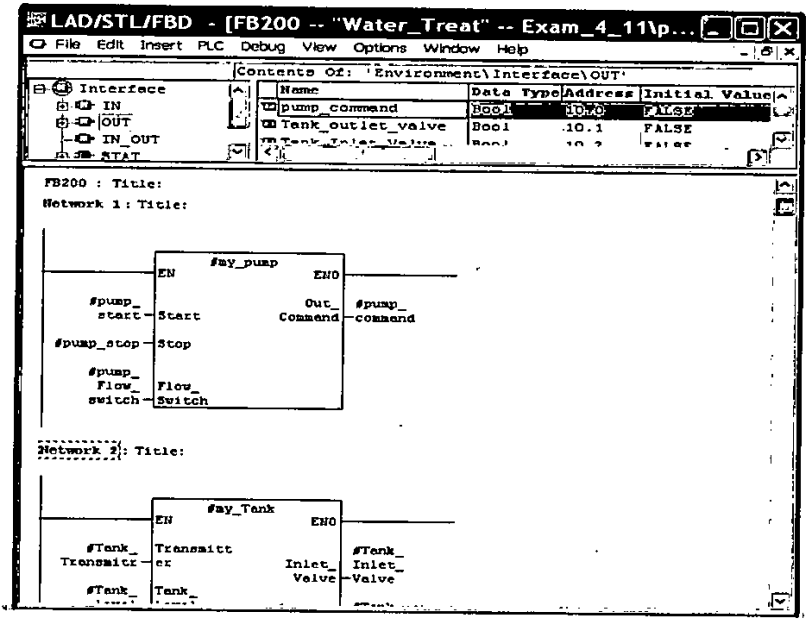
شکل ۴-۱۱۴ فراخوانی FBهای تعریف شده به‌عنوان متغیر استاتیک در مثال ۴-۱۴

اگر در همین شرایط FB200 را ذخیره کرده و در OB1 صدا بزنیم ودیتا بلاک DB200 را به آن اختصاص بدهیم خواهیم دید که تمام پارامترهای مربوط به FB100 و FB101 در دیتا بلاک مربوط به FB200 وارد شده است.



شکل ۴-۱۱۵ دیتابلاک استفاده شده توسط چند FB در مثال ۴-۱۴

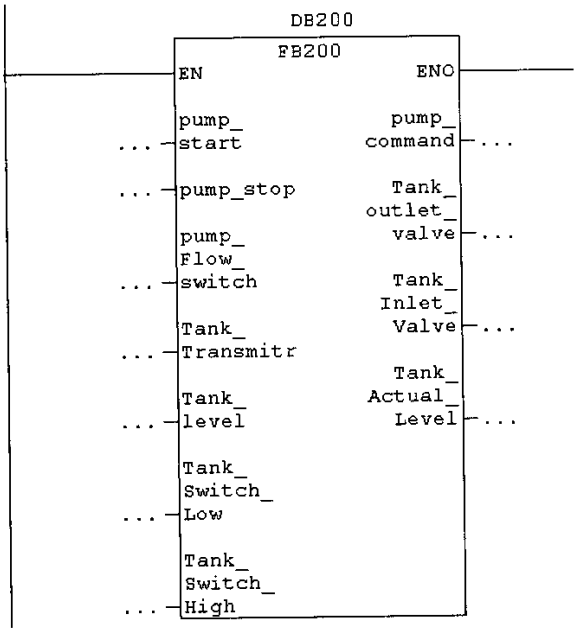
در عمل برای اینکه بتوان ورودی و خروجی‌های دو FB100 و FB101 را از طریق FB200 در اختیار کاربر قرار داد، در FB200 پارامترهای ورودی و خروجی مشابه با ورودی و خروجی FB100 و FB101 را تعریف کرده و مانند شکل ۴-۱۱۶ به آنها اختصاص می‌دهند.



شکل ۴-۱۱۶ ادامه برنامه مثال ۴-۱۴

در این شرایط اگر FB200 در محیط OB1 فراخوان شود، شکل ۴-۱۱۷ را خواهیم داشت که همه پارامترها در دسترس هستند.

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
```



شکل ۴-۱۱۷ فراخوانی FB200 در OB1 در مثال ۴-۱۴

تغییر در ساختار FB در حین کار

در بحث مربوط به FC تغییر در FC در حین کار مورد بحث قرار گرفت. FB از برخی جنبه‌ها رفتاری متفاوت دارد که در این قسمت حالت‌های مختلف آن مورد بحث قرار می‌گیرد. به‌طور کلی تغییر در FB می‌تواند یکی از حالات زیر باشد:

- تغییر در برنامه FB
- تغییر در متغیرهای Temp
- تغییر در متغیرهای Stat
- تغییر در پارامترهای ورودی / خروجی

تغییر در برنامه FB در حین کار

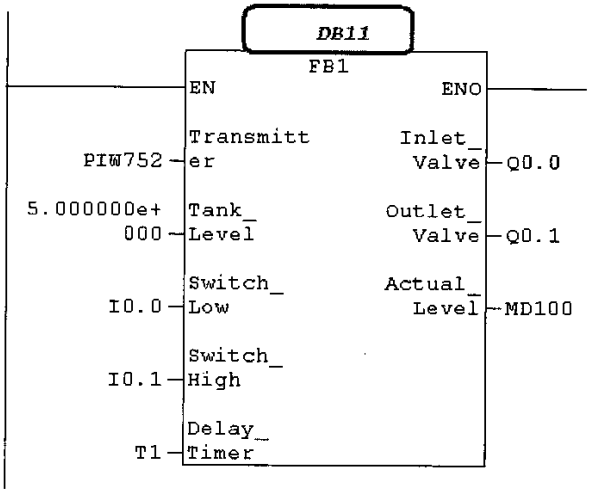
اگر FB قبلاً ایجاد و برنامه نویسی و دانلود شده و CPU در حال اجرای آن است، در این شرایط در FB پوشه offline تغییری در برنامه ایجاد و دانلود کنیم اگر برنامه به‌درستی نوشته شده باشد هیچ مشکلی برای CPU پیش نخواهد آمد و در سیکل جدید FB جدید را اجرا خواهد نمود.

تغییر در متغیرهای TEMP

این حالت کاملاً مشابه تغییر متغیرهای TEMP مربوط به FC است که قبلاً مورد بحث قرار گرفت.

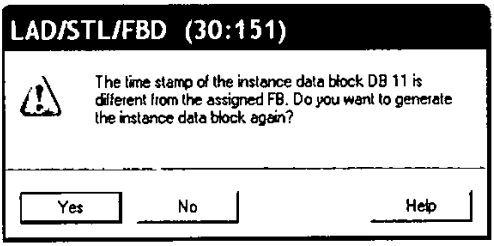
تغییر در متغیرهای STAT

اگر تغییری در متغیرهای Static یک FB داده شود، مثلاً یک STAT جدید اضافه شود؛ اگر بدون اینکه این متغیر در برنامه FB استفاده شود FB را ذخیره و دانلود کنیم مشکلی برای CPU پیش نخواهد آمد، ولی اگر بلاک ماقبل که FB در آن صدا زده شده است را باز کنیم مانند شکل ۴-۱۱۸ می‌بینیم که DB بالای بلاک به رنگ قرمز در آمده است.



شکل ۴-۱۱۸ اشکال ناشی از تغییر در ساختار متغیرهای STAT

در این شرایط اگر روی DB که به رنگ قرمز درآمده کلیک کنیم پیغام زیر ظاهر می‌شود که نیاز به Update شدن DB را بیان می‌کند. با تایید این پیام DB اصلاح شده و به رنگ عادی در می‌آید و نیازی به دانلود نیز ندارد.

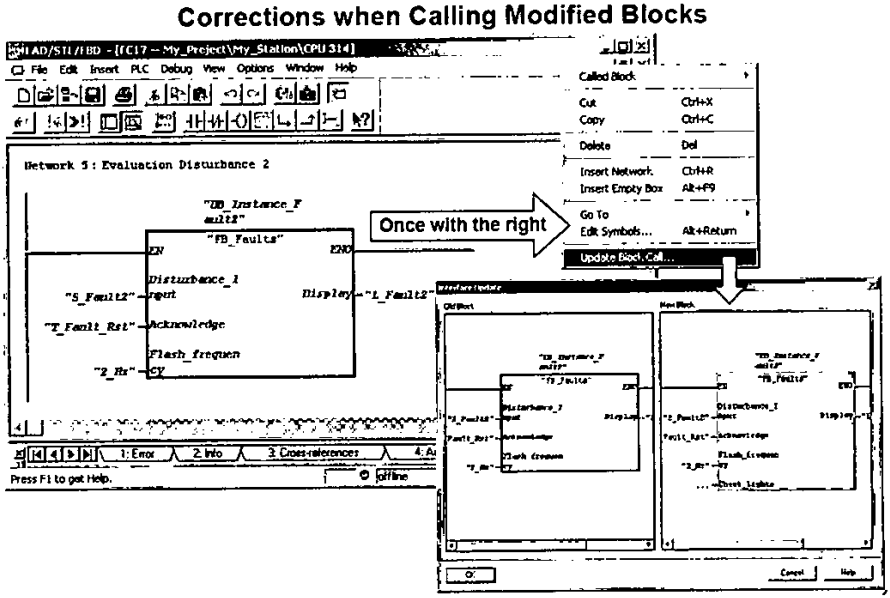


شکل ۴-۱۱۹ اخطار ناشی از نیاز Update شدن DB

اگر از متغیر STAT ایجاد شده در برنامه FB استفاده شود به محض دانلود FB منجر به فالت روی CPU شده و در بافر پیغام Area Length Error را خواهیم داشت. در این حالت بایستی بلاک ماقبل را باز کرده و به روش قبلی آن را Update کنیم و DB را نیز دانلود کنیم تا مشکل برطرف گردد.

تغییر در IN/OUT

اگر پارامترهای ورودی و خروجی FB را تغییر دهیم، پس از دانلود منجر به فالت روی CPU خواهد شد. در این شرایط اگر بلاک ماقبل را باز کنیم کل FB با رنگ قرمز نشان داده می‌شود. برای رفع مشکل شبیه توضیحاتی که برای FC داده شد می‌توان روی بلاک مورد نظر راست نموده و گزینه Update Block Call را انتخاب نمود. در اینصورت در کادری شکل‌حالت‌قبل‌بلاک و حالت فعلی بلاک نمایش داده می‌شود. با کلیک روی گزینه OK حالت جدید بلاک انتخاب می‌شود.



شکل ۴-۱۲۰ اشکال ناشی از تغییر در ساختار پارامترهای ورودی و خروجی FB

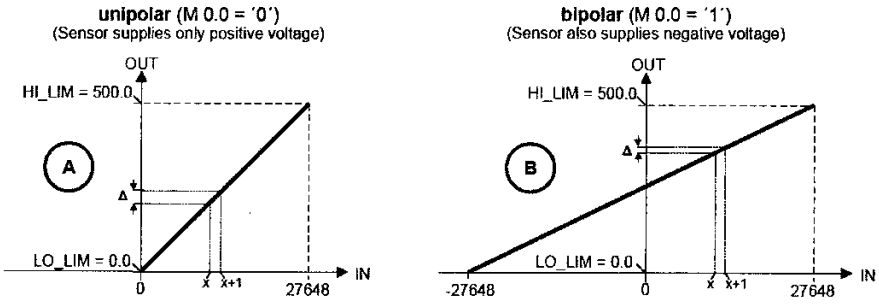
پس از اتمام این کار روی DB که هنوز به رنگ قرمز است یکبار کلیک کرده تا Update شود. سپس FB و DB و بلاک ماقبل را دانلود می‌کنیم تا مشکل برطرف گردد.

۴-۷ پرسش و تحقیق

در مورد بلاک‌های برنامه‌نویسی S5 تحقیق کرده و آنها را با بلاک‌های برنامه‌نویسی S7 مقایسه کنید.

۴-۸ تمرین

تمرین ۱: مشابه FC105 فانکشن Scale را با توجه به نمودارهای زیر طراحی و برنامه نویسی کنید.

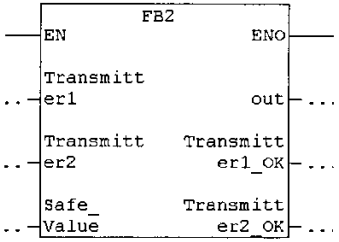


شکل ۴-۱۲۱ شکل مربوط به تمرین ۱

تمرین ۲: فانکشنی بنویسید که برای یک کارت آنالوگ ۸ ورودی حدود Scale هر کانال را بگیرد و نتایج Scale شده را در هشت خروجی نشان دهد.

تمرین ۳: فانکشن بلاکی مانند شکل ۴-۱۲۲ طراحی کنید که آدرس دو ترانسمیتر فشار نصب شده روی مخزن و مقدار Safe را در ورودی بگیرد آنگاه اگر:

- هر دو سیگنال سالم بود، میانگین آنها را به خروجی Out بدهد.
- اگر یکی مشکل داشت، مقدار سالم را به خروجی بفرستد.
- اگر هر دو مشکل داشت، مقدار Safe_Value را به خروجی بفرستد.
- وضعیت ترانسمیترها در خروجی FB نشان داده شود.



شکل ۴-۱۲۲ شکل مربوط به تمرین ۳

فصل ۴

فصل ۵

آشنایی با SFC وSFB

در این فصل ضمن معرفی کلی SFC و SFB، روش کار با SFC46، تایمرها و کانترهای IEC و چگونگی استفاده از آنها در برنامه‌نویسی تشریح می‌شود.

چکیده مطالب

- SFC و SFB فانکشن‌های سیستمی هستند.
- فانکشن‌های سیستمی توسط سازنده تهیه شده و در CPU موجود هستند.
- SFC شبیه FC و SFB شبیه FB فراخوان می‌شود بنابراین SFB نیاز به DB دارد.
- برای فراخوانی فانکشن‌های سیستمی از کتابخانه از زیر مجموعه Standard Library استفاده می‌شود.
- SFC46 هرجا فراخوان شود منجر به توقف CPU می‌گردد.
- SFC47 در پردازش برنامه تاخیر ایجاد می‌کند.
- SFC1 تاریخ و زمان CPU را بر می‌گرداند.
- SFB0 و SFB1 و SFB2 کانترهای IEC هستند که نسبت به کانترهای معمولی قابلیت‌های بیشتری دارند.
- SFB3 و SFB4 و SFB5 تایمرهای IEC هستند که نسبت به تایمرهای معمولی قابلیت‌های بیشتری دارند.
- تشریح SFC و SFBها در کتاب سطح تکمیلی آمده است.

فصل ۵

۵-۱ مقدمه

همانطور که قبلاً اشاره شد، در CPU گروهی از بلاک‌ها تحت عنوان بلاک‌های سیستمی وجود دارند که هر کدام برای انجام عمل خاصی به‌کار می‌روند. این بلاک‌ها از قبل توسط سازنده برنامه‌نویسی شده و کاربر می‌تواند ضمن فراخوانی آنها، تنظیمات آنها را انجام دهد. این بلاک‌ها شامل دو دسته زیر هستند:

- SFC به معنی System Function
- SFB به‌معنی System Function Block.

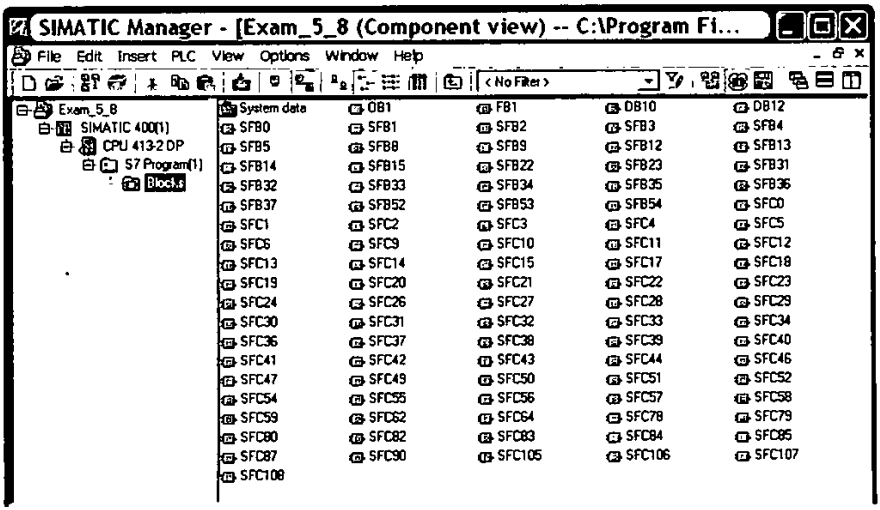
گروهی دیگر از بلاک‌های سیستمی نیز وجود دارند که به نام SDB شناخته می‌شوند. این بلاک‌ها به‌منظور ذخیره‌سازی تنظیمات سخت‌افزاری و شبکه مورد استفاده قرار گرفته و به‌طور اتوماتیک ایجاد می‌شوند و حاوی دیتا هستند ولی SFC و SFBها حاوی کد برنامه نویسی می‌باشند.

در این فصل فقط به SFC و SFBها در حد آشنایی پرداخته می‌شود و فقط چند نمونه از آنها تشریح می‌گردد. بحث کامل SFC و SFBها در کتاب سطح تکمیلی آورده شده است.

۵-۲ معرفی کلی SFC و SFB

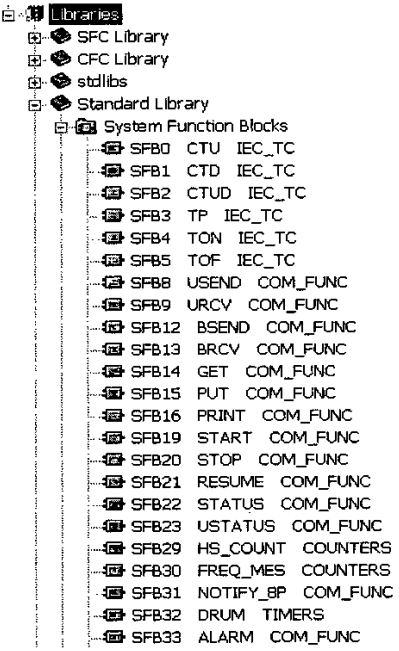
۵-۲-۱ نکات کلی

- SFC و SFB در سیستم عامل CPU موجود می‌باشند. این بلاک‌ها را می‌توان در هر CPU در پنجره Online شبیه شکل ۵-۱ مشاهده نمود.



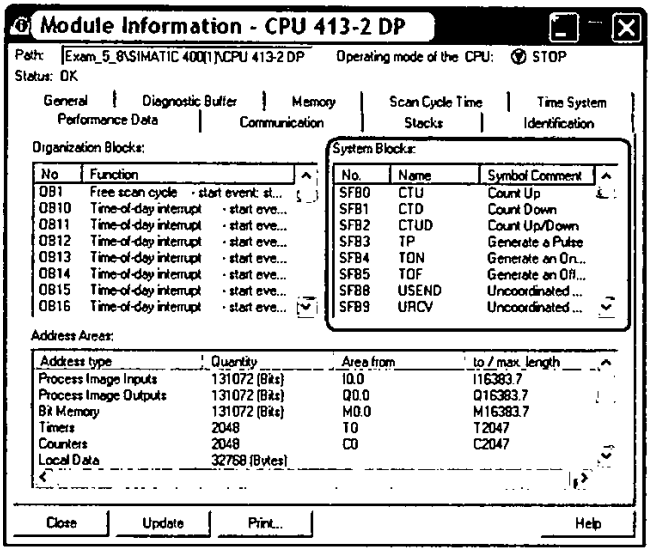
شکل ۵-۱ لیست SFC و SFBهای CPU در حالت Online

- SFC و SFBها را نمی‌توان از داخل CPU پاک کرد زیرا روی ROM ذخیره شده‌اند.
- محل دسترسی به این بلاک‌ها از قسمت Library موجود در نرم‌افزار Step7 می‌باشد. در برنامه‌ی کاربر می‌توان به‌مانند فراخوانی FC و FB، SFC و SFB را فراخوانی نمود.
- هر CPU شماره SFC و SFBهای خاصی را پشتیبانی می‌کند. بنابراین نمی‌توان هر SFC یا هر SFB را برای هر CPU دلخواه فراخوان کرد. به‌عنوان مثال SFB14 خاص S7-400 است و برای S7-300 قابل استفاده نیست.
- در صورت فراخوانی یک SFC یا SFB غیرمجاز در برنامه در هنگام دانلود با پیغام خطای Unable to copy the Block مواجه می‌شویم.
- برای مشاهده لیست SFC و SFBهای مجاز در CPU می‌توان در حالت Online از پنجره Module Info از سربرگ Performance Data مانند شکل ۵-۳ استفاده نمود.



شکل ۵-۲ محل دسترسی به SFC و SFBها در کتابخانه نرم‌افزار

فصل ۵



شکل ۵-۳ لیست SFC و SFBهای CPU در سربرگ Performance Data

- وقتی ورژن Firmware مربوط به CPU ارتقا پیدا می‌کند، ممکن است SFC و SFBهای جدیدی را نیز ساپورت کند. نحوه ارتقاء ورژن در پیوست ۱ آمده است.
- SFC و SFBها Protect هستند و نمی‌توان آنها را باز کرد. شماره آنها نیز قابل تغییر نیست.
- ساختار SFC شبیه ساختار FC است و به همان نحو صدا زده می‌شود.
- ساختار SFB شبیه ساختار FB است و همراه با DB صدا زده می‌شود.
- CPUهای یکپارچه دارای SFBهای بیشتری نسبت به نوع معمولی هستند که برای کاربردهای خاص آن CPU نظیر شمارش سریع طراحی شده‌اند.
- در صورتی‌که SFC یا SFB به CPU دانلود نشود باعث توقف CPU نمی‌شود، زیرا این بلاک‌ها در سیستم عامل CPU موجود می‌باشند.

۵-۲-۲ وظایف SFC و SFB

SFCها و SFBها برای کاربردهای خاص و بعضاً پیچیده طراحی شده‌اند.

از مهمترین کاربردهای SFBها می‌توان به موارد زیر اشاره کرد:

- SFBهای مخصوص ارتباطات بین چند PLC
- SFBهای مخصوص برنامه‌نویسی CPUهای Compact و IFM
- SFBهای مورد استفاده به‌عنوان تایمرها و کانترهای IEC
- SFBهای مورد استفاده در کنترل حلقه‌بسته

از مهمترین کاربردهای SFCها می‌توان به موارد زیر اشاره کرد:

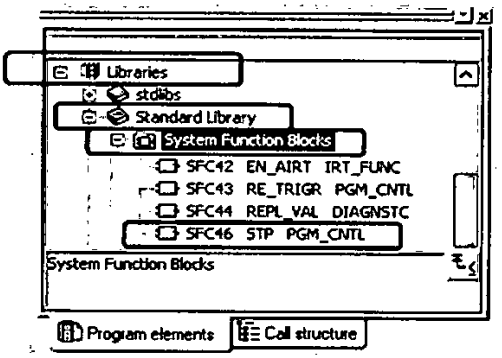
- ارتباطات شبکه
- مدیریت وقفه‌ها (فعال‌سازی یا غیر فعال‌سازی وقفه)
- برگرداندن اطلاعات سیستم
- ایجاد دیتابلاک
- توقف CPU
- جابه‌جایی حجم زیاد دیتا
- خواندن و تنظیم تاریخ و زمان CPU
- و ...

در این کتاب فقط برخی از SFCها و SFBها همراه با مثال تشریح می‌گردد.

۵-۳ آشنایی با چند SFC

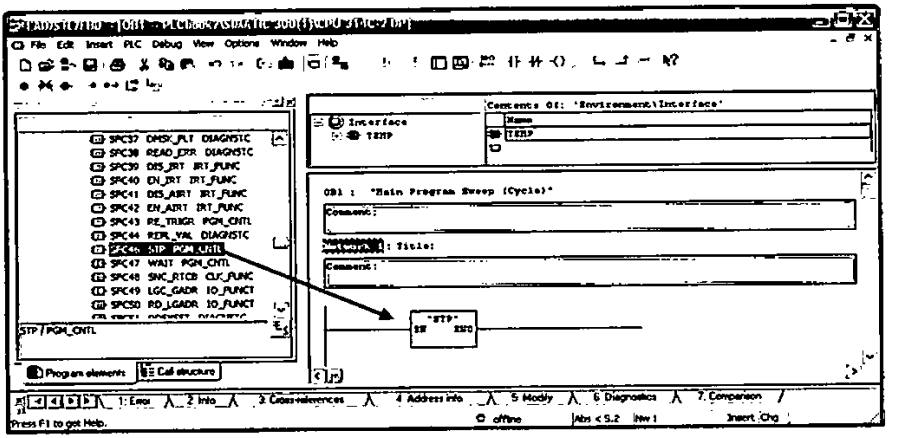
۵-۳-۱ توقف CPU با SFC46

SFC 46 با نام سمبولیک " STP" بلاکی است که به‌منظور توقف PLC به‌کار می‌رود. محل دسترسی به آن در Library نرم‌افزار LAD/STL/FBD مطابق شکل ۵-۴ است.



شکل۵-۴ مسیر فراخوانی SFC 46 از Library نرم‌افزار LAD/STL/FBD

SFC46 را شبیه فراخوانی FC به برنامه منتقل می‌کنیم.



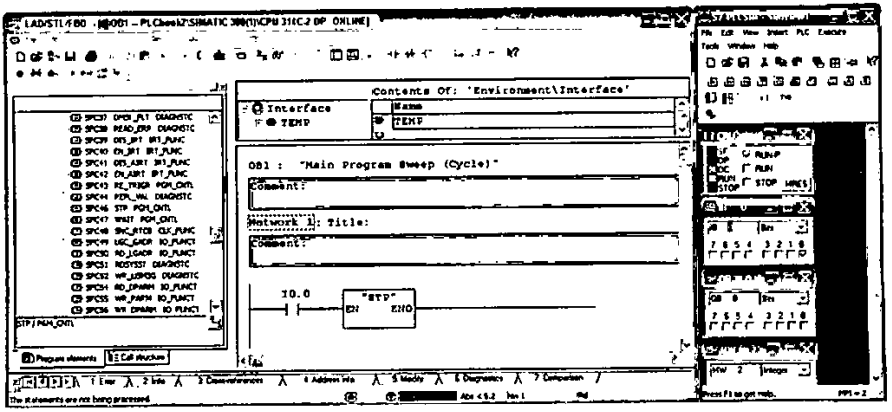
شکل ۵-۵ قراردادن SFC 46 در نتورک ۱ از برنامه OB1

همانطور که در شکل ۵-۵ مشخص است، این بلاک دارای هیچ پارامتری نمی‌باشد فقط پارامترهای EN و ENO که در همه بلاک‌های قابل فراخوانی (FC، FB، SFC و SFB) موجود می‌باشند، در این بلاک نیز موجود می‌باشند. در واقع می‌توان با کنترل پایه EN فعال یا غیرفعال شدن این بلاک را کنترل نمود.

مثال ۵-۱

برنامه‌ای بنویسید که با فشرده‌شدن شستی I0.0 (تیغه باز)، CPU متوقف شود.

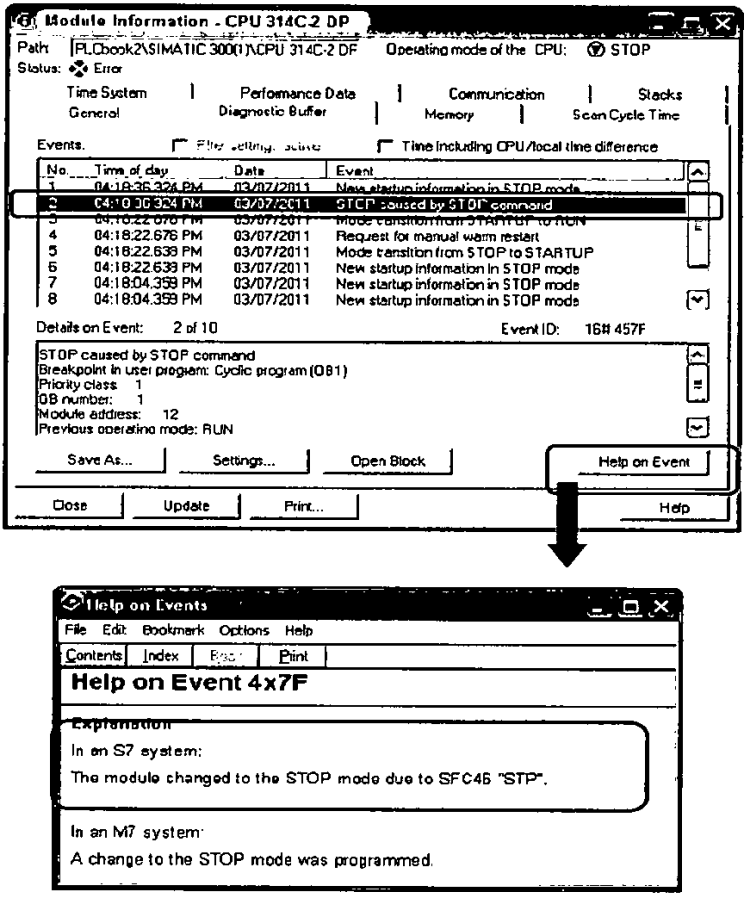
حل: شکل ۵-۶ حالتی را نشان می‌دهد که شرط فعال‌شدن SFC 46 فعال شدن ورودی I0.0 در نظر گرفته شده است. همانطور که مشخص است، اگرچه در Simulation مد کنترلی در حالت RUN-P قرار داد ولی به‌دلیل فعال شدن I0.0، عملاً CPU متوقف شده است.



شکل ۵-۶ حل و تست مثال ۵-۱

پیام بافر

در هنگامی‌که CPU به‌دلیل فراخوانی SFC 46 متوقف گردد، در بافر تشخیص عیب CPU (Diagnostic Buffer) پیامی مانند شکل ۵-۷ نشان داده می‌شود. با کلیک بر روی گزینه Help on Event می‌توان علت توقف را مشاهده نمود.

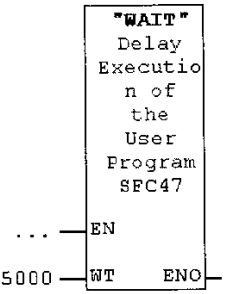


شکل ۵-۷ پیام بافر در هنگام توقف CPU ناشی از فراخوانی SFC 46

در شرایطی که توقف CPU با کلید روی آن یا ناشی از فالت باشد؛ پیغام بافر با آنچه در شکل ۵-۷ نشان داده شده متفاوت خواهد بود.

۵-۳-۲ تأخیر در پردازش CPU با SFC47

فراخوانی این فانکشن منجر به تأخیر زمانی در اجرای برنامه به اندازه زمان مشخص شده در ورودی Wait می‌گردد. عدد صحیح به این ورودی داده می‌شود که مفهوم آن میکروثانیه است؛ پس ماکزیمم تاخیر زمانی ۳۲۷۶۷ میکروثانیه خواهد بود.

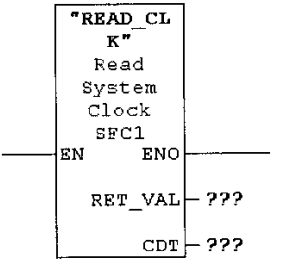


شکل ۵-۸ SFC47

به‌عنوان مثال اگر برنامه فوق در OB1 به‌کار رود موجب می‌شود که سیکل اسکن نسبت به حالت عادی ۵ میلی ثانیه بیشتر طول بکشد.

۵-۳-۳ خواندن تاریخ و زمان CPU با SFC1

SFC1 با نام سمبلیک Read_CLK هرجا فراخوان شود تاریخ و زمان CPU را بر می‌گرداند.



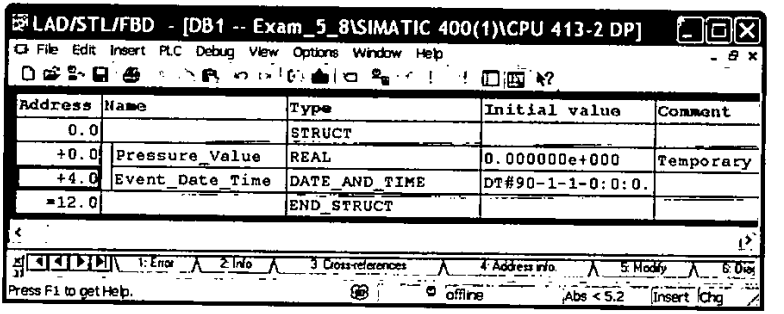
شکل ۵-۹ SFC1

همانطور که دیده می‌شود این SFC دو خروجی دارد. خروجی اولRET_VAL که از جنس Word است و در صورت مشکل در اجرای SFC کد خطا را بر می‌گرداند. خروجی دوم CDT است که تاریخ و زمان CPU را بر می‌گرداند و از جنس Date And Time است بنابراین نمی‌توان آنرا در متغیر حافظه که ماکزیمم ۳۲ بیتی است ذخیره کرد. برای این کار می‌توان از دیتا بلاک به صورتی‌که در فصل‌های قبل تشریح شد استفاده نمود.

مثال ۵-۲

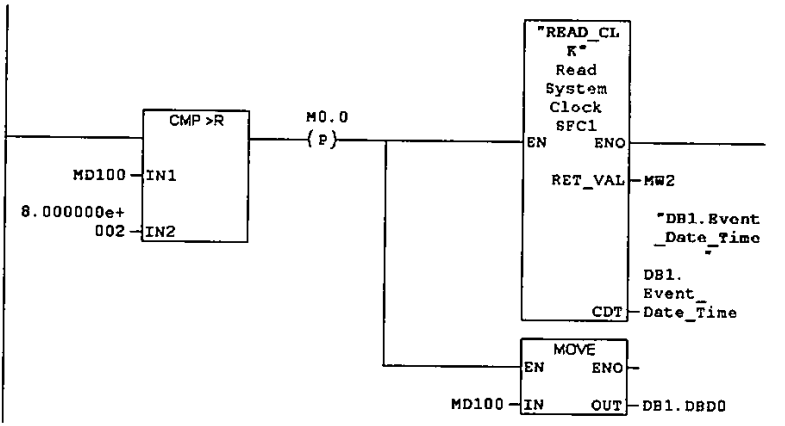
برنامه‌ای بنویسید که اگر مقدار دما که در MD100 ذخیره شده از 800 بیشتر شد این مقدار و تاریخ و زمان وقوع آن در دیتا بلاک ذخیره شود.

حل: ابتدا DB1 را با دو سطر به‌صورت زیر ایجاد می‌کنیم.



شکل ۵-۱۰ دیتا بلاک مثال ۵-۲

برنامه به‌صورت زیر خواهد بود.



شکل ۵-۱۱ برنامه مثال ۵-۲

تذکر: در کتاب سطح تکمیلی به‌طور کامل با فانکشن‌های مربوط به تاریخ و زمان آشنا خواهید شد.

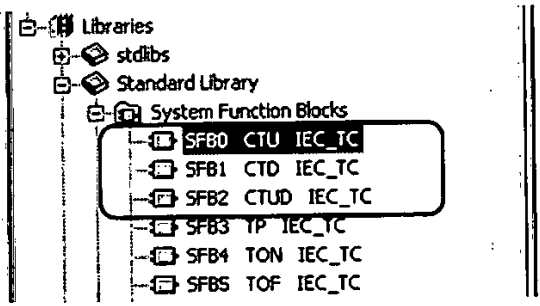
۵-۴ آشنایی با چند SFB

۵-۴ -۱ آشنایی با کانترهای IEC

در S7 گروهی از SFBها وجود دارند که وظیفه‌ی آنها انجام عمل شمارش می‌باشد. اصول عملکرد این بلاک‌ها بر اساس استاندارد IEC 1131-3 بوده و به همین دلیل به آنها اصطلاحاً کانترهای IEC می‌گویند. این کانترها در عملکرد و برخی از خصوصیات تفاوت‌هایی با کانترهای معمولی دارند. به‌طور کلی سه نوع کانتر IEC وجود دارد:

- "SFB 0 "CTU
- "SFB 1 "CTD
- "SFB 2 "CTUD

این بلاک‌ها در Library نرم‌افزار موجود بوده و از مسیر نشان داده شده در شکل ۵-۱۲ فراخوانی می‌شوند.



شکل۵-۱۲ مسیر دسترسی به کانترهایIEC

تفاوت‌ها

جدول ۵-۱ تفاوت‌های کانترهای IEC و کانترهای معمولی را نشان می‌دهد.

جدول ۵-۱

	کانترهای IEC	کانترهای معمولی
فرمت	SFB	C
نوع داده خروجی	Integer	BCD
ماکزیمم مقدار قابل شمارش	32767	999
ماندگاری	به ماندگاری دیتا بلاک اختصاص داده شده بستگی دارد.	قابل تعریف به‌صورت ماندگار در CPU

کانتر افزایشی "SFB 0 "CTU

این SFB یک شمارنده افزایشی می‌باشد و همانطور که در شکل ۵-۱۳ مشخص است دارای پارامترهای زیر می‌باشد:

EN: فعال‌ساز بلاک و از نوع BOOL بوده و هرگاه یک شود بلاک اجرا می‌گردد.

ENO: از نوع BOOL و نشان‌دهنده صحت فراخوانی بلاک می‌باشد.

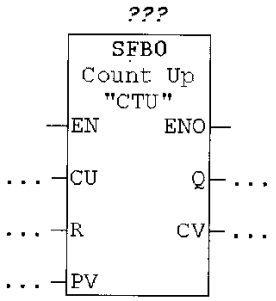
CU: فعال‌ساز افزایشی بوده و هرگاه یک لبه‌ی بالارونده به آن اعمال شود، خروجی کانتر یک عدد افزایش می‌یابد. این پارامتر از نوع BOOL می‌باشد.

R: ریست کانتر بوده و هرگاه یک شود کانتر ریست می‌گردد. این پارامتر از نوع BOOL می‌باشد.

PV: مقدار دلخواهی را جهت مقایسه با خروجی CV و تعیین وضعیت خروجی Q می‌توان در این قسمت وارد نمود. این پارامتر از نوع Integer می‌باشد.

CV: مقدار شمارش شده توسط کانتر به فرم Integer در آن قرار می‌گیرد.

Q: این پارامتر از نوع BOOL بوده و اگر مقدار CV بزرگتر یا مساوی مقدار PV باشد، Q دارای مقدار یک منطقی بوده و در غیر اینصورت صفر می‌گردد.



شکل۵-۱۳ نمایش SFB0 در زبان LAD

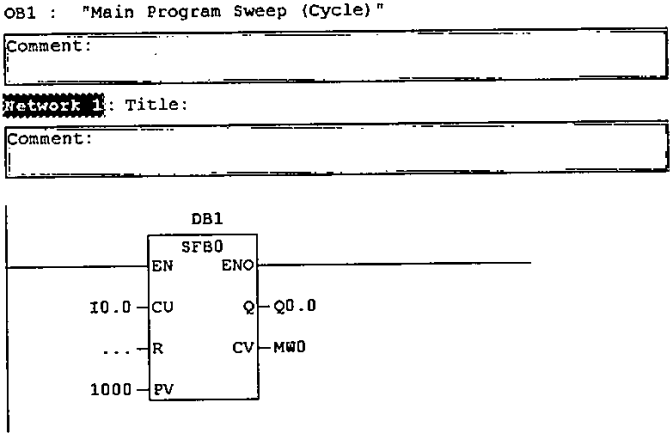
نکته: کلیه کانترهای IEC نیاز به دیتابلاک اختصاصی داشته که شماره آن در قسمت بالای SFB درج می‌گردد.

فصل ۵

مثال ۵-۳

در یک پروسه صنعتی محصول نهایی تولید شده از مقابل سنسور I0.0 (تیغه باز) عبور کرده و در انبار قرار می‌گیرد. ظرفیت انبار ۱۰۰۰ جعبه بوده و لازم است هرگاه انبار پر شد لامپ آلارمی (Q0.0) روشن شود. برنامه‌ی لازم را طراحی نمایید.

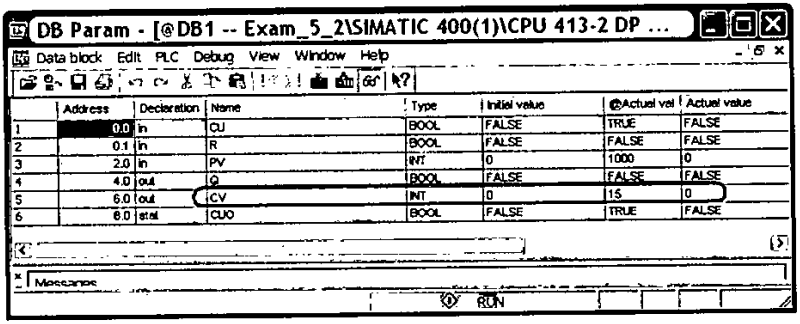
حل: از آنجایی که کانترهای معمولی تا عدد ۹۹۹ را می‌توانند بشمارند، بنابراین یا باید از یک برنامه ترکیبی استفاده نمود یا از کانترهای IEC. از آنجا که کار کردن با کانترهای IEC ساده‌تر است، راه حل استفاده از این کانترها ارائه می‌گردد.



شکل ۵-۱۴ برنامه مثال ۵-۳

همانطور که مشاهده می‌شود، با استفاده از این نوع کانتر دیگر نیازی به استفاده از مقایسه‌گر نمی‌باشد. زیرا این نوع کانتر دارای مقایسه‌گری است که مقدار وارد شده در پارامتر PV را با مقدار شمارش‌شده توسط کانتر مقایسه می‌کند. در مثال فوق به‌منظور انجام عمل ریست می‌توان آدرس یک شستی را به پارامتر R اختصاص داد. تست و تحلیل برنامه‌ی فوق بر عهده خواننده می‌باشد.

اگر DB1 اختصاص داده شده به SFB0 را باز و مانیتور کنیم، شمارش کانتر را در آن مانند شکل ۵-۱۵ خواهیم دید.



شکل ۵-۱۵ دیتا بلاک مثال ۵-۳

کانتر کاهشی "SFB 1 "CTD

این SFB یک کانتر کاهشی می‌باشد و همانطور که در شکل ۵-۱۶ مشخص است، دارای پارامترهای زیر می‌باشد:

EN: فعال‌ساز بلاک و از نوع BOOL بوده و هرگاه یک شود بلاک اجرا می‌گردد.

ENO: از نوع BOOL و نشان‌دهنده صحت فراخوانی بلاک می‌باشد.

CD: فعال‌ساز کاهشی بوده و هرگاه یک لبه‌ی بالارونده به آن اعمال شود، خروجی کانتر یک عدد کاهش می‌یابد. این پارامتر از نوع BOOL می‌باشد.

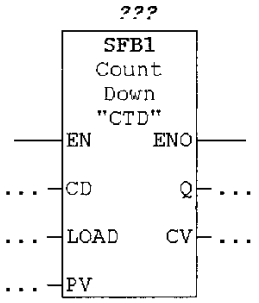
R: ریست کانتر بوده و هرگاه یک شود کانتر ریست می‌گردد. این پارامتر از نوع BOOL می‌باشد.

LOAD: از نوع BOOL بوده و هرگاه یک شود عدد وارد شده در پارامتر PV را به خروجی منتقل می‌کند.

PV: می‌توان مقدار دلخواهی را به فرم Integer در این پارامتر وارد نموده و سپس با یک نمودن پارامتر LOAD، آنرا به خروجی کانتر انتقال داد تا کانتر از این عدد شروع به شمارش نزولی نماید.

CV: مقدار شمارش شده توسط کانتر به فرم Integer در آن قرار می‌گیرد.

Q: این پارامتر از نوع BOOL بوده و اگر مقدار CV کوچکتر یا مساوی صفر باشد، Q دارای مقدار یک منطقی بوده و در غیر اینصورت صفر می‌گردد.

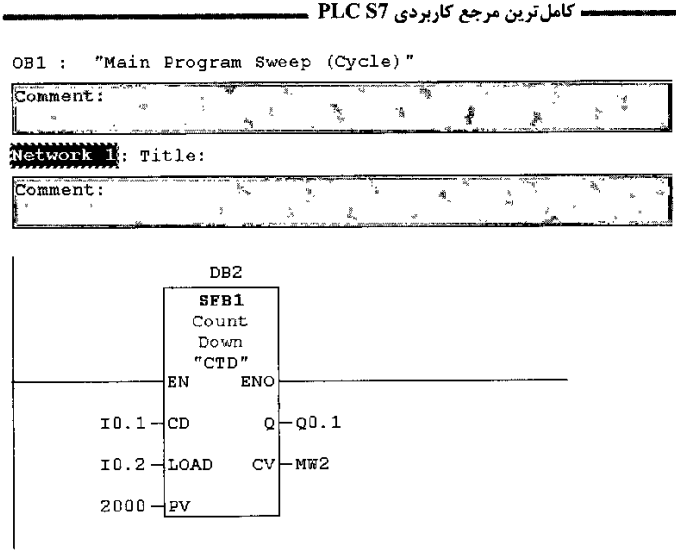


شکل ۵-۱۶ نمایش SFB1 در زبان LAD

مثال ۵-۴

برنامه‌ای طراحی نمایید که با قطع و وصل شدن I0.1، یک کانتر از عدد ۲۰۰۰ شروع به شمارش نزولی نموده و هنگامی که به عدد صفر رسید، خروجی Q0.1 روشن شود. با وصل شدن شستی I0.2 مجدداً مقدار ۲۰۰۰ در خروجی کانتر قرار بگیرد.

فصل ۵

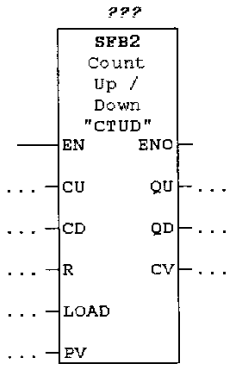


شکل ۵-۱۷ برنامه مثال ۵-۴

تحلیل مثال فوق بر عهده خواننده می‌باشد.

کانتر افزایشی کاهشی "SFB 2 "CTUD

این SFB یک کانتر افزایشی/ کاهشی بوده و همه پارامترهای هر دو نوع SFB0 و SFB1 را دارا می‌باشد. از این کانتر هم به‌عنوان کانتر افزایشی و هم به‌عنوان کانتر کاهشی می‌توان استفاده نمود. در شکل ۵-۱۸ این SFB و پارامترهای آن نشان داده شده است.

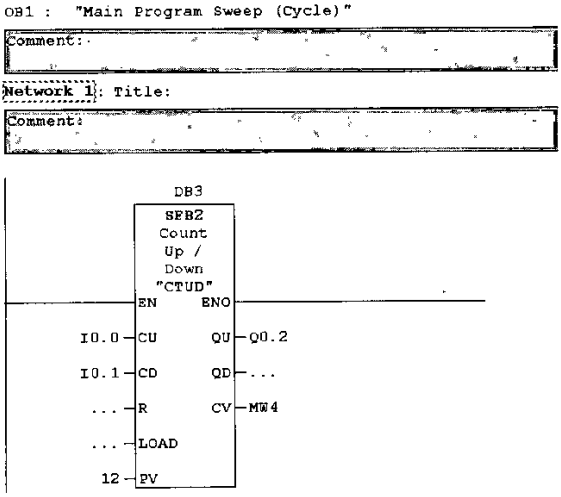


شکل ۵-۱۸ نمایش SFB2 در زبان LAD

همانطور که در شکل ۵-۱۸ مشخص است، پارامترهای SFB2 به‌صورت جداگانه در SFB0 و SFB1 موجود بوده‌اند. نقش این پارامترها در SFB2 همان نقشی است که این پارامترها در SFB0 و SFB1 داشته‌اند لذا از ذکر مجدد آنها خودداری می‌گردد. فقط به این نکته اشاره می‌شود که پارامتر QU خروجی مربوط به حالت افزایشی و QD خروجی مربوط به حالت کاهشی است.

مثال ۵- ۵

در یک پروسه صنعتی تعدادی جعبه وارد انبار شده و تعدادی جعبه از انبار خارج می‌گردد. جعبه‌های ورودی به انبار توسط سنسور I0.0 (تیغه باز) و جعبه‌های خروجی از انبار توسط سنسور I0.1 (تیغه باز) تشخیص داده می‌شوند. لازم است برنامه‌ای طراحی شود که اولاً تعداد جعبه‌های موجود در انبار در حافظه MW4 ذخیره شود و ثانیاً اگر تعداد جعبه‌های موجود در انبار بیشتر از عدد ۱۲ شد لامپ هشدار Q0.2 روشن شود.



شکل ۵-۱۹ برنامه مثال ۵-۵

با توجه به نکات گفته شده، تست و تحلیل برنامه فوق بر عهده خواننده است.

فصل ۵

۵-۴-۲ آشنایی با تایمرهای IEC

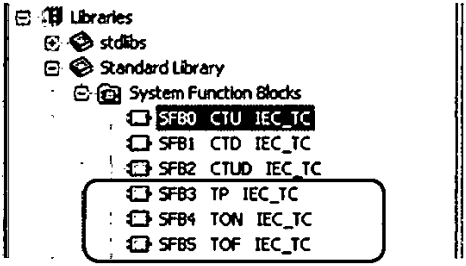
در S7 گروهی از SFBها وجود دارند که نقشی مانند تایمر را ایفا می‌نمایند. اصول عملکرد این بلاک‌ها بر اساس استاندارد IEC 1131-3 بوده و به همین دلیل به آنها اصطلاحاً تایمرهای IEC می‌گویند. در واقع این بلاک‌ها نقشی مشابه تایمرهای معمولی موجود در System Memory یک CPU را ایفا می‌کنند ولی تفاوت اساسی نسبت به آنها دارند.

به‌طور کلی سه نوع تایمر IEC وجود دارد:

- SFB 3 "TP"

- SFB 4 "TON"
- SFB 5 "TOF"

این بلاک‌ها در Library نرم‌افزار موجود بوده و از مسیر نشان داده شده در شکل ۵-۲۰ فراخوانی می‌شوند.



شکل ۵-۲۰ مسیر دسترسی به تایمرهای IEC

تفاوت‌ها

جدول ۵-۲ تفاوت‌های تایمرهای IEC و تایمرهای معمولی را نشان می‌دهد.

جدول ۵-۲

	تایمرهای IEC	تایمرهای معمولی
فرمت	SFB	T
فرمت زمان	TIME	S5TIME
ماکزیمم زمان قابل محاسبه	24D_20H_31M_23S_648MS	2H46M30S
حداقل زمان قابل محاسبه	1 ms	10 ms
ماندگاری	به ماندگاری دیتا بلاک اختصاص داده شده بستگی دارد.	قابل تعریف به‌صورت ماندگار در CPU

تایمر پالس "SFB 3 "TP

عملکرد این تایمر تا حدی شبیه تایمر Extended Pulse (یعنی S_PEXT) می‌باشد، یعنی با لبه ورودی خروجی یک می‌شود و اگر ورودی قطع شود به‌کار خود ادامه می‌دهد. ولی تفاوت آن با S_PEXT در این است که اگر در طول کار تایمر، ورودی قطع و وصل شود اهمیتی نداده و به‌کار خود ادامه می‌دهد در حالی که در تایمر S_PEXT اگر در بین کار ورودی قطع و وصل شود، تایمر از نو زمان‌سنجی را شروع می‌کند. همانطور که در شکل ۵-۲۱ مشخص است، تایمر SFB3 دارای پارامترهای زیر می‌باشد:

EN: فعال‌ساز بلاک و از نوع BOOL بوده و هرگاه یک شود بلاک اجرا می‌گردد.

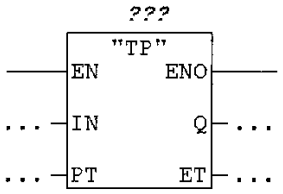
ENO: از نوع BOOL و نشان‌دهنده صحت فراخوانی بلاک می‌باشد.

IN: با یک شدن این پایه، تایمر شروع به‌کار می‌کند.

PT: زمان مورد نظر جهت کار تایمر به فرمت Time

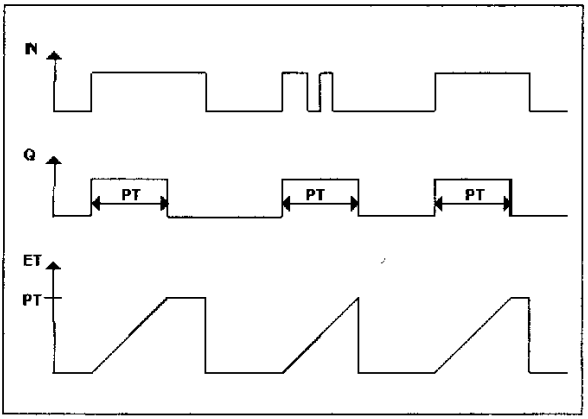
ET: زمان باقیمانده از کار تایمر را به فرم Time نشان می‌دهد.

Q: این پارامتر از نوع BOOL بوده و حکم خروجی تایمر را دارد. با فعال شدن آدرس اختصاص یافته به پایه IN، خروجی Q روشن شده و پس از سپری شدن زمان وارد شده در پایه PT خروجی Q خاموش می‌شود.



شکل ۵-۲۱ نمایش SFB3 در زبان LAD

شکل ۵-۲۲ عملکرد این تایمر را نشان می‌دهد.



شکل ۵-۲۲ نمایش عملکرد SFB3

مثال ۵-۶

در یک پروسه صنعتی مانند شکل ۵-۲۳ گاز مرطوب وارد مخزن شده و پس از جداسازی آب،گاز خشک از مخزن خارج می‌شود.

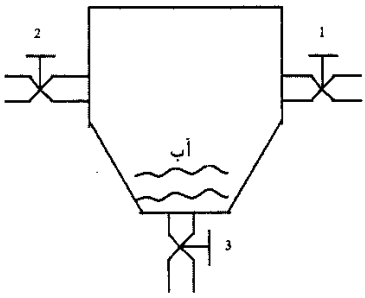
برای پیاده‌سازی منطق کنترل لازم است:

ولو ۱: دو روز باز بوده و پس از دو روز به مدت ۳۰ دقیقه بسته شود. (Q124.0)

ولو ۲: همواره باز باشد. (Q124.2)

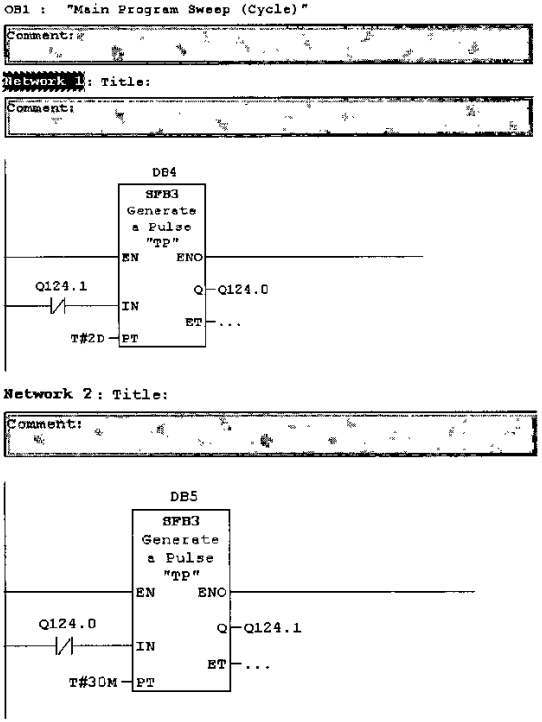
ولو ۳: ۳۰ دقیقه باز بوده و سپس به مدت دو روز بسته شود. (Q124.1)

فصل ۵



شکل ۵-۲۳ پروسه مورد نظر جهت مثال ۵-۵

برنامه مورد نظر

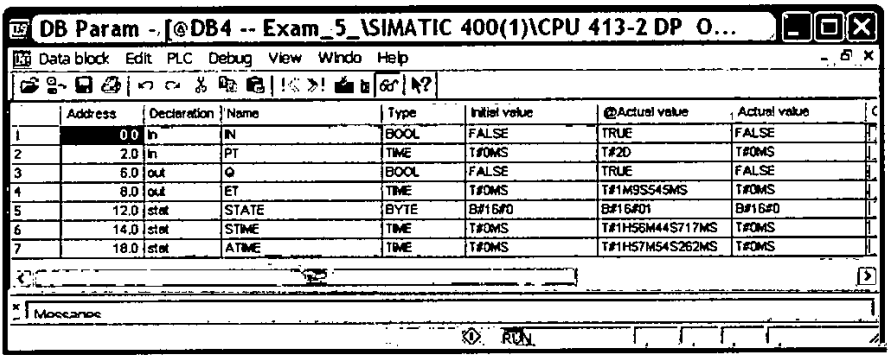


شکل ۵-۲۴ برنامه مثال ۵-۶

تذکر: همانطور که در مثال فوق دیده می‌شود، هر تایمر IEC نیاز به دیتا بلاک خاص خود دارد.

مثال ۵- ۷ بررسی ماندگاری تایمر IEC

اگر در مثال ۵-۶ PLC از نوع S7-300 باشد، دیتابلاک‌ها روی کارت فلش ذخیره خواهند شد بنابراین تایمرها حالت ماندگار خواهند داشت و در صورت قطع و وصل تغذیه پاک نخواهند شد. خواننده می‌تواند این موضوع را با PLC چک کند. وقتی تایمر اول در حال کار است تغذیه را قطع و مجدداً وصل نمایید. خواهید دید که تایمر از جایی که قطع شده ادامه می‌دهد. این موضوع به‌صورت واضح‌تر در دیتابلاک DB4 در حالت Online قابل مشاهده است.



	Address	Declaration	Name	Type	Initial value	@Actual value	Actual value
1	0.0	in	IN	BOOL	FALSE	TRUE	FALSE
2	2.0	in	PT	TIME	T#0MS	T#2D	T#0MS
3	6.0	out	Q	BOOL	FALSE	TRUE	FALSE
4	8.0	out	ET	TIME	T#0MS	T#1M9S545MS	T#0MS
5	12.0	stat	STATE	BYTE	B#16#0	B#16#01	B#16#0
6	14.0	stat	STIME	TIME	T#0MS	T#1H56M44S717MS	T#0MS
7	18.0	stat	ATIME	TIME	T#0MS	T#1H57M54S262MS	T#0MS

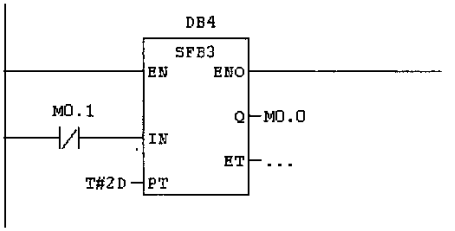
شکل ۵-۲۵ دیتا بلاک مثال ۵-۷

ولی مشکلی که وجود دارد این است که در صورت قطع و وصل تغذیه، سیستم همیشه از ابتدا شروع می‌کند یعنی ولو گاز را باز می‌کند. در شرایطی که در وسط باز بودن ولو آب تغذیه قطع و وصل شود برنامه درست کار نخواهد کرد. علت این است که آدرس‌های Q حالت ماندگاری ندارد و با قطع و وصل تغذیه هر دو صفر می‌شوند. بنابراین در شروع کار همواره تایمر موجود در Network1 فعال و تایمر دوم غیر فعال است. برای رفع این مشکل می‌توان برنامه را به‌صورت زیر تغییر داد و از آدرس‌های M که حالت ماندگاری دارند استفاده نمود.

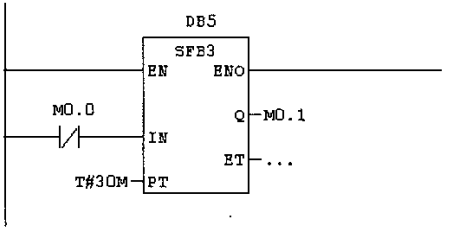
فصل ۵

OB1 : "Main Program Sweep (Cycle)"

Network 1 : Title:

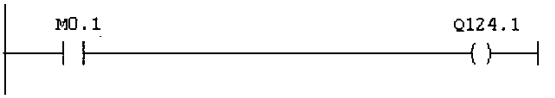


Network 2 : Title:



Network 3 : Title:

Network 4 : Title:



شکل ۵-۲۶ دیتابلاک مثال ۵-۷

تایمر تأخیر در وصل "TON" SFB 4

عملکرد این SFB شبیه یک تایمر از نوع On Delay می‌باشد و همانطور که در شکل ۵-۲۷ مشخص است، دارای پارامترهای زیر می‌باشد:

EN: فعال‌ساز بلاک و از نوع BOOL بوده و هرگاه یک شود بلاک اجرا می‌گردد.

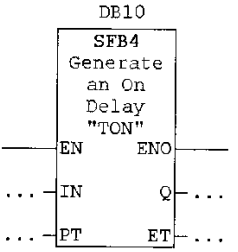
ENO: از نوع BOOL و نشان‌دهنده صحت فراخوانی بلاک می‌باشد.

IN: با یک شدن این پایه، تایمر شروع به‌کار می‌کند.

PT: زمان مورد نظر جهت کار تایمر به فرمت Time

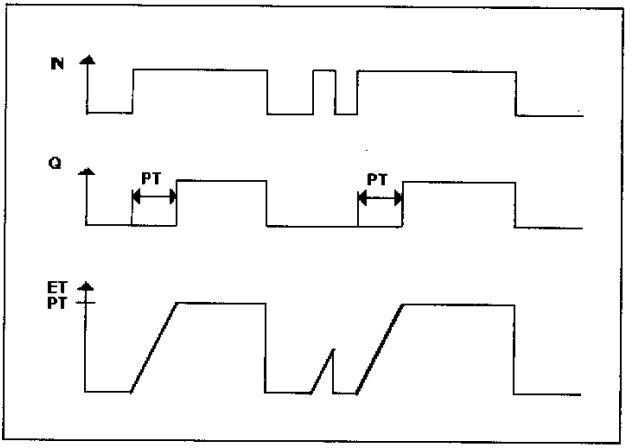
ET: زمان باقیمانده از کار تایمر را به فرم Time نشان می‌دهد.

Q: این پارامتر از نوع BOOL بوده و حکم خروجی تایمر را دارد. با یک شدن آدرس اختصاص یافته به پایه IN، تایمر شروع به کار نموده و پس از سپری شدن زمان وارد شده در پایه PT خروجی Q یک می‌شود. این خروجی تا زمانی که آدرس اختصاص یافته به پارامتر IN دارای مقدار یک منطقی است روشن مانده و در صورتی‌که آدرس اختصاص یافته به پارامتر IN صفر منطقی شود، خروجی Q نیز صفر می‌گردد.



شکل ۵-۲۷ نمایش SFB4 در زبان LAD

شکل ۵-۲۸ عملکرد این تایمر را نشان می‌دهد.



شکل ۵-۲۸ نمایش عملکرد SFB4

فصل ۵

تایمر تأخیر در قطع "SFB 5 "TOF

عملکرد این SFB شبیه یک تایمر از نوع Off Delay بوده و دارای پارامترهای زیر می‌باشد:

EN: فعال‌ساز بلاک و از نوع BOOL بوده و هرگاه یک شود بلاک اجرا می‌گردد.

ENO: از نوع BOOL و نشان‌دهنده صحت فراخوانی بلاک می‌باشد.

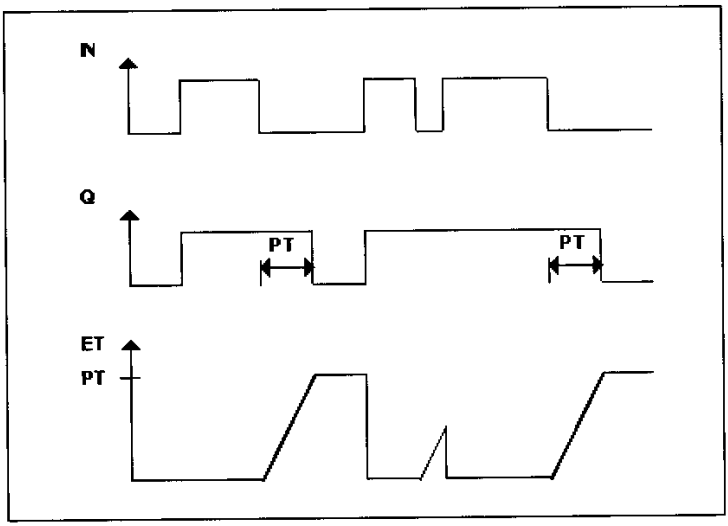
IN: با اعمال لبه بالارونده به این پایه، خروجی Q یک شده و با اعمال لبه پایین‌رونده، تایمر شروع به‌کار می‌کند.

PT: زمان مورد نظر جهت کار تایمر به فرمت Time

ET: زمان باقیمانده از کار تایمر را به فرم Time نشان می‌دهد.

Q: این پارامتر از نوع BOOL بوده و حکم خروجی تایمر را دارد. با اعمال لبه بالارونده به پارامتر IN خروجی Q یک می‌شود. با اعمال لبه پایین‌رونده به پارامتر IN تایمر شروع به کار نموده و پس از پایان زمان وارد شده در پارامتر PT، خروجی Q صفر می‌شود. عملکرد این SFB مانند تایمر S_OFFDT می‌باشد.

شکل ۵-۲۹عملکرد این SFB را نشان می‌دهد.



شکل ۵-۲۹ نمایش عملکرد SFB4

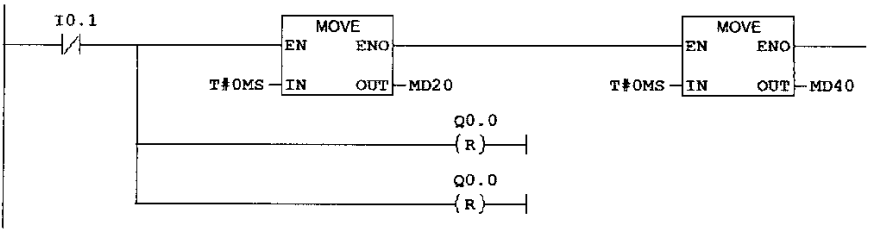
مثال ۵-۸

در یک سیستم صنعتی جهت پمپاژ آب از دو عدد موتور پمپ که یکی رزرو دیگری است استفاده می‌شود. با فشردن شدن شستی استارت I0.0 (تیغه باز) سیستم روشن شده و با فشرده شدن شستی استپ I0.1 (تیغه بسته) سیستم خاموش می‌شود. هنگامی که سیستم روشن می‌شود موتور پمپ۱ (Q0.0) روشن شده و پس از سپری‌شدن ۲۰ ساعت خاموش

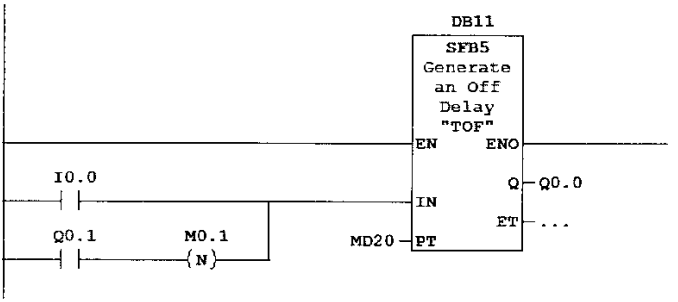
می‌شود. سپس موتور رزرو به مدت ۴ ساعت روشن شده و پس از آن خاموش می‌شود تا موتور پمپ۱ مجدداً روشن شود. این روند تا زمانی که سیستم روشن است ادامه پیدا می‌کند.

Network 1 : Title:

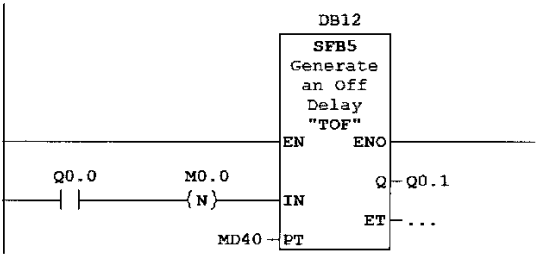
Network 2 : Title:



Network 3 : Title:



Network 4 : Title:



شکل ۵-۳۰ برنامه مثال ۵-۸

فصل ۵

مثال ۵-۹ کنترل ترتیب عملیات میکسر

فانکشن‌بلاکی برای کنترل یک میکسر طراحی کنید که توسط تایمرهای IEC عملیات کنترل میکسر را به‌صورت زیر انجام دهد:

با استارت اپراتور

۱- ولو ورودی ۱ به مدت ۲۰ ثانیه باز شود سپس بسته شود.
۲- پس از بسته شدن ولو ۱، ولو ورودی شماره ۲ به مدت ۱۰ ثانیه باز سپس بسته شود.
۳- پس از بسته شدن ولو شماره ۳، موتور همزن به مدت ۳۰ ثانیه کار کند سپس از کار بیفتد.
۴- پس از اینکه میکسر از کار افتاد، ولو خروجی به مدت ۲۰ ثانیه باز بماند سپس بسته شود.
۵- تا زمانی که اپراتور استپ را فعال نکرده، سیکل از مرحله ۱ تکرار شود.

با فرمان استپ اپراتور

سیکل فعلی کامل ولی سیکل جدید شروع نشود.

تذکر ۱: همه SFBها و FB اصلی از یک دیتا بلاک multi instance استفاده کنند.

تذکر ۲: در صورت قطع و وصل تغذیه سیستم از مرحله ای که قطع شده کار را ادامه دهد

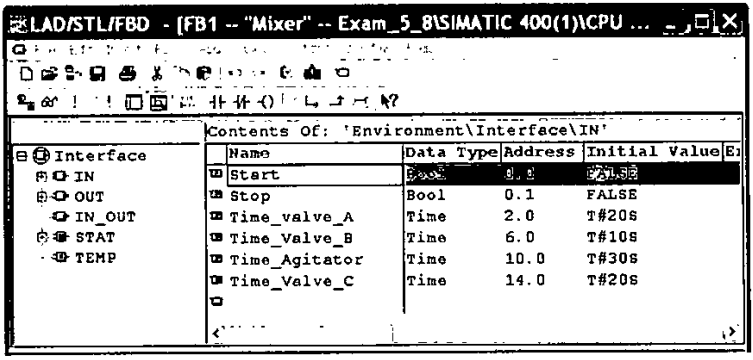
حل: لازم است فانکشن بلاکی طراحی شود که فرمان‌های استارت و استپ را به‌عنوان ورودی بگیرد و خروجی‌های لازم برای سه ولو و میکسر را برگرداند.

شستی استارت نرمال باز و شستی استپ نرمال بسته است.

مراحل کار

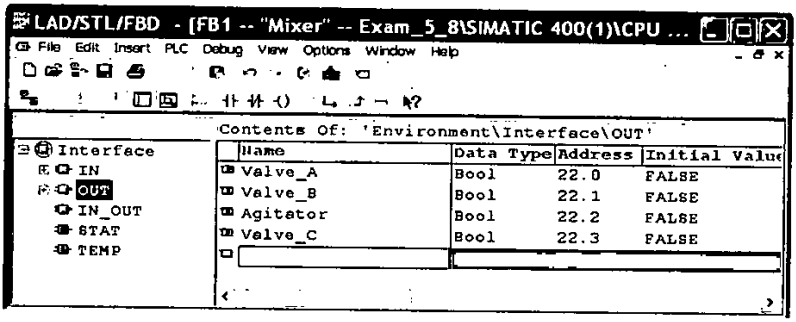
قدم اول: FB1 را با نام سمبلیک Mixer ایجاد می‌کنیم.

قدم دوم: ورودی‌های FB که فرمان استارت و استپ و زمان‌های تایمر است را به‌صورت زیر تعریف می‌کنیم. همانطور که دیده می‌شود، برای زمان‌ها مقادیر اولیه طبق خواسته برنامه تعریف شده است که اگر حتی به این ورودی چیزی اختصاص داده نشود برنامه با مقادیر اولیه کار خود را انجام دهد.



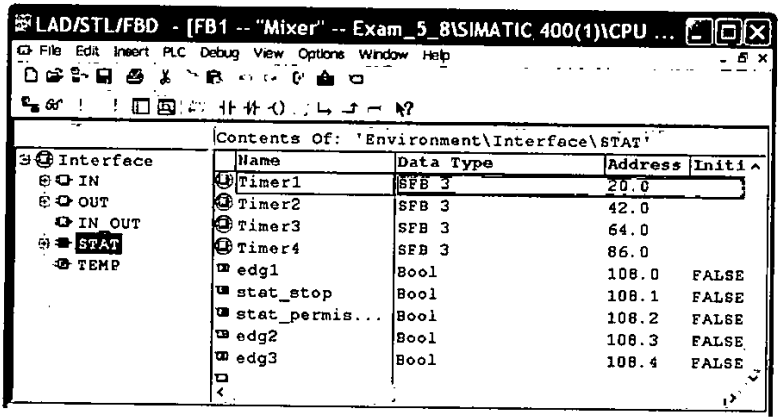
شکل ۵-۳۱ ورودی‌های FB مثال ۵-۹

قدم سوم: خروجی‌های FB را به‌صورت زیر تعریف می‌کنیم.



شکل ۵-۳۲ خروجی‌های FB مثال ۵-۹

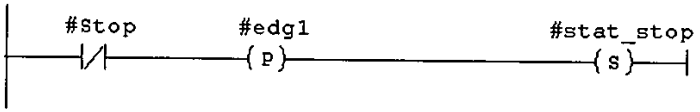
قدم چهارم: متغیرهای STAT را به‌صورت زیر تعریف می‌کنیم. دقت شود به‌دلیل نیاز به عملکرد Multi instance برای هر تایمر IEC یک متغیر STAT تعریف می‌گردد. برای تشخیص لبه و فیلیپ فلاپ و ... نیز متغیر STAT تعریف شده است.



شکل ۵-۳۳ متغیرهای استاتیک FB مثال ۵-۹

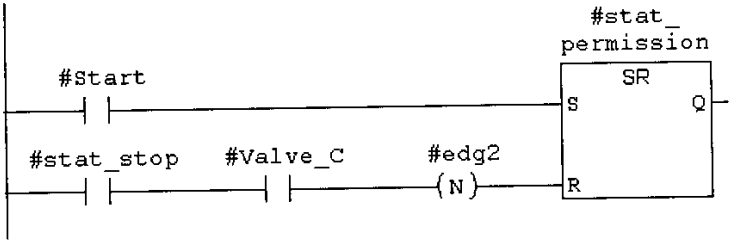
قدم پنجم: برنامه‌نویسی FB به‌صورت زیر انجام می‌شود. تحلیل به عهده خواننده است.

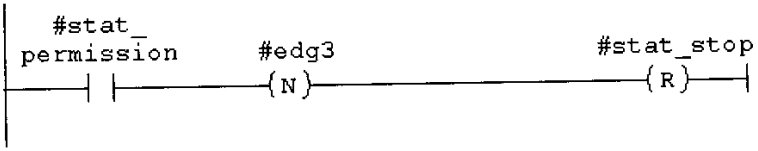
```
FB1 : Title:
Network 1: Title:
```



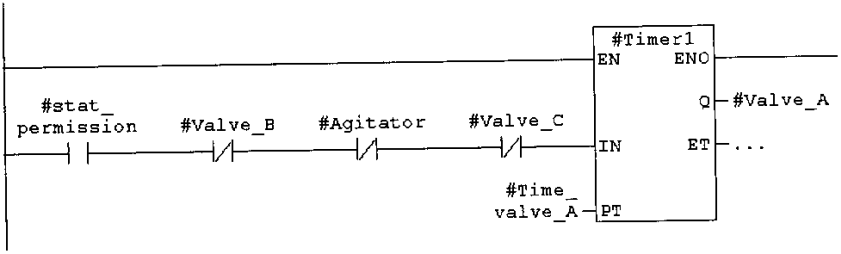
فصل ۵

Network 2 : Title:

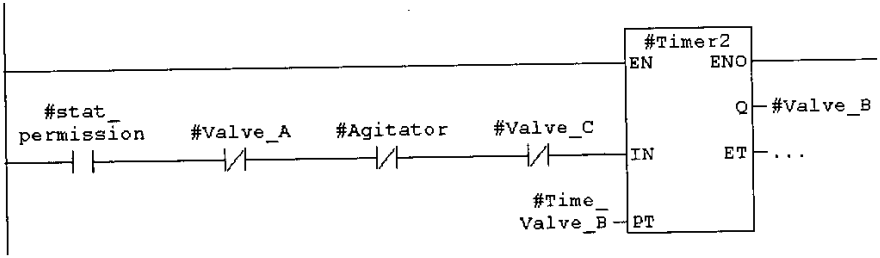




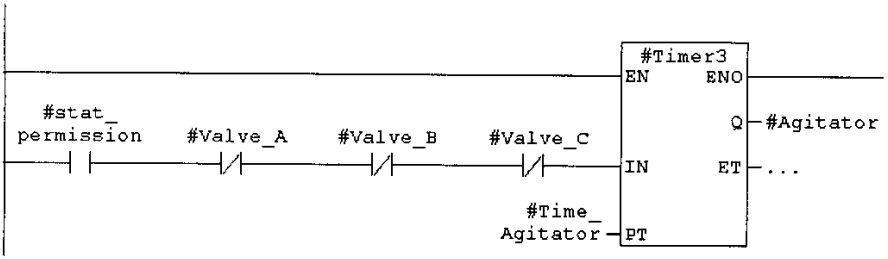
Network 4 : Title:



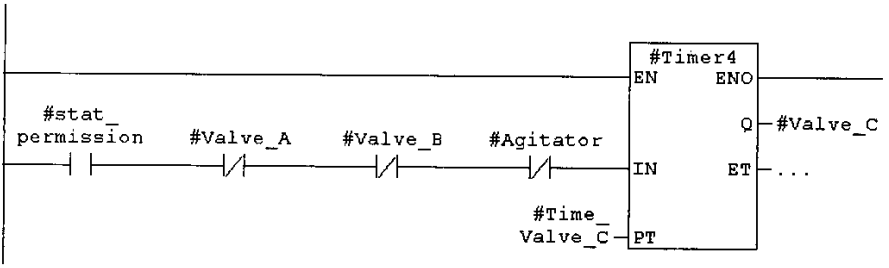
Network 5 : Title:



```
Network 6 : Title:
```



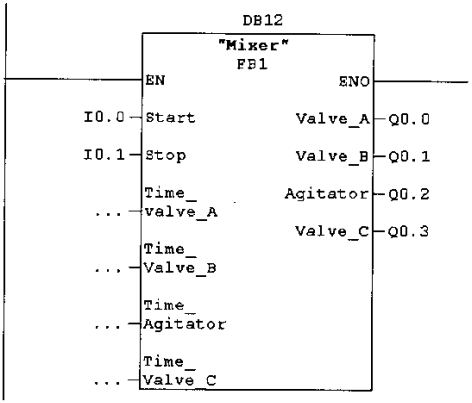

```
Network 7 : Title:
```



شکل ۵-۳۴ برنامه FB مثال ۵-۹

قدم ششم: فراخوانی FB1 از OB1 همراه با DB مانند شکل ۵-۳۵. همانطور که دیده می‌شود، اگر ورودی‌های زمان را خالی بگذاریم با مقادیر اولیه کار خواهد کرد. در صورت لزوم می‌توان به آنها زمان دلخواه را به فرمت #T اختصاص داد (مثلاً T#5S).

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
```



شکل ۵-۳۵ فراخوانی FB در OB1 مثال ۵-۹

فصل ۵

به این روش FB1 را می‌توان برای چند میکسر نیز فراخوان کرد.

۵-۵ پرسش و تحقیق

آیا در سایر PLCها نیز فانکشن‌های سیستمی وجود دارد ؟

۵-۶ تمرین

مثال ۵- ۹ را به‌صورتی کامل کنید که پس از اینکه همزن از کار افتاد با تأخیر ۵ ثانیه ولو خروجی باز شود.

فصل ۶

برنامه‌نویسی راه‌اندازی PLC

در این فصل ضمن تشریح مد کاری Startup به نحوه برنامه‌نویسی راه‌اندازی در PLC پرداخته شده است.

چکیده مطالب

- وضعیت Startup یک وضعیت گذرای کاری CPU می‌باشد که در آن بررسی سخت‌افزار و انتقال تنظیمات به آنها و اجرای برنامه‌ی راه‌اندازی انجام می‌پذیرد.
- راه‌اندازی می‌تواند Warm یا Cold یا Hot باشد. تنظیم پیش فرض به‌صورت Warm است.
- راه‌اندازی‌های فوق ممکن است به‌صورت دستی توسط کاربر یا به‌صورت خودکار توسط سیستم عامل CPU باشد. حالت خودکار وقتی ممکن است که سیستم در مد RUN باشد و تغذیه قطع و وصل شود.
- برای هر نوع راه‌اندازی OB خاصی توسط CPU اجرا می‌شود.
- OB100 در راه‌اندازی Warm اجرا می‌شود.
- OB101 در راه‌اندازی Hot اجرا می‌شود.
- OB102 در راه‌اندازی Cold اجرا می‌شود.
- مدیریت راه‌اندازی مانند اختصاص مقادیر اولیه با برنامه‌نویسی در OBهای فوق انجام می‌شود.

۶-۱ مقدمه

همانطور که در کتاب سطح مقدماتی اشاره شد، وقتی CPU از مد کاری Stop به مد RUN گذر می‌کند، قبل از اینکه وارد سیکل کاری نرمال خود شود مرحله راه‌اندازی را طی می‌کند.

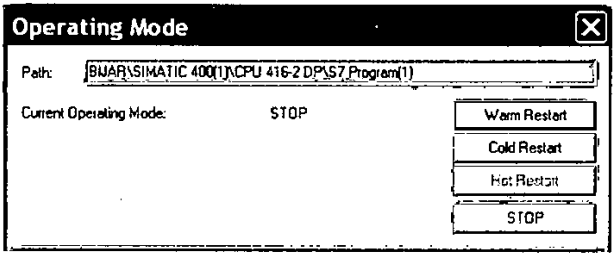
وضعیت Startup یک وضعیت گذرای کاری CPU می‌باشد که در آن اتفاقاتی مانند بررسی سخت‌افزار، اجرای برنامه‌ی راه‌اندازی و ... انجام می‌پذیرد. یکی از نکات مهم در زمان راه‌اندازی اجرای بلاک‌های راه‌اندازی است که با مدیریت این بلاک‌ها می‌توان برخی از مشکلات را کاهش داد. در این فصل ضمن بررسی رفتار CPU در راه‌اندازی، بلاک‌های راه‌اندازی و تنظیمات آنها تشریح می‌شود.

۶-۲ رفتار PLC در راه‌اندازی

قبل از اینکه CPU بتواند در مد RUN قرار بگیرد، ابتدا در وضعیت StartUP قرار می‌گیرد. به‌طور کلی خصوصیات وضعیت StartUP به شرح زیر می‌باشد:

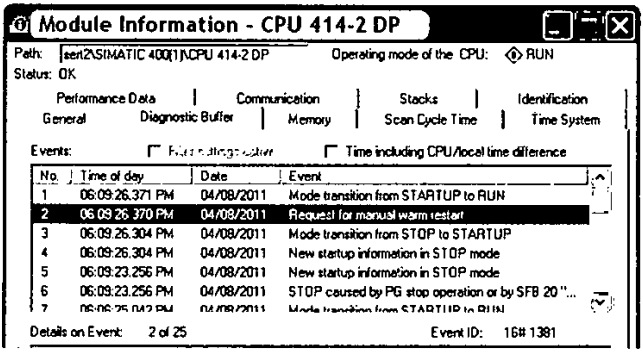
- اطلاعات سخت‌افزاری و شبکه که قبلاً پیکر بندی و دانلود شده در طول راه‌اندازی توسط CPU به ماژول‌های مربوطه ارسال می‌شود. فاز راه‌اندازی فاز شناسایی سخت‌افزار نیز هست. به همین علت است که وقتی شبکه پروفی‌باس تعریف می‌شود راه‌اندازی CPU طولانی‌تر است.
- در مد راه‌اندازی CPU بلاک‌های برنامه‌نویسی راه‌اندازی را در صورت وجود اجرا می‌کند.
- سنکرون‌سازی بین CPUهای Multicomputing در فاز راه‌اندازی انجام می‌شود.
- در سیستم‌های افزونه سنکرون‌سازی بین CPUهای Redundant در مد راه‌اندازی صورت می‌گیرد. در افزونگی نرم‌افزاری کاربر بایستی با استفاده از برنامه‌نویسی در بلاک‌های راه‌اندازی، سنکرون‌سازی را ایجاد نماید.
- در راه‌اندازی تایمرها Update می‌شوند و Run Time Meterها شروع به کار می‌کنند. RTMها در کتاب سطح تکمیلی بحث می‌شوند.
- در مد راه‌اندازی چراغ RUN روی CPU چشمک‌زن می‌شود.
- در راه‌اندازی خروجی‌های روی SMها غیر فعال می‌شوند.
- در راه‌اندازی CPU تحت تأثیر وقفه قرار نمی‌گیرد.
- در حین راه‌اندازی، CPU قابل نوشتن است یعنی می‌توان به آن دانلود کرد.
- در مرحله راه‌اندازی بلاک‌های خاصی که به بلاک‌های راه‌اندازی موسوم هستند در صورت وجود اجرا می‌شوند. در واقع متناسب با نوع راه‌اندازی یک OBخاص در نظر گرفته شده است که در زمان راه‌اندازی اجرا می‌شود. با برنامه‌ریزی این بلاک‌ها می‌توان برخی از تنظیمات مورد نظر در باره برنامه‌ی اصلی را انجام داد. مثلاً در برنامه‌راه‌اندازی می‌توان به برخی از متغیرها مقدار خاصی را اختصاص داد یا برخی از متغیرها را ری‌ست نمود.
- برنامه نوشته شده در OBهای راه‌اندازی (متناسب با نوع راه‌اندازی) اجرا می‌شود.
- سه نوع راه‌اندازی Hot , Cold , Warm وجود دارد که راه‌اندازی به یکی از این سه روش می‌تواند انجام شود. برای هر کدام از انواع راه‌اندازی فوق بلاک OB خاصی وجود دارد (OB100,OB102 , OB101).

- راه‌اندازی‌های فوق می‌تواند به‌صورت دستی[1] باشد. به‌عنوان مثال وقتی کاربر از طریق نرم‌افزار و مسیر PLC > Operating Mode اقدام به Stop و Start سیستم می‌کند، مد راه‌اندازی دستی است.



شکل ۶-۱ راه‌اندازی به‌صورت Manual

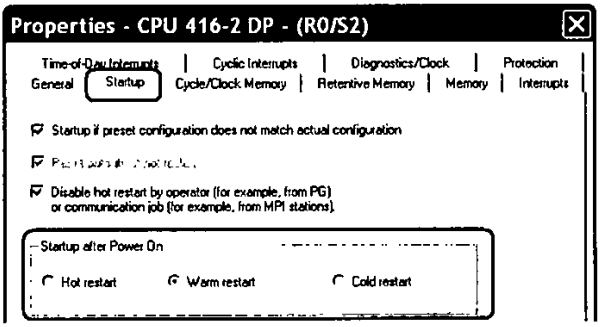
اگر راه‌اندازی به‌صورت دستی انجام شود پیغام Manual Restart در بافر مانند شکل ۶-۲ ثبت می‌گردد.



شکل ۶-۲ پیغام بافر در حالت راه‌اندازی دستی

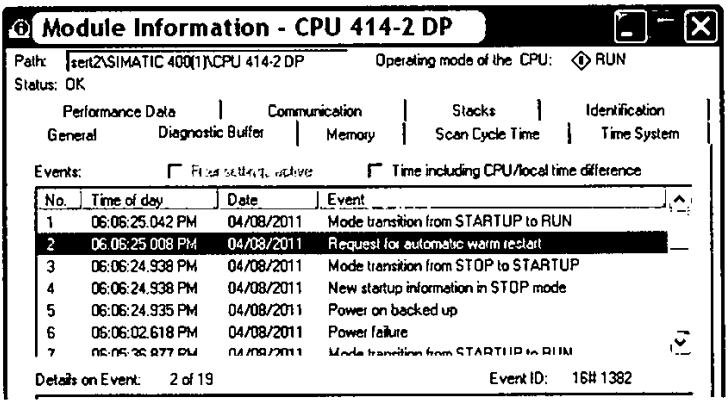
راه‌اندازی می‌تواند به‌صورت خودکار باشد، به این معنی که اگر سیستم در مد RUN باشد و تغذیه قطع و مجدداً وصل شود، CPU به‌طور خودکار راه‌اندازی می‌شود نوع راه‌اندازی بستگی به تنظیمی دارد که در پارامترهای CPU تنظیم شده است.

1. Manual



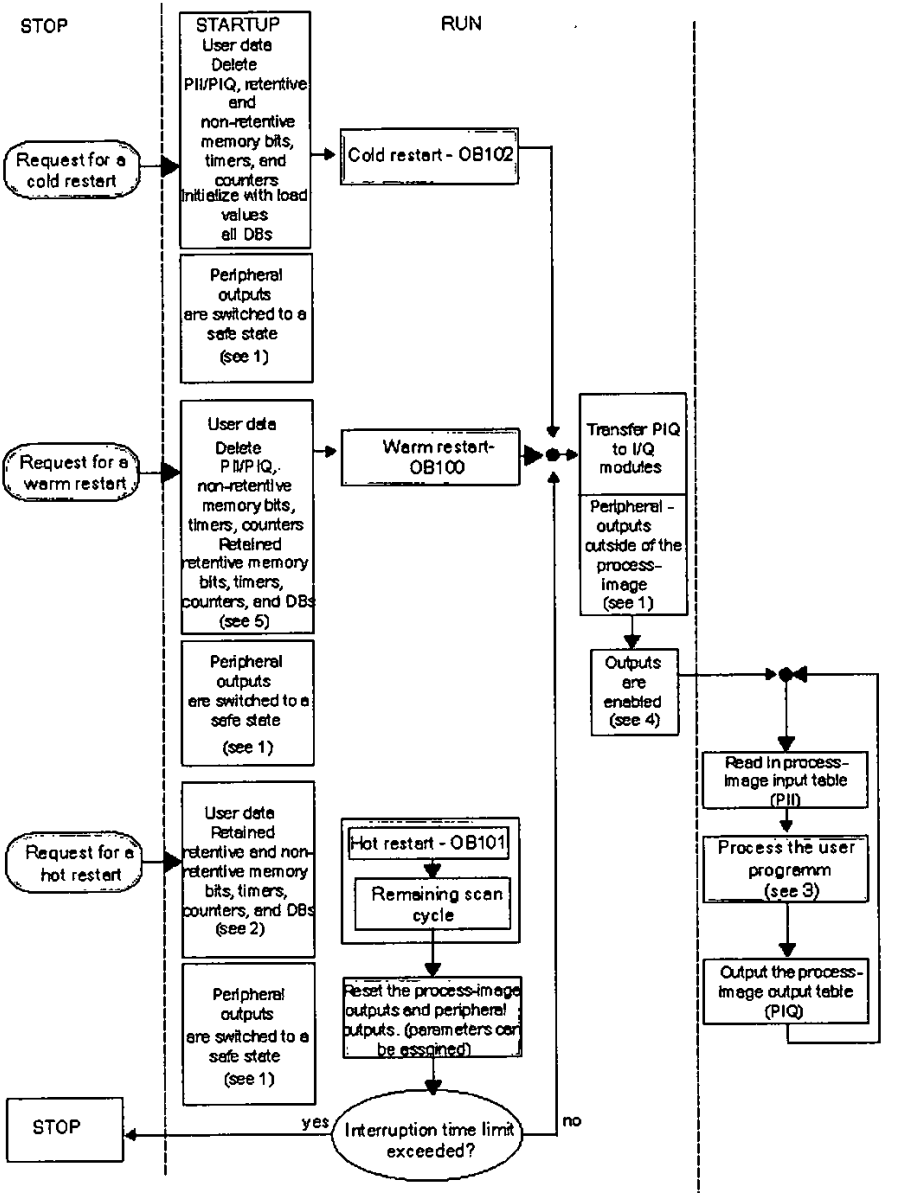
شکل ۶-۳ تنظیم نوع راه‌اندازی برای حالت خودکار

اگر راه‌اندازی به‌صورت خودکار پس از قطع و وصل تغذیه انجام شود، پیغام Automatic Restart در بافر مانند شکل ۶-۴ ثبت می‌گردد.



شکل ۶-۴ پیغام بافر در راه‌اندازی خودکار

شکل ۶-۵ رفتار PLC در مد راه‌اندازی را نشان می‌دهد. باید توجه نمود که برنامه OBهای راه‌اندازی فقط دز زمان راه‌اندازی CPU یک بار اجرا می‌شوند.



شکل ۶-۵ رفتار CPU در حالت راه‌اندازی

۶-۳ انواع راه‌اندازی

همانطور که اشاره شد، سه نوع راه‌اندازی برای CPU وجود دارد Warm,Cold,Hot که در زمان راه‌اندازی یکی از این سه مد انتخاب می‌شود. به‌طور پیش‌فرض برای CPUها مد Warm انتخاب شده است وCPU به‌طور اتوماتیک به روش Warm راه‌اندازی می‌شود، سایر مدها بایستی به‌صورت دستی تنظیم شوند تا CPU بر اساس آنها عمل کند. ویژگی‌های مدهای راه‌اندازی در کتاب سطح مقدماتی تشریح شد. در اینجا صرفاً عناوین این ویژگی‌ها را مرور می‌کنیم.

Warm Restart

- وقتی سوئیچ روی CPU در وضعیت RUN یا RUN-P باشد و تغذیه قطع و وصل شود، CPU به‌طور خودکار به‌صورت Warm راه‌اندازی می‌شود؛ مگر اینکه تنظیمات راه‌اندازی توسط کاربر به‌صورت دستی روی Cold یا Hot قرار گرفته باشد.
- اگر CPU درخواست Memory Reset داشته باشد (مانند وضعیتی که پس از تعویض کارت حافظه پیش می‌آید و چراغ Stop چشمک‌زن می‌شود)، در این وضعیت راه‌اندازی Warm امکان‌پذیر نیست.
- در راه‌اندازی Warm برنامه از ابتدا شروع می‌شود.
- در راه‌اندازی Warm حافظه Retentive پاک نمی‌شود.
- OB100 مربوط به راه‌اندازی Warm است.

Hot Restart

- این نوع راه‌اندازی خاص S7-400 است و S7-300 چنین قابلیتی ندارد.
- تنظیم راه‌اندازی Hot بایستی به‌صورت دستی در پارامترهای CPU انجام شده باشد.
- در این راه‌اندازی وجود باتری پشتیبان روی منبع تغذیه ضروری است.
- در راه‌اندازی HOT برنامه از جایی که قطع شده ادامه می‌یابد.
- در راه‌اندازی HOT هیچ‌یک از نواحی حافظه پاک نمی‌شود و متغیرها چه Retentive و چه non-Retentive همگی باقی می‌مانند.
- OB101 مربوط به این نوع راه‌اندازی است.

Cold Restart

- این نوع راه‌اندازی در CPUهای S7-300 فقط برای CPU318-2 وجود دارد ولی همه CPUهای 400 آن را پشتیبانی می‌کنند.
- در این نوع راه‌اندازی OB102 توسط CPU فراخوانی می‌شود.
- تنظیم آن در پارامترهای CPU مانند شکل ۶-۳ است.
- در این راه‌اندازی، اجرای برنامه کاربر از ابتدا آغاز می‌شود.
- در این راه‌اندازی کلیه‌ی محتویات System Memory، چه به صورت ماندگار[1] تعریف شده باشند یا به صورت غیر ماندگار، اطلاعات خود را از دست می‌دهند (مقادیر آنها پاک می‌شوند).

1. Retentive

جدول ۶-۱ به‌طور خلاصه عملیاتی که در هر کدام از روش‌های راه‌اندازی انجام می‌شود را نشان می‌دهد.

