Kinco®

PLC

راهنمای نصب و راه‌اندازی

کنترلرهای منطقی برنامه‌پذیر سری K5 کینکو

ویرایش ۰.۱ - ۱۳۹۳/۰۴

فهرست مطالب

مقدمه :

در این کتاب قصد ما آشنایی کاربران با KINCO – K3 یا همان PLC(Programmable Logic Controller) کارخانه kincoمیباشد. KINCO-K3 از نظر قانون د سته بندی های PLC در رده micro PLCها قرار دارد .بنابراین میتوان از آنها در کنترل ماشین آلات صنعتی ، پروسه هایی در مقیاس کوچک و نیز در صنایعی مانند :صنایع بسته بندی ،مواد غذایی ،نساجی ، صنایع چاپ ،سیستم های تهویه و.... میتوان استفاده نمود .

فصل اول :

۱.۱:آشنایی با مفاهیم Kinco-K3 :

در این قسمت هدف آشنایی با یک سری از مفاهیمی است که کاربر را در درک بهتر مطالب یاری مینماید .

Micro PLC:

بر اساس قوانین دسته بندی عمومی ، micro plcها عموما به plcهایی اشاره دارد که توانایی کنترل تعداد نقاط زیر ۱۲۸ نقطه را دارا هستند.

این نوع PLC ها معمولا دارای ساختاری به صورت کامپکت میباشند . به عبارت دیگر دارای تعداد مشخصی کانال های ورودی و خروجی ، خروجی منبع تغذیه ، ورودی ها و خروجی های سرعت بالا و.... بر روی ماژول CPU میباشند.

ماژول CPU:

ماژول CPU در اصل همان هسته اصلی سیستم کنترلی میباشد.برنامه کنترلی پس از آنکه توسط نرم افزار مربوطه نوشته شد و بر روی ماژول دانلود شد ، در حافظه های داخلی این ماژول ذخیره گردیده و سپس اجرا میشود. در همان زمان که برنامه در حال اجرا میباشد ، CPU عملیات چک و تست را نیز در مورد عملکرد صحیح CPU، فضاهای حافظه و نیز وضعیت ماژول های افزایشی را انجام میدهد.

ماژول های افزایشی و BUS افزایشی :

ماژول های افزایشی د ر PLC های سری KINCO-K3 به منظور افزایش تعداد کانال های ورودی و خروجی استفاده میگردد .

BUSافزایشی اتصال و ارتباط ماژول های افزایشی و CPU را برقرا ر مینماید. ارتباط بین BUSافزایشی و ماژول CPU از طریق یک کابل فلت 16-CORE میباشد.

KINCOBUILDER:

نرم افزاربرنامه نویسی PLC های سری KINCO-K3 بوده که مطابق با استاندارد IEC61131-3 میباشد.این نرم افزار دو زبان برنامه نویسی IL و LD را به منظور استفاده کاربران پشتیبانی میکند.

CPU firmware:

در اصل سیستم عامل داخلی ماژول CPU میباشد که در حافظه FLASH ذخیره شده است . زمانی که تغذیه CPU برقرار میشود، عملکرد این سیستم عامل آغاز شده و تمامی وظایف ماژول CPU را مدیریت مینماید.

USER PROGRAM:

برنامه ای است که توسط کاربر در نرم افزار KINCOBUILDER نوشته میشود . این برنامه پس از دانلود بر روی ماژول CPU در حافظه FRAM ذخیره می گردد.هر زمان که CPU روشن میگردد ، ابتدا این برنامه از حافظه FRAM خوانده شده و به منظور اجرا به حافظه RAM منتقل میگردد.

Main program و scan cycle :

CPU تمامی فرامین برنامه را به صورت پیوسته و دوره ای اجرا مینمایدد .این اجرای دوره ای فرامین را اسکن مینامیم .

MAIN برنامه اصلی ترین بخش پروژه میباشد . MAIN برنامه در هر سیکل اسکن یک بار اجرا میگردد.در هر پروژه تنها یک MAIN وجود دارد.

Free protocol communication:

ماژول cpu دارای پورت های ارتباطی سریال میباشد که پروتکل مخصوص برنامه نویسی ، Modbus RTU و نیز ارتباط FREE PTOTOCOL را پشتیبانی میکند.ارتباط FREE PROTOCOL به برنامه شما این اجازه را میدهد که به صورت کامل پورت های ارتباطی CPU را کنترل نماید.کاربر میتواند به منظور اجرای پروتکل های ارتباطی تعریف شده توسط کاربر برای ارتباط با تمامی تجهیزات هوشمند استفاده نماید. این مد هردو پروتکل باینری و ASCII را پشتیبانی میکند.

I/O Image Area:

این فضا شامل فضای تصویر ورودی (INPUT IMAGE AREA) و فضای تصویر خروجی (OUTPUT IMAGE AREA) میباشد . در ابتدای هر سیکل اسکن وضعیت سیگنال ها از کانال های ورودی به فضای تصویر ورودی منتقل میگردد.در پایان هر سیکل اسکن نیز مقدار ذخیره شده در فضای حافظه خروجی به کانال های خروجی انتقال میابند.

RETENTIVE RANGE:

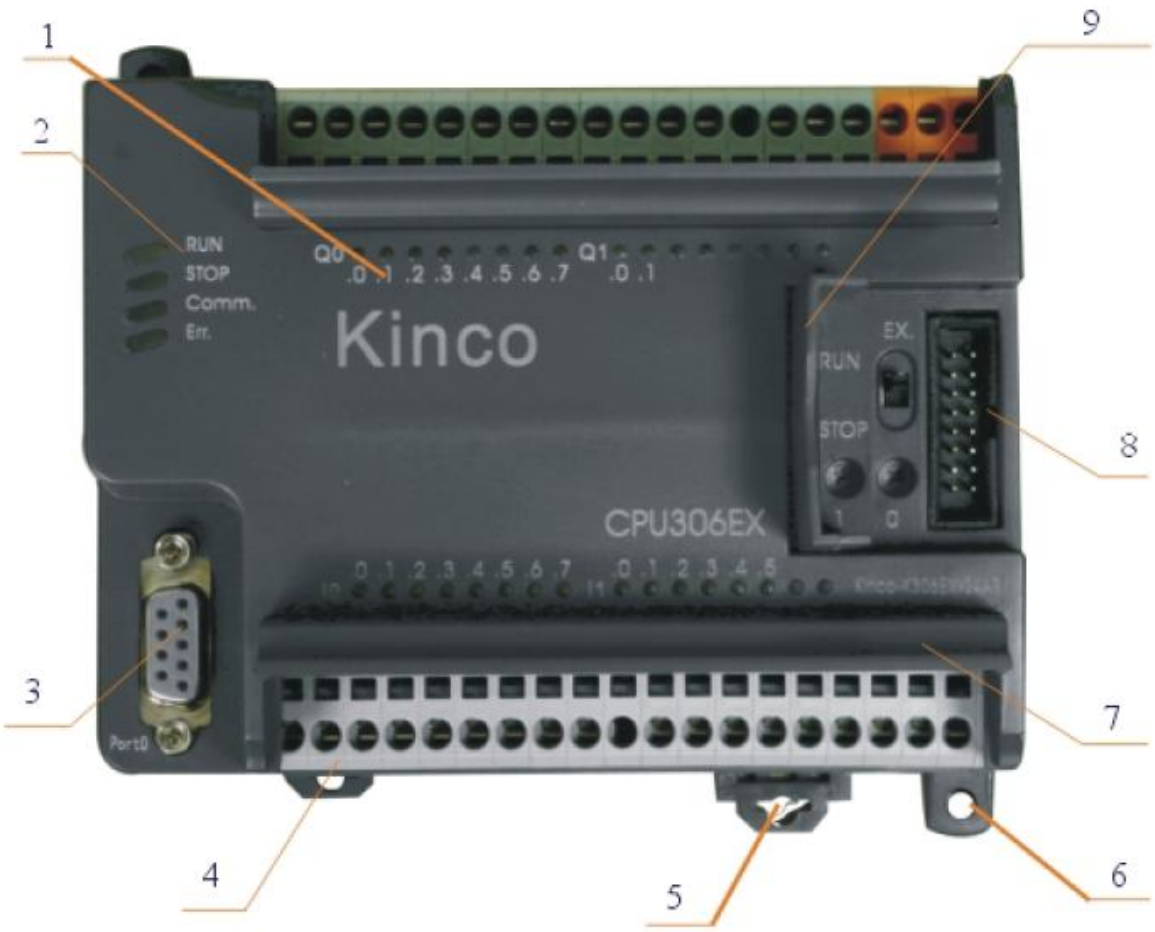
با استفاده از تنظیمات قسمت HARDWARE در نرم افزار KINCO BUILDER شما میتوانید ۴ رنج از داده ها را که در حافظه RAM قرار گرفته ان و میخواهید با قطع برق از بین نروندبه ماندگار نمایید .

در این حالت در زمانی که تغذیه cpuقطع میگردد، داده هایی که در فضای حافظه RAM قرار دارند توسط یک خازن داخلی نگهداری میشوند.بنابراین داده ها با قطع برق حفظ خواهند شد .

:DATA BACKUP

در این حالت کاربر میتواند برخی از داده ها را در حافظه های E2PROMو یا FRAM ذخیره نماید .توجه داشته باشید که هر کدام از این حافظه ها طول عمر مشخصی دارند به عنوان مثال حافظه E2PROM قابلیت نوشتن تا ۱۰۰ هزار بار و حافظه FRAM قابلیت نوشتن و خواندن به صورت نامحدود را دارد.

۱.۲: ساختار اصلی CPU:



۱: چراغ های (LED) مربوط به ورودی ها و خروجی ها

۲: چراغ های (LED) مربوط به وضعیت CPU(این چراغ ها مشخص مینماید که CPU در کدام وضعیت خود قراردارد)

۳:پورت ارتباطی (RS232,RS485)

۴:ترمینالها

۵: DIN RAIL برای نصب

۶:سوراخی که برای نصب

۷: درپوش که روی ترمینال قرار میگیرد

۸: پورت EXPANSION(از این پورت برای اضافه کردن کارت های افزایشی استفاده میشود)

۹:درپوش برای پورت EXPANSION

۱.۳: قوانین ویژه نام گذاری :

۱.۳.۱: قوانین نام گذاری محصولات :

نام محصول در PLCهای KINCOبیان گر نوع عملکرد و کاربرد آنها میباشد . با توجه به نام محصول میتوان به نوع محصول و نیز مشخصات پی برد .بنا براین اطلاع از قانون نام گذاری میتواند شما را در انتخاب قطعات مورد نظر یاری نماید.