جدول ۶-۱

	In Warm Restart	In Cold Restart	In Hot Restart
Clear I stack/B stack	Yes	Yes	No
Clear volatile memory bits, timers, counters	Yes	No	No
Clear all memory bits, timers, counters	No	Yes	No
Clear process-image output table	Yes	Yes	selectable
Reset outputs of digital signal modules	Yes	Yes	selectable
Discard hardware interrupts	Yes	Yes	No
Discard time-delay interrupts	Yes	Yes	No
Discard diagnostic interrupts	Yes	Yes	Yes
Update the system status list (SZL)	Yes	Yes	Yes
Evaluate module parameters and transfer to modules or transfer default values	Yes	Yes	Yes
Execution of the relevant startup OB	Yes	Yes	Yes
Execute remaining cycle (part of the user program not executed due to the power down)	No	No	Yes
Update the process-image input table	Yes	Yes	Yes
Enable digital outputs (cancel OD signal) after transition to RUN	Yes	Yes	Yes

جدول ۶-۲ تأثیر هر کدام از انواع راه‌اندازی را روی بخش‌های مختلف حافظه CPU نشان می‌دهد.

جدول ۶-۲

	EPROM (Memory Card or Integrated)								
	CPU with Backup Battery				CPU without Backup Battery				
Data	Blocks in load memory	DB in work memo-ry	Memory bits, timers, counters (defined as retentive)	Memory bits, timers, counters (defined as volatile)	Blocks in load memory	DB in work memory (defined as re-tentive)	DB in work memory (defined as volatile	Memory bits, timers, counters (defined as re-tentive	Memory bits, timers, counters (defined as volatile)
Warm restart on S7-300	X	X	X	0	VC	VX	V	X	0
Warm restart on S7-400	X	X	X	0	VC	---	V	0	0

Cold restart on S7-300	X	0	0	0	VC	V	V	0	0
Cold restart on S7-400	X	0	0	0	VC	---	V	0	0
Hot restart on S7-400	X	X	X	X	Only warm restart per-mitted				

X	means	data retained
VC	means	logic block retained in EPROM, any overloaded logic blocks are lost
VX	means	data block is retained only if on the EPROM retentive data are taken from the NV-RAM (loaded or created data blocks in the RAM are lost)
0	means	data are reset or erased (content of DBs)
V	means	data are set to the initialization value taken from the EPROM memory
---	means	not possible as no NV-RAM available

۶-۴ OBهای راه‌اندازی

۶-۴-۱ انواع OBهای راه‌اندازی

OBهای راه‌اندازی فقط در زمان راه‌اندازی CPU (مد کاری Startup) اجرا می‌شوند. متناسب با هر نوع راه‌اندازی یک OB خاص در نظر گرفته شده است، این OBها عبارتند از:

- OB100 جهت راه‌اندازی Warm
- OB101 جهت راه‌اندازی Hot
- OB102 جهت راه‌اندازی Cold

زمانی که CPU در مد Startup قرار بگیرد، متناسب با اینکه نوع راه‌اندازی کدام‌یک از موارد فوق تعیین شده است، در صورت وجود OB مربوط به آنرا اجرا می‌کند. OBهای راه‌اندازی در شرایط عادی برنامه‌ی کاربر اجرا نمی‌شوند، از اینرو دستوراتی که در آنها نوشته شود فقط در زمان راه‌اندازی یک‌بار اجرا می‌شود. پس اگر در حین کار CPU یعنی در مد RUN دانلود شوند، اجرا نخواهند شد.

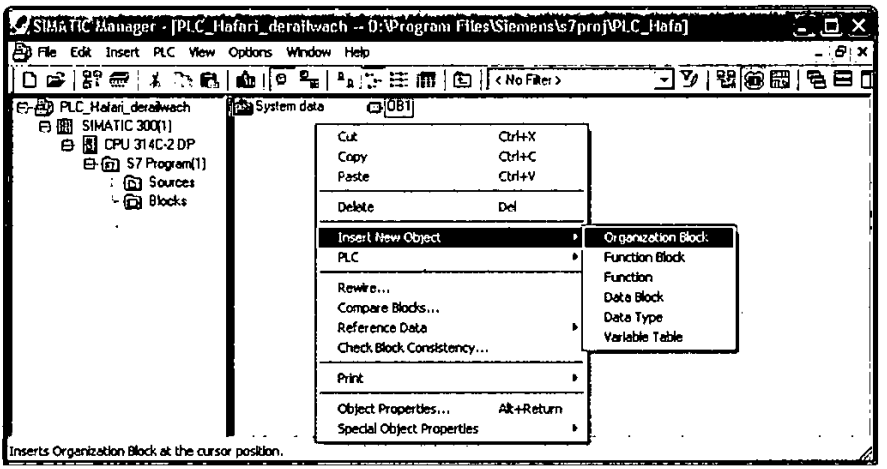
با توجه به نکات فوق کاربر می‌تواند این OBها را ایجاد نموده و با برنامه‌نویسی آنها شرایط راه‌اندازی را مدیریت کند. از جمله مواردی که در برنامه‌نویسی این OBها استفاده می‌شود می‌توان موارد زیر را نام برد:

- اختصاص مقدار اولیه به متغیرها
- تست وضعیت خروجی‌ها
- ساخت سیگنال‌های همیشه صفر و همیشه یک

نکته: عدم وجود OBهای راه‌اندازی اشکالی در کار CPU ایجاد نمی‌نماید.

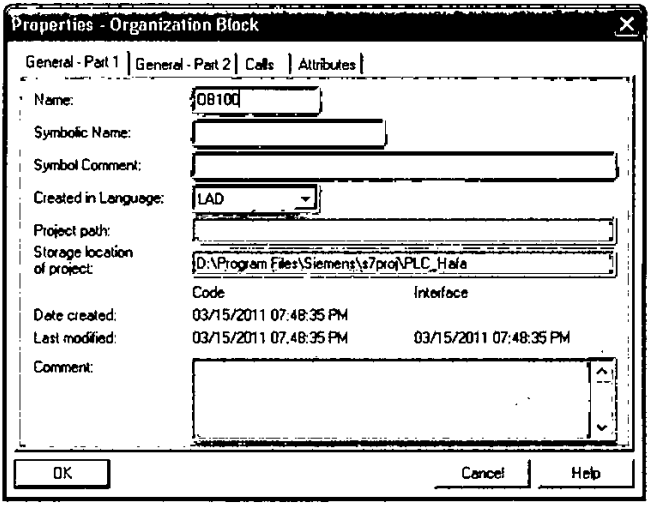
۶-۴-۲ ایجاد OBهای راه‌اندازی

برای ایجاد OBهای راه‌اندازی می‌توان مشابه روش ایجاد FC یا FB در محیط Simatic Manager عمل نموده و مانند شکل ۶-۶ گزینه Organization Block را انتخاب کرد.



شکل ۶-۶ ایجاد OB

هنگامی‌که OB مورد نظر ایجاد شد، باید شماره آنرا نیز تعیین نمود. برای این کار می‌توان در فیلدی که در زمان ایجاد OB نمایش داده می‌شود (شکل ۶-۷) شماره دلخواه را به‌عنوان شماره OB وارد نمود.



شکل ۶-۷ ایجاد OB100

۶-۴-۳ برنامه‌نویسی OBهای راه‌اندازی

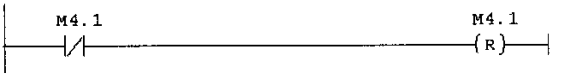
جهت برنامه‌نویسی OBهای راه‌اندازی می‌توان پس از ایجاد OB مورد نظر توسط نرم‌افزار LAD/STL/FBD آنرا باز نموده و به یکی از زبان‌های فوق برنامه مورد نظر را در آن نوشت.

مثال ۶-۱ ساخت سیگنال‌های همیشه یک و همیشه صفر

در برنامه‌های کاربردی معمولاً سیگنالی که وضعیت آن همیشه یک[1] و سیگنالی که وضعیت آن همواره صفر[2] باشد مورد نیاز است. به‌عنوان نمونه اگر خواننده فانکشن FC105 Scale را از بحث آنالوگ بیاد داشته باشد، ورودی Biopolar آن در برخی شرایط باید همیشه یک و در برخی شرایط باید همیشه صفر باشد. بهترین روش در برنامه‌نویسی آن است که در برنامه راه‌اندازی این سیگنال‌ها را ایجاد کرده و در برنامه‌های عادی نظیر OB1 و FB و FC و ... از آنها استفاده کنیم.
سیگنال‌های فوق را به روش‌های مختلف می‌توان به‌سادگی ایجاد نمود. شکل ۶-۸ نمونه‌هایی از روش ایجاد سیگنال همیشه یک و همیشه صفر را نشان می‌دهد.

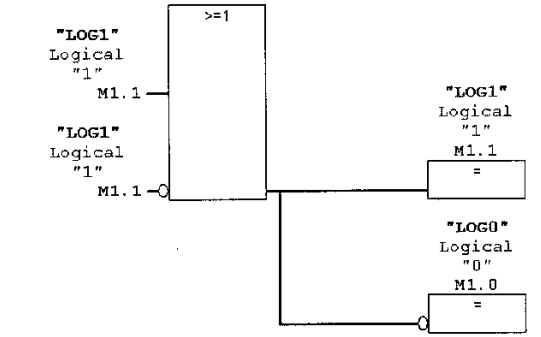
```
OB100 :  "Complete Restart"
Network 1: Title:
```


```
Network 2: Title:
```



روش دیگر

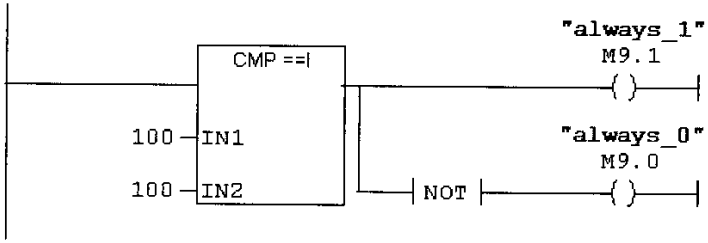
```
OB100 :  "Complete Restart"
Network 1:
```



1. always true
2. always false

روش دیگر

```
OB100 :  "Complete Restart"
Network 1: Title:
```



شکل ۶-۸ چند روش برای ساخت سیگنال‌های همیشه صفر و همیشه یک

نکته ۱: ساخت سیگنال‌های فوق در OB1 و برنامه‌های معمولی نیز امکان‌پذیر است، ولی دستورات فوق در OB راه‌اندازی فقط یک بار اجرا می‌شوند اما در OB1 در هر سیکل پردازش می‌گردند.

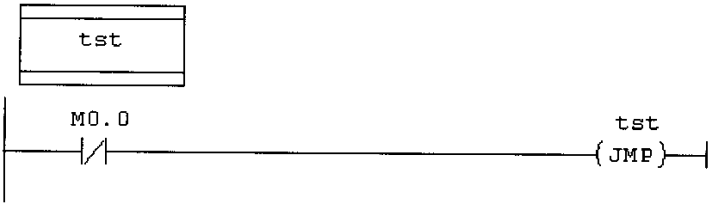
نکته ۲: اگر از مدهای راه‌اندازی Cold و Hot استفاده می‌شود، برنامه فوق در OB102 , OB101 نوشته می‌شود. به‌طور کلی بهتر است که در همه OBهای راه‌اندازی منطق فوق را داشته باشیم.

نکته ۳: حافظه‌هایی که برای سیگنال‌های فوق استفاده می‌شوند نباید در برنامه‌نویسی اشتباهاً روی آنها Write کرد. برای اجتناب از این اشتباه استفاده از نام سمبلیک برای حافظه‌های فوق توصیه می‌گردد.

مثال ۶-۲ لوپ در راه‌اندازی

اگر در برنامه راه‌اندازی پردازش CPU در لوپ بیفتد نمی‌تواند از آن خارج شود و در این مد باقی مانده و چراغ RUN چشمک‌زن باقی می‌ماند. در این وضعیت سیکل اسکن هنوز شروع نشده، بنابراین Watchdog آن که حد ماکزیمم سیکل اسکن است فعال نخواهد شد.

```
OB100 :  "Complete Restart"
Network 1: Title:
```



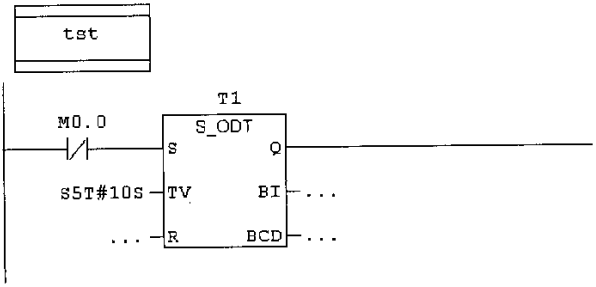
شکل ۶-۹ حلقه بی‌نهایت در OB راه‌اندازی

مثال ۶-۳ استفاده از تایمر در برنامه راه‌اندازی

اگر به‌طور معمولی در برنامه‌نویسی راه‌اندازی از تایمر استفاده شود، اگر زمان کار تایمر بیش از زمان راه‌اندازی باشد، نتیجه کار تایمر قابل استفاده نخواهد بود.

برنامه زیر در OB100 باعث می‌شود تا زمان‌سنجی تایمر تمام نشده سیستم در مد راه‌اندازی باقی بماند.

```
OB100 :  "Complete Restart"
Network 1: Title:
```



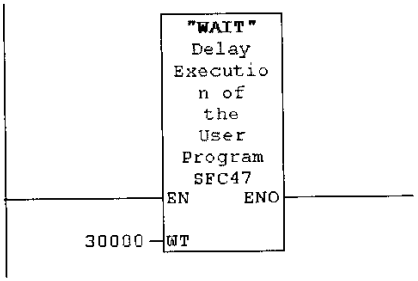

```
Network 2: Title:
```

شکل ۶-۱۰ استفاده از تایمر در راه‌اندازی

مثال ۶-۴ استفاده از SFC47 در برنامه راه‌اندازی

برنامه زیر CPU را به مدت ۳۰ میلی‌ثانیه در مد راه‌اندازی نگه می‌دارد.

```
OB100 :  "Complete Restart"
Network 1: Title:
```

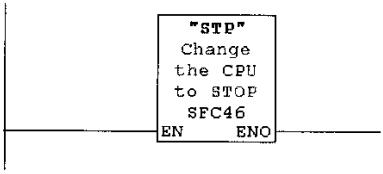


شکل ۶-۱۱ تأخیر در پردازش CPU در راه‌اندازی با SFC47

مثال ۶-۵ استفاده از SFC46 در برنامه راه‌اندازی

در یک فرآیند که با S7-400 کنترل می‌شود، شرایط به‌صورتی است که در صورت قطع و وصل تغذیه نباید سیستم به‌صورت Hot راه‌اندازی شود حتی اگر تنظیم راه‌اندازی توسط کاربر روی Hot Restart باشد.
برای این منظور کافیست در OB101 بلاک SFC46 را صدا بزنیم.

```
OB101 :  "Restart"
Network 1: Title:
```

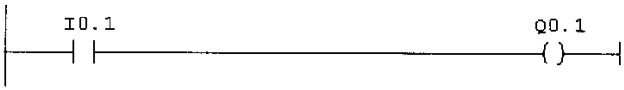


شکل ۶-۱۲ جلوگیری از راه‌اندازی CPU در حالت راه‌اندازی HOT

مثال ۶- ۶ استفاده از ورودی‌ها در برنامه راه‌اندازی

در ابتدای راه‌اندازی حافظه‌های PII و PIQ پاک می‌شوند. با توجه به توضیحاتی که در کتاب سطح مقدماتی در مورد سیکل اسکن داده شد، می‌دانیم که برنامه‌نویسی برای اعمال فرمان به خروجی‌ها در مد راه‌اندازی امکان‌پذیر است، زیرا CPU قبل از اجرای سیکل اسکن این فرامین را به PIQ می‌فرستد. ولی استفاده از آدرس‌های ورودی در مد راه‌اندازی امکان‌پذیر نیست زیرا هنوز CPU وارد سیکل اصلی که Update کردن PII در آن انجام می‌شود نشده است. به‌عنوان مثال، برنامه زیر حتی اگر در هنگام راه‌اندازی ورودی I0.0 فعال باشد نمی‌تواند فرمانی به Q0.0 بفرستد.

```
OB100 :  "Complete Restart"
Network 1: Title:
```



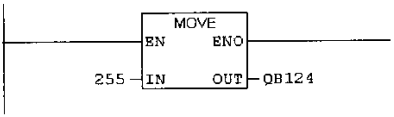
شکل ۶-۱۳ برنامه نادرست در راه‌اندازی به‌دلیل استفاده از آدرس ورودی

مثال ۶-۷

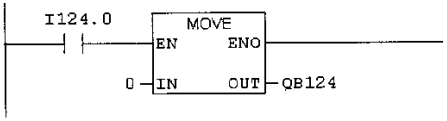
برنامه‌ای بنویسید که با شروع به کار PLC به‌طور اتوماتیک ۸ خروجی Q124.0 الی Q124.7 روشن شده و با فشردن شستی I124.0 خروجی‌های فوق خاموش شوند.

حل: جهت روشن نمودن همزمان ۸ خروجی می‌توان مقدار ۲۵۵ را توسط دستور Move به خانه QB124 منتقل کرده و جهت خاموش نمودن خروجی‌ها مقدار 0 را به QB124 منتقل نمود. اگر برنامه در OB1 نوشته شود، خروجی‌ها روشن شده ولی خاموش نمی‌شوند، چون برنامه‌ی OB1 به‌طور دائم اجرا می‌گردد لذا دائما خروجی‌ها روشن می‌باشند؛ اما اگر برنامه روشن‌شدن خروجی‌ها در OB100 و خاموش شدن خروجی‌ها در OB1 نوشته شود مشکل فوق برطرف می‌شود.

```
OB100 :  "Complete Restart"
Network 1: Title:
```




```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
```

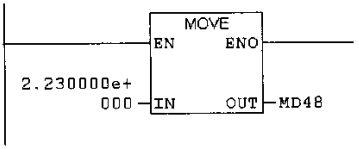


شکل ۶-۱۴ برنامه مثال ۶-۷

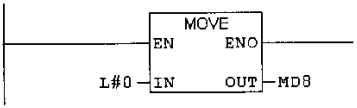
مثال ۶-۸ اختصاص مقدار اولیه

یکی از کاربردهای OBهای راه‌اندازی اختصاص مقدار اولیه به متغیرهاست. به‌عنوان مثال در صورت نیاز می‌توان متغیرهای Memory Bit را با مقادیر اولیه پر کرد تا در هر بار قطع و وصل تغذیه مقدار اولیه مورد نظر در آنها قرار بگیرد. شکل ۶-۱۵ نمونه‌ای از این برنامه را در OB100 نشان می‌دهد.

```
OB100 : Title:
Network 1: Title:
```




```
Network 2: Title:
```

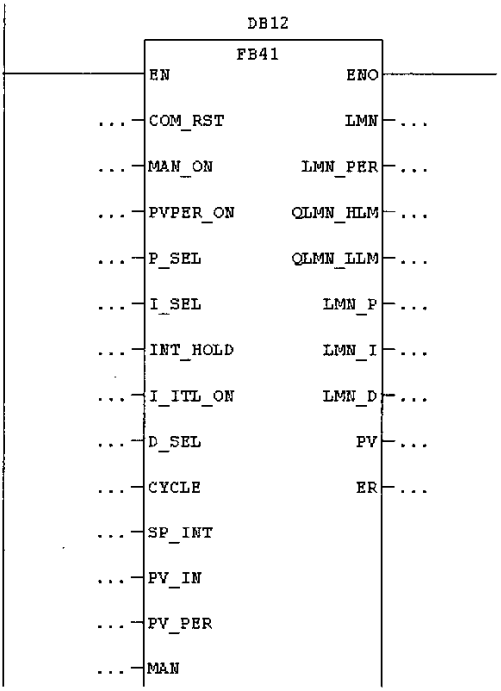


شکل ۶-۱۵ اختصاص مقادیر اولیه به متغیرها در راه‌اندازی

در برخی کاربردها لازم می‌شود تا در هر بار راه‌اندازی دیتا بلاک اختصاص FB با مقادیر اولیه تنظیم شود. برای این منظور نیز از OBهای راه‌اندازی استفاده می‌گردد. به‌عنوان مثال اگر در S7-300 در بین کار یک لوپ کنترلی که با

FB41 کار می‌کند تغذیه قطع و وصل شود آخرین دیتاهای قبلی در DB باقی می‌ماند. برای اینکه در راه‌اندازی DB فوق ری‌ست شود کافیست آنرا در OB100 بدون اینکه هیچ پارامتری به ورودی یا خروجی آن اختصاص دهیم صدا بزنیم.

```
OB100 :  "Complete Restart"
Network 1 : Title:
```



شکل ۶-۱۶ Initialize کردن لوپ کنترلی FB41 در راه‌اندازی

۶-۴-۴ استفاده از پارامترهای Temp در OBهای راه‌اندازی

همانطور که در فصل ۴ اشاره شد، هر OB دارای تعدادی متغیر محلی از نوع Temp می‌باشد که اطلاعاتی را در اختیار کاربر قرار می‌دهد. نوع اطلاعات در هر OB ممکن است با سایر OBها متفاوت باشد اما کلیه OBهای راه‌اندازی دارای متغیرهای Temp یکسانی هستند. جدول ۶-۳متغیرهای Temp این OBها را نشان می‌دهد.

جدول ۶-۳

Variable	Type	Description
OB10x_EV_CLASS	BYTE	Event class and identifiers: B#16#13: active
OB10x_STRTUP	BYTE	Startup request: • B#16#81: Manual warm restart • B#16#82: Automatic warm restart • B#16#83: • Request for manual hot restart B#16#84: • Request for automatic hot restart B#16#85: • Request for manual cold restart B#16#86: • Request for automatic cold restart B#16#87: • Master: Request for manual cold B#16#88: restart Master: Request for automatic cold restart • B#16#8A: Master: Request for manual warm restart • B#16#8B: Master: Request for automatic warm restart • B#16#8C: Standby: Request for manual restart • B#16#8D: Standby: Request for automatic restart
OB10x_PRIORITY	BYTE	Priority class: 27
OB10x_OB_NUMBR	BYTE	OB number (100, 101, or 102)
OB10x_RESERVED_1	BYTE	Reserved
OB10x_RESERVED_2	BYTE	Reserved
OB10x_STOP	WORD	Number of the event that caused the CPU to stop
OB10x_STRT_INFO	DWORD	Supplementary information about the current startup
OB10x_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

همانطور که در جدول ۶-۳ دیده می‌شود، هر کدام از متغیرهای Temp اطلاعات خاصی را در اختیار کاربر قرار می‌دهند. کاربر می‌تواند با مدیریت این متغیرها اطلاعات جامعی از راه‌اندازی CPU به‌دست آورد. مثلاً مشخص است که متغیر محلی OB10x_STARTUP نوع راه‌اندازی را نشان می‌دهد، بنابراین با انجام یک برنامه‌نویسی صحیح می‌توان نوع راه‌اندازی را برای اپراتور یا کاربر مشخص نمود.

می‌توان در هر یک از OBهای 100 ,101,102 این متغیر را در حافظه‌ای ذخیره کرد و از کدهایی که به این روش به‌دست می‌آید در OB1 استفاده نمود.

اطلاعات کمکی بیشتر در مورد راه‌اندازی در متغیر OB10X_STRT_INFO که یک Dword است ذخیره می‌شود. ۸ بیت آخر طبق جدول ۶-۴ اطلاعات اضافی را برمی‌گرداند. همانطور که در این جدول دیده می‌شود، برخی کدها فقط برای سیستم‌های افزونه H ظاهر می‌شود.

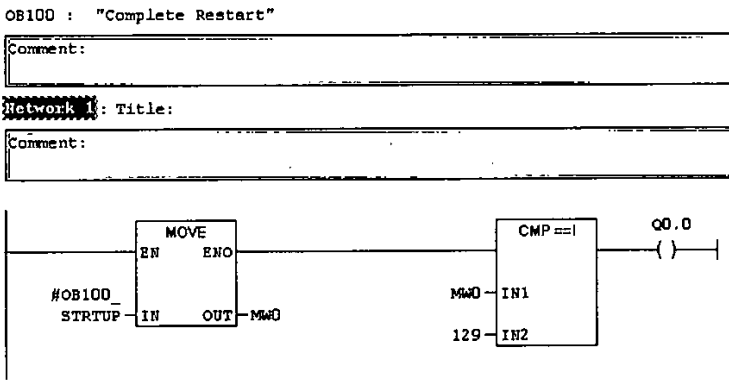
جدول ۶-۴

Bit No.	Meaning	Possible Binary Values	Explanation
31 - 24	Startup Information	0000 xxxx	Rack number 0 (H CPUs only)
		0100 xxxx	Rack number 1 (H CPUs only)
		1000 xxxx	Rack number 2 (H CPUs only)
		0001 xxxx	Multicomputing (S7-400 only)
		0010 xxxx	Operation of more than one CPU in the segmented rack (S7-400 only)
		xxxx xxx0	No difference between expected and actual configuration (S7-300 only)
		xxxx xxx1	Difference between expected and actual configuration (S7-300 only)

مثال ۶-۹

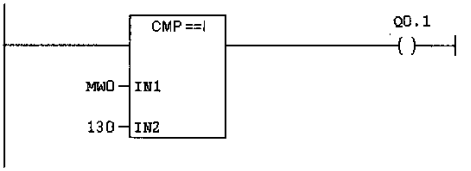
برنامه‌ای بنویسید که در صورت انجام راه‌اندازی Manual Warm لامپ Q0.0 و در صورت راه‌اندازی Automotic Warm لامپ Q0.1 روشن شود.

حل: از آنجایی که مثال مربوط به راه‌اندازی Warm می‌باشد لذا باید از OB100 استفاده نمود. از روی جدول ۶-۴ مشخص است که متغیر OB100 Startup کدی را می‌دهد که نشان‌دهنده نوع راه‌اندازی است. برای راه‌اندازی Manual Warm کد B#16#81 که معادل عدد ۱۲۹ در فرمت Integer می‌باشد نمایش داده شده و برای Automotic Warm کد B#16#82 که معادل عدد ۱۳۰ در فرمت Integer است نمایش داده می‌شود. برنامه مورد نظر را می‌توان به روش زیر در OB100 پیاده‌سازی نمود.



Network 2 : Title:

Comment:



شکل ۶-۱۷ برنامه مثال ۶-۹

۶-۵ پرسش و تحقیق

در مورد برنامه‌نویسی راه‌اندازی در سایر PLCها تحقیق کنید.

۶-۶ تمرین

برنامه‌ای بنویسید که با توجه به نوع راه‌اندازی خروجی‌های زیر روشن شوند:

- راه‌اندازی Warm: Q124.5
- راه‌اندازی Cold: Q124.6
- راه‌اندازی Hot: Q124.7

فصل ۷

برنامه‌نویسی وقفه‌ها در PLC

در این فصل انواع وقفه‌های موجود در PLC همراه با مثال‌های متنوع تشریح شده است.

فصل ۷

چکیده مطالب

- استفاده از برنامه‌نویسی وقفه‌ها در کنترل فرآیندهای صنعتی بزرگ اجتناب‌ناپذیر است.
- وقفه‌ها به‌دلیل اولویت بالایی که دارند کار نرمال CPU را برای لحظاتی قطع می‌کنند، از اینرو می‌توانند سیکل اسکن را افزایش دهند.
- برخی وقفه‌ها مبتنی بر زمان، برخی مبتنی بر Event و برخی دیگر مبتنی بر Error می‌باشند.
- هر گروه از وقفه‌ها دارای OBهای خاص خود هستند.
- OB1x مربوط به وقفه‌هایی است که در تاریخ و زمان مشخصی فعال می‌شوند. تکرار آنها بسته به تنظیمات کاربر دارد.
- OB2x مربوط به وقفه‌هایی است که در اثر بروز شرایط خاص با تأخیر اجرا می‌گردند.
- OB3x وقفه‌هایی هستند که طبق سیکل زمانی مشخص به‌طور دائم تکرار می‌شوند.
- OB4x وقفه‌هایی هستند که اجرای آنها وابسته به یک رخداد سخت‌افزاری است.
- OB5x وقفه‌های مربوط به شبکه Profibus DP هستند.
- OB6x وقفه‌های مربوط به سیستم‌های Multicomputing هستند.
- OB7x وقفه‌های مربوط به سیستم‌های افزونه S7-400H می‌باشند.
- OB8x وقفه‌های مبتنی بر اشکالات سخت‌افزاری هستند.
- OB12x وقفه‌های مربوط به اشکالات برنامه‌نویسی هستند.

۷-۱ مقدمه

یکی از امکانات پیشرفته برنامه‌نویسی در PLC استفاده از امکانات وقفه‌هاست. مطالبی که در کتاب سطح مقدماتی و این کتاب تا اینجا بیان شده است همگی بر مبنای برنامه‌نویسی نرمال در OB1 بوده است. در این فصل خواهیم دید که با شناخت وقفه‌ها و برنامه‌نویسی آنها خواننده می‌تواند به افق‌های جدیدی در برنامه‌نویسی دست یابد و منطق‌های پیچیده‌تر را ساده‌تر پیاده‌سازی نماید.

وقفه[1] را به‌صورت ساده می‌توان با عبارت زیر بیان کرد:

*هر عاملی که **کار نرمال را برای لحظاتی** متوقف کند وقفه نامیده می‌شود.*

در جمله فوق روی دو کلمه تاکید شده است: کار نرمال و برای لحظاتی. در زندگی روزمره وقتی مشغول انجام یک کار روتین هستیم و در بین آن کار مهمتری پیش می‌آید کار روتین را برای لحظاتی قطع کرده و به کار مهمتر می‌پردازیم ولی پس از پایان کار مهم به کار روتین قبلی برگشته و آن را ادامه می‌دهیم.

از عبارت فوق می‌توان نتیجه گرفت که:

اهمیت وقفه‌ها نسبت به کار نرمال بیشتر است به همین جهت می‌توانند آنرا قطع کنند.

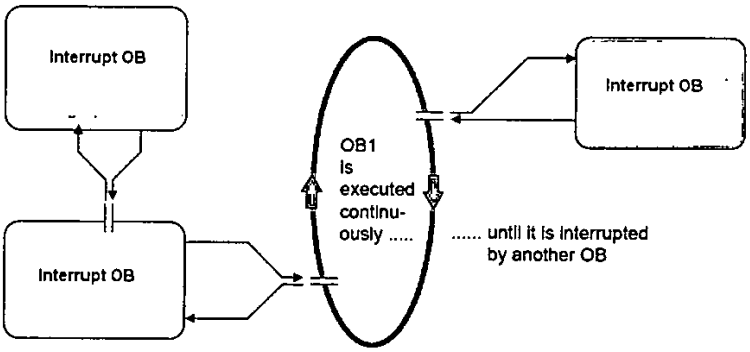
ممکن است در حین اجرای یک وقفه، وقفه دیگری که از قبلی مهمتر بوده پیش بیاید. بدیهی است وقفه جدید می‌تواند وقفه قبلی را نیز قطع کند. پس وقفه‌ها ممکن است تودرتو اتفاق بیفتند.

استفاده از وقفه منحصر به PLC نیست. وقفه‌ها یکی از اجزای مهم در معماری کامپیوتر هستند. توسط امکانات وقفه پردازنده می‌تواند اجرای دستورالعمل‌های جاری را موقتاً متوقف کرده و به اجرای دستورات دیگری بپردازد. بنابراین وقفه به مفهوم توقف کار پردازش CPU نیست بلکه انتقال اجرا از برنامه جاری به برنامه دیگر است.

با این توضیحات به PLC برمی‌گردیم. وقتی PLC استارت می‌شود ابتدا به اجرای فرامین راه‌اندازی که در OB100 یا OB101 یا OB102 نوشته شده می‌پردازد سپس کار روتین خود را شروع می‌کند. کار نرمال در PLC اجرای سیکل اسکن است و تا زمانی که مشکلی به‌وجود نیامده این کار به‌طور مداوم تکرار می‌شود. برنامه‌ای که CPU در سیکل اسکن به‌طور نرمال اجرا می‌کند OB1 است که Free Cycle خوانده می‌شود. OB1 از دیدگاه سیستم عامل درجه اهمیت کم دارد. اگر شرایطی پیش بیاید که وقفه‌ای فعال شود به‌صورت زیر عمل می‌شود:

- OB1 در هر نقطه‌ای که باشد قطع می‌شود و اطلاعات محل قطع شدن در یک Stack ذخیره می‌شود.
- برنامه وقفه اجرا می‌شود.
- پس از پایان برنامه وقفه، OB1 با اطلاعات ذخیره شده قبلی کار را ادامه می‌دهد.

1. Interrupu



شکل ۷-۱ تأثیر وقفه در پردازش بلاک‌ها

منظور از برنامه وقفه، OB با شماره خاصی است که برای CPU تعریف شده است.

نکته‌ای که بایستی به آن توجه شود این است که برنامه وقفه را نباید با FC و FB اشتباه کرد. FC و FB در برنامه توسط برنامه‌نویس فراخوان می‌شوند ولی برنامه وقفه که در OB خاصی نوشته می‌شود دستوری برای فراخوانی ندارد. در واقع OBها که بلاک‌های پایه هستند توسط سیستم عامل CPU فراخوان می‌شوند. بنابراین بدون اینکه در OB1 اثری از فراخوانی OB وقفه ببینیم عملاً سیستم عامل، OB1 را در شرایط وقفه قطع می‌کند و OB وقفه را فراخوان می‌نماید.

به‌طور کلی وقفه‌ها به دو صورت می‌توانند فعال شوند:

- **توسط عامل داخلی CPU:** عامل داخلی می‌تواند یک برنامه یا تنظیم خاصی یا فالت داخلی در خود CPU باشد.
- **توسط عامل خارجی:** عامل خارجی می‌تواند ناشی از یک سخت‌افزار بیرون CPU نظیر ماژول I/O یا منبع تغذیه یا اجزای شبکه باشد.

خلاصه نکات مهم وقفه‌ها

۱- اهمیت وقفه‌ها نسبت به برنامه روتین بیشتر است.

۲- هر وقفه دارای یک درجه اهمیت است.

۳- هر وقفه‌ای که دارای درجه اهمیت بالاتر باشد نه تنها برنامه روتین بلکه وقفه‌های کم اهمیت‌تر ازخودرامی‌تواند قطع کند.

۴- هر وقفه با یک OB یا یک دسته OB معرفی می‌شود.

۵- OB1 کمترین اولویت را دارد و OBهای وقفه می‌توانند آنرا قطع کنند.

۶- صدا زدن OB وقفه در بین اجرای یک OB دیگر توسط سیستم عامل CPU انجام می‌شود. دستوری برای فراخوانی وقفه نداریم.

۷- وقفه‌ها را می‌توان توسط فانکشن‌های سیستمی خاص غیر فعال[1] یا فعال[2] نمود.

۸- OBهای راه‌اندازی به‌دلیل درجه اولویت بالا تحت تأثیر بسیاری از وقفه‌ها قرار نمی‌گیرند.

1. Mask
2. Unmask

۷-۲ کاربرد وقفه‌ها

در CPU فقط با یک وقفه سر و کار نداریم. وقفه‌ها دارای انواع مختلف و دسته‌بندی‌های مختلف هستند که هر کدام برای کاربرد خاصی طراحی شده‌اند. تشریح کامل خانواده وقفه‌ها در ادامه آمده است. در اینجا صرفاً به نمونه‌ای از کاربردهای آنها اشاره می‌کنیم.

اجرای برنامه خاص در تاریخ و زمان و مشخص

اگر نیاز باشد که برنامه‌ای در تاریخ و زمان مشخصی اجرا شود، یک روش این است که در سیکل روتین برنامه‌ای به‌طور مداوم تاریخ و زمان CPU را چک کند و به محض رسیدن به زمان مورد نظر دستورات لازم را اجرا کند. بدیهی است این روش پردازش مداومی را توسط CPU نیاز دارد. روش دیگر این است که از وقفه مربوط به تاریخ و زمان استفاده شود. در این حالت در سیکل روتین پردازش فوق را نداریم و به محض رسیدن تاریخ و زمان CPU به تاریخ و زمان تعیین شده وقفه فعال شده و برنامه را اجرا می‌نماید.

اجرای برنامه خاص به‌صورت منظم با فاصله زمانی ثابت

اجرای برنامه‌های سیکل روتین در PLC هیچ نظم خاصی ندارند. اگر برنامه موجود در OB1 کم باشد، تکرار اجرای آن سریعتر اتفاق می‌افتد و اگر برنامه OB1 زیاد باشد، زمان سیکل‌های اجرا طولانی‌تر خواهد بود. به همین علت به OB1 اصطلاحاً Free Cycle گفته می‌شود.

اگر لازم باشد که CPU بدون توجه به کم یا زیاد بودن برنامه، آنرا طبق زمان‌بندی ثابت و منظمی اجرا کند این کار توسط OB1 امکان پذیر نیست و نیاز به وقفه‌هایی دارد که به‌صورت سیکلی اجرا می‌شوند.

اجرای فوری برنامه خاص در صورت تغییر شرایط یک سیگنال

همانطور که قبلاً ذکر شد، CPU در ابتدای سیکل اسکن وضعیت ورودی‌ها را می‌خواند و در حافظه PII ذخیره می‌کند، سپس در برنامه از مقدار ذخیره شده در حافظه استفاده می‌کند. اگر سیکل اسکن به‌دلیل وجود برنامه طولانی زیاد باشد ممکن است فرامینی که بر اساس خواندن یک ورودی تولید می‌شود با تأخیر باشد. برای رفع این مشکل می‌توان از وقفه‌های سخت‌افزاری استفاده کرد. در این شرایط به محض فعال شدن سیگنال برنامه روتین قطع شده و برنامه وقفه سریعاً اجرا خواهد شد.

اجرای برنامه خاص در صورت بروز فالت داخلی یا خارجی

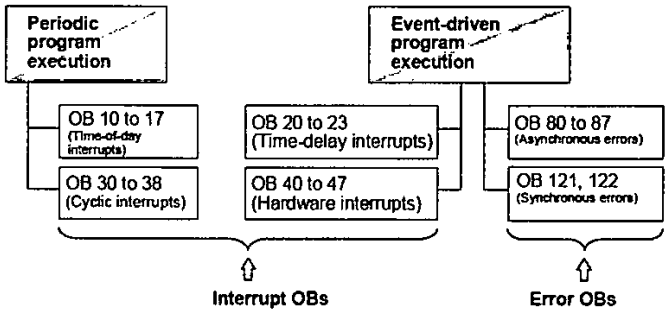
در شرایط عادی وقتی فالت داخلی یا خارجی روی CPU رخ می‌دهد، رفتار آن را تحت تأثیر قرار می‌دهد و ممکن است پردازش را متوقف نماید. با استفاده از وقفه‌های خاص نه تنها می‌توان از توقف CPU جلوگیری کرد بلکه می‌توان در شرایط بروز فالت، کنترل سیستم را به سمت شرایط مورد نظر هدایت نمود.

۷-۳ انواع وقفه‌ها

وقفه‌ها را از دیدگاه‌های مختلف می‌توان دسته‌بندی نمود.

دسته‌بندی از نظر زمان اجرا

- **وقفه‌های پریودیک:** اساس فعال شدن این وقفه‌ها زمان CPU است و اجرا بر اساس آن صورت می‌گیرد.
- **وقفه‌های غیرپریودیک:** این وقفه‌ها وابسته به زمان نیستند بلکه در شرایط خاصی که شرط آنها فعال شود اجرا شوند.



شکل ۷-۲ انواع OBهای وقفه موجود در S7

تقسیم‌بندی بر اساس Event /Error

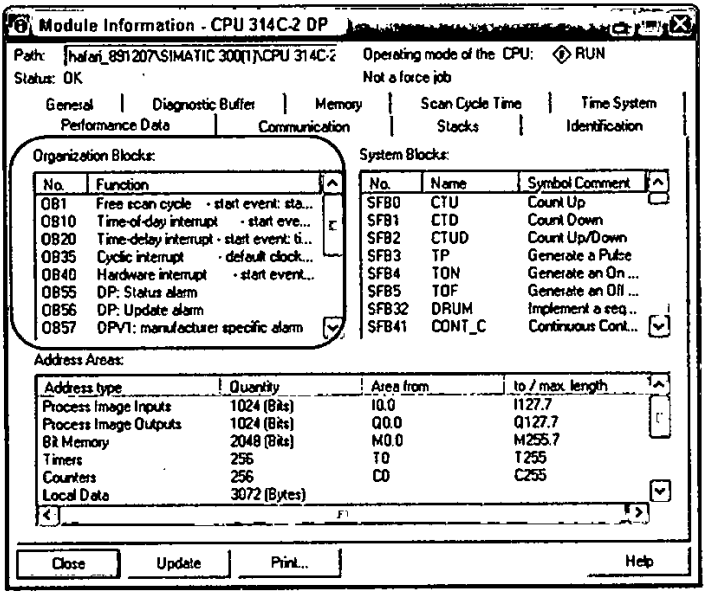
- **وقفه‌های مبتنی بر Event**: عملکرد این وقفه‌ها مبتنی بر یک رخداد است. رخداد می‌تواند ناشی از یک عامل خارجی مانند تحریک یک سیگنال باشد. تحریک سیگنال مزبور به معنی فالت نیست.
- **وقفه‌های مبتنی بر Error**: در حالت قبل رخدادی که باعث فعال شدن وقفه می‌شد یک حالت نرمال بود ولی در این حالت اشکال یا فالت داخلی یا خارجی باعث فعال شدن وقفه می‌گردد.

با توجه به توضیحات فوق مهمترین OBهای PLC در جدول ۷-۱ دسته‌بندی و مشخص شده‌اند.

جدول ۷-۱

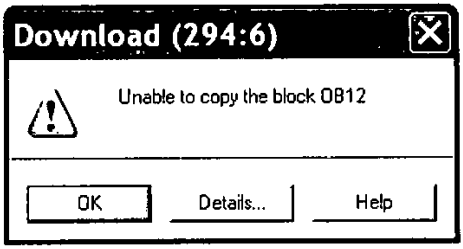
اجرای سیکلی برنامه	OB1	Main Cycle Execution
راه‌اندازی	OB100	Warm Restart
	OB101	Hot Restart
	OB102	Cold Restart
وقفه‌های مبتنی بر Event	OB10 to OB17	Time of Day Interrupt
	OB20 to OB23	Time Delay Interrupt
	OB30 to OB38	Cyclic Interrupt
	OB40 to OB47	Hardware Interrupt
	و ...	
وقفه‌های مبتنی بر Error	OB80 to OB87	Asynchronous Errors
	OB121 و OB1122	synchronous Errors

بر این اساس، خانواده‌های مختلفی از وقفه‌ها را در لیست فوق مشاهده می‌کنیم. این خانواده‌ها در زیر به اجمال معرفی شده‌اند و تشریح آنها در ادامه خواهد آمد. نکته‌ای که بایستی به آن توجه کرد این است که هر CPU تعداد مشخصی از این OBها را ساپورت می‌کند بنابراین ممکن است در یک CPU خاص برخی از آنها قابل استفاده نباشند. لیست کامل OB هایی که در CPU قابل استفاده است را می‌توان در پنجره Module Information در سربرگ Performance Data مشاهده نمود.



شکل ۷-۳ لیست OBها در پنجره Module Info

در صورت استفاده از OBهایی که در لیست فوق موجود نیستند، در هنگام دانلود با پیغام خطای زیر مواجه می‌شویم که مشابه پیغام غیر مجاز بودن آدرس‌هاست.



شکل ۷-۴ خطای استفاده از OB غیرمجاز

خانواده Time of Day Interrupt

این خانواده برای وقفه‌های تاریخ و زمانی به‌کار می‌روند. با استفاده از آنها CPU می‌تواند برنامه خاصی را در تاریخ مشخصی اجرا و در صورت لزوم تکرار کند. استفاده از این وقفه نیاز به تنظیماتی در پارمترهای CPU دارد. این وقفه با OB10 تا OB17 برنامه‌نویسی می‌شود.

خانواده Time Delay Interrupt

توسط این وقفه می‌توان در اجرای بخشی از برنامه تأخیر ایجاد نمود. این کار می‌تواند با دقت بالا انجام شود و زمان تأخیر می‌تواند با دقت 1ms باشد.

این وقفه با OB20 تا OB23 برنامه‌نویسی می‌شود.

خانواده Cyclic Interrupt

توسط این وقفه‌ها CPU طبق نظم و زمان‌بندی خاصی برنامه وقفه را اجرا خواهد کرد. این وقفه بدون نیاز به تنظیمات خاصی در صورت وجود بلاک‌های OB ذکر شده طبق زمان‌های پیش‌فرض اجرا خواهد شد. کاربر در صورت لزوم می‌تواند این زمان‌ها را تغییر دهد.

این وقفه با OB30 تا OB38 برنامه‌نویسی می‌شود.

خانواده Hardware Interrupt

توسط این وقفه‌ها در صورت بروز شراط خاص سخت افزاری (Event) برنامه وقفه اجرا خواهد شد. استفاده از این وقفه در صورتی امکان‌پذیر است که ماژول سخت‌افزاری قابلیت این وقفه را داشته باشد و این قابلیت فعال شده باشد.

این وقفه با OB40 تا OB47 برنامه‌نویسی می‌شود.

خانواده Asynchronous Error Interrupt

این وقفه‌ها در صورت بروز فالت آسنکرون توسط CPU اجرا می‌شوند. منظور از فالت‌های آسنکرون عمدتاً خطای سخت‌افزاری است که به‌صورت داخلی یا خارجی برای CPU رخ می‌دهد. این وقفه‌ها بیشترین درجه اولویت را دارند و می‌توانند سایر وقفه‌ها را قطع کنند.

این وقفه با OB80 تا OB87 برنامه‌نویسی می‌شود.

خانواده Synchronous Error Interrupt

این وقفه‌ها در صورت بروز فالت‌های سنکرون که مربوط به برنامه‌نویسی هستند فراخوان می‌شوند. برنامه‌نویسی آنها با OB121 و OB121 انجام می‌شود.

۷-۴ اولویت‌بندی وقفه‌ها

درجه اولویت OB وقفه‌ها و سایر OBها در جدول ۷-۲ آورده شده است. توجه به نکات زیر ضروری است:

- بیشترین درجه اهمیت وقفه‌ها 28 است که مربوط به وقفه‌های خطای آسنکرون می‌باشد.
- درجه اهمیت OBهای راه‌اندازی 27 است، بنابراین فقط OBهایی با اولویت 28 می‌توانند آنرا قطع کنند.

درجه اولویت OBها در S7-300 قابل تغییر نیست و در S7-400 نیز هر OB را در بازه خاصی می‌توان تغییر داد. به‌عنوان مثال نمی‌توان درجه اولویت 26 را به 2 تقلیل داد ولی می‌توان آنرا به 25 یا 24 کاهش داد.

- اگر درجه اولویت قابل تغییر باشد و به آن صفر اختصاص داده شود، OB مربوطه غیرفعال می‌گردد. در OBهای مبتنی بر Error نمی‌توان درجه اولویت را صفر کرد.

جدول ۷-۲

OB	Start Event	Default	Explanation
OB1	End of startup or end of OB1	1	Free cycle
OB10	Time-of-day interrupt 0	2	
OB11	Time-of-day interrupt 1	2	
OB12	Time-of-day interrupt 2	2	Time of Day Interrupt
OB13	Time-of-day interrupt 3	2	
OB14	Time-of-day interrupt 4	2	
OB15	Time-of-day interrupt 5	2	
OB16	Time-of-day interrupt 6	2	
OB17	Time-of-day interrupt 7	2	
OB20	Time-delay interrupt 0	3	
OB21	Time-delay interrupt 1	4	Time Delay Interrupt
OB22	Time-delay interrupt 2	5	
OB23	Time-delay interrupt 3	6	
OB30	Cyclic interrupt 0 (default interval: 5 s)	7	
OB31	Cyclic interrupt 1 (default interval: 2 s)	8	
OB32	Cyclic interrupt 2 (default interval: 1 s)	9	Cyclic interrupts
OB33	Cyclic interrupt 3 (default interval: 500 ms)	10	
OB34	Cyclic interrupt 4 (default interval: 200 ms)	11	
OB35	Cyclic interrupt 5 (default interval: 100 ms)	12	
OB36	Cyclic interrupt 6 (default interval: 50 ms)	13	
OB37	Cyclic interrupt 7 (default interval: 20 ms)	14	
OB38	Cyclic interrupt 8 (default interval: 10 ms)	15	
OB40	Hardware interrupt 0	16	
OB41	Hardware interrupt 1	17	
OB42	Hardware interrupt 2	18	
OB43	Hardware interrupt 3	19	Hardware interrupts
OB44	Hardware interrupt 4	20	
OB45	Hardware interrupt 5	21	
OB46	Hardware interrupt 6	22	
OB47	Hardware interrupt 7	23	
OB55	Status interrupt	2	
OB56	Update interrupt	2	DPV1 interrupts
OB57	Manufacturer specific interrupt	2	

OB60	SFC35 "MP_ALM" call	25	**Multicomputing interrupt**
OB61	Synchronous Cycle Interrupt 1	25	
OB62	Synchronous Cycle Interrupt 2	25	**Synchronous Cycle Interrupt**
OB63	Synchronous Cycle Interrupt 3	25	
OB64	Synchronous Cycle Interrupt 4	25	
OB65	Technology synchronization interrupt	25	**Technology synchronization interrupt**
OB70	I/O redundancy error (only in H CPUs)	25	
OB72	CPU redundancy error (only in H CPUs)	28	**Redundancy error interrupts**
OB73	Communication redundancy error OB (only in H CPUs)	25	
OB80	Time error	25-28	
OB81	Power supply fault	25-28	
OB82	Diagnostic interrupt	25-28	**Asynchronous error interrupts**
OB83	Insert/remove module interrupt	25-28	
OB84	CPU hardware fault	25-28	
OB85	Program error	25-28	
OB86	Failure of an expansion rack, DP master system or station for distributed I/Os	25-28	
OB87	Communication error	25-28	
OB88	Processing interrupt	28	
OB90	Warm or cold restart or delete a block being executed in OB90 or load an OB90 on the	29	**Background cycle**
OB100	Warm restart	27	
OB101	Hot restart	27	**Startup**
OB102	Cold restart	27	
OB121	Programming error	Priority of the OB	**Synchronous error interrupts**
OB122	I/O access error		

نکته در مورد عملکرد OB90

در بین OBها درجه‌ی اولویت ۱ به OB1 تعلق دارد و به‌نظر می‌رسد که کم‌ترین درجه‌ی اولویت را دارد ولی در واقع OB90 دارای کم‌ترین اولویت است؛ اگرچه به‌ظاهر عدد ۲۹ به درجه‌ی اولویت OB90 اختصاص داده شده است. این OB از خانواده وقفه‌ها نیست و به آن اصطلاحاً Background OB گفته می‌شود. علت این نام‌گذاری آن است که در برخی موارد ممکن است اجرا شود. در بحث سیکل اسکن CPU که در کتاب سطح مقدماتی تشریح شد با پارمتری به‌عنوان مینیمم زمان سیکل اسکن آشنا شدیم.

اگر سیکل اسکن در عمل از حد مینیمم تعیین شده کمتر باشد، CPU به اندازه زمان باقی مانده صبر می‌کند سپس سیکل بعدی را شروع می‌نماید. اگر OB90 وجود داشته باشد در زمان باقیمانده فوق اجرا می‌گردد و اگر OB90 وجود نداشته باشد، CPU پس از سپری شدن زمان باقیمانده، سیکل بعدی را شروع می‌کند.

۷-۵ بررسی تأثیر وقفه روی زمان سیکل اسکن CPU

همانطور که قبلاً بحث شد، سیکل اسکن CPU به‌صورت نرمال شامل سه مرحله شکل ۷-۵ است.

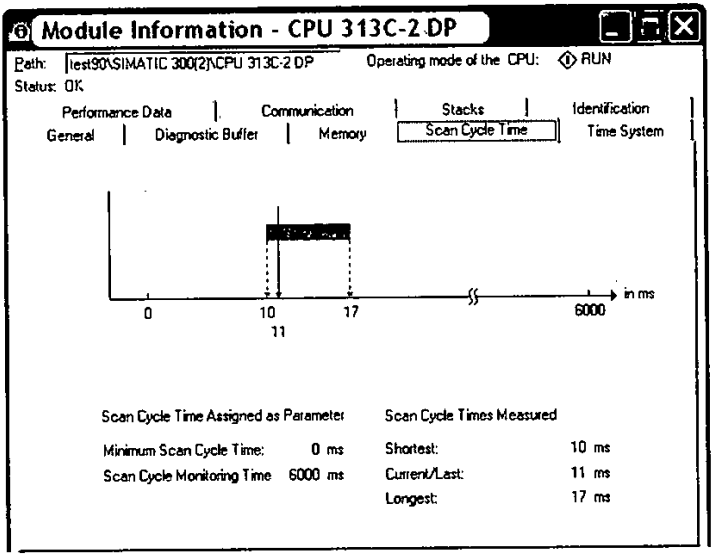
Scan Cycle 1			Scan Cycle 2		
PIQ Updating	PII Updating	OB1 Processing	PIQ Updating	PII Updating	OB1 Processing

شکل ۷-۵ مراحل سیکل اسکن CPU

اگر فرض کنیم هیچ وقفه‌ای اتفاق نیفتد باز هم ممکن است زمان سیکل اسکن متغیر باشد. تغییر زمان سیکل بستگی به برنامه ای دارد که در OB1 نوشته شده است، بویژه وقتی از دستورات شرطی[1] استفاده شود .به‌عنوان مثال وقتی که در یک سیکل از یک فانکشن enable می‌گردد یا شرایط اجرای یک لوپ فراهم می‌شود زمان این سیکل نسبت به سیکل قبلی بیشتر خواهد بود.

به همین علت است که در حالت Online برای سیکل اسکن سه وضعیت گزارش می‌شود:

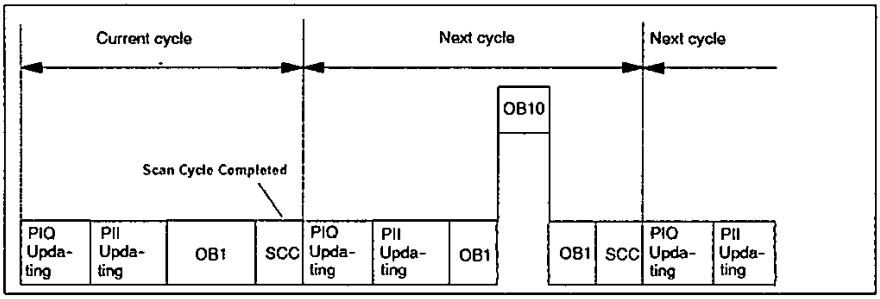
- کوتاه‌ترین سیکل (Shortest)
- طولانی‌ترین سیکل (Longest)
- سیکل فعلی (Current)



شکل ۷-۶ سیکل اسکن Online

اگر در حین اجرای یک سیکل وقفه‌ای اتفاق بیفتد نیز بدیهی است که زمان آن سیکل نسبت به سیکل قبلی بیشتر خواهد بود. شکل ۷-۷ حالتی را نشان می‌دهد که اجرای یک سیکل به‌دلیل اجرای وقفه OB10 بیشتر طول کشیده است.

1. Conditional

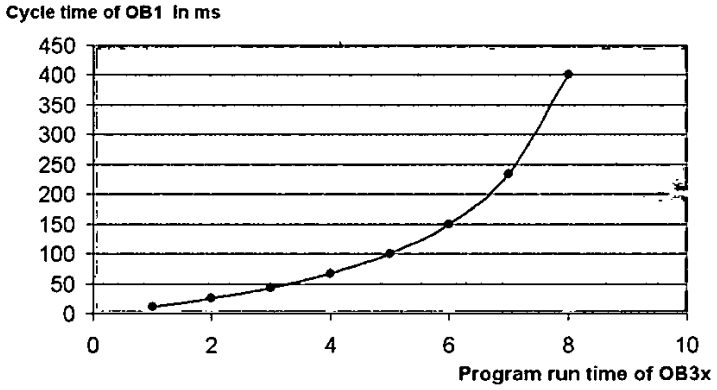


شکل ۷-۷ افزایش سیکل اسکن به‌دلیل وقفه

تأثیر وقفه‌ها روی سیکل اسکن را می‌توان به دو دسته تقسیم کرد:

۱- وقفه‌هایی که به‌طور مرتب تکرار می‌شوند

وقفه Cyclic از این دسته است، همانطور که در جدول ۷-۲ دیده می‌شود، هر کدام از OBهای این خانواده دارای یک زمان هستند و تکرار وقفه بر مبنای این زمان می‌باشد. بدیهی است که اگر زمان فوق خیلی کم باشد ممکن است در سیکل اسکن وقفه فوق بیش از یک بار تکرار شود. اگر برنامه موجود در وقفه‌های فوق زیاد باشد زمان اجرا طولانی‌تر و سیکل اسکن بیشتر خواهد بود. شکل ۷-۸ نمونه‌ای از این حالت را نشان می‌دهد. محور افقی زمان اجرای OBهای وقفه سیکلی و محور افقی زمان کلی سیکل برنامه است.



شکل ۷-۸ تأثیر OB3x روی سیکل اسکن

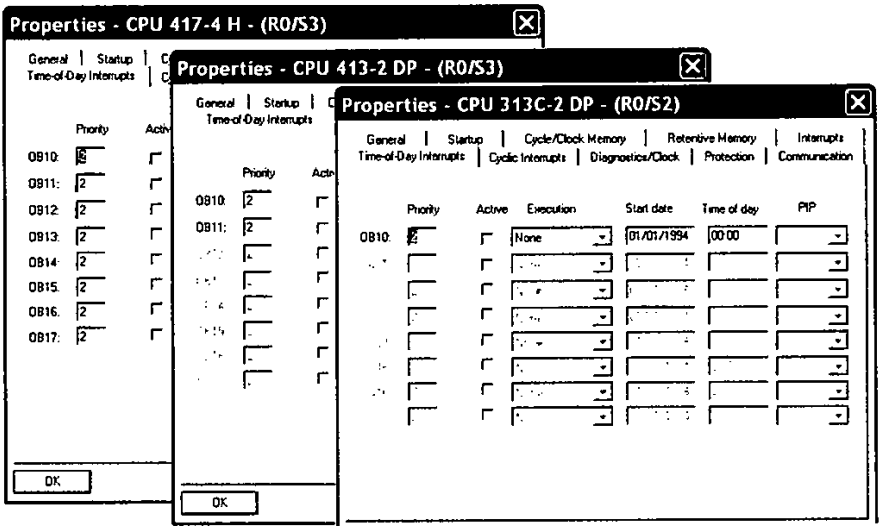
وقفه‌های Time of Day نیز اگر به‌صورت پریودیک تنظیم شوند روی سیکل اسکن به‌طور مداوم تأثیر می‌گذارند. ولی زمان تکرار آنها با فاصله طولانی و حداقل یک دقیقه یک بار است.

۲- وقفه‌هایی که فقط در برخی شرایط تأثیر می‌گذارند

بسیاری از وقفه‌ها فقط در شرایط خاصی فعال می‌شوند و تأثیر آنها روی سیکل اسکن به‌صورت موردی و موقت است. وقفه‌های سخت افزاری و وقفه‌های مبتنی بر Error از این دسته هستند. بدیهی است که این وقفه‌ها فقط برخی از سیکل‌ها را تحت تأثیر قرار می‌دهند.

۷-۶ وقفه‌های OB1x (Time-of-Day Interrupt)

وقفه‌ها تاریخ زمانی که به اختصار TOD Interrupt خوانده می‌شوند وقفه‌هایی هستند که در تاریخ و زمان مشخصی فعال می‌شوند. این وقفه‌ها با OB10 تا OB17 برنامه‌نویسی می‌شوند به همین علت آنها را وقفه‌های OB1x می‌نامند. هر CPU ممکن است یک یا چند OB وقفه TOD را پشتیبانی نماید. این OBها را می‌توان در پنجره online شکل ۷ - ۳ یا در پارامترهای CPU به‌صورت Offline مانند شکل ... مشاهده نمود.



شکل ۷-۹ وقفه OB1x در چند نوع CPU

برای اینکه بلاک‌های وقفه TOD اجرا شوند، باید زمان و تاریخ اجرای آنها و همچنین نحوه اجرای آنها را تنظیم نمود. این تنظیم را می‌توان در محیط HW Config یا با استفاده از برخی SFCهای خاص انجام داد.

تنظیم وقفه TOD

برای استفاده صحیح از این وقفه‌ها باید ابتدا آنها را **تنظیم** و سپس **فعال** نمود. به سه روش می‌توان این عمل را انجام داد.

۱- **تنظیم** در محیط HW Config، **فعال‌سازی** در محیط HW Config

۲- **تنظیم** در محیط HW Config، **فعال‌سازی** توسط فراخوانی SFC30

۳- **تنظیم** توسط فراخوانی SFC28، **فعال‌سازی** توسط فراخوانی SFC30

تذکر: با فراخوانی SFC29 می‌توان وقفه TOD را غیرفعال نمود.

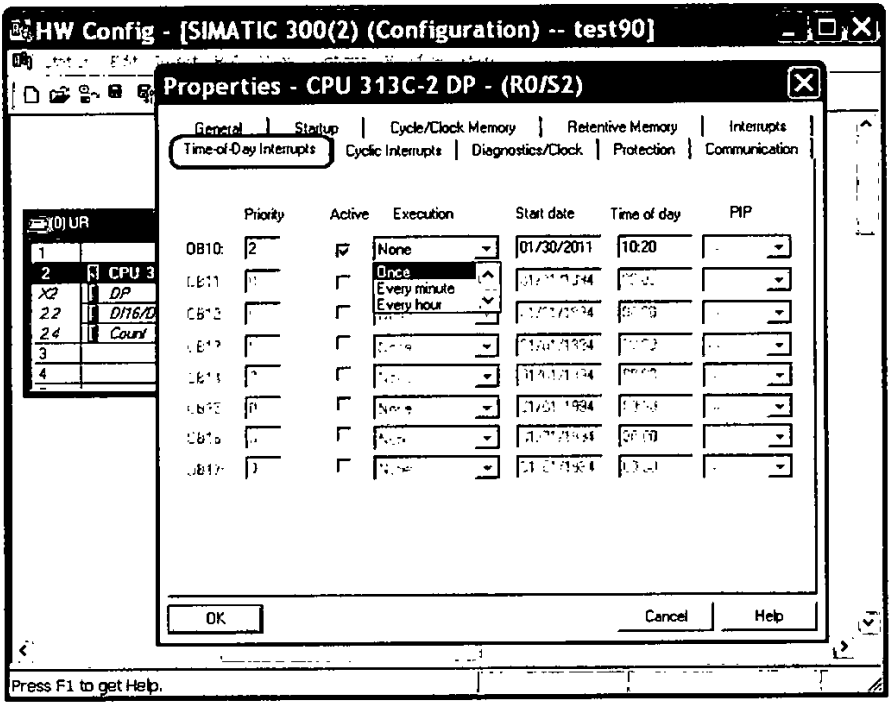
در این کتاب فقط روش اول مورد بحث قرار می‌گیرد. روش‌های دوم و سوم در کتاب سطح تکمیلی آورده شده است.

فصل ۷

تنظیم وقفه TOD در محیط HW Config

برای تنظیم وقفه TOD در محیط HW Config لازم است مراحل زیر طی شود.

۱- مطابق شکل ۷-۱۰ روی CPU موجود در رک دوبارکلیک نمایید.



شکل ۷-۱۰ تنظیمات OB1x

۲- در پنجره باز شده سربرگ Time of day interrupt را انتخاب نمایید.

گزینه‌های موجود در این سربرگ همانطور که در شکل دیده می‌شود عبارتند از:

Priority: در این فیلد کلاس اولویت OB مربوطه نمایش داده می‌شود.

Active: با علامت‌زدن این گزینه وقفه برای OB مورد نظر فعال می‌شود.

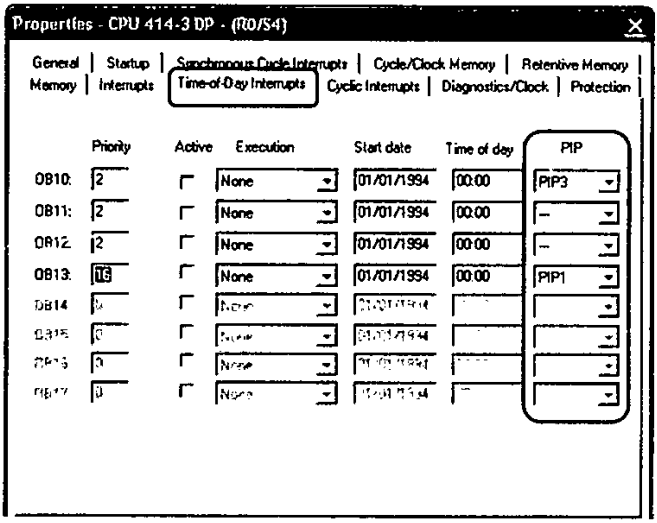
Execution: در این فیلد می‌توان نوع اجرای وقفه را مشخص نمود. نوع اجرا می‌تواند یکی از موارد زیر انتخاب شود که در صورت فرا رسیدن تاریخ و زمان تعیین شده:

- **Once:** وقفه فقط یک بار فعال می‌گردد.
- **Every minute:** وقفه هر دقیقه یک بار فراخوان می‌شود.
- **Hourly:** وقفه هر ساعت یک بار فعال می‌گردد.
- **Daily:** وقفه هر روز یک بار فعال می‌گردد.
- **Weekly:** وقفه هر هفته یک بار فعال می‌گردد.
- **Monthly:** وقفه هر ماه یک بار فعال می‌گردد.
- **At the end of each month:** وقفه در روز پایانی هر ماه فعال می‌گردد.
- **Every year:** وقفه هر سال یک بار فعال می‌شود.

Start date: در این فیلد می‌توان تاریخ شروع اجرای وقفه را به‌صورت *سال/ روز/ ماه* وارد نمود. به‌عنوان مثال تاریخ 01/30/2012 اشاره به روز سی‌ام از ماه ژانویه سال ۲۰۱۲ دارد.

Time of day: در این فیلد می‌توان زمان اجرای وقفه را بر حسب ساعت و دقیقه تعیین نمود. به‌عنوان مثال زمان وقفه‌ای که قرار است در ساعت هشت و بیست دقیقه صبح اجرا شود به‌صورت 08:20 نوشته می‌شود. برای ساعت هشت و بیست دقیقه شب این زمان به‌صورت 20:20 در این فیلد وارد می‌شود. امکان تنظیم ثانیه وجود ندارد.

PIP: در CPUهایی که از پارتیشن‌بندی ناحیه Process Image پشتیبانی می‌کنند، می‌توان در این فیلد ناحیه‌ای از PII را انتخاب نمود تا در هنگام فراخوانی این وقفه Update شود.



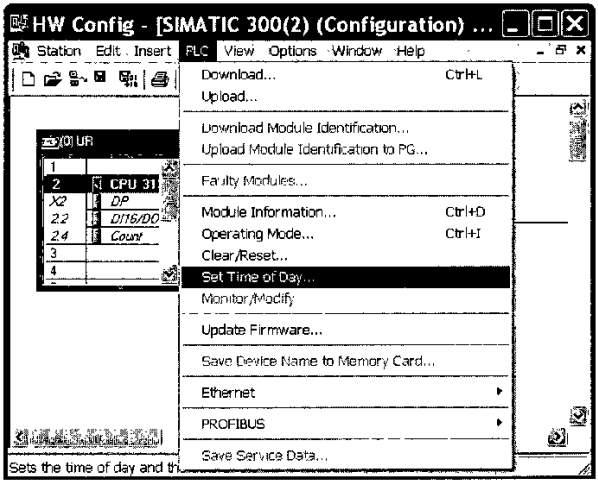
شکل ۷-۱۱ انتخاب ناحیه‌ای از PII جهت Update

۳- پس از اینکه تنظیمات مورد نظر انجام شد، باید تغییرات را Save & Compile نموده و به PLC دانلود نمود.

نکات مهم و قابل توجه در مورد وقفه TOD

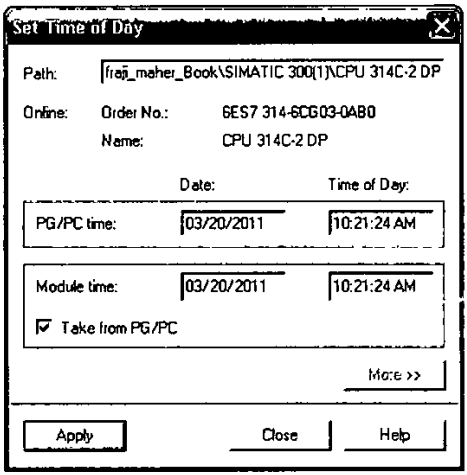
- **نکته ۱:** همانطور که در تنظیمات Execution مشخص است، با انتخاب هر کدام از گزینه‌های فوق (به‌غیر از Once) اجرای بلاک به‌صورت سیکلی خواهد بود.
- **نکته ۲:** اگر تاریخ را روی سی و یکم ماه تنظیم کنیم و دوره اجرا را Every month بدهیم، در اینصورت وقفه فوق برای ماه‌هایی که کمتر از ۳۱ روز هستند اجرا می‌شود. باید توجه داشت که در ماه‌های میلادی ماه فوریه ۲۸ روز و برخی ماه‌های دیگر ۳۰ روزه هستند. اگر نیاز به اعمال وقفه در آخر هر ماه داریم، در اینصورت بهتر است گزینه End of month را برای Execution انتخاب نماییم.
- **نکته ۳:** از آنجا که پس از فعال شدن این وقفه CPU مرتباً تاریخ و زمان خود را با تاریخ و زمان تنظیم شده برای OB فوق چک می‌کند، قبل از هر چیز باید تاریخ و زمان CPU را چک کرد که صحیح باشد. برای این کار دو روش وجود دارد:

روش اول: در حالی که کامپیوتر به PLC متصل است، در Hwconfig روی CPU در اسلات مربوطه کلیک کرده، سپس مانند شکل۷-۱۲ از مسیر PLC > Set time of day استفاده کنیم.



شکل ۷-۱۲ منوی مربوط به تاریخ و زمان CPU

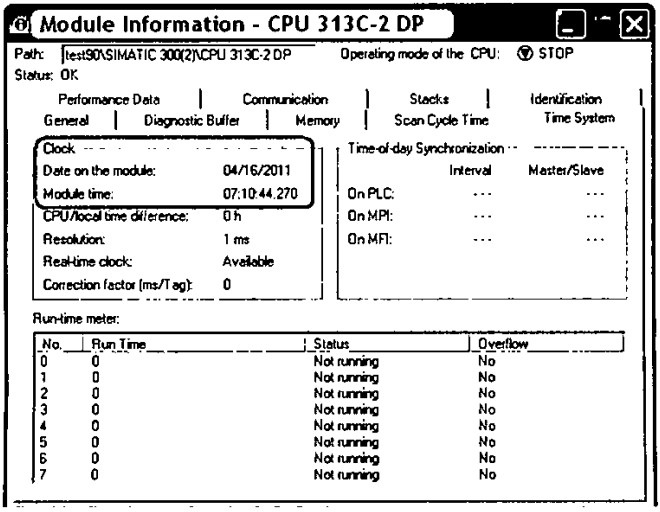
گزینه Set Time of Day علاوه بر محیط Hwconfig در برنامه اصلی Step7 و در زیر برنامه‌های LAD/STL/FBD و در Netpro نیز قابل دسترس است. پس از کلیک روی آن، مطابق شکل ۷-۱۳، می‌توانیم تاریخ و زمان کامپیوتر را در جلوی PG/PC time و تاریخ و زمان PLC را در جلوی Module time مشاهده کنیم.



شکل ۷-۱۳ نحوه تنظیم تاریخ و زمان CPU

همانطور که مشاهده می‌شود، به‌طور پیش‌فرض گزینه Take From PG/PC فعال است. در این حالت اگر روی Apply کلیک کنیم، تاریخ و زمان کامپیوتر به PLC منتقل می‌گردد. اگر بخواهیم تاریخ و زمان PLC را به‌صورت دستی تنظیم کنیم، باید چک باکس فوق را غیرفعال نموده و تاریخ و زمان دلخواه را وارد کنیم.

روش دوم: در پنجره Module Info مربوط به CPU در سربرگ Time System می‌توان تاریخ و زمان CPU را به‌صورت Online مانند شکل ۷-۱۴ مشاهده کرد.

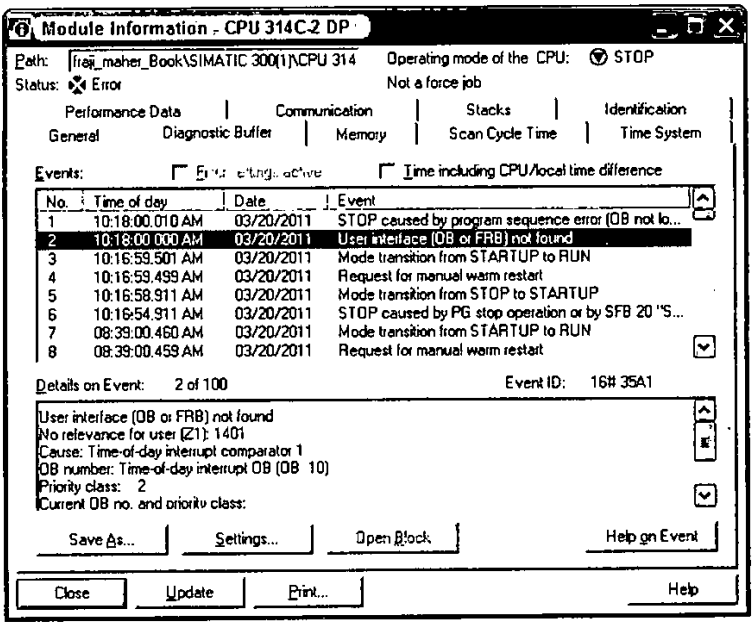


شکل ۷-۱۴ تاریخ و زمان CPU در پنجره Module Info

- **نکته ۴:** در نسخه‌های قدیمی Step7 مانند نسخه V5.2 و پایین‌تر فرمت تاریخ و زمان در پنجره فوق با فرمت تاریخ و زمان جلوی OB1x متفاوت است و باید جای ماه و روز را تغییر داد، ولی در نسخه‌های جدید این مشکل وجود ندارد و کاربر می‌تواند تاریخ و زمان را به همان فرمتی که در پنجره Online مشاهده می‌کند وارد نماید.
- **نکته ۵:** در تنظیم تاریخ و زمان باید توجه داشت که تاریخ و زمان تنظیم شده بایستی نسبت به تاریخ و زمان فعلی جلوتر باشد و CPU باید یک بار به این تاریخ و زمان برسد تا اعمال وقفه کرده و OB را اجرا کند. اگر به‌عنوان مثال تاریخ یک ماه قبل از تاریخ CPU باشد، هیچ‌گاه این وقفه فعال نخواهد شد هر چند دوره اجرای آن ماهانه یا کمتر انتخاب شده باشد.
- **نکته ۶:** در صورتی‌که وقفه OB1x را به روشی که گفته شد تنظیم و فعال کرده و دانلود کنیم بایستی OB مربوطه را نیز به PLC دانلود کنیم؛ در غیر اینصورت یعنی اگر OB مربوطه ایجاد و دانلود نشده باشد، در زمان فراخوانی وقفه CPU متوقف می‌گردد.

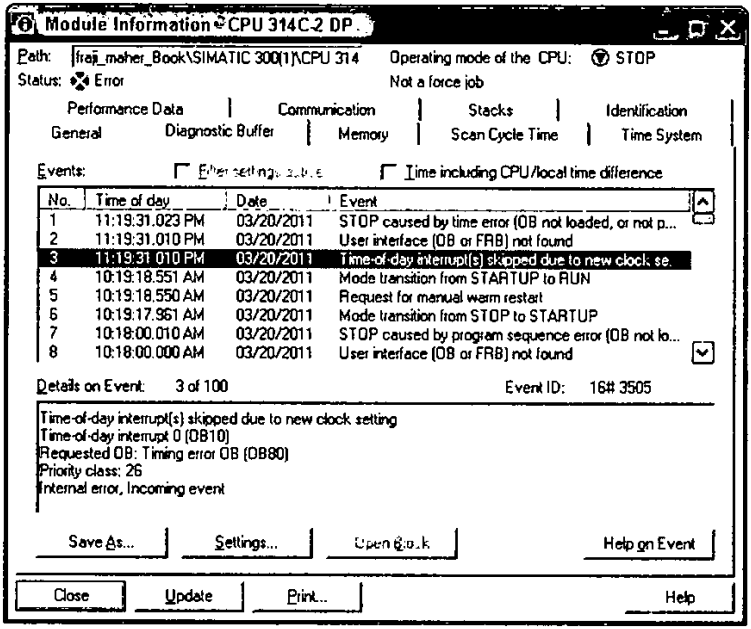
فصل ۷

در این حالت روی CPU300 چراغ SF و روی CPU400 چراغ INTF روشن می‌گردد. اگر محتویات بافر CPU را مشاهده کنیم، مانند شکل۷-۱۵ پیغام User Interface (OB or FRB) not Found را می‌بینیم. این حالت را می‌توان با استفاده از OB85 ماسک نمود. یعنی وضعیتی که اگر OB1x موجود نبود فقط چراغ فالت روشن شود ولی CPU متوقف نگردد. توضیحات بیشتر در این مورد در ادامه بحث بیان می‌گردد.



شکل ۷-۱۵ پیغام بافر مبنی بر عدم وجود OB10

- **نکته ۷:** وقتی وقفه TOD تنظیم و فعال شده ولی هنوز زمان اجرای آن نرسیده باشد، اگر کاربر بدون توجه به آن به‌طور دستی تاریخ و زمان CPU را جلو بکشد به‌صورتی که از روی تاریخ و زمان وقفه پرش کند، منجر به روشن شدن چراغ فالت SF یا INTF شده و CPU نیز متوقف می‌گردد. در این شرایط پیامی مانند شکل ۷-۱۶ در بافر CPU خواهیم دید. این اشکال را می‌توان توسط OB80 ماسک نمود.



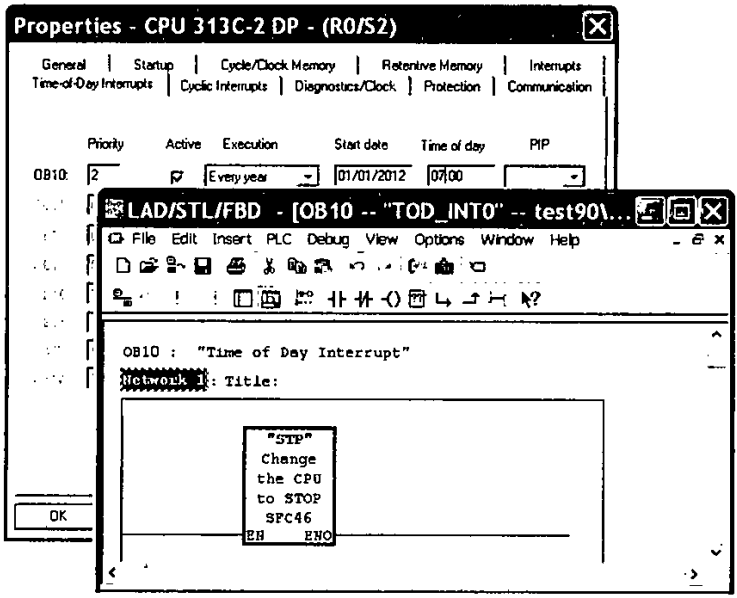
شکل ۷-۱۶ پیغام خطای پرش از روی وقفه TOD

- **نکته ۸:** برای جلوگیری از سوء استفاده‌های احتمالی از این وقفه، توصیه می‌شود در هنگام تحویل‌گیری پروژه وضعیت فعال بودن آن توسط Hwconfig یا توسط فانکشن‌های سیستمی بررسی شود و در صورت فعال بودن وقفه فوق، بررسی شود که آیا OB1x مربوطه در CPU موجود است و اگر موجود است، در برنامه آن چه دستوراتی نوشته شده است. مثال ۷-۱ یک نمونه از کاربردهای ناصحیح این وقفه را نشان می‌دهد.

مثال ۷-۱ توقف PLC در ابتدای هر سال میلادی

در برنامه زیر وقفه OB10 برای اول ژانویه 2012 تنظیم شده است و به‌طور سالانه در همین روز تکرار می‌شود.

در OB10 فانکشن SFC46 که منجر به توقف CPU می‌گردد صدا زده شده است. این برنامه موجب توقف CPU در ساعت ۷ صبح روز اول ژانویه در هر سال می‌شود. اگر چه پس از توقف می‌توان به‌صورت CPU را به‌صورت دستی RUN کرد، ولی یک لحظه توقف می‌تواند خسارات زیادی را برای فرآیند به‌دنبال داشته باشد.



شکل ۷-۱۷ برنامه مثال ۷-۱

- **نکته ۹:** اگر وقفه TOD به‌صورت Once فعال شده و به‌عنوان مثال در برنامه OB1x مربوطه دستور روشن شدن یک خروجی نوشته شده باشد، خروجی مزبور روشن می‌گردد. در این شرایط اگر تغذیه CPU قطع شود و خروجی Off شود با وصل تغذیه، خروجی روشن خواهد شد حتی اگر شرایط وقفه مربوط به قبل بوده و زمان آن سپری شده باشد. اگر وقفه TOD به‌صورت سیکلی مانند Every miute فعال شده و اتفاق فوق بیفتد، خروجی خاموش می‌ماند.
- **نکته ۱۰:** در مثال ذکر شده در نکته ۹، فرض کنید چند لحظه قبل از اینکه وقفه فعال شود، تغذیه قطع شود (یعنی هنوز خروجی مورد نظر روشن نشده و تغذیه قطع شود). در این شرایط اگر سیستم همچنان بدون برق بماند و لحظه‌ای که تغذیه وصل می‌شود زمان وقفه سپری شده باشد، باز می‌بینیم که وقفه عمل کرده و خروجی روشن می‌شود.
- **نکته ۱۱:** اگر برنامه نوشته شده در OB1x را به‌صورت Online مانیتور کنیم، قسمت RUN که به رنگ سبز در پایین پنجره ظاهر می‌شود ثابت است و حرکت نمی‌کند در حالی که در OB1 این قسمت به‌طور مداوم در حال حرکت است. این مطلب بیانگر این است که OB1x در آن لحظه اجرا نمی‌شود. بدیهی است که بسته به تنظیم انجام شده OB1x فقط در یک لحظه یا لحظات خاصی اجرا خواهد شد.

کاربردهای وقفه‌های TOD

به این وقفه نباید با دیدگاه منفی نگاه کرد. این وقفه ابزار قدرتمندی را برای برنامه‌نویسان PLC فراهم می‌کند که بدون آن نوشتن برخی برنامه‌ها بسیار پیچیده و دشوار خواهد بود. اگر فرض کنیم که وقفه‌های OB1x در PLC وجود نداشت، برای اینکه بتوان کار خاصی را در تاریخ و زمان مشخصی انجام داد نمی‌توانستیم از تایمر و دستورات دیگر استفاده

کنیم. تنها راه این بود که به‌طریقی بتوانیم در هر سیکل اسکن تاریخ و زمان CPU را بخوانیم و با تاریخ و زمان مورد نظر مقایسه کرده و در صورت یکسان بودن فرامین لازم را صادر کنیم. این روش با معایب زیر همراه است:

- نیاز به استفاده از SFC1 برای خواندن تاریخ و زمان CPU دارد.
- نیاز به جدا سازی تاریخ و زمان برای اینکه بتوان آنها را مقایسه نمود وجود دارد.
- برنامه‌های فوق در هر سیکل اسکن پردازش می‌شوند و همیشه CPU بایستی بار اضافی آنها را به‌دوش بکشد.

با توجه به موارد فوق می‌بینیم که وجود وقفه TOD در CPU تا چه حد از نظر کاربرد اهمیت دارد. از نمونه‌های کاربرد عملی می‌توان به موارد زیر اشاره نمود:

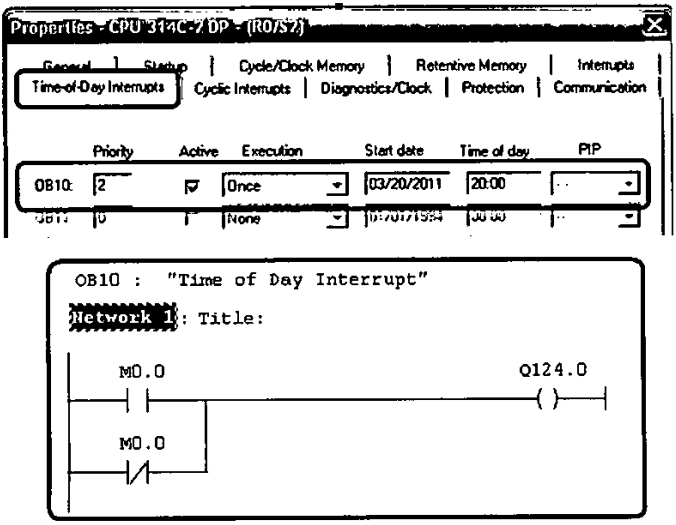
- تبادل دیتا بین دو یا چند PLC در زمان‌های مشخصی
- انتقال اطلاعات PLC به کامپیوتر برای گزارش‌گیری در پایان هر ماه یا پایان هر سال
- سنکرون‌سازی و تنظیم ساعت چند سیستم متصل به شبکه یک بار در هر روز بر اساس ساعت یکی از آنها که به‌عنوان Master تعیین می‌شود.
- گزارش‌گیری از شیفت‌های مختلف تولید
- و ....

ارائه چند مثال برای وقفه OB1x

مثال‌هایی که در ادامه برای فهم بهتر عملکرد OB1x ارائه می‌گردند با سیمولاتور PLCSIM قابل تست هستند.

مثال ۷-۲ روشن شدن یک موتور در تاریخ و زمان مشخص

برنامه زیر در تاریخ 03/20/2011 در ساعت ۸ عصر یک موتور به آدرس Q124.0 را روشن می‌کند.

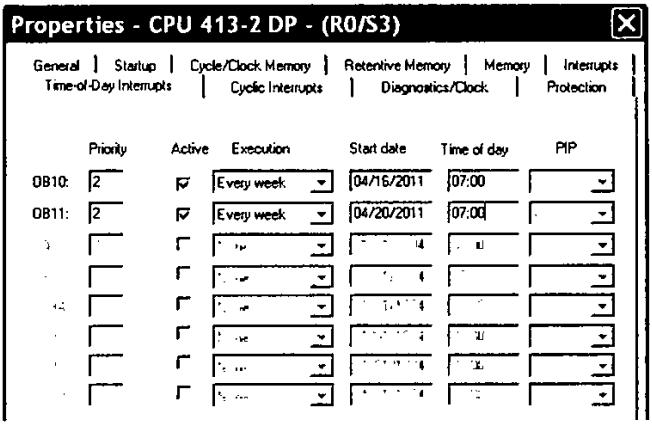


شکل ۷-۱۸ تنظیم و فعال‌سازی OB10 در مثال ۷-۲

مثال ۷-۳

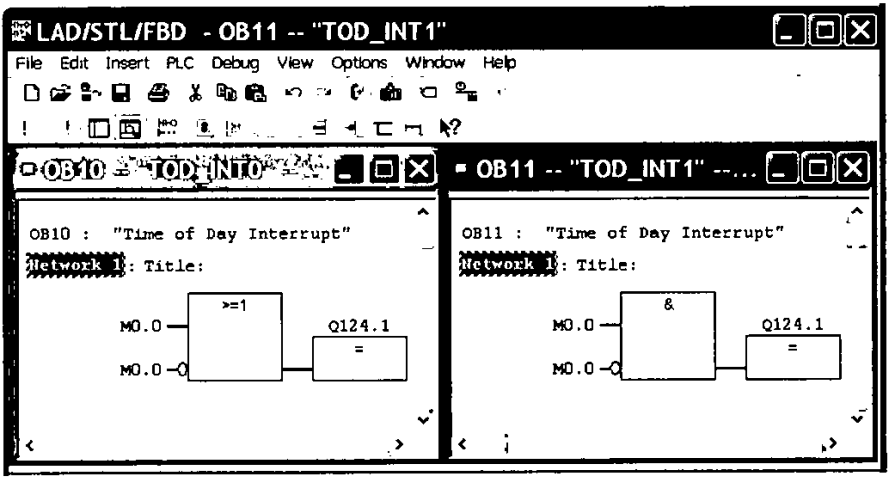
برنامه‌ای بنویسید که یک سیستم تهویه مطبوع (Q124.1) هر هفته شنبه ساعت ۷ صبح روشن شده و چهارشنبه ساعت ۷ صبح خاموش شود. PLC مورد استفاده از نوع S7-400 است و OB10 و OB11 را ساپورت می‌کند.

حل: کافیست ابتدا OB10 را برای شنبه آینده ساعت ۷ صبح و OB11 را برای چهارشنبه بعد از آن ساعت ۷ صبح مانند شکل ۷-۱۹ به‌صورت هفتگی تنظیم کنیم.



شکل ۷-۱۹ تنظیمات مثال ۷-۳

برنامه‌نویسی OB10 و OB11 به‌صورت زیر است. همانطور که دیده می‌شود این کار به‌سادگی انجام می‌گیرد در صورتی‌که بخواهیم این برنامه را با روش‌های معمولی بدون وقفه بنویسیم بسیار طولانی خواهد بود.



شکل ۷-۲۰ برنامه مثال ۷-۳

مثال ۷-۴

برنامه مثال ۷-۳ را برای S7-300 که فقط یک OB10 را ساپورت می‌کند بنویسید.

حل: ابتدا تنظیمات OB10 را به‌صورت زیر انجام می‌دهیم:

نوع اجرا: هر روز

تاریخ اجرا: اولین شنبه آینده

ساعت اجرا: ۷ صبح

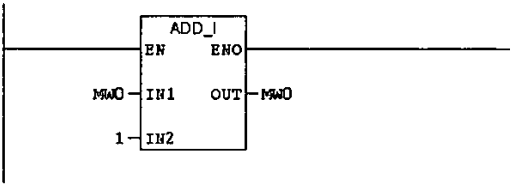
برنامه را به‌صورت زیر در OB10 می‌نویسیم. در این برنامه MW0 یک شمارنده است که هر روز یکی به آن اضافه می‌شود. پس روز شنبه مقدار آن 1 و روز چهارشنبه مقدار آن 5 خواهد بود با مقایسه MW0 با دو عدد 1 5 می‌توان دستورات روشن و خاموش کردن را صادر کرد. در این برنامه وقتی مقدار MW0 به عدد 7 رسید نشان دهنده روز جمعه است که در این شرایط MW0 به صفر بر می‌گردد تا برای هفته بعد نیز به‌درستی عمل کند.

OB10 : "Time of Day Interrupt"

Comment:

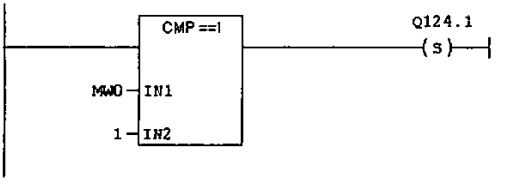
Network 1 : Title:

شمارش روزهای هفته



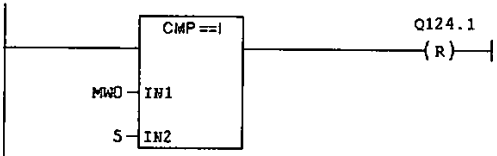
Network 2 : Title:

روز شنبه



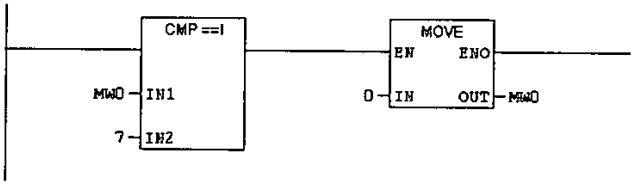
Network 3 : Title:

روز چهارشنبه



Network 4 : Title:

روز جمعه



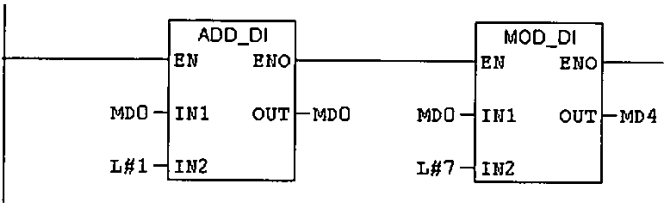
شکل ۷-۲۱ برنامه مثال ۷-۴

تمرین ۷-۱: مثال‌های ۷-۳ و ۷-۴ را به‌صورتی بازنویسی کنید که سیستم تهویه مطبوع هر هفته شنبه ۷ صبح روشن و چهارشنبه **۵ عصر** خاموش شود.

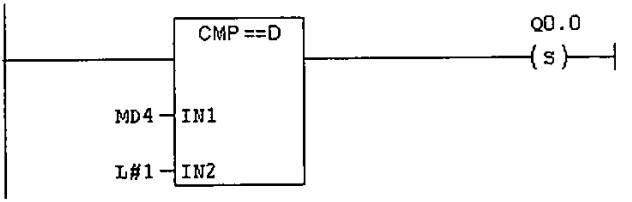
مثال ۷-۵

راه حل دیگر برای برنامه OB10 در مثال ۷-۴ به‌صورت زیر است آنالیز آن بر عهده خواننده می‌باشد.

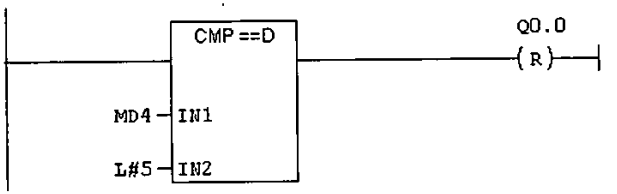
```
OB10 :  "Time of Day Interrupt"
Network 1 : Title:
```



Network 2 : Title:



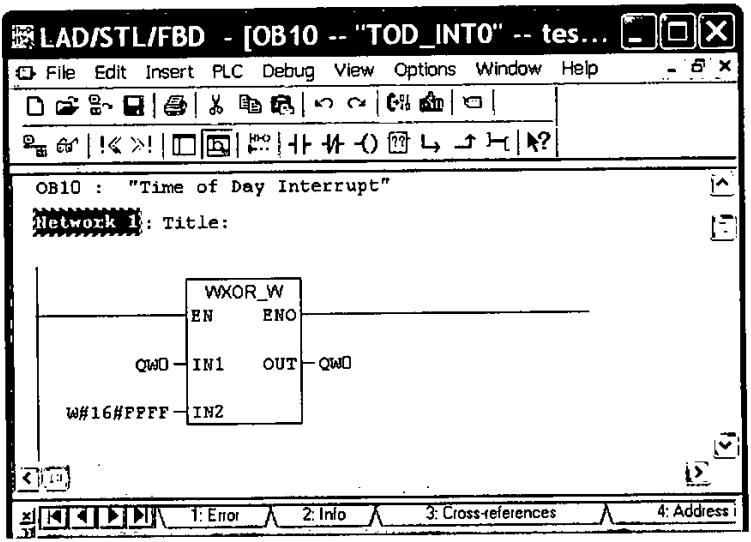
Network 3 : Title:



شکل ۷-۲۲ برنامه مثال ۷-۵

مثال ۷-۶

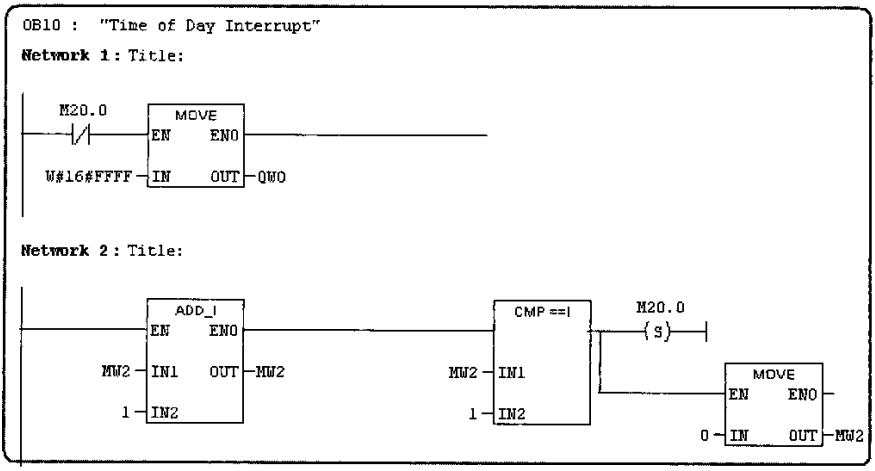
برنامه زیر تمام خروجی‌های QW0 را به‌طور مداوم یک دقیقه روشن و یک دقیقه خاموش می‌کند. تنظیم OB10 در Hwconfig به‌صورت Every minute است.