نام هر محصول از اصول و قانون زیر پیروی میکند:

Product name: *module type + 3 + subtype + serial number*

:Module Type

CPU: ماژول CPU

PM: ماژول افزایش ورودی و خروجی

3: بیان گر سری KINCO-K3

:SUBTYPE

یک رقم میباشد :

۰: بیان گر این است که ماژول مورد نظر CPU میباشد

۲: بیان گر این است که ماژول مورد نظریک ماژول افزایش ورودی یا خروجی دیجیتال میباشد.

۳: بیان گر این است که ماژول مورد نظریک ماژول افزایشی ورودی یا خروجی آنالوگ میباشد.

(سیگنال های دیجیتال : سیگنال هایی است که مقدار آن یا صفر است یا یک .سیگنال های آنالوگ : سیگنال هایی است که شامل مقادیر پیوسته میباشد و رنج استاندارد آن در PLC های KINCO،برای سیگنال های جریانی :

4mA -20mA ,0-20mA و برای سیگنال های ولتاژی : 10 V-10 V- ,1V-5V میباشد.در ادامه بیشتر این مطلب را توضیح خواهیم داد)

SERIAL NUMBER:

چنانچه ماژول مورد نظر CPU باشد:

CPU:۴ با تعداد ۱۴ ورودی / خروجی دیجیتال بر روی آن

CPU :۶ با تعداد ۲۴ ورودی / خروجی دیجیتال بر روی آن

CPU:۸ با تعداد ۴۰ورودی / خروجی دیجیتال بر روی آن

چنانچه ماژول مورد نظر کارت افزایشی ورودی /خروجی دیجیتال باشد:

۱: ماژول از نوع ورودی دیجیتال میباشد

۲: ماژول از نوع خروجی دیجیتال میباشد

۳: ماژول از نوع ورودی /خروجی دیجیتال میباشد.

چنانچه ماژول مورد نظر کارت افزایشی ورودی /خروجی آنالوگ باشد:

۱: ماژول از نوع ورودی آنالوگ میباشد.

۲: ماژول از نوع خروجی آنالوگ میباشد.

۳: ماژول از نوع ورودی /خروجی آنالوگ میباشد.

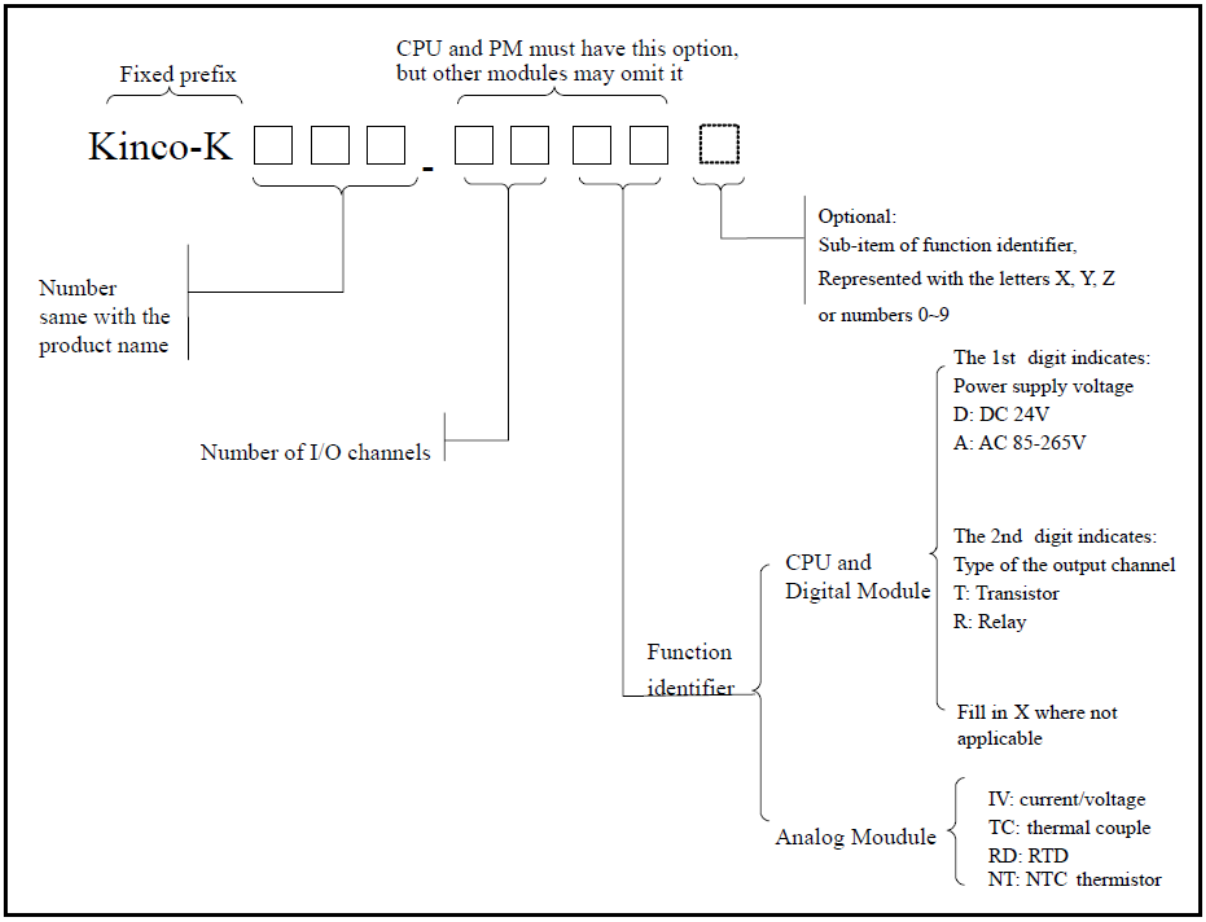
به عنوان مثال : CPU306 اشاره دارد به ماژول CPU با ۲۴ ورودی و خروجی دیجیتال

PM321: اشاره دارد به کارت افزایشی از نوع ورودی دیجیتال

۱.۳.۲: کد محصول(ORDER NUMBER):

هر محصول دارای یک کد محصول منحصر به فرد میباشد . به عبارت دیگر نام محصول اشاره به خانواده ای از محصولات دارد ولی کد محصول یک کد مشخص است که فقط یک محصول را شامل میشود.

Kinco-K + module code + feature code



بر

اسا الگوی ارائه شده در بالا :

KINCO-K306-24DT: بیان گر CPU با ۲۴ ورودی و خروجی بر روی خود ماژول –تغذیه ورودی 24 VDC و نیز خروجی ترانزیستوری میباشد.

(خروجی ها میتوانند رله ای یا ترانزیستوری باشند:

خروجی رله ای : در این مدل خروجی از یک رله داخلی در CPUگرفته میشود . بنابراین میتوان از این خروجی ها برای بارهای معمول و واقعی که با برق ۲۲۰ ولت هم روشن میشود استفاده نمود.

خروجی ترانزیستوری: در این مدل ،خروجی از یک ترانزیستور گرفته میشود . بنابراین فقط میتوان از این خروجی ها برای بارهای 24 VDC استفاده کرد . چنانچه بخواهیم از این مدل ها برای روشن کردن بارهای ۲۲۰ ولت استفاده نماییم باید خروجی را قبل از اتصال به بار به یک رله برد سپس خروجی رله را به بار متصل کرد)

K321-08DX: بیان گر ماژول افزایشی ورودی دیجیتال که دارای ۸ کانال ورودی ۲۴ ولت میباشد.

۱.۴ :سخت افزار KICO-K3:

در این قسمت به منظور آشنایی بیشتر با ساختار کلی PLCهای KINCO –K3به صورت کامل ماژول CPU ، کارت های افزایشی و سایر موارد سخت افزاری توضیح داده خواهد شد .

همان طور که در بالا اشاره شد PLC های KINCO-K3 در زیرمجموعه Micro plc ها قرار میگیرند و شامل یک خروجی تغذیه 24 VDC،پورت های ارتباطی و نیز کانال های ورودی و خروجی دیجیتال برروی ماژول CPUمیباشد. همچنین kinco-k3 قابلیت افزایش تعداد ورودی و خروجی های (آنالو گ و دیجیتال) را از طریق اضافه کردن ماژول های افزایشی داراست .

به طور کلی محصولات ذکر شده مطابق با لیست ارائه شده در پایین میباشد:

Type	Name	Order no.	Description
CPU module	CPU304	Kinco-K304-14AT	AC85~265V power supply, with 14 I/O, DI 8*DC24V, DO 6*DC24V, max output current per channel 0.75A. CPU304 has no expansion port.
		Kinco-K304-14AR	AC85~265V power supply, with 14 I/O, DI 8*DC24V, DO 6*Relay, max output current per channel 3A. CPU304 has no expansion port.
		Kinco-K304-14AX	AC85~265V power supply, with 14 I/O, DI 8*DC24V, DO 3*DC24V/3*Relay, max output current per channel 0.75A/3A. CPU304 has no expansion port.
	CPU304EX	Kinco-K304EX-14AR	AC85~265V power supply, with 14 I/O, DI 8*DC24V, DO 6*Relay, max output current per channel 3A
	CPU306	Kinco-K306-24DT	DC24V power supply, with 24 I/O, DI 14*DC24V, DO 10*DC24V, max output current per channel 0.75A
		Kinco-K306-24DR	DC24V power supply, with 24 channels, DI 14*DC24V, DO 10*Relay, max output current per channel 3A
		Kinco-K306-24AR	AC85~265V power supply, with 24 I/O, DI 14*DC24V, DO 10*Relay, max output current per channel 3A
	CPU306EX	Kinco-K306EX-24AR	AC85~265V power supply, with 24 I/O, DI 14*DC24V, DO 10*Relay, max output current per channel 3A, 1 RS232 and 1 RS485 port.
		Kinco-K306EX-24AT	AC85~265V power supply, with 40 I/O, DI 24*DC24V, DO 16*DC24V, max output current per channel 0.75A, 1 RS232 and 1 RS485 port.