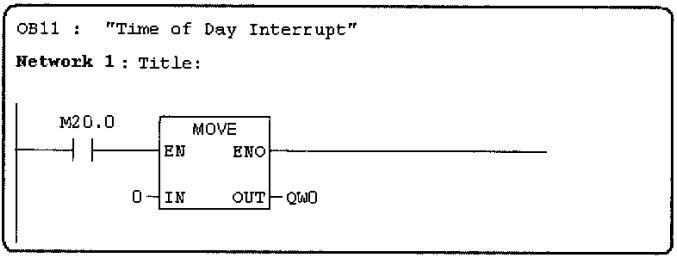


شکل ۷-۲۳ برنامه مثال ۷-۶

مثال ۷-۷

برنامه زیر هر دو دقیقه یک بار تمام خروجی‌های QW0 را خاموش و روشن می‌کند. این برنامه در S7-400 با OB10 و OB11 به‌صورت زیر نوشته شده است. آنالیز آن بر عهده خواننده است.





شکل ۷-۲۴ برنامه مثال ۷-۷

مثال ۷-۸ تفکیک تولید شیفت‌های مختلف

فرآیندی به‌صورت سه شیفت زیر کار می‌کند:

شیفت ۱: از ساعت ۷ الی ۱۵

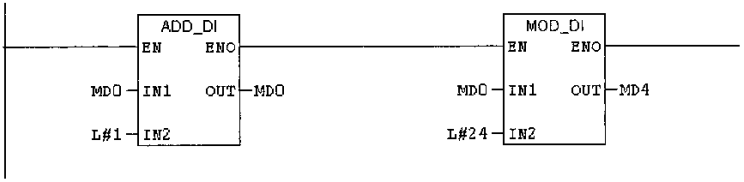
شیفت ۲: از ساعت ۱۵ الی ۲۳

شیفت ۳: از ساعت ۲۳ الی ۷ صبح روز بعد

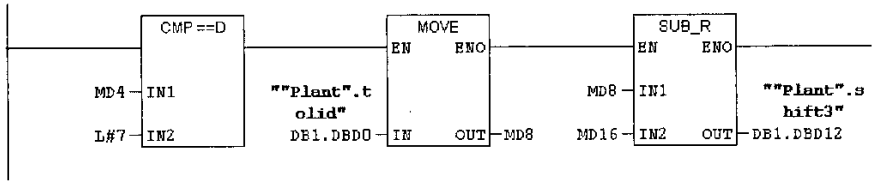
توسط S7-300 تولید به‌طور مداوم در DB1 ذخیره می‌شود. برنامه‌ای بنویسید که تولید هر شیفت را به‌صورت جداگانه محاسبه کرده و در DB1 ذخیره نماید.

حل: اگر PLC از نوع S7-400 بود و دارای سه OB1x متفاوت بود هر OB را برای یک شیفت اختصاص داده و برنامه فوق به‌سادگی قابل نوشتن بود ولی در اینجا PLC از نوع S7-300 است و فقط OB10 را ساپورت می‌کند. به همین جهت OB10 را به‌صورت هر ساعت یک بار تنظیم کرده و شروع آن را از ساعت 01 بامداد قرار می‌دهیم.
برنامه به‌صورت زیر است. آنالیز بر عهده خواننده می‌باشد.

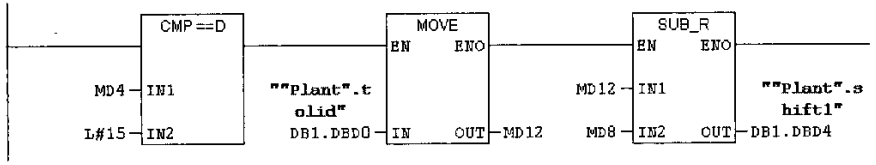
```
OB10 :  "Time of Day Interrupt"
Network 1 : Title:
```



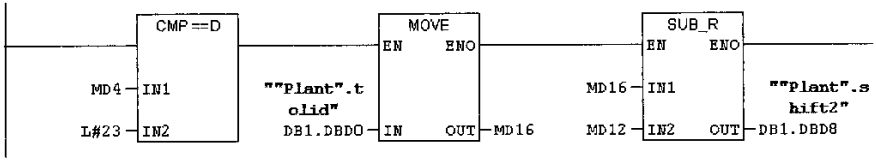

```
Network 2 : Title:
```




```
Network 3 : Title:
```




```
Network 4 : Title:
```



شکل ۷-۲۵ برنامه مثال ۷-۸

برنامه فوق در S7-300 که دیتابلاک‌ها روی کارت فلش ذخیره می‌شوند مشکلی ندارد ولی در S7-400 اگر دیتابلاک‌ها روی RAM ذخیره شده و CPU در بین کار متوقف و دوباره RUN شود، نتایج غلط خواهد بود. برای رفع مشکل باید از باتری پشتیبان استفاده شود یا دیتا بلاک‌ها به کارت فلش منتقل گردند.

متغیرهای محلی OBهای وقفه TOD

همانطور که قبلاً نیز اشاره شد، همه‌ی OBها دارای متغیرهای محلی می‌باشند که اطلاعاتی متناسب با وظیفه بلاک و تنظیمات آن در اختیار کاربر قرار می‌دهند. جدول ۷-۳ متغیرهای محلی وقفه‌های TOD را نشان می‌دهد.

جدول ۷-۳

Variable	Type	Description
OB10_EV_CLASS	BYTE	Event class and identifiers: B#16#11 = interrupt is active
OB10_STRT_INFO	BYTE	B#16#11: start request for OB10 (B#16#12: start request for OB11) : : (B#16#18: start request for OB17)
OB10_PRIORITY	BYTE	Assigned priority class; default 2
OB10_OB_NUMBR	BYTE	OB number (10 to 17)
OB10_RESERVED_1	BYTE	Reserved
OB10_RESERVED_2	BYTE	Reserved
OB10_PERIOD_EXE	WORD	The OB is executed at the specified intervals: W#16#0000: once W#16#0201: once every minute W#16#0401: once hourly W#16#1001: once daily W#16#1201: once weekly W#16#1401: once monthly W#16#1801: once yearly W#16#2001: end of month
OB10_RESERVED_3	INT	Reserved
OB10_RESERVED_4	INT	Reserved
OB10_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

مثال ۷-۹

برنامه‌ای بنویسید که بتوان نوع اجرای OB10 را به‌صورت زیر مشخص نمود:

نوع اجرای OB10	خروجی زیر روشن شود
Once	Q124.0
Every minute	Q124.1
Hourly	Q124.2
Daily	Q124.3
Weekly	Q124.4

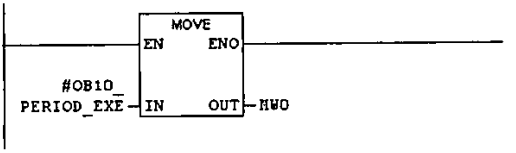
حل: همانطور که در جدول ۷-۴ مشخص است، متغیر محلی #OB10_PERIOD_EXE نشان‌دهنده نوع تنظیم اجرای بلاک است. جدول ۷-۴ کدهای نشان‌دهنده این پارامتر برای مثال فوق را نشان می‌دهد.

جدول ۷-۴

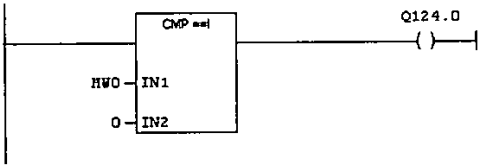
نوع اجرا	معادل عدد کد به فرم Integer	کد به فرم HEX
Once	0	W#16#0000
Every minute	513	W#16#0201
Hourly	1025	W#16#0401
Daily	4097	W#16#1001
Weekly	4609	W#16#1201

حال به‌سادگی می‌توان توسط دستورات مقایسه‌گر مقدار متغیر محلی #OB10_PERIOD_EXE را با مقادیر جدول فوق (ستون معادل عدد به فرم Integer) مقایسه نمود. برنامه مورد نظر به‌صورت زیر در OB10 نوشته می‌شود.

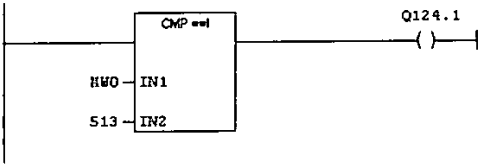
```
OB10 :  "Time of Day Interrupt"
Comment:

Network 1: Title:
Comment:
```

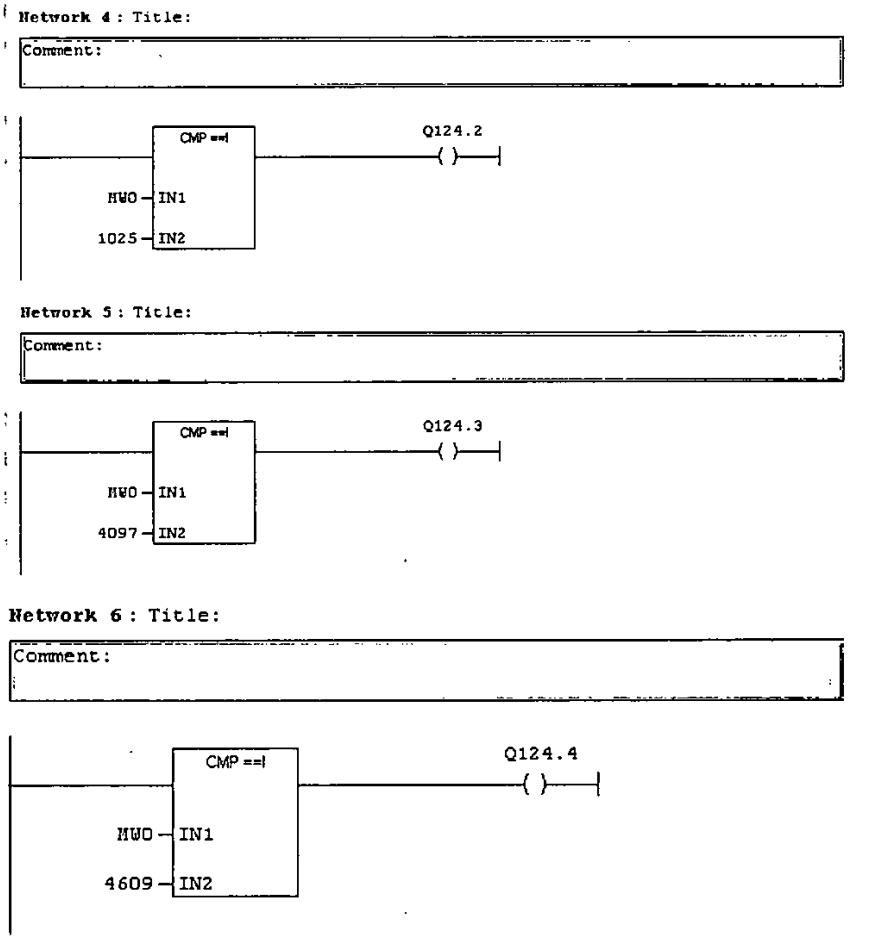



```
Network 2 : Title:
Comment:
```




```
Network 3 : Title:
Comment:
```

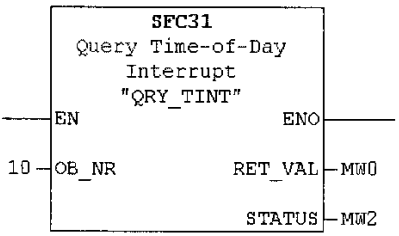




شکل ۷-۲۶ برنامه مثال ۷-۹

بررسی وضعیت وقفه TOD با استفاده از SFC 31

توسط این SFC می‌توان وضعیت یک OB وقفه TOD را مشاهده نمود. در شکل ۷-۲۷ فانکشن سیستمی SFC 31 نشان داده شده است.



شکل ۷-۲۷ نمایش SFC 31 در زبان LAD

همانطور که مشخص است، پارامترهای اصلی این بلاک عبارتند از:

OB_NR: در این پارامتر شماره OBای که می‌خواهیم وضعیت آن مشاهده شود را به فرم Integer وارد می‌کنیم.

RET_VAL: این پارامتر یک کد خطا را به صورت یک متغیر Integer بر می‌گرداند.

STATUS: این پارامتر کدی را به‌صورت Word نمایش می‌دهد که نشان‌دهنده وضعیت OB مورد نظر می‌باشد. از روی این کد می‌توان موارد مانند فعال یا غیرفعال بودن این وقفه، دانلود یا عدم دانلود وقفه به CPU و ... را مشخص نمود. جدول ۷-۵ کدهای قابل نمایش در پارامتر STATUS و مفهوم هر کدام از آنها را نشان می‌دهد. در ادامه بحث مثالی در این ارتباط ارائه می‌گردد.

جدول ۷-۵

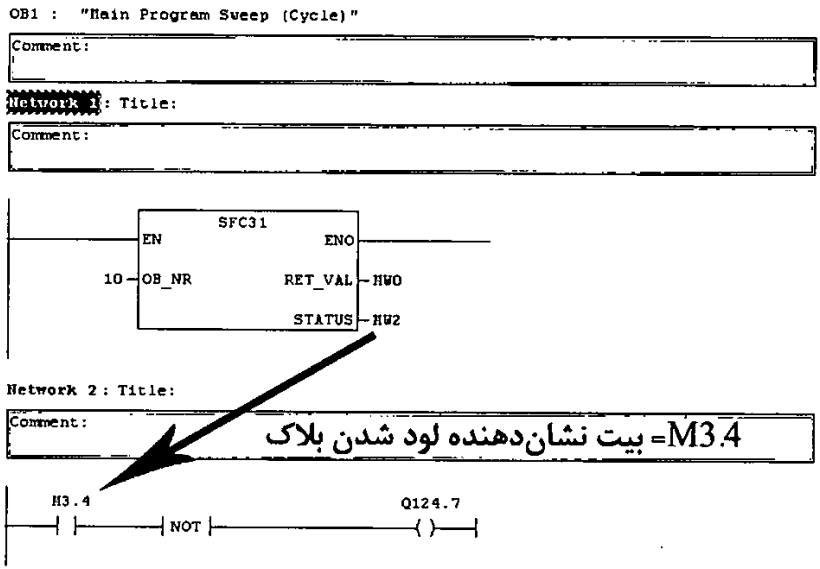
Bit	Value	Meaning
0	0	Time-of-day interrupt is enabled by operating system.
1	0	New time-of-day interrupts are accepted.
2	0	Time-of-day interrupt is not activated or has elapsed.
3	-	-
4	0	Time-of-day interrupt OB is not loaded.
5	0	The execution of the time-of-dayfunction. interrupt OB is not disabled by an active test
6	0	Base for the time-of-day interrupt is the basic time
	1	Base for the time-of-day interrupt is the local time

مثال ۷-۱۰

برنامه‌ای بنویسید که در صورت عدم دانلود OB10 به PLC لامپ Q124.7 روشن شود.

حل: همانطور که در جدول ۷-۵ مشخص است، بیت‌های شماره 0 الی 6 از پارامتر STATUS استفاده شده‌اند و هر کدام کدی را برمی‌گرداند. بیت شماره 4 نشان‌دهنده دانلود یا عدم دانلود بلاک مورد نظر است، بنابراین با استفاده از این بیت می‌توان تشخیص داد که آیا بلاک مورد نظر دانلود شده است یا خیر.

با توجه به نکات بیان شده برنامه به صورت زیر پیاده‌سازی می‌شود. تحلیل برنامه بر عهده خواننده قرار دارد.



شکل ۷-۲۸ برنامه مثال ۷-۱۰

اتفاقات تأثیرگذار بر اجرای وقفه TOD

برخی از اتفاقات و تنظیمات می‌تواند روی اجرای وقفه TOD تاثیرگذار باشد. این موارد در جدول۷-۶ نشان داده شده است.

جدول ۷-۶

شرایط پیش آمده	نتیجه
در برنامه کاربر SFC29 فراخوانی شود	اجرای وقفه TOD به حالت Cancel در می‌آید. در این حالت باید مجدداً تنظیم و فعال‌سازی وقفه را انجام داد.
فراخوانی وقفه TOD و عدم دانلود آن به PLC	در این حالت سیستم عامل OB85 را فراخوانی می‌کند که در صورت عدم وجود آن CPU به حالت Stop در می‌آید.
پرش از روی وقفه با تغییر زمان و تاریخ CPU	در این حالت سیستم عامل OB80 را فراخوانی می‌کند که در صورت عدم وجود آن CPU به حالت Stop درمی‌آید. در صورت وجود OB80، وقفه TOD صرفنظر از زمان و تاریخ تنظیم شده به‌صورت آنی فراخوانی و اجرا می‌گردد. اطلاعات مربوط به پرش، در OB80 می‌گیرد.
تنظیم ساعت CPU به زمان قبل از اجرای وقفه TOD	در S7-400 و CPU 318: در صورتی‌که قبلاً وقفه فعال شده باشد، اجرا نمی‌شود. در S7-300 : وقفه اجرا می‌شود.
انجام راه‌اندازی از نوع Cold یا Warm	همه وقفه‌های TOD که با استفاده از SFCها تنظیم شده‌اند به حالت تنظیم‌شده توسط نرم‌افزار S7 (تنظیم انجام‌شده در HW Config) برمی‌گردند. اگر یک وقفه برای اجرا در تاریخ خاصی تنظیم و فعال شده

	باشد ولی به‌دلیل خاموش بودن CPU وقفه اجرا نشده باشد، پس از راه‌اندازی Warm یا Cold وقفه اجرا می‌گردد.
فراخوانی همزمان دو OB وقفه TOD	OB80 فراخوانی می‌گردد، در صورت عدم وجود این OB ، CPU متوقف می‌گردد. در صورت وجود OB80، این OB و اولین OB وقفه همزمان اجرا شده و سپس OB دوم وقفه اجرا می‌شود.

بررسی همزمانی اجرای دو وقفه OB1x

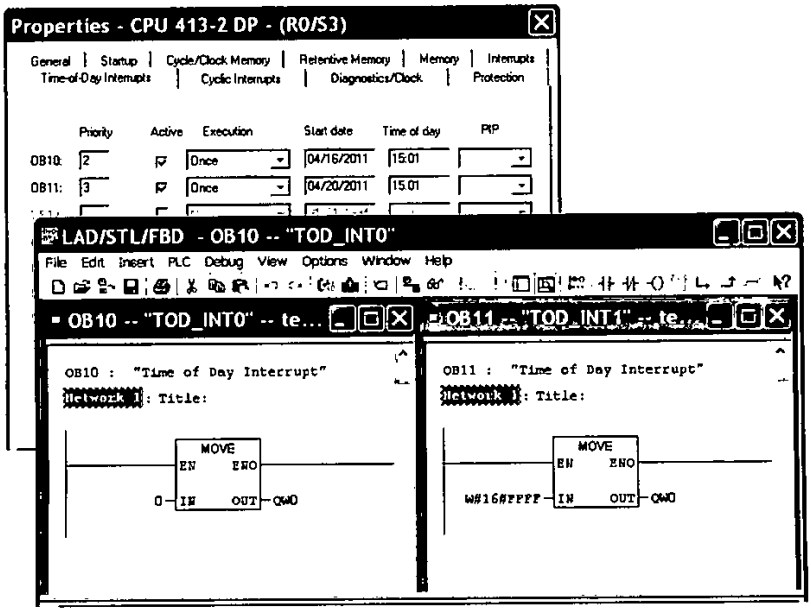
ممکن است دو وقفه OB1x برای یک تاریخ و زمان یکسان تنظیم شده باشند. در اینصورت پس از فرارسیدن زمان:

- اگر اولویت دو وقفه متفاوت باشد، ابتدا اولویت بالاتر سپس اولویت پایین‌تر اجرا می‌شود.
- اگر اولویت‌ها یکسان باشند، ابتدا OB با شماره بالاتر سپس OB با شماره پایین‌تر اجرا می‌شود. به‌عنوان مثال ابتدا OB11 سپس OB10 اجرا می‌شود.

موارد فوق با مثال زیر بررسی می‌شوند.

مثال ۷-۱۱

در برنامه زیر هر دو وقفه OB10 و OB11 که اولویت متفاوت دارند برای یک تاریخ و زمان فعال شده اند. در OB10 دستور خاموش شدن خروجی‌ها و در OB11 دستور روشن شدن همان خروجی‌ها نوشته شده است.



شکل ۷-۲۹ برنامه مثال ۷-۱۱

اگر بلاک‌ها را مانیتور کنیم می‌بینیم که رأس زمان مقرر هر دو اجرا می‌شوند ولی در عمل خروجی‌ها خاموش می‌گردند. علت این است که ابتدا OB11 اجرا شده و خروجی‌ها را روشن می‌کند ولی به‌دنبال آن OB10 که اولویت پایین‌تر دارد اجرا می‌شود و خروجی‌ها را صفر می‌نماید. پس فرمان‌های نهایی که به خروجی‌ها منتقل می‌شود فرمان صفر است.

فصل ۷

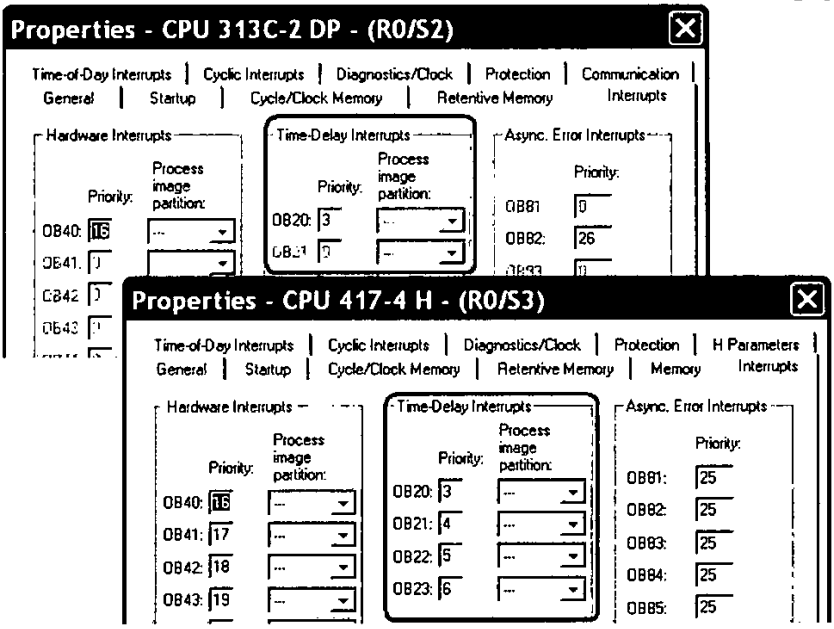
تذکر: اگر در مثال فوق اولویت هر دو OB یکسان باشد نیز همان نتیجه به‌دست خواهد آمد.

۷-۷ وقفه‌های OB2x (Time Delay Interrupt)

این وقفه‌ها برای اجرای تأخیری برنامه به‌کار می‌روند. این تأخیر زمانی می‌تواند با دقت زیاد و در حد حتی یک میلی‌ثانیه باشد. برنامه ای که لازم است با تأخیر اجرا شود در OB2x نوشته می‌شود ولی این کار به تنهایی کافی نیست. زمان تأخیر و شرایطی که باید بر اساس آن OB2x اجرا شود لازم است توسط فانکشن سیستمی تنظیم و فعال گردد.

لازم است توجه شود که به‌صورت عادی بدون OB2x نیز می‌توان در اجرای بخشی از برنامه با برنامه‌نویسی در FC یا FB تأخیر ایجاد کرد ولی تفاوتی که در مورد این وقفه وجود دارد این است که اولاً OB توسط سیستم عامل فراخوان می‌شود. بعلاوه زمان تأخیر می‌تواند خیلی کوتاه باشد که در کار به‌صورت معمولی ایجاد تأخیر با این دقت قابل دسترس نیست.

هر CPU تعداد OB خاصی از این وقفه را پشتیبانی می‌کند. شکل ۷-۳۰ سربرگ Interrupt را در چند نمونه CPU نشان می‌دهد.



شکل ۷-۳۰ وقفه OB2x در چند نمونه CPU

همانطور که در این شکل دیده می‌شود، درجه اولویت این وقفه‌ها نسبت به OB1x نیز بالاتر است و می‌تواند آنها را قطع کند.

نحوه استفاده از وقفه OB2x

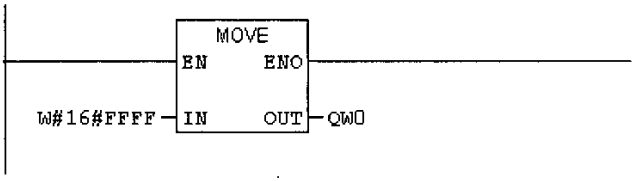
این وقفه در پنجره Hwconfig هیچ گزینه‌ای برای فعال‌سازی ندارد. فعال کردن و استفاده از آن با برنامه‌نویسی انجام می‌شود. برای این کار SFC32 مورد نیاز است. این SFC در برنامه‌های روتین مانند OB1 صدا زده شده و شماره OB2x مورد نظر و زمان تأخیر به SFC داده می‌شود. OB2x نیز به‌صورت مجزا برنامه‌نویسی می‌گردد. در این شرایط پس از فعال شدن SFC32 پس از گذشت زمان تأخیر تعیین شده، برنامه OB2x اجرا خواهد شد. برای فهم بهتر، مراحل کار را با ذکر یک مثال تشریح می‌کنیم.

مثال ۷- ۱۲ ایجاد تأخیر برای روشن شدن خروجی

با استفاده از وقفه OB2x برنامه‌ای بنویسد که با فعال شدن I0.0 خروجی‌های QW0 با تأخیر ۳ میلی‌ثانیه روشن شوند.

حل: ابتدا OB20 را ایجاد کرده و برنامه روشن شدن خروجی‌ها را در آن می‌نویسیم.

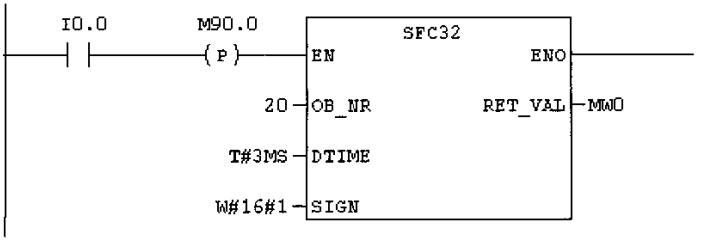
```
OB20 :  "Time Delay Interrupt"
Network 1: Title:
```



شکل ۷-۳۱ برنامه مثال ۷-۱۲

سپس در OB1 مانند شکل ۷-۳۲ SFC32 را صدا می‌زنیم.

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1 : Title:
```



شکل ۷-۳۲ ادامه برنامه مثال ۷-۱۲

همانطور که در این شکل دیده می‌شود، فعال‌سازی این SFC با لبه مثبت I0.0 انجام می‌شود، در مورد سایر ورودی و خروجی‌های آن به نکات زیر توجه نمایید:

- **OB_NR:** شماره OB مورد نظر به این ورودی داده می‌شود. پس در مثال فوق OB20 با تأخیر فعال خواهد شد.
- **DTIME:** زمان تأخیر به فرمت Time به این ورودی داده می‌شود. درمثال‌فوق‌این‌زمان ۳ میلی‌ثانیه داده شده است.
- **SIGN:** در این ورودی کاربر کد هگز دلخواهی را وارد می‌کند تا از آن در OB20 استفاده کند. این ورودی در اینجا عدد 1 داده شده است و در این برنامه استفاده‌ای ندارد. مثال بعدی استفاده از این ورودی را روشن می‌سازد.
- **RET_VAL:** این خروجی کد وضعیت اجرای SFC را برمی‌گرداند. اگر این خروجی صفر باشد وضعیت نرمال است، در غیر اینصورت خطایی وجود دارد.

پس از اینکه OB1 و OB20 به PLC دانلود شد، با لبه مثبت I0.0 و تأخیر 3ms خروجی‌ها فعال خواهند شد. برای تست با PLC یا سیمولاتور زمان تأخیر را افزایش دهید و وضعیت خروجی‌ها را ببینید.

فصل ۷

مثال ۷-۱۳ ایجاد تأخیر برای روشن و خاموش شدن خروجی

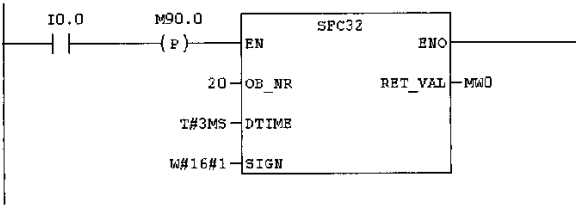
برنامه مثال ۷-۱۲ را به‌صورتی کامل کنید که:

- اگر I0.0 فعال شد، خروجی‌ها ی QW0 با تأخیر ۳ میلی‌ثانیه روشن شوند.
- اگر I0.1 فعال شد، خروجی‌های QW0 با تأخیر ۴ میلی‌ثانیه خاموش شوند.

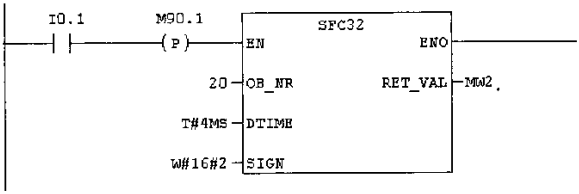
این مثال با هدف آشنایی با ورودی SIGN از SFC32 طراحی شده است.

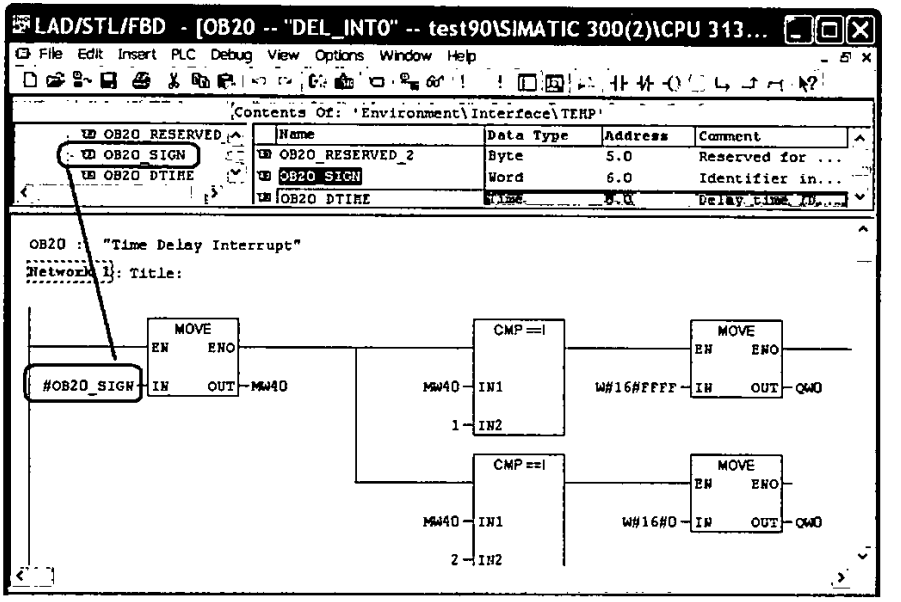
در برنامه OB1 دو بار SFC32 را به‌صورت زیر صدا می‌زنیم. همانطور که دیده می‌شود، برای روشن شدن عدد 1 و برای خاموش شدن عدد 2 را به ورودی SIGN اختصاص داده‌ایم. کد SIGN در OB20 توسط متغیر محلی OB20_SIGN در دسترس قرار می‌گیرد که از آن برای کنترل خروجی‌ها استفاده شده است.

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1 : Title:
```



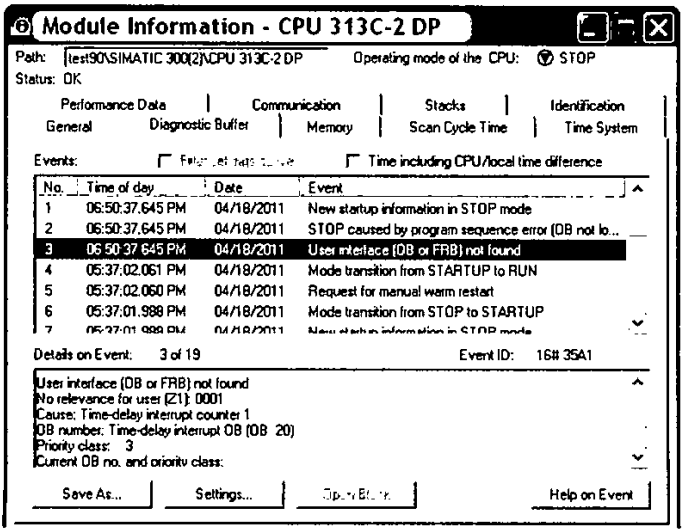

```
Network 2 : Title:
```





شکل ۷-۳۳ برنامه مثال ۷-۱۳

تذکر: اگر SFC32 برنامه‌نویسی و دانلود شود ولی وقفه‌ای که در آن مشخص شود به PLC دانلود نشده باشد، بعد از فعال شدن SFC32 و گذشت زمان تأخیر تعیین شده، CPU متوقف می‌شود و پیغام زیر در بافر ثبت می‌گردد.

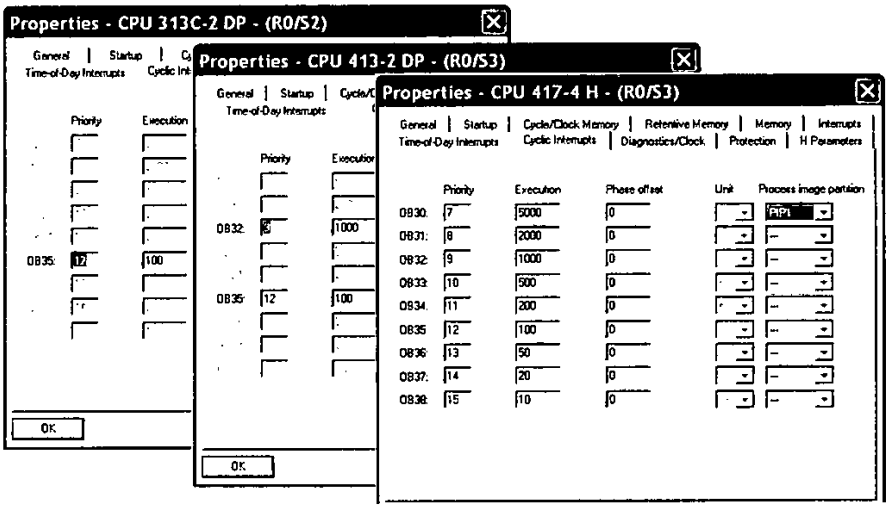


شکل ۷-۳۴ پیغام خطای ناشی از دانلود نشدن OB2x

فصل ۷

۷-۸ وقفه‌های OB3x (Cyclic Interrupt)

در بحث قبل دیدیم که وقفه‌های OB1x می‌توانند به‌صورت پریودیک تنظیم شوند؛ مثلاً هر دقیقه یک بار توسط CPU فراخوان گردند. گروه دیگری از وقفه‌ها هستند که وقفه سیکلی نام دارند و فقط به‌صورت پریودیک و مداوم کار می‌کنند. این وقفه‌ها با فواصل زمانی مشخص فراخوانی می‌گردند که زمان اجرای آنها قابل تغییر و تنظیم است. برای این گروه از وقفه‌ها OB30 تا OB38 معرفی شده است که بسته به نوع CPU ممکن است برخی از آنها قابل استفاده باشند. شکل ۷-۳۵ این OBها را برای سه نوع CPU متفاوت نشان می‌دهد.



شکل ۷-۳۵ وقفه OB3x در چند نمونه CPU

همانطور که در این شکل دیده می‌شود، زمان این وقفه‌ها دارای مقادیر پیش‌فرض است که برحسب میلی ثانیه است. بنابراین نسبت به OB1x دوره اجرای بسیار کوتاهی دارند.

جدول۷-۷ OBهای وقفه سیکلی و زمان تناوب آنها را در حالت پیش‌فرض نشان می‌دهد.

جدول ۷-۷

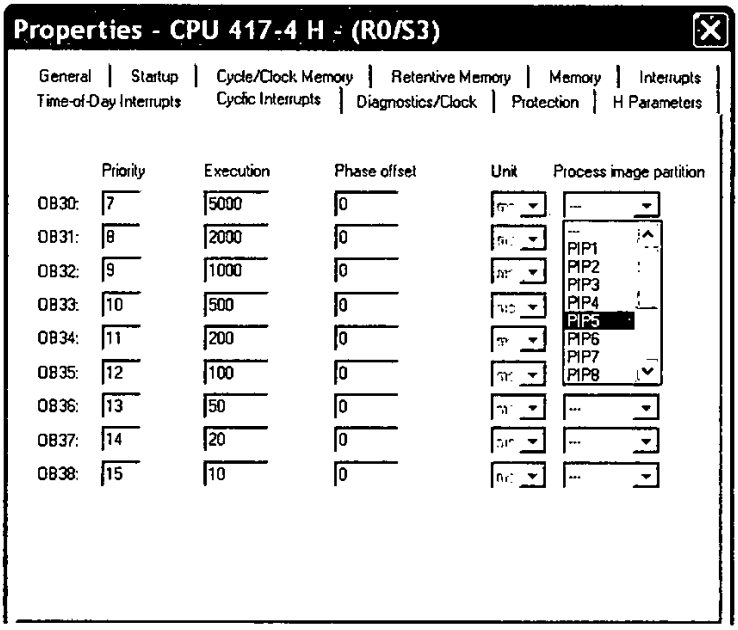
OB Number	Default Interval	Default Priority Class
OB30	5 s	7
OB31	2 s	8
OB32	1 s	9
OB33	500 ms	10
OB34	200 ms	11
OB35	100 ms	12
OB36	50 ms	13
OB37	20 ms	14
OB38	10 ms	15

همانطور که در این جدول مشاهده می‌شود، اولویت این گروه از وقفه‌ها نسبت به اولویت خانواده OB1x بالاتر است و بین ۷ تا ۱۵ می‌باشد. بنابراین این وقفه‌ها نه تنها OB1 بلکه OB1xها را نیز قطع می‌کنند.

تنظیم وقفه سیکلی در محیط HW Config

وقفه‌های OB3x برخلاف نوع OB1x نیاز به فعال‌سازی ندارند. در تنظیمات Hwconfig نیز دیده می‌شود که هیچ گزینه‌ای برای Activeسازی این وقفه‌ها وجود ندارد. به‌عبارت دیگر برای هر OB3x زمان از قبل تعریف شده‌ای وجود دارد و به‌محض اینکه OB3x مربوطه به CPU دانلود شود، به‌صورت سیکلی طبق زمان مورد نظر تکرار می‌شود. پس مشکلی که در OB1x وجود داشت و در صورت فعال‌سازی و عدم دانلود OB منجر به توقف CPU می‌شد در اینجا وجود ندارد.

در وقفه‌های OB3x در بسیاری از CPUها فقط زمان تکرار را می‌توان تغییر داد و در برخی دیگر از CPUها برخی از گزینه مانند شکل ۷-۳۶ قابل تغییر است.



شکل ۷-۳۶ تنظیمات وقفه OB3x

همانطور که در شکل ۷-۳۶ مشاهده می‌شود، در این سربرگ گزینه‌های زیر موجود می‌باشد:

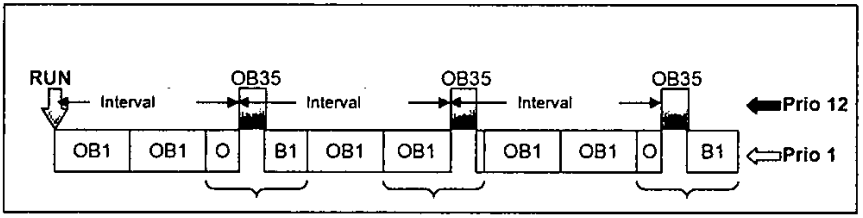
Priority

عدد نشان داده شده در این فیلد، بیانگر کلاس اولویت بلاک مورد نظر می‌باشد. مثلاً در شکل ۷-۳۶ کلاس اولویت OB35 عدد 12 نشان داده شده است. لازم به ذکر است که عدد صفر در کادر Priority یک بلاک نشان‌دهنده عدم

پشتیبانی CPU از آن بلاک بوده و در واقع آن بلاک در CPU موجود نیست. در اکثر CPUهای S7-300 امکان تغییر کلاس اولویت وجود ندارد، اما در CPUهای S7-400 می‌توان کلاس اولویت را تغییر داد.

Execution

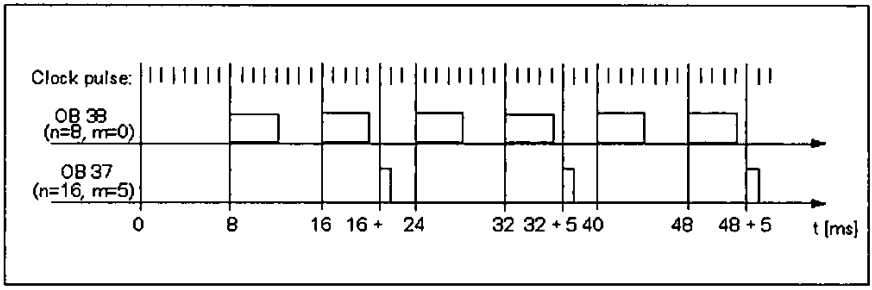
در این فیلد زمان اجرای بلاک بر حسب میلی‌ثانیه تعیین می‌شود. در واقع بلاک با فاصله‌ی زمانی تعیین شده در این فیلد به‌صورت سیکلی فراخوانی و در صورت وجود اجرا می‌شود. مثلاً در شکل فوق مشخص است که زمان اجرای OB35 روی 100 میلی‌ثانیه تنظیم شده است، یعنی هر 100 میلی‌ثانیه OB35 یک بار فراخوانی و اجرا می‌شود. لازم به ذکر است که زمان پیش‌فرض موجود در این فیلد را می‌توان تغییر داد. شکل ۷-۳۷ چگونگی اجرای OB وقفه سیکلی را بر اساس زمان تنظیم‌شده نشان می‌دهد.



شکل ۷-۳۷ نحوه فراخوانی و اجرای OB وقفه سیکلی

Phase offset

در این فیلد می‌توان زمانی را برحسب میلی‌ثانیه (یا زمان مبنای دیگری که در فیلد Unit تنظیم شده است) وارد نمود که در صورت فراخوانی همزمان دو وقفه سیکلی اجرای یکی از آنها با تأخیر صورت پذیرد. در صورتی‌که دو وقفه سیکلی همزمان فراخوانی شوند، OB80 که مربوط به خطاهای زمانی است فراخوانی شده و در صورت عدم اجرا CPU متوقف می‌گردد. با استفاده از تعیین Offset مناسب می‌توان از این رخداد جلوگیری نمود. شکل ۷-۳۸ این موضوع را نشان می‌دهد.



شکل ۷-۳۸ تاثیر Offset در اجرای وقفه سیکلی

در شکل فوق حرف n نشان‌دهنده زمان فراخوانی و حرف m زمان Offset است. همانطور که مشخص است، زمان فراخوانی OB38 برابر 8 میلی‌ثانیه و زمان Offset آن برابر صفر در نظر گرفته شده است. همچنین زمان فراخوانی

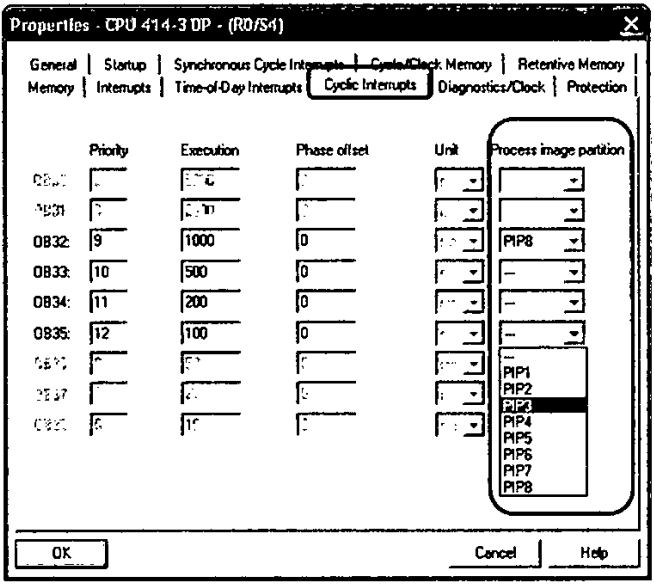
OB37 برابر 16 میلی‌ثانیه و زمان Offset آن برابر 5 میلی‌ثانیه در نظر گرفته شده است. در هنگام اجرای این وقفه‌ها در سکیل دوم یعنی در زمان 16 میلی‌ثانیه باید هر دو OB37 و OB38 همزمان فراخوانی شوند. در این زمان OB38 اجرا شده ولی OB37 پس از زمان Offset یعنی بعد از 5 میلی‌ثانیه اجرا می‌شود.

Unit

در این فیلد واحدِ زمان جهت اجرای OB وقفه سیکلی نشان داده شده و در برخی CPUها امکان تنظیم آن وجود دارد. به‌صورت پیش‌فرض این زمان روی ms (میلی‌ثانیه) تنظیم شده است. در CPUهایی‌که امکان تنظیم این زمان وجود دارد، واحد میکرو ثانیه نیز قابل انتخاب می‌باشد.

Part Process Image (Process Image Partition)

در CPUهایی که ناحیه PII آنها به‌صورت پارتیشن‌بندی شده است، می‌توان Update شدن بخشی از ناحیه‌ی PII را در زمان فراخوانی OBهای‌وقفه‌سیکلی انجام‌داد. برای‌این‌منظور کافیست در فیلد مربوطه یکی از ناحیه‌های PII را انتخاب نمود.

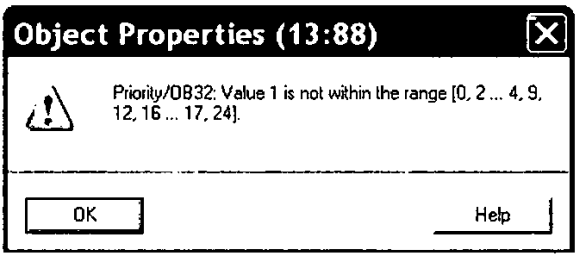


شکل ۷-۳۹ انتخاب ناحیه‌ای از PII جهت Update توسط وقفه‌های سیکلی

نکات مهم و قابل توجه در استفاده از وقفه‌های OB3x

- **نکته ۱:** همانطور که ذکر شد، در صورتی‌که تنظیمات وقفه‌های سیکلی انجام شود ولی عملاً OBهای مربوطه ایجاد و دانلود نشوند، مشکل خاصی برای CPU به‌وجود نمی‌آید. این در حالی است که در اکثر OBهای وقفه، عدم وجود OB مربوطه باعث توقف CPU می‌گردد.

- **نکته ۲:** این وقفه‌ها به‌دلیل زمان تکرار کوتاهی که دارند بار بیشتری را به CPU تحمیل می‌کنند. به‌عنوان مثال در نظر بگیرید که سیکل اسکن یک CPU در حالت عادی بدون وقفه 100 ms باشد و وقفه OB3x با زمان 10ms به آن دانلود شود. در اینصورت در هر سیکل اسکن ممکن است ده بار وقفه اتفاق بیفتد. هر چه زمان OB3x کوتاه‌تر باشد تعداد دفعات وقفه در سیکل اسکن بیشتر می‌شود.
اگر در مثال ذکر شده فرض کنیم که زمان اجرای دستورات نوشته شده در OB3x به اندازه 5 ms طول بکشد و در یک سیکل اسکن وقفه ده بار تکرار شود در نتیجه سیکل اسکن به اندازه 50 ms افزایش خواهد یافت و اگر از حد مجاز تعریف شده بیشتر شود CPU متوقف خواهد شد.
- **نکته ۳:** اگر برنامه نوشته شده در OB3x زیاد باشد یا زمان تعیین شده برای تکرار OB3x خیلی کوتاه باشد به‌صورتی که CPU نتواند در زمان تعیین شده برای تکرار OB3x برنامه را به پایان برساند، در اینصورت CPU با فالت متوقف شده و پیغام مربوط به Time Error در بافر ثبت می‌گردد. به‌عنوان مثال فرض کنید که وقفه OB38 با زمان 10ms تنظیم شده ولی برنامه‌ای که در آن نوشته شده زیاد باشد و CPU نتواند قبل از 10ms پردازش این برنامه را تمام کند در اینصورت چراغ SF یا INTF روشن می‌شود و CPU متوقف می‌گردد. این خطا را می‌توان با استفاده از OB80 که در ادامه تشریح می‌شود ماسک نمود.
- **نکته ۴:** با استفاده از فانکشن‌های سیستم SFC39 تا SFC42 می‌توان وقفه‌های سیکلی را غیرفعال کرد یا به آنها تأخیر داد یا مجدداً فعال نمود.
- **نکته ۵:** اگر درجه اولویت این وقفه قابل تغییر باشد هر عددی نمی‌توان به آن اختصاص داد. فقط درجه اولویت‌های خاصی قابل استفاده است در غیر اینصورت با پیغام خطای زیر مواجه می‌شویم.



شکل ۷-۴۰ خطای ناشی از اختصاص درجه اولویت نادرست به وقفه

- **نکته ۶:** اگر تایمر در OB3x استفاده شود، در برخی موارد پاسخ درستی به‌دست نمی‌آید. علت این است که اولاً ممکن است زمان تایمر کمتر از زمان OB3x باشد که در اینصورت زمانی خروجی تایمر قبل استفاده است که Ob3x در سیکل بعدی فراخوان شده باشد. در این حالت اگر چه تایمر در حافظه سیستم به‌درستی کار می‌کند ولی نتیجه کار آن با تأخیر ظاهر می‌شود. اگر حتی زمان تایمر بیش از زمان OB35 باشد باز عملکرد آن درست نخواهد بود. به‌عنوان مثال اگر زمان OB35 روی یک ثانیه و زمان تایمر روی دو ثانیه تنظیم شده باشد، ممکن است وقتی که ورودی فعال

کننده تایمر یک می‌شود هنوز OB3x فراخوان نشده باشد و تایمر با تأخیر فعال می‌شود در نتیجه خروجی آن بیش از ۲ ثانیه بعد عمل می‌کند. با توجه به این نکات توصیه می‌شود تایمر را در OB3x به‌کار نبرید.

- **نکته ۷:** مشابه بحث تایمرها این مشکل برای کانتر نیز وجود دارد. اگر کانتری در OB35 صدا زده شود و ورودی آن قبل از اینکه OB3x فراخوان شود یک و سپس صفر شود، OB3x آنرانخواهددیدو نتیجه شمارش درست نخواهد بود.

موارد کاربرد وقفه‌های سیکلی

وقفه‌های OB3x در عمل بسیار پر کاربرد هستند. در کنترل فرآیندهای بزرگ ممکن است OB1x یا OB2x در CPU موجود نباشد ولی بعید است که از یکی از وقفه‌های OB3x استفاده نشده باشد. این وقفه در PCS7 که به‌عنوان سیستم DCS زیمنس مطرح می‌شود نیز بسیار مهم است. بسیاری از بلاک‌های برنامه‌نویسی PCS7 بایستی در OB3x فراخوان شوند و اگر در OB1 صدا زده شوند کار نمی‌کنند.

به‌طور کلی هر جا که لازم است کاری به‌صورت منظم و تکراری با فاصله‌های زمانی کوتاه اجرا شود، از OB3x استفاده می‌شود. برخی از کاربردهای مهم آن در عمل عبارتند از:

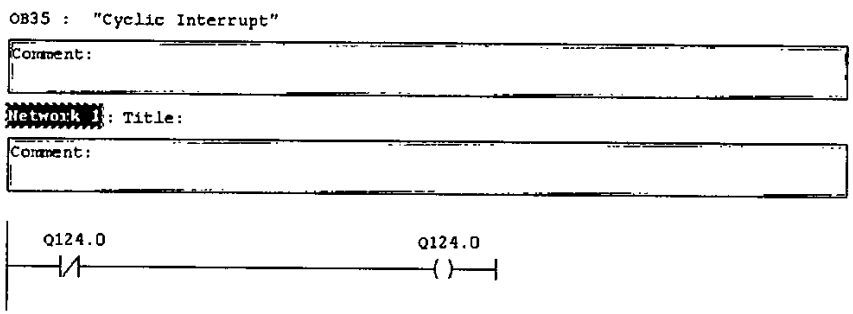
- **PID Control:** در کنترل لوپ زمان نمونه‌برداری از فیدبک و ارسال فرمان به عملگر بایستی با نظم خاصی انجام شود. برای این منظور از OB3x استفاده می‌گردد. PID Control در فصل بعد تشریح شده است.
- **تبادل دیتا روی شبکه‌های صنعتی:** وقتی چند PLC روی یک شبکه با یکدیگر تبادل دیتا می‌کنند لازم می‌شود که فرستنده با نظم خاصی اقدام به ارسال کند. زیرا اگر فاصله زمانی از حدی کمتر باشد ممکن است هنوز دیتای قبلی به‌طور کامل ارسال نشده (بافر خالی نشده) و ارسال مجدد اتفاق بیفتد که در این حالت فالت ارتباطی رخ می‌دهد. این موضوع در کتاب‌های شبکه صنعتی به‌طور کامل مورد بحث قرار گرفته است.
- **ایجاد کنتور نرم‌افزاری:** کنتورهای نرم‌افزاری براساس انتگرال‌گیری از یک متغیر با تغییرات نامشخص عمل می‌کنند. برای انتگرال‌گیری با نظم مشخص بهترین گزینه استفاده از OB3x است. مثالی از طراحی کنتور نرم افزاری در ادامه تشریح خواهد شد.

ارائه چند مثال برای وقفه OB3x

مثال ۷-۱۴ تولید پالس با فرکانس مشخص

با استفاده از OB35 برنامه‌ای بنویسید که بتواند پالسی با فرکانس 4HZ ایجاد نماید.

حل: زمان تناوب برابر است با 1/f که در این فرمول f فرکانس می‌باشد. بنابراین زمان تناوب این پالس برابر 250 =1/4 ms می‌باشد. پس زمان اجرای OB35 را روی زمان 125 ms یعنی نصف زمان تناوب تنظیم نموده و سپس برنامه‌ی زیر را در OB35 می‌نویسیم.

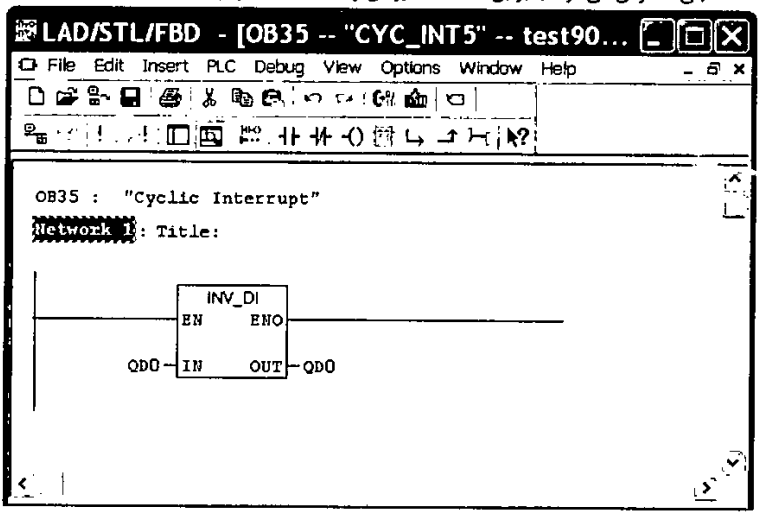


شکل ۷-۴۱ برنامه مثال ۷-۱۴

همانطور که مشخص است، در این برنامه در سیکل اول خروجی روشن و در سیکل بعدی خاموش می‌شود. در واقع در هر دو سیکل فراخوانی OB35 ، خروجی Q124.0 یک بار روشن خاموش می‌شود. از آنجایی که فاصله زمانی اجرای OB35 روی 125 میلی‌ثانیه تنظیم شده است، لذا در یک ثانیه 8 بار اجرا شده که معادل چهار بار روشن خاموش شدن Q124.0 است. بنابراین فرکانس 4HZ به‌وجود می‌آید. به این روش می‌توان هر پالس با فرکانس دلخواه را ایجاد نمود.

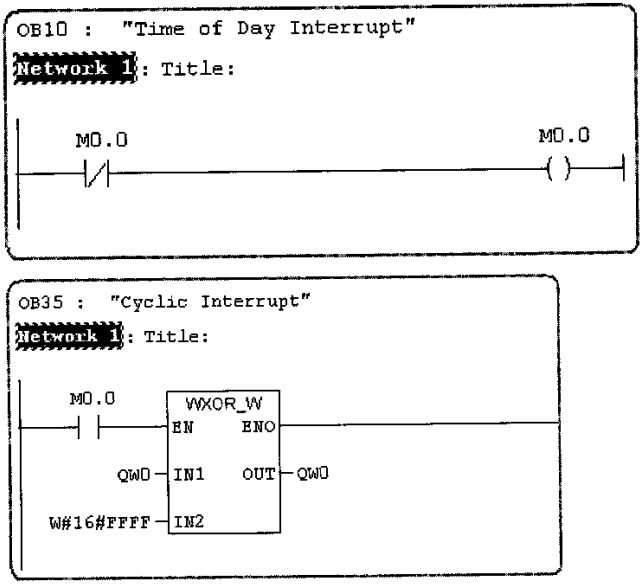
مثال ۷-۱۵ خاموش و روشن شدن همه خروجی‌ها

برنامه زیر در OB35 که زمان آن 100ms است نوشته شده است. هر بار که OB35 فراخوان می‌شود تمام ۳۲ خروجی نسبت به حالت قبلی معکوس می‌شوند بنابراین 100 ms روشن و 100 ms خاموش خواهند بود.



شکل ۷-۴۲ برنامه مثال ۷-۱۵

تمرین ۷- ۲: در برنامه زیر از OB10 با دوره اجرای هر دقیقه یک بار و OB35 با دوره اجرای 100ms استفاده شده است. بررسی کنید وضعیت QW0 چگونه خواهد بود ؟



شکل ۷-۴۳ برنامه تمرین ۷-۲

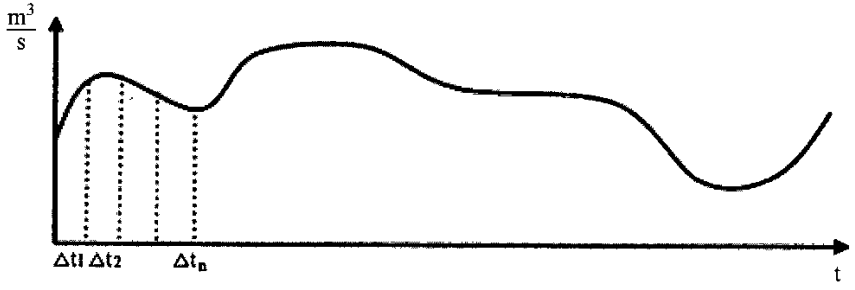
مثال ۷-۱۶ کنتور نرم‌افزاری

کنتورهای سخت‌افزاری براساس انتگرال‌گیری از یک مقدار متغیر با فاصله زمانی ثابت که اصطلاحاً delta T گفته می‌شود کار می‌کنند. بر همین اساس می‌توان کنتوری به‌صورت نرم‌افزاری طراحی کرد که با برنامه‌نویسی مشابه کنتورهای سخت افزاری کارکند. بعلاوه دارای امکانات دلخواه باشد به‌عنوان مثال بتوان آنرا روی محیط HMI نمایش داده روند تغییرات مصرف را بررسی کرد، آلارم ایجاد نمود و ... .

مثالی که در اینجا طرح می‌شود برای طراحی کنتور آب در یک کارخانه بزرگ است که با انتگرال‌گیری از دبی لحظه‌ای که از فلو ترانسمیتر دریافت می‌کند کار می‌کند.

دبی لحظه‌ای آب توسط فلو ترانسمیتر اندازه‌گیری می‌شود. این ترانسمیتر به ازای فلوی 0 الی $10\ m^3$ در ثانیه مقدار 4 الی 20 mA به کارت AI در PLC تحویل می‌دهد.

حل: در این مثال در واقع ما با یک منحنی با تغییرات نامشخص که میزان لحظه‌ای دبی آب مصرفی را نشان می‌دهد مواجه هستیم. سطح زیر این منحنی، معرف میزان کل مصرف است. پس بایستی از آن انتگرال گرفت.



شکل ۷-۴۴ تغییرات دبی آب در مثال ۷-۱۶

از آنجا تغییرات نامشخص و تابع مصرف است، نمی‌توان از منحنی که تابع آن مشخص نیست به روال معمولی انتگرال گرفت. یک روش آن است که سطح را با مستطیل‌هایی با عرض ثابت تقریب بزنیم و مساحت آنها را محاسبه و با هم جمع کنیم.

همانطور که در شکل ۷-۴۴ مشخص است، می‌توان در زمان‌های $\Delta t1$ الی Δtn میزان دبی را اندازه‌گیری نموده و طبق فرمول زیر عمل نمود:

$$\text{سطح زیر منحنی} = \sum(\Delta t1 * \text{(مقدار لحظه‌ای دبی)}) + (\Delta t2 * \text{(مقدار لحظه‌ای دبی)}) + \ldots + (\Delta t3 * \text{مقدار لحظه‌ای دبی})$$

اگر بتوان در فواصل زمانی مشخصی مقدار دبی را اندازه‌گیری نموده و در فاصله زمانی ضرب نمود، می‌توان مقدار سطح زیر منحنی که همان مقدار دبی کل باشد را محاسبه نمود. اگر فواصل زمانی اندازه‌گیری دبی غیر برابر باشند، عملاً امکان برنامه‌نویسی برنامه‌ای جهت محاسبه‌ی دبی کل وجود ندارد. بنابراین باید به‌صورتی عمل کرد که فواصل نمونه‌برداری (اندازه‌گیری لحظه‌ای دبی) یکسان باشد. اگر برنامه مورد نظر در OB1 نوشته شود، از آنجایی که فواصل زمانی اجرای OB1 مقدار ثابتی نیست و در هر سیکل ممکن است با سیکل دیگر متفاوت باشد عملاً برنامه درست کار نمی‌کند. بهترین OB جهت برنامه‌نویسی کنتور نرم‌افزاری، بلاک‌های وقفه سیکلی می‌باشند. در این مثال برنامه مورد نظر در OB35 نوشته می‌شود. زمان اجرای OB35 در محیط روی 100 ms تنظیم شده است.

بنابراین فرمول قبلی به‌دلیل یکسان بودن فواصل نمونه‌برداری به‌صورت زیر ساده می‌شود.

$$\text{سطح زیر منحنی} = \sum \Delta t * \text{مقدار لحظه‌ای دبی}$$

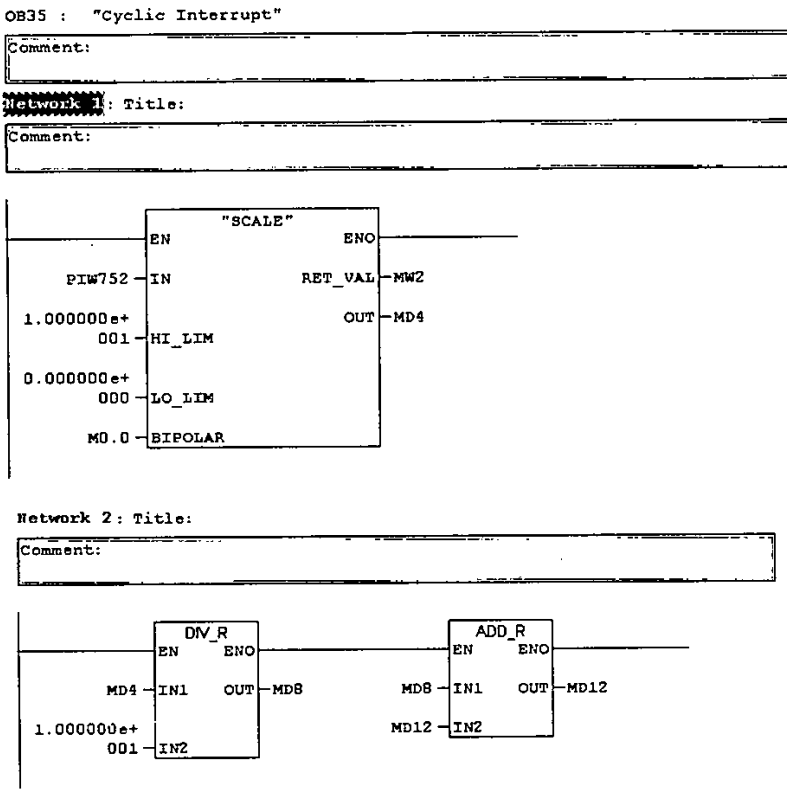
اگر نمونه‌گیری در OB35 و در فواصل 100 ms انجام شود، با توجه به اینکه واحد اندازه‌گیری فلو نیز m^3/s است بنابراین داریم:

$$\text{سطح زیر منحنی} = \sum 100\,\text{ms} * \text{مقدار لحظه‌ای دبی} \; (m^3/s)$$

با ساده‌سازی داریم:

$$\text{سطح زیر منحنی} = \sum (m^3) \frac{\text{مقدار لحظه ای دبی}}{10}$$

بنابراین کافیست مقدار لحظه‌ای دبی را از ترانسمیتر خوانده و پس از Scale کردن، آنرا بر ۱۰ تقسیم نماییم. در هر بار فراخوانی OB35 این مقدار باید با مقدار قبلی جمع زده شده و در یک خروجی ذخیره شود. برنامه مورد نظر به‌صورت زیر در OB35 پیاده‌سازی می‌شود.



شکل ۷-۴۵ برنامه مثال ۷-۱۶

میزان مصرف در MD8 قابل دسترس است که می‌توان آنرا به محیط مانیتورینگ انتقال داد.

مشکلی که در برنامه فوق وجود دارد این است که اگر زمان OB35 در تنظیمات سخت‌افزاری تغییر کند کنتور درست کار نخواهد کرد. راه حل این است که زمان نمونه‌برداری به‌صورت یک متغیر در برنامه وارد شود تا در صورت تغییر محاسبات دچار مشکل نگردد. برای این کار لازم است با متغیرهای محلی OB35 آشنا شویم.

متغیرهای محلی OBهای وقفه سیکلی

جدول ۷-۸ متغیرهای محلی وقفه‌های سیکلی را نشان می‌دهد.

جدول ۷-۸

Variable	Type	Description
OB35_EV_CLASS	BYTE	Event class and identifiers B#16#11: interrupt is active
OB35_STRT_INF	BYTE	• B#16#30: Start request for cyclic interrupt OB with special criteria (only for H-CPUs and there only if explicitly configured for them) • B#16#31: start request for OB30 • B#16#36: start request for OB35 • B#16#39: start request for OB38 • B#16#3A: Start request for cyclic interrupt OB with special criteria (only for S7-300 and there only if explicitly configured for them)
OB35_PRIORITY	BYTE	Assigned priority class: defaults 7 (OB30) to 15 (OB38)
OB35_OB_NUMBR	BYTE	OB number (30 to 38)
OB35_RESERVED_1	BYTE	Reserved
OB35_RESERVED_2	BYTE	Reserved
OB35_PHASE_OFFSET	WORD	• If OB35_STRT_INF=B#16#3A: phase offset in • µs In all other cases: phase offset in ms
OB35_RESERVED_3	INT	Reserved
OB35_EXC_FREQ	INT	• If OB35_STRT_INF=B#16#3A: phase offset in • µs In all other cases: interval in milliseconds
OB35_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

این متغیرها تنظیمات انجام شده برای وقفه را بر می‌گردانند. به‌عنوان مثال متغیر محلی OB35_EXC_FREQ زمان تنظیم شده در hwconfig را به‌صورت یک عدد صحیح برمی‌گرداند.

مثال ۷-۱۷ رفع مشکل کنتور نرم‌افزاری مثال قبلی

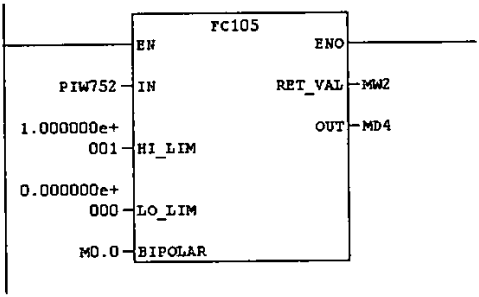
در مثال قبلی ذکر شد که برنامه در صورتی درست کار می‌کند که زمان وقفه 100 ms باشد. اگر این زمان توسط کاربر تغییر داده شود، محاسبات کنتور غلط خواهد بود. برای رفع مشکل فوق از متغیر #OB35_EXC_FREQ به‌عنوان زمان Delta T در برنامه به‌صورت یک متغیر استفاده می‌کنیم. از آنجا که این متغیر از جنس Integer است بایستی آن را به Real تبدیل کرد تا محاسبات برنامه درست باشد. برنامه به‌صورت زیر نوشته شده است. تحلیل برنامه بر عهده خواننده است.

OB35 : "Cyclic Interrupt"

Comment:

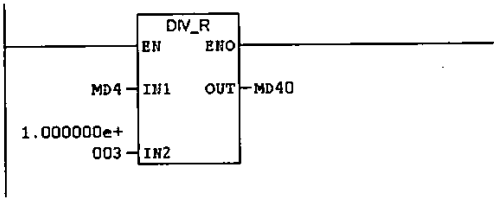
Network 1 : Title:

Scaling Flow



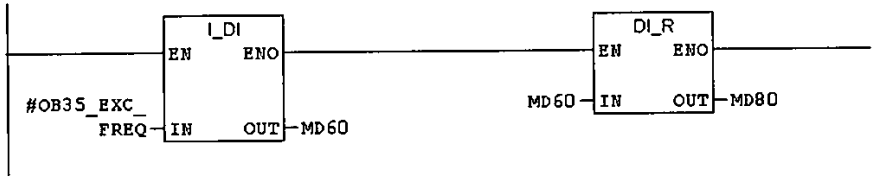
Network 2 : Title:

convert m3/s to m3/ms

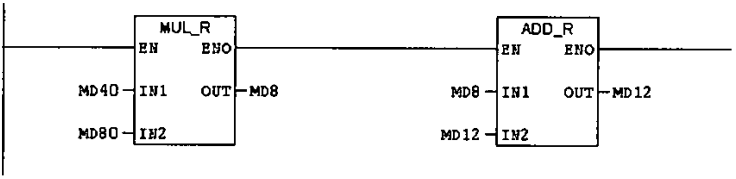


Network 3 : Title:

convert interval from integer to real



```
Network 4 : Title:
integrating
```

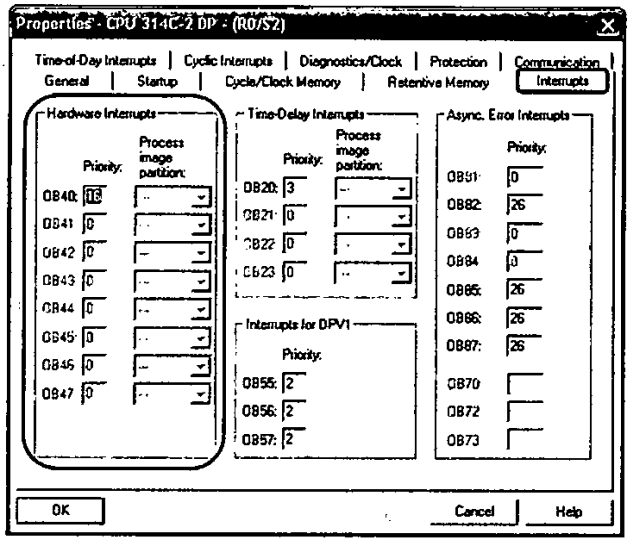


شکل ۷-۴۶ برنامه مثال ۷-۱۷

۷-۹ وقفه‌های OB4x (Hardware Interrupt)

OBهای وقفه Hardware Interrupt همانطور که از نامشان مشخص است وابسته به سخت‌افزار هستند و اجرای آنها به تغییر شرایط در سمت سخت‌افزار PLC بستگی دارد. این وقفه‌ها به وقفه‌های واکنش سریع نیز موسوم هستند. زیرا عملکرد آنها به‌گونه‌ای است که در صورت بروز شرایط سخت‌افزاری خاص CPU کار عادی خود را رها کرده و این وقفه را اجرا می‌کند. منظور از شرایط سخت‌افزاری که منجر به تریگر شدن این وقفه می‌شوند شرایط فالت در سخت افزار نیست بلکه شرایطی مانند بسته شدن یک سوئیچ یا بالارفتن یک مقدار آنالوگ و امثال این‌هاست. به‌عبارت دیگر می‌توان گفت که این وقفه برای ورودی و خروجی‌های آنالوگ و دیجیتال که بسیار حساس هستند استفاده می‌شود.

در S7، OB40 الی OB47 برای این وقفه در نظر گرفته شده است. در برخی از CPUها فقط یک OB از این خانواده مانند شکل ۷-۴۷ وجود دارد.



شکل ۷-۴۷ سربرگ Interrupt در پنجره مشخصات CPU

همانطور که در شکل دیده می‌شود، درجه اولویت این وقفه نسبت به وقفه‌های قبلی که تا اینجا بحث شد بالاتر است، بنابراین این وقفه می‌تواند تمامی آنها را نیز قطع کند. مقادیر پیش‌فرض در جدول ۷-۹ آمده است. می‌توان به هریک از آنها بسته به نیاز عددی بین ۱۶ تا ۲۳ اختصاص داد.

جدول ۷-۹

Organization Blocks	Priority Class
OB40	16
OB41	17
OB42	18
OB43	19
OB44	20
OB45	21
OB46	22
OB47	23

نکته دیگری که در شکل ۷-۴۷ مشخص است این است که وقفه‌های OB4x در پارامترهای CPU تنظیمی برای فعال شدن ندارند. در واقع تنظیم فعال‌سازی بایستی روی کارت I/O صورت گیرد.

به‌طور کلی شرایط استفاده از این وقفه‌ها عبارتست از:

۱- کارت I/O قابلیت این وقفه را داشته باشد. فقط کارت‌های SM که در تنظیمات آنها گزینه Hardware Interrupt وجود دارد دارای این قابلیت هستند.

۲- قابلیت فوق روی کارت فعال شده و به PLC دانلود شده باشد.

۳- OB4x برنامه‌نویسی و دانلود شده باشد.

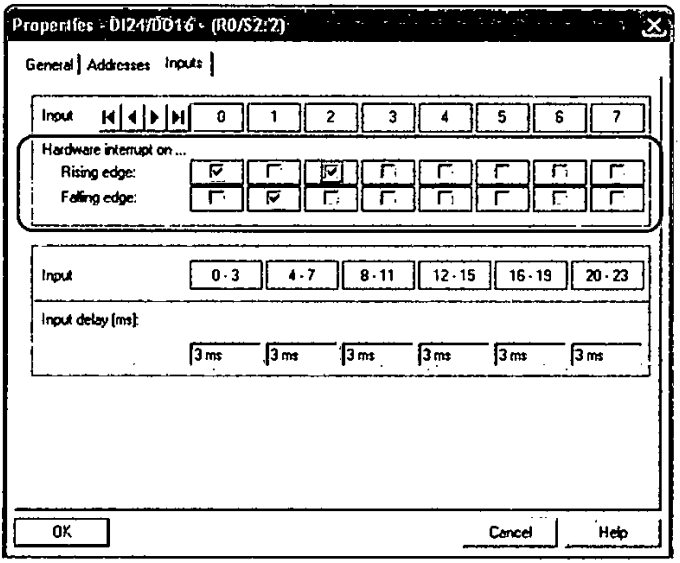
تنظیمات وقفه Hardware Interrupt

در کارت‌های DI

این تنظیم را می‌توان برای کارت‌های DI که این قابلیت را دارا هستند به‌گونه‌ای انجام داد تا در صورت اعمال لبه‌ی بالارونده یا پایین رونده به کانال مورد نظر این وقفه فعال شود. برای این کار باید در محیط HWConfig روی ماژول مورد نظر دوبار کلیک نموده و در سربرگ Inputs در قسمت Hardware Interrupt یکی از گزینه‌های زیر را انتخاب نمود:

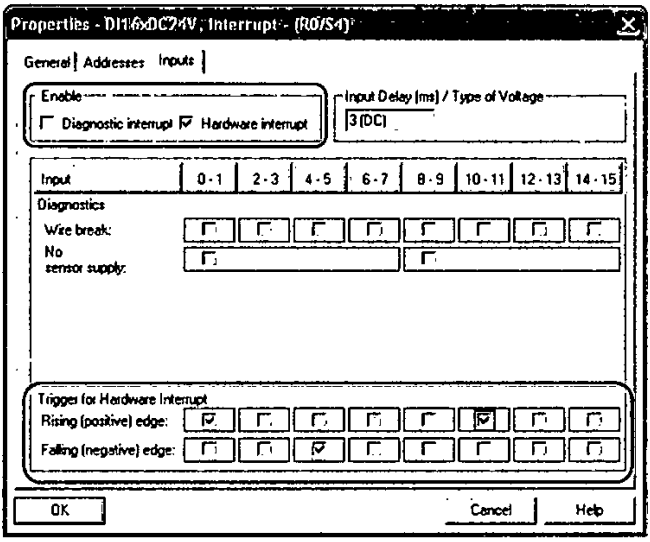
Rising edge: در صورتی‌که این گزینه انتخاب شود، با اعمال لبه‌ی بالارونده سیگنال به کانال مورد نظر وقفه سخت‌افزاری فعال می‌شود.

Falling edge: در صورتی‌که این گزینه انتخاب شود، با اعمال لبه‌ی پایین‌رونده سیگنال به کانال مورد نظر وقفه سخت‌افزاری فعال می‌شود.



شکل ۷-۴۸ تنظیم وقفه سخت‌افزاری

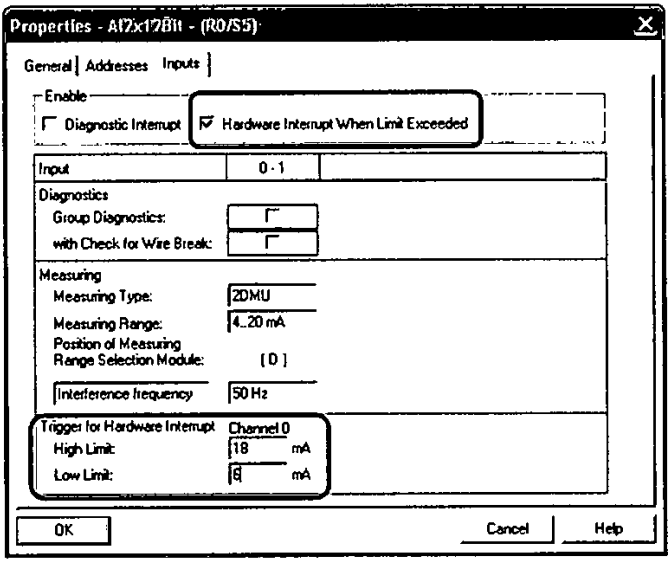
در برخی از کارت‌ها علاوه بر انتخاب گزینه Rising یا Falling باید گزینه Hardware Interrupt را علامت زد تا این قابلیت امکان فعال‌شدن را داشته باشد. لبه بالا رونده برای سوئیچ‌های نرمال باز و لبه پایین رونده برای سوئیچ‌های نرمال بسته استفاده می‌شود.



شکل ۷-۴۹ تنظیم وقفه سخت‌افزاری در کارت DI

در کارت‌های AI

در کارت‌های آنالوگ ورودی که دارای این قابلیت هستند می‌توان برای سیگنال ورودی یک محدوده مجاز تعیین نمود. در صورتی‌که سیگنال ورودی از محدوده مجاز خارج باشد؛ این وقفه فعال می‌شود. مثلاً در شکل ۷-۵۰ نوع سیگنال ورودی جریانی و رنج آن 4mA الی 20mA انتخاب شده است. در این کارت با علامت‌زدن گزینه Hardware interrupt when limit value exceeded وقفه فعال شده و می‌توان در کادر پایین صفحه در قسمت Trigger for Hardware Interrupt در قسمت High Limit محدوده مجاز بالا و در قسمت Low Limit محدوده مجاز پایین را مشخص نمود. در این شکل محدوده بالا 18 mA و محدوده پایین 6 mA تعیین شده است. اگر سیگنال ورودی خارج از این محدوده باشد وقفه سخت‌افزاری فراخوانی می‌گردد.

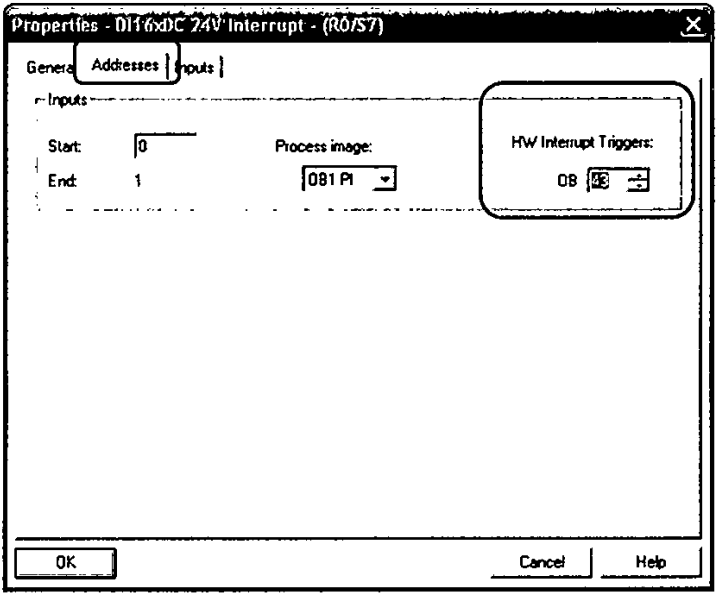


شکل ۷-۵۰ تنظیمات وقفه سخت‌افزاری در کارت AI

اگر در پنجره فوق تنظیم کانال به‌صورت ولتاژی بود، حدود تریگر براساس ولتاژ انجام می‌شد.

نکات مهم و قابل توجه در استفاده از وقفه‌های OB4x

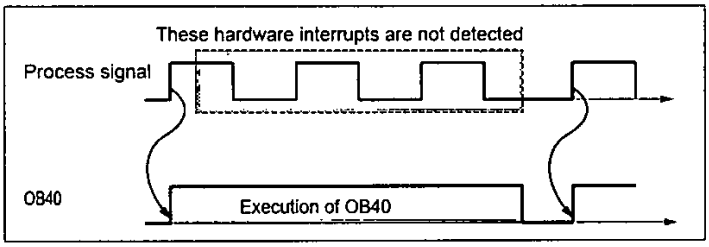
- **نکته ۱:** در صورت فعال‌شدن وقفه Hardware Interrupt ، OB40 یا بلاک دیگری که تنظیم شده است فراخوان می‌شود. بدیهی است که عدم وجود OB وقفه موجب توقف CPU می‌گردد.
- **نکته ۲:** در صورتی‌که CPU بیش از یک بلاک جهت وقفه سخت‌افزاری پشتیبانی نماید، باید در پنجره مشخصات کارت مورد نظر و در سربرگ Address، بلاک وقفه سخت‌افزاری را انتخاب نمود. شکل ۷-۵۱ این موضوع را نشان می‌دهد.



شکل ۷-۵۱ تعیین شماره OB وقفه سخت‌افزاری در کارت DI 16xDC24V

- **نکته۳:** هنگامی که وقفه سخت‌افزاری اتفاق می‌افتد سیستم عامل اسلاتی که ماژول مورد نظر در آن قرار گرفته است را شناسایی نموده و OB وقفه متناسب با تنظیمی که در آن ماژول صورت گرفته را اجرا می‌کند. هنگامی که اجرای بلاک وقفه به پایان رسید ماژول یک پیام تصدیق به سیستم عامل ارسال می‌کند. در صورتی که قبل از پایان اجرای بلاک وقفه و ارسال پیام تصدیق، وقفه سخت‌افزاری دیگری در همان ماژول اتفاق بیافتد یکی از دو حالت زیر پیش می‌آید:

۱- اگر اتفاق در کانالی رخ دهد که قبلاً وقفه سخت‌افزاری را فعال نموده است، در خواست جدید اجرای وقفه از بین می‌رود. به عبارتی دیگر تا پایان اجرای وقفه قبلی و ارسال پیام تصدیق توسط ماژول، وقفه دیگری فراخوانی و اجرا نمی‌شود. شکل ۷-۵۲ این موضوع را نشان می‌دهد.



شکل ۷-۵۲ عملکرد وقفه OB40

۲- اگر اتفاقی که باعث فعال‌سازی وقفه می‌گردد در سایر کانال‌های آن ماژول رخ دهد، درخواست اجرای وقفه ذخیره شده و پس از پایان اجرای وقفه قبلی، درخواست جدید اجرا می‌گردد.

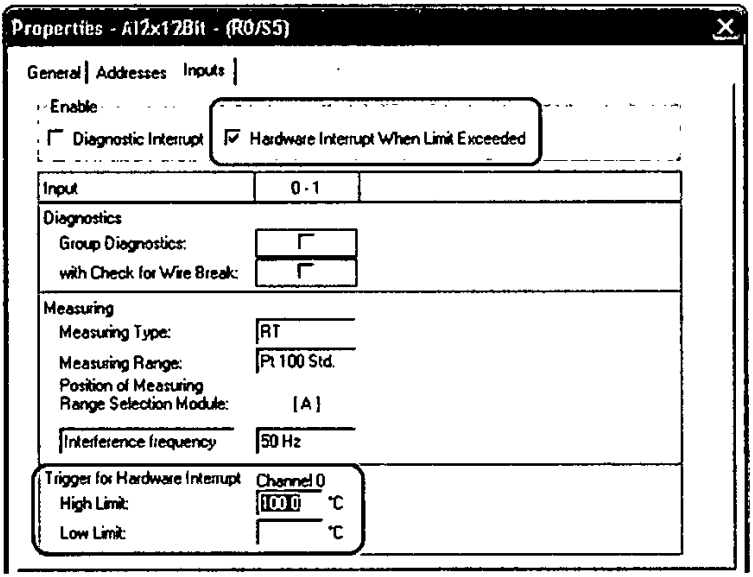
- **نکته۴:** توسط SFCهای 39 تا 42 می‌توان این وقفه را غیرفعال، دچار تأخیر در اجرا یا مجدداً فعال نمود.
- **نکته ۵:** توسط SFCهای 55 تا 57 نیز می‌توان پارامترهای لازم جهت تنظیم وقفه سخت‌افزاری را به یک ماژول اختصاص داد.

مثال ۷-۱۸

با استفاده از قابلیت Hardware Interrupt برنامه‌ای بنویسد که درصورتی‌که دمای ارسالی یک RTD به کارت آنالوگ ورودی از ۱۰۰ درجه بیشتر شد، خروجی Q14.7 روشن شود.

حل: برای حل این مثال لازم است مراحل زیر طی شود:

۱- در محیط HW Config وارد مشخصات کارت AI شده و قابلیت Hardware Interrupt را فعال می‌نماییم.



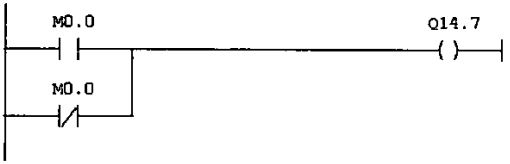
شکل ۷-۵۳ تنظیمات مثال ۷-۱۸

۲- کانال مورد نظر را روی حالت RTD تنظیم نموده و در قسمت High limit مقدار ۱۰۰ را وارد می‌کنیم.

۳- تنظیمات را پس از Save & Compile به PLC دانلود می‌کنیم.

۴- OB40 را ایجاد نموده و برنامه مورد نظر را در آن می‌نویسیم. سپس این بلاک را به PLC دانلود می‌نماییم.

```
OB40 :  "Hardware Interrupt"
Comment:

Network 1: Title:
Comment:
```



شکل ۷-۵۴ برنامه مثال ۷-۱۸

متغیرهای محلی OBهای وقفه سخت‌افزاری

جدول ۷-۱۰متغیرهای محلی وقفه‌های سخت‌افزاری را نشان می‌دهد.

جدول ۷-۱۰

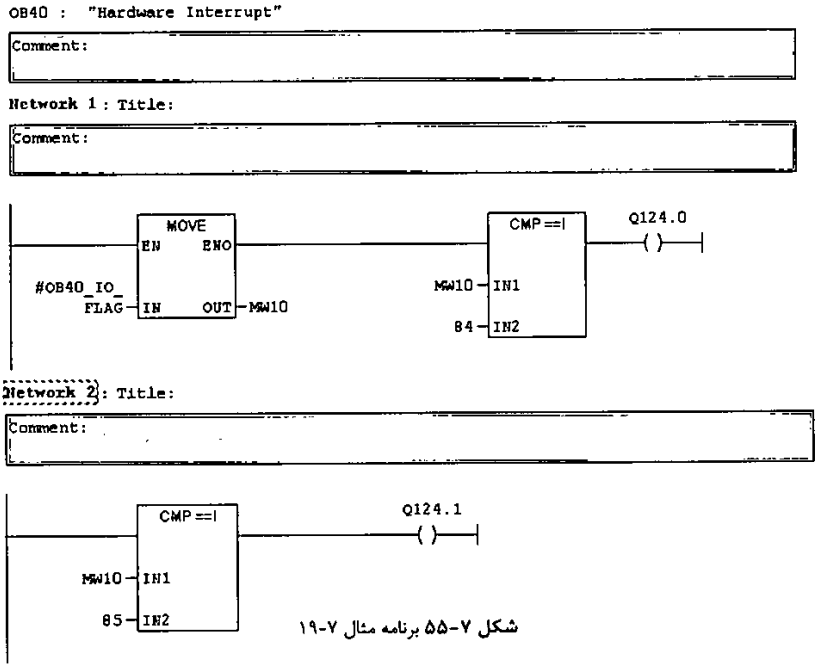
Variable	Type	Description
OB40_EV_CLASS	BYTE	Event class and identifiers: B#16#11: interrupt is active
OB40_STRT_INF	BYTE	• B#16#41: interrupt via interrupt line 1 • B#16#42: interrupt via interrupt line 2 (only with an S7-400) • B#16#43: interrupt via interrupt line 3 (only with an S7-400) • B#16#44: interrupt via interrupt line 4 (only with an S7-400) • B#16#45: WinAC: interrupt triggered via PC
OB40_PRIORITY	BYTE	Assigned priority class: defaults 16 (OB40) to 23 (OB47)
OB40_OB_NUMBR	BYTE	OB number (40 to 47)
OB40_RESERVED_1	BYTE	Reserved
OB40_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB40_MDL_ADDR	WORD	Logical base address of the module that triggers the interrupt

OB40_POINT_ADDR	DWORD	• For digital modules: bit field with the statuses of the inputs • on the module(Bit 0 corresponds to the first input) The • assignment the bits from OB40_POINT_ADDR to the channels in the module can be found in the description for the given module. For analog modules: Bit field, Informing which channel has exceeded which limit (for detailed info on the structure refer to /71/ or /101/). For CPs or IMs: Module interrupt status (not user relevant)
OB40_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

مثال ۷-۱۹

برنامه نوشته شده در مثال ۷-۱۸ را به‌گونه‌ای تغییر دهید که بتوان مشخص نمود کارتی که باعث فعال‌شدن وقفه شده است ورودی است یا خروجی. اگر کارت ورودی باشد خروجی Q124.0 و اگر کارت ورودی باشد Q124.1 روشن شود.

حل: همانطور که در جدول ۷-۱۰ مشخص است، متغیر محلی OB40_IO_FLAG کدی می‌دهد که نشان‌دهنده ورودی یا خروجی بودن کانالی است که باعث اجرای وقفه سخت‌افزاری شده است. این متغیر به ازای ماژول ورودی کد W#16#54 که معادل عدد 84 در کد Integer است و به ازای ماژول ورودی کد W#16# 55 که معادل عدد 85 در کد Integer است را بر می‌گرداند. بنابراین می‌توان با مقایسه عدد قرار گرفته در این پارامتر، نوع ماژول را مشخص نمود. برنامه‌ی زیر در OB40 نوشته می‌شود.



شکل ۷-۵۵ برنامه مثال ۷-۱۹

۷-۱۰ وقفه‌های OB8x (Asynchronous Error)

آنچه تاکنون ذکر شد مربوط به به وقفه‌های مبتنی بر Event بود. وقفه‌های دیگری نیز داریم که مبتنی بر Error و خطا هستند. این خطاها را می‌توان به دو دسته آسنکرون و سنکرون تقسیم نمود. وقفه‌های آسنکرون مربوط به اشکالات سخت افزاری و وقفه‌های سنکرون مربوط به اشکالات برنامه‌نویسی هستند.

به‌طور کلی مزیت بزرگ استفاده از این OBها چه برای خطاهای سنکرون و چه برای خطاهای آسنکرون مدیریت و هدایت خطاست.

فرض کنیم خطایی رخ دهد و OB مزبور در حافظه موجود نباشد. ممکن است PLC متوقف شده و چراغ SF یا INTF/EXTF روشن شود. روشن شدن چراغ‌های فالت می‌تواند دلایل متعدد داشته باشد و کمکی برای رفع سریع عیب نکند. در این شرایط معمولاً وصل شدن به PLC و دیدن اطلاعات موجود در Diagnostic Buffer یا Istack راهی برای شناسایی عیب است که البته مستلزم صرف زمانی است که شاید هر ثانیه‌ی آن از نظر اقتصادی برای کارگاه مهم باشد. با به‌کار بردن OBهای فوق و برنامه‌نویسی مناسب آنها این زمان را می‌توان به حداقل رسانید به‌گونه‌ای که با بروز خطا اپراتور روی سیستم HMI بتواند فالت را همراه با اطلاعات جزئی مشاهده کند. علاوه بر مانیتورینگ فالت می‌توان با برنامه‌نویسی این دسته از OBها با وقوع خطا سیستم را به سمت ایمن و مطمئنی که از قبل در نظر گرفته شده هدایت کرد. از اینرو به این OBها بلاک‌های Error Handling نیز می‌گویند.

برای وقفه‌های آسنکرون تعداد 8 بلاک (OB80-87)درنظرگرفته‌شده است. جدول ۷-۱۱ این OBها را نشان می‌دهد.