Expansion I/O module	CPU308	Kinco-K308-40AT	AC85~265V power supply, with 40 I/O, DI 24*DC24V, DO 16*DC24V, max output current per channel 0.75A, 1 RS232 and 1 RS485 port.
		Kinco-K308-40AR	AC85~265V power supply, with 40 I/O, DI 24*DC24V, DO 16*Relay, max output current per channel 3A, 1 RS232 and 1 RS485 port.
		Kinco-K308-40AX	AC85~265V power supply, with 40 I/O, DI 24*DC24V, DO 4*DC24V/12*Relay, max output current per channel 0.75A/3A, 1 RS232 and 1 RS485 port.
	PM321	Kinco-K321-08DX	DI 8*DC24V
		Kinco-K321-16DX	DI 16*DC24V
	PM322	Kinco-K322-08DT	DO 8*DC24V, max output current per channel 0.75A
		Kinco-K322-16DT	DO 16*DC24V, max output current per channel 0.75A
		Kinco-K322-08XR	DO 8*Relay, max output current per channel 3A
		Kinco-K322-16XR	DO 16*Relay, max output current per channel 3A
	PM323	Kinco-K323-08DT	DI 4*DC24V, DO 4*DC24V, Max output current per channel 0.75A
		Kinco-K323-08DR	DI 4*DC24V, DO 4*Relay, Max output current per channel 3A
		Kinco-K323-16DT	DI 8*DC24V, DO 8*DC24V, Max output current per channel 0.75A
		Kinco-K323-16DR	DI 8*DC24V, DO 8*Relay, max output current per channel 3A
		Kinco-K323-08DTX	DIO 8*DC24V, diplex use, Max output current per channel 0.75A
	PM331	Kinco-K331-04IV	4 analog input channels, 0-20ma/4-20ma/±10V/1-5V optional for each channel
		Kinco-K331-04RD	4 RTD input channels, Pt100/Cu50, 2/3 wire optional for each channel
	PM332	Kinco-K332-02IV	2 analog output channels, 0-20ma/4-20ma/±10V/1*5V optional for each channel
	PM333	Kinco-K333-03IV	2 analog input channels, 1 analog output channel, 4-20ma/1-5V/0-10V optional for each channel
		Kinco-K333-04IV	2 analog input channels, 2 analog output channel, 4-20ma/1-5V/0-10V optional for each channel
Expansion Power Modules	PS380	Kinco-K380	Supply voltage:AC85~265V. Capability for expansion bus: +5V≤1300mA +24V≤250mA
Accessories and software	AS370	Kinco-K370-XXX	Programming cable, XXX indicates length of cable

باید توجه داشت که تعداد نقاط کنترلی و درنهایت ماکزیمم کارت افزایشی که میتوان به هریک از cpu های مشخص شده در لیست بالا اضافه نمود دارای محدودیت میباشد.

جدول زیر ماکزیمم ورودی ها و خروجی هایی که هر CPU میتواند پشتیبانی کند ارائه شده است :

	DI	DO		AI	AO	Number of Expansion modules
		Total	Relay			
CPU304	8	6	0*	0	0	0
CPU304EX	64	64	21*	16	16	4
CPU306	64	64	21*	16	16	4
CPU306EX	256	256	46*	32	32	15
CPU308	256	256	46*	32	32	15

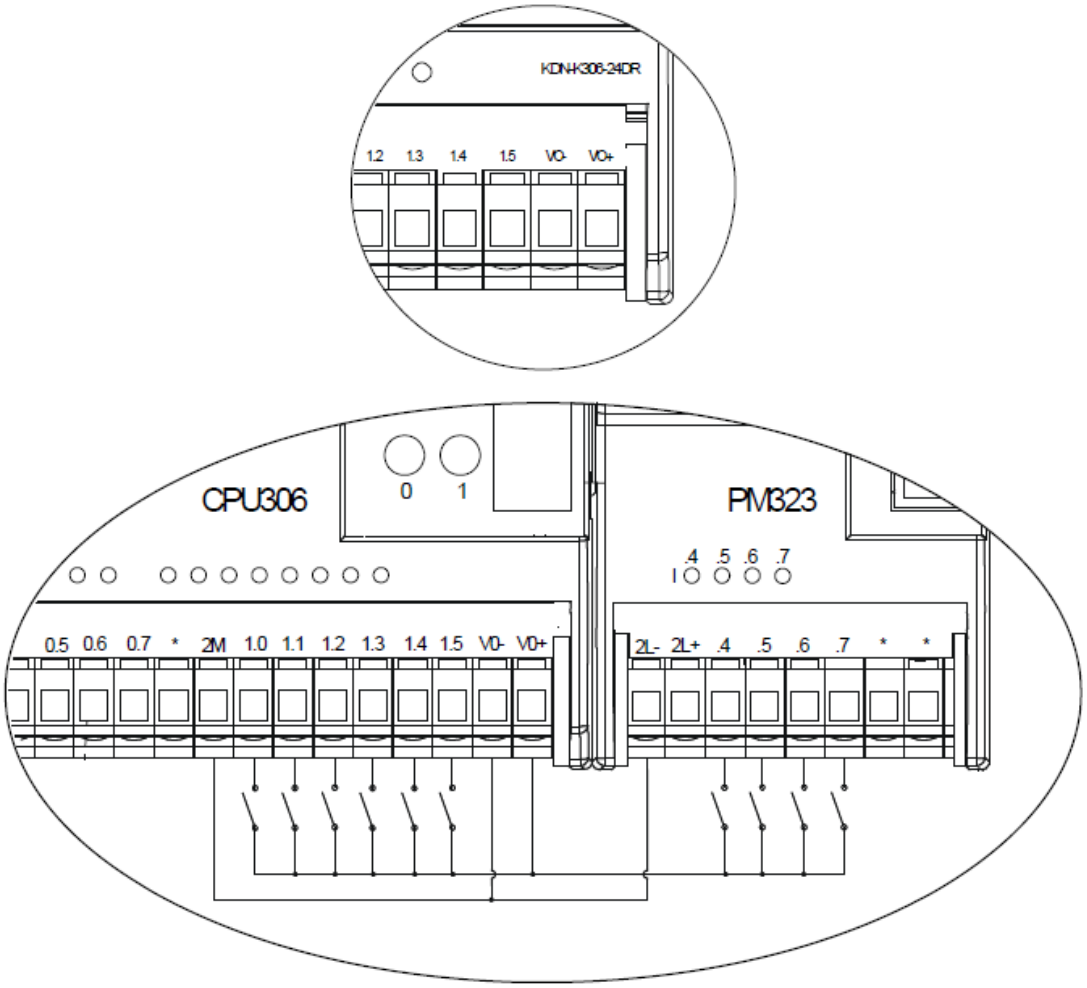
توجه داشته باشید که علامت * در جدول بالا بیان گر تعداد خروجی رله ای بر روی ماژول افزایشی میباشد .بنابراین این تعداد بدون در نظر گرفتن خروجی های رله ای بر روی cpu میباشد.