جدول ۷-۱۱

Type of error	Example	OB	Priority
Time error	Maximum scan cycle time exceeded	OB80	26
Power supply fault	Backup battery failure	OB81	
Diagnostic interrupt	Wirebreak at input of diagnostics-capable module	OB82	
Insert / remove interrupt	Removal of a signal module during operation of an S7-400™	OB83	
CPU hardware fault	Incorrect signal level at the MPI interface	OB84	
Program execution error	Error in updating the process image (module defective)	OB85	
Rack fault	Failure of an expansion device or a DP slave	OB86	
Communication error	Error in reading message frame	OB87	26 / 28

در ادامه این بخش هرکدام از این OBها جداگانه تشریح می‌شوند.

OB80 (Time Error)

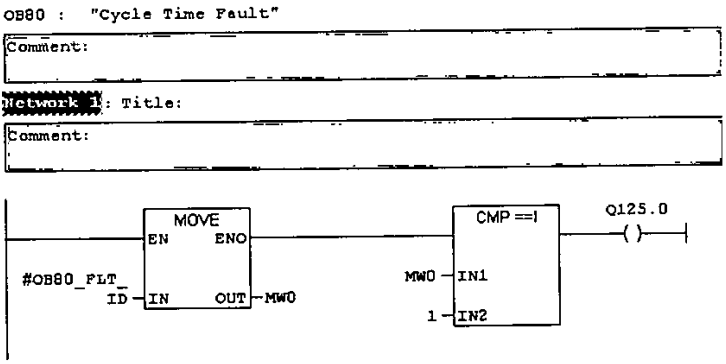
این OB به OB خطای زمانی موسوم می‌باشد. در بخش‌های قبل همین فصل با برخی از موارد کاربرد OB80 آشنا شدید. نمونه‌ای از مواردی که در این فصل به آن اشاره شد می‌توان به پرش از روی زمان تنظیم شده جهت وقفه TOD یا

مثال ۷-۲۰

برنامه‌ای بنویسید که در صورت افزایش زمان سیکل‌اسکن از حد مجاز، خروجی Q125.0 روشن شود.

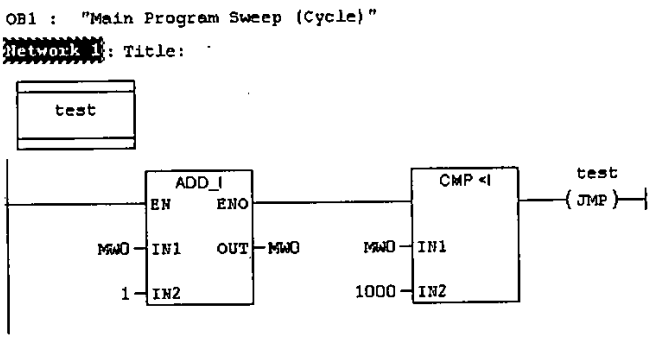
حل: یکی از خطاهایی‌که باعث فراخوانی OB80 می‌شود، افزایش زمان سیکل‌اسکن از حد مجاز است. اما خطاهای دیگری نیز باعث فراخوانی OB80 می‌شوند، بنابراین اگر در این OB فقط دستوری نوشته شود که خروجی Q125.0 روشن شود و به دلیل دیگری مثلاً پرش از روی وقفه TOD این OB (OB80) اجرا شود، خروجی Q125.0 نیز روشن می‌شود؛ و این در حالی است که عملاً خطای افزایش سکل‌اسکن از حد مجاز اتفاق نیافتاده است.

راه حل مناسب استفاده از متغرهای محلی OB80 می‌باشد. همانطور که در جدول متغیرهای محلی OB80 مشاهده می‌شود، متغیر محلی OB80_FLT_ID# یک کد خطا برمی‌گرداند که با توجه به آن می‌توان نوع خطا را متوجه شد. در صورت بروز خطای افزایش زمان سیکل‌اسکن از حد مجاز کد B#16#01 که معادل عدد ۱ در فرم Integer است در متغیرمحلی OB80_FLT_ID# قرارمی‌گیرد. ازاینرو می‌توان برنامه مورد نظر را به‌صورت زیر در OB80 پیاده‌سازی نمود.



شکل ۷-۵۷ برنامه مثال ۷-۲۰

تمرین ۷-۳: برنامه زیر در OB1 نوشته شده است. چرا پس از دانلود ابتدا مشکلی وجود ندارد ولی پس از چند لحظه CPU متوقف می‌گردد؟



شکل ۷-۵۸ برنامه تمرین ۷-۳

خطای پرش از روی وقفه TOD

همانطور که در بخش وقفه TOD اشاره شد، در صورتی‌که زمان CPU به‌گونه‌ای تغییر نماید که از روی زمان اجرای وقفه TOD پرش نماید OB80 اجرا می‌شود. در این شرایط متغیر محلی #OB80_FLT_ID کد W#16#05 را بر می‌گرداند. این کد معادل عدد 5 در فرم Integer است. در صورت عدم وجود OB80 ، CPU متوقف می‌گردد.

خطای فرا خوانی مجدد وقفه سیکلی قبل از پایان اجرای آن

اگر زمان وقفه OB3x کوتاه یا برنامه آن زیاد باشد، ممکن است CPU قبل از اتمام اجرای بلاک مجدداً آن را صدا بزند. در این حالت اگر OB80 موجود نباشد، CPU متوقف می‌گردد و اگر موجود باشد در هر بار که این خطا رخ می‌دهد پیغام Time Error را در بافر می‌نویسد. در این شرایط متغیر محلی #OB80_FLT_ID کد W#16#02 را بر می‌گرداند که می‌توان از آن برای آشکارسازی این نوع فالت استفاده نمود.

(Power Supply Error) OB81

OB81 جهت شناسایی خطای منبع تغذیه و باتری پشتیبان در نظر گرفته شده و خاص S7-400 است. در صورتی‌که خطایی مرتبط با منبع تغذیه رخ دهد این بلاک فراخوانی شده و در صورت وجود از توقف CPU جلوگیری به‌عمل می‌آورد.

متغیرهای محلی OB81

متغیرهای محلی موجود در OB81 کدهای را بر می‌گردانند و با استفاده از آنها در برنامه‌نویسی می‌توان نوع خطای به‌وجود آمده را مشخص نمود. جدول۷-۱۳ متغیرهای محلی موجود در OB81 را نشان می‌دهد.

جدول ۷-۱۳

Variable	Type	Description
OB81_EV_CLASS	BYTE	Event class and identifiers: B#16#38: outgoing event B#16#39: incoming event
OB81_FLT_ID	BYTE	Error code: (possible values) B#16#21, B#16#22, B#16#23, B#16#25, B#16#26, B#16#27, B#16#31, B#16#32, B#16#33)
OB81_PRIORITY	BYTE	Priority class; can be assigned via STEP 7 (hardware configuration) For example, possible values for the RUN mode: 2-26
OB81_OB_NUMBR	BYTE	OB number (81)
OB81_RESERVED_1	BYTE	Reserved
OB81_RESERVED_2	BYTE	Reserved
OB81_RACK_CPU	WORD	• Bits 0 to 7: B#16#00

		• Bits 8 to 15: -For a standard CPU: B#16#00 -For a H-CPU: Bits 8 to 10: Rack no., Bit 11: 0=Reserve CPU, 1=Master CPU, Bits 12 bis 15: 1111
OB81_RESERVED_3	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_4	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_5	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_6	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

از جمله متغیرهای محلی مهم موجود در OB81 می‌توان به موارد زیر اشاره نمود:

- **OB81_EV_CLASS:** توسط این متغیر مشخص می‌شود که آیا خطا به‌وجود آمده است (Incoming) یا خطا برطرف شده است (Outgoing).
- **OB81_FLT_ID:** در این متغیر کدی نمایش داده می‌شود که از روی آن می‌توان متوجه نوع خطای به‌وجود آمده شد. این کدها و معانی آنها در جدول ۷-۱۴ نشان داده شده است.
- **OB81_RESERVED3-6:** در این متغیرها کد نمایش خطای منبع تغذیه یا باتری پشتیبان در رک‌های توسعه نشان داده می‌شود.

جدول ۷-۱۴

OB81_EV_CLASS	OB81_FLT_ID	Meaning
B#16#39/B#16#38	B#16#21:	At least one back-up battery of the central rack is exhausted/problem eliminated (BATTF) Note: This event occurs only if one of the two batteries fails (if there are redundant back-up batteries). If the second battery should also happen to fail, the event will not occur again.
B#16#39/B#16#38	B#16#22:	Back-up voltage in the central rack failed/problem eliminated (BAF)
B#16#39/B#16#38	B#16#23:	Failure of the 24 V power supply in the central rack/problem eliminated.
B#16#39/B#16#38	B#16#25:	At least one back-up battery in at least one redundant central rack is exhausted/problem eliminated (BATTF)
B#16#39/B#16#38	B#16#26:	Back-up voltage in at least one redundant central rack failed/problem eliminated (BAF)
B#16#39/B#16#38	B#16#27:	Failure of the 24 V supply in at least one redundant central rack

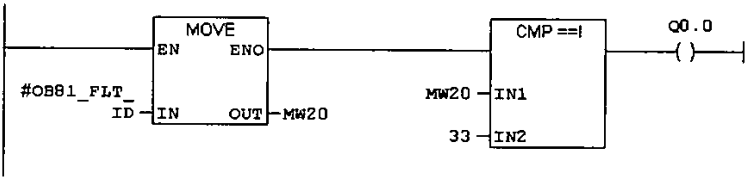
B#16#39/B#16#38	B#16#31:	At least one back-up battery of at least one expansion rack is exhausted/problem eliminated (BATTF).
B#16#39/B#16#38	B#16#32:	Back-up voltage in at least one expansion rack failed/problem eliminated (BAF)
B#16#39/B#16#38	B#16#33:	Failure of the 24 V power supply in at least one expansion rack/problem eliminated.

مثال ۷-۲۱ شناسایی خطای باتری پشتیبان

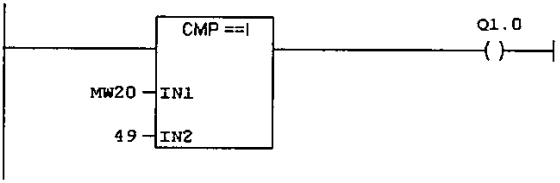
برنامه‌ای بنویسید که در صورت بروز خطا در باتری پشتیبان رک اصلی، خروجی Q0.0 روشن شده و در صورت بروز خطای باتری پشتیبان در رک توسعه خروجی Q1.0 روشن گردد.

حل: با توجه به متغیرهای محلی موجود در OB81 (با توجه به کد متغیر محلی OB81_FLT_ID) برنامه به‌صورت زیر در OB81 نوشته می‌شود. تحلیل برنامه بر عهده خواننده واگذار می‌شود.

```
OB81 :  "Power Supply Fault"
Comment:

Network 1: Title:
Comment:
```




```
Network 2: Title:
Comment:
```



شکل ۷-۵۹ برنامه مثال ۷-۲۱

(Diagnostic Interrupt) OB82

از بین ماژول‌های موجود در S7-300 و S7-400، برخی دارای قابلیت‌های ویژه‌ای می‌باشند. یکی از این قابلیت‌ها، قابلیت Hardware Interrupt است که در بخش‌های قبلی همین فصل تشریح گردید. قابلیت دیگر Diagnostic Interrupt است که فقط در برخی از کارت‌های I/O وجود دارد. در کارت‌هایی‌که دارای این قابلیت هستند امکان شناسایی برخی از خطاها (مانند قطعی تغذیه، قطعی سیم، اتصال کوتاه و . . .) وجود دارد و در صورت بروز این اشکالات ماژول درخواست وقفه Diagnostic را به CPU اعلام می‌کند. در این شرایط CPU، OB82 را فراخوانی و در صورت موجود بودن این OB، آنرا اجرا می‌کند. با برنامه‌نویسی این OB می‌توان آلارم‌های دقیق برای ایرادات مختلف ایجاد کرد. بدیهی است که اگر فالت رخ دهد و OB82 موجود نباشد، منجر به توقف به CPU می‌شود.

برخی از مواردی که در کارت‌های با قابلیت Diagnostic Interrupt قابل شناسایی است عبارتند از:

- قطعی سیم
- قطعی تغذیه ماژول
- اتصال کوتاه
- اتصال زمین
- اشکال در تغذیه سنسورها
- سوختن فیوز کارت خروجی
- تنظیم اشتباه ماژول انتخاب رنج در کارتهای آنالوگ
- خطای Common mode در کارت‌های آنالوگ

تنظیم Diagnostic Interrupt در محیط HW Config

برای تنظیم این وقفه لازم است مراحل زیر طی شود:

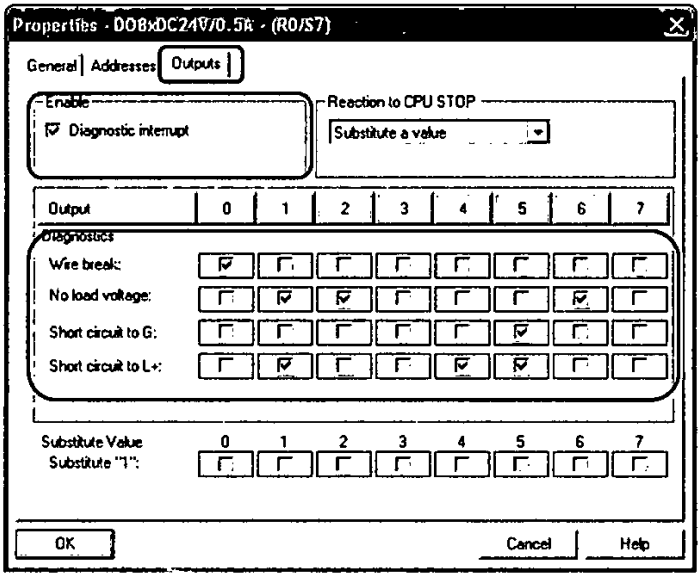
۱- در محیط HW Config روی کارتی که دارای قابلیت Diagnostic Interrupt است، دوبارکلیک نموده و وارد پنجره مشخصات کارت شوید.

۲- در پنجره مشخصات کارت در سربرگ Input (برای کارت‌های ورودی) یا سربرگ Out Put (برای کارت‌های خروجی) وارد شوید.

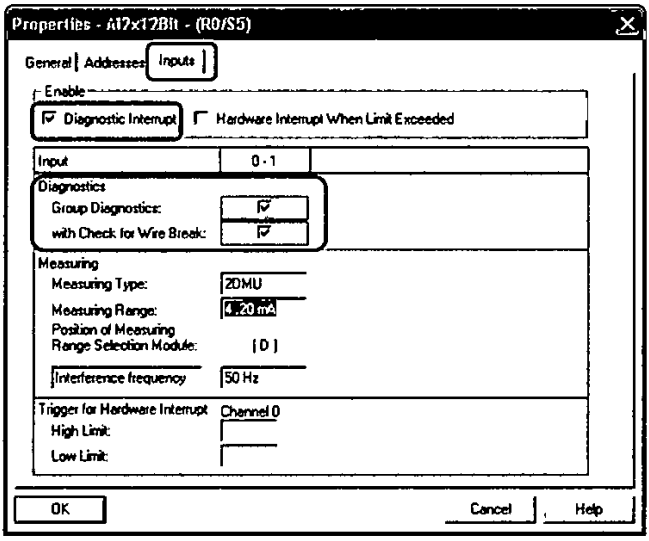
۳- گزینه Diagnostic Interrupt را علامت بزنید.

۴- در قسمت Diagnostics نوع خطایی که باید قابل شناسایی باشد را انتخاب نمایید.

۵- تغییرات انجام شده را ذخیره و دانلود نمایید.



شکل ۷-۶۰ تنظیم وقفه Diagnostic در کارت DO



شکل ۷-۶۱ تنظیم وقفه Diagnostic در کارت AI

متغیرهای محلی OB82

متغیرهای محلی OB82 اطلاعات مهمی درباره خطای پیش آمده، ماژولی که این خطا در آن به‌وجود آمده است و برخی موارد دیگر را در اختیار کاربر قرار می‌دهد. با برنامه‌نویسی مناسب در OB82 می‌توان به‌راحتی بسیاری از خطاها را تشخیص داده و از توقف CPU نیز جلوگیری نمود. جدول ۷-۱۵ متغیرهای محلی OB82 را نشان می‌دهد.

جدول ۷-۱۵

Variable	Type	Description
OB82_EV_CLASS	BYTE	Event class and identifiers: • B#16#38: outgoing event • B#16#39: incoming event
OB82_FLT_ID	BYTE	Error code (B#16#42)
OB82_PRIORITY	BYTE	• Priority class; can be assigned via STEP 7 (hardware configuration)
OB82_OB_NUMBR	BYTE	OB number (82)
OB82_RESERVED_1	BYTE	Reserved
OB82_IO_FLAG	BYTE	Input module: B#16#54 Output module:B#16#55
OB82_MDL_ADDR	WORD	Logical base address of the module where the fault occurred
OB82_MDL_DEFECT	BOOL	Module is defective
OB82_INT_FAULT	BOOL	Internal fault
OB82_EXT_FAULT	BOOL	External fault
OB82_PNT_INFO	BOOL	Channel fault
OB82_EXT_VOLTAGE	BOOL	External voltage failed
OB82_FLD_CONNCTR	BOOL	Front panel connector not plugged in
OB82_NO_CONFIG	BOOL	Module is not configured
OB82_CONFIG_ERR	BOOL	Incorrect parameters on module
OB82_MDL_TYPE	BYTE	• • Bit 0 to 3: Module class Bit 4: Channel information exists • Bit 5: User information exists Bit 6: Diagnostic interrupt from substitute Bit 7: Reserve • •
OB82_SUB_MDL_ERR	BOOL	Submodule is missing or has an error
OB82_COMM_FAULT	BOOL	Communication problem
OB82_MDL_STOP	BOOL	Operating mode (0: RUN, 1: STOP)
OB82_WTCH_DOG_FLT	BOOL	Watchdog timer responded
OB82_INT_PS_FLT	BOOL	Internal power supply failed
OB82_PRIM_BATT_FLT	BOOL	Battery exhausted
OB82_BCKUP_BATT_FLT	BOOL	Entire backup failed

OB82_RESERVED_2	BOOL	Reserved
OB82_RACK_FLT	BOOL	Expansion rack failure
OB82_PROC_FLT	BOOL	Processor failure
OB82_EPROM_FLT	BOOL	EPROM fault
OB82_RAM_FLT	BOOL	RAM fault
OB82_ADU_FLT	BOOL	ADC/DAC error
OB82_FUSE_FLT	BOOL	Fuse tripped
OB82_HW_INTR_FLT	BOOL	Hardware interrupt lost
OB82_RESERVED_3	BOOL	Reserved
OB82_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

برخی از متغیرهای محلی مهم در OB82 عبارتند از:

OB82_IO_FLAG: کدی که در این متغیر نمایش داده می‌شود مشخص می‌کند که ماژولی که خطا در آن به‌وجود آمده است ورودی است یا خروجی:
Input module: B#16#54
Output module: B#16#55

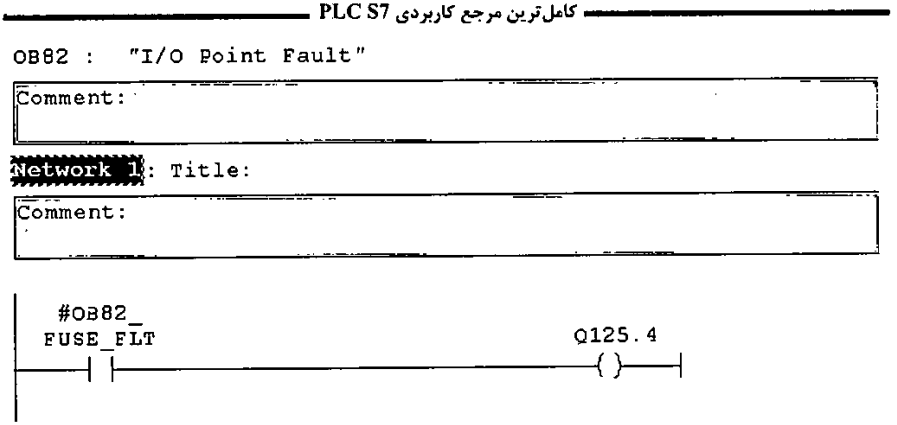
OB82_MDL_ADDR: این متغیر آدرس بیس ماژولی که خطا در آن اتفاق افتاده است را برمی‌گرداند. همانطور که در جدول ۷-۱۵مشاهده می‌شود، متغیرهای بعد از OB82_MDL_ADDR هر کدام نوع خطا را مشخص می‌کنند.

مثال ۷-۲۲

فرض کنید در پیکربندی انجام شده S7-300 از یک کارت خروجی با 8 خروجی دیجیتال که دارای قابلیت Diagnostic Interrupt است استفاده شده است. این کارت می‌تواند سوختن فیوز خروجی خود را تشخیص دهد. برنامه‌ای بنویسید که در صورت سوختن فیوز این کارت، خروجی Q125.4 روشن شود.

حل: همانطور که در جدول ۷-۱۵ مشخص است، متغیر محلی #OB82_FUSE_FLT که از نوع BOOL می‌باشد سوختن فیوز را مشخص می‌کند. با استفاده از این متغیر می‌توان به‌طور کلی مشخص نمود که فیوز یکی از کارت‌های خروجی سوخته است. اما اینکه دقیقاً کدام کارت دچار مشکل شده است، مشخص نیست. در مثال‌های بعدی آدرس کارتی که دچار اشکال شده است را نیز مشخص می‌کنیم.

برنامه مورد نظر جهت این مثال به‌صورت زیر است.

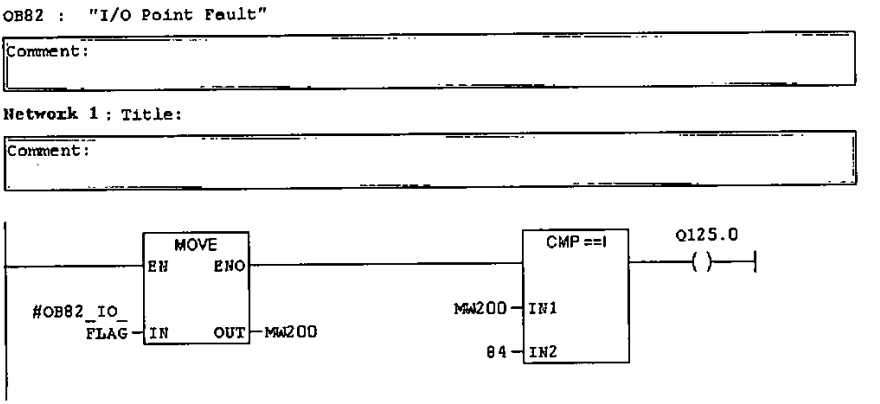


شکل ۷-۶۲ برنامه مثال ۷-۲۲

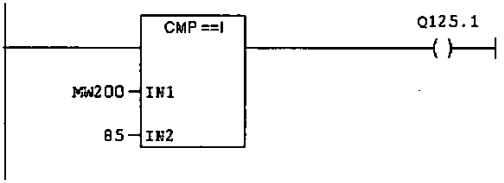
مثال ۷-۲۳

در مثال قبل مشخص کنید که اشکال به‌وجود آمده در کارت ورودی است یا خروجی؟ اگر خطا در کارت ورودی است Q125.0 و اگر خطا در کارت خروجی است Q125.1 روشن شود.

حل: با استفاده از متغیر محلی OB82_IO_FLAG می‌توان برنامه را به‌صورت زیر نوشت.



```
Network 2: Title:
Comment:
```



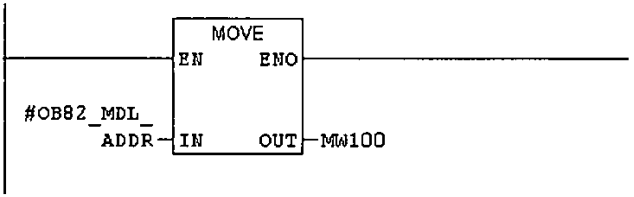
شکل ۷-۶۳ برنامه مثال ۷-۲۳

مثال ۷-۲۴

در مثال قبل برنامه را به‌گونه‌ای طراحی نمایید تا آدرس کارتی که دچار اشکال شده است در MW100 نماش داده شود.

حل: با استفاده از متغیر محلی OB82_MDL_ADDR می‌توان برنامه را به‌صورت زیر نوشت.

```
OB82 :  "I/O Point Fault"
Comment:
Network 1: Title:
Comment:
```



شکل ۷-۶۴ برنامه مثال ۷-۲۴

مثال ۷-۲۵

در مثال قبل فرض شود سه کارت DO با قابلیت Diagnostic Interrupt استفاده شده است. آدرس بیس این کارت‌ها عبارتست از:

کارت اول: آدرس بیس 0

کارت دوم: آدرس بیس 4

کارت سوم: آدرس بیس 8

برنامه‌ای بنویسید که اگر فیوز هر کدام از کارت‌ها بسوزد به‌صورت زیر آلارم‌هایی روشن شوند:

کارت اول: Q124.0

کارت دوم: Q124.1

کارت سوم: Q124.2

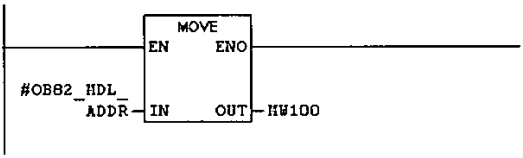
حل: این برنامه در واقع ترکیبی از برنامه مثال‌های قبل می‌باشد.

OB82 : "I/O Point Fault"

Comment:

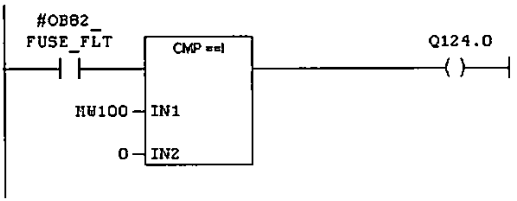
Network 1 : Title:

Comment:



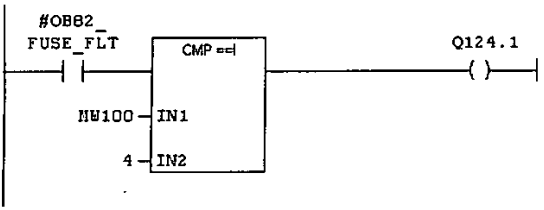
Network 2 : Title:

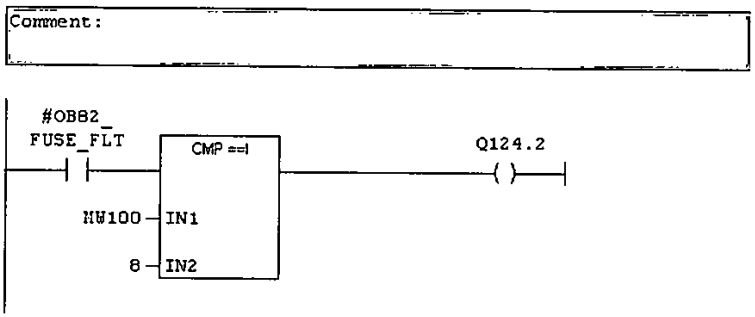
Comment:



Network 3 : Title:

Comment:





شکل ۷-۶۵ برنامه مثال ۷-۲۵

(Insert / Remove Module Interrupt) OB 83

این وقفه که در S7-400 کاربرد دارد، در موارد زیر توسط سیستم عامل CPU فراخوانی می‌گردد:

- قراردادن یا برداشتن یک ماژول در حین کار CPU
- پس از اصلاح پارامترهای یک ماژول در نرم‌افزار Step7 و دانلود تغییرات به CPU در مد RUN

در موارد فوق عدم وجود OB83 باعث توقف CPU می‌گردد.

قراردادن و برداشتن ماژول

هرگاه یک ماژول در حین کار CPU از روی رک برداشته شده یا ماژولی در رک قرار داده شود، این وقفه فراخوانی می‌شود.

نکته: CPU، PS و IM را نمی‌توان در مد RUN تعویض یا جابه‌جا نمود.

برای اینکه CPU بتواند قراردادن و برداشتن ماژول از روی رک را تشخیص دهد، نیاز به فاصله زمانی ۲ ثانیه بین برداشتن و قراردادن ماژول دارد. اگر ماژول در حین کار CPU از روی رک برداشته شود، قاعدتاً باید OB83 فراخوانی شود. اما از آنجایی که قراردادن یا برداشتن ماژول روی رک در فواصل یک‌ثانیه‌ای بررسی می‌شود، در صورتی‌که در برنامه مستقیماً از آدرس‌های فوق استفاده شده باشد، خطای دسترسی به ورودی/خروجی تشخیص داده شده و وقفه متناسب با آن فراخوانی می‌شود.

نکته: در S7-300 و در شرایطی‌که از Remote I/O استفاده شود، CPU 318 رفتاری مانند CPUهای S7-400 داشته و امکان قراردادن یا برداشتن ماژول وجود دارد. همچنین CPUهایی که پورت مربوط به شبکه پروفی‌نت دارند می‌توانند این قابلیت را در شبکه پروفی‌نت پشتیبانی نمایند.

متغیرهای محلی OB83

متغیرهای محلی OB83 اطلاعات خوبی در اختیار کاربر قرار می دهند. از جمله اینکه ماژول مورد نظر چه نوع ماژولی است، آیا عمل قراردادن ماژول انجام شده یا برداشتن و . . . . جدول ۷-۱۶ متغیرهای محلی OB83 را نشان می‌دهد.

جدول ۷-۱۶

Variable	Type	Description
OB83_EV_CLASS	BYTE	Event class and identifiers: • B#16#32: End of reassignment of module parameters • B#16#33: Start of reassignment of module parameters • B#16#38: module inserted • B#16#39: module removed or not responding, or end of parameter assignment
OB83_FLT_ID	BYTE	Error code: (possible values B#16#51, B#16#54, B#16#55, B#16#56, B#16#58, B#16#61, B#16#63, B#16#64, B#16#65, B#16#66, B#16#67, B#16#68, B#16#84)
OB83_PRIORITY	BYTE	• Priority class; can be assigned via STEP 7 (hardware configuration)
OB83_OB_NUMBR	BYTE	OB number (83)
OB83_RESERVED_1	BYTE	Identification of module or submodule/interface module
OB83_MDL_TD	BYTE	Range: • B#16#54: Peripheral input (PI) • B#16#55: Peripheral output (PQ)
OB83_MDL_ADDR	WORD	• Central or distributed PROFIBUS DP: Logical base address of the module affected. If it is a mixed module, it is the smallest logical address used in the module. If the I and O addresses in the mixed block are equal, the logical base address is the one that receives the event identifier. Distributed PROFINET IO: Logical base address of the module/submodule
OB83_RACK_NUM	WORD	• If OB83_RESERVED_1 =B#16#A0: number of submodule/interface submodule (low byte) • If OB83_RESERVED_1 = B#16#C4: -central: rack number -distributed PROFIBUS DP: number of DP station (low byte) and DP master system ID (high byte) -distributed PROFINET IO: physical address: identifier bit (bit 15, 1 = PROFINET IO), IO system ID (bits 11 to 14) and device number (bits 0 to 10)
OB83_MDL_TYPE	WORD	• Central or distributed PROFIBUS DP: Module type of affected module (X: irrelevant to the user): -W#16#X5XX: analog module -W#16#X8XX: function module -W#16#XCXX: CP -W#16#XFXX: digital module • Distributed PROFINET IO -W#16#8101: module type of the inserted module is the same as the module type of the removed module

		-W#16#8102: module type of the inserted module is not the same as the module type of the removed module
OB83_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

جدول ۷-۱۷ کدهای مربوط به متغیرهای OB83_EV_CLASS و OB83_FLT_ID که بیانگر دلیل فراخوانی OB83 است را نشان می‌دهد.

جدول ۷-۱۷

OB83_EV_CLASS	OB83_FLT_ID	Meaning
B#16#39	B#16#51	PROFINET IO module removed
B#16#39	B#16#54	PROFINET IO submodule removed
B#16#38	B#16#54	PROFINET IO submodule inserted and matches configured submodule
B#16#38	B#16#55	PROFINET IO submodule inserted, but does not match configured submodule
B#16#38	B#16#56	PROFINET IO submodule inserted, but error with module parameters
B#16#38	B#16#58	PROFINET IO submodule, access error corrected
B#16#39	B#16#61	Module removed or not responding OB83_MDL_TYPE: Actual module type
B#16#38	B#16#61	Module inserted. Module type OK OB83_MDL_TYPE: Actual module type
B#16#38	B#16#63	Module inserted but incorrect module type OB83_MDL_TYPE: Actual module type
B#16#38	B#16#64	Module inserted but problem (module ID cannot be read) OB83_MDL_TYPE: Configured module type
B#16#38	B#16#65	Module inserted but error in module parameter assignment OB83_MDL_TYPE: Actual module type
B#16#39	B#16#66	Module not responding, load voltage error
B#16#38	B#16#66	Module responds again, load voltage error corrected
B#16#33	B#16#67	Start of module reconfiguration
B#16#32	B#16#67	End of module reconfiguration
B#16#39	B#16#68	Module reconfiguration terminated with error
B#16#38	B#16#84	Interface module inserted
B#16#39	B#16#84	Interface module removed

به‌عنوان نمونه:

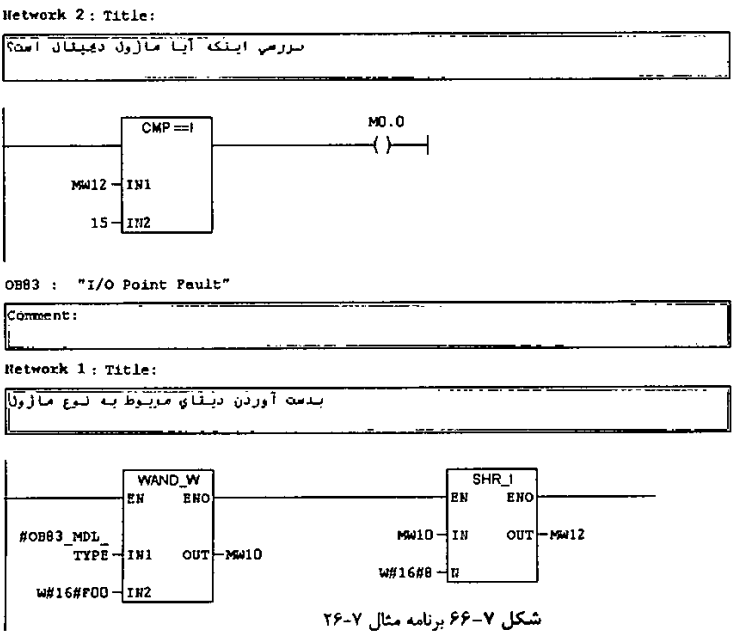
- اگر OB83_FLT_ID مقدار 61 و OB83_EV_CLASS مقدار 39 را برگرداند، یعنی کارت برداشته شده یا پاسخ نمی‌دهد.
- اگر OB83_FLT_ID مقدار 61 و OB83_EV_CLASS مقدار 38 را برگرداند، یعنی کارت گذاشته شده و نوع آن درست است.
- اگر OB83_FLT_ID مقدار 63 را برگرداند، یعنی کارت گذاشته شده و نوع آن اشتباه است.
- اگر OB83_FLT_ID مقدار 64 را برگرداند، یعنی کارت گذاشته شده ولی اشکال دارد.
- اگر OB83_FLT_ID مقدار 65 را برگرداند، یعنی کارت گذاشته شده ولی در پارامترهای اختصاص داده شده به آن مشکل وجود دارد.

مثال ۷-۲۶

برنامه‌ای بنویسید که در صورت قراردادن یک ماژول دیجیتال در رک و OK بودن وضعیت و پارامترهای آن، خروجی Q124.0 روشن شود.

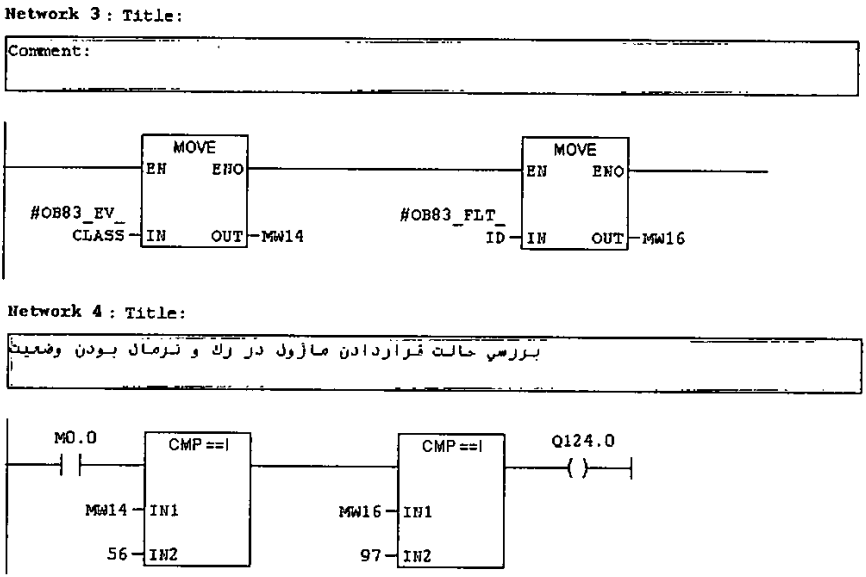
حل: با توجه به جدول‌های۷-۱۶ و ۷-۱۷ ، با استفاده از متغیر محلی OB83_MDL_TYPE می‌توان تشخیص داد که آیا ماژول مورد نظر ماژول دیجیتال است یا خیر. توسط متغیر محلی OB83_EV_CLASS و OB83_FLT_ID می‌توان متوجه شد که آیا ماژول به‌صورت صحیح در رک قرارداده شده است یا خیر؟

بنابراین برنامه مورد نظر به‌صورت زیر در OB83 پیاده‌سازی می‌شود.



شکل ۷-۶۶ برنامه مثال ۷-۲۶

در Network1، برنامه به‌گونه‌ای طراحی شده است که چهار بیت کم ارزش از بایت اول، متغیر محلی OB83_MDL_TYPE را از این متغیر جدا نموده و پس از 8 شیفت به راست در خانه MW12 ذخیره می‌نماید. بدین ترتیب می‌توان عدد موجود در MW12 را با کدی که در چهار بیت کم‌ارزش از متغیر محلی OB83_MDL_TYPE قرار می‌گیرد مقایسه نموده و در صورتی‌که ماژول دیجیتال باشد و این متغیر کد F را به‌صورت HEX برگرداند (معادل ۱۵ در فرم Integer)، حافظه M0.0 یک شود.



شکل ۷-۶۷ ادامه برنامه مثال ۷-۲۶

در Networkهای 3 و 4 وضعیت OK با توجه به کدهایی که در متغیرهای OB83_EV_CLASS و OB83_FLT_ID قرار می‌گیرد بررسی می‌شود.

(CPU Hardware Fault) OB84

این وقفه در صورت بروز برخی اشکالات در سیستم‌های خاص مانند S7-400H و WinAC فعال می‌شود که در این کتاب مورد بحث قرار می‌گیرند. تنها موردی که این وقفه برای سیستم‌های معمولی ممکن است فعال شود حالتی است که اشکالی در حافظه رخ دهد ولی ایراد برطرف گردد.

(Priority Class Error) OB85

OB85 که به آن Program Sequence Error نیز گفته می‌شود، یکی از OBهای خطای غیرهمزمان است که در شرایط زیر فراخوانی می‌گردد:

- فراخوانی یک OB وقفه و عدم وجود OB مربوطه (به استثنای OB80، OB81، OB82، OB83 و OB86)
- در شرایط بروز خطای عدم دسترسی به دیتابلاک اختصاصی یک SFB
- خطا در دسترسی سیستم عامل به یک ماژول
- در شرایط بروز خطا در Update شدن ناحیه Process Image

متغیرهای محلی OB85

جدول ۷-۱۸ متغیرهای محلی موجود در OB85 را نشان می‌دهد.

جدول ۷-۱۸

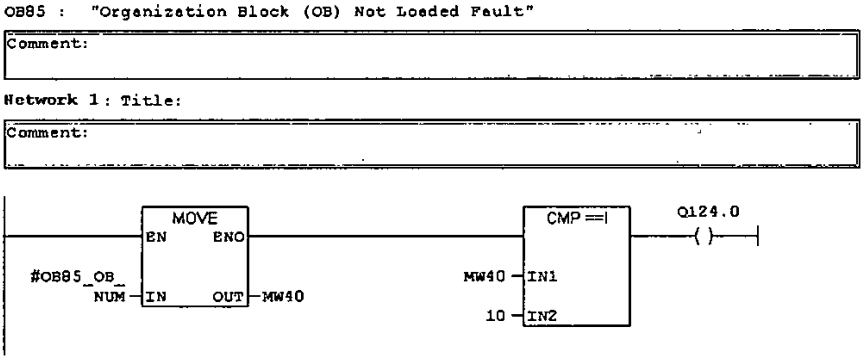
Variable	Type	Description
OB85_EV_CLASS	BYTE	Event class and identifiers: B#16#35 B#16#38 (only with error codes B#16#B3 and B#16#B4) B#16#39 (only with error codes B#16#B1, B#16#B2, B#16#B3 and B#16#B4)
OB85_FLT_ID	BYTE	Error code (possible values: B#16#A1, B#16#A2, B#16#A3, B#16#A4, B#16#B1, B#16#B2, B#16#B3, B#16#B4))
OB85_PRIORITY	BYTE	• Priority class; can be assigned via STEP 7 (hardware configuration)
OB85_OB_NUMBR	BYTE	OB number (85)
OB85_RESERVED_1	BYTE	Reserved
OB85_RESERVED_2	BYTE	Reserved
OB85_RESERVED_3	INT	Reserved
OB85_ERR_EV_CLASS	BYTE	Class of the event that caused the error
OB85_ERR_EV_NUM	BYTE	Number of the event that caused the error
OB85_OB_PRIOR	BYTE	Priority class of the OB that was active when the error occurred
OB85_OB_NUM	BYTE	Number of the OB that was active when the error occurred
OB85_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

متغیر محلی OB85_FLT_ID کدهای خطا را بر می‌گرداند که برخی از آنها عبارتند از:

- اگر OB85_FLT_ID مقدار A1 را برگرداند، یعنی OB خاصی صدا زده شده که در حافظه موجود نیست.
- اگر OB85_FLT_ID مقدار A3 را برگرداند، یعنی سیستم عامل در هنگام دسترسی به کارت‌ها با خطا مواجه شده است.
- اگر OB85_FLT_ID مقدار B1 را برگرداند، یعنی Update شدن PII مشکل دارد.
- اگر OB85_FLT_ID مقدار B2 را برگرداند، یعنی Update شدن PIQ مشکل دارد.

مثال ۷-۲۷

برنامه‌ای بنویسید که در صورت فعال شدن وقفه TOD و عدم دانلود OB10 مربوطه، خروجی Q124.0 روشن شود.



شکل ۷-۶۸ برنامه مثال ۷-۲۷

(Rack Failure) OB86

OB86 یکی از OBهای مهم در برنامه‌نویسی PLC S7 محسوب می‌گردد. مواردی که این OB توسط سیستم عامل فراخوانی می‌شود عبارتند از:

- وجود خطا در رک توسعه (فقط در S7-400) نظیر قطعی کابل اتصال رک توسعه به اصلی یا خطا در منبع تغذیه رک توسعه
- خطا در شبکه پروفی‌باس مانند قطعی کابل شبکه، عدم تنظیم صحیح Dip Switch مربوط به ET200 در شبکه پروفی‌باس و ...

یکی از کاربردهای مهم OB86 شناسایی خطا در شبکه پروفی‌باس می‌باشد. مثلاً اگر یک Node از دسترس خارج شود با استفاده از این OB می‌توان شماره Node مورد نظر را مشخص نمود. کاربرد دیگر OB86 تشخیص خطا در رک توسعه می‌باشد که فقط در S7-400 امکان‌پذیر است.

این OB در کتاب شبکه صنعتی پروفی‌باس بطور کامل تشریح گردیده است. در اینجا صرفاً به مثالی که کاربرد آنرا برای رک توسعه نشان می‌دهد اکتفا می‌کنیم. اشکال ارتباطی بین رک اصلی و رک‌های توسعه در صورت عدم وجود OB86 منجر به توقف CPU می‌گردد.

در اینجا نیز خطا چه در هنگام وقوع (incoming) و چه در صورت برطرف شدن (outgoing) قابل آشکارسازی است. متغیرهای Temp در اینجا نیز کاربرد دارند.

OB86_EV_CLASS برای Incoming مقدار ۳۹ و برای Outgoing مقدار ۳۸ را برمی‌گرداند.

برای خطای رک

- OB86_FLT_ID مقدار C1 را برمی‌گرداند.
- OB86_MDL_ADDR آدرس بیس کارت IM را نشان می‌دهد.
- OB86_Z23 که یک DWord است با بیت‌های خود شماره رک را مشخص می‌کند.

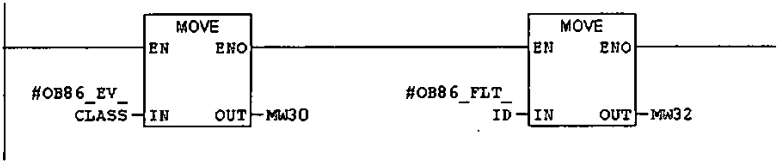
بیت 0	همیشه صفر است
بیت ۱	وقتی 1 شود، یعنی خطا در رک شماره 1
بیت 2	وقتی 1 شود، یعنی خطا در رک شماره 2
.....	....
بیت 21	وقتی 1 شود، یعنی خطا در رک شماره 21

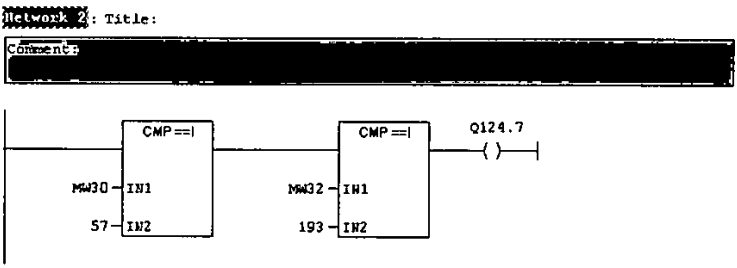
مثال ۷- ۲۸

برنامه‌ای بنویسید که در صورت اشکال در رک توسعه، خروجی Q124.7 روشن شود.

حل: با توجه به متغیرهای محلی موجود در OB86 برنامه به‌صورت زیر در این OB نوشته می‌شود.

```
OB86 :  "Loss Of Rack Fault"
Comment:

Network 1 : Title:
Comment:
```





شکل ۷-۶۹ برنامه مثال ۷-۲۸

فصل ۷

(Communication Error) OB87

این OB در شرایط بروز اشکال در شبکه‌های MPI که برای تبادل دیتا بین چند PLC از طریق Global Data Communication (تبادل دیتای سراسری) پیاده‌سازی می‌شود کاربرد دارد. تشریح آن در کتاب‌های جداگانه آمده است.

(Processing Interrupt) OB88

این OB در شرایطی مانند موارد زیر که پردازش CPU دچار مشکل می‌شود فراخوان می‌گردد. بدیهی است که عدم وجود این OB منجر به توقف CPU در شرایط بروز فالت خواهد شد.

۱- عمق تودرتویی Nesting Depth بیش از حد باشد. به‌عنوان مثال FC و FBهای زیادی به‌صورت تودرتو بیش از حد مجاز فراخوان شده‌اند.

۲- در مواردی که متغیرهای Local Data توسط کاربر تغییر داده و دانلود شوند، ولی CPU نتواند آنها را طبق تغییرات انجام شده اختصاص دهد OB88 فراخوان می‌شود. بحث Local Data در فصل ۴ تشریح شده است.

۷-۱۱ وقفه‌های OB12x (Synchronous Errors)

همانطور که در ابتدای این فصل اشاره شد، خطاهای همزمان خطاهایی هستند که عمدتاً به دلیل اشکالات برنامه‌نویسی ایجاد می‌گردند. جدول ۷-۱۹ OBهای مربوط به Synchronous Error را نشان می‌دهد.

جدول ۷-۱۹

Type of error	Example	OB	Priority
Programming error	A block that is not present in the CPU is called in the program	OB121	Same as that of the OB interrupted as a result of the error
Access error	A module that is either defective or not present is addressed in the program (such as direct access to a non-existent I/O module)	OB122	

(Programming Error) OB121

این OB مربوط به خطاهای برنامه‌نویسی بوده و در موارد زیر فراخوانی می‌شود:

- فراخوانی FC یا FB که دانلود نشده است
- استفاده از DB که دانلود نشده است
- استفاده از آدرس نامعتبر در DB
- استفاده از شماره نامعتبر برای تایمر یا کانتر در برنامه‌نویسی

با ایجاد و برنامه‌نویسی صحیح این بلاک می‌توان از توقف CPU جلوگیری نمود. این بلاک از بلاک‌های پرکاربرد در مرحله تست و راه‌اندازی سیستم و حتی در سایر مراحل می‌باشد.

متغیرهای محلی OB121

اطلاعات مختلفی توسط متغیرهای محلی برگردانده می‌شود. جدول ۷-۲۰ اطلاعاتی که توسط متغیر محلی OB121_SW_FLT برمی‌گردد را نشان داده و برای هر حالت، وضعیت متغیر محلی OB121_FLT_REG مشخص شده است.

جدول ۷-۲۰

Error code	Meaning
B#16#21:	BCD conversion error
OB121_FLT_REG:	ID for the register concerned (W#16#0000: accumulator 1)
B#16#22:	Area length error when reading
B#16#23:	Area length error when writing
B#16#28:	Read access to a byte, word, or double word with a pointer whose bit address is not 0.
B#16#29:	Write access to a byte, word, or double word with a pointer whose bit address is not 0.
	Incorrect byte address. The data area and access type can be read from OB121_RESERVED_1.
OB121_RESERVED_1:	• Bits 7 to 4 access type. -0: bit access,
	-1: byte access,
	-2: word access,
	-3: double word access
	• Bits 3 to 0 memory area: -0: I/O area
	-1: process-image input table
	-2: Process-image output table
	-3: bit memory
	-4: global DB

	-5: instance DB -6: own local data -7: local data of caller
B#16#24: B#16#25: OB121_FLT_REG:	Range error when reading Range error when writing Contains the ID of the illegal area in the low byte (B#16#86 of own local data area)
B#16#26: B#16#27: OB121_FLT_REG:	Error for timer number Error for counter number Illegal number
B#16#30: B#16#31: B#16#32: B#16#33: OB121_FLT_REG:	Write access to a write-protected global DB Write access to a write-protected instance DB DB number error accessing a global DB DB number error accessing an instance DB Illegal DB number
B#16#34: B#16#35: B#16#3A: OB121_FLT_REG: B#16#3C: OB121_FLT_REG: B#16#3D: OB121_FLT_REG: B#16#3E: OB121_FLT_REG: B#16#3F: OB121_FLT_REG:	FC number error in FC call FB number error in FB call Access to a DB that has not been loaded; the DB number is in the permitted range DB number Access to an FC that has not been loaded; the FC number is in the permitted range FC number Access to an SFC that is not available; the SFC number is in the permitted range SFC number Access to an FB that has not been loaded; the FB number is in the permitted range FB number Access to an SFB that is not available; the SFB number is in the permitted range SFB number

به‌عنوان نمونه به نمونه‌ای از کدهای OB121_SW_FLT که در جدول ۷-۲۱ آمده توجه کنید.

جدول ۷-۲۱

کد خطا	نوع خطا
21	خطا در تبدیل BCD
26	غلط بودن شماره تایمر
27	غلط بودن شماره کانتر
30	خطای نوشتن روی DB اشترکی Write Protect شده

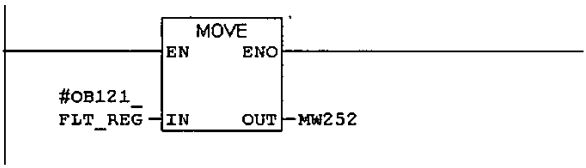
31	خطای نوشتن روی DB اختصاصی Write Protect شده
32	غلط بودن شماره DB اشتراکی
33	غلط بودن شماره DB اختصاصی
34	اشکال در شماره FC در هنگام فراخوانی
35	اشکال در شماره FB در هنگام فراخوانی
3A	دسترسی به DB که به حافظه دانلود نشده
3C	دسترسی به FC که به حافظه دانلود نشده
3D	دسترسی به SFC که به حافظه دانلود نشده
3E	دسترسی به FB که به حافظه دانلود نشده
3F	دسترسی به SFB که به حافظه دانلود نشده

مثال ۷-۲۹

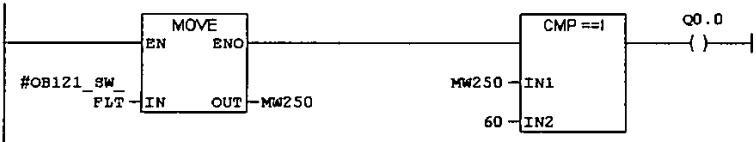
برنامه‌ای بنویسید که در صورت فراخوانی یک FC و عدم دانلود آن، خروجی Q0.0 روشن شده و شماره FC مذکور در MW252 نمایش داده شود.

حل: با توجه به متغیرهای محلی OB121 ، می‌توان برنامه را به‌صورت زیر ارائه نمود.

```
Network 2 : Title:
Comment:
```




```
OB121 :  "Programming Error"
Comment:
Network 1 : Title:
Comment:
```



شکل ۷-۷۰ برنامه مثال ۷-۲۹

(I/O Access Error) OB122

هنگامی‌که CPU در هنگام خواندن یا نوشتن دیتا در یک آدرس مرتبط با ماژول دچار خطا شود، OB122 فراخوانی می‌گردد. مثلاً ممکن است در ضمن برنامه‌نویسی، کاربر از یک آدرس Peripheral نامعتبر استفاده نماید. در اینصورت هنگامی که CPU به آن دستور برسد OB122 را فراخوانی می‌کند. عدم وجود OB122 در این شرایط منجر به توقف CPU می‌گردد.

متغیرهای محلی موجود در OB122

جدول ۷-۲۲ متغیرهای محلی موجود در OB122 را نشان می‌دهد.

جدول ۷-۲۲

Variable	Type	Description
OB122_EV_CLASS	BYTE	Event class and identifiers: B#16#29
OB122_SW_FLT	BYTE	Error code: • B#16#42: I/O access error, reading • B#16#43: I/O access error, writing
OB122_PRIORITY	BYTE	Priority class: • Priority class of the OB where the error occurred
OB122_OB_NUMBR	BYTE	OB number (122)
OB122_BLK_TYPE	BYTE	Type of block where the error occurred (B#16#88: OB, B#16#8C: FC, B#16#8E: FB) (no valid number is entered here for an S7-300)
OB122_MEM_AREA	BYTE	Memory area and access type: • Bit 7 to 4: Access type -0: Bit access -1: Byte access -2: Word access -3: DWord access • Bit 3 to 0: memory area -0: I/O area -1: Process image of the inputs -2: Process image of the outputs
OB122_MEM_ADDR	WORD	Memory address where the error occurred
OB122_BLK_NUM	WORD	Number of the block with the MC7 command that caused the error (no valid number is entered here for an S7-300)
OB122_PRG_ADDR	WORD	Relative address of the MC7 command that caused the error (no valid number is entered here for an S7-300)
OB122_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

مثال ۷-۳۰

برنامه‌ای بنویسید که در ضمن اجرای برنامه‌ی CPU اگر یک خطای I/O Access در خواندن دیتا به‌وجود آمد، خروجی Q0.1 روشن شده و شماره بلاک در MW82 نمایش داده شود. همچنین نوع بلاک به شرح زیر مشخص شود:

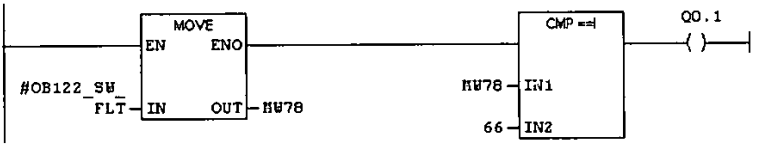
OB: M0.0= 1

FC: M0.1= 1

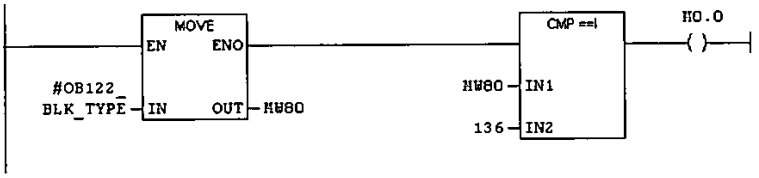
FB: M0.2= 1

حل: با استفاده از متغیرهای محلی موجود در OB122 داریم:

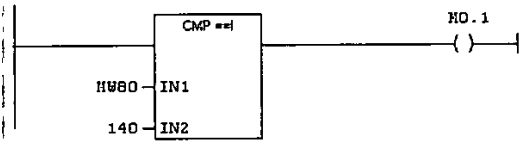
```
OB122 :  "Module Access Error"
Comment:

Network 1 : Title:
Comment:
```

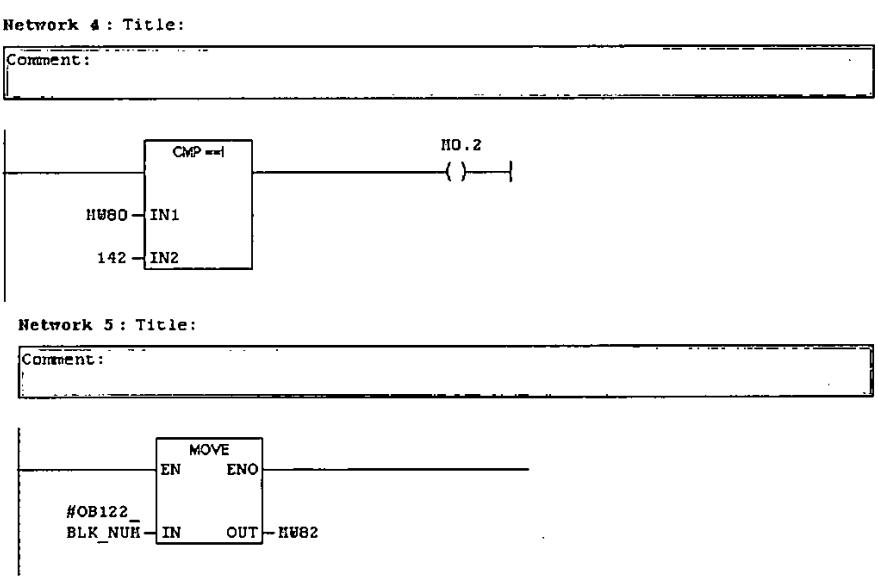



```
Network 2 : Title:
Comment:
```




```
Network 3 : Title:
Comment:
```





شکل ۷-۷۱ برنامه مثال ۷-۳۰

۷-۱۲ آشنایی با سایر وقفه‌ها

به‌جز وقفه‌هایی که تا اینجا مورد بحث قرار گرفت، وقفه‌های دیگری نیز در برخی CPUها وجود دارند که دارای کاربرد خاص می‌باشند. این وقفه‌ها را می‌توان به دسته‌بندی زیر تقسیم نمود:

۱- وقفه‌های OB5x

۲- وقفه‌های OB6x

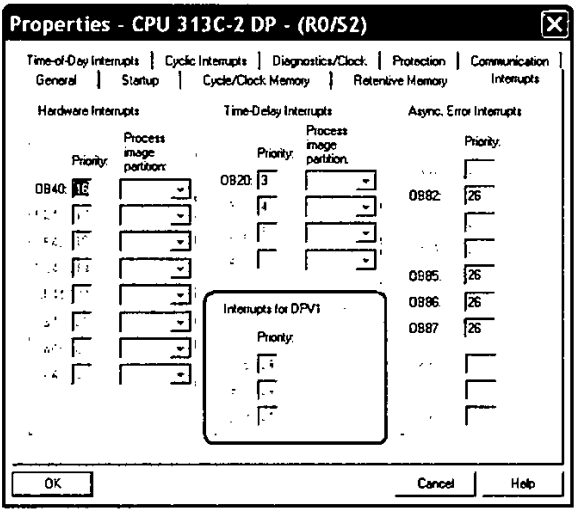
۳- وقفه‌های OB7x

این وقفه‌ها در ادامه معرفی شده‌اند.

۷-۱۲-۱ وقفه‌های OB5x

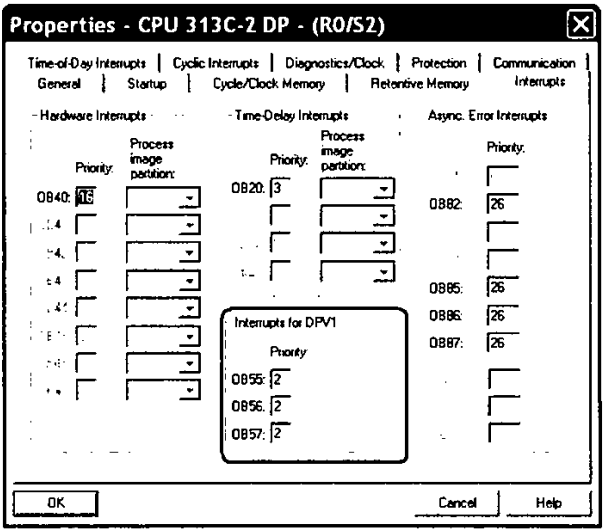
این وقفه‌ها که به خانواده وقفه‌های DP V1 موسوم هستند، همانطور که از نامشان پیداست مربوط به شبکه Profibus DP می‌باشند و برای CPUهایی که Version 1 پروفی‌باس را ساپورت می‌کنند قابل استفاده می‌باشند. بحث کامل پروفی‌باس در کتاب جداگانه‌ای از مؤلف آورده شده است. در اینجا صرفاً در حد آشنایی به انواع OBهای این خانواده اشاره می‌شود.

شکل ۷-۷۲ بخش وقفه‌ها را در پارامترهای CPU 313C-2DP را با شماره سفارش 6ES7 313-6CE00-0AB0 نشان می‌دهد. همانطور که در این شکل دیده می‌شود، CPU فوق این وقفه‌ها را ساپورت نمی‌کند.



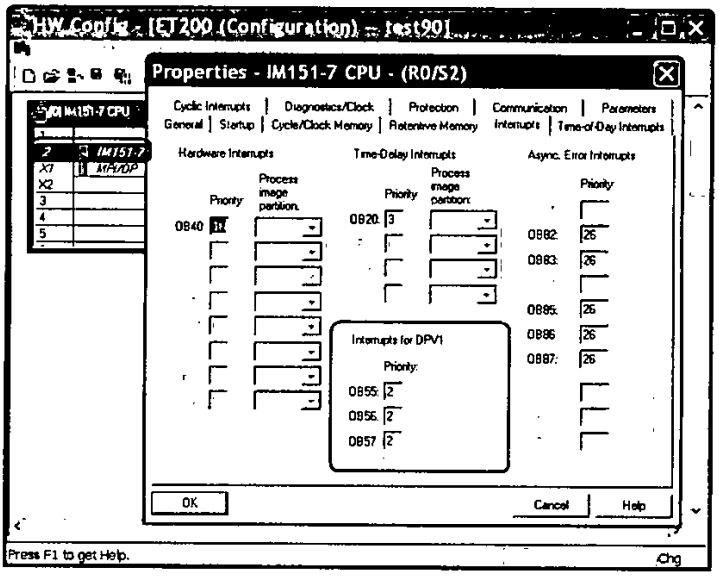
شکل ۷-۷۲ وضعیت وقفه‌های OB5x در CPU313C قدیمی

شکل ۷-۷۳ همان بخش را برای مدل جدیدتری از CPU 313C-2DP با شماره سفارش 6ES7313-6CF03-0AB0 نشان می‌دهد. همانطور که در این شکل دیده می‌شود، CPU فوق این وقفه‌ها را ساپورت می‌کند.



شکل ۷-۷۳ وضعیت وقفه‌های OB5x در CPU313C جدید

وسیله Slave ممکن است یک Remote I/O باشد. به‌عنوان مثال ET200های مجهز به CPU این وقفه‌ها را ساپورت می‌کنند. شکل ۷-۷۴ پارامترهای ET200S مدل IM151-7 را که مجهز به CPU است نشان می‌دهد.



شکل ۷-۷۴ وضعیت وقفه‌های OB5x در ET200S مجهز به CPU

همانطور که در شکل‌های قبل دیده می‌شود، وقفه‌های این خانواده دارای درجه اولویت ۲ بوده و شامل موارد زیر هستند:

- **وقفه OB55 موسوم به Status Interrupt**

وقتی که مد کاری وسیله Slave متصل به Profibus DP V1 تغییر کند، این وقفه فعال می‌شود. به‌عنوان مثال وقتی که وضعیت وسیله از Stop به Run یا برعکس عوض می‌شود این وقفه تریگر می‌شود.

- **وقفه OB56 موسوم به Update Interrupt**

وقتی که پارامترهای تنظیمی وسیله Slave متصل به Profibus DP V1 تغییر کند این وقفه فعال می‌شود.

- **وقفه OB57 موسوم به Manufacturer specific Interrupt**

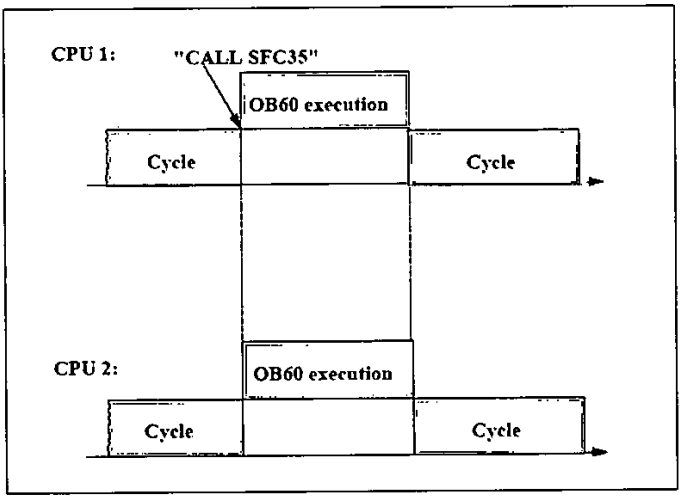
این وقفه همانطور که از نامش پیداست، به‌صورت خاص و مربوط به سازنده وسیله Slave است که به Profibus DP V1 متصل می‌باشد.

۷-۱۲-۲ وقفه‌های OB6x

این وقفه‌ها به سه دسته اصلی تقسیم می‌شوند که هر کدام کاربرد خاص خود را دارند.

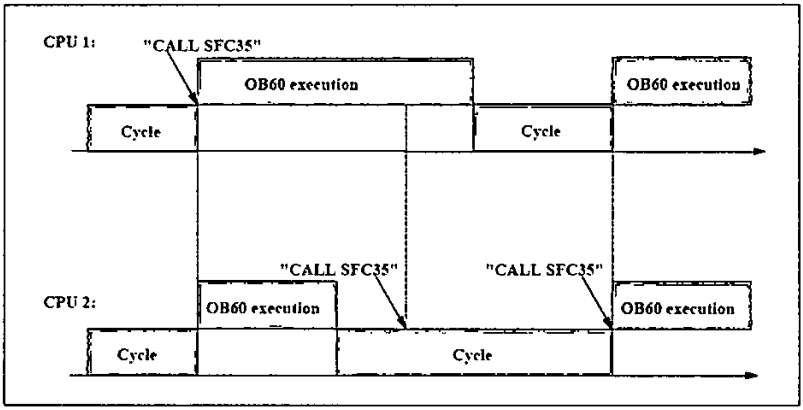
- **وقفه OB60 موسوم به Multicomputing Interrupt**

این وقفه همانطور که از نامش پیداست، خاص سیستم‌های multicomputing است. این سیستم‌ها که توسط S7-400 پیاده‌سازی می‌شوند دارای چند CPU روی یک رک هستند که برنامه‌های متفاوتی را اجرا می‌کنند. اگر لازم باشد که همه این CPUها در یک لحظه از زمان برنامه خاصی را با هم اجرا کنند، این برنامه بایستی در OB60 نوشته شود. شرط فعال‌سازی OB60 این است که در برنامه اصلی از SFC35 استفاده شود. هرگاه یک CPU این SFC را صدا بزند همه CPUهای Multicomputing با هم دچار وقفه شده و OB60 را اجرا می‌کنند.



شکل ۷-۷۵ وضعیت اجرای وقفه‌های OB6x

معمولاً برنامه OB60 در این CPUها یکسان است ولی اگر برنامه آنها متفاوت باشد یا اگر CPUها از نظر سرعت پردازش یکسان نباشند و اجرای OB60 را در زمان‌های مختلف انجام دهند، در اینصورت اگر قبل از اینکه اجرای OB60 در همه CPUها تمام شود مجدداً SFC35 در یکی از CPUها فراخوان شود در اینصورت OB60 در هیچ‌یک از آنها شروع نخواهد شد تا پس از اتمام اجرای OB60 قبلی مجدداً SFC35 صدا زده شود. شکل ۷-۷۶ این موضوع را بهتر نشان می‌دهد.

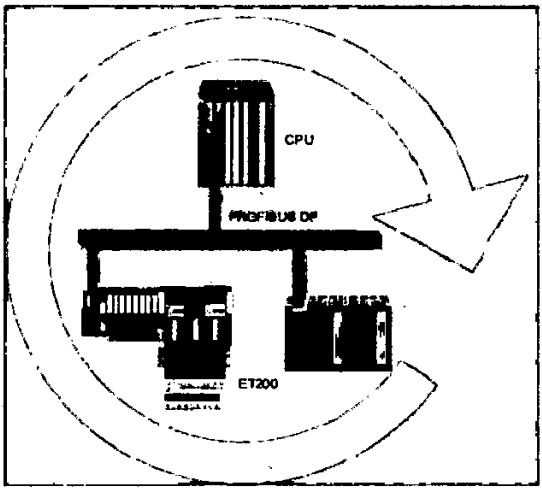


شکل ۷-۷۶ متفاوت بودن زمان اجرای OB6x

- **وقفه‌های OB61 تا OB64 موسوم به Synchronous cycle interrupts**

این وقفه‌ها برای سنکرون‌سازی سیکل شبکه Profibus DP با سیکل اجرای برنامه به‌کار می‌روند. در شبکه DP وسایل متصل به باس یکی پس از دیگری توسط CPU اصلی خوانده می‌شوند. CPU پس از خواندن آخرین وسیله مجدداً به اولین وسیله مراجعه می‌کند و این کار را دائماً تکرار می‌نماید. به این سیکل اصطلاحاً Bus Cycle گفته می‌شود. اگر این سیکل با سیکل اجرای برنامه یکسان نباشد در اینصورت خواندن اطلاعات شبکه در برنامه یا ارسال فرامین به وسایل شبکه ممکن است از نظر زمانی درست نباشد.

برای سنکرون‌سازی بین زمان Bus Cycle و زمان Scan Cycle می‌توان از OB61 تا OB64 استفاده نمود. این OBها در صورت لزوم با ایجاد تأخیرهای لازم دو سیکل فوق را همزمان می‌کنند.



شکل ۷-۷۷ سنکرون‌سازی سیکل باس با سیکل برنامه

- **وقفه OB65 موسوم به Technology synchronization interrupt**

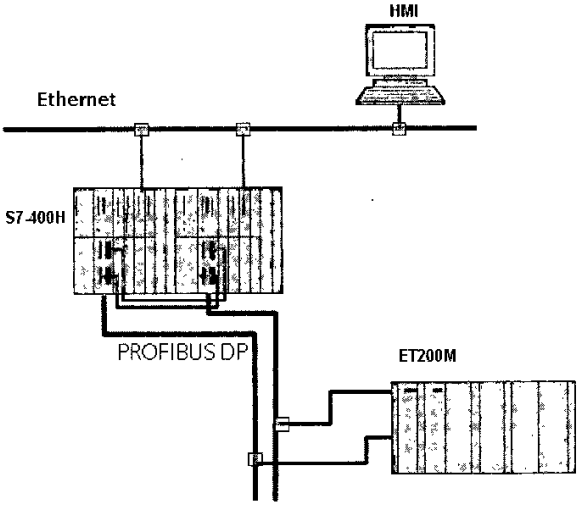
این وقفه‌ها خاص CPUهای تکنولوژیکی CPU31x-T هستند که در Motion Control کاربرد دارند. با استفاده از این وقفه‌ها برنامه وقتی شروع می‌شود که بلاک‌های تکنولوژیکی Update شده باشند.

۷-۱۲-۳ وقفه‌های OB7x

این وقفه‌ها خاص سیستم‌های افزونه سخت‌افزاری می‌باشد. این سیستم‌ها به H موسوم هستند و توسط S7-400H پیاده‌سازی می‌شوند. بحث کامل این سیستم‌ها در کتاب جداگانه‌ای از مولف آورده شده است و در اینجا فقط نکاتی را به اجمال یادآور می‌شویم.

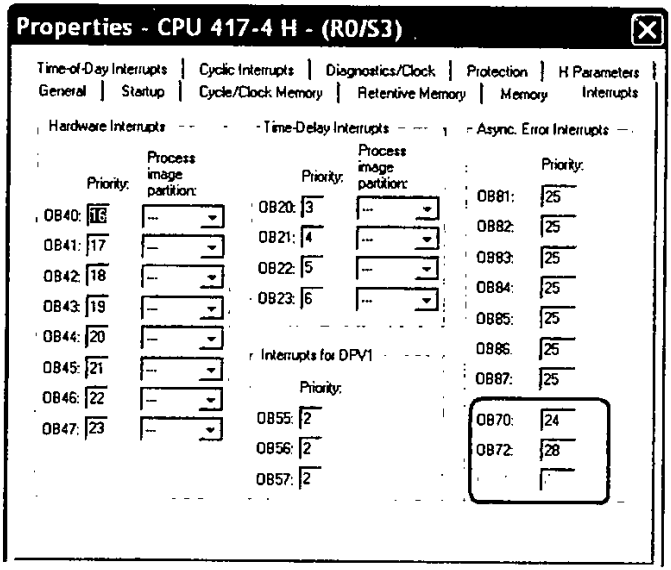
در سیستم‌های S7-400H با بحث افزونگی در دو سطح مواجه هستیم؛ یکی افزونگی در سطح کنترل و دیگری افزونگی در سطح I/O:

- CPU Redundancy: در سیستم H دو CPU وجود دارد که یکی Master و دیگری Standby می‌باشد. اگر یکی از این دو دچار وضعیت Stop شود خطای CPU Redundancy پیش می‌آید.
- I/O Redundancy: در سیستم‌های H معمولاً I/Oها روی ET200M با قابلیت افزونه قرار گرفته‌اند که توسط دو کابل شبکه پروفی‌باس به دو CPU متصل است و هر CPU می‌تواند با I/Oها از مسیر این شبکه ارتباط بگیرد. اگر یکی از این باس‌های ارتباطی دچار مشکل شود خطای I/O Redundancy پیش می‌آید.



شکل ۷-۷۸ شماتیک برخی از اجزای سیستم افزونه

برای سیستم‌های H علاوه بر وقفه‌هایی که در S7-400 معمولی وجود داشت وقفه‌هایی نیز به‌طور خاص برای حالت افزونه در نظر گرفته شده است. این وقفه‌ها با OB7x شروع می‌شوند و در شکل ۷-۷۹ نمایش داده شده‌اند.



شکل ۷-۷۹ وقفه‌های OB7x مربوط به S7-400H

این OBها همانطور که در شکل دیده می‌شود، عبارتند از:

- **OB70 موسوم به I/O Redundancy Error**

زمانی که افزونگی I/O دچار مشکل شود این وقفه فعال می‌گردد. قطع شدن یکی از کابل‌های شبکه یا جدا شدن کانکتور یا قطع تغذیه یکی از ماژول‌های رابط ET200 نمونه‌هایی از این شرایط هستند. لازم به ذکر است که در صورت بروز فالت I/O redundancy حتی اگر OB70 موجود نباشد منجر به توقف هیچ‌کدام از دو CPU نخواهد شد. فقط چراغ REDF روی CPU روشن شده و در بافر پیغام I/O Redundancy ثبت می‌گردد.

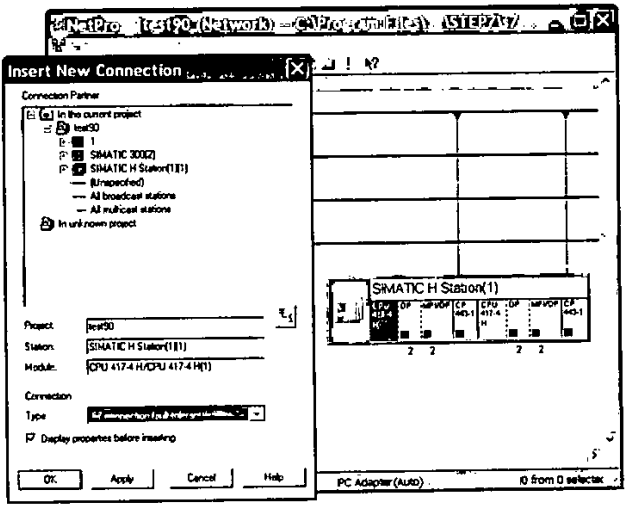
با برنامه‌نویسی OB70 می‌توان آلارم‌های لازم را تولید کرده و سیستم کنترل را به سمت شرایط مناسب هدایت نمود.

- **OB72 موسوم به CPU Redundancy Error**

زمانی که افزونگی CPUها دچار مشکل شود این وقفه فعال می‌گردد. قطع شدن فیبر نوری بین CPUها، توقف یک CPU، قطع تغذیه یک CPU نمونه‌هایی از این شرایط هستند. لازم به ذکر است که در صورت بروز فالت‌های ذکر شده حتی اگر OB72 موجود نباشد منجر به توقف CPU اصلی که در حال کار است نخواهد شد. فقط چراغ REDF روی CPUها روشن شده و در بافر پیغام CPU Redundancy ثبت می‌گردد. در اینجا نیز با برنامه‌نویسی OB72 می‌توان خطا را مدیریت کرد و آلارم‌های لازم را تولید نمود.

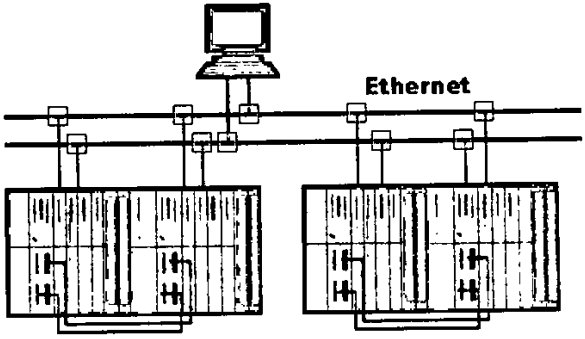
- **OB73 موسوم به Communication Redundancy Error**

برای تبادل دیتا بین سیستم H و سایر سیستم‌ها ایجاد یک اتصال در Netpro مورد نیاز می‌باشد این اتصال به S7 Fault Tolerant Connection موسوم است.



شکل ۷-۸۰ اتصال S7 Connection Fault Tolerant در Netpro

در عمل برای ارتباط بین سیستم مانیتورینگ و سیستم H و نیز برای ارتباط بین دو سیستم H از این اتصال استفاده می‌شود. شبکه ارتباطی معمولاً اترنت صنعتی است که می‌تواند مانند شکل ۷-۸۰ به‌صورت تکی یا مانند شکل ۷-۸۱ به‌صورت افزونه باشد. در هرحال اگر مشکلی در ارتباط S7 Fault Tolerant پیش بیاید وقفه OB73 فعال خواهد شد. عدم وجود این وقفه منجر به توقف CPU نمی‌گردد. برنامه‌نویسی آن برای مدیریت فالت و تولید آلارم مفید است.



شکل ۷-۸۱ استفاده از شبکه اترنت افزونه در سیستم‌های S7-400H

۷-۱۳ پرسش و تحقیق

- چگونه می‌توان وقفه‌ها را مدیریت کرد تا در وسط OB1 آن را قطع نکنند بلکه وقتی OB1 به پایان رسید فعال شوند.
- آیا در سایر PLCها وقفه‌ها به اندازه PLCهای زیمنس متنوع هستند؟

فصل ۷

۷-۱۴ تمرین

- کنتور نرم‌افزاری مثال ۷-۱۷ را که با تقریب مستطیلی طراحی شده را با تقریب ذوزنقه ای حل کنید. سپس آن را به‌صورت یک FB طراحی کنید که بتوان در پروژه‌های مختلف از آن استفاده کرد.
- فانکشن‌بلاکی به‌عنوان تایمر تأخیر در وصل نرم‌افزاری طراحی کنید که زمان را به‌صورت یک عدد اعشاری بگیرد و پس از گذشت زمان مورد انتظار خروجی آن یک شود. عملکرد در OB35 است و پس از فعال شدن ورودی مورد نظر با هر بار فراخوانی OB35 زمان آن با زمان‌های فراخوانی قبلی جمع شده و در صورت رسیدن به زمان مورد نیاز خروجی فعال گردد.

فصل ۸

کنترل PID با PLC

در این فصل چگونگی کنترل لوپ‌های پیوسته و پله‌ای با استفاده از فانکشن بلاک‌های موجود در کتابخانه نرم‌افزار Step7 تشریح می‌شود.

چکیده مطالب

- سیستم‌های کنترل به دو دسته حلقه باز و حلقه بسته تقسیم می‌شوند. حلقه باز برای اعمال فرمان دستی به سیستم استفاده می‌شود و حلقه بسته برای کنترل خودکار به‌کار می‌رود.
- پارامترهای P , I, D در سیستم حلقه بسته کاربرد دارند و با تنظیم آنها پاسخ لوپ بهینه خواهد بود.
- در عمل برای تنظیم لوپ از روش‌های تجربی استفاده می‌شود.
- استراتژی‌های مختلفی برای لوپ کنترلی وجود دارد مانند Cascade، Ratio، Override.
- لوپ کنترلرها انواع مختلف دارند. برخی به‌طور خاص برای کنترل لوپ طراحی شده اند و برخی چند منظوره هستند.
- PLC سیستم کنترل چند منظوره است که از آن می‌توان برای کنترل تعداد محدودی لوپ کنترلی استفاده کرد.
- در PLC پیاده‌سازی لوپ کنترلی با استفاده از فانکشن‌های از قبل نوشته شده انجام می‌شود. FB41 و FB42 از این نمونه هستند.
- برا ی کنترل پیوسته از FB41 استفاده می‌شود.
- برای کنترل پله‌ای عملگرهای سه مرحله‌ای از FB42 استفاده می‌شود.
- برای کنترل پله‌ای عملگرهای دو مرحله‌ای از FB41 به همراه FB43 استفاده می‌شود.
- FBهای فوق در OB3x فراخوان می‌شوند.

فصل ۸

۸-۱ مقدمه

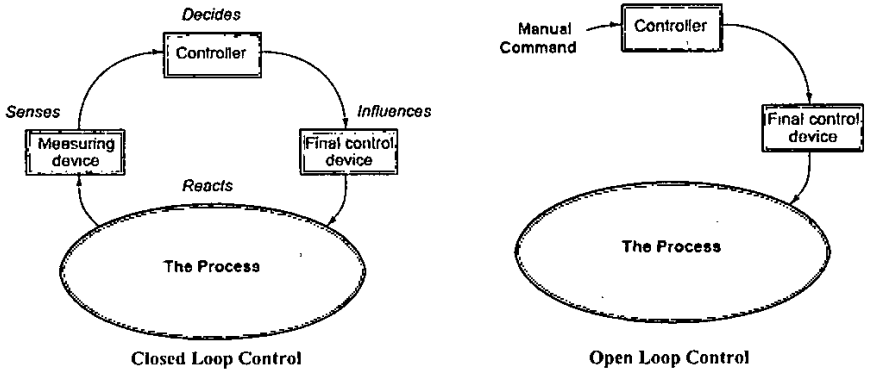
بحث کنترل PID بحث مفصلی است که تشریح آن نیاز به کتاب جداگانه ای دارد. در این کتاب هدف آن نیست که موضوع PID به‌طور کامل بحث شود، بلکه هدف به‌طور خاص نحوه پیاده‌سازی و کنترل لوپ توسط PLC می‌باشد. در این چارچوب از بررسی معادلات تئوری حاکم بر سیستم اجتناب شده و صرفاً نکات کاربردی مد نظر قرار گرفته است. با این وجود یادآوری برخی مفاهیم و اصطلاحات مربوط به PID ضرورت داشته که ابتدا به آنها پرداخته شده است.

۸-۲ مفاهیم و اصطلاحات PID Control

۸-۲-۱ کنترل حلقه باز و حلقه بسته

سیستم‌های کنترلی به دو نوع حلقه باز و حلقه بسته تقسیم می‌شوند. در نوع حلقه باز فیدبکی از فرآیند گرفته نمی‌شود ولی نوع حلقه بسته بر مبنای فیدبک استوار است.

وقتی صحبت از PID Control به میان می‌آید عملاً با یک لوپ کنترلی سروکار داریم. کنترل PID از اجزایی تشکیل می‌شود که ساختار آنها در واقع به‌صورت یک حلقه است، از اینرو به آن لوپ کنترلی می‌گویند. در یک حلقه کنترلی سنسور به‌عنوان عنصر اولیه کمیت مورد نظر (مانند فشار، فلو، دما و...) را اندازه‌گیری کرده و به کنترلر می‌فرستد. کنترلر با توجه به فیدبک دریافتی تصمیم می‌گیرد که چه فرمانی به المان نهایی کنترل[1] بفرستد، تغییر وضعیت عملگر منجر به تغییر کمیت فرآیندی می‌شود. کنترلر با دریافت فیدبک جدید فرمان را در صورت لزوم تغییر می‌دهد و این حلقه به‌طور سیکلی و مداوم تکرار می‌گردد.

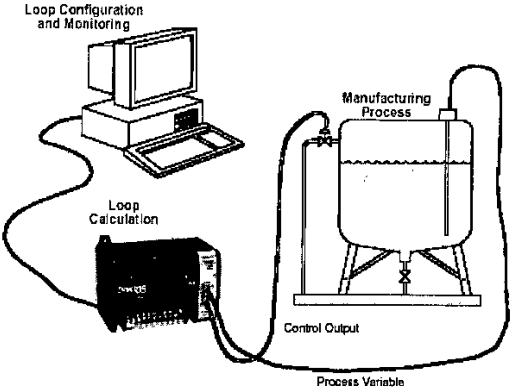


شکل ۸-۱ سیستم کنترل حلقه باز و حلقه بسته

به‌عنوان مثال یک لوپ کنترل سطح از اجزایی مانند شکل ۸-۲ تشکیل شده است، که عملاً یک حلقه را تشکیل می‌دهند. ترانسمیتر میزان سطح را حس می‌کند و سیگنال سطح را به کنترلر می‌فرستد. کنترلر براساس مقدار مبنای تعیین

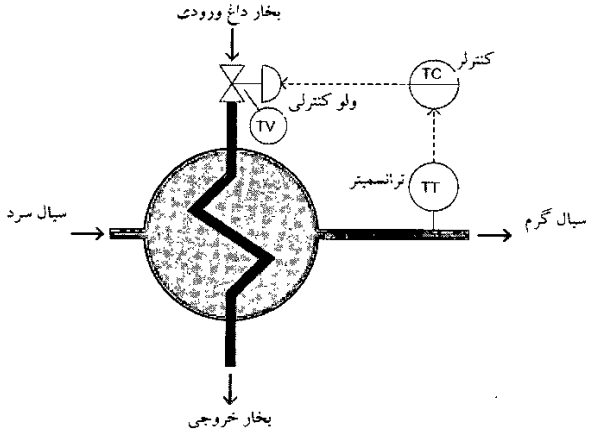
1. Actuator

شده نسبت به باز و بسته کردن ولو کنترلی اقدام می‌کند. پس در این شکل مجموعه ترانسمیتر، کنترلر، ولو کنترلی و مخزن یک حلقه را تشکیل می‌دهند.



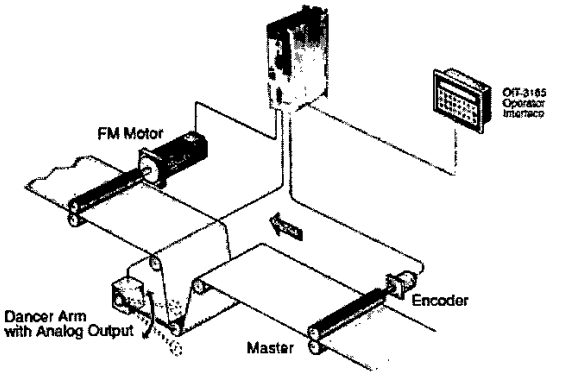
شکل ۸-۲ نمونه لوپ کنترل سطح

شکل ۸-۳ یک لوپ کنترل دما را نشان می‌دهد. ترانسمیتر دمای آب را به کنترلر می‌فرستد و کنترلر با توجه به فیدبک دریافتی ولو مسیر بخار داغ را باز و بسته می‌کند تا دمای سیال را تنظیم کند.



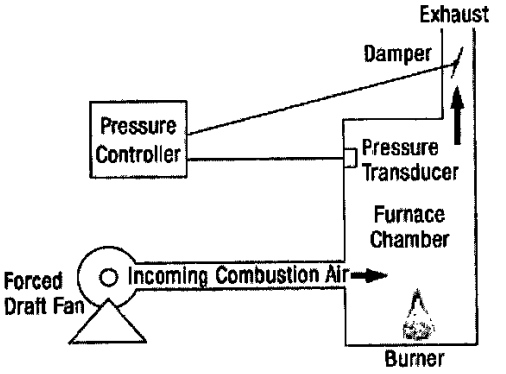
شکل ۸-۳ مثالی از لوپ دما

در شکل ۸-۴ سرعت و میزان کشش ورق کنترل می‌شود. فیدبک از Encoder و Dancer دریافت شده و دور موتور توسط کنترلر تنظیم می‌گردد.



شکل ۸-۴ نمونه لوپ کنترل سرعت

در شکل ۸-۵ فشار داخل کوره توسط ترانسمیتر اندازه‌گیری شده و فرمان لازم توسط کنترلر به دریچه کنترل فشار داده می‌شود.

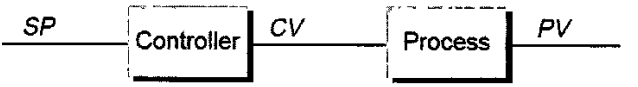


شکل ۸-۵ نمونه لوپ کنترل فشار

به‌طور کلی سیستم‌های حلقه بسته و حلقه باز به‌صورت زیر مدل می‌شوند:

- **مدل سیستم حلقه باز**

در این سیستم فرمان مبنا مستقیماً توسط کنترلر به خروجی منتقل می‌گردد و فیدبکی در کار نیست.



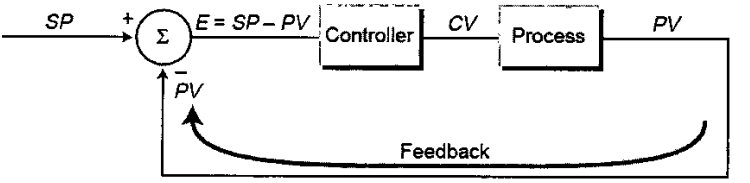
شکل ۸-۶ مدل سیستم حلقه باز

که در آن:

SP: Setpoint
CV:Controlled Variable
PV: Process Varriable

- **مدل سیستم حلقه بسته**

اگر کنترلر به‌صورت حلقه بسته باشد، مدل شکل ۸-۷ را داریم. همانطور که در این شکل دیده می‌شود از تفاضل SP و PV سیگنال Error ساخته می‌شود که براساس آن فرمان کنترلی ایجاد می‌گردد.



شکل ۸-۷ مدل سیستم حلقه بسته

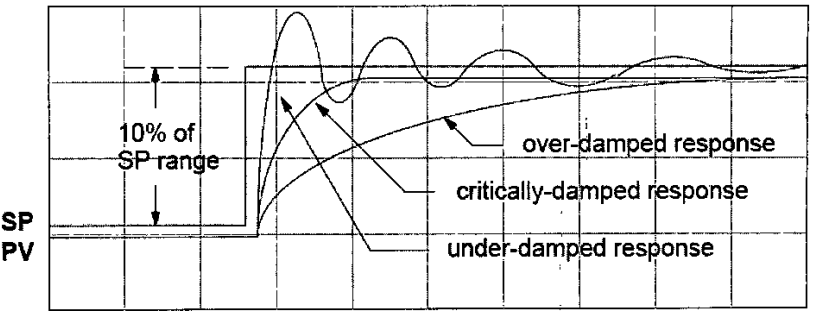
تذکر: در سیستم‌های حلقه بسته امکان ایجاد سیستم حلقه باز (بدون فیدبک) نیز در نظر گرفته می‌شود تا در مواقع لزوم بتوان به‌صورت دستی از آن استفاده نمود.

۸-۲-۲ شناخت پارامترهای P ,I,D

کنترلر حلقه بسته بایستی به‌صورتی عمل کند که فرآیند به‌صورت صحیح کنترل گردد. منظور از کنترل صحیح آن است که اگر مقدار مبنا تغییر کرد، کنترلر به‌صورتی به عملگر فرمان بدهد که پاسخ سیستم یعنی روند تغییرات PV به‌صورت زیر باشد:

- ناپایدار نباشد یعنی مرتباً اختلاف مقدار مبنا و مقدار واقعی زیاد نشود.
- نوسانی نباشد یعنی مرتباً حول نقطه مبنا کم و زیاد نشود.
- کند نباشد یعنی در کوتاهترین زمان ممکن به پایداری برسد.
- Overshoot یا Undershoot زیاد نداشته باشد و منجر به ضربه زدن به خروجی (Actuator) نگردد.

شکل ۸-۸ پاسخ‌های مختلف یک لوپ کنترلی را نشان می‌دهد:



شکل ۸-۸ نمونه پاسخ‌های مختلف یک فرآیند پس از اعمال پله

برای رسیدن به حالت ایده‌آل یا نزدیک به حالت ایده‌آل، پارامترهایی با عنوان P و I و D در کنترلرها تنظیم می‌گردد؛ از اینرو به آن PID Controller می‌گویند که در آن

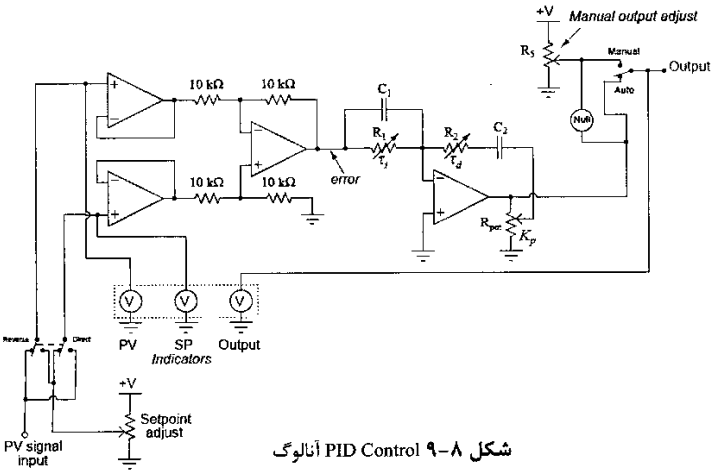
Proportional : **P**

Integral : **I**

Derivative :**D**

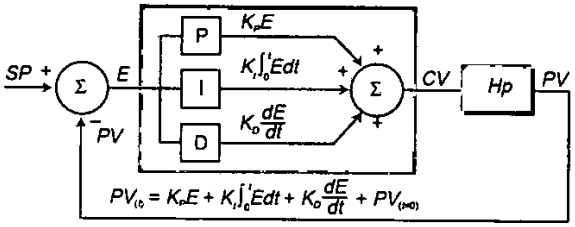
نکات قابل توجه

- در کنترلرهای میکروپروسسوری این ضرایب به‌صورت نرم‌افزاری هستند.
- در کنترلرهای نیوماتیکی این ضرایب با تغییر ولوهای تنظیمی تغییر می‌کنند.
- در کنترلرهای آنالوگ این تنظیم با تغییر مقاومت انجام می‌شود. شکل ۸-۹ نمونه‌ای از PID Controller آنالوگ را نشان می‌دهد.