۱.۴.۱: خروجی 24VDC بر روی ماژول CPU:

برروی ماژول CPU یک خروجی 24vDC به صورت مستقل وجود دارد که با +VOو -VO نشان داده شده است . از این خروجی میتوان برای تغذیه سنسورهای مورد نیاز و یا کانال های ورودی و یا تغذیه کارت ها ی افزایشی در پروژه میتوان استفاده نمود.جدول زیر حداکثر جریانی که این خروجی 24VDC در هر ماژول CPU دارد را نشان میدهد:

CPU304	300 mA
CPU304EX. CPU306. CPU306EX	DC power: 300 mA AC poer: 500mA
CPU308	500 mA

تصویر زیر نشان دهنده این خروجی و یک نمونه سیم بندی میباشد:



۱.۴.۲:اتصالات بین ماژول PLC و کارت های افزایشی :

Expansion BUS(busافزایشی):

Bus افزایشی راه اتصال ماژول cpu و با کارت ها ی افزایشی و نیز کارت های افزایشی با یکدیگر میباشد.

ارتباط هر کارت و busافزایشی از طریق یک کابل فلت 16 –core ایجاد میشود که در این کابل کانالهای داده سرعت بالا ، آدرس کانالها ، تغذیه 5VDC،تغذیه 24VDC و زمین در این کابل مشخص میگردد.

Bus افزایشی در انتهای سمت راست هر کارت میباشد.

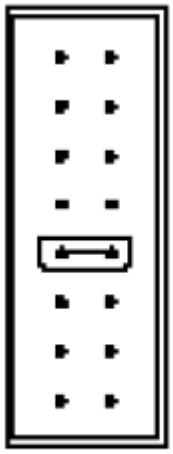
مدار تغذیه CPU و نیز ماژول PS380 هم تغذیه ۵ ولت و هم ۲۴ ولت را برای BUS افزایشی فراهم میکنند.بنابراین تغذیه ۵ ولت برای تامین تغذیه مدار داخلی ماژول افزایشی به کار میرود.ماکزیمم جریان هر دو منبع تغذیه ی (در BUS افزایشی) فراهم شده توسط هر کدام از CPUها در جدول زیر نشان داده شده است :

		5VDC power supply	24VDC power supply
CPU304		-	-
CPU304EX. CPU306. CPU306EX	AC power	720mA	165mA
	DC power	720mA	120mA
CPU308		1400mA	320mA
PS380		1300mA	250mA

The Max Currents of 5VDC and 24VDC Power Supply in the Expansion Bus

در هر اتصال بین CPU و ماژول های افزایشی ، همیشه ماژول CPU در انتهای سمت چپ سیستم قرار گرفته و ماژول های افزایشی به ترتیب توسط کابل افزایشی (که در سمت چپ هرکدام از ماژول ها قرار دارد) به CPUمتصل میشوند و نهایتا در طرف راست ماژول CPUقرار میگیرند.سوکت ۱۶ پینی که در سر کابل افزایشی کارت اول وجود دارد در پورت افزایشی سمت راست ماژول CPU قرار میگیرد .سپس سوکت ماژول دوم هم در پورت افزایشی ماژول اول(که در سمت راست ماژول اول میباشد) قرارمیگیرد .سایر کارتهای افزایشی هم با همین روند به ترتیب قرار خواهند گرفت.

همان طور که در بالا اشاره شد CPU308 میتواند ۱۵ کارت افزایشی را پشتیبانی نماید .چنانچه طول busافزایشی از ۱ متر بیشتر شد و یا تعداد کارتهای زیاد شد توصیه میشود به منظور افزایش پایداری ارتباط پین ۹ و ۱۰ پورت افزایشی آخرین ماژول را به یکدیگر متصل نمایید (jumperکنید).



۱.۴.۳:پورت های ارتباطی :

CPU های Kinco-K3 دارای پورت های ارتباطی RS232 و RS485 میباشد.همچنین این CPU ها پروتکل های Modbus RTU و نیز مد protocol free را نیز پشتیبانی مینماید. (CPU های KINCO به صورت از پیش تعیین شده میتوانند به عنوان Modbus RTU slave به کار برده شوند)

همچنین PLC های Kinco-k3 میتوانند با HMI هایی را که پروتکل Modbus RTU را پشتیبانی میکنند،ارتباط برقرار کنند. از طرف دیگر میتوان از مد free-protocol میتوان برای ارتباط با سایر تجهیزات هوشمندی که دارای پروتکل ارتباطی مخصوص به خود هستند استفاده نمود.