شکل ۸-۹ PID Control آنالوگ

در کنترلر PID ضرایب P و I و D روی سیگنال Error اعمال شده و فرمان لازم به کنترلر را طبق فرمول نشان داده شده در شکل ۸-۱۰ ایجاد می‌کنند.



شکل ۸-۱۰ رابطه بین ورودی و خروجی‌های لوپ

فصل ۸

در سیستم‌های میکروپروسسوری این فرمول در کنترلر به‌صورت زیر پیاده‌سازی می‌شود:

$$Mn = Kc * en + Ki * \sum_{i=1}^{n} ei + Kr * (en - en-1) + Mo$$

Control Output — Proportional Term — Integral Term — Derivative Term — Initial Output — Bias Term

Ts = Sample rate
Kc = Proportional gain
Ki = Kc * (Ts/Ti) coefficient of integral term
Kr = Kc * (Td/Ts) coefficient of derivative term
Ti = Reset time (integral time)
Td = Rate time (derivative time)

SPn = Set Point for sampling time "n" (SP value)
PVn = Process variable for sampling time "n" (PV)
en = SPn – PVn = Error term for sampling time "n"
M0 = Control Output for sampling time "0"
Mn = Control Output for sampling time "n"

برای شناخت یک PID Controller لازم است رفتار پارامترهای P و I و D را به‌طور مستقل مورد بررسی قرار دهیم.

کنترل کننده نوع P (تناسبی)

اگر کنترلر فقط تناسبی باشد، سیگنال Error در ضریب تناسبی که Kp یا Gain گفته می‌شود ضرب شده و سیگنال فرمان تولید می‌شود. بنابراین خروجی متناسب با ورودی طبق رابطه زیر تغییر می‌کند.

$$m = Kpe + b$$

که در آن:

m = Controller output

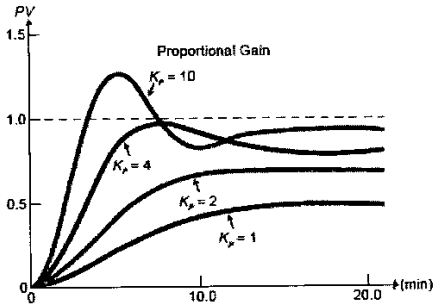
e = Error (SP-PV)

Kp = Proportional gain

b = Bias

ویژگی‌ها

- کنترلر تناسبی نسبتاً سریع است و هرچه Kp بیشتر باشد، سرعت پاسخ آن بیشتر است.
- خطای حالت ماندگار[1] دارد، یعنی همیشه بین مقدار مبنا و مقدار واقعی اختلاف وجود دارد.
- افزایش مقدار P خطای ماندگار را کاهش می‌دهد ولی ممکن است سیستم را نوسانی کند.
- در سیستم‌های کنترلی ساده که خطای ماندگار در آنها قابل قبول است می‌تواند به‌کار رود.



شکل ۸-۱۱ تاثیر ضریب P روی پاسخ سیستم

- اگر ضریب Kp بسیار بزرگ باشد کنترلر به‌صورت On/Off عمل خواهد کرد یعنی وقتی SP>PV، فرمان 100% به عملگر می‌فرستد و وقتی SP<PV، فرمان 0% به عملگر ارسال می‌کند.
- در برخی کاربردها کنترلر بایستی به‌صورت Direct عمل کند، یعنی افزایش فیدبک منجر به افزایش فرمان شود. در برخی کاربردها کنترلر بایستی به‌صورت Reverse عمل کند یعنی افزایش فیدبک منجر به کاهش فرمان می‌شود. با تغییر علامت ضریب Kp می‌توان کنترلر را از حالت Direct به Reverse تبدیل کرد.
- در برخی موارد به‌جای ضریب Kp از ضریب PB استفاده می‌کنند که عکس Kp است. PB مخفف Proportional Band است و برحسب % بیان می‌شود.

$$PB=1 / Kp \quad \text{یا} \quad Kp = 1/PB$$

پس این دو را نباید با هم اشتباه کرد. به‌عنوان مثال:

- Gain = 20 ; Proportional band = 5%
- Proportional band = 20%; Gain = 5

کنترل کننده نوع I (انتگرال‌گیر)

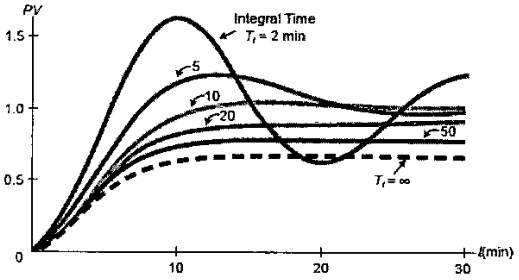
این کنترلر با انتگرال‌گیری از سیگنال error فرمان به عملگر را ایجاد می‌کند. پس در واقع در هر لحظه سیگنال error فعلی را با errorهای قبلی جمع می‌کند و آنقدر فرمان را تغییر می‌دهد تا error صفر شود. ضریب Ti زمان انتگرال‌گیری است.

1. Steady State Error

$$m = \frac{1}{\tau_i} \int e\, dt$$

ویژگی‌ها

- خطای ماندگار ندارد.
- هرقدر Ti بزرگتر باشد، پاسخ سیستم کندتر می‌شود.
- اگر Ti کوچک انتخاب شود منجر به Overshoot و نوسانی‌شدن پاسخ می‌شود. زیرا انتگرال‌گیر سعی می‌کند در زمان کوتاهتری خطا را صفر کند.
- در سیستم‌های کنترلی سریع می‌تواند به‌کار رود ولی معمولاً I به تنهایی استفاده نمی‌شود.



شکل ۸-۱۲ تاثیر ضریب I روی پاسخ سیستم

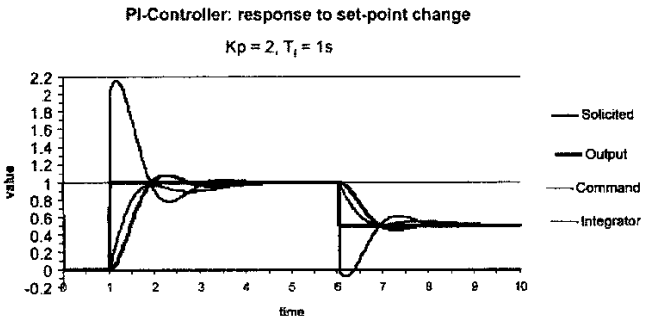
کنترل کننده نوع PI

با ترکیب کنترلر P و I ساخته می‌شود، بنابراین رابطه زیر برآن حاکم است:

$$m = K_p e + \frac{1}{\tau_i} \int e\, dt + b$$

ویژگی‌ها

- سرعتش به‌خاطر وجود عنصر P نسبتاً خوب است.
- خطای ماندگار آن به‌خاطر وجود عنصر I ناچیز است.
- کاربرد فراوان در عمل دارد. بسیاری از لوپ‌های کنترلی به‌صورت PI کار می‌کنند.



شکل ۸-۱۳ پاسخ سیستم PI

کنترل کننده نوع D

در این کنترلر فرمان براساس مشتق‌گیری از سیگنال error ایجاد می‌گردد. به‌عبارت دیگر همیشه وضعیت فعلی سیگنال error نسبت به وضعیت لحظه قبل مقایسه می‌شود و اگر تغییر کرد فرمان تولید می‌شود. پس در این حالت صفر بودن error اهمیت ندارد بلکه تغییرات آن مهم است، به‌عبارت دیگر اگر خطا در سیستم باقی بماند و تغییر نکند مشتق‌گیر واکنشی نشان نمی‌دهد:

$$m = \tau_d \frac{dc}{dt}$$

به‌طور کلی مشتق‌گیر سیستم را حساس می‌کند. با وجود مشتق‌گیر ممکن است یک نویز ضعیف نیز منجر به تغییر وضعیت خروجی گردد.

مشتق‌گیر در عمل به تنهایی به‌کار نمی‌رود و برای لوپ‌های کند همراه با P کاربرد دارد.

کنترل کننده نوع PD

- سرعتش به‌خاطر عنصر D بالاست. بنابراین در لوپ‌هایی که ماهیت آنها کند است مانند لوپ دما کاربرد دارد. در لوپ‌های سریع منجر به نوسان می‌شود.
- ضریب D معمولاً کوچک انتخاب می‌شود تا سیستم نوسانی نگردد.
- شبیه نوع P خطای حالت ماندگار دارد.

کنترل کننده نوع PID

به‌خاطر وجود ویژگی‌های هر سه عنصر P,I,D بهترین کنترل کننده است به‌شرط اینکه ضرایب مزبور درست و متناسب با شرایط پروسه تنظیم گردند. این کنترلر از فرمول پایه یا فرمول ایده‌آل زیر تبعیت می‌کند. تفاوت فرمول ایده‌آل آن است که ضریب Kp روی مشتق‌گیر و انتگرال‌گیر نیز اعمال می‌شود.

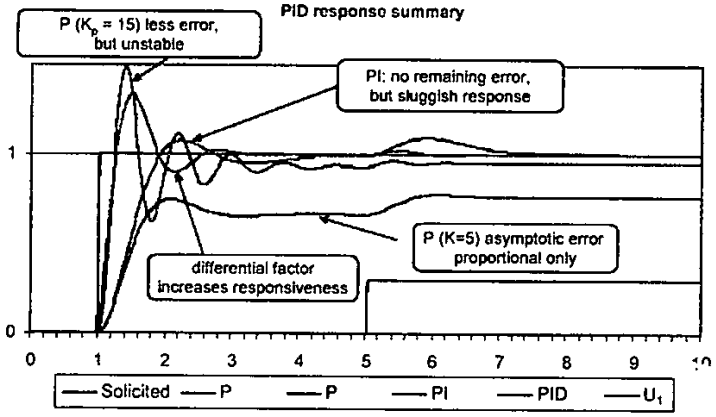
فرمول ایده ال PID یا ISA PID	فرمول پایه PID
$m = K_p\left(e + \frac{1}{\tau_i}\int e\,dt + \tau_d\frac{de}{dt}\right) + b$	$m = K_p e + \frac{1}{\tau_i}\int e\,dt + \tau_d\frac{de}{dt} + b$

برخی از سازندگان به‌صورت Optional کنترلر PID را به‌صورت فرمول زیر نیز ارائه می‌کنند که در آن به‌جای آنکه مشتق‌گیری از error انجام شود، مشتق‌گیری از PV طبق فرمول زیر صورت می‌گیرد. در این کنترلر تغییرات ناگهانی در Setpoint منجر به ایجاد ضربه به عملگر نمی‌شود.

$$m = K_p e + \frac{1}{\tau_i}\int e\,dt + \tau_d\frac{d\mathrm{PV}}{dt} + b$$

فصل ۸

شکل ۸-۱۴ کنترلرهای P و PI و PID را با یکدیگر مقایسه کرده است.



شکل ۸-۱۴ مقایسه بین کنترلرهای P و PI و PID

تاثیر افزایش ضرایب P,I,D روی پارامترهای PID کنترل در جدول ۸-۱ آورده شده است:

جدول ۸-۱

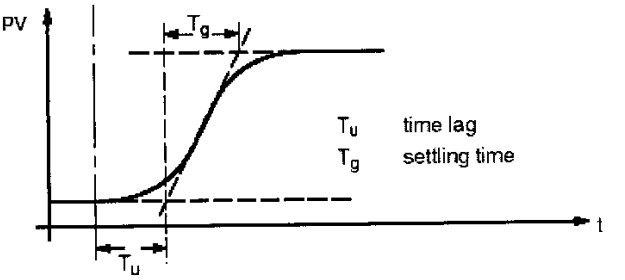
	سرعت پاسخ	Overshoot	خطای ماندگار
افزایش Kp	کاهش	افزایش	کاهش
افزایش Ki (کاهش Ti)	کاهش	افزایش	حذف
افزایش Kd	تغییر اندک	کاهش	تغییر اندک

۸-۲-۳ مقایسه انواع لوپ‌ها

در صنعت با لوپ‌های کنترلی متنوعی سروکار داریم که رفتار آنها با یکدیگر متفاوت است. از نظر سرعت پاسخ می‌توان آنها را به سه دسته زیر تقسیم کرد:

- **لوپ‌های کند:** از نمونه این لوپ‌ها می‌توان به لوپ دما اشاره کرد. عناصر و اجزای این لوپ ماهیتی کند دارند. به‌عنوان مثال از زمان فرمان باز شدن ولو گاز یک کوره تا رسیدن دما به نقطه مبنا تأخیر زمانی وجود دارد.
- **لوپ‌های سریع:** از نمونه این لوپ‌ها می‌توان به لوپ فلو و لوپ سرعت اشاره کرد. در لوپ فلو به محض فرمان به عملگر میزان فلو سریعاً تغییر خواهد کرد.
- **لوپ‌های با سرعت متوسط:** این لوپ‌ها بین لوپ‌های کند و تند قرار می‌گیرند. لوپ فشار از این نمونه است.

به‌صورت تئوری با توجه به نسبت زمان نشست[1] و زمان تاخیر[2] می‌توان یک تقسیم‌بندی برای سیستم‌ها ارائه کرد.



شکل ۸-۱۵ زمان نشست و زمان تأخیر در پاسخ سیستم

جدول ۸-۲

کنترل مشکل است.	0< Tg/Tu <3
سیستم قابل کنترل است.	3< Tg/Tu <10
کنترل ساده است.	10< Tg/Tu

نسبت فوق در برخی کاربردهای عملی برای لوپ دما بیش از ۱۰ و برای لوپ فشار حدود ۵ و برای لوپ فلو کمتر از ۲ می‌باشد.

توجه: برای لوپ‌های سریع از مشتق‌گیر استفاده نکنید. این لوپ‌ها نیاز به انتگرال‌گیر دارند.

1. Settling Time

2.Time Lag

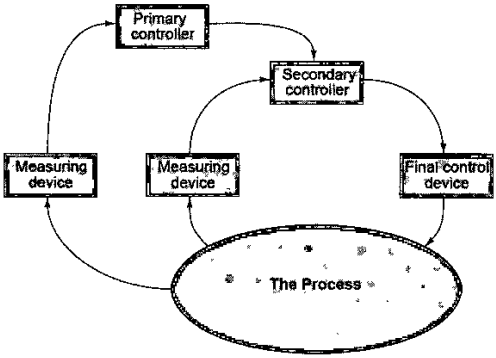
۸-۲-۴ استراتژی‌های مختلف کنترل لوپ

آنچه تا اینجا مورد بحث قرار گرفت، یک لوپ ساده کنترلی بود ولی در عمل روش‌های کنترلی دیگری که تکمیل کننده لوپ ساده است استفاده می‌شود. در اینجا به برخی از روش‌های کنترلی مورد استفاده در صنعت اشاره می‌شود. پیاده‌سازی این روش‌ها در کنترلر توسط نرم‌افزار به‌سادگی قابل اجراست و برخی سازندگان، بلاک‌های برنامه‌نویسی آماده شده‌ای برای آنها ارائه داده‌اند.

۱- Cascade Control

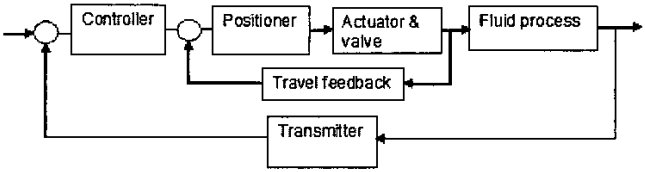
در این روش کنترل به دو بخش Primary و Secondary تقسیم می‌شود. این دو بخش دارای فیدبک‌های مجزایی از فرآیند هستند. مقدار مبنای مورد نظر به Primary داده می‌شود ولی مقدار مبنای Secondary از خروجی Primary گرفته می‌شود. کنترل المان نهایی به عهده Secondary می‌باشد. این روش در جایی که کنترل دقیق مورد نیاز است استفاده می‌شود.

فصل ۸



شکل ۸-۱۶ Cascade Control

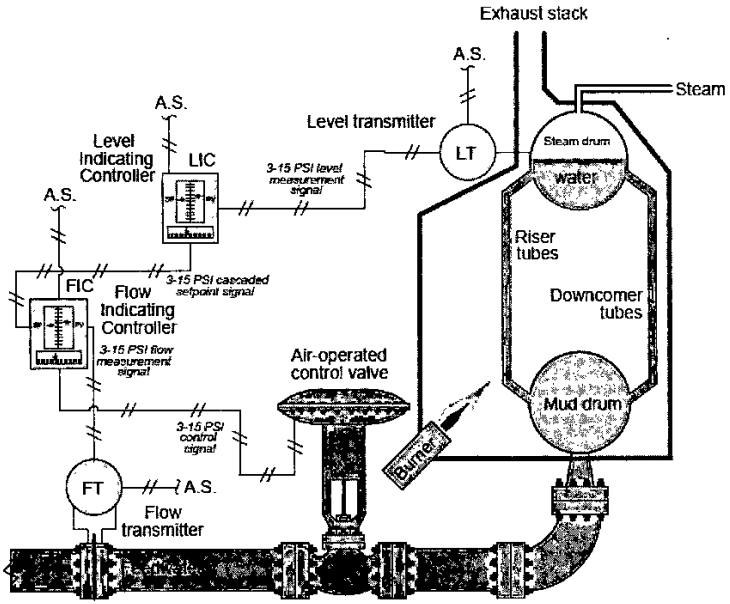
نوع متداول این استراتژی در پوزیشنر ولوهای کنترلی است. اگر ولو مستقیماً از کنترلر اصلی فرمان بگیرد(کنترلر ساده) در اینصورت فرمان بر اساس مقدار فرآیندی مانند فشار یا فلو یا دما و ... خواهد بود؛ ولی وقتی از پوزیشنر استفاده می‌شود، فرمان تولید شده توسط کنترلر اول که درصد باز شدن ولو را مشخص می‌کند به پوریشنر داده می‌شود. پوزیشنر براساس فیدبک موقعیت ولو آنرا به‌صورتی کنترل می‌کند تا به درصد مورد درخواست کنترلر اول برسد. این نوع کنترل دقیقتر و سریعتر از کنترلر ساده می‌باشد.



شکل ۸-۱۷ استفاده از پوزیشنر در لوپ Cascade

به کنترلر اولیه Master Controller و به کنترلر ثانویه Slave Controller نیز می‌گویند.

شکل ۸-۱۸ یک بویلر بخار را با لوپ Cascade نشان می‌دهد. هدف کنترل سطح مخزن بالایی در حد مورد نظر است. فیدبک سطح به کنترلر اول داده می‌شود. خروجی این کنترلر مقدار فلوی آب تزریقی را تعیین می‌کند این مقدار به‌عنوان مبنا به کنترلر دوم که فیدبک فلو را می‌گیرد و به ولو کنترلی فرمان می‌دهد، داده می‌شود.

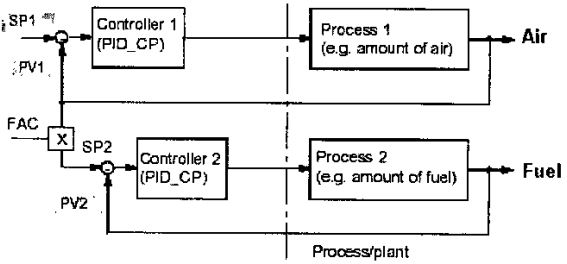


شکل ۸-۱۸ نمونه لوپ Cascade

۲- Ratio Control

با این نوع کنترل که کنترل نسبت نامیده می‌شود، در زندگی روزمره نیز سرو کار داریم. وقتی از شیرهای آب که دارای دو ولو گرم و سرد هستند برای شستسو استفاده می‌کنیم آنها را به نسبتی باز می‌کنیم تا آب با دمای مطلوب خارج گردد. این یک نوع کنترل نسبت است.

این نوع کنترل کاربردهای زیادی در صنعت دارد. سیستم‌های احتراق کوره‌ها که در آن لازم است گاز و هوا به نسبت معینی با هم مخلوط شوند از این نمونه است. همانطور که در شکل ۸-۱۹ دیده می‌شود، مقدار مبنای وارد شده به کنترلر هوا داده شده و ضریبی از آن به کنترلر سوخت اعمال می‌شود. این ضریب توسط کاربر قابل تغییر است.

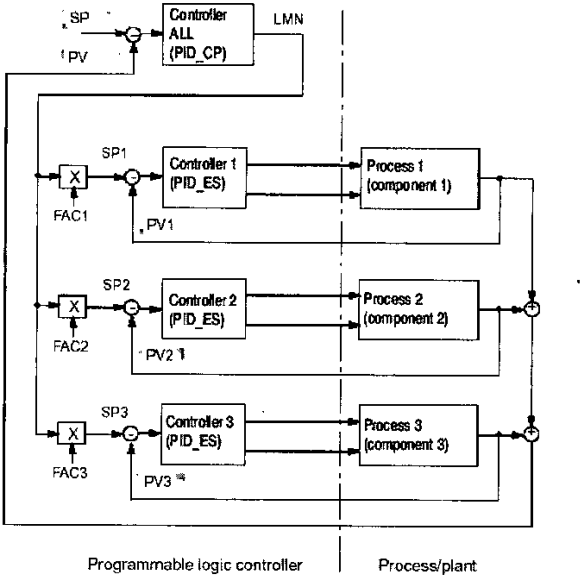


شکل ۸-۱۹ لوپ Ratio Control

نمونه دیگری از این لوپ Blending Control نام دارد.

این لوپ همانطور که از اسمش پیداست، کنترل ترکیب مواد را بر عهده دارد. در این مثال، سه ماده مختلف با مقدار مشخصی با هم ترکیب می‌شوند و کنترل مقدار هر کدام که درصدی از حجم کل ماده است توسط کنترلر خاص آن ماده انجام می‌شود. کنترل حجم کل توسط کنترل کننده اصلی انجام می‌پذیرد. همانطور که دیده می‌شود این سیستم نیز دارای یک SP می‌باشد.

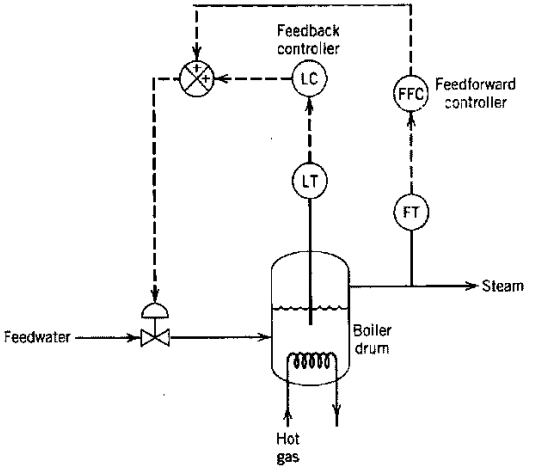
فصل ۸



شکل ۸-۲۰ لوپ Blending Control

۵- Feedforward Control

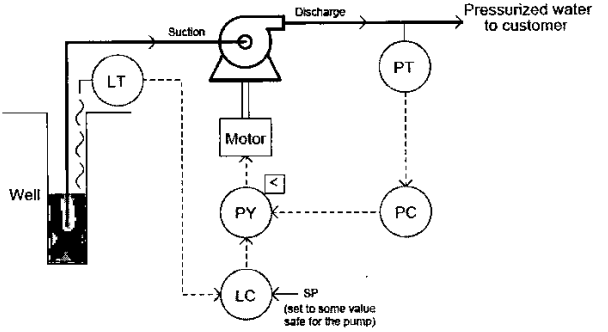
با استفاده از این روش می‌توان اغتشاش (Disturbance) را در لوپ کنترلی حذف نمود و کنترل دقیق‌تری به‌دست آورد. برای این منظور اغتشاش به‌عنوان یک سیگنال متغیر اندازه‌گیری شده و در فرمان نهایی تولید شده دخالت داده می‌شود. به همین علت به آن (Feedforward) گفته می‌شود. در شکل ۸-۲۱ برای کنترل دقیق سطح آب، فلوی بخارخروجی را اندازه‌گیری‌کرده و سیگنال کنترل سطح اصلاح می‌شود.



شکل ۸-۲۱ کنترل Feedforward

۶- Override Control

در این روش دو لوپ کنترلی وجود دارند که در برخی شرایط یکی بر دیگری برتری می‌یابد (Overriding). در شکل ۸-۲۲ تا زمانی که سطح سیال از مقدار مشخصی پایین‌تر نیامده، لوپ فشار برتری دارد و دور موتور پمپ را لوپ فشار تعیین می‌کند ولی اگر سطح از حدی کمتر شد، لوپ سطح بر لوپ فشار غلبه می‌یابد و دور موتور را کند می‌کند حتی اگر لوپ فشار درخواست دور بالاتر داشته باشد. این کار برای حفاظت پمپ است که بدون آب کار نکند.



شکل ۸-۲۲ کنترل Override

۸-۲-۵ روش تنظیم لوپ[1]

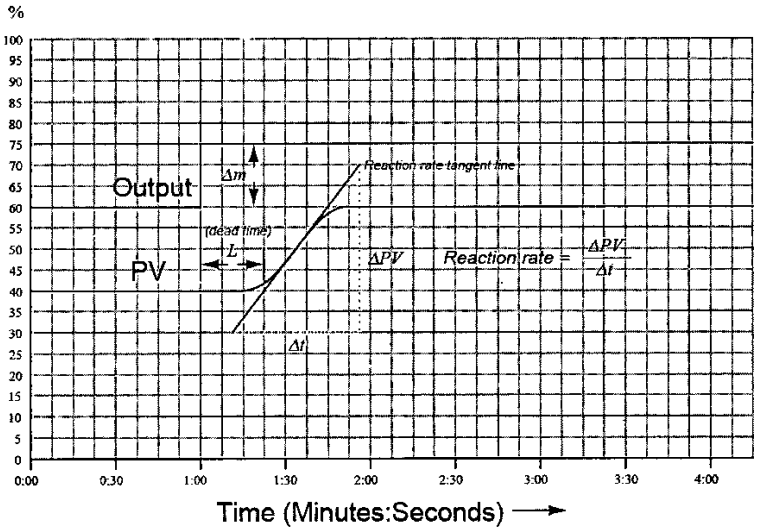
تنظیم لوپ PID در عمل که تابع تبدیل سیستم در دست نیست، معمولاً با استفاده از برخی روش‌های تجربی انجام می‌گیرد. قبل از انجام Loop Tuning بایستی شناخت کافی از فرآیند و لوپ کنترلی وجود داشته باشد و به‌ویژه تمهیدات ایمنی مربوط به آن به‌خوبی فراهم شده باشد.

در تنظیم لوپ نباید از ابتدا سیستم را به‌صورت حلقه بسته آزمایش نمود زیرا رفتار فرآیند به‌دلیل اعمال ضرایب اصلاحی کنترلر به‌خوبی قابل شناسایی نخواهد بود. بهترین روش آن است که کنترلر در مد Manual قرار گیرد و با اعمال پله (تغییر Set Point) پاسخ سیستم مورد ارزیابی قرار گیرد. برای این کار ابزارهای لازم نظیر ثبات یا کامپیوتری که بتواند روند تغییرات PV را ثبت نماید مورد نیاز خواهد بود.

برای تنظیم لوپ روش‌های مختلفی ارائه شده است که به برخی از آنها اشاره می‌شود:

۱- روش Ziegler-Nichols open-loop

دو دانشمند به نام‌های زیگلر و نیکولز روش‌هایی را برای تنظیم لوپ ارائه کرده‌اند که در اینجا به یک روش موسوم به حلقه باز اشاره می‌کنیم. در این روش سیستم را به‌صورت حلقه باز مورد آزمایش قرار داده و با اعمال پله به آن وضعیت خروجی و PV مورد بررسی قرار می‌گیرد. به شکل ۸-۲۳ توجه کنید.



شکل ۸-۲۳ تنظیم لوپ به روش زیگلر نیکولز

سپس با توجه به نمودار مقادیر L (برحسب دقیقه) و delta M (برحسب درصد)، نسبت زیر را به‌دست می‌آوریم:

1. PID Loop Tuning

$$R = \frac{\Delta \text{PV}}{\Delta t} = \frac{[\text{Percent rise}]}{[\text{Minutes run}]}$$

با توجه به مقادیر به‌دست آمده پیشنهاد زیگلر نیکولز آن است که در حالت حلقه بسته:

- اگر کنترلر فقط به‌صورت P بسته شود آنگاه:

$$K_p = \frac{\Delta m}{RL}$$

اگر کنترلر به‌صورت PI بسته شود، آنگاه:

$$K_p = 0.9\frac{\Delta m}{RL}$$

$$\tau_i = 3.33L$$

و اگر کنترلر به‌صورت PID بسته شود، آنگاه:

$$K_p = 1.2\frac{\Delta m}{RL}$$

$$\tau_i = 2L$$

$$\tau_d = 0.5L$$

توجه شود که ضرایب به‌دست آمده از فرمول‌های فوق جواب قطعی نیستند و ممکن است لازم باشد تا با تغییراتی پاسخ مناسب به‌دست آید.

۲- روش اکتشافی[1]

در این روش با جمع آوری دیتاهای به‌دست آمده از آزمایش‌های مختلف، بهترین ضرایب P و I و D کشف می‌شوند. روش کار به این صورت است که:

۱- ابتدا کنترلر را به‌صورت حلقه بسته و فقط به‌صورت P ببندید، برای این منظور ضریب Ki و Kd صفر می‌شوند.(اگر به‌جای Ki از Ti استفاده می‌شود آنرا بسیار بزرگ انتخاب کنید.)

۲- ضریب P را نزدیک به 1 قرارداهید و به‌تدریج افزایش داده تا به نقطه‌ای برسید که پاسخ به‌صورت نوسانی است.

۳- ضریب P را روی نصف مقداری که منجر به نوسان شده قرار دهید.(نوسان از بین خواهد رفت.)

۴- ضریب Ki را به‌تدریج زیاد کنید (Ti را به‌تدریج کم کنید) تا به نقطه‌ای برسید که نوسان ایجاد شود.

۵- ضریب Ki را روی نصف مقداری که منجر به نوسان شده فیکس کنید.(نوسان از بین می‌رود.)

۶- ضریب Kd را به‌تدریج افزایش دهید تا نوسان ایجاد شود.

۷- Kd را روی نصف مقداری که منجر به نوسان شده فیکس کنید.

مراحل فوق را برای نقاط کار دیگر امتحان کنید تا به نقاط مطلوب برسید.

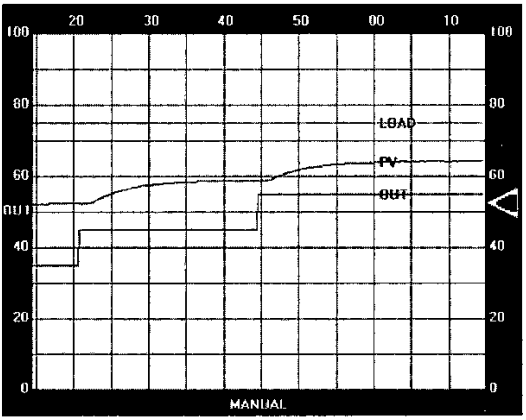
1. Heuristic

توصیه‌ها

- در سیستم‌های سریع مشتق‌گیر به‌کار نبرید.
- اگر سیگنال با نویز همراه است از مشتق‌گیر استفاده نکنید.
- برای Time Lag کمتر انتگرال‌گیر را کاهش دهید.
- برای حذف خطای ماندگار انتگرال‌گیر استفاده کنید.

مثالی از تنظیم لوپ دما توسط روش زیگلر نیکولز

در فرآیندی برای تنظیم لوپ کنترل دما از روش حلقه باز زیگلر نیکولز استفاده شده است. با اعمال فرمان در حالت حلقه باز وضعیت خروجی برحسب درصد و پاسخ PV در طول زمان (برحسب دقیقه) به‌صورت شکل ۸-۲۴ بوده است.



شکل ۸-۲۴ نمونه تنظیم لوپ دما به روش زیگلر نیکولز

با آنالیز این نمودار مقدار تغییرات خروجی به اندازه 10% و تغییرات PV در مدت ۳۰ دقیقه به اندازه 30% و مقدار Deadband=1.25 min بوده است.

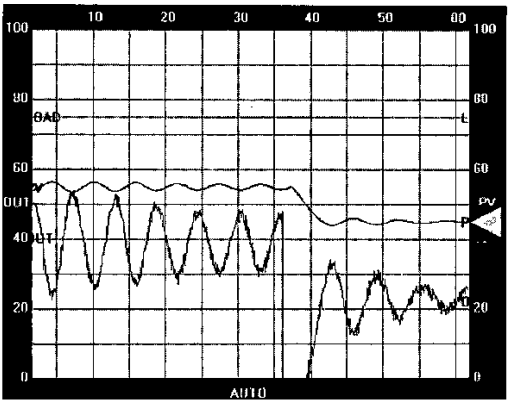
براساس روابط زیگلر نیکولز که قبلاً معرفی شد برای یک کنترلر PID ضرایب به‌صورت زیر خواهد بود:

$$K_p = 1.2\frac{\Delta m}{RL} = 1.2\frac{10\%}{\frac{30\%}{30\ \text{min}}1.25\ \text{min}} = 9.6$$

$$\tau_i = 2L = (2)(1.25\ \text{min}) = 2.5\ \text{min}$$

$$\tau_d = 0.5L = (0.5)(1.25\ \text{min}) = 0.625\ \text{min}$$

با اعمال این ضرایب و قرار دادن کنترلر در مد حلقه بسته پاسخ سیستم به‌صورت زیر ظاهر شده است.



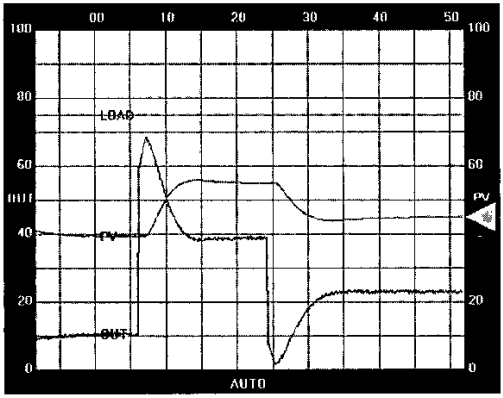
شکل ۸-۲۵ پاسخ سیستم پس از اعمال ضرایب زیگلر نیکولز

همانطور که دیده می‌شود این پاسخ مناسب نیست، زیرا:

مقدار PV (منحنی بالایی) دارای نوسان Peak-to Peak به اندازه .20% hsj

خروجی (منحنی پایینی) نیز نوسانی و در عین حال همراه با نویز است.

با کاهش مشتق‌گیر و افزایش انتگرال‌گیر و کاهش P پاسخ بهبود پیدا کرده و با مقادیر Kp=3 و Ti=5 و Td=0.5 پاسخ به‌صورت زیر در آمده که نسبتاً خوب می‌باشد؛ ولی باز تاثیرات کوچکی از نویز روی خروجی دیده می‌شود.



شکل ۸-۲۶ پاسخ نسبتاً خوب تنظیم لوپ دما

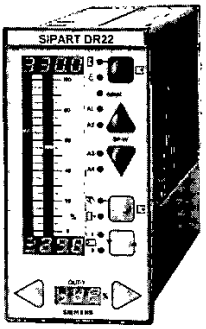
۸-۳ انواع سخت‌افزار کنترل لوپ

لوپ کنترلرهایی که دارای پردازشگر هستند به انواع مختلف تقسیم می‌شوند.

۱- لوپ کنترلرهای خاص

این لوپ‌کنترلرها که در اتاق کنترل یا در نزدیکی سیستم تحت کنترل نصب می‌شوند، دارای ترمینال‌هایی هستند که ترانسمیتر و عملگر به آن متصل می‌شود. برخی از انواع این کنترلرها می‌توانند چند لوپ را همزمان کنترل کنند (به‌عنوان مثال برخی از آنها امکان کنترل ۸ لوپ را دارا هستند).

SIPART DR ساخت زیمنس از این لوپ کنترلرهاست که دارای انواع مختلف می‌باشد. این لوپ کنترلر را می‌توان با استفاده از صفحه کلیدش یا با استفاده از نرم‌افزار SIPROM پارامتردهی کرد.



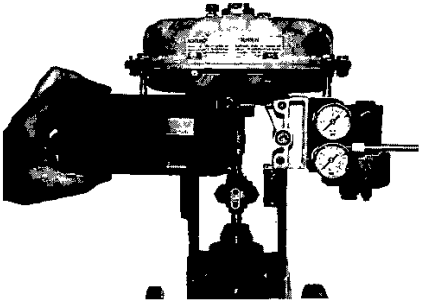
شکل ۸-۲۷ Sipart DR22

فصل ۸

نکات قابل توجه

- مقدار Set Point توسط صفحه کلید روی آنها وارد می‌شود.
- ضرایب P,I,D و سایر پارامترها نیز از طریق صفحه کلید قابل تنظیم است.
- توسط نرم‌افزار خاص نیز می‌توان به آنها متصل شد و پارامترهای آنها را تغییر داد.
- انواع مدرن آنها امکان اتصال به شبکه را دارا هستند. از طریق شبکه می‌توان به آنها مقادیر مبنا را ارسال کرد و مقادیر PV و برخی دیتاها را دریافت نمود.

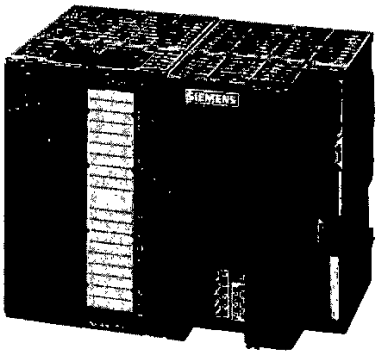
نمونه دیگر از لوپ کنترلرهای خاص می‌توان به Positioner اشاره کرد که برای کنترل موقعیت ولو به‌کار رفته و روی خود ولو نصب می‌شود. شکل ۸-۲۸ پوزیشنر ساخت زیمنس با نام SIPART PS2 را نشان می‌دهد.



شکل ۸-۲۸ Sipart PS2

۲- PLC همراه با کارت FM

برای کنترل PID به‌ویژه وقتی با لوپ‌های سریع سروکار داریم می‌توان از کارت‌های خاص کنترل PID که به Function Module موسوم هستند استفاده نمود. این کارت‌ها روی رک کنار سایر اجزای PLC نصب می‌شوند ولی بدون دخالت PLC کار کنترل لوپ را انجام می‌دهند. ترمینال‌های اتصال به ترانسمیتر و عملگر روی خود کارت تعبیه شده است. برای انجام تنظیمات این کارت‌ها نرم‌افزار خاص آن FM مورد نیاز است. نحوه کار با FMها در کتاب سطح تکمیلی آورده شده است.



شکل ۸-۲۹ کارت FM353

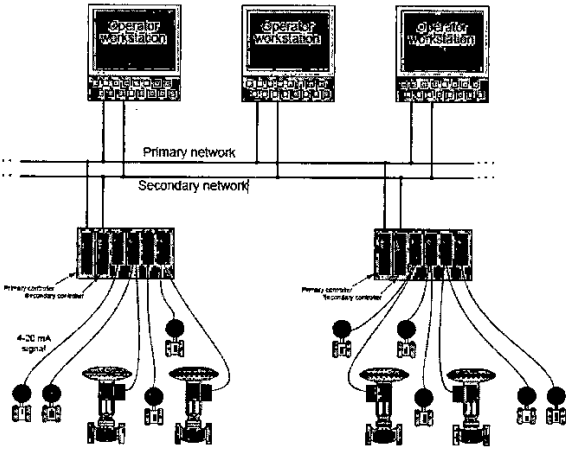
۳- PLC بدون کارت FM

در مقایسه این روش با روش استفاده از کارت FM می‌توان به موارد زیر اشاره کرد:

- نسبت به استفاده از کارت FM کم هزینه‌تر است زیرا از همان کارت‌های I/O معمولی برای اتصال به ترانسمیتر و عملگر استفاده می‌شود.
- اطمینان کمتری دارد زیرا همه لوپ‌ها در یک CPU پردازش می‌شوند ولی در FM برای هر چند لوپ یک FM مجزا وجود دارد.
- برای لوپ‌های سریع ممکن است پاسخ خوبی نداشته باشد.

۴- DCS

وقتی تعداد لوپ کنترلی زیاد باشد نیاز به ساختاری غیرمتمرکز برای کنترل آنها وجود دارد. در این موارد نمی‌توان از PLC یا لوپ کنترلرهای خاص استفاده کرد. بهترین طرح در این شرایط استفاده از سیستم‌های DCS است. DCS دارای چندین کنترلر است که هر کدام لوپ‌های کنترلی مستقلی را پردازش می‌کنند. شکل ۸-۳۰ ساختار نمونه یک سیستم DCS را نشان می‌دهد. تفاوت بین DCS و PLC در کتاب سطح مقدماتی ذکر شد.

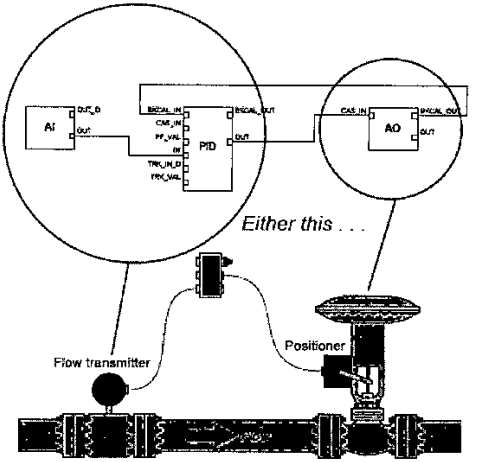


شکل ۸-۳۰ شماتیک ساختار DCS

فصل ۸

۵- Fieldbus

با استفاده از ترانسمیترها و کنترل ولوهای مدرن، امروزه امکان پیاده سازی لوپ کنترلی بین خود وسایل فراهم شده است. ترانسمیتر و کنترل ولو دارای پردازشگر هستند و می‌توان فانکشن PID را در خود آنها تنظیم نمود. شکل ۸-۳۱ نشان می‌دهد که PID در ترانسمیتر اجرا شده و فرمان از ترانسمیتر به ولو اعمال گردیده است. توضیحات بیشتر در این زمینه را در کتاب تکنولوژی فیلدباس تألیف محمدرضا ماهر – احمد حیدریان ببینید.

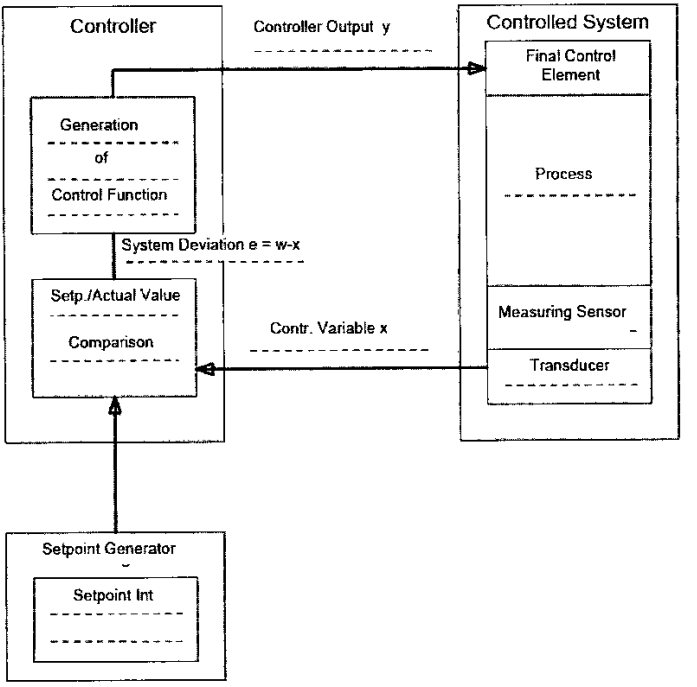


شکل ۸-۳۱ کنترل لوپ از طریق فیلدباس

۸-۴ PID کنترل با PLCهای S7

۸-۴-۱ نکات کلی

اگر بخواهیم از PLC بدون کارت FM برای PID کنترل استفاده کنیم، به روش‌های مختلف می‌توان با برنامه‌نویسی این کار را انجام داد. بلاک‌های خاصی توسط سازنده برای این منظور تهیه شده که در کتابخانه نرم‌افزار موجود هستند. در برخی از CPUهای یکپارچه نظیر CPU314C-2DP فانکشن‌بلاک‌های سیستمی برای کنترل لوپ تعبیه شده است ولی به‌طور کلی برای همه CPUهای S7-300 و S7-400 بلاک‌هایی که در کتابخانه نرم‌افزار وجود دارد قابل استفاده می‌باشد.



شکل ۸-۳۲ کنترل لوپ با PLC

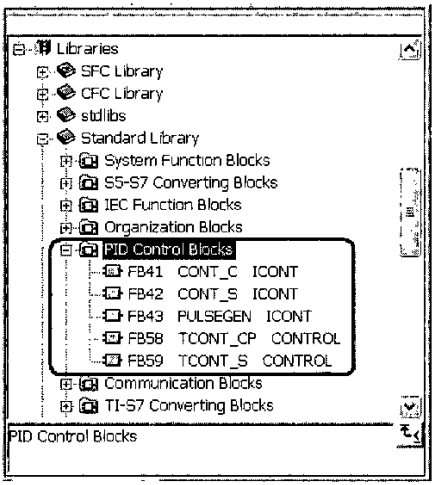
لازم به ذکر است که برای کنترل لوپ با PLC برخی ابزارهای نرم‌افزاری جداگانه‌ای نیز عرضه شده که با نصب آنها بلاک‌ها و امکانات بیشتری برای کار کنترل در دسترس خواهد بود. این نرم‌افزارهای انتخابی عبارتند از:

- Standard PID Control
- Modular PID Control

این نرم‌افزارها در این کتاب مورد بحث نیستند. در اینجا هدف طراحی و برنامه‌نویسی لوپ کنترلی از طریق بلاک‌های کتابخانه‌ای است که با نصب Step7 در دسترس قرار می‌گیرند. در Step7 سه فانکشن بلاک برای PID کنترل تعبیه شده است که عبارتند از:

- FB41 برای کنترل پیوسته[1] با نام سمبلیک Cont_C
- FB42 برای کنترل پله‌ای[2] با نام سمبلیک Cont_S
- FB43 برای تولید پالس با نام سمبلیک Pulsegen که در برخی کاربردها همراه با FB41 به‌کار می‌رود.
- FB58 به‌طور خاص برای کنترل پیوسته دما طراحی شده است.
- FB59 : به‌طور خاص برای کنترل پله‌ای دما طراحی شده است.

این بلاک‌ها در کتابخانه مانند شکل ۸-۳۳ در زیر مجموعه PID Control Blocks قرار دارند.



شکل ۸-۳۳ بلاک‌های PID در کتابخانه نرم‌افزار

FB41 و FB42 و FB43 کاربرد چند منظوره دارند و برای لوپ‌های مختلف (فشار، فلو، سطح، دما و...) قابل استفاده هستند. در این فصل به تشریح این سه بلاک می‌پردازیم. فانکشن‌بلاک‌های خاص کنترل دما مورد بحث قرار نمی‌گیرند. لازم به ذکر است که برخی طراحان سیستم‌های اتوماسیون از هیچ‌یک از این بلاک‌ها استفاده نمی‌کنند و خود اقدام به طراحی بلاک کنترلی برای کاربرد مورد نظر خود می‌نمایند و آنرا به کتابخانه نرم‌افزار اضافه کرده و در برنامه صدا می‌زنند.

انتخاب FB از کتابخانه نرم‌افزار Step7 باید با توجه به نوع عملگر یا Actuator صورت پذیرد. به‌طور کلی سه نوع عملگر داریم:

۱- عملگرهای تناسبی[3] با سیگنال تحریک پیوسته

این عملگرها به‌طور پیوسته متناسب با سیگنال (مثلاً ۴ تا ۲۰ میلی‌آمپر) کار می‌کنند. Proportional Valve نمونه‌ای از این عملگرهاست. برای کنترل این نوع عملگرها توسط PLC استفاده از FB41 کافیست. پس در این حالت PLC بایستی دارای کارت آنالوگ ورودی برای فیدبک و آنالوگ خروجی برای فرمان باشد.

1. Continuous Control
2. Step Control
3. Proportional

۱- عملگرهای تناسبی با سیگنال تحریک غیر پیوسته (پالس)

این عملگرها معمولاً دو حالت دارند مثلاً Open/Close یا On/Off. می‌توان یک هیتر برقی که برای گرم کردن کوره به‌کار می‌رود را مثال زد که با خاموش و روشن شدن گرمای مورد نظر را تولید می‌کند. برای کنترل این عملگرها علاوه بر FB41 فانکشن بلاک تولید کننده پالس یعنی FB43 نیز لازم است. پس در این حالت PLC بایستی دارای کارت آنالوگ ورودی برای فیدبک و دیجیتال خروجی برای فرمان باشد.

۲- عملگرهایی که به‌صورت Step و به‌صورت ۳ مرحله‌ای کنترل می‌شوند

اغلب عملگرهای موتوری از این نوع هستند و وضعیت آنها DOWN-O-UP است. برای کنترل اینها استفاده از FB42 به تنهایی کافیست. پس در این حالت PLC بایستی دارای کارت آنالوگ ورودی برای فیدبک و دیجیتال خروجی برای فرمان باشد که برای هر لوپ دو دیجیتال خروجی مورد نیاز است.

نحوه استفاده از FB برای PID کنترل

فانکشن‌بلاک‌های فوق الذکر که برای لوپ کنترل به‌کار می‌روند لازم است به‌صورت سیکلی توسط OBهای وقفه سیکلی[1] صدا زده شوند. این OBها همانطور که قبلاً تشریح شد، با شماره‌های OB30 تا OB38 هستند که بسته به نوع CPU ممکن است برخی از آنها موجود باشد. هر OB دارای یک زمان پیش‌فرض است که در جلوی نام آن در پارامترهای CPU در Hwconfig قابل مشاهده است. این زمان را می‌توان در صورت لزوم کاهش یا افزایش داد.

با توجه به این موارد قدم‌هایی که باید برای استفاده از FB لوپ کنترل برداشت عبارتند از:

۱- انتخاب OB3x مورد نظر و تنظیم زمان اجرای آن در Hwconfig و دانلود به PLC

۲- ایجاد OB3x فوق در پوشه Blocks برنامه Simatic Manager

۳- بازکردن OB توسط LAD/STL/FBD

۴- صدا زدن فانکشن بلاک PID همراه با یک DB دلخواه مثال : CALL FB41 , DB1

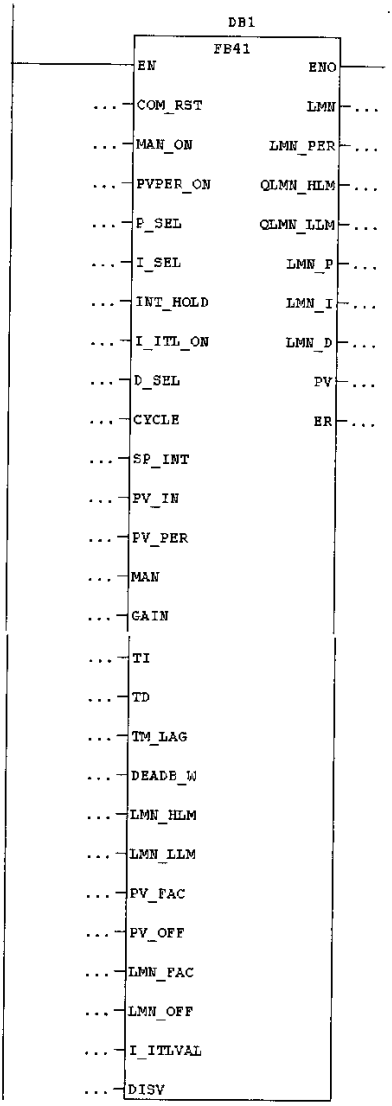
۵- تنظیم ورودی و خروجی‌های FB مطابق با توضیحاتی که در صفحات بعد خواهد آمد.

۸-۴-۲ استفاده از FB41 برای کنترل پیوسته

وقتی OB3x توسط LAD/STL/FBD باز است مانند شکل ۸-۳۴ از پنجره Program Element سمت چپ با ماوس FB41 را انتخاب کرده و به داخل Network وارد کرده و در بالای آن نام DB مورد نظر را می‌نویسیم. پارمترهای FB مانند شکل ۸-۳۴ ظاهر خواهند شد.

1. Cyclic Interrupt

OB35 : "Cyclic Interrupt"
Network 1 : Title:



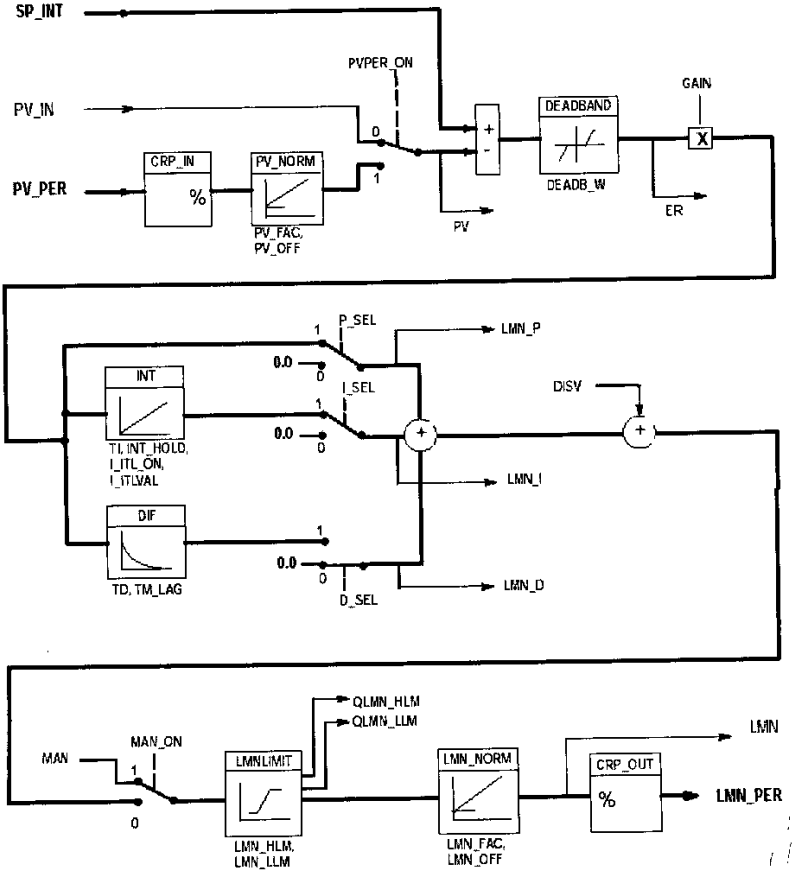
شکل ۸-۳۴ بلاک FB41

همانطور که دیده می‌شود FB41 دارای ورودی و خروجی‌های زیادی است. در عمل لازم نیست که همه این ورودی و خروجی‌ها آدرس‌دهی شوند. بدیهی است حداقل مواردی که الزاماً بایستی مشخص شوند، عبارتند از:

- **مقدار مبنا:** ورودی SP_IN
- **فیدبک:** ورودی PV_IN یا PV_PER
- **ضرایب P,I,D:** ورودی‌های Gain و Ti و Td
- **فرمان به خروجی:** خروجی LMN یا LMN_PER

قبل از معرفی ورودی و خروجی‌های FB41 لازم است با عملکرد آن آشنا شویم. برای این منظور از بلوک دیاگرام آن که در شکل ۸-۳۵ نشان داده شده است استفاده می‌کنیم. در این دیاگرام:

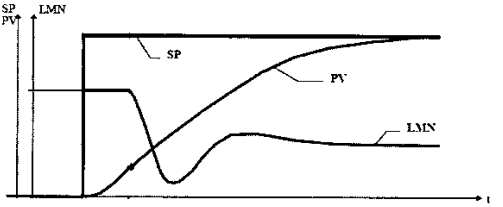
- فلش‌های به سمت داخل ورودی‌های FB هستند مانند SP_INT.
- فلش‌های به سمت بیرون خروجی‌های FB هستند مانند LMN.



شکل ۸-۳۵ بلوک دیاگرام FB41

FB41 در دیدگاه کلی

با توجه به بلوک دیاگرام می‌توان دید که کنترلر دو مقدار SP و PV را می‌گیرد، سپس آنها را مقایسه کرده و پس از عبور از فیلتر DEADBAND سیگنال خطا یا ER را تولید می‌کند. سپس ضرایب P و I و D را بسته به نیاز در آن اعمال کرده و نتیجه آنها را با سیگنال اغتشاش DISV جمع می‌کند. در نهایت سیگنال تولید شده را بین دو حد بالا و پایین محدود و به عملگر می‌فرستد.



شکل ۸-۳۶ ورودی و خروجی‌های اصلی FB41

فصل ۸

FB41 در دیدگاه دقیق

مقدار مبنا SetPoint

مقدار مبنا به ورودی SP_INT داده می‌شود. این ورودی از جنس Real است و معمولاً یک آدرس حافظه مانند MD یا دیتابلاک مانند DBD به آن اختصاص داده می‌شود. اپراتور از طریق سیستم مانیتورینگ مقدار مبنا را وارد می‌کند. این مقدار در حافظه فوق نشسته و مقدار آن به ورودی SP_INT داده می‌شود.

فیدبک PV

همانطور که در شکل دیده می‌شود، فیدبک می‌تواند به PV_IN یا PV_PER داده شده و کنترلر بسته به وضعیت سوئیچ PV_PER_ON یکی از این دو را برای مقایسه استفاده می‌کند.

PV_PER را می‌توان مستقیماً به آدرس آنالوگ ورودی که ترانسمیتر به آن متصل است، اختصاص داد. این ورودی از جنس Word است و به‌عنوان مثال می‌توان به این ورودی آدرس PIW752 را اختصاص داد.

اگر از این ورودی استفاده شود FB41 به‌صورت داخلی آنرا توسط بلوک CRP_IN بین 100- تا 100+ درصد Scale می‌کند. این کار بر اساس فرمول و جدول زیر انجام می‌شود:

$$CRP\_IN = PV\_PER * \frac{100}{27648}$$

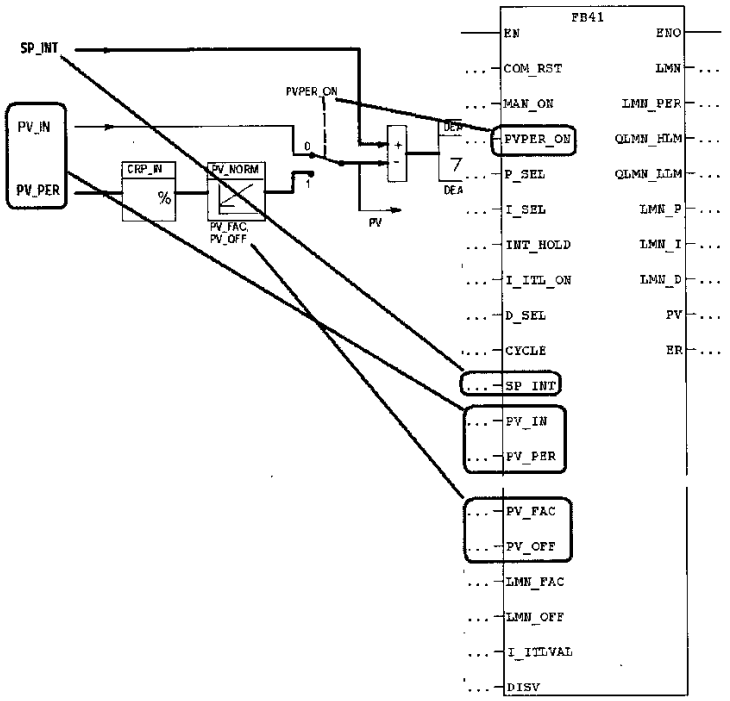
جدول ۸-۳

PV_PER	CRP_IN
32767	118.515%
...	...
27648	100%
...	...
0	0%
...	...
-27648	-100%
...	...
-32768	-118.515%

نکته: اگر از PV_PER استفاده شود، چون مقدار تبدیل به درصد می‌شود بایستی اپراتور نیز مقدار مبنا را برحسب درصد وارد کند.

پس از بلاک CRP_IN بلاک دیگری با عنوان PV_NORM قرار دارد که برای نرمالیزه کردن مقدار Scale شده به‌کار می‌رود، یعنی می‌توان مقدار Scale شده را در ضریب PV_FAC که به‌طور پیش‌فرض یک است ضرب کرد یا آنرا با مقدار آفست PV_OFF که به‌طور پیش‌فرض صفر است جمع نمود.

اگر مسیر PV_PER را در بلوک دیاگرام دنبال کنیم می‌بینیم سوئیچ PV_PER_ON در مسیر آن قطع است یعنی به‌صورت پیش‌فرض این ورودی انتخاب نمی‌شود. برای انتخاب کردن این ورودی لازم است ورودی PVPER_ON که از جنس Bool است را در FB41 یک کنیم، یعنی یک سیگنال همیشه یک به آن اختصاص دهیم.

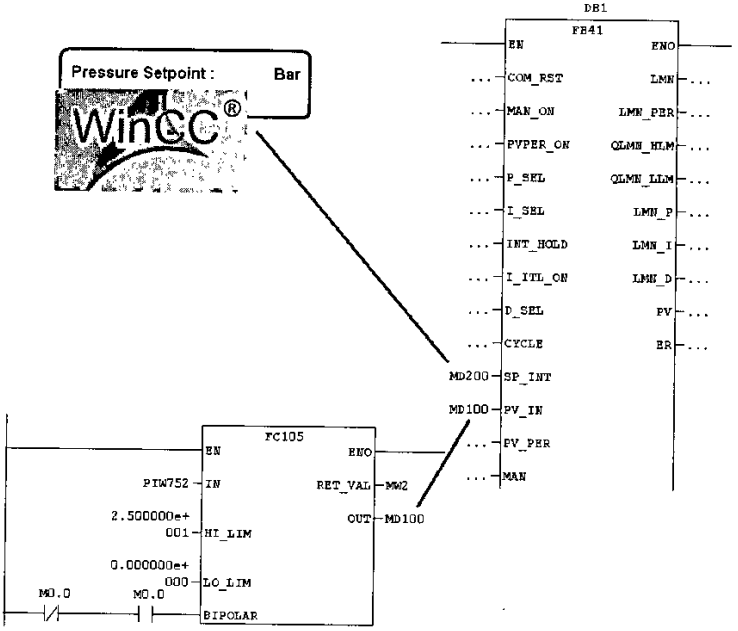


شکل ۸-۳۷ اتصال فیدبک به FB41

اگر لازم باشد که اپراتور مقدار مبنا را با توجه به کمیت مورد نظر بر حسب واحدی به‌جز درصد وارد کند، در اینصورت بایستی از ورودی PV_IN استفاده کند. به‌عنوان مثال برای لوپ فشار اگر لازم است که مبنا برحسب Bar بین 25-0 داده

شود یا برای لوپ سطح اگر لازم باشد که مقدار مبنا برحسب متر بین 5-0 داده شود، در این شرایط نباید فیدبک را به ورودی PV_PER متصل نمود بلکه باید از PV_IN استفاده کرد.

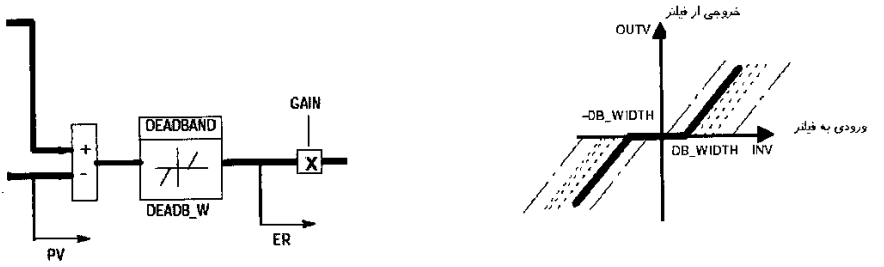
PV_IN برخلاف PV_PER مستقیما به آدرس آنالوگ متصل نمی‌شود. این ورودی از جنس Real است. در عمل فیدبک ترانسمیتر را به فانکشن Scale مانند FC105 داده و آنرا بین حدود دلخواه Scale می‌کنند سپس خروجی FC105 را به این ورودی اختصاص می‌دهند.



شکل ۸-۳۸ استفاده از PV_IN

بلوک DeadBand

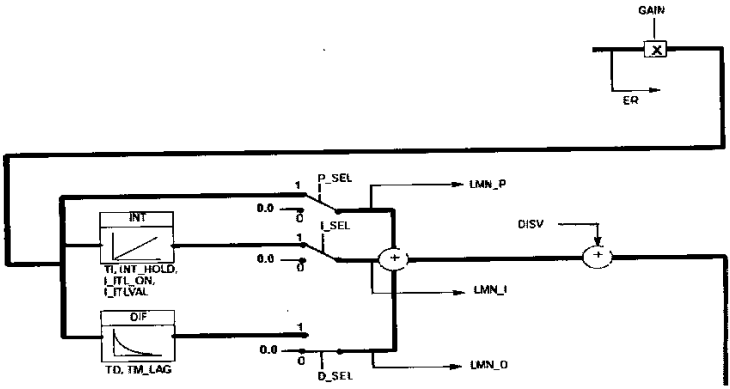
پس از محاسبه SP-PV سیگنال ایجاد شده به فیلتر Dead Band داده می‌شود و از آنجا سیگنال خطا ER تولید می‌شود. به‌طور پیش‌فرض مقادیر این فیلتر صفر است بنابراین تاثیری روی نتیجه تفاضل فوق ندارد. اگر در عمل به‌دلیل تاثیر نویز سیگنال دریافتی از ترانسمیتر حاوی نویز باشد و فیلتر فوق فعال نشود، این نویز به کنترلر منتقل شده و خروجی را دچار نوسان می‌کند به‌ویژه اگر مشتق‌گیر فعال شده باشد. برای فعال‌سازی این فیلتر پهنای فیلتر به‌صورت عدد اعشاری به ورودی DEADB_W داده می‌شود، این مقدار معمولاً کوچک است زیرا دامنه نویزها زیاد نیست. فرض کنید این مقدار عدد 2.5 داده شود در اینصورت تغییرات SP-PV به اندازه 2.5-/+ دیده نمی‌شود.



شکل ۸-۳۹ استفاده از Dead Band

اعمال ضرایب PID

با دنبال کردن مسیر بلوک دیاگرام می‌بینیم که ضریب P به‌عنوان ورودی Gain که یک عدد Real است در سیگنال Error ضرب می‌شود. سپس سیگنال ضرب شده به بلوک انتگرال‌گیر و بلوک مشتق‌گیر داده می‌شود و سرانجام این سه با هم جمع می‌گردند. همانطور که مشخص است این کنترلر طبق فرمول ISA PID طراحی شده است.



شکل ۸-۴۰ بلوک‌های PID در FB41

از شکل ۸-۴۰ مشخص است که به‌طور پیش‌فرض کنترلر به‌صورت PI بسته شده است و کاربر در صورت نیاز به مشتق‌گیر لازم است سوئیچ D_SEL را فعال کند. به‌همین شکل کاربر می‌تواند انتگرال‌گیر را با استفاده از سوئیچ I_SEL از مدار خارج کند.

ضریب انتگرال‌گیر به‌صورت زمان است و به ورودی Ti به فرمت T# داده می‌شود. توجه داشته باشید که Ti عکس Ki در فرمول‌های PID می‌باشد.

ضریب مشتق‌گیر نیز از جنس زمان است و به ورودی Td به فرمت T# داده می‌شود.

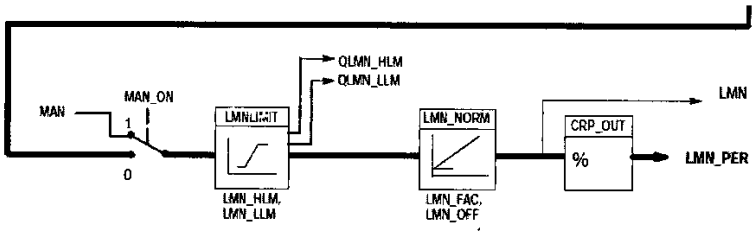
اعمال Disturbance

در کاربردهایی که لازم است کنترلر به‌صورت Feedforward بسته شود، می‌توان سیگنال اغتشاش را به ورودی DISV مخفف Disturbance Value داد. این ورودی از جنس Real است. پس اگر سیگنال اغتشاش به‌عنوان مثال از یک ترانسمیتر دریافت می‌شود، آنرا Scale کرده و نتیجه را به این ورودی اعمال می‌کنیم. Disv در شکل قبل مشخص شده است.

مد کاری Manual

در ادامه بلوک دیاگرام پس از DisV مشاهده می‌کنیم که به‌صورت پیش‌فرض مسیر PID قطع است و سیستم به‌صورت Manual کار می‌کند. اگر لازم است که کنترلر را به حالت Auto در آوریم بایستی سوئیچ Man_on را به وضعیت 0 برگردانیم، یعنی یک سیگنال همیشه صفر به این ورودی در FB41 بدهیم.

در حالت Manual فرمان مورد نظر به‌صورت یک مقدار Real به ورودی MAN داده می‌شود.



شکل ۸-۴۱ مد Manual در FB41

اعمال فرمان به خروجی

همانطور که در بلوک دیاگرام دیده می‌شود، فرمان نهایی چه در حالت Auto و چه در حالت Manual از بلوک محدود کننده LMN_LIMIT می‌گذرد، با اختصاص حدود بالا و پایین می‌توان از اعمال فرمان‌های غیر مجاز به عملگر جلوگیری نمود. به‌طور پیش‌فرض حد بالا 100 و حد پایین 0 در نظر گرفته شده است.

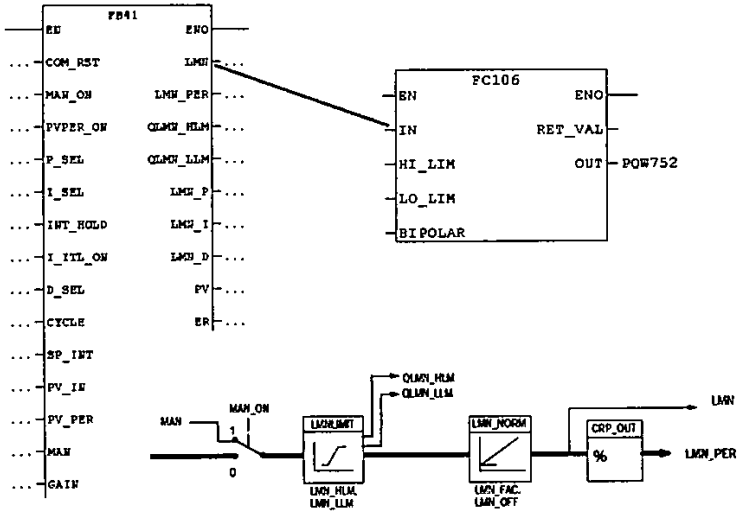
پس از بلوک محدود کننده باز یک بلوک نرمالیزه‌سازی با عنوان LMN_NORM وجود دارد که می‌توان سیگنال تولید شده را در ضریبی ضرب یا با مقدار آفست جمع نمود.

در انتها دیده می‌شود که این کنترلر دارای دو خروجی LMN و LMN_PER است. بحث این دو خروجی شبیه بحثی است که در مورد PV_IN و PV_PER داشتیم. به‌عبارت دیگر:

LMN_PER به‌طور مستقیم به آدرس آنالوگ خروجی که عملگر به آن متصل است اختصاص می‌یابد (مثلاً PQW752). این خروجی همانطور که دیده می‌شود از یک بلوک Unscale به‌نام CRP_OUT که فرمان تولید شده را بین صفر تا 100% برمی‌گرداند استفاده می‌کند. براساس فرمول زیر:

$$LMN\_PER = LMN * \frac{27648}{100}$$

LMN به‌صورت یک عدد Real است که لازم است ابتدا به فانکشن Unscale مانند FC106 داده شده، سپس خروجی FC106 به آنالوگ خروجی اختصاص می‌یابد.



شکل ۸-۴۲ استفاده از خروجی LMN

لیست پارامترهای ورودی خروجی FB41

پارامترهای ورودی FB41

پارامتر	نوع دیتا	مقدار پیش فرض
COM_RST	BOOL	FALSE

این پارامتر Complete Restart است که هرگاه True یعنی 1 شود، حافظه FB یعنی مقادیر ضبط شده در DB ری‌ست شده و با مقادیر جدید شروع به‌کار می‌کند. می‌توان در OB راه‌اندازی FB41 را صدا زد و با 1 کردن ورودی فوق، آنرا برای کار کنترلی آماده ساخت.

پارامتر	نوع دیتا	مقدار پیش فرض
MAN_ON	BOOL	TRUE

این پارامتر لوپ را به‌صورت باز در آورده و تنظیمات P,I,D روی آن بی تاثیر است. در این حالت مقدار ورودی از پارامتر MAN که یک مقدار اعشاری است گرفته می‌شود.

پارامتر	نوع دیتا	مقدار پیش فرض
PVPER_ON	BOOL	FALSE

این پارامتر برای مواردی که مقدار Process Value باید از کارت I/O گرفته شود لازم است ست شود، یعنی به آن مقدار True اختصاص داده شود.

P_SEL	BOOL	TRUE

این پارامتر که به‌صورت پیش‌فرض 1 است، اجازه می‌دهد ضریب Proportional به لوپ اعمال شود. در صورت عدم نیاز به GAIN کافیست آنرا 0 کنیم.

I_SEL	BOOL	TRUE

این پارامتر که به‌صورت پیش‌فرض 1 است، اجازه می‌دهد ضریب INTEGRAL به لوپ اعمال شود. ضریب انتگرال‌گیر توسط TI داده می‌شود. در صورت عدم نیاز به انتگرال‌گیر کافیست آنرا 0 کنیم.

INT_HOLD	BOOL	FALSE

برای HOLD کردن انتگرال‌گیر به‌کار می‌رود و در حالت عادی غیر فعال است. برای فعال کردن کافیست به آن 1 اختصاص دهیم که در اینصورت خروجی انتگرال‌گیر FREEZ می‌شود یعنی تغییر مقدار نمی‌دهد. از این ورودی در پروسه‌های کند که زمان زیادی طول می‌کشد تا به پایداری برسند و در صورت تغییرات سریع مقدار مبنا خطای زیادی توسط انتگرال‌گیر انباشته می‌شود می‌توان استفاده کرد و انتگرال‌گیر را تا نزدیکی نقطه پایداری ثابت و بدون تاثیر نگه داشت.

I_ITL_ON	BOOL	FALSE

برای دوباره از سرگیری یا Initialization انتگرال‌گیر به‌کار می‌رود. اگر فعال شود مقدار انتگرال‌گیری شده با مقدار ورودی I_ITVAL ست می‌شود.

D_SEL	BOOL	FALSE

این پارامتر که به‌صورت پیش‌فرض 0 است اگر 1 شود، اجازه می‌دهد ضریب مشتق‌گیر به لوپ اعمال شود. ضریب مشتق‌گیر توسط TD مشخص می‌گردد.

CYCLE	TIME	T#1s

این پارامتر که از جنس زمان است زمان نمونه برداری را تعیین می‌کند. به‌طور پیش‌فرض ۱ ثانیه است، ولی می‌توان آنرا به 1ms یا بزرگتر تغییر داد، ولی باید توجه کرد که بین این زمان و زمان تنظیم شده برای OB3x رابطه‌ای وجود داشته باشد. اگر زمان OB روی 100ms و Cycle=1s باشد در اینصورت هر ۱۰ بار که OB صدا زده می‌شود یک بار FB اجرا می‌شود.

با استفاده از Cycle می‌توان چند لوپ کنترلی مجزا در یک OB3X قرار داد که زمان نمونه‌برداری آنها متفاوت باشد. ولی همه آنها باید با زمان OB نسبت صحیحی داشته باشند.

SP_INT	REAL	0.0

مقدار مبنا Set Point است که به کنترلر داده می‌شود.

PV_IN	REAL	0.0

مقدار Process Varriable است که به‌صورت داخلی و نه از کارت I/O داده می‌شود.

PVPER	WORD	W#16#0000

مقدار Process Varriable است که از کارت I/O به‌صورت WORD گرفته می‌شود.

MAN	REAL	0.0

مقدار MANUAL است که وقتی کنترلر در مد دستی است به آن اعمال می‌شود.

GAIN	REAL	2.0

ضریب Proportional است و مقدار پیش‌فرض آن ۲ می‌باشد.

TI	TIME	T#20s

ضریب انتگرال‌گیر است و باید TI>=CYCLE باشد.

TD	TIME	T#10s

ضریب مشتق‌گیر است و باید TD>=CYCLE باشد.

TM_LAG	TIME	T#2s

برای ایجاد تأخیر در پاسخ وقتی از مشتق‌گیر استفاده می‌شود به‌کار می‌رود.

DAEDB_W	REAL	0.0

Dead Band فیلتری برای سیگنال ER منتجه از مقایسه SP و PV است.

LMN_HLM	REAL	100.0

حد بالا برای خروجی ایجاد شده توسط کنترل (Manipulated Value)

LMN_LLM	REAL	0.0

حد پایین برای خروجی ایجاد شده توسط کنترل (Manipulated Value)

PV_FAC	REAL	1.0

فاکتوری که برای ضرب مقدار PV_PER و نرمالیزه کردن آن به‌کار می‌رود.

PV_OFF	REAL	0.0

آفستی است که با مقدار PV_PER جمع می‌شود و برای نرمالیزه کردن آن به‌کار می‌رود.

LMN_FAC	REAL	1.0

فاکتوری که برای ضرب مقدار LMN (خروجی کنترلر) و نرمالیزه کردن آن به‌کار می‌رود.

LMN_OFF	REAL	0.0

آفستی است که با مقدار LMN جمع می‌شود و برای نرمالیزه کردن آن به‌کار می‌رود.

I_ITVAL	REAL	0.0

مقداری که برای Initial کردن انتگرال‌گیر به‌کار می‌رود. (در صورتی‌که I_ITL_ON یک شده باشد)

DISV	REAL	0.0

اثر اغتشاش یا disturbance که برای کنترل Feed Forward اعمال می‌شود.

پارامترهای خروجی FB41

پارامتر	نوع دیتا	مقدار پیش‌فرض
LMN	REAL	0.0

Manipulated Value یا خروجی نهایی کنترل

LMN_PER	W#16#0000	0.0

مقدار خروجی نهایی کنترل به‌صورت WORD برای ارسال به کارت I/O

QLMN_HLM	BOOL	FALSE

وقتی خروجی LMN به حد ماکزیمم LMN_HLM برسد این خروجی 1 می‌شود.

QLMN_LLM	BOOL	FALSE

وقتی خروجی LMN به حد مینیمم LMN_LLM برسد این خروجی 1 می‌شود.

LMN_P	REAL	0.0

خروجی تولید شده توسط قسمت Proportional (یعنی پس از اعمال Gain)

LMN_I	REAL	0.0

خروجی تولید شده توسط قسمت انتگرال‌گیر یعنی باکس INT

LMN_D	REAL	0.0

خروجی تولید شده توسط قسمت مشتق‌گیر یعنی باکس DIF

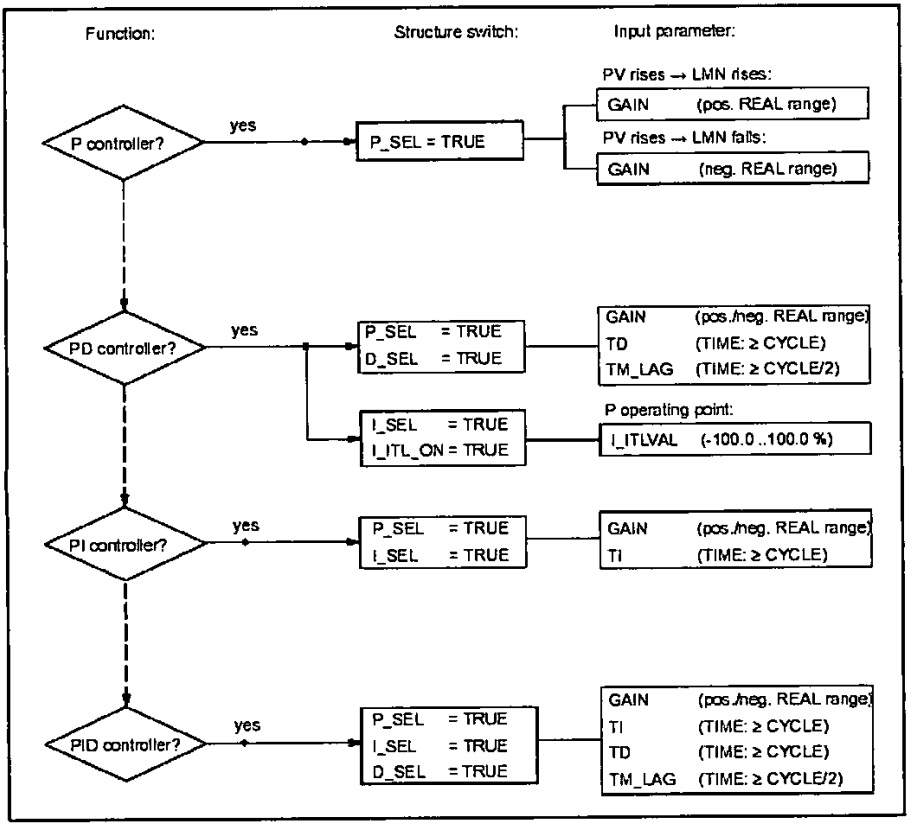
PV	REAL	0.0

مقدار Process Value

ER	REAL	0.0

مقدار سیگنال Error که از مقایسه SP و PV به‌دست آمده است.

فلوچارت شکل ۸-۴۳ می‌تواند راهنمایی برای انتخاب پارامترهای PID باشد.



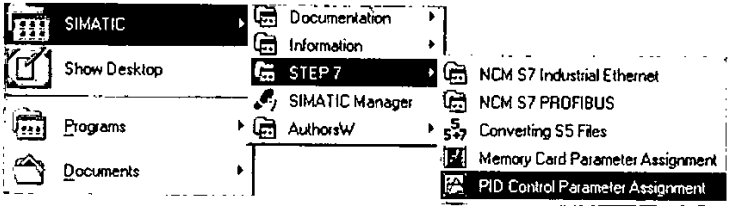
شکل ۸-۴۳ انتخاب پارامترهای PID

کار با زیر برنامه PID Parameter Assignment

پس از نصب Step7 برنامه‌ای با عنوان PID Parameter Assignment نیز نصب می‌شود. توسط این برنامه تمام پارامترهای FB41 در دسترس و قابل تغییر است. کاربرد این برنامه در زمان تنظیم لوپ است. به‌عبارت دیگر به‌جای اینکه

پارامترهایی مانند ضرایب P,I,D را در ورودی FB41 تغییر دهیم، از این برنامه استفاده می‌کنیم و پس از رسیدن به پاسخ مناسب مقدار نهایی پارامترها را در ورودی FB41 به‌صورت ثابت قرار می‌دهیم. برنامه فوق دارای ابزار رسم منحنی است که روند تغییرات PV و SP و LMN را در آن می‌توان دید.

این برنامه را که در زیر شاخه Step7 برنامه نصب شده است می‌توان مشاهده و از آنجا اجرا نمود.

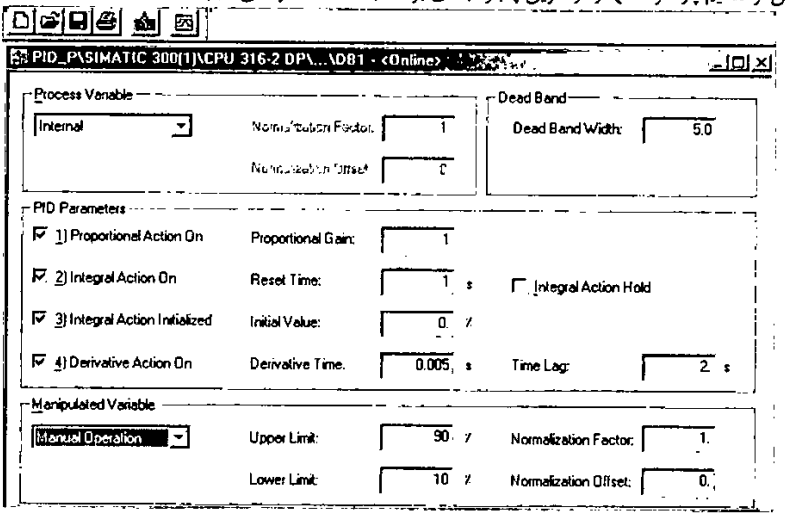


شکل ۸-۴۴ مسیر اجرای برنامه PID Control Parameter Assignment

فصل ۸

پس از اجرای برنامه در حالی که PLC در حال کار بوده و کامپیوتر به آن متصل است، مراحل زیر را انجام می‌دهیم:

- انتخاب File > Open
- انتخاب On Line در پنجره‌ای که ظاهر می‌شود.
- انتخاب دیتابلاک مربوط به فانکشن بلاک PID کنترل
- در این مرحله پنجره‌ای مانند شکل ۸-۴۵ باز می‌شود که توسط آن می‌توان پارامترهایی که برای لوپ کنترل در FB41 وارد می‌کردیم مستقیماً در این پنجره وارد کرد. توجه شود که در اینجا نمی‌توان تعیین کرد که ورودی را از کجا بگیرد یا به کجا بفرستد و صرفاً تغییر پارامترهای لوپ امکان پذیر است.
- پس از تنظیم پارامترها آنها را از طریق پنجره بالای برنامه به PLC دانلود می‌کنیم.



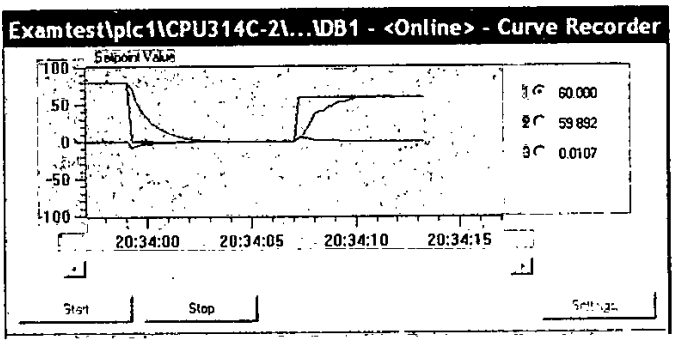
شکل ۸-۴۵ پنجره پارامترهای PID Control Parameter Assignment

پارامترهای فوق را با پارامترهای ورودی FB41 مقایسه کنید.

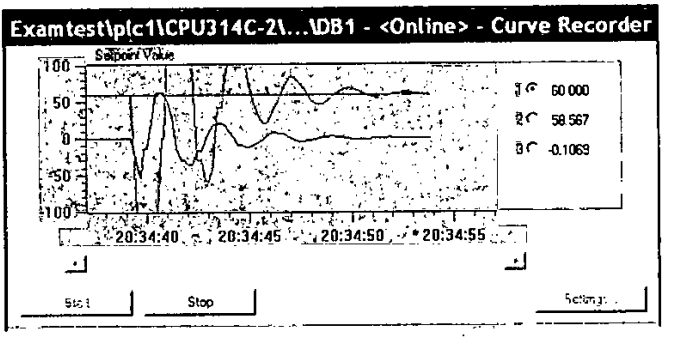
با کلیک کردن روی آیکن Curve Recorder بالای برنامه، پنجره‌ای مانند شکل ۸-۴۶ داریم که اگر روی Start کلیک کنیم، می‌توانیم به‌صورت لحظه‌ای وضعیت موارد زیر را ببینیم:

- Set Point
- Manipulated Varriable
- Process Value

شکل‌های ۸-۴۶ و ۸-۴۷ زیر نمونه‌هایی از منحنی ضبط شده را برای یک کنترلر P و PI نشان می‌دهند.



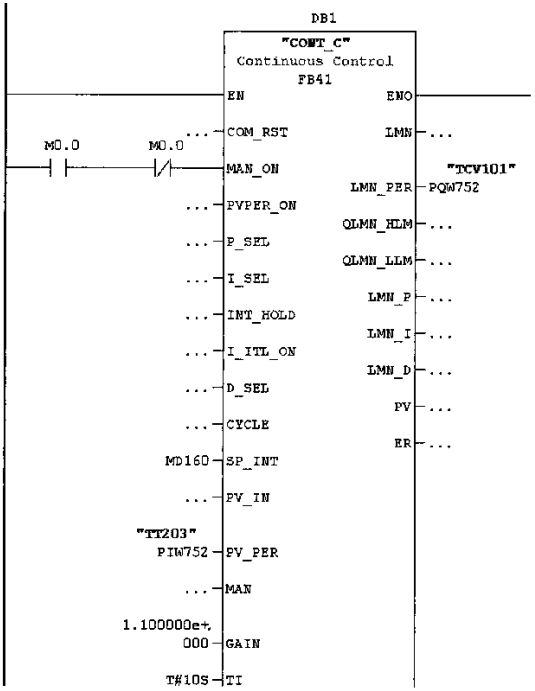
شکل ۸-۴۶ منحنی کنترلر P در برنامه PID Control Parameter Assignment



شکل ۸-۴۷ منحنی کنترلر PI در برنامه PID Control Parameter Assignment

مثال ۸-۱

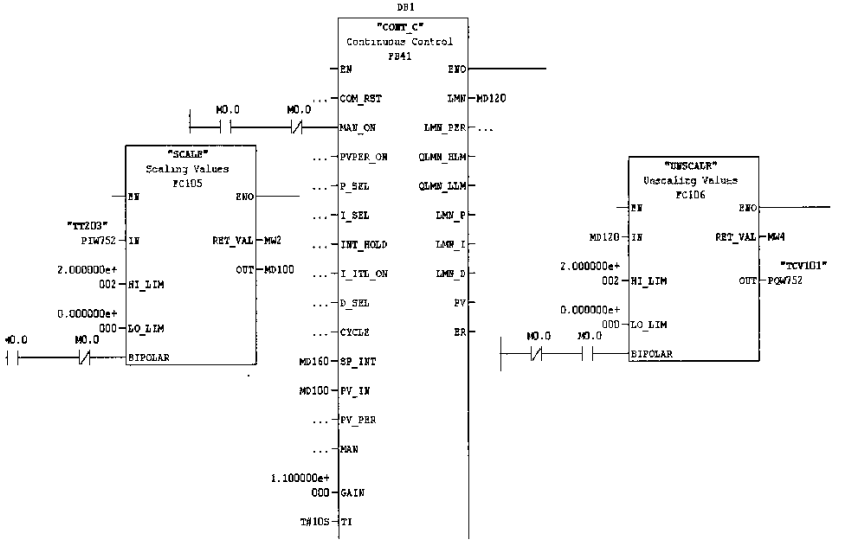
برنامه زیر یک لوپ ساده کنترلی را نشان می‌دهد. سیگنال آنالوگ فیدبک بین 100-0 تغییر می‌کند. مقدار مبنا نیز توسط اپراتور بین 100-0 وارد می‌شود که در MD160 می‌نشیند. به پایه‌های ورودی که در شکل نشان داده نشده‌اند مقداری اختصاص داده نشده است.



شکل ۸-۴۸ برنامه مثال ۸-۱

مثال ۸-۲

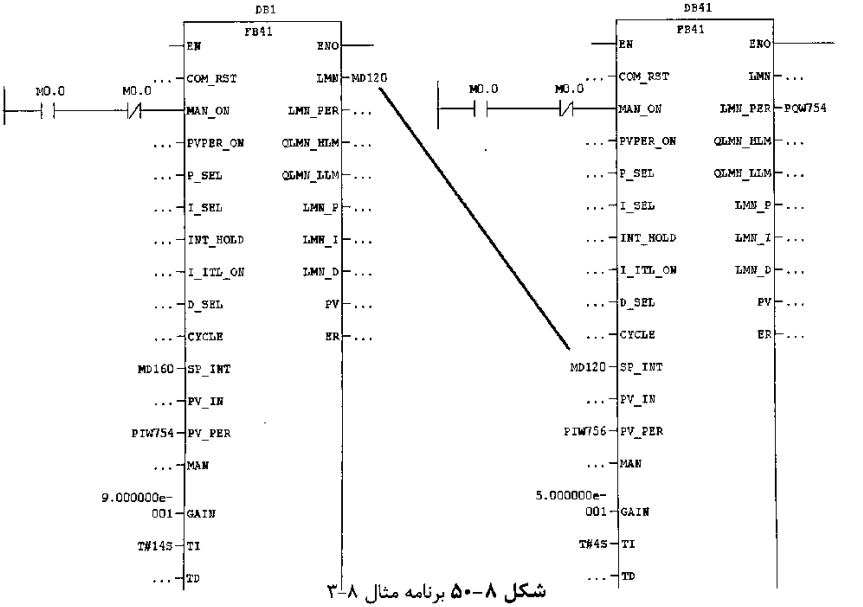
برنامه مثال ۸-۱ وقتی ترانسمیتر بین 0 تا 200 کالیبره شده است به‌صورت زیر خواهد بود. همانطور که در این شکل دیده می‌شود فانکشن‌های Scale و Unscale مورد نیاز هستند.



شکل ۸-۴۹ برنامه مثال ۸-۲

مثال ۸-۳

برنامه زیر نمونه یک کنترل Cascade را که با دو FB41 پیاده‌سازی شده است نشان می‌دهد.



شکل ۸-۵۰ برنامه مثال ۸-۳

شبیه‌سازی و تست لوپ کنترلی با استفاده از سیمولاتور

در عمل لوپ کنترلی توسط فرآیند بسته می‌شود. بدیهی است هر قدر فرمان به عملگر فرایند تغییر کند فیدبک دریافتی از فرآیند نیز متغیر خواهد بود.

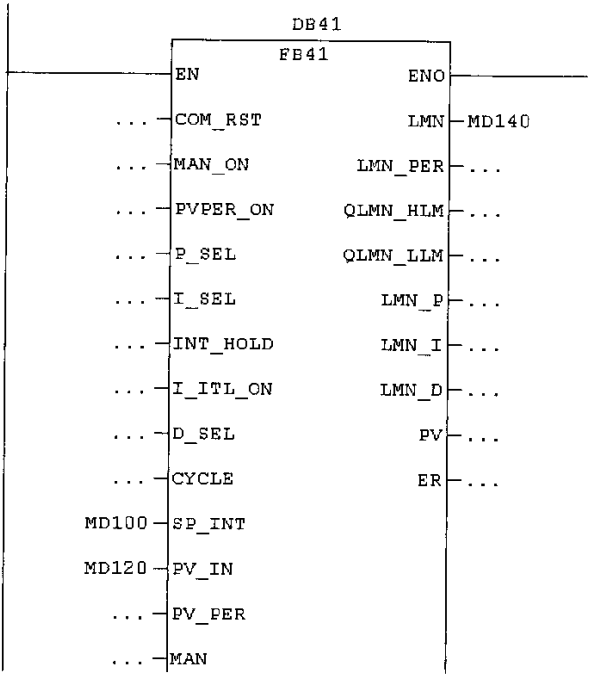
اگر خواننده بخواهد بدون وجود فرآیند و تجهیزات کنترلی رفتار یک لوپ را سیموله نموده و تأثیر پارامترهای P و I و D را بررسی کند، می‌تواند به شکلی که در ادامه ذکر می‌شود یک فیدبک متغیر ایجاد کرده و توسط برنامه PID Control Parameter Assignment آن را تست نماید. این برنامه و توضیحات ذکر شده هیچ ارزش عملی ندارد و صرفاً جهت تست به‌صورت آموزشی با سیمولاتور طراحی شده است.

مراحل کار

۱- ایجاد OB35 و فراخوانی FB41 در آن به‌صورت برنامه زیر. فقط به ورودی‌های SP و PV و خروجی LMN آدرس اختصاص می‌دهیم. سایر ورودی و خروجی‌ها را خالی رها می‌کنیم.

فصل ۸

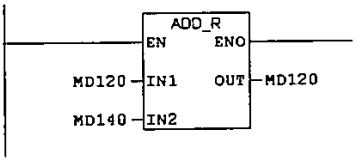
```
OB35 :  "Cyclic Interrupt"
Network 1: Title:
```



شکل ۸-۵۱ FB41 برای کار با سیمولاتور

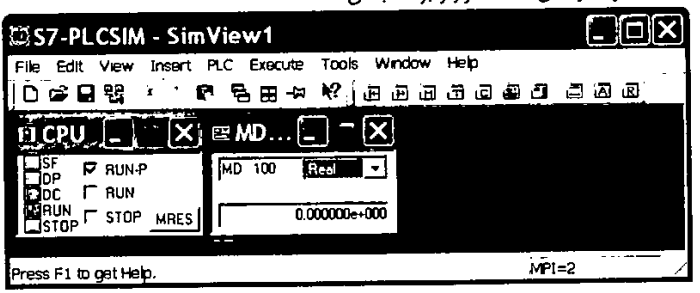
۲- اضافه کردن برنامه زیر در Network2 از OB35. این برنامه منجر به ایجاد یک فیدبک متغیر می‌شود. همانطور که می‌بینید، مقادیر PV و LMN با یکدیگر جمع شده و نتیجه به PV منتقل شده است.

```
Network 2 : Title:
```



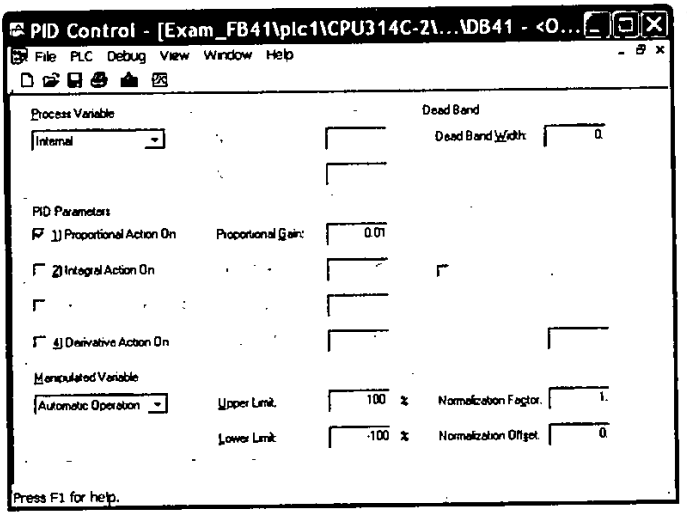
شکل ۸-۵۲ ایجاد فیدبک متغیر برای کار با سیمولاتور

۳- اجرای PLCSIM و دانلود کل سخت افزار و برنامه به آن



شکل ۸-۵۳ استفاده از PLCSIM برای شبیه سازی لوپ

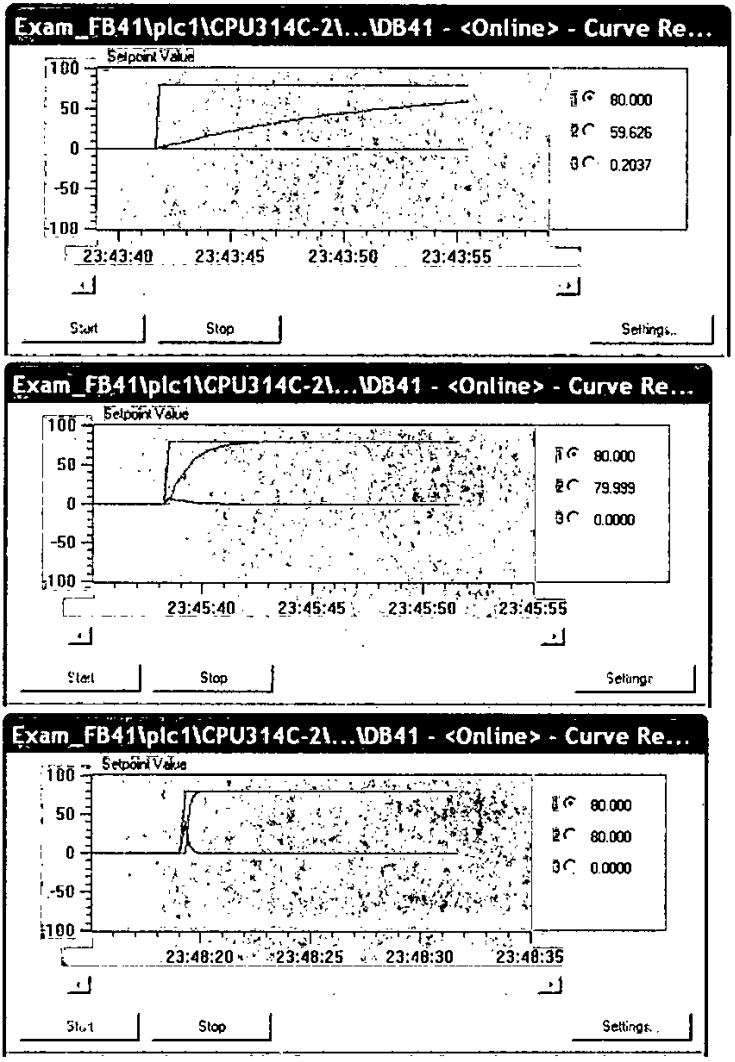
۴- اجرای برنامه PID Control Parameter Assignment و باز کردن DB لوپ به‌صورت Online تا پنجره زیر ظاهر شود.



شکل ۸-۵۴ پارامترهای لوپ برای شروع تست شبیه‌سازی

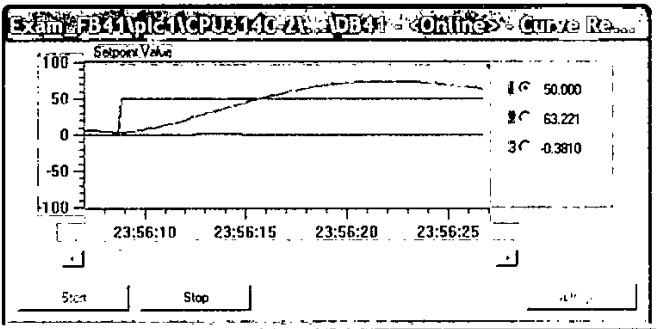
۵- در این پنجره تنظیمات را مانند شکل ۸-۵۴ انجام داده و دانلود می‌کنیم. دقت شود که برای شروع تست فقط P رافعال و I و D را غیرفعال می‌کنیم.

۶- از مقدار Gain = 0.01 شروع می‌کنیم. با تغییر MD100 توسط سیمولاتور پله اعمال می‌کنیم و نتیجه را روی Curve Recorder می‌بینیم. Gain را تدریجاً افزایش داده و نتایج را با هم مقایسه می‌کنیم همانطور که در شکل ۸-۵۵ دیده می‌شود، افزایش Gain سرعت پاسخ را افزایش داده است.

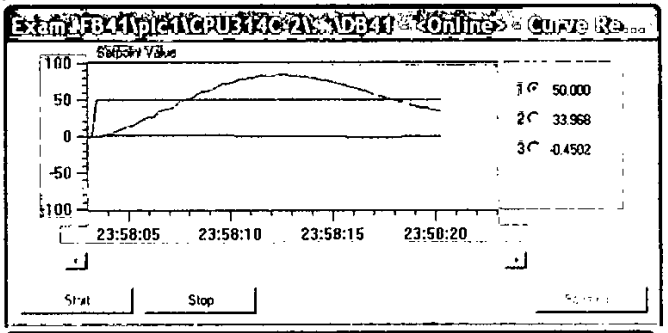


شکل ۸-۵۵ شبیه‌سازی تاثیر Gain روی پاسخ سیستم

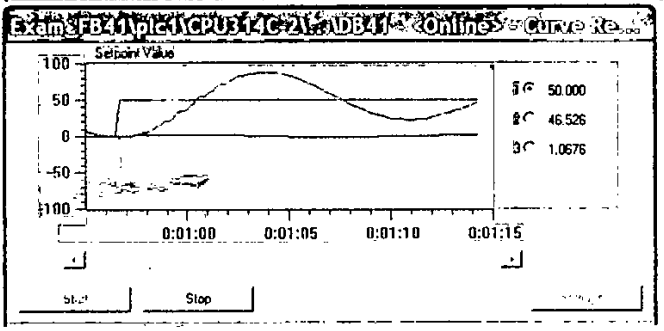
۷- Gain را روی مقدار 0.01 ثابت کرده و انتگرال‌گیر را با مقدار پیش‌فرض Ti=20s وارد می‌کنیم. پله اعمال می‌کنیم ونتیجه را روی رکوردر می‌بینیم. با کاهش تدریجی انتگرال‌گیر مشاهده می‌شود که پاسخ Overshoot پیدا می‌کند و اگر انتگرال‌گیر خیلی کم باشد منجر به نوسان می‌شود. شکل‌های زیر این تاثیر را نشان می‌دهند.



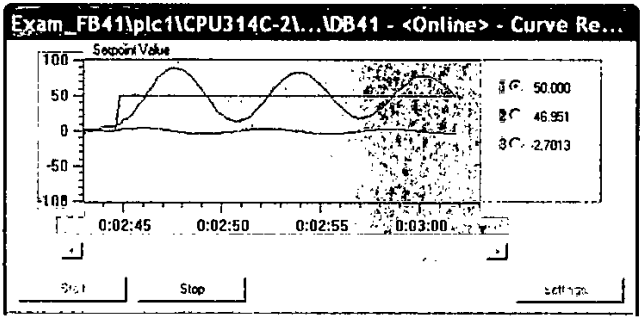
Gain=0.01
Ti=20s



Gain=0.01
Ti=10s



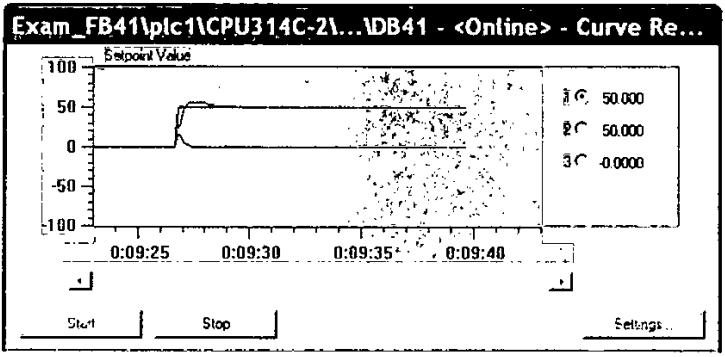
Gain=0.01
Ti=5s



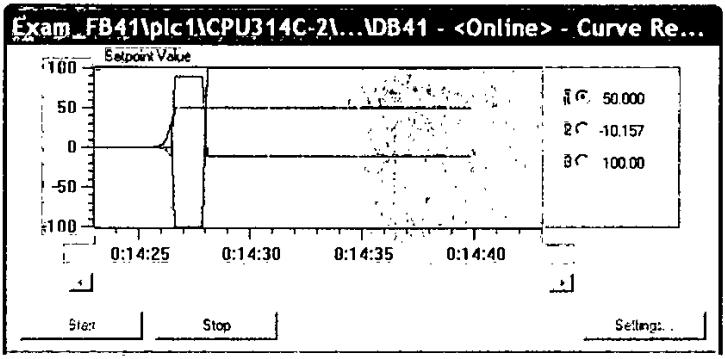
Gain=0.01
Ti=1s

شکل ۸-۵۶ شبیه‌سازی تاثیر Ti روی پاسخ سیستم

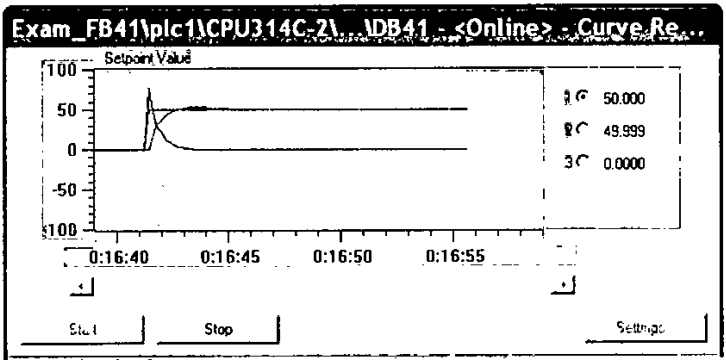
۸- Gain=0.5 و Ti=10s را انتخاب کرده پاسخ را چک می‌کنیم. سپس مشتق‌گیر را که مقدار پیش فرض آن 20s است وارد می‌نماییم. همانطور که دیده می‌شود پاسخ ناپایدار است زیرا خروجی روی مقدار 100 یا 100- ثابت می‌ماند. با کاهش Td این وضعیت بهبود می‌یابد.



Gain=0.5
Ti=10s
بدون
مشتق گیر



Gain=0.5
Ti=10s
Td=10s



Gain=0.5
Ti=10s
Td=5s

شکل ۸-۵۷ شبیه‌سازی تاثیر Td روی پاسخ سیستم

۹- Gain و Ti و Td را به دلخواه به‌صورتی تنظیم می‌کنیم که پاسخ مناسبی داشته باشیم. اکنون Deadband =5.0 انتخاب می‌کنیم. تغییر setpoint را به آرامی انجام می‌دهیم. تا وقتی اختلاف کمتر از 5.0 -/+ است کنترلر هیچ واکنشی نشان نمی‌دهد.

۸-۴-۳ استفاده از FB42 برای کنترل پله‌ای

FB42 برای عملگرهای سه مرحله‌ای[1] استفاده می‌شود، مانند ولوهای موتوری[2] که یکی از سه حالت زیر را دارد:

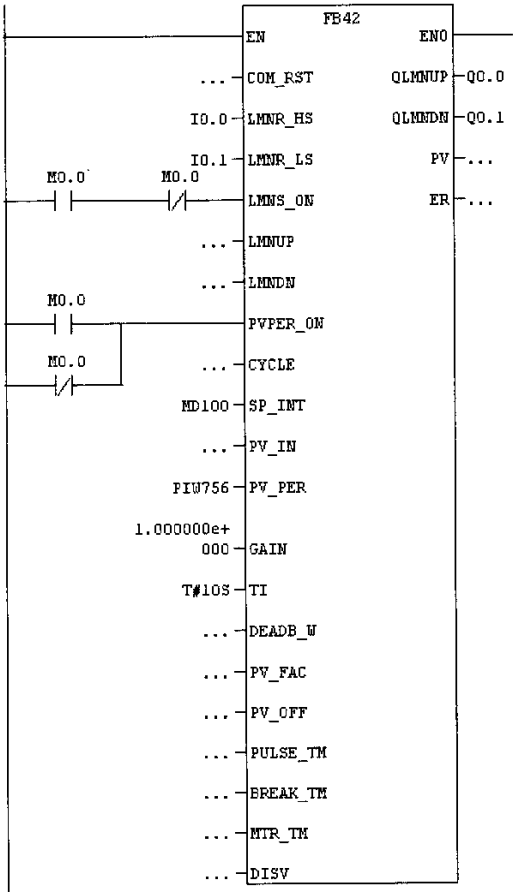
- چرخش Forward
- چرخش Backward
- بدون تغییر وضعیت

FB42 در دیدگاه کلی

این بلاک مقدار مبنا و فیدبک فرآیند را می‌گیرد و با اعمال ضرایب P و I در سیگنال خطا، دو خروجی به‌صورت پالس تولید می‌کند. بنابراین برخلاف FB41 که سیگنال خروجی پیوسته داشت در اینجا سیگنال خروجی به‌صورت صفر و یک است.

FB42 مشتق‌گیر ندارد و PI را ساپورت می‌کند. این بلاک را می‌توان برای کنترل ساده یا کنترل Cascade و برخی دیگر از روش‌های کنترلی به‌کار برد. در روش Cascade این بلاک فقط می‌تواند به‌عنوان کنترلر ثانویه استفاده شود.

1. Three step Actuator
2. Motorized Valve



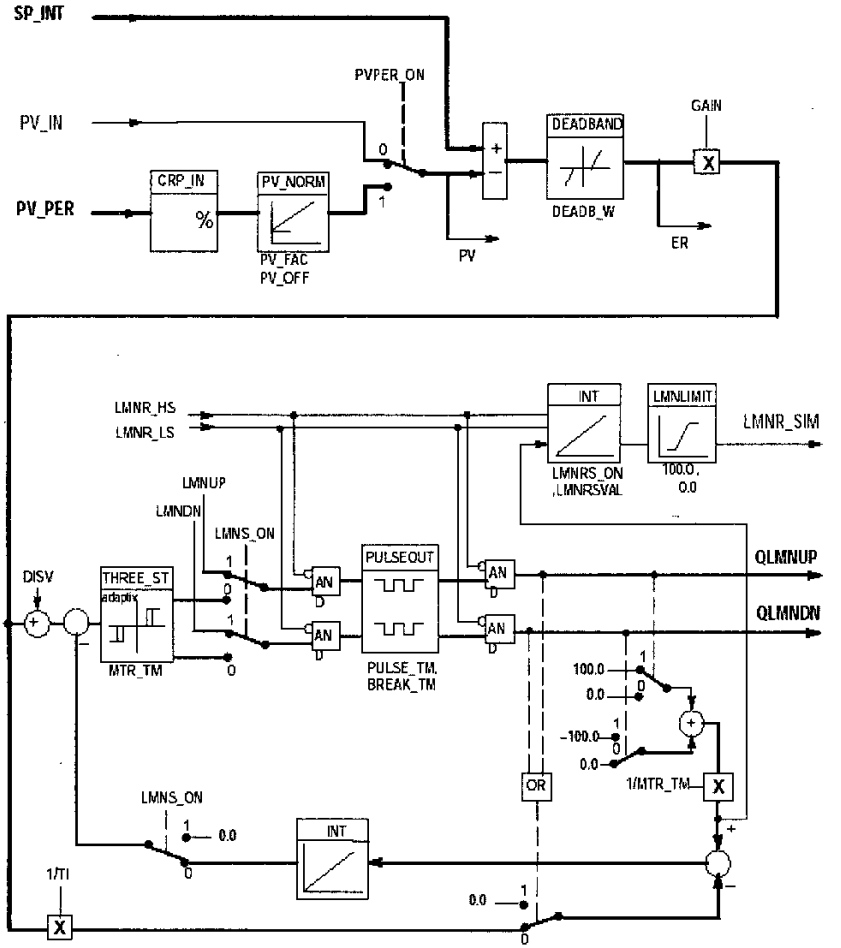
شکل ۸-۵۸ FB42

FB42 در دیدگاه دقیق

بلوک دیاگرام این کنترلر به‌صورت شکل ۸-۵۹ است. همانطور که در شکل دیده می‌شود، برخی از قسمت‌ها مشابه FB41 هستند.

مشابه FB41 فیدبک فرآیند از طریق PV_IN یا PV_PER وارد می‌شود و ضمن مقایسه با مقدار مبنا که به ورودی SP_INT داده شده از فیلتر DeadBand عبور می‌کند و سیگنال Error ساخته می‌شود.

پس از ساخته شدن سیگنال ER ضریب Gain در آن ضرب می‌شود. برخلاف FB41 که سه بخش موازی برای P,I,D وجود داشت و در آن نتیجه آنها با هم جمع می‌شد، در اینجا بخش مشتق‌گیر وجود ندارد.



شکل ۸-۵۹ بلوک دیاگرام FB42

خروجی‌های نهایی FB42

خروجی این کنترلر به‌صورت پالس است که توسط دو بیت خروجی QLMNUP و QLMNDN به عملگر ارسال می‌گردد. با توجه به وضعیت این بیت‌هاست که عملگر حرکت می‌کند مثلاً برای موتور:

جدول ۸-۴

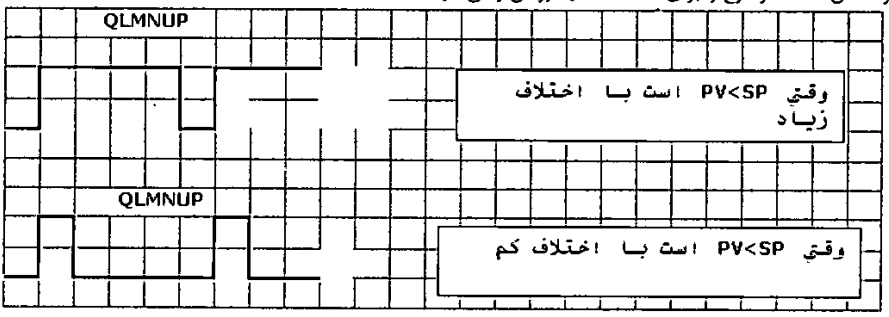
حرکت عملگر	QLMNUP	QLMNDN
Forward	1	0
Backward	0	1
بدون تغییر	0	0

تذکر: وضعیتی که هر دو بیت یک باشند بوجود نمی‌آید.

بدیهی است با توجه به موارد فوق:

- اگر PV<SP باشد، QLMNUP یک می‌شود.
- اگر PV>SP باشد، QLMNDN یک می‌شود.

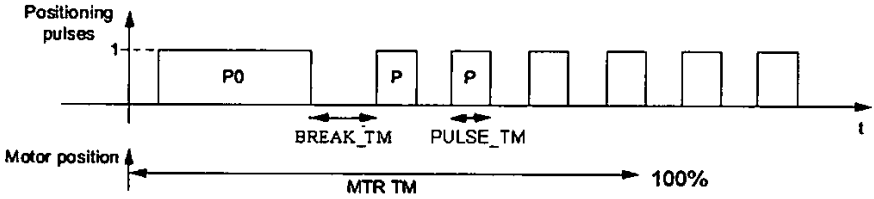
باید توجه داشت که این دو خروجی به‌صورت پالس ظاهر می‌شوند. این گونه نیست که مثلاً وقتی PV<SP است QLMNUP یک شود و یک بماند. بلکه بسته به اینکه چقدر PV با SP اختلاف دارد فرکانس پالس متفاوت خواهد بود.شکل ۸-۶۰ موضوع را برای QLMNUP روشن‌تر می‌سازد.



شکل ۸-۶۰ نمونه پالس خروجی FB42

در حالتی که PV>SP است، دقیقاً همین وضعیت برای QLMNDN نیز وجود دارد.

عرض پالس و زمان وقفه بین دو پالس به پارامترهای تنظیمی که در شکل ۸-۶۱ نشان داده شده و به‌عنوان ورودی FB42 برحسب زمان داده می‌شوند، بستگی دارد.



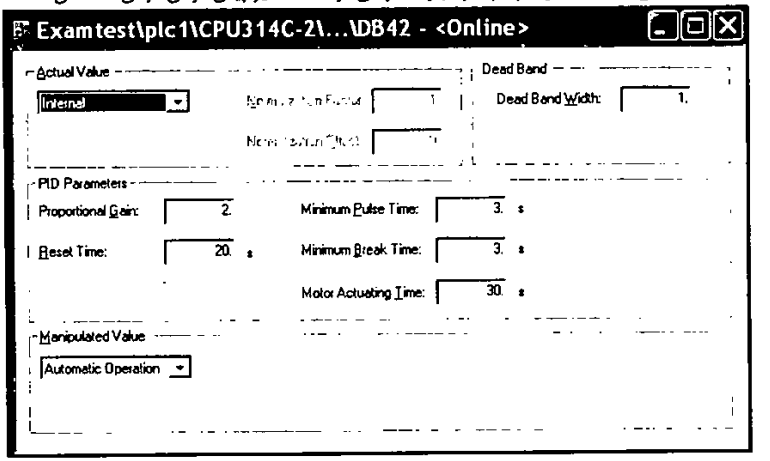
شکل ۸-۶۱ پارامترهای مربوط به پالس FB42

MTR_TM: زمان مورد نیاز برای این که ولو بتواند در حداکثر فاصله مورد نیاز جابه‌جا شود.

PULSE_TM: مینیمم زمان پالس که ولو بر اساس آن بتواند حرکت کند.

BREAK_TM: حداقل زمان مورد نیاز برای وقفه بین دو پالس است؛ یعنی وقتی پالس فعلی قطع شد به اندازه این زمان صبر کند سپس پالس بعدی فعال شود.

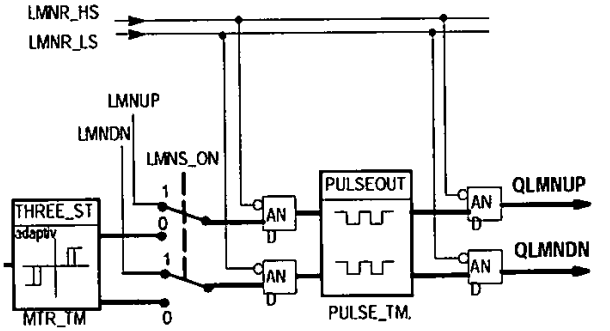
ورودی‌های فوق دارای مقادیر پیش‌فرض هستند. در صورتی‌که برنامه PID Control Parameter Assignment به‌صورتی که قبلا توضیح داده شد اجرا شود، پنجره زیر ظاهر می‌شود که مقادیر پیش فرض در آن مشخص است.



شکل ۸-۶۲ پنجره تنظیمات FB42

تاثیر لیمیت سوئیچ‌های عملگر بر پالس ایجاد شده

دو لیمیت سوئیچ که در عملگر برای موقعیت High و Low تعبیه شده‌اند می‌توانند به‌ترتیب به ورودی‌های LMNR_HS و به LMNR_LS متصل شوند. با توجه به بلوک دیاگرام دیده می‌شود که خروجی پالس‌ساز با معکوس لیمیت سوئیچ‌ها AND شده بنابراین در صورت فعال شدن این ورودی‌ها، فرمان یک بودن از خروجی مربوطه برداشته می‌شود.



شکل ۸-۶۳ تاثیر لیمیت سوئیچ‌های عملگر در FB42

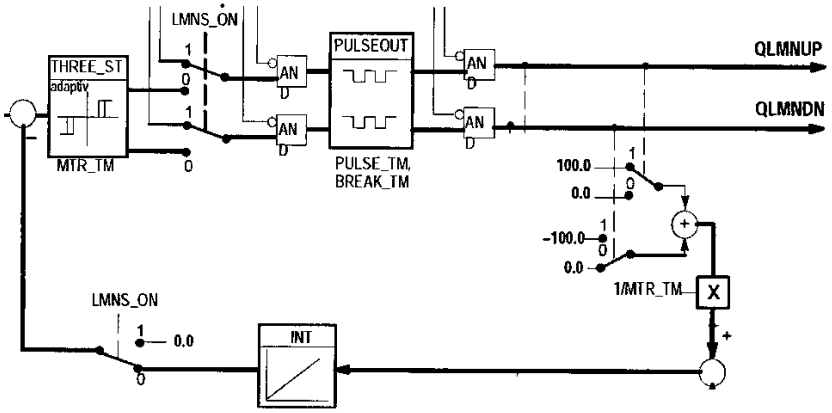
مد Manual

FB42 به‌صورت پیش‌فرض در مد دستی قرار می‌گیرد. ورودی LMNS_ON در حالت عادی یک است و فرمان UP و Down توسط ورودی‌های LMNUP و LMNDN به‌صورت دستی داده می‌شود.

برای حالت Auto لازم است ورودی LMNS_ON صفر شود.

فیدبک موقعیت ولو

FB42 فیدبک را از فرآیند می‌گیرد ولی فیدبک موقعیت ولو را شبیه‌سازی می‌کند. برای این منظور وقتی QLMNUP فعال است عدد 100 و وقتی QLMDN فعال است عدد 100- به جمع کننده داده شده و نتیجه در 1/MTR_TM ضرب شده و از آن انتگرال‌گیری می‌شود.

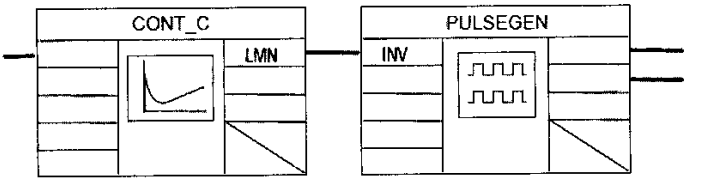


شکل ۸-۶۴ شبیه‌سازی فیدبک موقعیت در FB42

۸-۴-۴ استفاده از FB43

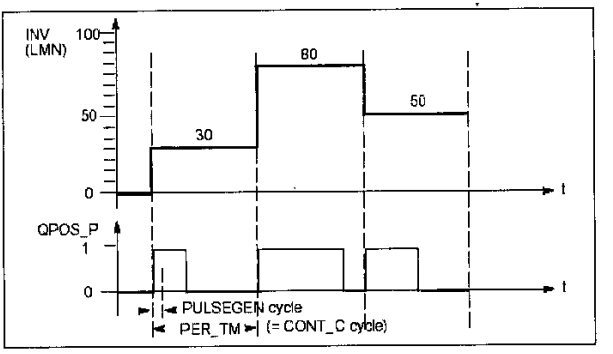
FB43 با نام Pulse_Gen بلاکی است که همراه با FB41 به‌کار می‌رود وهمانطور که از نامش پیداست تولید پالس می‌کند. از اینرو برای عملگرهای سه حالته (3 step) یا دو حالته (2 Step) به‌کار می‌رود. انتخاب نوع عملگر در ورودی‌های این بلاک انجام می‌شود.

پس در نگاه کلی تفاوت FB42 با ترکیب FB41 و FB43 در این است که FB42 فقط برای عملگرهای ۳ حالته به‌کار می‌رفت ولی در اینجا نوع دو حالته نیز ساپورت می‌شود. بعلاوه FB42 فقط به‌صورت PI بسته می‌شد. در اینجا چون از FB41 استفاده می‌شود امکان استفاده از PID وجود دارد.



شکل ۸-۶۵ ارتباط بین FB41 و FB43

با توجه به مقدار LMN پالسی که در دو خروجی FB43 ایجاد می‌شود دارای فرکانس و پهنای خاصی خواهد بود. هر قدر مقدار LMN بیشتر باشد، پهنای پالس بیشتر است. شکل ۸-۶۶ این مطلب را بهتر نشان می‌دهد.



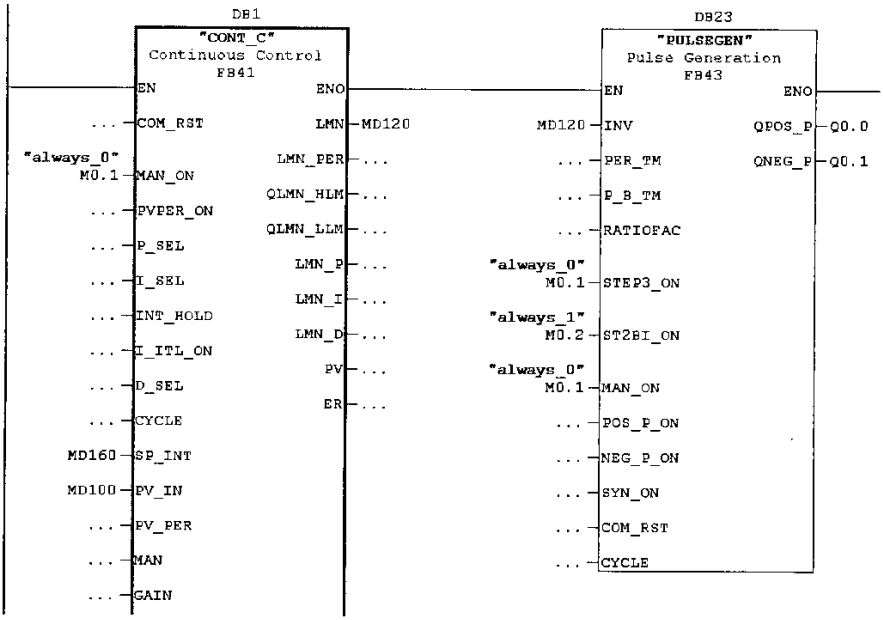
شکل ۸-۶۶ پالس‌های ایجاد شده با FB43

سیکل پالس یا به اصطلاح Period Time آن توسط پارامتر PER_TM تعیین می‌شود. در یک پریود ممکن است FB43 چندین بار فراخوان شود بنابراین در یک PER_TM چندین PULSEGEN Cycle وجود دارد.

به‌عنوان مثال در شکل ۸-۶۶ در حالت اول که مقدار LMN=30% است و طول PER_TM به اندازه ۱۰ سیکل باشد، خروجی QPOS_P به اندازه ۳ سیکل یک است و به اندازه ۷ سیکل صفر است. برای ورودی 80% خروجی به اندازه ۸ سیکل یک و به اندازه ۲ سیکل صفر است.

بدیهی است هر چقدر سیکل PulseGen کمتر باشد دقت سیگنال بالاتر است.

شکل ۸-۶۷ مثالی از ترکیب FB41 و FB43 را نشان می‌دهد.



شکل ۸-۶۷ مثالی از FB43

مدهای کاری FB43

این FB دارای ۴ مد کاری به‌صورت جدول ۸-۵ است که در هر حالت ورودی‌های خاصی از بلاک بایستی یک یا صفر شوند.

جدول ۸-۵

Switch / Mode	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bipolar control range (-100 % to +100 %)	FALSE	FALSE	TRUE
Two-step control with monopolar control range (0 % ... 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

مد کاری Three Step

در این حالت از هر دو خروجی QPOS_P و QNEG_P استفاده می‌شود عملکرد شبیه آنچه برای FB42 ذکر شد می‌باشد. به‌عنوان مثال برای کنترل سرمایش و گرمایش یک سیستم دو خروجی FB43 به‌صورت زیر خواهند بود. همانطور که دیده می‌شود حالتی که هر دو خروجی یک باشند وجود ندارد.

	Heat	Off	Cool
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

مد کاری Three Step به دو نوع تقسیم می‌شود:

- **متقارن:** در این حالت عرض پالس برای سرمایش و گرمایش یکسان است.
- **نامتقارن:** اگر لازم باشد که برای یکی از حالات سرمایش یا گرمایش عرض پالس با دیگری متفاوت باشد از ورودی Ratio استفاده می‌کنیم.

۱- اگر Ratio < 1 باشد، در اینصورت زمان پالس منفی (QNEG_P) در این ضریب ضرب شده و عرض آن کوتاهتر از پالس مثبت (QPOS_P) خواهد بود. به‌عنوان مثال برای Ratio=0.5، عرض پالس حالت منفی نصف عرض پالس حالت مثبت خواهد بود.

۲- اگر Ratio>1 باشد در اینصورت زمان پالس مثبت بر این ضریب تقسیم شده و عرض آن کمتر از پالس منفی خواهد بود. به‌عنوان مثال Ratio=2.0 به مفهوم این است که عرض پالس مثبت نصف پالس منفی خواهد بود.

مد کاری Two Step

این مد کاری فقط برای عملگر دو حالته On/Off استفاده می‌شود. آدرس عملگر به خروجی QPOS_P متصل می‌گردد و همانطور که در جدول۸-۶ دیده می‌شود مد Two Stepدارای دو حالت است:

- **Bipolar:** در این حالت QNEG_P معکوس QPOS_P خواهد بود.
- **Monopolar:** در این حالت QNEG_P صفر است و وضعیت QPOS_P روی آن تأثیر ندارد.

مد Manual

در مد دستی ورودی MAN_ON را یک کرده و فرمان‌های دستی را به POS_P_ON و NEG_P_ON می‌دهیم. بسته به اینکه مدهای 3step یا 2step کدام‌یک انتخاب شده باشند فرمان به‌صورت جدول ۸-۶ خواهد بود.

جدول ۸-۶

	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

سنکرون‌سازی

با توجه به نکات ذکر شده برای PulseGen و PER_TM ممکن است در عمل FB41 و FB43 در دو OB سیکلی مختلف که زمان آنها یکسان نیست به‌کار بروند. برای سنکرون‌سازی اتوماتیک بین این دو ورودی SYN_ON را یک می‌کنند.

۸-۵ نکات مهم در به‌کارگیری بلاک‌های PID Control

Sampling Time

زمان تنظیم شده برای OB3x در واقع زمان نمونه برداری[1] است. فرض کنید این زمان ۱۰۰ میلی ثانیه باشد. در اینصورت هر ۱۰۰ میلی‌ثانیه یک بار با وقفه‌ای که اعمال می‌شود OB فوق اجرا می‌گردد یعنی از فرآیند ورودی می‌گیرد و خروجی‌های تولید شده را به آن می‌فرستد. تنظیم این زمان مسئله مهمی است که برای کنترل لوپ باید مد نظر داشت و هر عددی قابل انتخاب نیست زیرا ممکن است منجر به نوسان سیستم شود. به‌عنوان مثال در یک لوپ سریع مانند کنترل سرعت یا فلو نمی‌توان زمان نمونه‌برداری را بزرگ انتخاب کرد و همینطور در یک لوپ کند مانند کنترل دما نمی‌توان زمان نمونه‌برداری را کوچک گرفت. اما اینکه دقیقاً چه مقداری باشد به دینامیک پروسه بستگی دارد و لازم است در عمل تنظیم گردد. با این وجود دو نکته را به‌عنوان راهنمایی می‌توان ارائه کرد.

۱- زمان نمونه برداری باید از زمانی که طول می‌کشد CPU فانکشن بلاک PID را اجرا کند بزرگتر باشد تا خطای Time Error به‌شرحی که در بخش قبل ذکر شد پیش نیاید.

۲- بهترین مقدار برای زمان نمونه برداری یک دهم مقدار ثابت زمانی سیستم است.

تعداد لوپ کنترلی قابل استفاده در یک OB3x

در یک OB3x می‌توان چندین لوپ را کنترل کرد. برای این منظور FB مربوطه را برای هر لوپ جداگانه CALL می‌کنیم و هر بار DB جداگانه‌ای به آن اختصاص می‌دهیم. این کار اگرچه امکان‌پذیر است ولی توصیه می‌شود فقط برای لوپ‌های مشابه به‌کار رود. مثلاً برای یک لوپ کند و یک لوپ تند استفاده نشود. در این حالت اولین مشکلی که جلب توجه خواهد کرد زمان نمونه‌برداری است که قطعاً نمی‌تواند برای هردو یکسان باشد و بایستی بر اساس لوپ سریع تنظیم گردد که زمان کوتاهتر خواهد بود.

تعداد لوپ‌های کنترلی در یک OB3x تا حدی می‌تواند باشد که زمان اجرای برنامه OB3x از زمان وقفه که در Hwconfig تعیین شده بیشتر نباشد. زمان اجرای برنامه OB3x تابع سرعت پردازش CPU می‌باشد. پس اگر CPU سرعت بالاتری داشته باشد تعداد لوپ بیشتری را می‌تواند کنترل کند.

تعداد لوپ کنترلی قابل استفاده در یک PLC

اگر CPU دارای چند OB3x باشد می‌توان در هر کدام از آنها دسته لوپ‌های مختلفی را قرار داد و تعداد کل لوپ‌ها بیشتر از حالتی است که CPU فقط یک OB3x را پشتیبانی می‌کند.

1. Sampling Time

اما در اینجا باید به این نکته مهم اشاره کرد که حتی اگر CPU دارای سرعت بالا باشد و چندین OB3x را نیز ساپورت کند ولی در عمل PLC را برای تعداد لوپ کم (کمتر از حدود۱۰۰ لوپ) استفاده می‌کنند. اگر تعداد لوپ‌ها زیاد باشد سیستم متمرکز گزینه مناسبی نیست و بهتر است از ساختار DCS استفاده شود.

۸-۶ پرسش و تحقیق

در مورد نحوه کار با نرم‌افزارهای Modular PID Control تحقیق کنید و امکانات آن را بررسی نمایید.

۸-۷ تمرین

با استفاده از دو FB41 یک سیستم Reatio Control برای سیستم احتراق یک کوره طراحی کنید. فرض کنید به ازای هر متر مکعب گاز، پنج متر مکعب هوا لازم است.

فصل ۹

عیب‌یابی در PLC

در این فصل به بررسی انواع اشکالات متداول در PLC پرداخته و نحوه رفع عیب آنها تشریح شده است.

چکیده مطالب

- فاز تست و راه‌اندازی بهترین زمان برای ارتقاء سطح مهارت عیب‌یابی افراد است.
- قدم اول در عیب‌یابی، بررسی پیغام‌ها و علائم ظاهر شده در نرم‌افزار Step7 است.
- مشاهده پیغام‌هایی که در Diagnostic Bufferr مربوط به CPU ثبت می‌شوند کلید یافتن بسیاری از اشکالات است.
- بهترین روش برای عیب‌یابی سخت افزار، مشاهده وضعیت ماژول‌ها در پنجره Online است.
- اشکالات برخی شبکه‌ها مانند Profibus-DP در محیط Hwconfig به‌صورت Online قابل مشاهده است ولی در سایر شبکه‌ها ممکن است هیچ چراغ فالتی روشن نشود و عیب‌یابی بر اساس خروجی‌های فانکشن‌های برنامه‌نویسی شبکه صورت می‌گیرد.
- اشکالات برنامه‌نویسی که منجر به توقف یا فالت CPU می‌شوند از طریق بافر CPU ردیابی می‌شوند.
- اشکالات برنامه‌نویسی که هیچ فالتی ظاهر نمی‌سازند از طریق Reference Data که در فصل بعد تشریح شده مشخص می‌گردند.

فصل ۹

۹-۱ مقدمه

یکی از نیازهای کاربران سیستم‌های اتوماسیون در صنعت آشنایی با روش‌های عیب‌یابی است. افرادی که در فاز طراحی کار می‌کنند نیاز کمتری در این زمینه دارند ولی افرادی که در تست و راه‌اندازی سیستم‌ها و نیز در زمان بهره‌برداری فعالیت دارند به‌شدت این نیاز را حس می‌کنند.

زمان تست و راه‌اندازی دوران بسیار مهمی برای بهره‌برداران سیستم اتوماسیون است. زیرا بسیاری از اشکالاتی که در زمان بهره برداری رخ می‌دهند و نیاز به رفع عیب دارند یک بار در زمان راه‌اندازی نیز رخ داده‌اند. به همین علت افرادی که در زمان تست و راه اندازی حضور دارند تجربه گرانبهایی کسب می‌کنند و در زمان بهره‌برداری بسیار موفق‌تر از سایرین که فقط در دوران بهره‌برداری وارد سیستم شده‌اند خواهند بود.

فاز بهره‌برداری نسبت به فاز تست و راه‌اندازی از نظر عیب‌یابی حساسیت بیشتری دارد و رفع عیب همراه با استرس بیشتر است. علت آن است که در بهره‌برداری اگر به‌دلیل بروز فالت سیستم دچار مشکل یا دچار توقف شود ممکن است خسارات جانی و مالی به‌دنبال داشته باشد. اگر حتی سیستم از نظر ایمنی مشکلی پیدا نکند ولی هر لحظه توقف از نظر اقتصادی خسارت به‌دنبال دارد. با توجه به این شرایط پرسنل بهره بردار بایستی دارای سطح مهارتی باشند که در حداقل زمان ممکن مشکل را برطرف سازند.

یک متخصص اتوماسیون در صنعت معمولاً لازم است بتواند به اجزای مختلف سیستم اتوماسیون سرویس دهی انجام دهد. این اجزا از سطح فیلد که در آن سنسورها و عملگرها نصب شده شروع می‌شود و سطح کنترل که در آن سیستم‌های PLC و DCS قرار گرفته را در برمی‌گیرد و حتی تا سطح مانیتورینگ که ابزارهای اپراتوری در آن سطح قرار دارد نیز ادامه می‌یابد. در این بین وسایل رابط از جمله ارتباطات شبکه نیز جزو مسئولیت‌های متخصص اتوماسیون است.

با توجه به گستردگی این حوزه نمی‌توان همه نکات عیب‌یابی را در این کتاب که در چارچوب PLC بحث می‌کند ارائه نمود. عیب یابی در شبکه‌های صنعتی و وسایل ابزاردقیق نیازمند کتاب‌های جداگانه‌ای است. در اینجا عیب‌یابی فقط از دیدگاه PLC مورد بحث قرار می‌گیرد.

قدم اول در عیب‌یابی

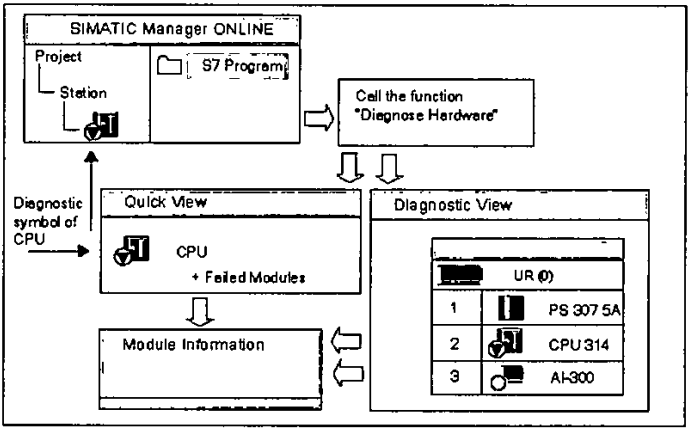
برای شروع بحث لازم است ابتدا دسته‌بندی مناسبی از انواع اشکالاتی که PLC را تحت تاثیر قرار می‌دهد ارائه کنیم. به‌طور کلی می‌توان اشکالات را به دو دسته زیر تقسیم نمود:

۱- اشکالاتی که منجر به روش شدن چراغ فالت CPU یا اجزای دیگر PLC می‌شوند.

۲- اشکالاتی که هیچ چراغ فالتی را روشن نمی‌کنند.

مهمترین کار در عیب‌یابی شناخت محدوده عیب یا Localize کردن آن است. در کار با PLC برای شناخت محدوده عیب استفاده از ابزارهای نرم‌افزاری ضروری است. توصیه‌های مهمی که در این زمینه می‌توان یادآور شد عبارتند از:

- شرایط را تغییر ندهید. به‌هیچ وجه وسایل فیلد، کابل و کانکتورهای شبکه، ترمینال‌ها اتصالات و ... را دست‌کاری نکنید. حتی اگر چراغ فالتی روی یک ماژول روشن است و حدس زده می‌شود که اشکال مربوط به این ماژول است باز هم پیشنهاد می‌شود که قبل از دست‌کاری کردن ماژول، وضعیت آن در نرم‌افزار چک شود.
- برای شناخت محدوده اشکال،در اولین قدم کامپیوتر را به PLC متصل کرده و از ابزارهای نرم‌افزاری که در ادامه توضیح داده می‌شود استفاده کنید.



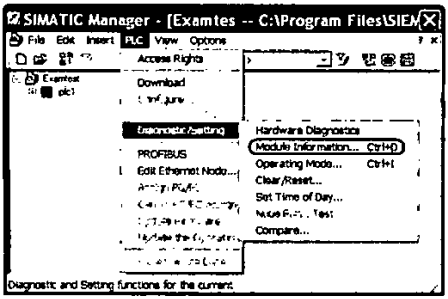
شکل ۹-۱ برخی ابزارهای عیب‌یابی در نرم‌افزار Step7

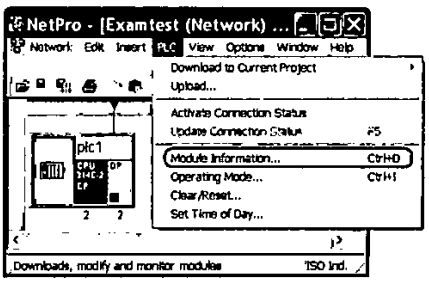
۹-۲ اشکالاتی که منجر به توقف CPU می‌گردند

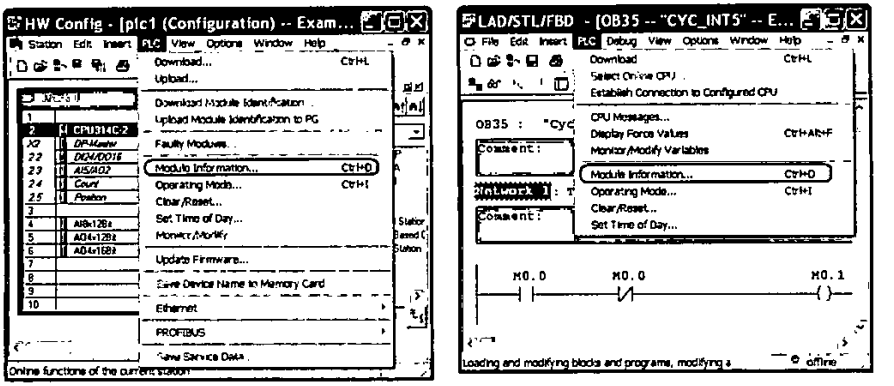
وقتی سیستم در حال کار است، بروز برخی اشکالات می‌تواند منجر به توقف CPU و بالطبع توقف فرآیند تحت کنترل گردد. بدیهی است در این حالت چراغ‌های فالت روی CPU روشن می‌گردد ولی از آنجا که این اشکالات بسیار متنوع هستند نمی‌توان حدس زد که چه مشکلی ممکن است پیش آمده باشد. در این وضعیت بدون اینکه بر اساس حدسیات کاری انجام شود، یا شرایط سیستم به هم بخورد لازم است قبل از هر اقدامی توسط کامپیوتری که برنامه Step7 روی آن نصب شده به PLC متصل شد و محتویات بافر CPU را مشاهده کرد. جهت یادآوری ذکر می‌شود که اطلاعات Diagnostic Buffer در پنجره Module Information قابل دسترس است. این پنجره در منوی PLC برنامه Step7 و زیر برنامه‌های دیگر به‌صورت زیر در دسترس است:

- در برنامه Step7 از مسیر PLC > Diagnostic/Setting
- در برنامه Hwconfig از منوی PLC به‌شرط اینکه قبلاً روی CPU در رک کلیک کرده باشیم.
- در برنامه Netpro از منوی PLC به‌شرط اینکه قبلاً روی CPU مربوط به PLC مورد نظر کلیک کرده باشیم.
- در برنامه LAD/STL/FBD از منوی PLC

تذکر: به‌جای استفاده از منوی PLC می‌توان از کلید میانبر Ctrl+D استفاده کرد.

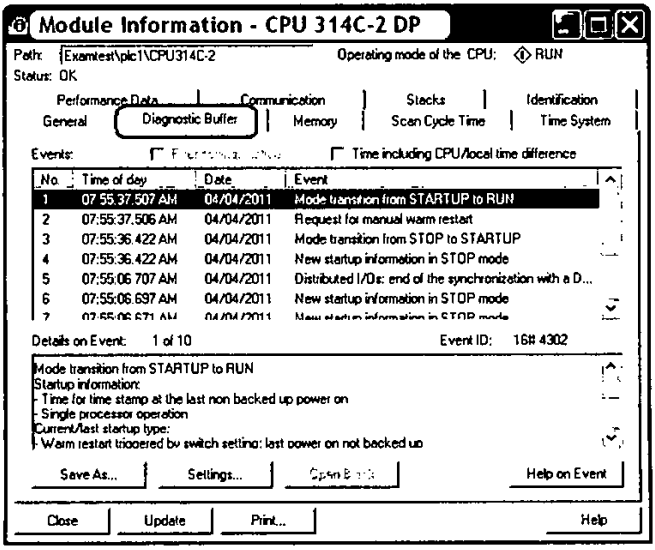






شکل ۹-۲ نحوه فعال‌سازی پنجره Module Information در محیط‌های مختلف

وقتی ارتباط بین کامپیوتر و PLC برقرار است، و زمانی که از روش‌های فوق پنجره Module Information ظاهر شود، در یکی از سربرگ‌های آن Diagnostic Buffer را مانند شکل ۹-۳ مشاهده می‌کنیم.

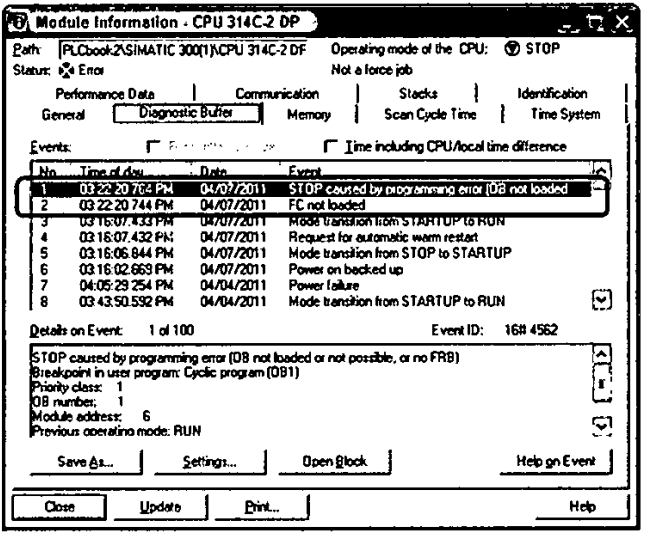


شکل ۹-۳ بخش Diagnostic Buffer در پنجره Module Information

نکات قابل توجه

- همیشه اتفاقات جدید در ابتدا ثبت می‌شوند.
- تاریخ و زمان دقیق در کنار هر Event ثبت می‌شوند.
- تعداد Eventها در S7-300 کم و حدود 100 و در S7-400 زیاد و در حد چند هزار اتفاق است. سعی کنید مقادیر پیش‌فرض را تغییر ندهید زیرا مقادیر بزرگ منجر به افزایش سیکل اسکن می‌گردند.
- وقتی بافر پر شود با وقوع هر اتفاق جدید، قدیمی‌ترین Event از بین می‌رود.
- هر اتفاق دارای یک ID منحصر به‌فرد است. لیست کامل Eventها در پیوست شماره ۳ آمده است.

وقتی CPU به‌دلیل بروز فالت Stop شود، این اتفاق در اولین سطر بافر دیده می‌شود و در زیر آن یعنی در سطر دوم علت فالت ثبت شده است. فاصله زمانی بین فالت و Stop شدن CPU معمولاً چند میلی‌ثانیه است.



شکل ۹-۴ یک نمونه اشکال منجر به Stop

در ادامه ضمن بررسی اشکالات مختلف، نحوه عیب‌یابی از طریق پنجره فوق را تشریح خواهیم کرد.

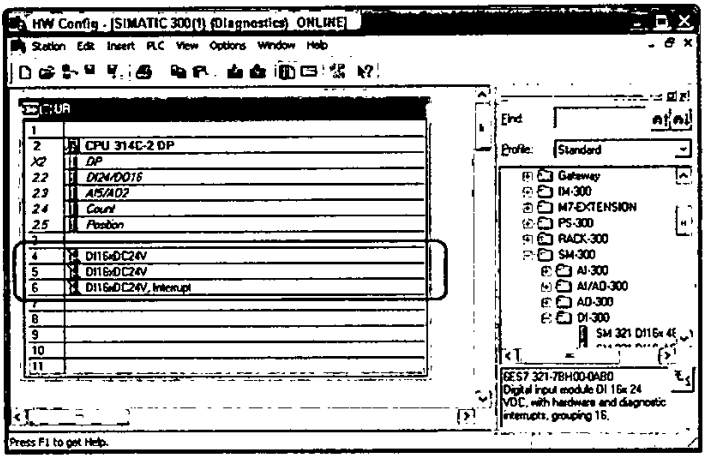
۹-۲-۱ اشکالات سخت‌افزاری

در برخی شرایط، توقف CPU ناشی از اشکالات سخت‌افزاری است. برخی از این اشکالات عبارتند از:

۱- عدم تطابق بین سخت افرار دانلود شده و سخت افزار واقعی

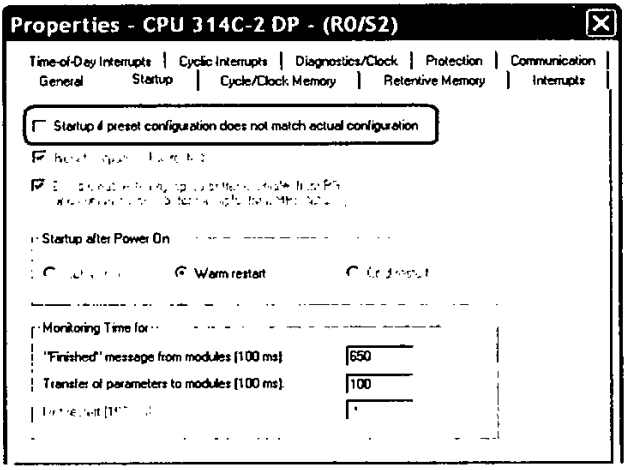
اگر کارت‌های I/O تعریف شده در رک در محیط Hwconfig با آنچه در عمل وجود دارد متفاوت باشد، به‌طور معمول چراغ فالت CPU یک لحظه روشن شده سپس خاموش می‌گردد و CPU به حالت RUN در می‌آید و به‌ظاهر همه چیز نرمال است؛ ولی بدیهی است که CPU نمی‌تواند به کارت‌هایی که غلط معرفی شده‌اند دسترسی داشته باشد.

در این شرایط اگر در Hwconfig حالت Online را فعال کنیم شکل ۹-۵ را خواهیم دید. همانطور که در این شکل دیده می‌شود روی ماژول‌هایی که غلط تعریف شده‌اند خط قرمز کشیده شده است.



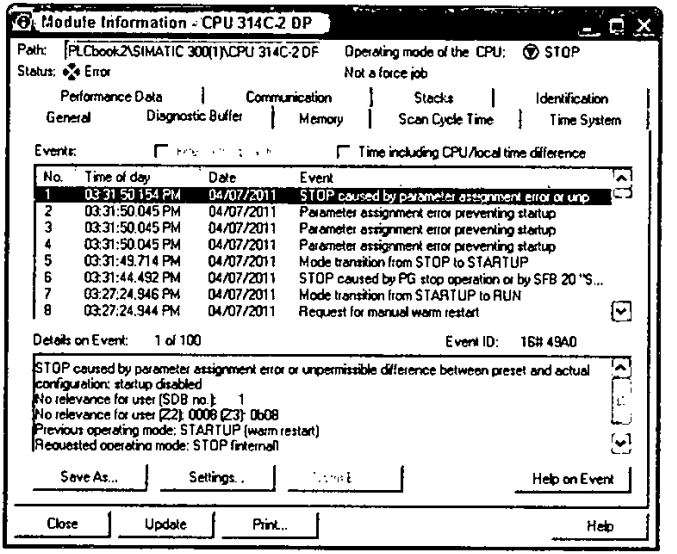
۹-۵ خط قرمز روی ماژول‌هایی که در دسترس CPU نیستند

همانطور که گفته شد، به‌طور پیش‌فرض CPU به این تناقض اهمیتی نمی‌دهد ولی اگر تنظیمی که مانند شکل ۹-۶ در پارامترهای CPU در بخش Startup وجود دارد غیرفعال کنیم CPU حساس می‌گردد و در صورت وجود کوچکترین اختلافی بین سخت افزار واقعی و سخت افزار تعریف شده، متوقف می‌گردد.



شکل ۹-۶ گزینه حساس‌سازی CPU به بررسی اختلاف بین سخت افزار واقعی با سخت افزار تعریف شده

در این شرایط پیغام زیر در بافر ثبت می‌گردد. در Hwconfig در حالت Online نیز خط قرمز را روی ماژول‌ها می‌بینیم.



شکل ۹-۷ پیغام بافر در شرایط عدم تطبیق سخت‌افزار

نکات قابل توجه

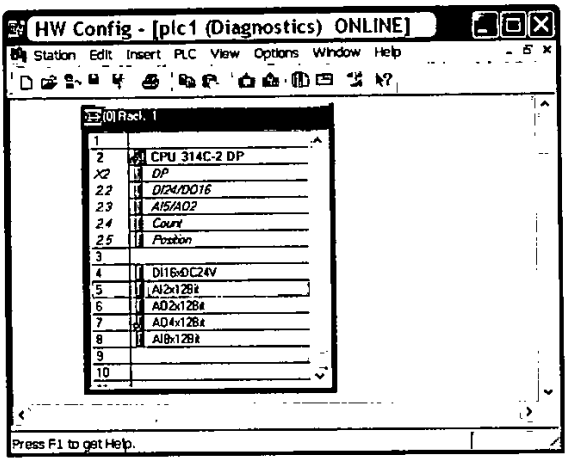
- اگر مدل CPU غلط تعریف شود دانلود به آن امکان‌پذیر نیست. به‌عنوان مثال نمی‌توان سخت‌افزار تعریف شده با CPU 312 را به CPU315 دانلود کرد ولی اگر مدل CPU از همان خانواده باشد ولی Order Number متفاوت باشد یا اگر Version متفاوت باشد امکان دانلود وجود دارد.
- کارت‌های شبکه (مانند کارت اترنت یا کارت پروفی‌باس) اگر در عمل موجود باشند ولی در Hwconfig تعریف نشوند، حتی وقتی که CPU به اختلاف واقعی و تعریف شده حساس نیست بازهم منجر به توقف CPU می‌شود و در بافر پیغام عدم تطابق سخت افزار ثبت می‌گردد.

فصل ۹

عیب‌یابی

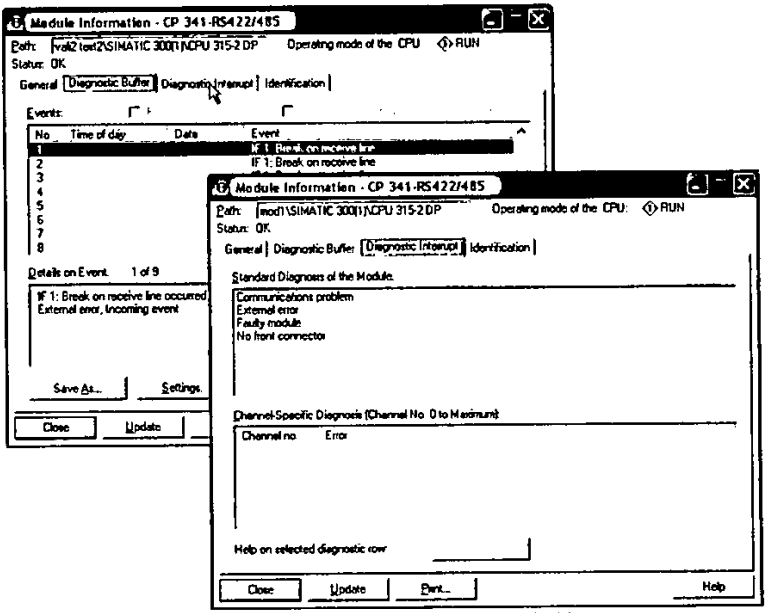
وقتی تعداد زیادی ماژول در عمل وجود داشته باشد عیب‌یابی بر اساس چراغ فالت روی ماژول مشکل و ناقص خواهد بود. به‌طور کلی بهترین ابزار برای عیب‌یابی مشکلات سخت‌افزاری استفاده از Hwconfig در حالت Online است. در حالت Online ممکن است خط قرمز یا دایره قرمز روی ماژول دیده شود.

- خط قرمز شبیه نمونه‌ای که در شکل ۹-۵ نشان داده شده است معرف این است که ماژول در دسترس CPU نیست. یعنی CPUروی باس آن را نمی‌بیند.
- دایره قرمز معرف این است که ماژول در دسترس است یعنی CPU روی باس آن را می‌بیند ولی ماژول دارای فالت است. نمونه‌هایی از این فالت‌ها را در ادامه خواهیم دید.



شکل ۹-۸ وجود دایره قرمز در پنجره Online

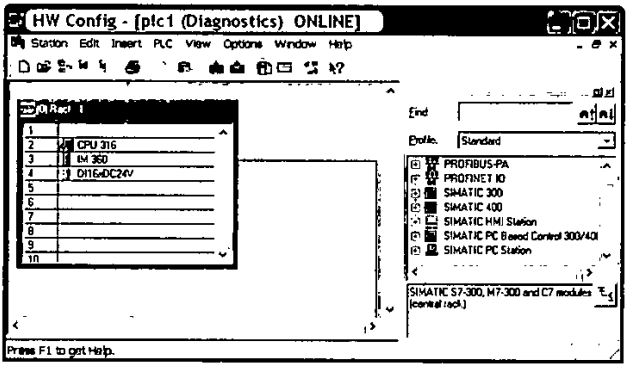
در هر دو حالت فوق یعنی وجود خط قرمز یا دایره قرمز، برای شناخت دقیق عیب بهترین روش آن است که روی ماژول مربوطه در حالت online دوبار کلیک کنیم. اطلاعاتی که در سربرگ Diagnostics نوشته شده برای رفع اشکال بسیار مفید هستند. این کار را می‌توان با کلید میانبر Ctrl+D روی ماژول مربوطه نیز انجام داد. شکل ۹-۹ این پنجره را برای یک کارت مدباس نشان می‌دهد.



شکل ۹-۹ پنجره Module Information برای کارت مدباس

تذکر ۱: پنجره Hwconfig در حالت Online به‌طور خودکار Update نمی‌شود. برای این کار بایستی با کلید F5 یا از طریق منوی View عمل Update را انجام داد.

تذکر ۲: اگر در پنجره Hwconfig در حالت Online خط قرمز یا دایره قرمز وجود نداشته باشد ولی رنگ ماژول طبیعی نبوده و کم رنگ است، این حالت نشان می‌دهد که این ماژول در پنجره Offline تعریف شده ولی به PLC دانلود نشده است. شکل ۹-۱۰ این موضوع را برای کارت DI نشان می‌دهد.



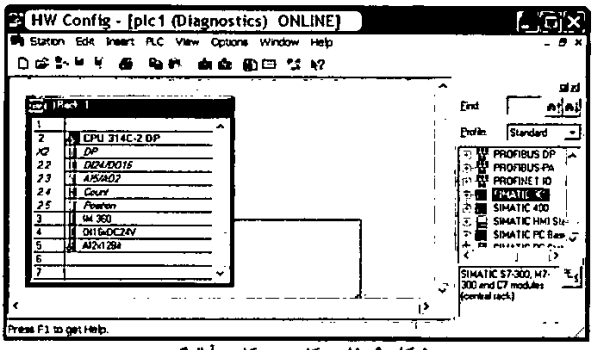
شکل ۹-۱۰ کم‌رنگ بودن ماژول در پنجره Online

۲- فعال کردن وقفه در تنظیمات سخت افزار بدون دانلود OB مربوطه

همانطور که در بحث وقفه‌ها ذکر شد، اگر وقفه‌های زیر در تنظیمات سخت افزار فعال شوند و OB مربوطه دانلود نگردد در شرایطی که وقفه فعال شود منجر به توقف CPU می‌گردد.

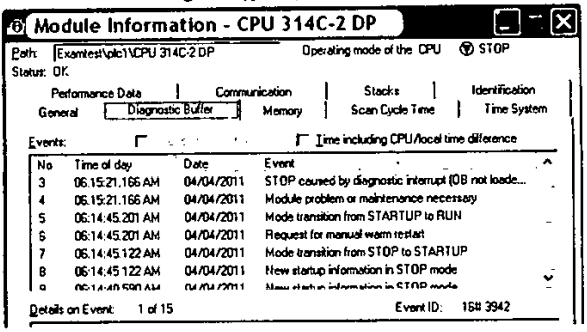
- فعال‌سازی وقفه Diagnostics بدون دانلود کردن OB82
- فعال‌سازی وقفه Hardware بدون دانلود کردن OB4x
- فعال‌سازی وقفه Time of Day بدون دانلود کردن OB1x

شکل ۹-۱۱ وضعیتی را نشان می‌دهد که کارت AI به‌دلیل بروز مشکل در مبدل A/D داخل آن منجر به فعال شدن وقفه Diagnostics شده است. CPU به‌دلیل موجود نبودن OB82 متوقف گردیده است.



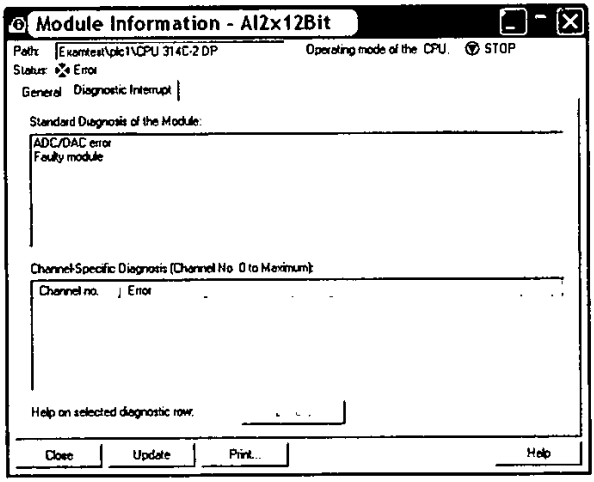
شکل ۹-۱۱ مشکل روی کارت آنالوگ

در این شرایط اگر پیغام ثبت شده در بافر CPU را ببینیم به‌صورت شکل ۹-۱۲ است.



شکل ۹-۱۲ بافر CPU وقتی کارت آنالوگ مشکل دارد

اگر در پنجره Online روی کارت AI دوبار کلیک کنیم، اطلاعات دقیق فالت را در پنجره‌ای مانند شکل ۹-۱۳ خواهیم دید.



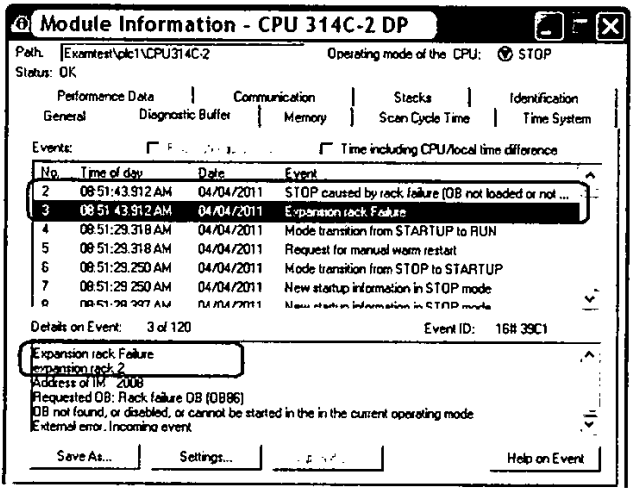
شکل ۹-۱۳ پنجره Module Info مربوط به کارت AI

۳- بروز اشکالات سخت‌افزاری بدون وجود وقفه مربوطه

در بحث وقفه‌ها به بررسی وقفه‌های Asynch. Error پرداخته شد و ذکر گردید که این وقفه‌ها عمدتاً مربوط به اشکالات سخت‌افزاری هستند. برخلاف مورد ذکر شده در بند ۲ قبلی در این حالت کاربر گزینه‌ای را در سخت‌افزار برای وقفه فعال نکرده است بلکه بروز مشکلی در سخت‌افزار منجر به توقف CPU شده است. این اشکالات برخی مربوط به CPU و برخی مربوط به ماژول‌های دیگر است. از اهم این اشکالات می‌توان به موارد زیر اشاره نمود:

- وارد کردن ماژول یا برداشتن ماژول از روی رک در حین کار سیستم بدون وجود OB83
- بروز مشکل Time Error برای CPU بدون وجود OB80
- بروز مشکل Priority Class برای CPU بدون وجود OB85
- بروز مشکل روی یک Slave متصل به پورت پروفی‌باس CPU بدون وجود OB86
- بروز مشکل روی IM رک توسعه بدون وجود OB86

به‌عنوان مثال اگر در حین کار تغذیه IM در رک توسعه قطع شود یا کابل یا کانکتور آن قطع شود و OB86 در CPU دانلود نشده باشد، CPU متوقف شده و پیغام زیر در بافر ثبت می‌گردد.



شکل ۹-۱۴ پیغام مربوط به اشکال در رک توسعه

در این شرایط اگر در بافر روی سطری که پیغام Expansion rack Failure در آن ثبت شده کلیک کنیم، شماره رک که دچار مشکل شده مشخص می‌شود.

لازم به ذکر است که در این حالت چراغ SF روی IM و چراغ فالت روی CPU روشن می‌گردد.

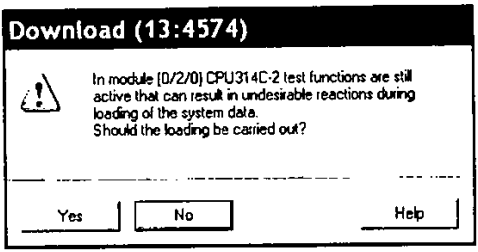
۳-دانلود سخت‌افزار وقتی پنجره Online باز است

دانلود سخت‌افزار در حین کار همیشه منجر به توقف CPU می‌شود. فقط در سیستم‌های افزونه[1] می‌توان بدون توقف فرآیند، سخت‌افزار را به PLC دانلود نمود.

معمولاً پس از دانلود سخت افزار اگر همه چیز به‌درستی تعریف شده باشد نباید مشکلی پیش بیاید و CPU پس از توقف دوباره به وضعیت عادی RUN برمی‌گردد. ولی اگر کاربر دقت نکند و زمانی که می‌خواهد سخت‌افزار را دانلود کند

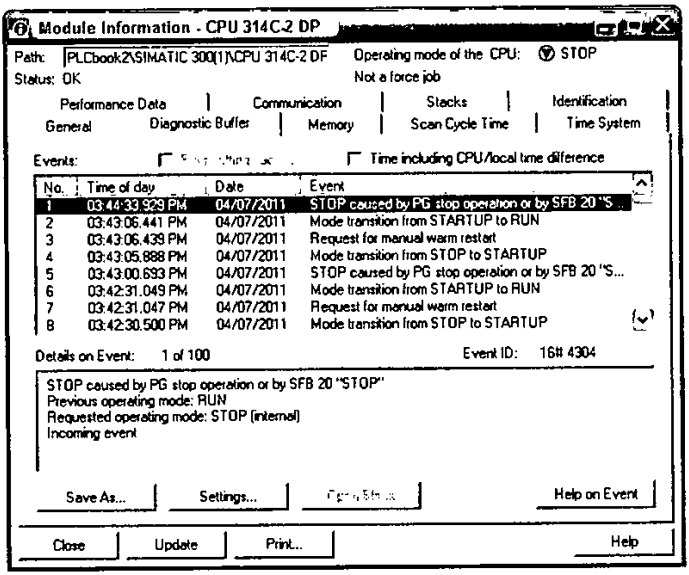
1. Redundant

پنجره Online در پروژه باز باشد مثلاً در محیط برنامه‌نویسی یا در محیط VAT نمایش Monitor فعال باشد، در اینصورت در هنگام دانلود سخت‌افزار با پیغام خطای زیر مواجه می‌شود.



شکل ۹-۱۵ پیغام خطای دانلود سخت افزار وقتی پنجره Online باز است

اگر با وجود اخطار فوق باز اقدام به دانلود سخت‌افزار شود، CPU دچار مشکل شده و RUN نمی‌شود و در بافر پیغام زیر را خواهیم دید.



شکل ۹-۱۶ پیغام بافر در صورت دانلود سخت افزار وقتی که پنجره Online باز است

۹-۲-۲ اشکالات شبکه

بحث اشکالات شبکه مفصل است و در کتاب‌های جداگانه از مولف آمده است. در اینجا به اختصار برخی نکات یاد آوری می‌شود:

- اشکالات شبکه مانند قطعی کابل و کانکتور و قطع شدن تغذیه وسیله و امثال آن فقط در برخی شبکه‌ها آن هم در برخی حالات منجر به توقف CPU می‌شود.
- فقط اشکالات شبکه‌هایی که به‌صورت Master/Slave کار می‌کنند و مستقیماً به پورت‌های روی CPU متصل هستند می‌توانند در عملکرد CPU اختلال ایجاد کنند و در برخی شرایط آنرا متوقف کنند. شبکه Profibus و Profinet از این نوع هستند.
- شبکه‌هایی که مستقیماً به CPU متصل نیستند بلکه از طریق کارت شبکه ارتباط دارند، اگر دچار مشکل شوند فقط روی کارت شبکه تاثیر می‌گذارند و CPU را دچار مشکل نمی‌کنند.
- در شبکه‌هایی که تبادل دیتا با فانکشن‌های Send/Receive انجام می‌دهند اگر آدرس کارت شبکه در فانکشن Send یا Recv غلط باشد منجر به توقف CPU می‌گردد و نظیر عدم دسترسی به آدرس‌های Peripheral، پیغام I/O Access Error در بافر ثبت می‌گردد.

تذکر: برای عیب‌یابی شبکه‌های Master-Slave نظیر پروفی‌باس، ابزارهای نرم افزاری و سخت افزاری جداگانه‌ای نیز عرضه شده است.

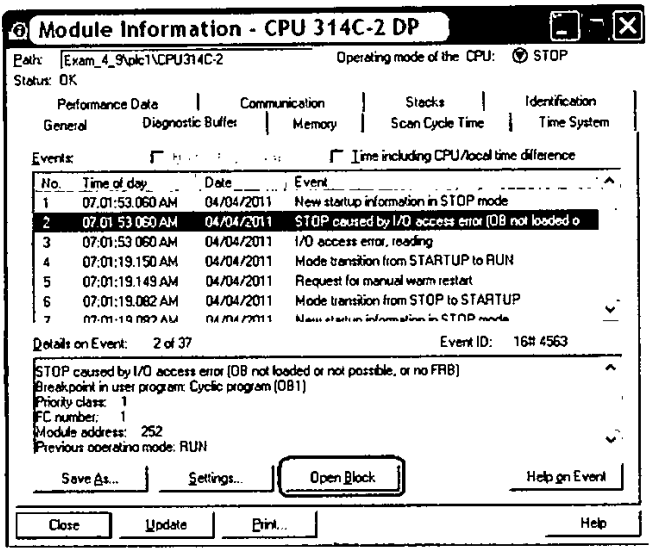
فصل ۹

۹-۲-۳ اشکالات برنامه‌نویسی

همانطور که در بحث وقفه‌ها ذکر شد، در صورتی‌که OBهای وقفه OB121 و OB122 در CPU موجود نباشد، برخی اشکالات برنامه‌نویسی می‌تواند منجر به توقف CPU شود. برخی موارد عبارتند از:

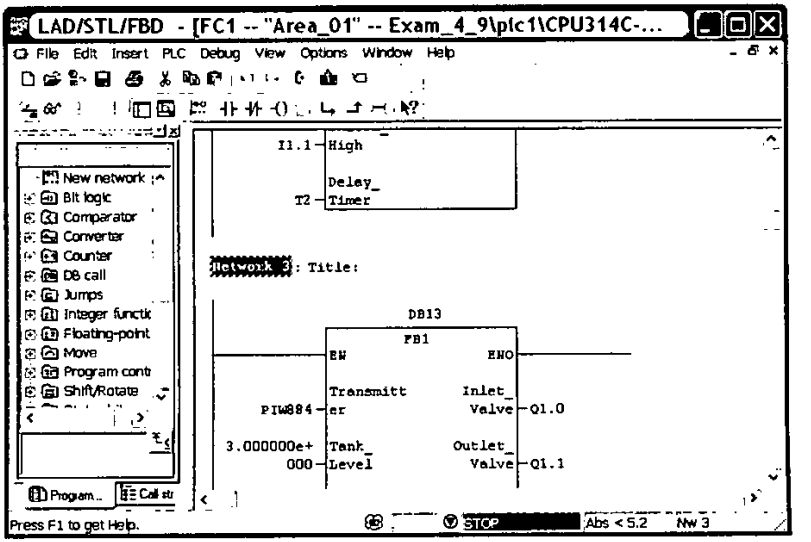
- آدرس غیر مجاز در S7-400 (مانند آدرس خارج از رنج برای ورودی، خروجی، تایمر، ...)
- عدم دسترسی به آدرس‌های Peripheral وقتی آدرس‌دهی به‌صورت PIW یا PQW باشد.
- فراخوانی FB یا FC که قبلاً دانلود نشده باشد.
- استفاده از آدرس دیتابلاکی که قبلاً دانلود نشده باشد.
- خطا در تبدیل BCD.
- و ...

در این حالت نیز اشکال به‌وجود آمده در بافر CPU ثبت می‌گردد. شکل ۹-۱۷ نمونه ای از اشکالات برنامه‌نویسی که منجر به توقف CPU شده است را نشان می‌دهد.



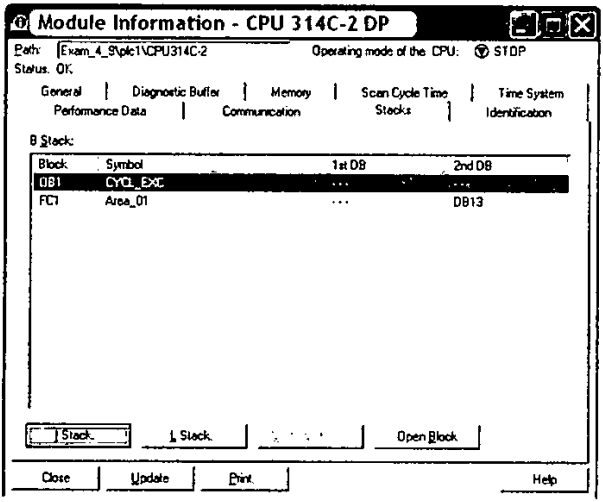
شکل ۹-۱۷ بافر CPU برای یک نمونه اشکال برنامه‌نویسی

برای شناسایی سطری که در آن این اشکال بوجود آمده است، در پنجره بافر ابتدا روی سطری که در آن پیغام Stop درج شده کلیک می‌کنیم تا گزینه Open Block در پایین پنجره فعال شود. با کلیک روی این گزینه بلاک برنامه‌نویسی مربوطه باز شده و به‌طور خودکار Network حاوی دستور مشکل‌دار مانند شکل ۹-۱۸ نمایش داده می‌شود.



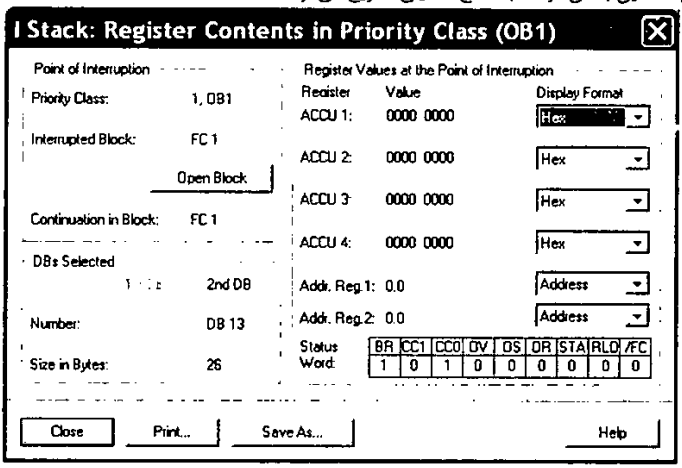
شکل ۹-۱۸ دسترسی به محل اشکال در برنامه‌نویسی که منجر به توقف شده است

روش دیگر برای یافتن مشکل برنامه‌نویسی استفاده از سربرگ Stacks در پنجره Module Information مانند شکل ۹-۱۹ است.



شکل ۹-۱۹ سربرگ Stacks در پنجره Module Information

شکل ۹-۱۹ نشان می‌دهد که در برنامه پس از اینکه FC1 در OB1 فراخوان شده مشکل به‌وجود آمده است. با کلیک روی I Stack در پایین پنجره فوق شکل ۹-۲۰ ظاهر خواهد شد که جزئیات بیشتری از شرایط CPU را در لحظه وقوع فالت نشان می‌دهد. این بخش در کتاب سطح تکمیلی تشریح می‌گردد.



اشکل ۹-۲۰ پنجره مربوط به Interrupt Stack

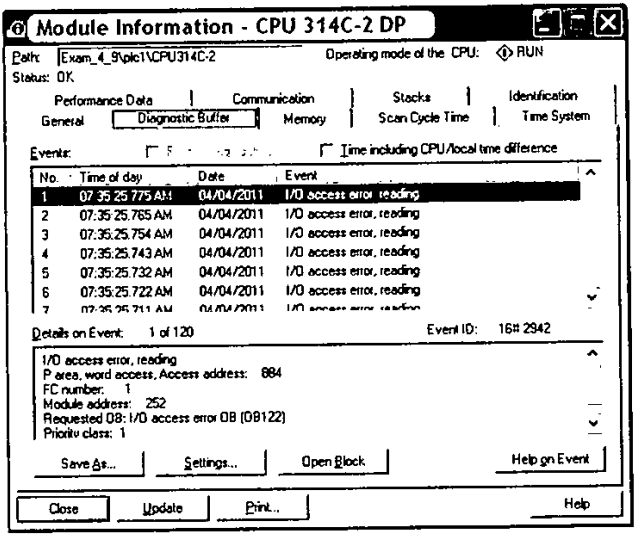
۹-۳ اشکالاتی که فقط چراغ فالت را روشن می‌کنند

۹-۳-۱ فالت و اختلال در کار کنترل بدون توقف CPU

در برخی حالات با وجود اینکه چراغ فالت CPU روشن است، ولی CPU در وضعیت RUN است و فقط بخشی از کار کنترل فرآیند مشکل دارد و کنترل قسمت‌های دیگر انجام می‌شود.

با توضیحاتی که در قسمت قبل و در فصل وقفه‌ها داده شد، مشخص است که چنین وضعیتی وقتی اتفاق می‌افتد که OB وقفه مربوط به فالت در CPU موجود باشد. با وجود این وقفه خطای به‌وجود آمده منجر به توقف CPU نخواهد شد.

به‌عنوان مثال فرض کنید که یک آدرس آنالوگ مشکل داشته باشد و OB122 نیز در CPU موجود باشد. در این شرایط CPU در سیکل اسکن وقتی به آدرس فوق می‌رسد چون نمی‌تواند آنرا بخواند OB122 را صدا زده و پیغام I/O Access Error را در بافر می‌نویسد، سپس به ادامه اجرای سیکل می‌پردازد. این کار در سیکل بعدی نیز تکرار می‌شود از اینرو محتوای بافر با پیغام فوق پر می‌شود.



شکل ۹-۲۱ پیغام‌های بافر در شرایط بروز یک نمونه مشکل با وجود OB وقفه

در این حالت آدرسی که مشکل دارد در پایین پنجره نوشته شده و با کلیک روی Open Block نیز می‌توان به سطر برنامه که این آدرس در آن به‌کار رفته پرش نمود.

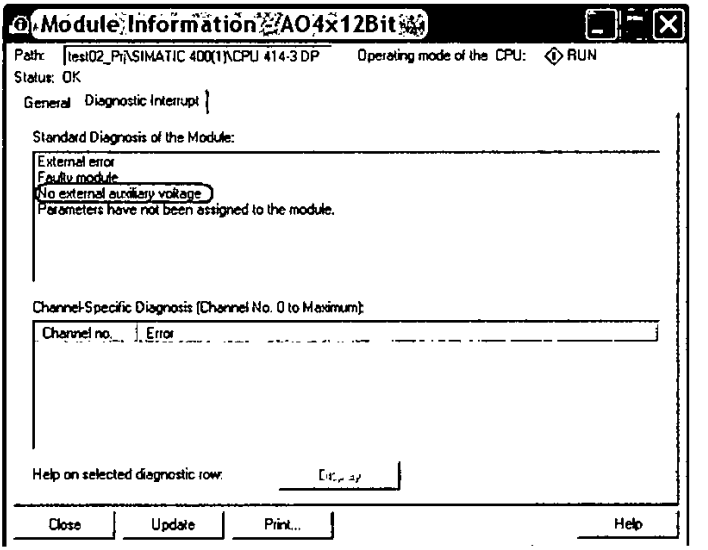
تذکر: اگر چه فیلتر شدن خطا در این حالت اجازه ادامه کار به CPU را می‌دهد ولی ممکن است آدرسی که مشکل دارد مربوط به ترانسمیتر مهمی باشد که عدم وجود سیگنال آن، کار کنترل را می‌تواند مختل کند. در این شرایط مدیریت خطا بایستی با برنامه‌نویسی وقفه OB122 که در فصل قبل تشریح شد انجام گیرد.

۹-۳-۲ فالت روی CPU و عدم اختلال در کار کنترل

در CPUهای نوع 400 که دارای دو چراغ فالت EXTF , INTF هستند در برخی شرایط فقط چراغ EXTF روشن ولی INTF خاموش است. این حالت نشانگر این است که فالت روی یکی از ماژول‌های بیرون از CPU رخ داده ولی CPU در پردازش برنامه خود هیچ مشکلی ندارد. یکی از مواردی که این حالت را ایجاد می‌کند فالت باتری منبع تغذیه است که در این حالت علاوه بر چراغ‌های فالت منبع تغذیه، چراغ فالت EXTF روی CPU نیز روشن می‌گردد.

۹-۳-۳ فالت روی سایر ماژول‌ها بدون تاثیر روی عملکرد CPU

در شرایطی که وقفه Diagnostic کارت فعال نیست برخی اشکالات چراغ فالت روی CPU را روشن می‌کنند ولی روی CPU هیچ چراغ فالتی روشن نمی‌گردد. به‌عنوان مثال برای کارت آنالوگ ورودی در S7-300 اگر تغذیه کارت قطع شود یا تنظیمات مربوطه به Selection Module نادرست باشد، چراغ SF کارت روشن می‌شود. این گونه اشکالات اگر چه تاثیری روی CPU ندارند ولی نتایج برنامه‌ای که از کارت فوق‌الذکر استفاده می‌کند مشکل خواهد داشت. در کارت‌های آنالوگ این قبیل اشکالات منجر به Overflow یا Underflow شدن سیگنال می‌شوند و در فانکشن SCALE FC105 خروجی RET_VAL را تحت تاثیر قرار می‌دهند که از روی آن می‌توان از اشکال با خبر شد. برای مشاهده جزئیات خطا می‌توان در Hwconfig از Module Information کارت مورد نظر استفاده نمود. شکل ۹-۲۲ خطای ناشی از قطع شدن تغذیه یک کارت آنالوگ خروجی را نشان می‌دهد.



شکل ۹-۲۲ Modul Info کارت AO4x12bit که تغذیه آن قطع شده است

در مثال فوق اگر Diagnostic ماژول فعال نباشد در محیط Hwconfig در حالت Online هیچ علامت قرمز رنگی روی کارت AO نمی‌بینیم. ولی با این وجود پیغام فوق در پنجره Module Info کارت قابل مشاهده است.

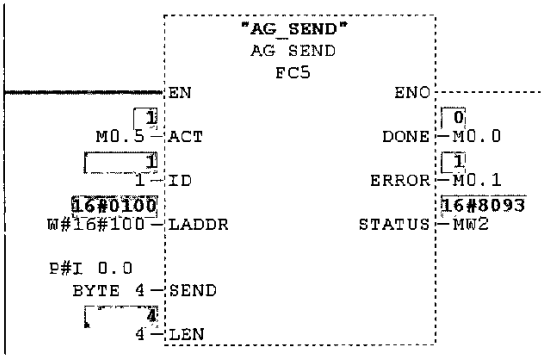
۹-۴ اشکالاتی که هیچ اثر ظاهری ندارند

دو دسته از اشکالات هستند که اثر ظاهری روی سخت افزار ندارند یعنی هیچ چراغ فالتی روی هیچ ماژولی روشن نیست ولی برنامه جواب نمی‌دهد. یکی مربوط به اشکالات برنامه‌نویسی و دیگری مربوط به اشکالات تبادل دیتا روی شبکه است.

۹-۴-۱ اشکالات شبکه

در شبکه‌هایی که تبادل دیتا به‌صورت Master – Master انجام می‌شود اگر یک وسیله دچار مشکل شود هیچ فالتی روی وسیله مقابل ظاهر نخواهد شد. از شبکه‌هایی نظیر Profibus-DP و Modbus و Industrial Ethernet می‌توان برای تبادل دیتای Master- Master استفاده کرد.

برای شناسایی این گونه اشکالات معمولاً از خروجی‌های مربوط به فانکشن‌های ارسال و دریافت استفاده می‌شود. شکل ۹-۲۳ نمونه‌ای از فانکشن Send مربوط به تبادل دیتا روی شبکه را نشان می‌دهد. همانطور که در این شکل دیده می‌شود این فانکشن دارای یک خروجی با عنوان Error می‌باشد که از جنس Bool است. زمانی که ارتباط برقرار است این خروجی صفر و زمانی که مشکلی وجود دارد این خروجی یک می‌شود. با انتقال این بیت به محیط مانیتورینگ می‌توان از وقوع خطا روی شبکه سریعتر مطلع گردید. شایان ذکر است که وقتی خروجی Error یک می‌شود کد خطا در خروجی STATUS به‌صورت یک کد هگز نشان داده می‌شود که می‌توان با استفاده از جداول راهنما به جزئیات خطا پی‌برد.



شکل ۹-۲۳ اشکال‌یابی شبکه براساس خروجی فانکشن ارسال دیتا

بحث شبکه و جزئیات خطایابی آن در کتاب‌های جداگانه آورده شده است.

۹-۴-۲ اشکالات برنامه‌نویسی

برخی اشکالات برنامه‌نویسی بدون اینکه هیچ اثر ظاهری به‌جای بگذارند و بدون اینکه چراغ فالتی را روشن کنند یا بیت خاصی را به‌عنوان Error یک کنند، منجر به اختلال در کار کنترل می‌گردند. از دیدگاه عیب‌یابی اینها جزء بدترین نوع اشکالات هستند. ولی باید توجه داشت که این اشکالات بیشتر در زمان طراحی برنامه و در فاز تست و راه‌اندازی دیده می‌شوند و پس از اینکه سیستم در بهره‌برداری قرار گرفت معمولاً به‌ندرت کاربران با این نوع اشکالات مواجه می‌شوند.

یکی از ابزارهای سودمند برای بررسی این اشکالات استفاده از سیمولاتور و جدول VAT است. به همین علت است که بسیاری از برنامه‌نویسان ابتدا منطق طراحی شده را با سیمولاتور چک می‌کنند تا اشکالات اولیه آن و از جمله اشکالات فوق‌الذکر را شناسایی و برطرف نمایند.

از نمونه اشکالات فوق‌الذکر می‌توان به موارد زیر اشاره نمود:

- استفاده از تایمر با شماره‌های تکراری در برنامه
- استفاده از کانتر با شماره‌های تکراری در برنامه
- نوشتن روی M,Q,DB در چند نقطه از برنامه
- تلاقی بین آدرس‌ها در برنامه

برای شناسایی این اشکالات ابزارهایی در نرم‌افزار Step7 وجود دارد. بهترین ابزار استفاده از Cross Reference است که نحوه استفاده از آن در فصل بعد تشریح گردیده است.