به علاوه میتوان تا ۳۲ PLC را با استفاده از RS485 در شبکه با استفاده از Modbus RTU و یا پروتکل مشخص شده توسط کاربر به یکدیگر متصل نمود.

1.5:شرایط محیطی :

شرایط محیطی که برای تمامی CPU های Kinco –k3باید در نظر گرفت در جدول زیر بیان شده است :

Operating temperature	0 --- +55 °C
Allowable relative humidity	95%, no condensation
Storage temperature	-20 --- +85 °C

فصل دوم : قواعد کلی در رابطه با CPU و عملکرد آن:

مقدمه :

ماژول CPUهسته اصلی و مرکزی در PLCهای سری KINCO-k3 میباشد و میتواند از طریق Busافزایشی به ماژول های افزایشی متصل شده ودر سیستم های کنترلی به کار رود.

CPU در هر سیکل اسکن مراحل "خواندن ورودی ها > اجرای برنامه >پردازش درخواست های ارتباطی> چک کردن وضعیت خود> نوشتن بر روی خروجیها > خواندن ورودی ها..... را به صورت دائم اجرا میکند. ضمنا" bus افزایشی را نیز به منظور در دسترس بودن تمامی کارت های افزایشی کنترل میکند.

Firmware ماژول CPU نیز در حافظه flash(غیر فرار و ماندنی) ذخیره میشود .وظیفه firmware مدیریت تمامی عملکرد های CPU میباشد.

هنگامی که برنامه از طریق نرم افزار KincoBuilder در ماژول CPU دانلود شد،در حافظه RAM قرار گرفته و سپس در حافظه FRAM ذخیره میگردد. زمانی که تغذیه CPUبرقرار میشود ، CPUبرنامه را از FRAMباز گردانده و تمامی عملیات منطقی را بر اساس برنامه اجرا میکند.

۲.۱:خازن داخلی :

ماژول CPU دارای یک خازن داخلی بوده که تمامی فضای RAMرا پس از قطع تغذیه CPUحفظ میکند.

زمانی که تغذیه مجددا وصل شد ، CPU حافظه RAMرا بررسی میکند ،فضاهای حافظه که به صورت retentiveتعریف شده باشند بدون تغییر باقی میمانند.(از طریق پنجره سخت افزار در نرم افزار kincoBuilder میتوان ۴رنج از فضای حافظه RAMرا retentive کرد .توضیحات مربوط به آن در بخش نرم افزار بیان شده است)

این خازن داده های حافظه RAM را در دمای طبیعی به مدت ۷۲ ساعت نگه داری میکند. پس از این مدت تمامی اطلاعات در حافظه RAM ممکن است که از بین برود.

۲.۲:فضای حافظه FRAM(Nonvolatile Memory ferroelectric):

CPUدارای یک فضای FRAM به منظور ذخیره سازی دائم برنامه و پیکربندی سخت افزاری میباشد.زمانی که تغذیه CPU برقرار میشود ، CPUبرنامه و اطلاعات مربوط به پیکربندی را از فضای FRAM به فضای RAM باز میگرداند.فضای FRAMدارای خصوصیات زیادی مانند قابلیت خواندن / نوشتن نامحدود ،قابلیت نگهداری داده ها به صورت دائم و.... میباشد.

قابلیت دیگر FRAM نگهداری دائم ۲۵۵ بایت داده در رنج های خاصی از فضای حافظه V میباشد.

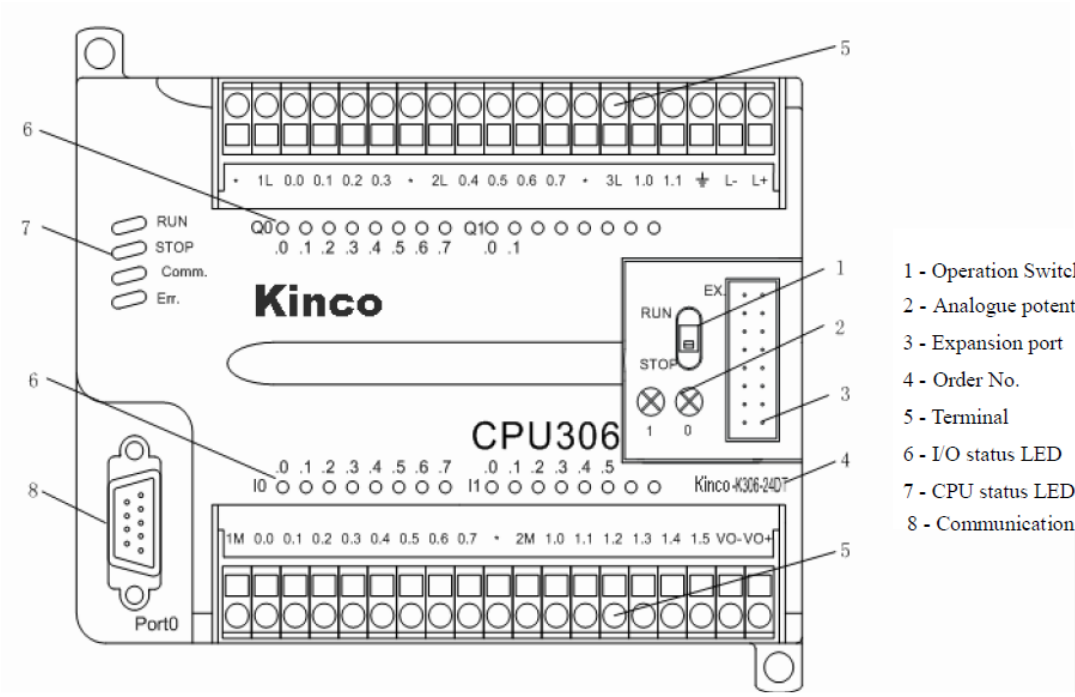
برای توضیحات بیشتر به پیوست Bمراجعه نمایید.