۹-۵ اشکالات ارتباط بین PLC و PC

۹-۵-۱ برقرار نبودن ارتباط

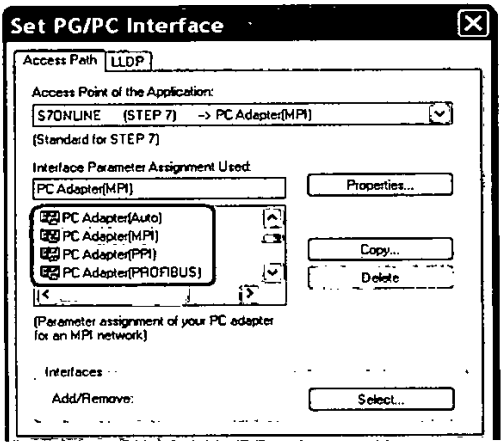
گاهی پیش می‌آید که PLC در حال اجرای برنامه است و هیچ مشکلی ندارد ولی کاربر نیاز دارد که کامپیوتر را به PLC متصل نموده و به کارهایی نظیر دانلود، آپلود و عیب‌یابی بپردازد ولی ارتباط برقرار نمی‌شود. برای رفع این نوع اشکالات به نکات زیر توجه کنید.

فصل ۹

۱- ارتباط با پورت MPI

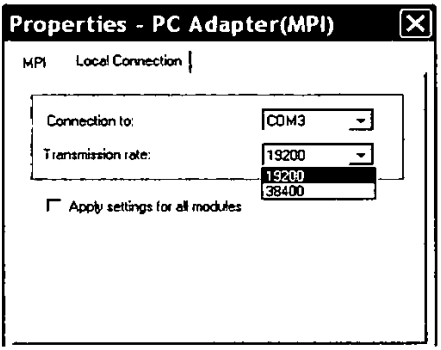
اگر از آداپتورهای قدیمی که به پورت COM کامپیوتر متصل می‌شوند استفاده می‌کنید موارد زیر را چک کنید:

- آیا چراغ آداپتور روشن است؟ اگر خاموش است کابل و اتصالات را بررسی کنید.
- آیا تنظیم Set PG/PC به‌درستی انجام شده است؟ این تنظیم بایستی مانند شکل ۹-۲۴ روی حالت Auto یا MPI باشد.



شکل ۹-۲۴ تنظیم PC Adapter

- آیا در تنظیمات پورت COM به‌درستی انتخاب شده است؟ این تنظیم با توجه به پورت ارتباطی بایستی صحیح باشد. به شکل ۹-۲۵ توجه کنید.

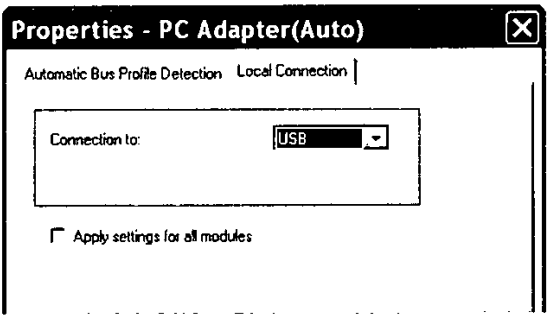


شکل ۹-۲۵ تنظیمات پورت COM برای PC Adapter

- آیا وضعیت دیپ سوئیچ روی آداپتور صحیح است؟ این دیپ سوئیچ دو وضعیت دارد 19200 و 38400 که بایستی با تنظیم موجود در شکل ۹-۲۵ یکسان باشد.

اگر از PC Adapterهای جدید که به USB متصل می‌شوند استفاده می‌کنید و مشکلی در ارتباط وجود دارد، موارد زیر را چک کنید:

- آیا نرم‌افزار درایور آداپتور نصب شده است؟
- آیا چراغ‌های روی آداپتور همگی روشن هستند؟ اگر همه خاموش باشند مشکل در ارتباط بین آداپتور با PLC است و اگر فقط چراغ USB خاموش است مشکل در ارتباط بین آداپتور و کامپیوتر است.
- آیا تنظیم Set PG/PC مانند شکل ۹-۲۶ صحیح است.



شکل ۹-۲۶ تنظیمات پورت USB برای PC Adapter

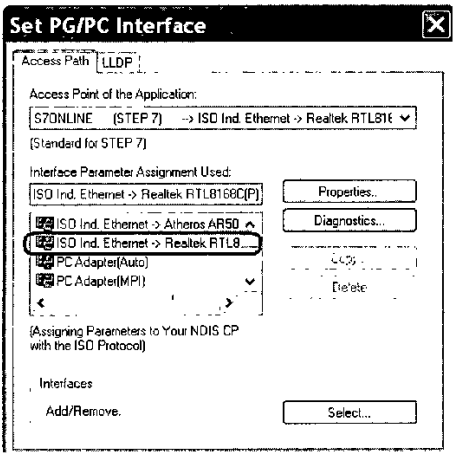
۲- ارتباط با پورت DP روی CPU

در ارتباط با پورت Profibus-DP روی CPU لازم است توجه شود که در اولین ارتباط نمی‌توان آداپتور را به آن متصل کرد. لازم است ابتدا پورت فعال شود، برای این کار بار اول ارتباط MPI را برقرار کرده و سخت‌افزار سیستم که در آن شبکه پروفی‌باس فعال شده باشد را تعریف و دانلود می‌کنیم. پس از آن می‌توان اتصال آداپتور به پورت DP را برقرار نمود. به‌شرط اینکه تنظیمات روی Auto یا Profibus که در شکل قبل نشان داده شده است، باشد.

۳- ارتباط با کارت شبکه اترنت

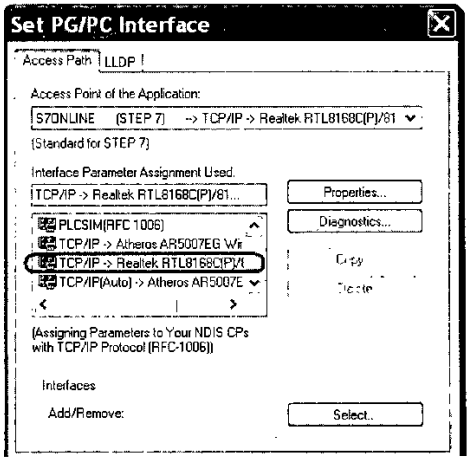
اگر روی PLC کارت شبکه اترنت وجود داشته باشد، می‌توان با توجه به نکات زیر از آن برای ارتباط با کامپیوتر استفاده نمود:

- در S7-300 ابتدا لازم است سخت افزار که در آن کارت اترنت تعریف شده و آدرس MAC کارت اترنت در آن مشخص شده توسط ارتباط MPI به PLC دانلود شده باشد. پس از آن می‌توان از طریق اترنت ارتباط برقرار کرد.
- در S7-400 می‌توان از ابتدا با اترنت از طریق MAC آدرس ارتباط برقرار کرد.
- برای ارتباط اترنت از طریق MAC لازم است تنظیم Set PG/PC به‌صورت شکل ۹-۲۷ باشد



شکل ۹-۲۷ تنظیم ارتباط MAC Address اترنت

- در صورتی‌که قبلاً آدرس IP را برای کارت اترنت تعریف کرده و دانلود کرده باشیم، می‌توان ارتباط اترنت را با آدرس IP برقرار کرد به‌شرط اینکه تنظیم Set PG/PC به‌صورت شکل ۹-۲۸ باشد.

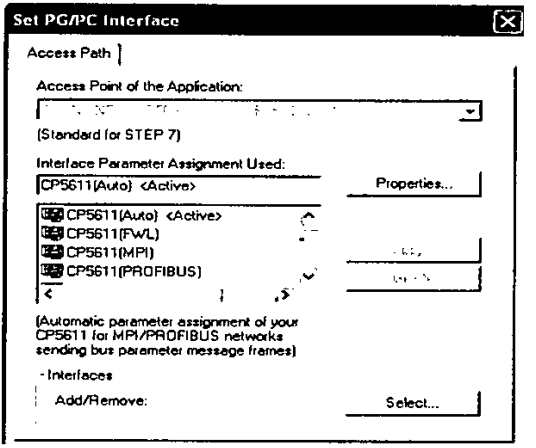


شکل ۹-۲۸ تنظیم ارتباط IP Address اترنت

در ارتباط اترنت می‌توان کابل اترنت را مستقیماً به کامپیوتر و PLC متصل کرد یا می‌توان از طریق سوئیچ این کار را انجام داد. در هر صورت بایستی مطمئن شد که ارتباط برقرار است. در سمت PLC روی کارت اترنت بایستی چراغ Link روشن باشد و در سمت کامپیوتر بایستی لینک شبکه برقرار باشد.

۴- ارتباط با کارت شبکه پروفی‌باس

اگر روی PLC کارت شبکه Profibus-DP مانند CP342-5 نصب شده باشد، نمی‌توان از طریق آداپتور به این کارت متصل شد. ولی اگر روی کامپیوتر کارت پروفی‌باس مانند CP5611 نصب شده باشد از طریق آن می‌توان به کارت پروفی‌باس PLC متصل شد. تنظیم کارت CP5611 در شکل ۹-۲۹ نشان داده شده است.

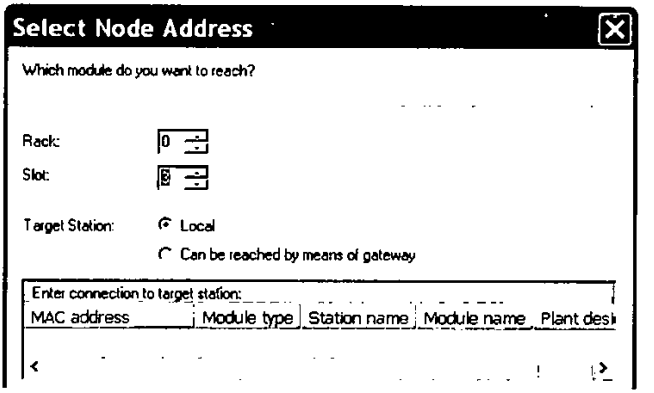


شکل ۹-۲۹ تنظیمات کارت CP5611

۹-۵-۲ مشکل در دانلود و آپلود با وجود برقراری ارتباط

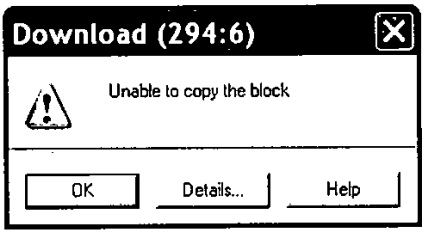
در برخی شرایط ارتباط بین PLC و کامپیوتر از نظر فیزیکی مشکل ندارد و تنظیمات Set PG/PC نیز درست است ولی عمل دانلود و آپلود مشکل دارد. در این شرایط به نکات زیر توجه کنید:

- برای آپلود وقتی ارتباط با کارت شبکه روی PLC برقرار است لازم است شماره اسلات CPU به‌صورت دستی در پنجره زیر وارد شود.



شکل ۹-۳۰ تنظیمات مربوط به آپلود

- در دانلود ممکن است بلاک یا آدرس غیر مجاز در برنامه وجود داشته باشد. در این شرایط با پیغام خطای زیر مواجه خواهیم شد.

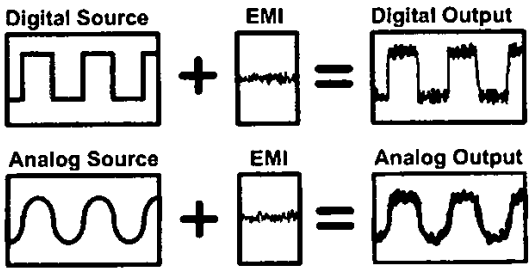


شکل ۹-۳۱ نمونه خطا در هنگام دانلود

این نوع خطاها دارای یک شماره منحصر بفرد هستند که در بالای شکل ۹-۳۱ نیز نمونه آنرا (294:6) می‌بینید. برای اطلاع از لیست کامل خطاهایی که در Step7 ظاهر می‌شود به پیوست ۴ مراجعه کنید.

۹-۶ اشکالات گذرای ناشی از نویز

یکی از بدترین نوع اشکالات در سیستم‌های اتوماسیون که عیب‌یابی آنها بسیار دشوار است اشکالات گذراست. این نوع اشکالات برای لحظاتی کار کنترل را دچار اختلال می‌کنند و ممکن است برخی چراغ‌های فالت را نیز برای لحظاتی روشن کنند. اما از آنجا که پس از گذشت زمان فوق که ممکن است خیلی کوتاه باشد سیستم به حالت عادی برمی‌گردد شناسایی عیب مشکل خواهد بود. به‌عنوان مثال استارت شدن یک موتور بزرگ یا انجام جوشکاری یا بروز رعد و برق ممکن است یک سیگنال آنالوگ را تحت تأثیر قرار داده و مقدار آنرا به‌صورتی تغییر دهد که سیستم کنترل فرمان‌های غلط صادر نماید. نویزهایی که نسبت به سیگنال اصلی دامنه کم دارند و مانند شکل ۹-۳۲ تأثیر می‌گذارند را می‌توان با استفاده از برنامه‌نویسی فیلتر کرد.



شکل ۹-۳۲ تاثیر نویز با دامنه کم روی سیگنال‌های دیجیتال و آنالوگ

ولی اگر در شرایطی که سیگنال با دامنه کم (مانند سیگنال ترموکوپل) وجود داشته باشد به‌شدت تحت تأثیر نویز قرار می‌گیرد.

برای جلوگیری از بروز این گونه مشکلات تأکید می‌شود که نکات و استانداردهای مربوط به کابل کشی و سیستم زمین به دقت مورد توجه قرار گیرد. برخی از این نکات در فصل کار با سیگنال‌های آنالوگ ذکر شد و برخی دیگر در بحث شبکه در کتاب‌های جداگانه آورده شده است.

۹-۷ چراغ‌های فالت روی برخی ماژول‌های PLC

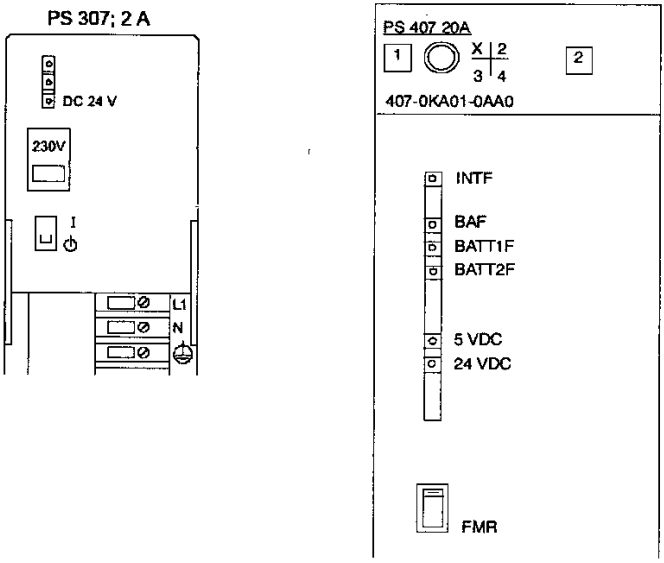
آشنایی با چراغ‌های فالت روی ماژول‌ها برای عیب‌یابی ضروری است. همه ماژول‌ها مجهز به چراغ فالت نیستند. به‌عنوان مثال ماژول‌های ورودی و خروجی دیجیتال معمولی فاقد چراغ فالت می‌باشند. در اینجا به بررسی برخی از ماژول‌های مهم که دارای LED نشان‌دهنده‌ی فالت هستند می‌پردازیم. این ماژول‌ها عبارتند از:

- فالت منبع تغذیه
- ماژول‌های ورودی و خروجی آنالوگ
- ماژول‌های رابط (IM)

در مورد چراغ‌های فالت روی CPU در کتاب سطح مقدماتی توضیح داده شد. سایر ماژول‌ها مانند کارت‌های CP و FM نیز بایستی در جای خود مورد بحث قرار گیرند.

۹-۷-۱ چراغ‌های فالت منبع تغذیه

منبع تغذیه PS300 چراغ فالت خاصی ندارد ولی منبع تغذیه PS400 دارای چراغ‌های فالت مختلفی است. شماتیک این دو منبع تغذیه در شکل ۹-۳۳ نشان داده شده است.



شکل ۹-۳۳ چراغ‌های فالت روی منبع تغذیه

منبع تغذیه S7-300 چراغ فالت خاص ندارد، فقط دارای LED 24VDC است که ممکن است یکی از سه حالت زیر را داشته باشد:

- روشن — حالت نرمال
- چشمک زن — اضافه جریان در خروجی منبع تغذیه
- تیره — اتصال کوتاه در خروجی منبع تغذیه – کاهش ولتاژ در ورودی منبع تغذیه

لازم به ذکر است که هر گونه مشکلی که برای PS300 پیش بیاید هیچ چراغ فالتی روی CPU روشن نخواهد شد چون بین PS300 و CPU هیچ ارتباط دیتا وجود ندارد. بنابراین در صورت بروز اضافه جریان در خروجی یا کاهش ولتاژ ورودی آن به‌جز چراغ فالت روی آن هیچ ابزاری که بتواند عیب را نمایش دهد وجود ندارد.

در منبع تغذیه S7-400 چراغ‌های فالتی که بیانگر وضعیت منبع تغذیه و باتری‌های آن است وجود دارد.

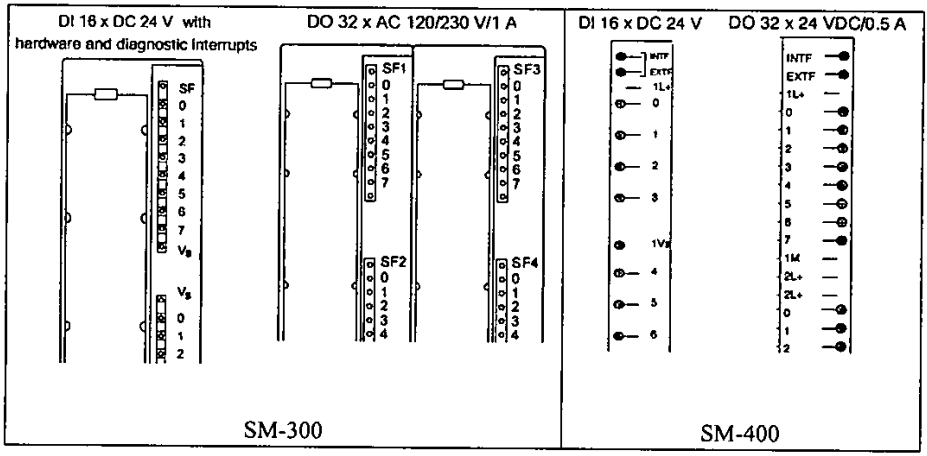
جدول ۹-۱ مفاهیم لامپ‌های PS400

لامپ	رنگ	وضعیت	مفهوم
INTF	قرمز	فالت	وجود خطای داخلی
5 VDC	سبز	نرمال	وجود ولتاژ خروجی 5V DC
24 VDC	سبز	نرمال	وجود ولتاژ خروجی 24V DC
BAF	قرمز	فالت	ضعیف شدن ولتاژ باتری روی backplane bus
BATTF	زرد	فالت	خالی شدن باتری یا عدم وجود آن

در شرایطی که چراغ فالت روی منبع تغذیه PS400 روشن شود، روی CPU نیز چراغ EXTF که بیانگر External Fault است روشن خواهد شد ولی چراغ INTF روی CPU روشن نمی‌شود.
روشن شدن EXTF ناشی از فالت منبع تغذیه هیچ اختلالی در کار CPU ایجاد نخواهد کرد و همانطور که قبلاً ذکر شد، OB81 را صدا می‌زند ولی عدم وجود OB81 نیز منجر به توقف CPU نخواهد شد.

۹-۷-۲ چراغ‌های فالت کارت‌های ورودی و خروجی

فقط برخی از کارت‌های ورودی و خروجی دارای چراغ‌های فالت هستند. نمونه‌هایی از این کارت‌ها در شکل ۹-۳۴ نشان داده شده است. همانطور که دیده می‌شود در SM-300 چراغ SF و در SM-400 چراغ‌های INTF و EXTF وجود دارد.



شکل ۹-۳۴ نمونه چراغ‌های فالت روی کارت‌های ورودی و خروجی

نکات قابل توجه

- در S7-300 کارت‌های ورودی و خروجی که دارای قابلیت وقفه Diagnostic هستند و کارت‌های خروجی که فیوز دارند مجهز به چراغ SF هستند. در این نوع در صورت سوختن فیوز چراغ SF روشن می‌شود. به‌عنوان مثال در کارت DO 32x AC 120/230 V /1A هر هشت خروجی یک فیوز دارند، از اینرو دارای چهار چراغ SF است.
- در S7-400 کارت‌های ورودی و خروجی که قابلیت Diagnostic دارند دارای دو چراغ INTF و EXTF هستند. مشکل داخلی کارت (نظیر سوختن فیوز در کارت‌های خروجی) چراغ INTF را روشن می‌کند ولی مشکلات بیرون کارت نظیر اتصال کوتاه ورودی به M یا L+ چراغ EXTF را روشن می‌نماید.

جدول ۹-۲ حالت‌های روشن شدن چراغ‌های INTF و EXTF را برای کارت‌های دیجیتال و آنالوگ ورودی S7-400 نشان می‌دهد. بدیهی است پیغام مربوطه در بافر کارت نیز ثبت می‌گردد.

جدول ۹-۲

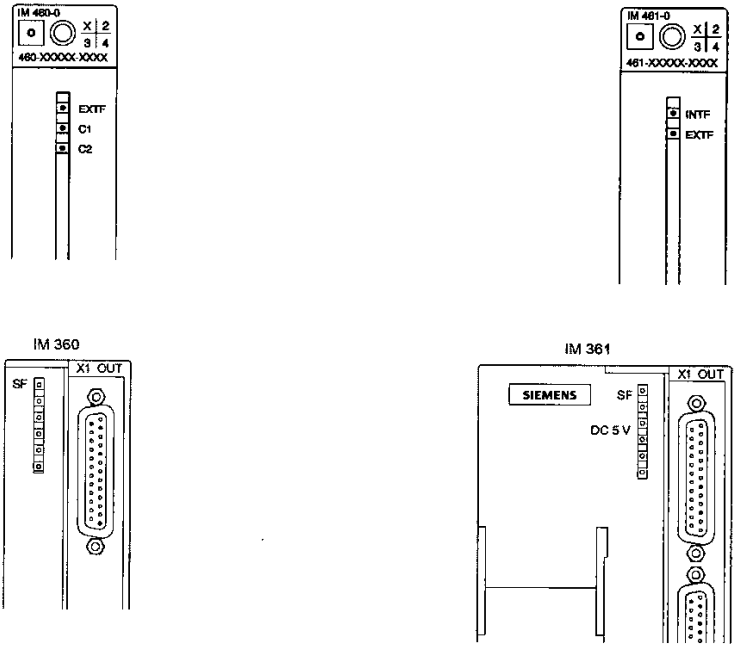
Diagnostic Message DI-400	LED	Scope of the Diagnosis
Module problem	INTF/EXTF	Module
Internal malfunction	INTF	Module
External malfunction	EXTF	Module
Channel error present	INTF/EXTF	Module
External auxiliary supply missing	EXTF	Module
Front connector missing	EXTF	Module
Module not parameterized.	INTF	Module
Wrong parameters	INTF	Module
Channel information available	INTF/EXTF	Module
STOP mode	–	Module
Internal voltage failure	INTF	Module
EPROM error	INTF	Module
Hardware interrupt lost	INTF	Module
Parameter assignment error	INTF	Channel

Short-circuit to M	EXTF	Channel
Short-circuit to L+	EXTF	Channel
Wire break	EXTF	Channel
Fuse blown	INTF	Channel
Sensor supply missing	EXTF	Channel/channel group
No load voltage L+	EXTF	Channel/channel group

Diagnostic Message AI-400	LED	Diagnostics Effective for
Module problem	INTF/EXTF	Module
Internal malfunction	INTF	Module
External malfunction	EXTF	Module
Channel error present	INTF/EXTF	Module
External auxiliary supply missing	EXTF	Module
Front connector missing	EXTF	Module
Module not configured.	INTF	Module
Wrong parameters	INTF	Module
Channel information available	INTF/EXTF	Module
Coding key incorrect or missing	INTF	Module
Thermocouple connection fault	EXTF	Module
STOP operating mode	–	Module
EPROM error	INTF	Module
RAM error	INTF	Module
ADC/DAC error	INTF	Module
Hardware interrupt lost	INTF	Module
Configuring/parameter assignment error	INTF	Channel
Short-circuit to M	EXTF	Channel
Wire break	EXTF	Channel
Reference channel error	EXTF	Channel
Underflow	EXTF	Channel
Overflow	EXTF	Channel
User connection not wired	EXTF	Channel
Open conductor in + direction	EXTF	Channel
Open conductor in – direction	EXTF	Channel
Run time calibration error	EXTF	Channel
Underrange or overrange	EXTF	Channel
Open conductor in the current source	EXTF	Channel
User calibration doesn't correspond to the parameter assignment	EXTF	Channel

۹-۷-۳ چراغ‌های فالت روی Interface Module

ماژول‌های IM برای ارتباط بین رک اصلی و رک توسعه به‌کار می‌روند. نمونه این ماژول‌ها در S7-300 ماژول‌های IM360/IM361 و در S7-400 ماژول‌های IM460/IM461 است که چراغ‌های فالت آنها در شکل ۹-۳۵ نشان داده شده است.



شکل ۹-۳۵ نمونه چراغ‌های فالت روی IM

فصل ۹

در S7-300

- اگر تغذیه IM361 نصب شده در رک شماره ۱ قطع شود، چراغ SF روی IM360 رک اصلی روشن می‌شود و CPU متوقف می‌گردد.
- اگر تغذیه IM361 نصب شده در رک شماره ۲ یا رک شماره ۳ قطع شود، چراغ SF روی IM360 رک اصلی و چراغ SF روی IM361های بالاتر روشن می‌گردد و CPU متوقف می‌شود.
- اگر کانکتور IM جدا شود، چراغ SF روی IMهایی که متصل نیست و IMهای نزدیک به CPU روشن می‌شود.

در S7-400

- در IM460 اگر خط ۱ به رک‌های توسعه وصل باشد چراغ C1 روشن و اگر خط ۲ به رک‌های توسعه وصل باشد، چراغ C2 روشن است و نشان‌دهنده درستی ارتباط است.
- در IM460 چراغ EXTF وقتی روشن می‌شود که کابل ارتباطی هر کدام از خطوط ۱ و ۲ قطع شود یا ترمینیتور انتها خطوط موجود نباشد.
- در IM461 اگر دیپ سوئیچ شماره‌ی رک صفر یا بیش از ۲۱ تنظیم شده باشد چراغ INTF روشن می‌گردد. در صورت قطع شدن کابل ارتباطی یا نبود ترمینیتور چراغ EXTF روشن خواهد شد.

۹-۸ پرسش و تحقیق

چه ابزارهای سخت افزاری برای عیب‌یابی PLC وجود دارد؟

۹-۹ تست‌های خودآزمایی

۱- بهترین ابزار در تشخیص خطاهایی‌که منجر به توقف CPU شده و چراغ فالت CPU را نیز روشن می‌کنند استفاده از کدام ابزار است؟

الف) جدول VAT
ب) پنجره Online
ج) Diagnostic Buffetr
د) ابزار Monitor

۲- اگر پیکربندی انجام‌شده با پیکربندی واقعی متفاوت باشد، مثلاً برخی از کارت‌های I/O متفاوت باشند، چگونه می‌توان این اشکال را مشاهده نمود؟

الف) جدول VAT
ب) پنجره Online در محیط HW Config
ج) Diagnostic Buffetr
د) ابزار Monitor

۳- اگر ماژولی در دسترس CPU قرار داشته باشد ولی ماژول دارای فالت باشد، در حالت Online در محیط HW Config کدام حالت اتفاق می‌افتد؟

الف) روی ماژول خط قرمز کشیده می‌شود.
ب) روی ماژول دایره قرمز کشیده می‌شود.
ج) روی ماژول خط یا دایره قرمز کشیده می‌شود.
د) هیچ علامت خاصی روی ماژول ظاهر نمی‌شود.

۴- کدام‌یک از OBهای زیر در صورت فعال‌سازی و عدم دانلود به CPU باعث توقف CPU نمی‌شود؟

الف) وقفه Diagnostics بدون دانلود کردن OB82
ب) وقفه Hardware بدون دانلود کردن OB4x
ج) وقفه Time of Day بدون دانلود کردن OB1x
د) OBهای Cyclic Interrupt بدون دانلود OB3X

۵- دانلود سخت افزار وقتی پنجره Online باز است، منجر به کدام‌یک از حالات زیر می‌شود؟

الف) روشن شدن چراغ فالت
ب) توقف CPU
ج) توقف CPU و روشن شدن چراغ فالت
د) عدم توقف CPU و روشن‌نشدن چراغ فالت

۶- در شناسایی اشکالات برنامه‌نویسی که باعث توقف CPU می‌گردند، از کدام‌یک از ابزارهای زیر می‌توان استفاده نمود؟

الف) Diagnostic Buffer
ب) Stack
ج) جدول VAT
د) موارد ۱ و ۲

۷- بدترین نوع اشکالات در PLC کدام‌یک از موارد زیر است؟

الف) فالت‌هایی که منجر به توقف CPU می‌گردند.
ب) فالت‌هایی که منجر به روشن شدن چراغ فالت می‌گردند.
ج) اشکال در منطق برنامه
د) اشکالات سخت‌افزاری

۸- مهم‌ترین عامل ایجاد اشکالات گذرا در PLC کدام مورد زیر است؟

الف) شبکه‌های صنعتی
ب) نویز
ج) اشکال در منطق برنامه
د) اشکالات سخت‌افزاری

۹- چشمک زدن لامپ DC 24v روی منبع تغذیه S7-300 بیانگر کدام اشکال می‌باشد؟

الف) اتصال کوتاه در خروجی منبع
ب) اتصال کوتاه در ورودی منبع
ج) اضافه جریان در خروجی منبع تغذیه
د) کاهش ولتاژ در ورودی منبع تغذیه

فصل ۱۰

ابزارهای بررسی، اصلاح و مقایسه برنامه PLC

در این فصل نحوه استفاده از Reference Data، Rewiring و Compare Blocks تشریح شده است.

چکیده مطالب

- با استفاده از Reference Data می‌توان به امکانات زیر دست یافت:
  ۱- آدرس‌های استفاده شده در برنامه
  ۲- آدرس‌های استفاده شده از حافظه سیستم
  ۳- آدرس‌های فاقد سمبل
  ۴- سلسه مراتب فراخوانی بلاک‌ها
- Reference Data یکی از ابزارهای اصلی در شناسایی اشکالات برنامه‌نویسی است.
- برای تغییر یک یا چند آدرس در کل برنامه از Rewire استفاده می‌شود.
- با Compare Blocks می‌توان بین بلاک‌های دو پروژه مقایسه انجام داد.

فصل ۱۰

۱۰-۱ مقدمه

در این فصل با برخی امکانات دیگر نرم‌افزار Step7 که برای عیب‌یابی و اعمال اصلاحات در پروژه مفید هستند آشنا می‌شویم. ابزارهای مورد بحث عبارتند از:

Reference Data

- توسط این ابزار پس از نهایی شدن برنامه، می‌توان لیست آدرس‌های استفاده شده و محل استفاده آنها و نحوه فراخوانی بلاک‌ها را مشاهده نمود. این ابزار در عیب‌یابی برنامه بسیار مفید است.

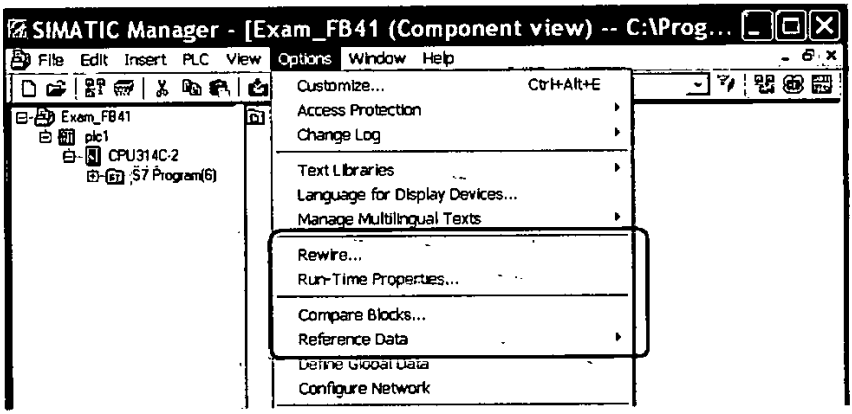
Rewiring

- توسط این ابزار می‌توان در صورت لزوم آدرس یا آدرس‌هایی را در کل برنامه تغییر داد.

Compare Blocks

- توسط این ابزار می‌توان بلاک‌ها را بین دو پروژه مقایسه کرد. امکان مقایسه بین پروژه Offline و Online نیز وجود دارد.

هر سه ابزار فوق در منوی Options در Simatic Manager مطابق شکل ۱۰-۱ در دسترس هستند.



شکل ۱۰-۱ مسیر ابزارهای مورد بحث در منوی Options

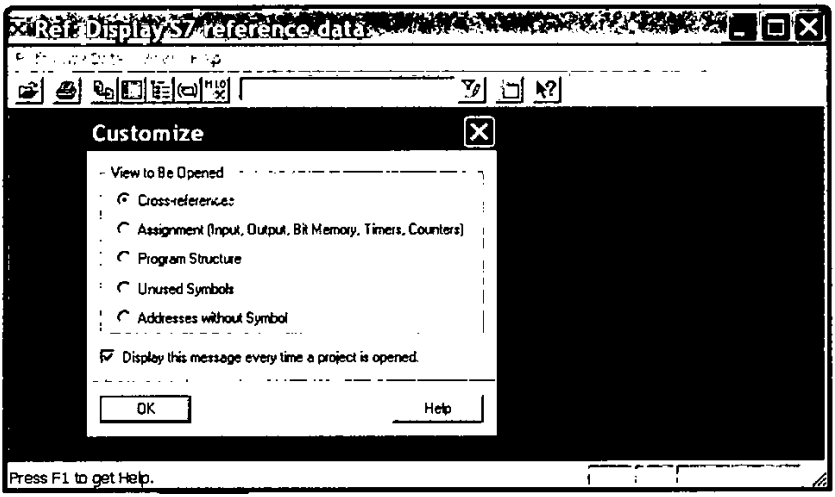
۱۰-۲ استفاده از Reference Data

Reference Data ابزاری است که علاوه بر ارائه دید کلی نسبت به برنامه، جزئیات لازم را در مورد آدرس‌ها، سمبل‌ها، بلاک‌های استفاده شده در برنامه مشخص می‌نماید. کاربرد آن در موارد زیر است:

- دیدن لیست آدرس‌های استفاده شده در برنامه برای شناسایی فضاهای خالی آدرس‌ها

- بررسی اینکه هر آدرس در چه بلاک‌هایی و در چه Network استفاده شده است.
- بررسی اینکه آدرس در کجا خوانده شده (Read) و در کجا نوشته شده است (Write).
- بررسی تلاقی بین آدرس‌ها
- بررسی آدرس‌های فاقد سمبل
- بررسی اینکه یک بلاک برنامه‌نویسی در چه بلاک‌هایی فراخوان شده است.

برای فعال کردن آن در Simatic Manager وقتی پوشه Blocks باز است، از مسیر Option > Reference Data>Display استفاده می‌کنیم. روش دیگر استفاده از منوی Options در محیط برنامه‌نویسی و همان مسیر ذکر شده است. به هر روش که روی Display کلیک کنیم با محیط شکل ۱۰-۲ مواجه خواهیم شد.



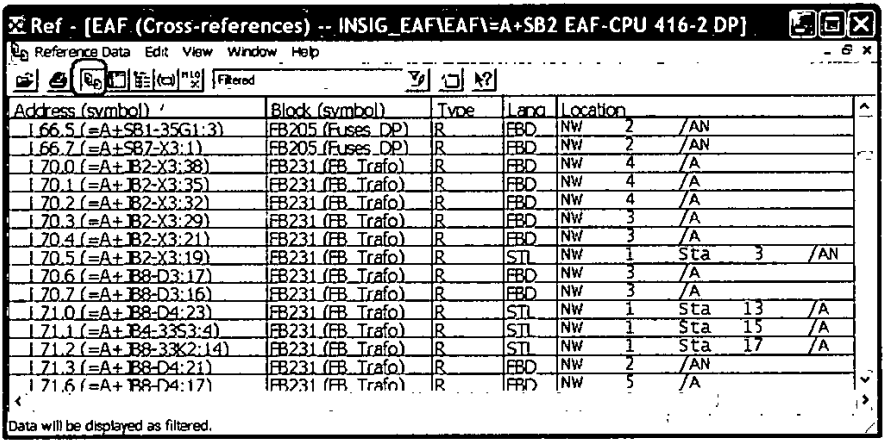
شکل ۱۰-۲ برنامه Reference Data

همانطور که دیده می‌شود در پنجره فوق می‌توان انتخاب کرد که کدام بخش از محیط Reference Data نمایش داده شود. انتخاب مورد نظر می‌تواند از طریق نوارابزار نیز انجام شود. در ادامه امکانات این محیط به‌ترتیب تشریح می‌شوند.

فصل ۱۰

Cross Reference

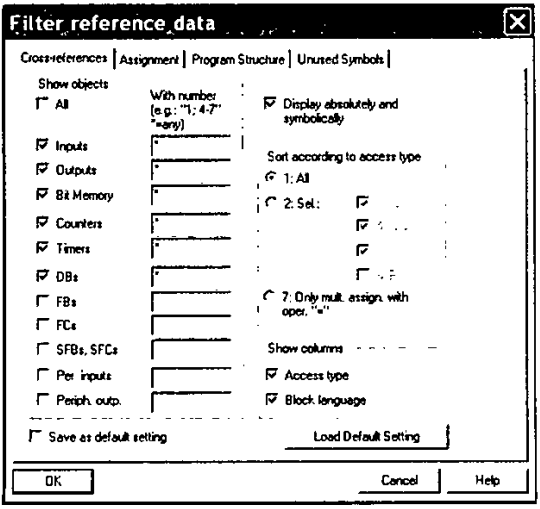
اگر در پنجره شکل ۱۰-۲ گزینه Cross Reference را انتخاب کنیم یا روی آیکن نشان داده شده در شکل ۱۰-۳ از نوارابزار کلیک کنیم، اطلاعات آدرس‌های استفاده شده در برنامه نشان داده خواهد شد.



شکل ۱۰-۳ مشاهده Cross Reference

نکات قابل توجه

- در این لیست کلیه آدرس‌های برنامه (I , Q , M ,T ,C) با نام سمبلیک، نوع و نیز نام بلاکی که این آدرس در آن استفاده شده است، نشان داده می‌شود.
- به‌طور پیش‌فرض آدرس‌های دیتا بلاک و آدرس‌های Peripheral نمایش داده نمی‌شوند. در واقع فیلتری برای نمایش فعال است. اگر روی آیکن کنار باکس زرد رنگ فیلتر کلیک کنیم، پنجره زیر ظاهر می‌شود که در آن می‌توان سایر گزینه‌ها را نیز انتخاب نمود.



شکل ۱۰-۴ فیلتر Cross Reference

- اگر بخواهیم فقط مورد خاصی مثلاً آدرس‌های PIW نمایش داده شود، در پنجره فوق Per.inputs را فعال و سایر گزینه‌ها را غیرفعال می‌کنیم. در این صورت لیستی شبیه شکل ۱۰-۵ خواهیم داشت.

Ref - [EAF (Cross-references) -- INSIG_EAF\EAF\=A+SB2 EAF-CPU 416-2 DP]

Reference Data Edit View Window Help

Filtered

Address (symbol)	Block (symbol)	Type	Lang	Location
PIW 706 (=A+JB17-40U1:3)	FB210 (FB_Hyd_System)	R	STL	NW 1 Sta 1
PIW 724 (=A+JB89-X1.3:3)	FB311 (FB_Ladle_Bubling)	R	STL	NW 1 Sta 23
PIW 726 (=A+JB89-X1.3:5)	FB311 (FB_Ladle_Bubling)	R	STL	NW 1 Sta 25
PIW 780 (=A+SB11-U1:15)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 1
PIW 782 (=A+SB11-U1:17)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 3
PIW 784 (=A+SB11-U1:19)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 5
PIW 786 (=A+SB11-U2:15)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 7
PIW 788 (=A+SB11-U2:17)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 9
PIW 790 (=A+SB11-U2:19)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 11
PIW 792 (=A+SB11-U3:15)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 13
PIW 794 (=A+SB11-U3:17)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 15
PIW 796 (=A+SB11-U3:19)	FB235 (FB_AnalogMeasurem...	R	STL	NW 1 Sta 17
PIW 896 (=A+JB185-35U1:+)	FB305 (FB_Lime_Injection)	R	STL	NW 1 Sta 47
PIW 996 (=A+JB181-35U1:+)	FB300 (FB_Carbon_Injection)	R	STL	NW 1 Sta 47

Data will be displayed as filtered.

شکل ۱۰-۵ نمایش فقط PIW در Cross Reference

- اگر روی آدرس مورد نظر دوبار کلیک کنیم، بلاک مربوطه باز شده و محلی که این آدرس به‌کار رفته نمایش داده می‌شود.
- در ستون Type حرف R یا W دیده می‌شود که نشان می‌دهد آدرس مورد نظر خوانده و یا نوشته شده است.

Ref - [EAF (Cross-references) -- INSIG_EAF\EAF\=A+SB2 EAF-CPU 416-...

Reference Data Edit View Window Help

Filtered

Address (symbol)	Block (symbol)	Type	Lang	Location
M 501.5 (F_Edge_EndOfHeat)	FC501 (FC_HeatData)	W	FBD	NW 2
M 501.6 (F_Edge_StartOfHeat)	FB300 (FB_Carbon_Injection)	R	FBD	NW 43
	FB305 (FB_Lime_Injection)	R	FBD	NW 46
	FB335 (FB_100ms)	R	FBD	NW 1
				NW 2
	FC501 (FC_HeatData)	R	FBD	NW 3
		W	FBD	NW 3
M 501.7 (F_Edge_EndHeat_Delay)	FC501 (FC_HeatData)	W	FBD	NW 10
M 502.1 (F_Arc_Is_On)	FC501 (FC_HeatData)	R	FBD	NW 15
M 502.2 (Edge_TriggerInterrupt)	FC501 (FC_HeatData)	W	FBD	NW 17

Data will be displayed as filtered.

شکل ۱۰-۶ مشاهده آدرس‌های به‌کار رفته در چند جای برنامه

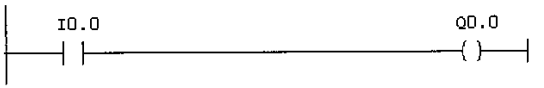
- اگر یک آدرس در چند جای برنامه به‌کار رفته باشد، در کنار آن آدرس علامت + ظاهر می‌شود که اگر آن را باز کنیم مانند آدرس M501..6 در شکل ۱۰-۶ محل‌های استفاده شده را می‌توانیم مشاهده کنیم.
- در ستون Langugae زبان برنامه‌نویسی نشان داده می‌شود.

- در ستون Location شماره Network که این آدرس در آن استفاده شده نشان داده می‌شود. همانطور که گفته شد، با دو بار کلیک روی آدرس بلاک مربوطه باز شده و Network فوق نشان داده می‌شود.

مثال ۱۰-۱ بررسی یک مشکل برنامه‌نویسی

در برنامه‌ای به‌اشتباه به یک خروجی در دو نقطه فرمان داده می‌شود. در Network1 برنامه زیر را داریم:

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1 : Title:
```

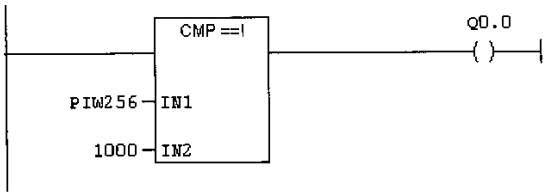


شکل ۱۰-۷ بخشی از برنامه مثال ۱۰-۱

پس از دانلود اگر برنامه را مانیتور کرده و I0.0 را فعال کنیم، نشان می‌دهد که خروجی Q0.0 یک شده است ولی عملاً خروجی فوق روی کارت DO فعال نمی‌شود.

علت این است که در Network 9 برنامه زیر نیز به خروجی فرمان می‌دهد چون این دستور بعد از دستور قبلی است، پس فرمان به خروجی براساس آن صادر می‌شود.

```
Network 9 : Title:
```



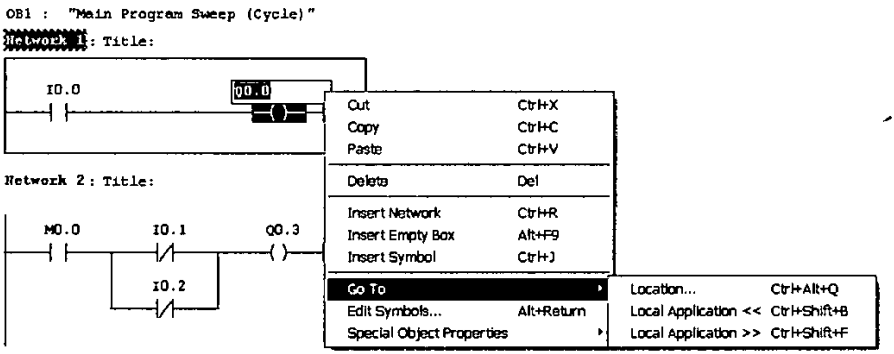
شکل ۱۰-۸ بخش دیگری از برنامه مثال ۱۰-۱

بنابراین به‌ویژه در زمان تست برنامه، با مانیتور کردن محیط برنامه‌نویسی اگر یک خروجی به رنگ سبز روشن است نمی‌توان اطمینان داشت که خروجی فوق عملاً روشن باشد. وضعیت نهایی خروجی از طریق کارت DO یا از طریق جدول VAT قابل مشاهده است.

فرض کنید نمی‌دانیم که فرمان به خروجی در چه تقاطی از برنامه اعمال شده است. برای پیدا کردن آن به دو روش می‌توان عمل کرد:

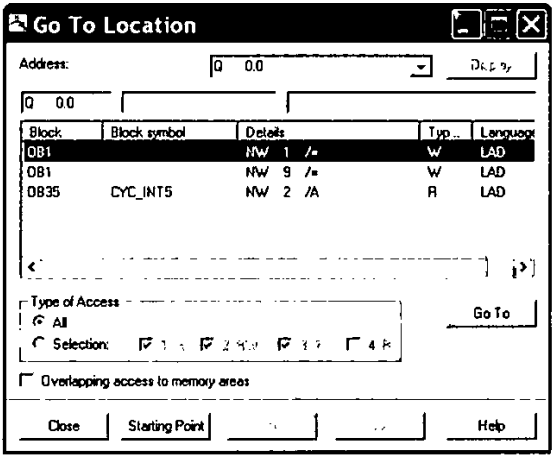
روش اول: با استفاده از GOTO Location

در این روش روی آدرس مورد نظر کلیک راست کرده و گزینه Goto Location را انتخاب می‌کنیم.



شکل ۱۰-۹ استفاده از Go To Location

در این هنگام پنجره‌ای مانند شکل ۱۰-۱۰ باز می‌شود، که در آن محل‌های استفاده از Q0.0 در برنامه نمایش داده می‌شود.

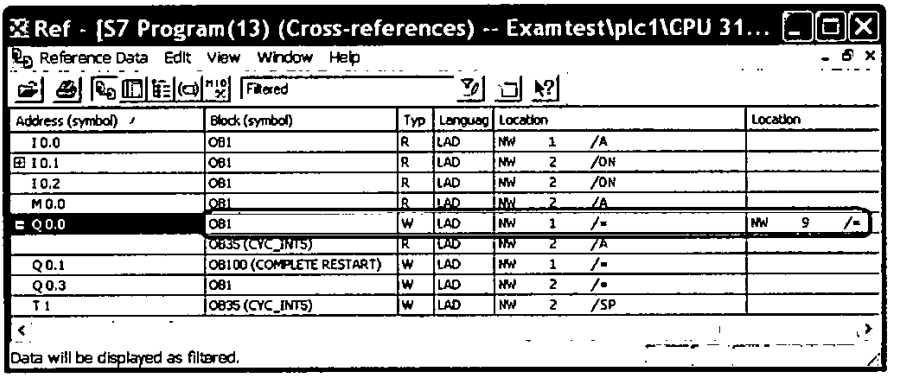


شکل ۱۰-۱۰ مشاهده محل‌های استفاده از یک آدرس در برنامه با Go To Location

از پنجره فوق مشخص است که آدرس Q0.0 در سه Network برنامه استفاده شده است. با توجه به ستون Type می‌بینیم که در دو نقطه حرف W وجود دارد یعنی روی Q نوشته شده است. اگر مورد دوم را انتخاب کرده و روی کلید Go To پایین پنجره کلیک کنیم به سطر مورد نظر پرش کرده و برنامه را نشان می‌دهد که با اصلاح آن مشکل برطرف خواهد شد.

روش دوم: با استفاده از Cross Reference

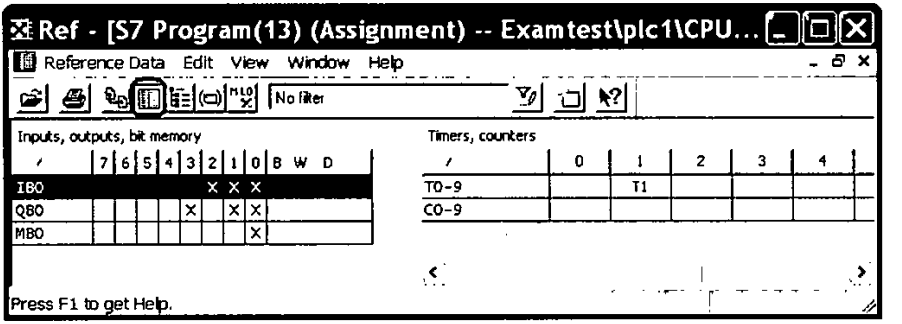
برای برنامه فوق پنجره Cross Reference به‌صورت زیر است. در اینجا نیز دیده می‌شود که فرمان نوشتن در دو Nework به‌کار برده شده است.



شکل ۱۰-۱۱ مشاهده محل‌های استفاده از یک آدرس در برنامه با Cross Reference

Assignment List

گزینه دیگری که در Reference Data وجود دارد جدول Assignment است. در این جدول می‌توان مشاهده کرد که از متغیرهای حافظه نظیر I,Q,M,T,.C چه آدرس‌هایی استفاده شده است.

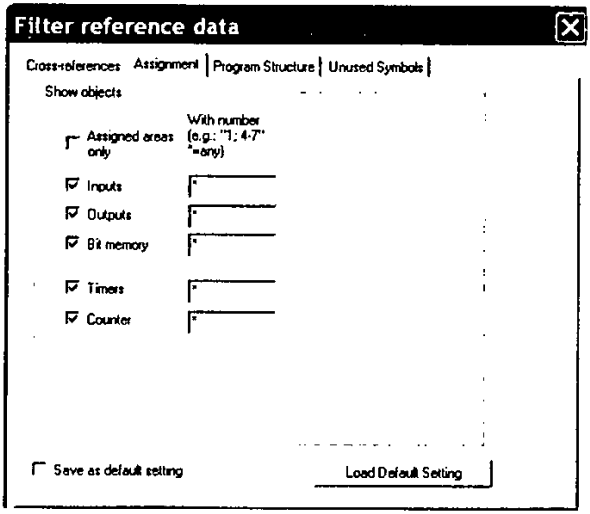


شکل ۱۰-۱۲ مشاهده Assignment List

نکات قابل توجه

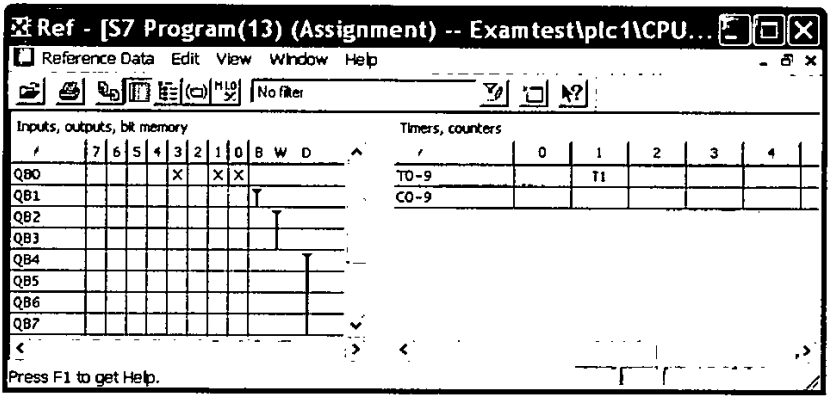
- در این لیست فقط متغیرهای System Memory نمایش داده می‌شوند، بنابراین امکان مشاهده DB و PIW و PQW وجود ندارد.

- در برنامه‌های بزرگ می‌توان با اعمال فیلتر فقط آدرس‌های دلخواه را لیست نمود.



شکل ۱۰-۱۳ فیلتر در Assignment List

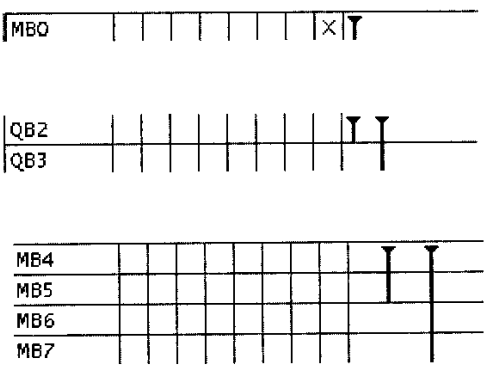
- علامت X در این لیست نشان می‌دهد که آدرس به‌صورت بیتی به‌کار رفته است.
- اگر یک سطر به رنگ آبی باشد و در جلوی آن یک خط آبی ترسیم شده باشد، مفهوم آن این است که آدرس به‌صورت بایتی به‌کار رفته است.
- اگر دو سطر آبی با یک خط مشخص شده باشد، نشان دهنده کاربرد آدرس به‌صورت Word است.
- اگر چهار سطر به‌صورت آبی و با یک خط مشخص شده باشند، معرف کاربرد آدرس به‌صورت DWord است.



شکل ۱۰-۱۴ نحوه نمایش آدرس‌ها در Assignment List

فصل ۱۰

اگر از حافظه ای به‌صورت ترکیبی استفاده شده باشد، علامت‌های فوق را به‌صورت همزمان مشاهده می‌کنیم. نمونه‌ها در شکل ۱۰-۱۵ آورده شده است.

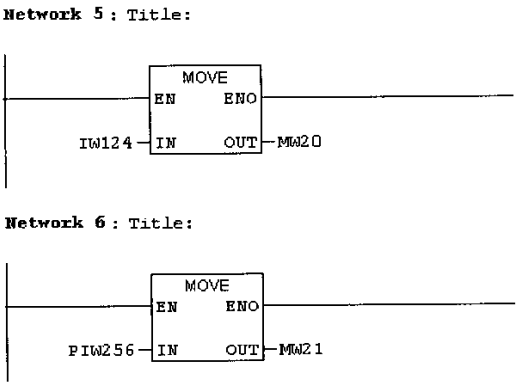


شکل ۱۰-۱۵ مشاهده تلاقی آدرس‌ها در Assignment List

مثال ۱۰-۲ مشکل برنامه‌نویسی که با Cross Reference قابل آشکارسازی نیست

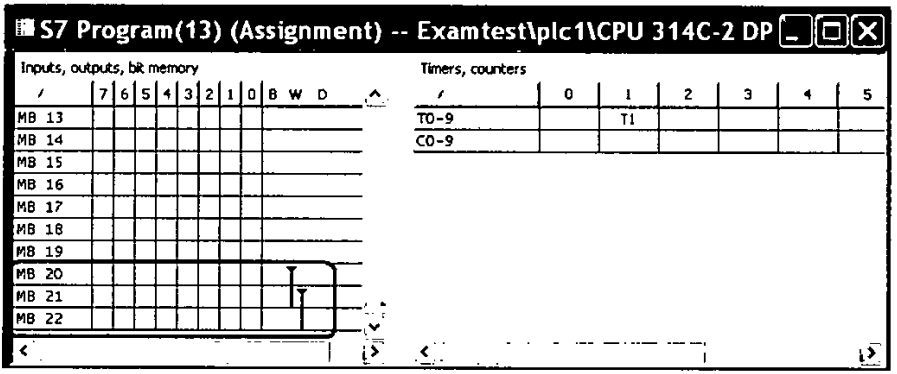
اگر در برنامه‌ای تلاقی آدرس از نمونه‌های ذکر شده در شکل ۱۰-۱۵ وجود داشته باشد، با Go To Location و با Cross reference نمی‌توان آنرا پیدا کرد.

به‌عنوان مثال در نظر بگیرید که دو آدرس Word در یک بایت مانند شکل ۱۰-۱۶ با یکدیگر تلاقی داشته باشند.



شکل ۱۰-۱۶ برنامه مثال ۱۰-۲

در این حالت نتیجه MW20 و MW21 هر دو اشتباه است. برای پیدا کردن اینگونه اشکالات به لیست Assignment مراجعه می‌کنیم. شکل ۱۰-۱۷ این تلاقی را نشان می‌دهد.

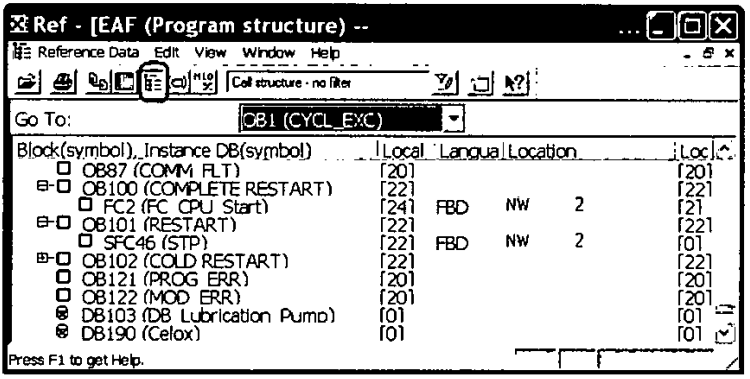


شکل ۱۰-۱۷ مشاهده تلاقی آدرس‌های مثال ۱۰-۲

همانطور که در فصل قبل اشاره شد، این نوع اشکالات هیچ چراغ فالتی را روشن نمی‌کنند ولی برنامه جواب درستی به‌دست نمی‌دهد. بنابراین پس از تکمیل برنامه‌نویسی در فاز تست و راه‌اندازی لازم است با ابزار Assignment List تلاقی آدرس‌ها چک شود.

Program Structure

با انتخاب این گزینه یا کلیک روی آیکن نشان داده شده در شکل ۱۰-۱۸، سلسله مراتب فراخوانی بلاک‌ها نمایش داده می‌شوند. روی هر بلاکی که دوبار کلیک شود باز می‌شود.



شکل ۱۰-۱۸ مشاهده Program Structure

در کنار نام بلاک‌ها علائم مختلفی ممکن است استفاده شود. به‌عنوان مثال علامت ضربدر در کنار بلاک نشان دهنده این است که بلاک در پروژه موجود است ولی فراخوان نشده است. این‌گونه بلاک‌ها نباید به PLC دانلود شوند چون فضای حافظه را اشغال می‌کنند.

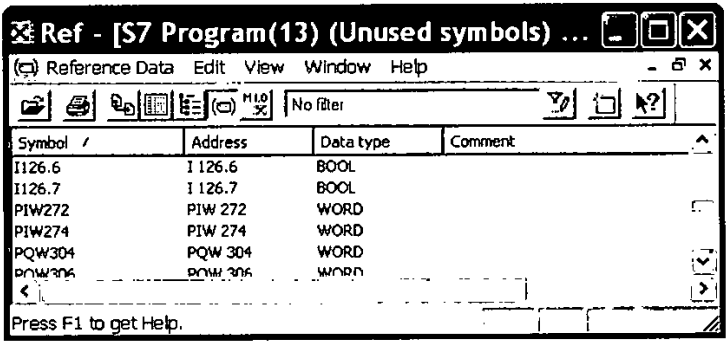
راهنمای برخی از علائم در شکل ۱۰-۱۹ آمده است.

Example	Meaning	Symbol
CALL FB10	Block called normally	
UC FB10	Block called unconditionally	
CC FB10	Block called conditionally	
-	Data block	
-	Block not called	

شکل ۱۰-۱۹ علائمی که در Program Structure ظاهر می‌شود

Unused Symbols

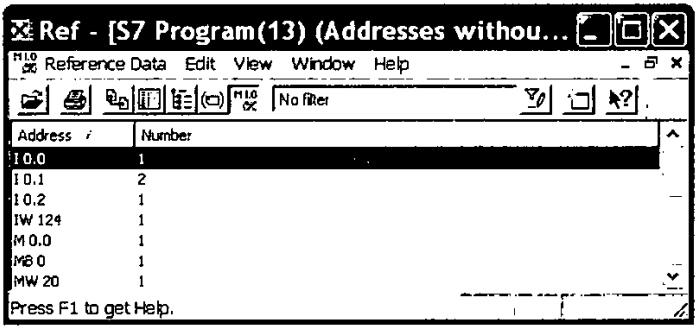
با استفاده از این ابزار می‌توان سمبل‌هایی که تعریف شده ولی استفاده نشده‌اند را مشخص کرد.



شکل ۱۰-۲۰ مشاهده آدرس‌های سمبلیک که در برنامه استفاده نشده‌اند

Address Without Symbols

۶- نمایش آدرس‌هایی که فاقد سمبل هستند.

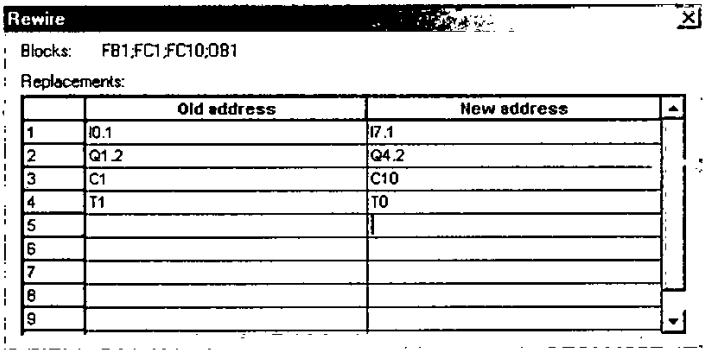


شکل ۱۰-۲۱ مشاهده آدرس‌های فاقد سمبل

۱۰-۳ استفاده از Rewiring

فرض کنید برنامه بزرگی مشتمل بر چندین بلاک و هر بلاک چندین سطر نوشته شده و نیاز باشد که آدرس یا آدرس‌هایی را در همه بلاک‌ها عوض کنیم. قطعاً انجام این کار به‌صورت دستی دشوار خواهد بود. با امکان Rewiring که در برنامه Simatic Manager منوی Option وجود دارد، این کار به‌سادگی انجام‌پذیر است.

پس از کلیک روی Rewiring پنجره‌ای مانند شکل ۱۰-۲۲ باز می‌شود که در آن آدرس‌های قدیم و جدید را وارد کرده و OK را انتخاب می‌کنیم.



شکل ۱۰-۲۲ جدول Rewiring

عمل Rewiring به‌ویژه در مواردی که کاربر ابتدا برنامه را نوشته و سپس سخت‌افزار را پیکربندی کرده باشد مورد نیاز خواهد بود. زیرا باید آدرس‌ها را مطابق آنچه در سخت افزار معرفی شده تغییر داد.

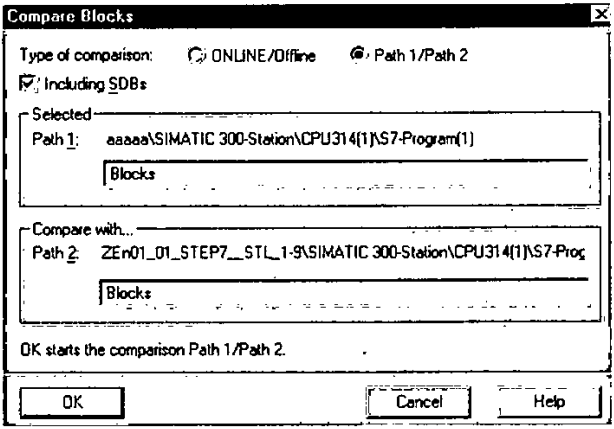
۱۰-۴ مقایسه بلاک‌ها Compare Blocks

می‌توان یک یا چند بلاک را از یک پروژه با پروژه دیگر مقایسه کرد. به‌عنوان مثال می‌توان برنامه PLC که در حال کار است را با برنامه‌ای که روی PC یا PG ذخیره شده باهم مقایسه و تفاوت‌های آنها را شناسایی کرد.

بلاک‌ها در دو حالت زیر می‌توانند مورد مقایسه قرار گیرند:

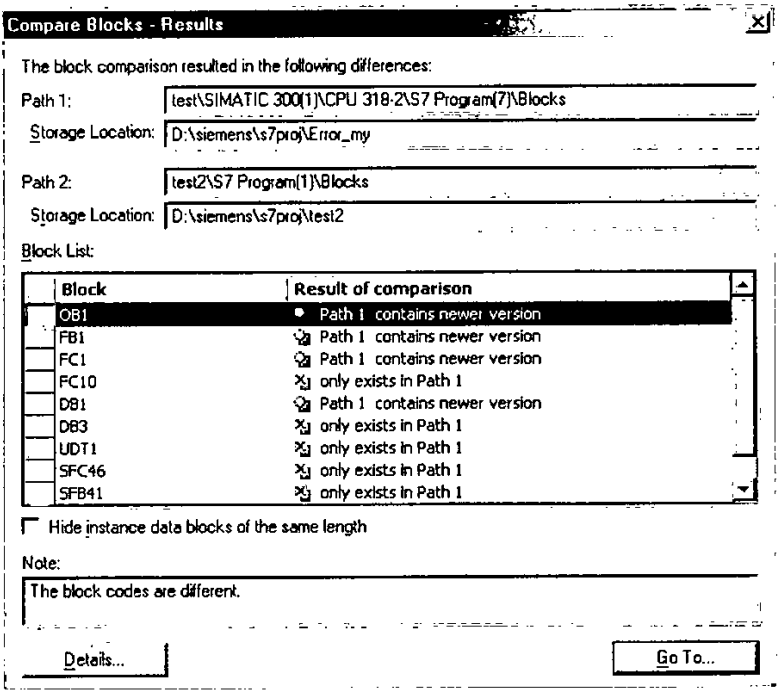
- یکی On Line و دیگری Off Line
- هردو Offline

در Simatic Manager با استفاده از مسیر Option > Compare Blocks پنجره زیر را خواهیم داشت.



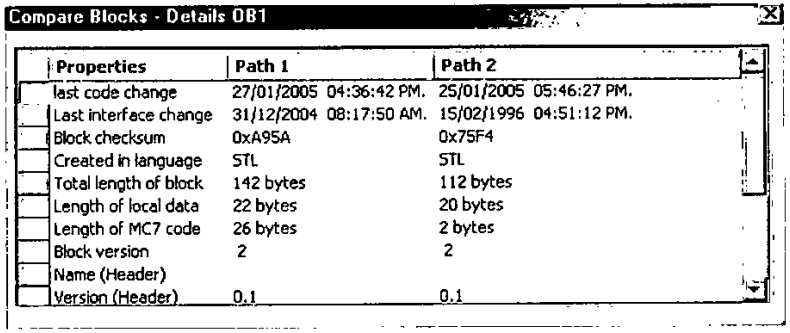
شکل ۱۰-۲۳ نحوه مقایسه بلاک‌های Online با Offline

اگر بخواهیم کل بلاک‌های دو پروژه را مقایسه کنیم، ابتدا روی پوشه Blocks اولی کلیک کرده سپس روی Path2 رفته و همزمان از پروژه دوم که آنرا در Simatic Manager باز کرده‌ایم، پوشه Blocks را انتخاب می‌نماییم. این کار را می‌توان برای دو بلاک هم‌نوع از یک یا دو پروژه نیز انجام داد. پس از انتخاب روی OK کلیک کرده تا مقایسه صورت گیرد. نتیجه مقایسه برای تک تک بلاک‌ها لیست می‌شود.



شکل ۱۰-۲۴ نتیجه مقایسه بلاک‌ها

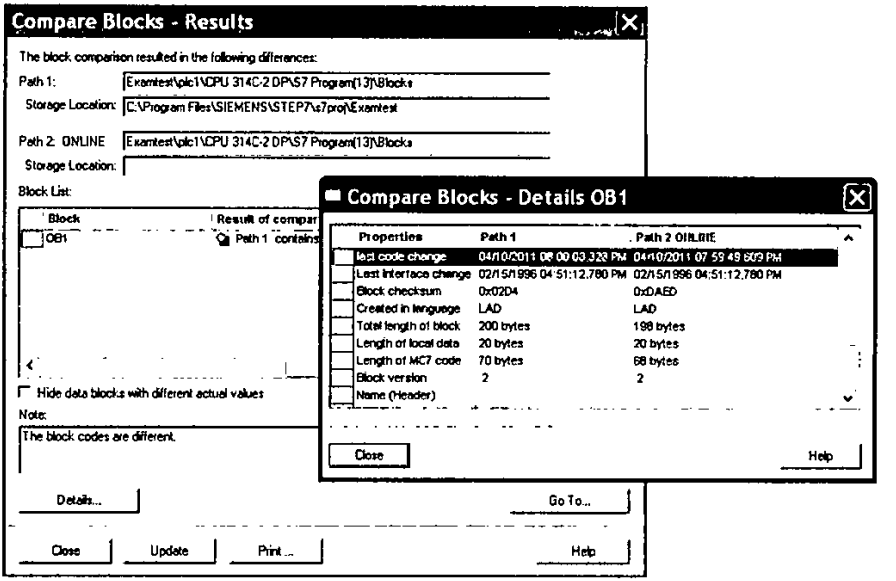
با کلیک کردن روی دکمه Detail باکسی مانند شکل ۱۰-۲۵ مشاهده خواهد شد که در آن موارد اختلاف با رنگ قرمز مشخص شده‌اند.



شکل ۱۰-۲۵ مشاهده موارد اختلاف بین بلاک‌ها

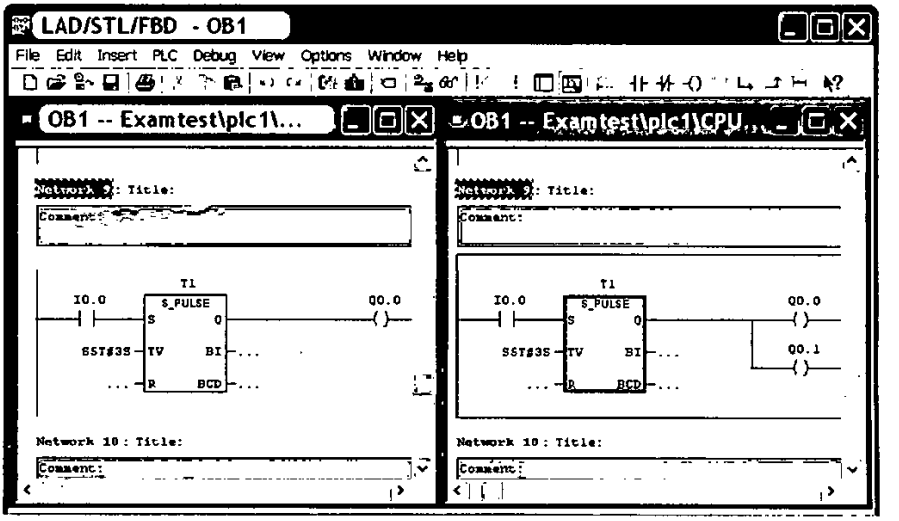
مثال ۱۰-۳

فرض کنید در OB1 اختلافی بین برنامه نوشته شده در Network9 بین Offline و Online وجود دارد. پس از مقایسه پنجره زیر را خواهیم دید.



شکل ۱۰-۲۶ مقایسه بلاک‌ها در مثال ۱۰-۳

حال اگر روی Go To کلیک کنیم، برنامه LAD/STL/FBD باز شده و در دو پنجره کنار هم Network که اختلاف دارد نمایش داده می‌شود.



شکل ۱۰-۲۷ نمایش جزئیات اختلاف دو بلاک مثال ۱۰-۳

۱۰-۵ پرسش و تحقیق

با استفاده از Compare Block وضعیت بلاک‌ها را در یک CPU آلوده به ویروس Stuxnet با بلاک‌های فاقد ویروس در پنجره Offline مقایسه کنید.

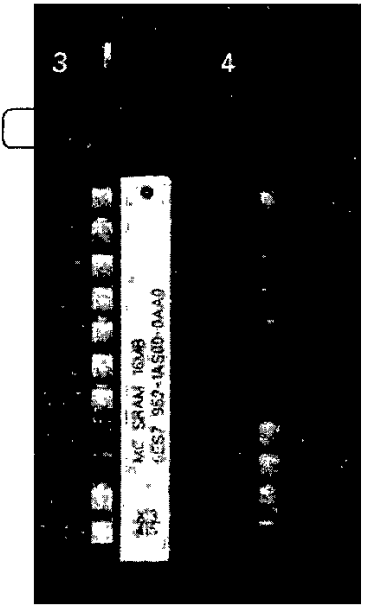
پیوست ۱

نحوه تغییر ورژن CPU

این پیوست نحوه تغییر ورژن CPU را به‌صورت Online و Offline تشریح می‌کند.

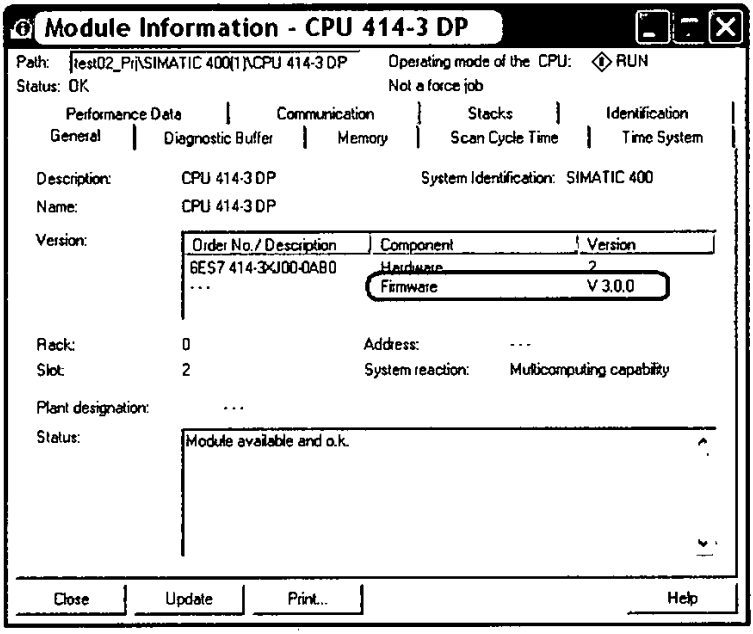
مقدمه

سیستم عامل هر CPU روی حافظه ماندگار داخل آن ذخیره شده است که به آن Firmware گفته می‌شود. هر CPU دارای ورژن خاصی از Firmware است که روی CPU نوشته شده است.



شکل پ۱-۱ ورژن CPU

برای مشاهده ورژن می‌توان در پنجره Module Information در سربرگ General آنرا مانند شکل پ۱-۲ مشاهده نمود.



شکل پ۱-۲ مشاهده ورژن CPU در پنجره Module Info

آنچه در پنجره شکل پ۱-۲ دیده می‌شود، ممکن است با آنچه روی CPU نوشته شده یکی نباشد. این حالت نشان می‌دهد که ورژن CPU تغییر پیدا کرده است.

Version می‌تواند Upgrade یا Downgarde گردد. اگر چه بهتر است همواره ورژن را به آخرین نسخه ارائه شده ارتقا داد، ولی در عمل در برخی حالات کاهش ورژن[1] مورد نیاز است. نمونه بارز این حالت در سیستم‌های افزونه (S7-400H) است. سیستم H در صورتی به‌صورت افزونه کار می‌کند که مدل و ورژن دو CPU دقیقاً یکسان باشد. اگر در حین کار یکی از CPUها آسیب ببیند و نیاز به تعویض داشته باشد و CPU جدید خریداری شده دارای ورژنی بالاتر از ورژن CPU در حال کار باشد، در اینصورت بایستی ورژن CPU جدید را Downgrade کرد. توجه شود که نمی‌توان در حال کار ورژن CPU را تغییر داد زیرا تغییر ورژن با قطع و وصل تغذیه همراه است. بنابراین در این شرایط بایستی CPU جدید را کاهش داد و منتظر زمانی بود که بتوان سیستم را توقف داد. در این شرایط می‌توان ورژن‌های دو CPU را Upgrade نمود.

به‌جز سیستم‌های افزونه در برخی شرایط که CPU از نوع معمولی است باز نیاز به تغییر ورژن وجود دارد. به‌عنوان مثال برخی از CPUها که ورژن‌های قدیمی دارند بعضی از SFC/SFBها را پشتیبانی نمی‌کنند ولی با ارتقاء Firmware آنها این مشکل برطرف می‌شود.

1. Downgarde

به‌طور کلی ورژن CPU را به دو صورت می‌توان Update نمود:

الف) به‌صورت Online

ب) به‌صورت Offline

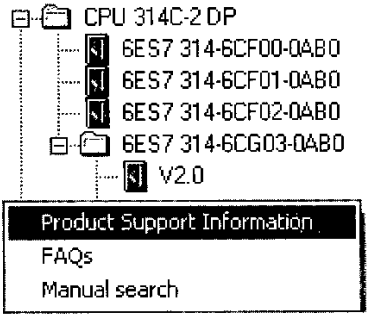
در حالت Online این کار از طریق نرم‌افزار Step7 انجام می‌شود. در حالت Offline تغییر ورژن توسط کارت حافظه فلش صورت می‌گیرد. این دو روش در ادامه تشریح شده‌اند.

تغییر ورژن CPU به صورت Online

برای هر CPU امکان تغییر ورژن به‌صورت Online وجود ندارد. به‌عنوان مثال برای CPUهای 400 که ورژن آنها بالاتر از 5.0 باشد این روش امکان‌پذیر است.

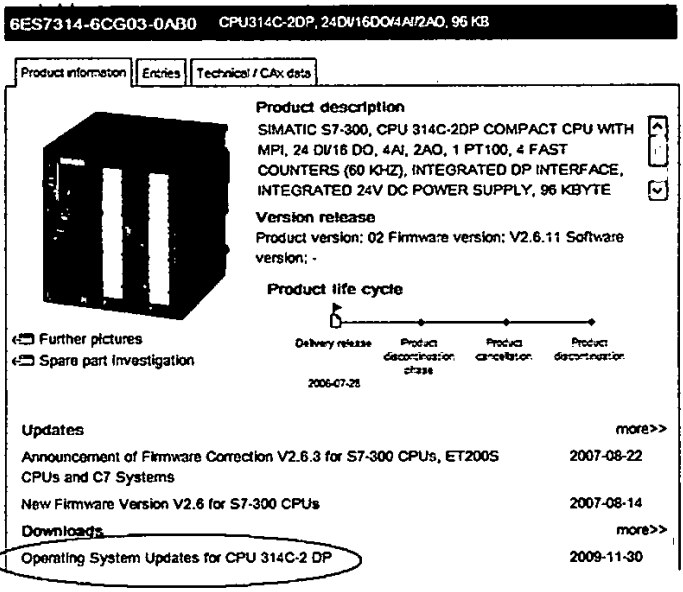
در این روش ابتدا فایل‌های Update مربوط به Firmware جدید، از سایت اینترنتی زیمنس دریافت شده و سپس از طریق اتصال مستقیم بین کامپیوتر و CPU، فایل‌های Update به CPU منتقل می‌گردد. مراحل انجام این روش به‌صورت زیر می‌باشد.

در کاتالوگ ماژول‌ها در برنامه HW Config، روی CPU مورد نظر کلیک راست نموده و مطابق شکل پ۱-۳ گزینه Product Suuport Information انتخاب می‌گردد.



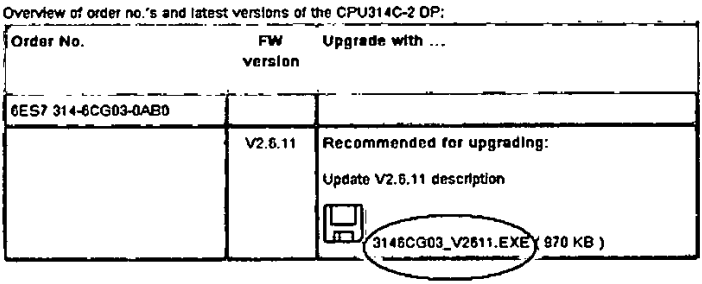
شکل پ۱-۳ مسیر دسترسی به فایل update

در اینصورت، به سایت اینترنتی زیمنس متصل شده و صفحه اختصاصی ماژول مورد نظر بارگذاری می‌شود. این صفحه برای CPU 314C-2DP به‌صورت نشان داده شده در شکل پ۱-۴ می‌باشد. در قسمت Downloads یک فایل Update برای سیستم عامل این CPU مشاهده می‌گردد. با کلیک بر روی این گزینه می‌توان به صفحه‌ای که فایل مورد نظر از آنجا قابل دانلود می‌باشد دسترسی پیدا نمود.



شکل پ۱-۴ مسیر فایل Update در سایت زیمنس

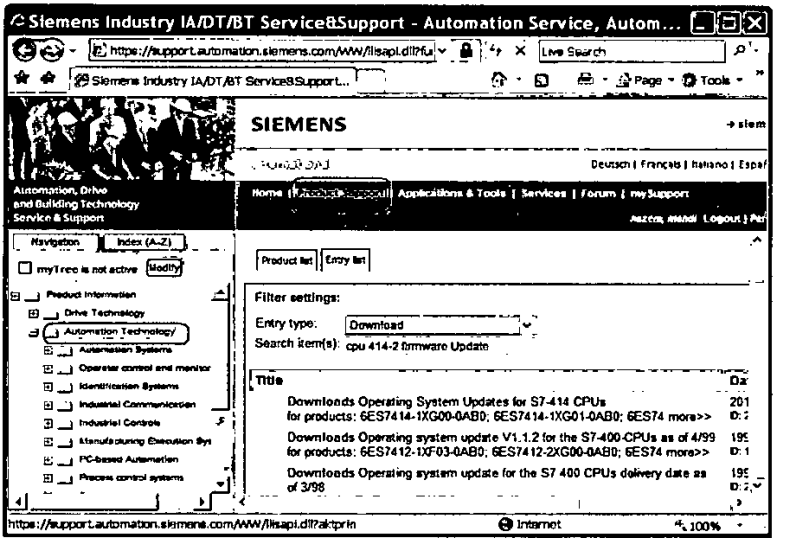
همانطور که در شکل پ۱-۴ مشخص است، آخرین ورژن این CPU تا زمان نگارش این کتاب، V2.6.11 می‌باشد. با کلیک بر روی قسمت نشان داده شده در شکل پ۱-۵ می‌توان فایل مورد نظر را دانلود نمود.



شکل پ۱-۵ دانلود فایل Update در سایت زیمنس

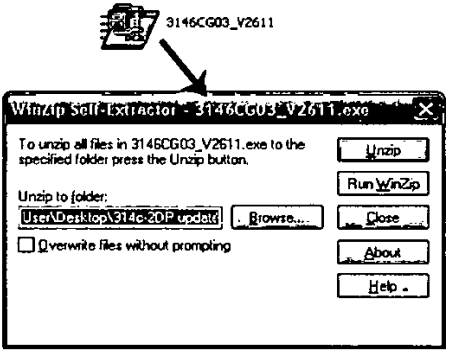
در صورتی‌که از روش فوق امکان دسترسی به سایت زیمنس نبود به آدرس زیر رفته و از بخش Product Spport بصورتی که در شکل پ۱-۶ نشان داده شده، فایل را جستجو کرده و دانلود کنید:

http://support.automation.siemens.com



شکل پ۱-۶ جستجوی فایل Update در سایت زیمنس

پس از اینکه فایل Update دانلود شد، مطابق شکل پ۱-۷ می‌توان بر روی فایل مورد نظر دوبارکلیک نموده و با انتخاب گزینه Unzip، آنرا در یک مسیر دلخواه باز نمود. فایل‌هایی با پسوند UPD در مسیر تعیین شده قرار می‌گیرند.



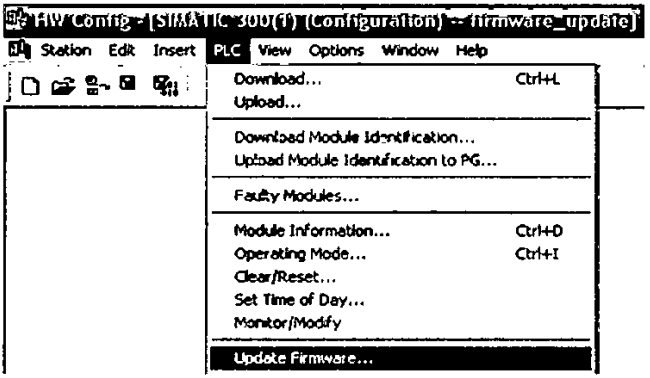
شکل پ۱-۷ Unzip نمودن فایل Update

پس از Unzip کردن سه فایل زیر در دسترس خواهند بود:

- BG_ABL.UPD
- CPU_HD.UPD
- KOMP_1.UPD

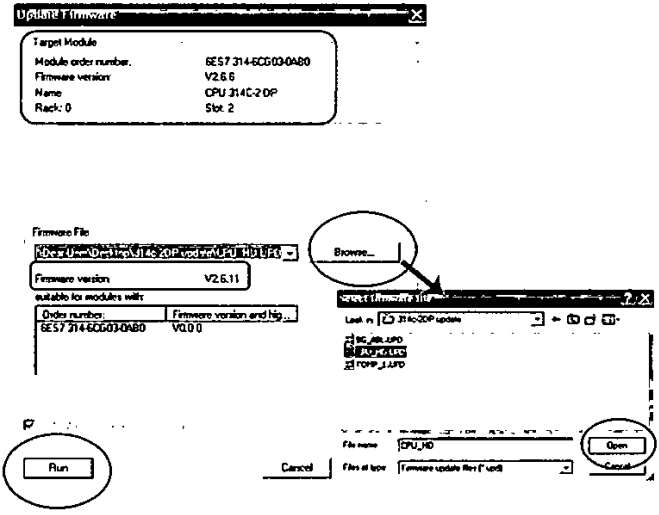
برای انتقال فایل مورد نظر به CPU، باید اتصال بین کامپیوتر و CPU را برقرار نمود. روش معمول، اتصال از طریق PC Adapter می‌باشد.

در محیط برنامه HW Config، روی CPU موجود در رک کلیک نموده و مطابق شکل پ۱-۸ از منوی PLC گزینه‌ی Update Firmware انتخاب می‌گردد.



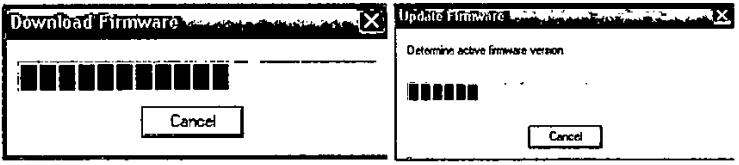
شکل پ۱-۸ انتخاب گزینه‌ی Update Firmware از منوی PLC

در صورتی‌که CPU قابلیت ارتقاء ورژن را پشتیبانی نماید، گزینه‌ی فوق، قابل انتخاب بوده و با انتخاب آن کادری به‌صورت شکل پ۱-۹ نمایان می‌شود. در این کادر در قسمت Target Module مشخصات CPU به همراه ورژن فعلی آن نمایش داده شده است. توسط گزینه‌ی Browse، فایل CPU_HD.UPD را از مسیر مورد نظر انتخاب و سپس بر روی گزینه‌ی Run کلیک می‌نماییم.



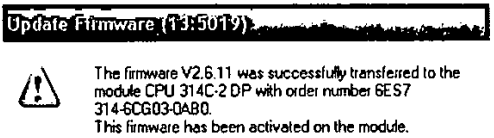
شکل پ۱-۹ مسیر انتخاب فایل Update

در اینصورت عملیات Update شروع شده و کادری به‌صورت شکل پ۱-۱۰ نمایش داده می‌شود.



شکل پ۱-۱۰ شروع کار Update

پس از پایان عملیات Update پیامی به‌صورت نشان داده شده در شکل پ۱-۱۱ مبنی بر پایان عملیات Update ورژن نمایش داده می‌شود.



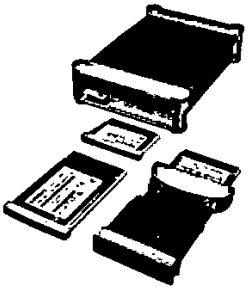
شکل پ۱-۱۱ پایان کار Update

پس از اتمام کار می‌توان نتیجه را در Module Information مربوط به CPU همانطور که در شکل پ۱-۲ نشان داده شد مشاهده نمود.

تغییر ورژن CPU به‌صورت Offline

در این روش کارت حافظه فلش مورد نیاز است که توسط PG یا USB Prommer زیمنس فایل UPD به آن منتقل گردد. ظرفیت کارت حافظه بسته به ورژن مورد نظر متفاوت است. به‌عنوان مثال برای CPUهای S7-400 کارت‌های حافظه زیر مورد نیاز است. ظرفیت‌های ذکر شده به‌عنوان حداقل است:

- Firmware-Version < 4.0.0: Flash 2 MB
- Firmware-Version >= 4.0.0: Flash 4 MB
- Firmware-Version >= 5.0.0: Flash 8 MB



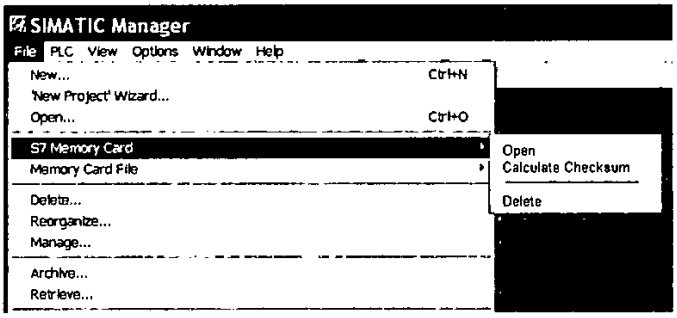
شکل پ-۱۲ پروگرامر زیمنس

پس از تهیه کارت حافظه به روشی که ذکر شد، فایل Update را از سایت زیمنس دانلود و آماده می‌کنیم. سپس مراحل زیر را انجام می‌دهیم:

۱- پاک کردن کارت حافظه

کارت را در اسلات PG یا پروگرامر قرار داده و توسط نرم‌افزار Step7 از مسیر زیر اطلاعات موجود در کارت حافظه را پاک می‌کنیم:

File > S7 Memory Card > Delete

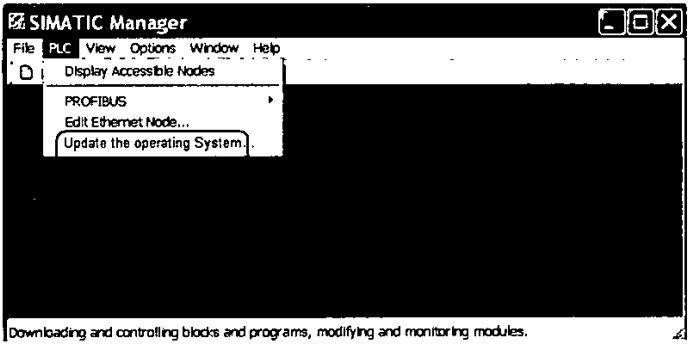


شکل پ۱-۱۳ مسیر پاک کردن کارت حافظه

تذکر: به هیچ وجه کارت حافظه زیمنس را از طریق کارت‌خوان‌های معمولی فرمت نکنید!

۲- انتقال فایل به کارت حافظه

از مسیر PLC > Update Operating System در محیط نرم‌افزارSimatic Manager، و در کادر باز شده مسیری که فایل‌های Update در آنجا قرار دارند انتخاب می‌گردد.



شکل پ۱-۱۴ مسیر انتقال فایل Update به کارت حافظه

در این زمان انتقال فایل‌ها به کارت حافظه شروع شده و پس پایان عملیات انتقال پیام زیر نمایش داده می‌شود:
The firmware update for the module was successfully transferred to the S7 memory card

۳- انتقال سیستم عامل از کارت حافظه به CPU

پس از اینکه کارت حافظه حاوی فایل Update تهیه گردید، ابتدا تغذیه‌ی CPU را قطع نموده و سپس آنرا درون اسلات مربوط به کارت حافظه در CPU قرار می‌دهیم. مجدداً تغذیه CPU را وصل نموده و آنرا روشن می‌نماییم. در این هنگام فایل مورد نظر به CPU منتقل می‌گردد. در طول مدت انتقال فایل از کارت حافظه به CPU، همه‌ی لامپ‌های روی CPU در وضعیت روشن قرار می‌گیرند. پس از حدود ۲ دقیقه که عملیات Update به پایان رسید، لامپ Stop به حالت چشمک‌زن در می‌آید. در این هنگام تغذیه CPU را قطع نموده و کارت حافظه‌را خارج می‌نماییم. پس از روشن نمودن مجدد CPU عمل ریست به صورت اتوماتیک انجام پذیرفته و CPU در حالت آماده به کار قرار می‌گیرد.

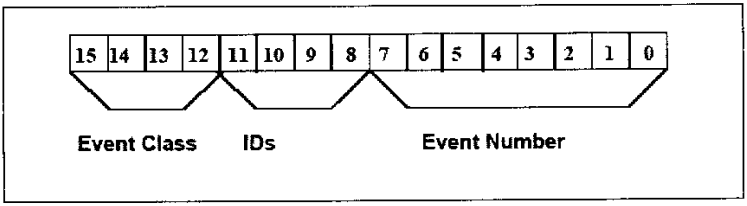
تذکر: در هر دو روش Online و Offline پس از اینکه کار تغییر ورژن تمام شد، تاریخ و زمان CPU را اصلاح کنید.

پیوست ۲

لیست Event IDهای بافر CPU

پیغام‌هایی که در Diagnostic Buffer مربوط به CPU ثبت می‌شوند هر کدام دارای ID خاصی هستند. لیست کامل Event IDها در این پیوست آمده است.

هر Event ID به‌صورت یک کد هگز ۱۶ بیتی ظاهر می‌شود که ساختاری شبیه شکل پ۲-۱ دارد.



شکل پ۲-۱ ساختار کد Event ID

چهار بیت سمت چپ معرف کلاس رخداد است که می‌تواند عدد هگز بین 1 تا F طبق جدول پ۲-۱ باشد.

جدول پ۲-۱

Event Class	
Number	**Event Class**
1	Standard OB events
2	Synchronous errors
3	Asynchronous errors
4	Mode transitions
5	Run-time events
6	Communication events
7	Events for fail-safe and fault-tolerant systems
8	Standardized diagnostic data on modules
9	Predefined user events
A, B	Freely definable events
C , D, E	Reserved
F	Events for modules other than CPUs (for example, CPs, FMs)

چهار بیت بعدی معرف ID رخ داد است که کدی به‌صورت جدول پ۲-۲ دارد.

جدول پ۲-۲

IDs	
Bit No. in the Event ID	**Meaning**
8	= 0 = 1 Event leaving state Event entering state
9	= 1 Entry in diagnostic buffer
10	= 1 Internal error
11	= 1 External error

لیست کامل Event IDها در ادامه آورده شده است

لیست Eventهای کلاس ۱ در جدول پ۲-۳ آمده است.

جدول پ۲-۳

Class 1 Event ID	
Event ID	**Event**
W#16#113A	Start request for cyclic interrupt OB with special handling (S7-300 only)
W#16#1155	Status alarm for PROFIBUS DP
W#16#1156	Update interrupt for PROFIBUS DP
W#16#1157	Manufacturer interrupt for PROFIBUS DP
W#16#1158	Status interrupt for PROFINET IO
W#16#1159	Update interrupt for PROFINET IO
W#16#115A	Manufacturer interrupt for PROFINET IO
W#16#115B	IO: Profile-specific interrupt
W#16#116A	Technology synchronization interrupt
W#16#1381	Request for manual warm restart
W#16#1382	Request for automatic warm restart
W#16#1383	Request for manual hot restart
W#16#1384	Request for automatic hot restart
W#16#1385	Request for manual cold restart
W#16#1386	Request for automatic cold restart
W#16#1387	Master CPU: request for manual cold restart
W#16#1388	Master CPU: request for automatic cold restart
W#16#138A	Master CPU: request for manual warm restart
W#16#138B	Master CPU: request for automatic warm restart
W#16#138C	Standby CPU: request for manual hot restart
W#16#138D	Standby CPU: request for automatic hot restart

لیست Eventهای کلاس ۲ در جدول پ۲-۴ آمده است. این Eventها در شرایط بروز خطاهای سنکرون که مربوط به برنامه‌نویسی هستند ثبت می‌گردند.

جدول پ۲-۴

Event Class 2 - Synchronous Errors		
Event ID	**Event**	**OB**
W#16#2521	BCD conversion error	OB 121
W#16#2522	Area length error when reading	OB 121
W#16#2523	Area length error when writing	OB 121
W#16#2524	Area error when reading	OB 121
W#16#2525	Area error when writing	OB 121
W#16#2526	Timer number error	OB 121
W#16#2527	Counter number error	OB 121
W#16#2528	Alignment error when reading	OB 121
W#16#2529	Alignment error when writing	OB 121
W#16#2530	Write error when accessing the DB	OB 121
W#16#2531	Write error when accessing the DI	OB 121
W#16#2532	Block number error when opening a DB	OB 121
W#16#2533	Block number error when opening a DI	OB 121
W#16#2534	Block number error when calling an FC	OB 121
W#16#2535	Block number error when calling an FB	OB 121
W#16#253A	DB not loaded	OB 121
W#16#253C	FC not loaded	OB 121
W#16#253D	SFC not loaded	OB 121
W#16#253E	FB not loaded	OB 121
W#16#253F	SFB not loaded	OB 121
W#16#2942	I/O access error, reading	OB 122
W#16#2943	I/O access error, writing	OB 122

لیست Eventهای کلاس ۳ در جدول پ۲-۵ آمده است. این Eventهای مربوط به خطاهای آسنکرون که بیشتر در ارتباط با سخت‌افزار هستند ثبت می‌گردند.

جدول پ۲-۵

Event Class 3 - Asynchronous Errors		
Event ID	**Event**	**OB**
W#16#3501	Cycle time exceeded.	OB 80
W#16#3502	User interface (OB or FRB) request error	OB 80

W#16#3503	Delay too long processing a priority class	-
W#16#3505	Time-of-day interrupt(s) skipped due to new clock setting	OB 80
W#16#3506	Time-of-day interrupt(s) skipped when changing to RUN after HOLD	OB 80
W#16#3507	Multiple OB request errors caused internal buffer overflow	OB 80
W#16#3508	Synchronous cycle interrupt-timing error	OB 80
W#16#3509	Interrupt loss due to excess interrupt load	OB 80
W#16#350A	Resume RUN mode after CiR	OB 80
W#16#350B	Technology synchronization interrupt - timing error	OB 80
W#16#3921/3821	BATTF: failure on at least one backup battery of the central rack/ problem eliminated	OB 81
W#16#3922/3822	BAF: failure of backup voltage on central rack/ problem eliminated	OB 81
W#16#3923/3823	24 volt supply failure on central rack / problem eliminated	OB 81
W#16#3925/3825	BATTF: failure on at least one backup battery of the redundant central rack/ problem eliminated	OB 81
W#16#3926/3826	BAF: failure of backup voltage on redundant central rack/ problem eliminated	OB 81
W#16#3917/3827	24 volt supply failure on redundant central rack / problem eliminated	OB 81
W#16#3931/3831	BATTF: failure of at least one backup battery of the expansion rack/ problem eliminated	OB 81
W#16#3932/3832	BAF: failure of backup voltage on expansion rack/ problem eliminated	OB 81
W#16#3933/3833	24 volt supply failure on at least one expansion rack/ problem eliminated	OB 81
W#16#3942	Module error	OB 82
W#16#3842	Module OK	OB 82
W#16#3951	PROFINET IO submodule removed	OB 83
W#16#3954	PROFINET IO interface submodule/submodule removed	OB 83
W#16#3854	PROFINET IO interface submodule/submodule and matches the configured interface submodule/submodule	OB 83
W#16#3855	PROFINET IO interface submodule/submodule inserted, but does not match the configured interface submodule/submodule	OB 83
W#16#3856	PROFINET IO interface submodule/submodule inserted, but error in module parameter assignment	OB 83
W#16#3858	PROFINET IO interface submodule access error	OB 83

	corrected	
W#16#3861	Module/interface module inserted, module type OK	OB 83
W#16#3961	Module/interface module removed, cannot be addressed	OB 83
W#16#3863	Module/interface module plugged in, but wrong module type	OB 83
W#16#3864	Module/interface module plugged in, but causing problem (type ID unreadable)	OB 83
W#16#3865	Module plugged in, but error in module parameter assignment	OB 83
W#16#3866	Module can be addressed again, load voltage error removed	OB 83
W#16#3966	Module cannot be addressed, load voltage error	OB 83
W#16#3367	Start of module reconfiguration	OB 83
W#16#3267	End of module reconfiguration	OB 83
W#16#3968	Module reconfiguration has ended with error	OB 83
W#16#3571	Nesting depth too high in nesting levels	OB 88
W#16#3572	Nesting depth for Master Control Relays too high	OB 88
W#16#3573	Nesting depth too high after synchronous errors	OB 88
W#16#3574	Nesting depth for block calls (U stack) too high	OB 88
W#16#3575	Nesting depth for block calls (B stack) too high	OB 88
W#16#3576	Local data allocation error	OB 88
W#16#3578	Unknown instruction	OB 88
W#16#357A	Jump instruction to target outside of the block	OB 88
W#16#3884	Interface module plugged in	OB 83
W#16#3984	Interface module removed	OB 83
W#16#3981	Interface error entering state	OB 84
W#16#3881	Interface error leaving state	OB 84
W#16#3582	Memory error detected and corrected by operating system	OB 84
W#16#3583	Accumulation of detected and corrected memo errors	OB 84
W#16#3585	Error in the PC operating system (only for LC RTX)	OB 84
W#16#3986	Performance of an H-Sync link negatively affected	OB 84
W#16#3587	Multi-bit memory error detected and corrected	OB 84
W#16#35A1	User interface (OB or FRB) not found	OB 85
W#16#35A2	OB not loaded (started by SFC or operating system due to configuration	OB 85
W#16#35A3	Error when operating system accesses a block	OB 85
W#16#35A4	PROFInet Interface DB cannot be addressed	OB 85

W#16#34A4	PROFInet Interface DB can be addressed again	OB 85
W#16#39B1	I/O access error when updating the process image input table	OB 85
W#16#39B2	I/O access error when transferring the process image to the output modules	OB 85
W#16#39B3/38B3	I/O access error when updating the process image input table	OB 85
W#16#39B4/38B4	I/O access error when transferring the process image to the output modules	OB 85
W#16#38C1	Expansion rack operational again (1 to 21), leaving state	OB 86
W#16#39C1	Expansion rack failure (1 to 21), entering state	OB 86
W#16#38C2	Expansion rack operational again but mismatch between setpoint and actual configuration	OB 86
W#16#39C3	Distributed I/Os: master system failure entering state	OB 86
W#16#39C4	Distributed I/Os: station failure, entering state	OB 86
W#16#38C4	Distributed I/Os: station failure, leaving state	OB 86
W#16#39C5	Distributed I/Os: station fault, entering state	OB 86
W#16#38C5	Distributed I/Os: station fault, leaving state	OB 86
W#16#38C6	Expansion rack operational again, but error(s) in module parameter assignment	OB 86
W#16#38C7	DP: station operational again, but error(s) in module parameter assignment	OB 86
W#16#38C8	DP: station operational again, but mismatch between setpoint and actual configuration	OB 86
W#16#39CA	PROFINET IO system failure	OB 86
W#16#39CB	PROFINET IO station failure	OB 86
W#16#38CB	PROFINET IO station operational again	OB 86
W#16#39CC	PROFINET IO station error	OB 86
W#16#38CC	PROFINET IO station error corrected	OB 86
W#16#39CD	PROFINET IO station operational again, but expected configuration does not match actual configuration	OB 86
W#16#39CE	PROFINET IO station operational again, but error(s) in module parameter assignment	OB 86
W#16#35D2	Diagnostic entries cannot be sent at present	OB 87
W#16#35D3	Synchronization frames cannot be sent	OB 87
W#16#35D4	Illegal time jump resulting from synchronization	OB 87
W#16#35D5	Error adopting the synchronization time	OB 87
W#16#35E1	Incorrect frame ID in GD	OB 87
W#16#35E2	GD packet status cannot be entered in DB	OB 87
W#16#35E3	Frame length error in GD	OB 87

W#16#35E4	Illegal GD packet number received	OB 87
W#16#35E5	Error accessing DB in communication SFBs for configured S7 connections	OB 87
W#16#35E6	GD total status cannot be entered in DB	OB 87

لیست Eventهای کلاس ۴ در جدول پ۲-۶ آمده است. این Eventها بیشتر مربوط به تغییر مدهای کاری CPU هستند.

جدول پ۲-۶

Event Class 4 - Stop Events and Other Mode Changes	
Event ID	**Event**
W#16#4300	Backed-up power on
W#16#4301	Mode transition from STOP to STARTUP
W#16#4302	Mode transition from STARTUP to RUN
W#16#4303	STOP caused by stop switch being activated
W#16#4304	STOP caused by PG STOP operation or by SFB 20 "STOP"
W#16#4305	HOLD: breakpoint reached
W#16#4306	HOLD: breakpoint exited
W#16#4307	Memory reset started by PG operation
W#16#4308	Memory reset started by switch setting
W#16#4309	Memory reset started automatically (power on not backed up)
W#16#430A	HOLD exited, transition to STOP
W#16#430D	STOP caused by other CPU in multicomputing
W#16#430E	Memory reset executed
W#16#430F	STOP on the module due to STOP on a CPU
W#16#4510	STOP violation of the CPU's data range
W#16#4318	Start of CiR
W#16#4319	CiR completed
W#16#4520	DEFECTIVE: STOP not possible
W#16#4521	DEFECTIVE: failure of instruction processing processor
W#16#4522	DEFECTIVE: failure of clock chip
W#16#4523	DEFECTIVE: failure of clock pulse generator
W#16#4524	DEFECTIVE: failure of timer update function
W#16#4525	DEFECTIVE: failure of multicomputing synchronization
W#16#4926	DEFECTIVE: failure of the watchdog for I/O access
W#16#4527	DEFECTIVE: failure of I/O access monitoring
W#16#4528	DEFECTIVE: failure of scan time monitoring
W#16#4530	DEFECTIVE: memory test error in internal memory

W#16#4931	STOP or DEFECTIVE: memory test error in memory submodule
W#16#4532	DEFECTIVE: failure of core resources
W#16#4933	Checksum error
W#16#4934	DEFECTIVE: memory not available
W#16#4935	DEFECTIVE: cancelled by watchdog/processor exceptions
W#16#4536	DEFECTIVE: switch defective
W#16#4540	STOP: Memory expansion of the internal work memory has gaps. First memory expansion too small or missing.
W#16#4541	STOP caused by priority class system
W#16#4542	STOP caused by object management system
W#16#4543	STOP caused by test functions
W#16#4544	STOP caused by diagnostic system
W#16#4545	STOP caused by communication system
W#16#4546	STOP caused by CPU memory management
W#16#4547	STOP caused by process image management
W#16#4548	STOP caused by I/O management
W#16#4949	STOP caused by continuous hardware interrupt
W#16#454A	STOP caused by configuration: an OB deselected with STEP 7 was being loaded into the CPU during STARTUP
W#16#494D	STOP caused by I/O error
W#16#494E	STOP caused by power failure
W#16#494F	STOP caused by configuration error
W#16#4550	DEFECTIVE: internal system error
W#16#4555	No restart possible, monitoring time elapsed
W#16#4556	STOP: memory reset request from communication system / due to data inconsistency
W#16#4357	Module watchdog started
W#16#4358	All modules are ready for operation
W#16#4959	One or more modules not ready for operation
W#16#4562	STOP caused by programming error (OB not loaded or not possible
W#16#4563	STOP caused by I/O access error (OB not loaded or not possible
W#16#4567	STOP caused by H event
W#16#4568	STOP caused by time error (OB not loaded or not possible)
W#16#456A	STOP caused by diagnostic interrupt (OB not loaded or not possible)
W#16#456B	STOP caused by removing/inserting module (OB not loaded or not possible)
W#16#456C	STOP caused by CPU hardware error (OB not loaded or not possible, or no FRB)STOP
W#16#456D	STOP caused by program sequence error (OB not loaded or not possible)

W#16#456E	STOP caused by communication error (OB not loaded or not possible)
W#16#456F	STOP caused by rack failure OB (OB not loaded or not possible)
W#16#4570	STOP caused by process interrupt (OB not loaded or not possible)
W#16#4571	STOP caused by nesting stack error
W#16#4572	STOP caused by master control relay stack error
W#16#4573	STOP caused by exceeding the nesting depth for synchronous errors
W#16#4574	STOP caused by exceeding interrupt stack nesting depth in the priority class stack
W#16#4575	STOP caused by exceeding block stack nesting depth in the priority class stack
W#16#4576	STOP caused by error when allocating the local data
W#16#4578	STOP caused by unknown opcode
W#16#457A	STOP caused by code length error
W#16#457B	STOP caused by DB not being loaded on on-board I/Os
W#16#497C	STOP caused by integrated technology
W#16#457D	Reset/clear request because the version of the internal interface to the integrated technology was changed.
W#16#457F	STOP caused by STOP command
W#16#4580	STOP: back-up buffer contents inconsistent (no transition to RUN)
W#16#4590	STOP caused by overloading the internal functions
W#16#49A0	STOP caused by parameter assignment error or non-permissible variation of setpoint and actual extension: Start-up blocked.
W#16#49A1	STOP caused by parameter assignment error: memory reset request
W#16#49A2	STOP caused by error in parameter modification: startup disabled
W#16#49A3	STOP caused by error in parameter modification: memory reset request
W#16#49A4	STOP: inconsistency in configuration data
W#16#49A5	STOP: distributed I/Os: inconsistency in the loaded configuration information
W#16#49A6	STOP: distributed I/Os: invalid configuration information
W#16#49A7	STOP: distributed I/Os: no configuration information
W#16#49A8	STOP: error indicated by the interface module for the distributed I/Os
W#16#43B0	Firmware update was successful
W#16#49B1	Firmware update data incorrect
W#16#49B2	Firmware update: hardware version does not match firmware
W#16#49B3	Firmware update: module type does not match firmware
W#16#43B4	Error in firmware fuse
W#16#43B6	Firmware updates canceled by redundant modules
W#16#49D0	LINK-UP aborted due to violation of coordination rules
W#16#49D1	LINK-UP/UPDATE sequence aborted

W#16#49D2	Standby CPU changed to STOP due to STOP on the master CPU during link-up
W#16#43D3	STOP on standby CPU
W#16#49D4	STOP on a master, since partner CPU is also a master (link-up error)
W#16#45D5	LINK-UP rejected due to mismatched CPU memory configuration of the sub-PLC
W#16#45D6	LINK-UP rejected due to mismatched system program of the sub-PLC
W#16#49D7	LINK-UP rejected due to change in user program or in configuration
W#16#45D8	DEFECTIVE: hardware fault detected due to other error
W#16#45D9	STOP due to SYNC module error
W#16#45DA	STOP due to synchronization error between H CPUs
W#16#43DC	Abort during link-up with switchover
W#16#45DD	LINK-UP rejected due to running test or other online functions
W#16#43DE	Updating aborted due to monitoring time being exceeded during the n-th attempt, new update attempt initiated
W#16#43DF	Updating aborted for final time due to monitoring time being exceeded after completing the maximum amount of attempts. User intervention required
W#16#43E0	Change from solo mode after link-up
W#16#43E1	Change from link-up after updating
W#16#43E2	Change from updating to redundant mode
W#16#43E3	Master CPU: change from redundant mode to solo mode
W#16#43E4	Standby CPU: change from redundant mode after error-search mode
W#16#43E5	Standby CPU: change from error-search mode after link-up or STOP
W#16#43E6	Link-up aborted on the standby CPU
W#16#43E7	Updating aborted on the standby CPU
W#16#43E8	Standby CPU: change from link-up after startup
W#16#43E9	Standby CPU: change from startup after updating
W#16#43F1	Reserve-master switchover
W#16#43F2	Coupling of incompatible H-CPUs blocked by system program
W#16#42F3	Checksum error detected and corrected by the operating system
W#16#42F4	Standby CPU: connection/update via SFC90 is locked in the master CPU

لیست Eventهای کلاس ۵ در جدول پ۲-۷ آمده است.

جدول پ۲-۷

Event Class 5 - Mode Run-time Events	
Event ID	**Event**
W#16#530D	New startup information in the STOP mode
W#16#510F	A problem as occurred with WinLC. This problem has caused the CPU to go into STOP mode or has caused a fault in the CPU
W#16#5311	Startup despite Not Ready message from module(s)

W#16#5545	Start of System reconfiguration in RUN mode
W#16#5445	Start of System reconfiguration in RUN mode
W#16#5380	Diagnostic buffer entries of interrupt and asynchronous errors disabled
W#16#5395	Distributed I/Os: reset of a DP master
W#16#5481	All licenses for runtime software are complete again.
W#16#5498	No more inconsistency with DP master systems due to CiR
W#16#5581	One or several licenses for runtime software are missing.
W#16#558A	Difference between the MLFB of the configured and inserted CPU
W#16#558B	Difference in the firmware version of the configured and inserted CPU
W#16#5598	Start of possible inconsistency with DP master systems due to CiR
W#16#5960	Parameter assignment error when switching
W#16#5961	Parameter assignment error
W#16#5962	Parameter assignment error preventing startup
W#16#5963	Parameter assignment error with memory reset request
W#16#5966	Parameter assignment error when switching
W#16#5969	Parameter assignment error with startup blocked
W#16#596A	PROFINET IO: IP address of an IO device already present
W#16#596B	IP address of an Ethernet interface already exists
W#16#596C	Name of an Ethernet interface already exists
W#16#596D	The existing network configuration does not mach the system requirements or configuration
W#16#5371	Distributed I/Os: end of the synchronization with a DP master
W#16#5979/5879	Diagnostic message from DP interface: EXTF LED on/off
W#16#597C	DP Global Control command failed or moved
W#16#597C	DP command Global Control failure or moved
W#16#59A0	The interrupt can not be associated in the CPU
W#16#59A1	Configuration error in the integrated technology
W#16#53A2	Download of technology firmware successful
W#16#59A3	Error when downloading the integrated technology
W#16#53A4	Download of technology DB not successful
W#16#55A5	Version conflict: internal interface with integrated technology
W#16#55A6	The maximum number of technology objects has been exceeded.
W#16#55A7	A technology DB of this type is already present.
W#16#53FF	Reset to factory setting

لیست Eventهای کلاس ۶ در جدول پ۲-۸ آمده است.

جدول پ۲-۸

Event Class 6 - Communication Events	
Event ID	**Event**
W#16#6316	Interface error when starting programmable controller
W#16#6390	Formatting of Micro Memory Card complete
W#16#6500	Connection ID exists twice on module
W#16#6501	Connection resources inadequate
W#16#6502	Error in the connection description
W#16#6510	CFB structure error detected in instance DB when evaluating EPROM
W#16#6514	GD packet number exists twice on the module
W#16#6515	Inconsistent length specifications in GD configuration information
W#16#6521	No memory submodule and no internal memory available
W#16#6522	Illegal memory submodule: replace submodule and reset memory
W#16#6523	Memory reset request due to error accessing submodule
W#16#6524	Memory reset request due to error in block header
W#16#6526	Memory reset request due to memory replacement
W#16#6527	Memory replaced, therefore restart not possible
W#16#6528	Object handling function in the STOP/HOLD mode, no restart possible
W#16#6529	No startup possible during the "load user program" function
W#16#652A	No startup because block exists twice in user memory
W#16#652B	No startup because block is too long for submodule - replace submodule
W#16#652C	No startup due to illegal OB on submodule
W#16#6532	No startup because illegal configuration information on submodule
W#16#6533	Memory reset request because of invalid submodule content
W#16#6534	No startup: block exists more than once on submodule
W#16#6535	No startup: not enough memory to transfer block from submodule
W#16#6536	No startup: submodule contains an illegal block number
W#16#6537	No startup: submodule contains a block with an illegal length
W#16#6538	Local data or write-protection ID (for DB) of a block illegal for CPU
W#16#6539	Illegal command in block (detected by compiler)
W#16#653A	Memory reset request because local OB data on submodule too short
W#16#6543	No startup: illegal block type
W#16#6544	No startup: attribute "relevant for processing" illegal
W#16#6545	Source language illegal
W#16#6546	Maximum amount of configuration information reached
W#16#6547	Parameter assignment error assigning parameters to modules (not on P

	bus, cancel download)
W#16#6548	Plausibility error during block check
W#16#6549	Structure error in block
W#16#6550	A block has an error in the CRC
W#16#6551	A block has no CRC
W#16#6353	Firmware update: Start of firmware download over the network
W#16#6253	Firmware update: End of firmware download over the network
W#16#6560	SCAN overflow
W#16#6881	Interface error leaving state
W#16#6905/6805	Resource problem on configured connections/eliminated
W#16#6981	Interface error entering state

لیست Eventهای کلاس ۷ در جدول پ۲-۹ آمده است. این Eventها مربوط به سیستم‌های F و H و FH هستند.

جدول پ۲-۹

Event Class 7 - H/F Events		
Event ID	**Event**	**OB**
W#16#72A2	Failure of a DP master or a DP master system	OB 70
W#16#72A3	Redundancy restored on the DP slave	OB 70
W#16#7301	Loss of redundancy (1 of 2) due to failure of a CPU	OB 72
W#16#7302	Loss of redundancy (1 of 2) due to STOP on the standby triggered by user	OB 72
W#16#7303	H system (1 of 2) changed to redundant mode	OB 72
W#16#7323	Discrepancy found in operating system data	OB 72
W#16#7331	Standby-master switchover due to master failure	OB 72
W#16#7333	Standby-master switchover due to system modification during runtime	OB 72
W#16#7334	Standby-master switchover due to communication error at the synchronization module	OB 72
W#16#7340	Synchronization error in user program due to elapsed wait time	OB 72
W#16#7341	Synchronization error in user program due to waiting at different synchronization points	OB 72
W#16#7342	Synchronization error in operating system due to waiting at different synchronization points	OB 72

W#16#7343	Synchronization error in operating system due to elapsed wait time	OB 72
W#16#7344	Synchronization error in operating system due to incorrect data	OB 72
W#16#734A	The "Re-enable" job triggered by SFC 90 "H_CTRL" was executed.	OB 72
W#16#73A3	Loss of redundancy on the DP slave	OB 70
W#16#73D8	Safety mode disabled	
W#16#73E0/72E0	Loss of redundancy in communication/ problem eliminated	OB 73
W#16#7520	Error in RAM comparison	OB 72
W#16#7521	Error in comparison of process image output value	OB 72
W#16#7522	Error in comparison of memory bits, timers, or counters	OB 72
W#16#73C1	Update process canceled	OB 72
W#16#73C2	Updating aborted due to monitoring time being exceeded during the n-th attempt (1 ≤n ≤max. possible number of update attempts after abort due to excessive monitoring time)	OB 72
W#16#75D1	Safety program: Internal CPU error	
W#16#75D2	Safety program error: Cycle time time-out	
W#16#75D6	Data corrupted in safety program prior to the output to F I/O	
W#16#75D7	Data corrupted in safety program prior to the output to partner F CPU	
W#16#75D9	Invalid REAL number in a DB	
W#16#75DA	Safety program: Error in safety data format	
W#16#73DB/72DB	Safety program: safety mode enabled/disabled	
W#16#75DC	Runtime group, internal protocol error	
W#16#75DD/74DD	Safety program: Shutdown of a fail-save runtime group enabled/disabled	
W#16#75DE/74DE	Safety program: Shutdown of the F program enabled/disabled	-
W#16#75DF/74DF	Start / end of F program initialization	-
W#16#75E1	Safety program: Error in FB "F_PLK" or "F_PLK_O" or F_CYC_CO" or "F_TEST" or "F_TESTC"	
W#16#7934	Standby-master switchover due to connection problem at the SYNC module	OB 72
W#16#7950	Synchronization module missing	OB 72
W#16#7951	Change at the SYNC module without Power On	OB 72
W#16#7952/7852	SYNC module removed/inserted	OB 72

W#16#7953	Change at the SYNC-module without reset	OB 72
W#16#7954	SYNC module: rack number assigned twice	OB 72
W#16#7955/7855	SYNC module error/eliminated	OB 72
W#16#7956	Illegal rack number set on SYNC module	OB 72
W#16#7960	Redundant I/O: Time-out of discrepancy time at digital input, error is not yet localized	
W#16#7961	Redundant I/O, digital input error: Signal change after expiration of the discrepancy time	
W#16#7962	Redundant I/O: Digital input error	-
W#16#796F	Redundant I/O: The I/O was globally disabled	-
W#16#7970	Redundant I/O: Digital output error	-
W#16#7980	Redundant I/O: Time-out of discrepancy time at analog input	-
W#16#7981	Redundant I/O: Analog input error	-
W#16#7990	Redundant I/O: Analog output error	-
W#16#79D3/78D3	Communication error between PROFIsafe and F I/O	-
W#16#79D4/78D4	Error in safety relevant communication between F CPUs	-
W#16#79D5/78D5	Error in safety relevant communication between F CPUs	-
W#16#75E2	Safety program: Area length error	-
W#16#79E3	F-I/O device input channel passivated	-
W#16#78E3	F-I/O device input channel depassivated	-
W#16#79E4	F-I/O device output channel passivated	-
W#16#78E4	F-I/O device output channel depassivated	-
W#16#79E5	F-I/O device passivated	-
W#16#78E5	F-I/O device depassivated	-
W#16#79E6	Inconsistent safety program	-
W#16#79E7	Simulation block (F system block) loaded	-

لیست Eventهای کلاس ۸ در جدول پ۲-۱۰ آمده است. این Eventها مربوط به ماژول‌هایی که قابلیت Diagnostic دارند می‌باشد.

جدول پ۲-۱۰

Event Class 8 - Diagnostic Events for Modules		
Event ID	**Event**	**Module type**
W#16#8x00	Module fault/OK	Any
W#16#8x01	Internal error	
W#16#8x02	External error	

W#16#8x03	Channel error	
W#16#8x04	No external auxiliary voltage	
W#16#8x05	No front connector	
W#16#8x06	No parameter assignment	
W#16#8x07	Incorrect parameters in module	
W#16#8x30	User submodule incorrect/not found	
W#16#8x31	Communication problem	
W#16#8x32	Operating mode: RUN/STOP (STOP: entering state, RUN: leaving state)	
W#16#8x33	Time monitoring responded (watchdog)	
W#16#8x34	Internal module power failure	
W#16#8x35	BATTF: battery exhausted	
W#16#8x36	Total backup failed	
W#16#8x40	Expansion rack failed	
W#16#8x41	Processor failure	
W#16#8x42	EPROM error	
W#16#8x43	RAM error	
W#16#8x44	ADC/DAC error	
W#16#8x45	Fuse blown	
W#16#8x46	Hardware interrupt lost	
W#16#8x50	Configuration/parameter assignment error	Analog input
W#16#8x51	Common mode error	
W#16#8x52	Short circuit to phase	
W#16#8x53	Short circuit to ground	
W#16#8x54	Wire break	
W#16#8x55	Reference channel error	
W#16#8x56	Below measuring range	
W#16#8x57	Above measuring range	
W#16#8x60	Configuration/parameter assignment error	Analog output
W#16#8x61	Common mode error	
W#16#8x62	Short circuit to phase	
W#16#8x63	Short circuit to ground	
W#16#8x64	Wire break	
W#16#8x66	No load voltage	
W#16#8x70	Configuration/parameter assignment error	Digital input

W#16#8x71	Chassis ground fault	
W#16#8x72	Short circuit to phase (sensor)	
W#16#8x73	Short circuit to ground (sensor)	
W#16#8x74	Wire break	
W#16#8x75	No sensor power supply	
W#16#8x80	Configuration/parameter assignment error	Digital output
W#16#8x81	Chassis ground fault	
W#16#8x82	Short circuit to phase	
W#16#8x83	Short circuit to ground	
W#16#8x84	Wire break	
W#16#8x85	Fuse tripped	
W#16#8x86	No load voltage	
W#16#8x87	Excess temperature	
W#16#8xB0	Counter module, signal A faulty	FM
W#16#8xB1	Counter module, signal B faulty	
W#16#8xB2	Counter module, signal N faulty	
W#16#8xB3	Counter module, incorrect value passed between the channels	
W#16#8xB4	Counter module, 5.2 V sensor supply faulty	
W#16#8xB5	Counter module, 24 V sensor supply faulty	

لیست Eventهای کلاس ۹ در جدول پ۲-۱۱ آمده است.

جدول پ۲-۱۱

Event Class 9 - Standard User Events	
Event ID	**Event**
W#16#9001	Automatic mode
W#16#9101	Manual mode
W#16#9x02	OPEN/CLOSED, ON/OFF
W#16#9x03	Manual command enable
W#16#9x04	Unit protective command (OPEN/CLOSED)
W#16#9x05	Process enable
W#16#9x06	System protection command
W#16#9x07	Process value monitoring responded
W#16#9x08	Manipulated variable monitoring responded

W#16#9x09	System deviation greater than permitted
W#16#9x0A	Limit position error
W#16#9x0B	Runtime error
W#16#9x0C	Command execution error (sequencer)
W#16#9x0D	Operating status running > OPEN
W#16#9x0E	Operating status running > CLOSED
W#16#9x0F	Command blocking
W#16#9x11	Process status OPEN/ON
W#16#9x12	Process status CLOSED/OFF
W#16#9x13	Process status intermediate position
W#16#9x14	Process status ON via AUTO
W#16#9x15	Process status ON via manual
W#16#9x16	Process status ON via protective command
W#16#9x17	Process status OFF via AUTO
W#16#9x18	Process status OFF via manual
W#16#9x19	Process status OFF via protective command
W#16#9x21	Function error on approach
W#16#9x22	Function error on leaving
W#16#9x31	Actuator (DE/WE) limit position OPEN
W#16#9x32	Actuator (DE/WE) limit position not OPEN
W#16#9x33	Actuator (DE/WE) limit position CLOSED
W#16#9x34	Actuator (DE/WE) limit position not CLOSED
W#16#9x41	Illegal status, tolerance time elapsed
W#16#9x42	Illegal status, tolerance time not elapsed
W#16#9x43	Interlock error, tolerance time = 0
W#16#9x44	Interlock error, tolerance time > 0
W#16#9x45	No reaction
W#16#9x46	Final status exited illegally, tolerance time = 0
W#16#9x47	Final status exited illegally, tolerance time > 0
W#16#9x50	Upper limit of signal range USR
W#16#9x51	Upper limit of measuring range UMR
W#16#9x52	Lower limit of signal range LSR
W#16#9x53	Lower limit of measuring range LMR
W#16#9x54	Upper alarm limit UAL
W#16#9x55	Upper warning limit UWL

W#16#9x56	Upper tolerance limit UTL
W#16#9x57	Lower tolerance limit LTL
W#16#9x58	Lower warning limit LWL
W#16#9x59	Lower alarm limit LAL
W#16#9x60	GRAPH7 step entering/leaving
W#16#9x61	GRAPH7 interlock error
W#16#9x62	GRAPH7 execution error
W#16#9x63	GRAPH7 error noted
W#16#9x64	GRAPH7 error acknowledged
W#16#9x70	Trend exceeded in positive direction
W#16#9x71	Trend exceeded in negative direction
W#16#9x72	No reaction
W#16#9x73	Final state exited illegally
W#16#9x80	Limit value exceeded, tolerance time = 0
W#16#9x81	Limit value exceeded, tolerance time > 0
W#16#9x82	Below limit value, tolerance time = 0
W#16#9x83	Below limit value, tolerance time > 0
W#16#9x84	Gradient exceeded, tolerance time = 0
W#16#9x85	Gradient exceeded, tolerance time > 0
W#16#9x86	Below gradient, tolerance time = 0
W#16#9x87	Below gradient, tolerance time > 0
W#16#9190/9090	User parameter assignment error entering/leaving
W#16#91F0	Overflow
W#16#91F1	Underflow
W#16#91F2	Division by 0
W#16#91F3	Illegal calculation operation

لیست Eventهای کلاس‌های A و B که مربوط به پیغام‌های دلخواه کاربر است، در جدول پ۲-۱۲ آمده است.

جدول پ۲-۱۲

Event Classes A and B - Free User Events	
Event ID	**Event**
W#16#Axyz	Events available for user
W#16#Bxyz	

پیوست ۳

لیست کدهای Error در Step7

در حین کار با نرم‌افزار Step7 ممکن است کاربر با Errorهای مختلفی مواجه شود. لیست کامل Errorها و علت آنها در این پیوست آورده شده است.

Error Number	Text	Cause
1:2063	TBI error 2063 Error during the function chdir	TBI can occur with the call AUTgetlinkedObjects. There should be an AUT API tracer in the meantime.
7:6160	CPUs can only be connected via K bus or MPI bus (line 1)	
11:103	Cannot read CPU information on module, such as rated type, actual type and status of the module. Module might not be configured.	
13:31		The online connection from the PG to the module (SDB addressee) could not be set up. Connection hardware is faulty (e.g. connection cable removed).
		The online connection from the PG to the module (SDB addressee) could not be set up. The module (SDB addressee) is in the wrong operating mode.
		The online connection from the PG to the module (SDB addressee) could not be set up. You have specified an incorrect MPI address with STEP 7.
13:32	There is a problem with the connection to the receiver of the HW configuration	The online connection from the PG to the module (SDB addressee) could not be set up. The module recognized an error with the next communications call.
		Connection hardware is faulty (e.g. connection cable removed)
		Module is in the wrong operating mode
		Storage bottleneck on the module
		Inconsistent HW configuration in the PG
		When loading the system data on to the memory card, this message is displayed if there are invalid SDBs in the SDB folder.
13:48	Internal error 0400 Station cannot be generated.	The station cannot be created in the selected project because ...
		The project directory is write-protected
		The project directory is on a network drive and the connection cannot be set up
		No more storage capacity on the target drive
13:76	Inconsistent hardware configuration in the programming device.	Inconsistent HW configuration in the PG
		Folder "...\SIEMENS\STEP7\S7TMP" is missing or write-protected
		Missing authorization or an options package that generates the SDBs for a specific module.
	"Error in generating the system data" when	The environment tag S7TMP is not set. The path

	downloading or saving a station in HW Config	c:\\SDBDATA\s7hwcnfk\down\r00s02\sdb0.dat is completely or partly invalid
13:77	While interpreting the system data blocks (SDBs) uploaded from the module, an error was recognized in the programming device and processing halted .	Inconsistent HW configuration in module, occurs when previous loading of SDBs into the module was incomplete or faulty.
		Options package for configuring specific modules (e.g. CPs) is not installed
13:136	The maximum possible number of slots within all inserted slaves has been exceeded by %1 slot(s)."	More slaves have been configured than permitted by the master system
13:181	Internal error:Multicomputig cannot have the type "inhomogenous"	Error in HW Config of first 4.02.x version
13:422	No master system is assigned to the DP slave.	Missing object
13:520	The address entered is already occupied. STEP 7 has already calculated the next free address.	
13:4003	Module cannot be inserted here. You can use only DP slaves in a master system.	An attempt was made to connect a device directly to the DP bus, which does not have its own interface.
13:4040	The number of slots of the CPU or the master system has been exceeded or exhausted.	Maximum number of modules already configured.
13:4241	The offline configuration (%2) differs from the hardware configuration of the PLC (%3). Do you still want to continue?	An external module (interbus interfacing) from the Phoenix company was slotted
13:4337	Unable to set up connection to module %1 (R%3!d!/S%4!d!).	The online connection from the PG to the module (SDB addressee) could not be set up.
		Connection hardware is faulty.
		Module (SDB addressee) is in the wrong operating mode.
		You have specified an incorrect MPI address in the dialog box when loading to a PLC.
		PG/PC with PROFIBUS interface is connected directly with an intelligent DP slave (e.g. ET 200X BM147/CPU) and the PG/PC interface is set incorrectly.

13:4343	An update is presently not possible. In one or more STEP 7 applications at least one GSE file or type file is being referenced.	In the different S7 applications there are objects (stations) open in which DP slaves are used. Thus it is no longer possible to install new GSD or type files, or even to update them.
13:4341	The GSE file (type file) %1 contains syntax errors. For this reason it cannot be interpreted.	STEP7 V5 + SP3 does a strict check of the GSD files. Some are not accepted, e.g. when the name is longer than 23 characters
13:4366	Error during complete restart of the module.	An error has occurred when attempting to start the module (restart). Possibly not all the CPUs of the multicomputing station have been started.
		An error has occurred when attempting to start the module (restart). The key switch of the CPU/FM is in the STOP position
		An error has occurred when attempting to start the module (restart). The operating mode switch of the CP is in the STOP position
13:4589	Details: (D221); cannot load SDB 7XX".	H connections are configured in your project. The configuration of high-availability S7 connections has changed in STEP 7 V5.1. There are individual old configurations whose system data after a change can no longer be assigned to the latest configured high-availability S7 connections. This then leads to abortion of the download.
15:2053 3	The maximal length of debug info (max. 64 Kbyte) is reached.	The maximal length of debug info of 64 Kbyte is dependent of the system and can not be added.
16:5016	Parameter can not be controlled	You are trying to control an operand whose address is outside the permitted range of the controlling CPU.
		You are trying to control an operand outside the process image for which there is no module available.
		The data block (DB) that is to be controlled is not in the CPU to be controlled. The data block (DB) is write-protected.
		The data block (DB) is write-protected.
		You are trying to control a periphery input (PI).
		With the function "Release PQ" you can only control peripheral outputs (PQ).
16:5053		In "Process Mode" it is not permitted to control or monitor I/Os.
30:13		The interface of a called block has been changed. This means that the parameters are not transferred properly to the changed block. The block call is expanded and then displayed marked as faulty.
30:15		The type object has a different time stamp to the opened block. The local symbols are displayed as pseudosymbols.
30:50	Error while generating the STL source.	The STL source cannot be generated, e.g. because it exists already and is write-protected, or because there is no more memory available on the drive.

30:484	The displayed block cannot be monitored because it does not match the block in the CPU. Do you want to download the displayed block to the CPU and then monitor it?	This is found to be caused by a discrepancy in the time stamps between the online block and the offline block and means that the blocks are different. Since the online information relating to the block sequence cannot be assigned to the individual program elements of the offline block, the block status cannot be displayed.
30:504	There are no parameter and local variable names available because there is a time stamp conflict between the interface of the block and the description of the parameter and local variable names.	The description of the parameter and local variable names has a different time stamp to the interface of the opened block. This is why the parameters and local variables are displayed as pseudosymbols without comments.
		The cause for this can be an actual hardware fault, or an interrupt conflict with another module.
30:53	Problems occurred during the automatic generation of STL sources. See Details!	There are protected blocks in the selection list of the STL source to be generated.
30:202		The test that "strikes" should check whether the block is there online and offline.
30:473	This operation is not possible in this position.	Programming error.
30:503		The type object of the block is not available, so the variable description is missing. The local symbols are displayed as pseudosymbols.
		The block is available online in the automation system, but not offline in the S7 user program.
		The block has been copied from the automation system into an S7 user program and now this S7 user program is to be opened where it did not exist before.
		A block is opened via "Available Users". In this case there is no S7 user program linked with the automation system.
		The associated function block of an instance data block to be opened does not exist.
32:282	PG Resourcen Error, by program status of a longer FB in PLCSIM	In Step7 V5.0 SP2 with PLCSIM V4.0 under NT, project and block were opened online. This probably led to high internal use of resources.
33:11		
33:384	The triggered action cannot be executed at this time. Check all other STEP 7 applications and exit them where necessary	The communications module is probably already being operated with other settings (e.g. with a different transmission rate).
		There might be other online functions active.
		Attempt to run the firmware update of a CPU directly via STEP 7 on a CPU
33:496	The called function is not available in S7-DOS or in the CPU	

33:498	Internal error: Function '%1' not implemented.	
33:501	System error!	Attempt to get online as normal user under WinNT.
33:511	System error!	Mixed installation
		NT: insufficient access rights.
		NT: main user access rights already assigned!
	EPROM: Memory card is missing	
	The CP 1413 occupies the memory area 000D 0000 to 000D FFFF (complete D segment) or 000E 0000 to 000E FFFF (complete E segment). Normally MPI boards also use address areas in the D segment (from 000D C000) meaning they interfere with communication with the CP 1413.	Can occur if the new connection to the database is not set up; for example when closing and opening available users, S7db_close and open is executed. Sometimes these do not match the current connection.
		Bus monitor Amprolyzer has been installed.
30:527	The interface of the multi instance call can only be updated, if the declaration of the multi instance is valid in the tag declaration table.	If you click "Yes" in the dialog box, the system corrects the corresponding entry in the detailed variable view for you.
33:3100		When you have set the option "PG/PC is the only master on the bus" in the dialog by "Set PG/PC Interface" and you will make an online connection by "Display Accessible Notes" to the CP5412 A2, so you will receive this error number if not a partner is connected on this modul.
33:8242	The EPROM driver cannot support the set programming interface on this computer	PC + external prommer. Cause of error not yet known: probably a setup problem where the Registry entries are not checked.
		Same message with a PG 720/740/760
33:8256	The external prommer is not connected to the corresponding port or the power supply has been interrupted.	The external prommer is not connected to the configured LPT interface.
		The power supply of the external prommer is interrupted.
		LPT on PC faulty.
		LPT incorrectly set.
33:1639 9	System error by S7OTBL Server Client-Communication.	When this message appears, the S7DOS server (S7otblsx.exe or S7otbxsx.exe) doesn't want to start.
		This server is a relatively conventional Exe that receives data from the line as proxy for programs that want to go online. It therefore has nothing to do with databases or OLE/COM - yet.

		What is the purpose of the server. S7DOS recognizes four call types, 3 of which are asynchronous. So that the application can continue to run in the case of asynchronous calls and exported if necessary, there must be a proxy that constantly "listens" to the line and receives any data that arrives. If the PLC sends data directly, only then can you start to store the applications again from the swap-out file.
		The S7DOS-DLL attempts to start the server. If this does not work, then it tries again and again. It has a timeout of 30 seconds in which time the server should be up. If nothing happens during 30seconds -no refresh of the application - then this is a clear indication of an unsuccessful server start.
33:16418	The monitoring time for a STEP 7 frame was exceeded.	
33:16418	Timeout by wait of WM_ENDE_L7.	The monitoring time for a STEP 7 telegram has expired.
		Faulty parameterization of the access point on the MPI, L2 and H1 card or on the corresponding bus.
		There is already a communications load at the moment due to a configured S7 communication and therefore no ONLINE connection is possible.
		Cold or warm restart of communications partner.
33:16642	Online: Internal error - wrong configuration of telegram.	Occurs when TeleService is in operation. Error image: a TeleService connection is set up between a US-Robotics Sportster Voice 33.6 fax modem (local; analog, external) and an M1 GSM module (system side). With this constellation an online connection is set up in TeleService, the available users are displayed, but if you double-click on, for example, MPI=2 (direct), you get this error message. When you close this error message, the block container is displayed and the remote connection to the system is disconnected (295:20493).
		Error described above also occurs with a connection between US- Robotics Sportster Voice 33.6 fax modem (local; analog, external) and the Penril Data Link 19.2 modem (system side, analog, external).
33:16647	The online connection was interrupted.	
33:16654	Online: The connection was broken. Check the online connection and select the instance data block again.	HW connection interrupted.
		CPU has no power supply. Power supply failure.
		S7-300: Group error LED (SF) lights.
		Interface is already occupied by another PG connection.
		Power-save Management is active in the computer.

33:16656	The error occurs frequently if a station address is set which differs from the actual network configuration	The error occurs frequently if an MPI address is set, which deviates from the actual network configuration. The error can occur in particular with the following operations:
		A block has been copied from an S7 program that is directly under a project and inserted in the block container of an S7 program that is under a module.
		For setting up the online connection it is no longer the MPI address in the Properties of the S7 program that is valid, but the configured MPI address of the module.
		A block has been opened in the "Available Users" window and then an attempt was made to save the block in a project.
		For setting up the online connection it is no longer the MPI address from the "Available Users" window that is valid, but the MPI address configured in the project.
		In the "Available Users" window a user has been marked and the "Monitor and control variable" function called. Then an attempt was made to save the variable, which is only possible offline in a project.
		For setting up the online connection it is no longer the MPI address from the "Available Users" window that is valid, but the MPI address configured in the project.
		The block container of a CPU with several DP interfaces cannot be opened over the DP interface in the online view. This online view is however possible over the MPI/DP interface. A online connection can be built up with the function "Available station" via the DP-interface of the CPU.
33:16662	Online: No connection. Node does not accept the connection setup. Check the online connection and select the instance data block again.	The partner is neither a CPU nor a CP.
		The partner is not a module in the PLC.
		The partner is a different PG or an external device.
		The partner cannot set up any more connections.
		Connection setup via DP interface without the CPU retaining it's relevant SDBs.
33:16914	Online: Station not online.	Power supply of the S7 is switched off.
		Connecting cable is not plugged in or is defective.
		Configured bus parameters do not match the S7 system.
		Incorrect access point is selected.
		PG: hardware fault

33:16917	The function is not supported by the selected hardware.	Occurs if you set up an ONLINE connection via a CP5412 A2 with the option "PC is the only master" (in the "Set PG/PC interface" tool) via "Available Users" and there is no user connected to the CP 5412 A2
33:16944	Online: It is not found an active contact.	User not connected to the bus.
		On the PC/TS adapter the LED for correct power supply lights. This LED lights if only the 24V supply is connect and the 5V supply is missing. The PC/TS adapter however needs the 24V and the 5V power supply through the PLC.
		The TS adapter is set to the wrong transmission rate on the MPI side.
		The PC/TS adapter hardware is damaged.
		The wrong driver is preset under "Set PG/PC interface".
33:17072	Online: no hardware is found.	No driver selected or incorrect address area of the MPI interface
		With Windows NT, check that the PC has an MPI-ISA card and not a CP5611.
33:17074	ONLINE: driver is wrong configured or not valid parameter in the registry	MPI-ISA card: of an STEP7 version (also installation upgrade and corrections) -Win95 hardware recognition has been run -other drivers (also CP...) tried out -another card installed in the PG/PC -sometimes hardware is defective
33:17075	All stations on this subnet must have the same value for the transmission rate. The interrupt may not be assigned for other hardware components. The MPI addresses in the network must be unique, meaning each address may only be assigned once. The local Station address must also not be greater than the highest station address in the programmable controller.	With this message there is usually an interrupt that is double-occupied.
		Occurs with SCENIC PRO M6, Onboard Controller occupies many IRQs
		WIN 95 does not clearly recognize the interrupt occupied through the 3Com-Etherlink 3 (ISA) 3c0509b. This can lead to interrupt sources being double-occupied.
		The reason for this can be an actual fault in the hardware, or an interrupt conflict with another module.
33:17077	Online: The setting local address of participant is assigned	The TS adapter had the same MPI address (0) as another user in the MPI network.
33:17088	Online: The selected communication driver can not be loaded. File is not found.	In V2.1 the PG/PC interface hardware is set to "none"
		Possibly wrong or no driver selected with Windows NT.
33:17090	Logical device is not in the registry.	
		The LogDev is in the Registry under the following key: HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\SINEC\ LogName \S7ONLINE\LogDevice

		No, that would be too simple. As usual for such a communication, each layer is implemented as an application layer. There is therefore at least one driver for Layer 7 (s7otbldx,s7otblsx), one for Layer 4 (s7otransx) and one for Layer 2 (s7ospc2). The drivers for Layer 2 still run in the Ring 0 of Windows, the remaining drivers in the Ring 3. There is also a central program that manages the transition from Ring 3 to Ring 0 including the transferring data to and fro (s7onlinx).
		With the CP 5611: driver warped in the Registry.
		"PG/PC is the only master on the bus" is not activated (not checked) although the PG/PC is the only master.
33:17104	The version of the adapter connected does not support the bus parameters set or it is not connected to the adapter type assigned to it in the program "Setting the PG/PC Interface".	Probably the versions of the drivers and the PC Adapter or PC/MPI card are incompatible.
33:17106	Communication to the PC/TS adapter is faulty.	The PC/MPI cable is plugged into a module that provides no power supply.
		The connection between PC adapter and CPU is defective.
		Connection between PC adapter and PC is defective.
		The TS adapter is not yet set under "Set PG/PC interface".
33:17108	Exit the application which is currently using the required port (COM port or USB)	Another application already occupies the serial interface or faulty setting of COM-IF upon display under NT.
33:17109	The driver for the required port is currently being used by another application. Another interface parameter assignment configuration may also be in use.	The item Direct Connection has been selected in the SIMATIC Manager under Options --> Set PG/PC interface.
		After interruption of a TS connection the serial driver is still reserved by STEP 7.
33:17110	The interface TS adapter is set for a modem connection and there is no remote connection to a TS adapter.	After dialing into a remote system via TeleService (connection is set up), the TeleService window was minimized and PLC --> Display Available Users was selected and then this error message was displayed. The reason is probably a bad telephone line or a connection that is too slow.
33:33285	It is a copied object variant available.	
33:33540	Message from the module. Too few resources available.	Message from the PLC when reading block when, for example, two upload processes are triggered simultaneously by 2 PG.
33:33794	Illegal request of service	The CFB in the communications partner is in the wrong state.
		PRINT block: Paper out.
		The program instance in the communications partner is in the wrong state (cf. START, STOP, RESUME block).

		When implementing a CPU with DP interfacing, you might get the following situation: a user is not available on the DP bus, the CPU remains about 30 seconds in startup and the RUN LED blinks. Then the CPU goes into RUN (as long as no I/O access to the missing I/Os puts the CPU into STOP). This is obviously a TIMEOUT effect, because the CPU doesn't acknowledge the service in the relevant time.
33:3456 2	The service requested is not supported by the module.	The service requested is not supported by the module.
		The object addressed does not exist on the module.
33:4241 8		Message comes when executing the function RAM TO ROM: requested is The service not supported by the module.
		The addressed object does not exist on the module.
33:5331 5	D043: Data Error: The references order is not available.	When controlling by monitoring variables, it might happen that when too many control jobs are sent to the CPU, the CPU rejects them because its buffer is too full.
33:5334 3		
33:5334 6	Resources error (D062), job list is too full	The variable table (VAT) can no longer be opened although the table could be opened previously without any problems using the same program and with the same CPU. Probably the VAT concerned has not be closed correctly in the CPU.
33:5334 7	(D063) Resources error: the trigger event is occupied.	It is no longer possible to monitor a block. This behavior is triggered by irregular termination of the process in the operating system.
		Message from the module. This function cannot be executed at the moment, because the resource required is already occupied. There are also too many test functions activated at the same time and the resources available on the module are exhausted.
33:5377 2	Note the restrictions of the module being used.	Message from a module.
33:5379 0	The module to be assigned parameters is currently busy assigning parameters to subordinate devices. It was been by the modules: 416-2DP + CP443-5 Extended	
33:5379 7	The distributed I/O system data block is structured incorrectly.	A diagnostics address outside the address area has been specified. (For example, Win AC lite was installed.)
33:5379 8	A mismatched CPU type or an internal structural error has been detected during block analysis.	When evaluating the blocks, an incompatible CPU type or internal structure error was detected.
33:5382 4	D204: The rules of coordinations were broken.	When attempting to load the system data, there was still a variable table open online.
33:5990 6	The communication software (S7DOS) and the interface module (CP) have not the	The customer has a STEP 7 V5+SP1 + new PDM and a CP 443-5 Ext > V2.7. The new protocol RPC 7 that is used for data

	same protocol.	record routing between the CP and S7DOS. Unfortunately versions up to version V2.7 of the CP do not have this function.
		The user is implementing the right components STEP 7 V5 + SP2 + PDM, but has a CP that does not include the DSGW.
34:11	Compiler information: handle not valid.	When monitoring an empty network with the LAD/FBD/STL editor.
34:21	Compiler information: element not found.	The element could not be found in the referenced database.
34:38	Compiler information: Compiler error	
34:39		You have attempted to rename a variable, but the name is already being used.
34:152	Reference data could not be generated.	Error in the database.
34:168	The moduls in the comand structure are protected and can not be generated.	One or more direct or indirect reference module is know how protected and can not be generated
34:173	At least one symbol assignment has changed	If you copy a symbolically addressed block from the library of an S7 project into another project and after you open the block. In the original project you generate an STL source from the block.
34:178	Block call invalid because interface was changed in the meantime.	With Step7 V5.0 the instance DB of the DB type is entered in the symbol list for an FB call. This is no longer permitted with Step7 V5.0. In this case the "Update Call" function doesn't work.
34:4321	Error in column 1, distribute the program in little blocks.	
34:4355	F Ze %2!06d! Sp %3!03d!: No PLC type description found for called or addressed block %4.	The block called is not available.
34:4461	The block %4 that was called/used is protected or created by S7-GRAPH and cannot be generated.	You are trying to generate a KNOW_HOW_PROTECT block into an STL source.
34:4469	The declaration range of your formal parameter does not match the declaration range of the actual parameter, for example, no OUTPUT paramater from the actual end can be assigned to an INPUT parameter from the formal end.	It is not allowed, that an OUT variable of the calling function block is parameterized with the IN variable of the function block called.

34:4485		The parameter assignments can be specified in any order in the ASCII source. The comments, however, are stored separately from the actual online block in the order entered. Thus, when exchanging parameters in the source, the order of comments might then be wrong. When the block is opened, the comments are then assigned incorrectly. In the worst case comments might even be lost. The following comments in the network are also affected, even when the exchanged parameters are not even commented.
		Message appears when opening a VAT. The directory S7USS is missing in the SIEMENS\STEP7 directory.
88:47	Unable to reach the module %1 by means of the online interface that is set.	Programming device is connected to the MPI interface of the CPU.
256:24	Function open Environment is not implemetiert	Multiple start of the SIMATIC Manager
256:26	Internal OLE Error.	Not enough space on the hard disk, the database cannot be started.
		Wrong DLL in Windows/System and/or wrong Registry entry.
		With V3.1 when printing
		WIN 95: the OLE was not installed properly when installing WINDOWS.
		Message appears when opening a VAT. The S7USS directory is missing in the directory SIEMENS\STEP7.
256:30	SCL: Programming of S7 moduls. The object was deleted.	Error occurs when you export an SCL source, delete it in the SIMATIC Manager and then import it again and compile it.
256:31	This object was not found.	An object that fulfills the criteria specified is not available. The function triggered cannot be executed. The assignments in the OFFLINE database with regard to the HW configuration are no longer consistent.
256:32	Not valid objects.	A link has obviously been lost.
256:35	Internal system error is not repairable.	
256:49	The quoted entry is not in the registry by Windows or contains a not valid value.	Error occurred when downloading SDBs into a remote CPU via TeleService. The cause here was a bad telephone line connection (analog) (in the USA). After several attempts to set up an ONLINE connection, the SDBs could be transferred without this error message. An attempt was also made to achieve a higher transmission rate with the ONLINE connection.
256:53	Internal error: Not valid pointer.	In PLCSIM an attempt was made to load the last simulation in PLCSIM via the menu item Simulation -> Last Simulation. Since the

		old project data (e.g. also block interfaces) has perhaps changed in the meantime, some of the data of the simulation originally created no longer matches the current project.
256:56	There is an internal error in the system.	Project might be damaged.
	STEP 7 V5.x: There is an internal error in the system.	
256:57	There is an not repairable error in the system during the programming. The function can not be implemented.	ProjID.pro might possibly exist when dearchiving.
256:62	Not repairable internal system error in refer to the file "symlist.dbf".	Project might be damaged. Computer might have crashed.
256:63		
256:68	The object ... cannot be edited because it is currently opened by another application or another user.	This message appears when you start the "File > Save as..." function together with the "With reorganization (slowly)" option if there are still applications open (e.g. Symbol Table) in the project which is being reorganized.
256:85		SIMATIC Manager expects formatted names.
256:89	The version 513 of the objekt XR00001 is not compatible in refer to the installed software.	Legacy from V2 in XUTIL catalog
256:92	The data can only be read.	This might be data, for example, from a project or library. The cause might be that there are no write rights for the directory or drive in which the data are located.
		The data is being processed by a different application or by a different user.
	The data are read-only.	You have copied a data backup of a project from a CD onto your hard disk and now cannot open this project. The reason for this is that all the files of the project have the attribute "write-protected" set. There are various ways of removing the write-protection from files. One such is described below.
256:93	The rights of write is missing in the directory.	The directory in which the S7P file is located is write-protected.
		Not enough space on the disk < 100 bytes
		If STEP 7 V4.x is concerned, then it might be that a file has not been deleted, which still specifies that the project is still on a write-protected drive.
256:94	Problems with the offline folder.	This message may be due to the fact that the project has been dearchived on a computer and edited with an options package whose version is no longer compatible with the current STEP 7 version.

256:100	Authorization of STEP 7 is not found.	No Basic Authorization installed. Error occurs when attempting to start an installed options package. Probably STEP 7 is only authorized as Mini.
256:104	Authorization of optional package is not found.	No user rights (authorization) found for the options software on drive C:.
		WIN NT: insufficient access rights.
256:107		The object specified could not be copied. There might not be enough space left on the data medium. Possibly the target directory is momentarily blocked by another application.
		Possibly another STEP 7 application has opened the object to be copied and has thus blocked it. In this case, close all other STEP 7 applications and trigger the process again.
256:111	Access denied	Your have repeatedly loaded a program in the SPS and now it is not possible to load the program again, also not after a new start.
256:119	There is no authorization for STEP 7. To operate this optional package, you require the authorization for "STEP 7 V4.x".	No authorization installed or the path in which the STEP 7 project is located is too long.
257:5	One or more objects of a type cannot be represented. Unable to load the server 's7uvpomx.dll' for _S7UV_LIST_CT-objects. Make sure that the required optional packages are installed.	The process variable server cannot be started. Installation problem with STEP 7. There must not be any incompatible version of WinCC installed on the computer.
		Mixed installation, a V4.0 or V4.01 was installed before the installation STEP 7 V5.x.
		Paths in Autoexec.bat are not set.
	One or more objects of a type cannot be represented. Unable to load the server 's7hkdmax.dll' for _S7H_DP_ETCR_INTGR_414_2_ROCF_CT-objects. Make sure that the required optional packages are installed.	A file required by STEP 7 has not been found. Probably the project contains objects for which you need an options package that is not installed to process them.
	One or more objects of a type cannot be represented. Unable to load the server 'p7spumax.dll' for _SIPROM_CONTAINER_CT-objects. Make sure that the required optional packages are installed.	There is no authorization permitting you to integrate SIMATIC PDM in STEP 7. This authorization is an options authorization which has to be installed in addition to the basic authorization of SIMATIC PDM.
	One or more objects of a type cannot be represented. Unable to load the server 's7jthomx.dll' for _AUT_TH_AT-objects. Make sure that the required optional packages are installed.	The S7 project to be opened contains an object that has been created with the options package Technological Hierarchies. This package is not installed or has not been recognized properly after an over-installation of STEP 7.
	One or more objects of a type cannot be represented. Unable to load the server 's7hk31ax.dll' for	Toolbox of SINUMERIK 840D is missing or is badly installed.

	_S7H_PLC315_2DPD_Modul_CT-objects. Make sure that the required optional packages are installed.	
		The S7 project is opened with STEP 7 V5.x and the SINUMERIK 840D toolbox is installed with version under V5.
	One or more objects of a type cannot be represented. Unable to load the server S7HK31AX.DLL for _S7H_6ES7_315_1AF03_0AB0_CT - objects. Make sure that the required optional packages are installed.	Your project contains objects that have been created with the options package C7_626P, but this options package is now not installed.
		The message appears with each project you open, even if it doesn't contain a C7_626P object.
	One or more objects of a type cannot be represented.Unable to load the server 's7wb53ax dll' for _S7H_6GK7_443_1BX00_0Xe_CT-objects. Make sure that the required optional packages are installed.	The NCM package is not installed and the S7 project contains corresponding objects.
	One or more objects of a type cannot be represented.Unable to load the server S7HK31AX.DLL for _S7H_PLC314_Modul_CT_objects. Make sure that the required optional packages are installed.	Toolbox of SINUMERIK 810 is missing or badly installed.
	One or more objects of a type cannot be represented.Unable to load the server "s7hsl7ax.dll" for _S7H_DP_PA_LINK_1_ETER_CT-objects. Make sure that the required optional packages are installed.	The error is probably caused by the permanent upgrading of STEP 7. There are several older versions of STEP 7 one on top of the other.
	One or more objects of a type cannot be represented.Unable to load the server "p7ss357x.dll" for _P7_FM357_6ES7_4AH00_0AE0_CT-objects. Make sure that the required optional packages are installed.	
	One or more objects of a type cannot be represented. Unable to load the server S7HK31AX.DLL for _P7_PUSTCH_CPU 314_CT_objects. Make sure that the required optional packages are installed.	
	One or more objects of a type cannot be represented.Unable to load the server S7HK31AX.DLL for _P7_POSTECH_CPU 314_CT_objects. Make sure that the required optional packages are installed.	Either the tool MCU-Pit is not installed at all (product of Simodrive/Sinumerik) or the tool MCU-Pit V4.x has been installed in combination with STEP 7 V5.x. Warning: only the version V4.10 of MCU-Pit is released for STEP 7 V5.x.

257:7	Type of object is not declared	A file needed by STEP 7 has not been found. Possibly the project contains objects for which you need an options package that is not installed to process them.
257:8	One or more objects of a type (%1!X!) cannot be represented. Make sure that the required optional packages are installed.	There is an object in the project that has to be parameterized with an S7 options package. This options package is not installed.
	One or more objects of a type (%1!X!) cannot be represented. Make sure that the required optional packages are installed.	There is an object in the project that has to be parameterized with an S7 options package. This options package is not installed.
257:9	The project contains objects of the optional package 'xxxx' that cannot be processed because the optional package is not installed or an outdated version is installed.	The project probably contains objects for which you need an options package that is not installed to process them, or a newer version of the options package.
257:24	creat environment is not implemented.	SIMATIC Manager has been started double or computer has crashed.
257:26	Not repairable internal system error. The link could not be inserted.	Message appears when you trigger the function "Display Reference Data". Links are distorted in the project.
		Message also appears when forwarding a project in conjunction with the symbol table.
257:79	The project or parts of it are currently being edited by another application.	This message appears when you start the "File > Reorganize..." function if there is still an application open (e.g. Symbol Table) in the project which is being reorganized.
257:90	The project '%1' contains errors in the internal data structure.	Project data damage by a crash/power failure on the computer. With the "Details" button you can receive more information on the error, if available.
257:94	Open of project: <Date and Time> S7fupos: Internal error during save of data of server S7ombstx.dll.	Inconsistent project. Application FM 453 has registered errors when saving the machine DBs. A possible cause is the shortage of storage space on the computer.
258:1	Unable to open project data %1.	IDB defined as GDB in the symbol table.
258:17	The directory C:\Siemens\STEP7\S7proj\temp exist already.	The temporary SDBDATA directory created must have been destroyed by the crash.
258:20		
258:15		Some files or data blocks are write-protected and, therefore, the data can only be accessed on a read-only basis. This may be due to the fact that write access does not exist for the directory or drive where this data is located, or the block may have been saved as write-protected for reference purposes.

		There are still blocks (FCs, FBs, DBs and/or OBs) open and, as a result, the associated projects and directories cannot be deleted.
		If the last remedy point does not help either and the project which you wish to delete is still present under the menu commands "File > Open..." and "File > Delete...", some of the files in the SIMATIC software may be damaged. In most cases, the repair function on the STEP 7 CD can be used to restore the software status on your PC.
275:502 0	There is an internal error in the file S7nlvblx.dll.	Save your entries and close the application. Switch your computer off/on and restart the application. If the error reoccurs, then please get in touch with your system administrator.
276:27		No valid connection path between local and remote partners (e.g. because of missing network connection or exhausted resources).
276:42	Loopback - Connecting are not allowed!	Loopback connections are not permitted. You can specify an identical address as local and remote user address, but then SDB generation is rejected with an error message.
276:502 0	An internal error occurred. (Call: AUTReloadObject)	
288:48	Unable to reach the module (z.B.) 413-2DP by means of the subnet assigned to the programming device/PC.	In a new project, for example, a hardware upload was made from the CPU to the PG. In the attempt to immediately download this upload, this error message appears. Problem: under Properties in Options -> Set PG/PC interface, 1 is preset as local user address on the MPI bus. But this does not match the address configured in the hardware upload and hence the error message.
289:2	The data base is write-protected or there are not enough memory.	XREF with V2.1.0 + standard control
289:25	Two modules could not be translated back.	XREF with V2.1.0 + standard control
289:35	The selected button can not be executed for the selected menu.	You have selected a block folder and triggered printing.
291:416		The file is occupied by another application for writing.
291:416	The symbol table is being by another process.	In the window that opens, the symbols are greyed out since editing is not possible. The Symbol Editor does not show any changes (it is not possible to save the table) and it is not possible to access the symbols.
291:416	The symbol table is occupied by another process	In this case you can no longer edit, delete or import the symbol table. The symbol table is not occupied by any other application for writing. You have stored the project locally on the hard disk. The same error is also displayed with the "Save with Reorganization" function of the project.
291:560	Internal error: Incorrect object ID.	The symtabs function has failed - ambiguous entry in the symbol list for FC names.

		When calling an FB, a DB was assigned that is already assigned to another FB in the symbol table.
291:139 2	No symbol table found. No symbol could be created.	
292:19	Error by opening of a file.	The file either has no valid B+C287trieve format or it is damaged.
293:1	The reference data could no be generated.	The data medium is write-protected.
		Under NT: possibly insufficient access rights.
293:2	The reference data could no be generated for the SFC14 or the FC22.	
294:6	Unable to copy the block %2.	The OBxy is not supported by the CPU.
		When programming an EPROM via an external prommer: the prommer cable is plugged in incorrectly.
294:23	Unable to copy the block %2. Do you want to continue the copy procedure?	A block of this type already exists and it may not be overwritten.
		The automation system is in the RUN operating mode.
		A block of this type and number is not permitted on the automation system.
		The block is using operands beyond the area of the CPU (e.g. EW 128 with S7-300 CPU).
		The block is using statements that are not realized in the CPU (e.g. ENT with S7-300 CPU).
		The block has to much local data (e.g. more than 256 bytes with S7-300 CPU).
		Referenced block SFC, SFB is not permitted in this module.
294:27	The block %2 is being processed by another application or another user at the moment.\nDo you want to create the reference data for the saved version of the block?	Multi-users: the block specified is being processed at the moment by another application or by another user. You can create a copy of the last version saved or cancel the function.
		Single Users: the block is opened ONLINE as OFFLINE.
295:172 40		
295:172 48	The order was interrupted to the TS-Adapter because of transmission error.	The initialization string is not correct or in this form does not work together with the modem connected to the system.
		On the system there is not a TS adapter but a PC adapter on the modem. The TS adapter has order number 6ES7972-0CA3x-0XA0
		Possibly the terminator on the PROFIBUS cable on the system side (TeleService) is switched on.

		Possible the modem is sending too much information to the TS adapter so that it doesn't get the CONNECT. (Occurs with V5 adapters)
		If you attempt to get connected via long telephone routes (abroad), then it might happen that some feedback messages have too long a runtime.
295:20 484	There is not installed a modem which is compatible with teleservice.	
295:20 486	The TAPI-Order was closed with the error "Order failure".	Basically communication between (local) PC and modem (via the serial connection) is not correct or cannot even be started.
		Under Windows (Start --> Settings --> Control Panel --> Modems) an incorrect (or no) modem driver has been installed, which does not work in conjunction with the modem connected.
		The wrong modem driver was selected under TeleService from the selection list of modems that are already installed under Windows.
		The modem is not connected to a serial interface (or to the wrong one).
		The modem is switched off.
295:32 768	TAPI-subsystem dues supply not expected error code <TAPI-Error-Code>	
		ALLOCATED 0x80000001
		INVALDEVICECLASS 0x80000023
296:53 09	The variables of the instance data block which contain system attributes for messages are not aggree with the variables in the function block.	Clearance: check the function block and save it again if necessary.
300:13	Data basis could not be started.	Occurs as a consequence of message 256:94 and details 256:24/256:63 when executing the function "Reorganize" or "Save as... with consistency check". This occurs when a link is lost in the SYBASE database.
		Occurs after dearchiving a project in STEP 7 V5.0. It has been created in the version V4.02.x.
314:8	At least one of the selected variables cannot be inserted at this point.	If, while you are processing a project in STEP 7 V5.3 which has been created with STEP 7 V5.1 and the error message described above appears in the LAD/STL/FBD editor when you insert a declaration line within an FB or FC, please check the names of the formal parameters in the declaration line for the block in question. This could be due to the fact that the name "Ret_Val" has been assigned to a formal parameter (IN, OUT, IN_OUT, STAT or TEMP variable).

724:21	OM TD/OP: Data basis conflict of version file: C:\[step7-pfad]\[Projektname]\S7TDOP.DBF has the version 50000. This program expected the version 10000.	The versions of STEP 7 and ProTool are incompatible with those of the project. STEP 7 is V4.02.x and ProTool V5.
1230:1 001	Opening of the station failed.	
1230:2 001	Unable to insert the object.	You have tried to slot an object that is not available (e.g. the header for an object group or module class).
1230:2 009	Changes cannot be made. Save your changes in other applications	The error message indicates that access to the object is not possible at the moment and that the object could not be blocked. This error message can have various causes: 1. The object is currently being processed by another application or user and has therefore been blocked. 2. There is insufficient memory available.
1230:2 015	Creating the configuration data failed.	Hard disk is full.
		No folder "...SIEMENS\STEP7\S7TMP" available or the folder is write-protected.
		S7 program or M7 program container is missing.
		Defective project?
		The compilation (= creation of the configuration data) has been interrupted.
	Error attempting to generate the configuration data. Message refers to the V2-projects by Save as with Reorganization.	Subnet addresses of different networks are identical.
1230:2 016	No changes can be made in this station.	No more modules can be inserted in the HW Config.
1230:2 018	Another application made changes to your station. Close this station and re-open it.	Another S7 application has changed, for example, via network, the station currently being processed. The changed objects (subrack, modules, slaves, etc.) can no longer be displayed. The station has been blocked for further processing.
1230:3 000	The system data could not be recreated because the configuration is inconsistent.	
2508:1 099	Error by programming of error diagnostic data for <name of variable>.	Group message, behind it detailed messages are listed that occurred during the last function executed.
2517:5	The start of an application has occurred an error.	Message appears when opening a HIGRAPH source. With the following software constellation: STEP 7 V5.0 + SP 3 S7-HIGRAPH V4.01.1 S7 project created with S7-Higraph V5.x
2665:4 156	Because of the changes made to the configuration, the configuration file for this SIMATIC PC station has to be created again	

	with NetPro. To do this, use the NetPro menu command "Network > Save and Compile ..."	
2760:11	communication to higraph cannnot be accessed	
2775:77	Address is not correct.	You have selected a user address that is already occupied.
2775:785	The bit rate of 187,5 kbit/s for the modul MBK-P is not supported in station SIMATIC 400(1).	In STEP7 V5+SP3 and higher the baud rate must not be written with a comma, e.g. 187,5_supp =1
2775:1829	The data management of "Set PG/PC interface" can not save the requested value.	An attempt was made to save a parameter value in the database of the program "Set PG/PC interface". This write procedure could not be concluded successfully.
2775:1860	Caution: there are other active nodes (sender parameter assignment master) in station %1 which also have to be reloaded.	
3020:13	File name entered is not valid, or Project is invalid, if you attempt to open a project.	This message appears when you stored your project on a network drive which is connected to the following path via the Internet Explorer or a logon script: \\<Servername>.<DNS-Suffix>\<Share> (Example: \\computer.siemens.de\share) or \\<IP address>\<share> (Example: \\192.168.1.1\share). However, this path cannot be found since the network connection is stored only symbolically in the registry.
3020:25	Projects which are opened can not be archived. Make sure that in non STEP 7 application objects of the projects ... are opened in the directory C:...	The reason for this error message are blocked files, which are also blocked though all the windows of editors and projects are closed. This is possible when the program is activ and the last access was to the archived project. Souch programs are not always to see as user window.

3020:47		This processing status is triggered by PKUnZip in different situations, in particular always when an existing project is overwritten during dearchiving and in the dearchiving options the entry "Overwrite objects with identical names" is not set to "All Files".
		The archive file has not been created with PKZip. In particular archives that have been created with WinZip or "PKZip for Windows" and contain "long file names" can lead to this error.
		The archive file has been created with a version of PKZip that is newer than the version of PKUnZip that is installed on the computer.
		The archive file is damaged. Checksum errors (so-called CRC errors) have been detected for one or more files. Therefore the files could not be dearchived correctly. The cause of such checksum errors can in particular be: damaged floppy disks, transfer errors when copying, in particular with network drives, file transfer or e-mail, or at the time of archiving the project or parts of it were still open.
		The directories in the archive file have too great a nesting depth to be able to be unpacked under the specified target directory. PKUnZip cannot create any directories whose absolute path name is longer than 66 characters (from the drive designation to the '\' at the end of the path).
		The project already exists and individual files could not be overwritten when dearchiving, because they are write-protected.
		The project already exists and individual files could not be overwritten when dearchiving, because the project or objects from the project are already open.
3020:49	Unable to retrieve the project. The archive file might be defective.	The archive file has not been created with PKZip. In particular archives that have been created with WinZip or "PKZip for Windows" and include "long file names" can lead to this error.
		The archive file has been created with a version of PKZip that is newer than the version of PKUnZip installed on this computer.
		The archive file is damaged.
		The data medium or the directory in which the dearchived project is to be stored is write-protected, or you have no write authorization for the network drive.

3022:4 121	The FB is not available The instance declaration is missing	If during compilation in S7-PDIAG the mentioned error message, then it is probably the case that the S7-PDIAG blocks have also been copied from another STEP 7 during the copying procedure. This is not permissible, because the associated SFBs are then no longer up to date.
3022:4 121	Invalid data type...",	The error message is probably caused by an associated value with an invalid data type. Please also refer to the information in the Online Help of S7-DIAG concerning the data types that are valid for associated values.
3280:5 03	The registry database is not set up correctly for the SIMATIC Manager. Install STEP 7 again.	Installation is faulty.
3280:7 08	No objects can be copied or moved from projects/libraries of the current version to projects/libraries of version 2.x.	In STEP 7 there is no downward compatibility, the data management in V2 and >V3 is not the same, so blocks created with STEP 7 V3 or higher cannot be copied or moved into V2.x projects .
3280:7 09	The project or the library '%1' is on a write-protected medium.	The project is either located on a write-protected floppy disk or you do not have write authorization for the drive.
		The project catalog was released in the Windows Explorer with read/write access.
3280:7 12	File name entered is not valid. Project is invalid. (If you attempt to open a project)	
3280:9 99	The internet explorer is not installed in your PC.	STEP 7 V5 has been installed without the Internet Explorer and you are trying to start the ONLINE Help.
		The ONLINE Help was started and the Internet Explorer was installed after the installation of STEP 7.
3531:1	The diagnostic block could not be created.	Standard blocks in the program originate from the PLC and not from the library. If the project has been created per upload by the PLC with an earlier version of STEP 7 or standard blocks have been copied from the PLC into the offline program, then the interface of the SFBs and SFCs in the project only has a Default Symbolic (parameter name = "IN0", "IN1" etc. instead of symbolic names as described in the Help). In this case the same error is reported.
3534:6 3	While reporting a system error: an internal error has ocurred	To be able to generate the reference data without this message, proceed as follows: 1. Delete Diagnostics-FB49 and Diagnostics-DB49 from your STEP 7 Project resp. Block container. 2. Newly generate the reference data by selecting the block container and the menu task Extras > Reference data > generate. This message does no longer appear when generating reference data.
3534:9 4	Inconsistent of hardware configuration	The hardware configuration is either inconsistent or it was made changing which are not still save and compiled.
4050:1		The resources file for this language is not in the directory in which you have installed this program. This is why the menus, dialog boxes and the online Help are in English in S7-PLCSIM.
4050:2		Closing the subwindow "CPU" simultaneously closes the

		simulation session. However, you can close a CPU and start a new simulation session with another CPU.
4050:3		You can only change the MPI address if there are no applications linked with the simulation.
4050:4		You can only ever have one simulated automation system activated. If you open a second simulated automation system, the one opened first is closed. If you want to continue the function, press "Yes". Press "No" to cancel.
4050:5		You cannot exit S7-PLCSIM if there is still a connection set up to STEP 7 (e.g. SIMATIC Manager, variable table, program editor). STEP 7 indicates in a message that the connection to the CPU has been interrupted. You can then continue working offline.
4050:6		When doing an overall reset (MRES), all the blocks are deleted and the memory of the simulated automation system is reset.
4050:7		You have attempted to open a simulation that has just recently been used, but which is no longer available or cannot be found. S7- PLCSIM deletes the file name from the menu.
4050:100		The address you have specified has a format that STEP 7 does not support.
4050:101		The value you have entered is not in the valid range for the selected format. For example, you cannot enter "FF" (hexadecimal value) in a field in which binary values are supposed to be entered.
4050:102		The value you have entered is not a decimal number.
4050:200		Since there is a danger of changing an actual online program by mistake, you cannot communicate with "real" CPUs whilst the simulation is activated.
4299:158	No connection to FM355/455.	Old firmware on the FM.
4431:2011	The Authorisation for SIMATIC PDM is not installed. SIMATIC PDM could not be started.	Authorization permitting you to integrate SIMATIC PDM in STEP 7 is missing. This authorization is an option authorization that has to be

		installed in addition to the basic authorization of SIMATIC PDM.
4502:29 8	System error 33	This meesage appears during the upload of the hardware configuration. It is possible that the load of the SDBs was not complete or incorrect.
4502:62 5	An error occured while creating or interpreting the system SDB's....". Error while creating systemdata....".	Regarding mentioned messages occur while saving and translating in HW config if the following applies: 1. You projected and assigned a CP to a subnet in your Hardware configuration. Files with the extension ."err" are missing in the folder "<Siemens-index>\Step7\S7WBX\RUL". 2. Files with the extension ."err" will be deleted by some optimization programs - like e.g. TuneUp.
4561:14 2	The Authorisation for Standard PID Control is not installed.	..
D033	Error of protokol	Job jam on the CPU.
D042	Error during notice by module status of CPU400.	A PQW is being monitored. You cannot read back PQWs, the CPU then reports this error.
D043	Error of data: The reference order is not available.	When controlling using Variable Monitoring it might happen that if too many control jobs are sent to the CPU, they are rejected by the CPU because its buffer is full.
D062	Resources error, job list is too full.	
D063		
D064		CPU 300 does not delete trigger events, the events then build up.
D0AA	D0AA: The time limit is exceeded in process operating.	Communications load is too high.
		Message appears with the function "Control via VAT" to an address from the memory areas I, O, M, Timer, Counter, DB, which is not available in the broadest sense.
D204	D204: An error in the rules of coordinating has occured.	When attempting to load system data, in the background a variable table was still open online.
D20C	The OB can not be copied because the belong layer is not available.	Message comes from a CPU. An attempt is being made to transfer an OB onto this CPU, which is not permissible.
D240	An error in the rules of coordinating has occured.	Occurs if a block that is to be overwritten online is still being accessed. In V4.01 with blocks in the call environment.
	An error in the rules of coordinating	Occurs when an object is accessed online (also

	has occured.	VAT).
	Error during opening of a empty document. Error during opening of a symbolic table.	An incompatible OPC Manager is installed. In the system path of Windows vcf1.lic is responsible for this.
	Setup does not start with error message: in function "Sdbitmap":unable to crate dialog make sure the _isres.dll is in _setup.lib	The _isres.dll is needed for switching languages.
D406	The requested informations are not available.	Number of possible connections has been exceeded.
EA 02	The protocols of communication software (S7DOS) and the interface module (CP) are not compatible.	The customer has a STEP 7 V5+SP1 + recent PDM and a CP 443-5 Ext > V2.7. The new RPC 7 protocol is used for data record routing between the CP and S7DOS. Unfortunately the CP up to version V2.7 does not support this function.
None	Error message by diagnostic: "Profibus/MPI Network diagnostic". Error: requested parameter could not be read from the register data basis!	The user is using the correct components STEP 7 V5 + SP2 + PDM, but has a CP that does not have the DSGW. Firmware/driver error
S7OTBLSTX	Incompatible versionen by S7OTBLDX.DLL and S7OTBLSX.DLL.	S7 applications are still running at Setup of STEP 7.
		STEP 7 has been installed on a computer on which an S7 options package or version of ProTool /WIN CC is already installed which is not compatible with the version of STEP 7.

منابع و مراجع

- **Siemens Manual Collection** SIEMENS
- **Sitrain Programming Intermediate Skills** SIEMENS
- **Sitrain Programming 2** SIEMENS
- **Standard Software for S7-300 and S7-400 PID Control** SIEMENS
- **PID Control** Karl Johan Astrom
- **Measurement and Instrumentation Principles** To Jane, Nicola and Julia
- **Lessons in Industrial Instrumentation** Tony R. Kuphaldt

- راهنمای جامع Step7 جلد اول و دوم — محمدرضا ماهر
- شبکه صنعتی Profibus — محمدرضا ماهر – علی کریم الدینی
- پیاده‌سازی منطق فازی با PLC — محمدرضا ماهر – غلامرضا آغاجری
- تکنولوژی فیلدباس — محمدرضا ماهر – احمد حیدریان
- پروژه‌های کاربردی با PLC S7 — احمد فرجی
- سیستم‌های کنترل خطی — علی خاکی صدیق
- کنترل صنعتی — محمود تارخ

کاملترین مرجع کاربردی کارور

PLC درجهٔ۲

مهندس احمد فرجی

نوبت چاپ: اول/۱۳۹۱، سوم/۱۳۹۲
۴۳۲ صفحه/ وزیری

مباحث اصلی شامل:

- آشنایی با کامپیوتر و مدارات منطقی
- موتورهای الکتریکی سه‌فاز و تک‌فاز و مدار فرمان
- آشنایی با ساختمان PLC و سنسورها و عملگرها
- پیکربندی سخت‌افزار S7 300 و S7 400
- برنامه‌نویسی در محیط SIMATIC Manager
- نمونه پروژه‌های کاربردی

مجموعه پروژه‌های کاربردی با

PLC SIEMENS

مهندس احمد فرجی

نوبت چاپ: اول/۱۳۸۹، ششم/ ۱۳۹۲
۲۵۶ صفحه/ وزیری

مباحث اصلی شامل:

- پروژه‌های سطح مقدماتی شامل سیستم پمپاژ آب، راه‌اندازی الکتروموتور سه‌فاز به‌صورت چپگرد/ راستگرد- ستاره/ مثلث- دستگاه کاتر و ...
- پروژه‌های سطح متوسط شامل کنترل نوار نقاله، محاسبهٔ ظرفیت انبار، کنترل دمای مخزن، چراغ راهنمایی و ...
- پروژه‌های سطح پیشرفته شامل کارخانهٔ نوشابه‌سازی، کارخانهٔ تولید شکر، کارخانهٔ تولید آهک
- پروژهٔ جامع شامل تمام خطوط موجود در کارخانه تولید قطعات فلزی
- ترفندهای برنامه‌نویسی شامل ساخت کانتر، ساخت تایمر

کاملترین مرجع کاربردی

PLC S7 SIEMENS

(سطح مقدماتی)

مهندس محمدرضا ماهر
مهندس احمد فرجی

نوبت چاپ: اول/۱۳۹۰، پنجم/۱۳۹۲
۷۳۶ صفحه/ وزیری

مباحث اصلی شامل:

- تاریخچهٔ سیستم‌های کنترل
- جایگاه PLC در سیستم‌های کنترل
- ساختار و اجزای PLC
- نکات طراحی و انتخاب PLC
- نکات نصب PLC
- آشنایی با سنسورها و عملگرها
- آشنایی با کنترلرهای زیمنس و انواع PLCهای S7
- نصب و استفاده از نرم‌افزار Step7
- پیکربندی سخت‌افزار S7 300 توسط HW Config
- مفاهیم برنامه‌نویسی
- برنامه‌نویسی با دستورات LAD و FBD
- نحوهٔ کار با Modify/Force
- ارتباط Online با PLC و ...

مجموعه پروژه‌های پیشرفتهٔ اتوماسیون

SIEMENS

مهندس مبین محسن‌زاده

نوبت چاپ: اول/۱۳۹۰، دوم/ ۱۳۹۰
۳۷۶ صفحه/ وزیری

مباحث اصلی شامل:

- پروژه‌های کاربردی با نرم‌افزارهای مانیتورینگ WinCC Flexible 2008 , ProTool V6.0
- پروژه‌های ترکیبی از سیستم‌های PLC و SCADA
- به‌کارگیری منطق فازی در PLC به کمک نرم‌افزار Fuzzycontrol++
- پروژه‌های کاربردی استفاده از شبکه‌های صنعتی Ethernet و MPI ،Profibus-DP , FDL , FMS
- پروژه‌های کاربردی به کمک نرم‌افزار جامع DCS زیمنس PCS7 V7.1
- پروژه‌ای جامع از ارتباط بین PLC و Inverter-Micro Master با قابلیت مانیتورینگ روی HMI
- پروژه‌ای کاربردی از ارتباط Remote I/Oهای ET200M با PLC
- بررسی کاربردی تجهیزات هیدرولیک و ...

کاملترین مرجع کاربردی شبکهٔ صنعتی

Profibus

(جلد اول)

مهندس سعید اسفندیارپور
با ویرایش و نظارت **مهندس محمدرضا ماهر**

نوبت چاپ: اول/۱۳۹۱، دوم/۱۳۹۲
۴۳۲ صفحه/ وزیری

مباحث اصلی شامل:

- جایگاه پروفی‌باس در اتوماسیون صنعتی
- انواع پروتکل‌های پروفی‌باس
- ویژگی‌های پروفی‌باس
- تجهیزات سخت‌افزاری پروفی‌باس
- پیکربندی Master-Slave
- برنامه‌نویسی Master-Slave
- عیب‌یابی

کاملترین مرجع کاربردی شبکهٔ صنعتی

Profibus

(جلد دوم)

مهندس سعید اسفندیارپور
با ویرایش و نظارت **مهندس محمدرضا ماهر**

نوبت چاپ: اول/۱۳۹۱، دوم/۱۳۹۲
۴۵۶ صفحه/ وزیری

مباحث اصلی شامل:

- ارتباط درایو با PLC از طریق پروفی‌باس
- Master-Slave با کارت شبکهٔ پروفی‌باس
- تبادل دیتا بین PLCها از طریق پروفی‌باس
- اتوماسیون PC-Based با پروفی‌باس
- ارتباط S7-1200 با پروفی‌باس
- مانیتورینگ از طریق پروفی‌باس
- آشنایی با Profibus-FMS
- آشنایی با Profibus-PA

کاملترین مرجع کاربردی

WinCC v7

مهندس احمد فرجی

نوبت چاپ: اول/۱۳۹۱، چهارم/ ۱۳۹۲
۷۳۶ صفحه/ وزیری

مباحث اصلی شامل:

- پیکربندی سیستم مانیتورینگ توسط نرم‌افزار WinCC
- تعریف ارتباطات و تگ‌ها
- طراحی گرافیک پروسه
- برنامه‌نویسی به زبان‌های C و VBS
- ارتباط مجتمع WinCC و SIMATIC Manager
- آرشیوگیری و نمایش Trend و Table
- نمایش پیام‌های آلارم
- گزارش‌گیری
- پیکربندی سیستم Client/Server و ریداندانت
- بیش از ۱۰۰ تمرین و مثال کاربردی

مجموعه پروژه‌های پیشرفتهٔ

PLC و WinCC

مهندس احمد فرجی

نوبت چاپ: اول/ ۱۳۹۲
۲۷۲ صفحه/ وزیری

مباحث اصلی شامل:

- پروژه‌های کنترل موتور
- پروژه‌های ترکیبی
- پروژه‌های PID
- پروژه‌های SCL
- پروژه‌های GRAPH
- تمرینات تخصصی WinCC

کاملترین مرجع کاربردی

PCS7

(جلد اول)

مهندس محمدرضا ماهر

نوبت چاپ: اول/۱۳۸۹، دوم/۱۳۹۰
۵۴۴ صفحه/ وزیری

مباحث اصلی شامل:

- جایگاه PCS7 در سیستم‌های کنترل
- اجزای سیستم PCS7
- نصب سخت‌افزار
- نصب نرم‌افزار
- ابزارهای نرم‌افزار
- پیکربندی سخت‌افزار و شبکه
- نحوهٔ کار و برنامه‌نویسی در محیط CFC
- برنامه‌نویسی با بلاک‌های کتابخانهٔ CFC و کتابخانهٔ PCS7
- کنترل ترتیبی با SFC
- آشنایی با محیط OS و امکانات آن
- گرافیک و آلارم و آرشیو در WinCC
- مثال‌های کاربردی و ...

کاملترین مرجع کاربردی

PCS7

(جلد دوم)

مهندس محمدرضا ماهر

نوبت چاپ: اول/ ۱۳۹۲
۶۹۶ صفحه/ وزیری

مباحث اصلی شامل:

- شناخت بلوک‌های پیشرفتهٔ کتابخانهٔ PCS7
- پیاده‌سازی استراتژی‌های مختلف PID
- کار با Model و Process Tag Type
- فانکشن‌نویسی با SCL
- طراحی و تغییر Faceplate
- پیاده‌سازی طرح Client Server
- پیاده‌سازی Profibus PA
- تبادل دیتا بین دو یا چند AS

کاملترین مرجع کاربردی شبکهٔ صنعتی

Modbus

مهندس محمدرضا ماهر

مهندس حمیدرضا رفیعیان

نوبت چاپ: اول/۱۳۹۰، سوم ۱۳۹۲

۶۷۲ صفحه/ وزیری

مباحث اصلی شامل:

- جایگاه شبکهٔ مدباس در اتوماسیون صنعتی
- ارتباطات فیزیکی و اجزای سخت‌افزاری شبکهٔ مدباس
- پروتکل‌های مختلف مدباس
- پیکربندی و برنامه‌نویسی مدباس در PLC زیمنس
- نحوهٔ عیب‌یابی و تست مدباس
- کاربرد مدباس در DCS یوکوگاوا
- کاربرد مدباس در PLCهای HIMA و ABB
- کاربرد مدباس در سیستم کنترل لرزش Bentley Nevada

کاملترین مرجع کاربردی

PID Control در صنعت

مهندس احمد فرجی

نوبت چاپ: اول/۱۳۹۳

۳۶۰ صفحه/ وزیری

مباحث اصلی شامل:

- آشنایی با انواع سیستم‌های صنعتی
- آشنایی با نقشه‌ها و علائم ابزاردقیق (P&ID و PFD)
- انواع استراتژی‌های کنترل لوپ (حلقه بسته)
- متدها (روش‌های) کترل لوپ (P,I,D)
- شناخت رفتار داینامیک سیستم‌های صنعتی
- روش‌های تنظیم لوپ (Tuning)
- آشنایی با انواع سخت‌افزارهای کنترل PID
- پیاده‌سازی کنترل PID توسط نرم‌افزار STEP7 و WinCC

کاملترین مرجع کاربردی تحلیل سیستم‌های قدرت با

ETAP

مهندس سعید احمدیان

نوبت چاپ: اول /۱۳۹۲

۶۶۴ صفحه/ وزیری

مباحث اصلی شامل:

- محاسبات سایزینگ کابل، ترانسفورماتور و مدارشکن های فشار قوی و فشار ضعیف
- محاسبات Arc Flash و تعیین فواصل مجاز
- محاسبات هماهنگی تجهیزات حفاظتی (رله‌ها و فیوزها)
- محاسبات تحلیل هارمونیکی، طراحی فیلتر و اسکن فرکانسی
- محاسبات طراحی زمین
- پیاده‌سازی پروژه‌های عملی و کاربردی در پایان هر فصل و ارائهٔ پروژهٔ پایانی

کتاب درسی

الکترومغناطیس مهندسی

ترجمهٔ دکتر محمود بهار

نوبت چاپ: اول/۱۳۹۲

۶۶۴ صفحه/ وزیری

مباحث اصلی شامل:

- آنالیز برداری
- قانون کولن و شدت میدان الکتریکی
- چگالی شار الکتریکی، قانون گاؤس و دیورژانس
- انرژی و پتانسیل
- رساناها و دی‌الکتریک‌ها
- ظرفیت الکتریکی
- میدان مغناطیسی ایستا
- نیروهای مغناطیسی، مواد، و القاوری
- میدان‌های متغیر با زمان
- خطوط انتقال
- موج تخت یکنواخت
- بازتاب و پاشندگی موج تخت
- موج‌های هدایت شده
- تابش الکترومغناطیسی و آنتن‌ها

پیاده‌سازی منطق فازی در

PLC

مهندس محمدرضا ماهر
مهندس غلامرضا آقاجری

نوبت چاپ: اول/۱۳۸۸، سوم/۱۳۹۲
۳۱۶ صفحه / وزیری

مباحث اصلی شامل:

- تاریخچهٔ منطق فازی
- اصول و مفاهیم منطق فازی
- نصب و استفاده از نرم‌افزار Fuzzy Control
- ارتباط Fuzzy Control با بلاک‌های S7
- ارتباط Fuzzy Control با چارت‌های CFC
- ارتباط Fuzzy Control با محیط WinCC
- مثال‌های کاربردی و ...

مباحثی ویژه در مهندسی برق و کامپیوتر با

MATLAB

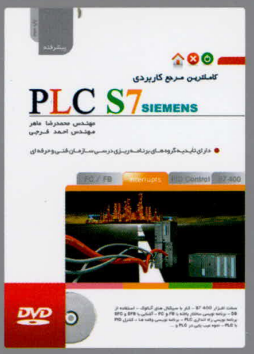
مهندس علی‌اکبر علمداری
مهندس عبداله دوستی عارف
مهندس رابعه کریمی مهابادی
مهندس زهرا رجبی

نوبت چاپ: اول/۱۳۹۰، دوم/۱۳۹۲
۶۲۴ صفحه/ وزیری

مباحث اصلی شامل:

- شبکه‌های عصبی
- پردازش تصویر
- منطق فازی
- الگوریتم ژنتیک
- مخابرات دیجیتال

NEGARANDE
ناشر کتاب‌های دانشگاهی D A N E S H



- The Most Complete Practical Reference of

PLC S7 SIEMENS

- By Mohammad reza Maher
  Ahmad Faraji

نگارنده دانش

انتشارات نگارنده دانش
تلفن: ۶۶۹۶۲۳۰۵ - ۶۶۹۶۲۰۵۳
تهران - صندوق پستی: ۹۱۶-۱۳۱۴۵
www.negarandedanesh.com
شابک: ۹۷۸-۶۰۰-۶۱۹۰-۰۲-۰