۲.۳:Real-Time clock (RTC):

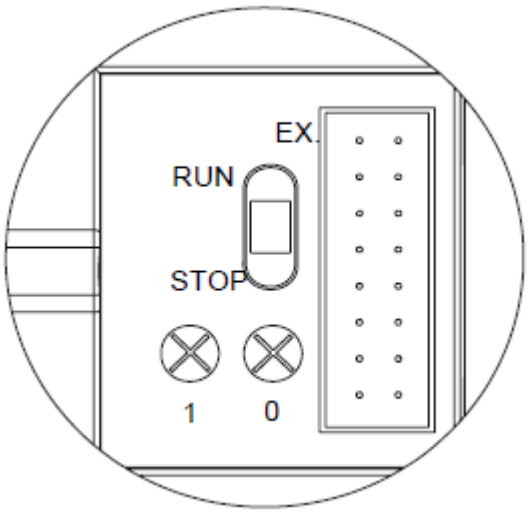
ماژول CPU دارای یک زمان سنج داخلی بوده که به صورت real-time میتوان برای نشان دادن زمان/تاریخ از آن استفاده نمود. این زمان سنج مطابق با فرمت BCDمیباشد. میتوان با استفاده از نرم افزار KincoBuilderبه صورت آنلاین اطلاعات RTC را خواند ویا آن را تنظیم نمود.

۲.۴:ساختار KINCO-K3:

تصویر زیر ساختار یک مدل از CPU های سری K3 (K306-24DT) را نشان میدهد:



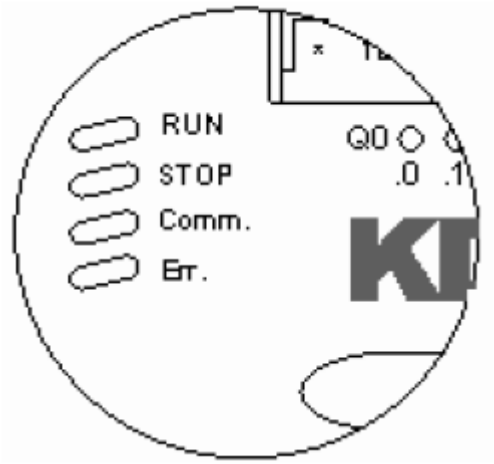
۲.۴.۱:Operation switch :



این سوئیچ دارای دو وضعیت بوده و به منظور تغییر وضعیت CPU از مد RUN به STOP استفاده میگردد.زمانی که CPU در وضعیت RUNقرار داشته باشد ، سیکل اسکن به صورت مداوم انجام میشود.

به علاوه میتوان با استفاده از نرم افزار KincoBuilder وضعیت CPU را از RUNبه STOP تغییر داد.

۲.۴.۲:وضعیت LED:



بر روی ماژول CPU ،۴ عدد LED وضعیت وجود دارد :

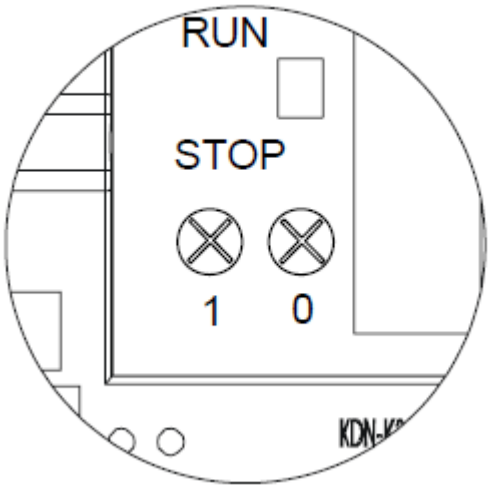
RUN LED: چنانچه به رنگ سبز باشد، CPUدر وضعیت RUN قرار دارد .

STOP LED: چنانچه به رنگ قرمز باشد، CPUدر وضعیت STOP قرار دارد .

Comm LED:چنانچه به رنگ سبز باشد، CPU در حال ارسال اطلاعات میباشد.

Err LED: چنانچه به رنگ قرمز باشد، بیان گر یک خطا ی اساسی در سیستم میباشد، این خطا باعث متوقف شدن اجرای برنامه میگردد. هنگامی که این خطا شناسایی شود CPUبه وضعیت Stopرفته و هم چراغ Err و هم چراغ Stop قرمز میشود.

۲.۴.۳:پتانسیومترهای آنالوگ :



ماژول CPU دارای دو پتانسیومتر آنالوگ با رزولوشن 10-bit میباشد که به ترتیب دارای شماره های ۰ و ۱ میباشند.

رنج تنظیمی آنها 0~1023 بوده و با چرخش آنها میتوان مقدار آنالوگی را کاهش ویا افزایش داد . چنانچه در جهت عقربه های ساعت چرخانده شود ، مقدارعددی آن افزایش و چنانچه خلاف جهت عقربه های ساعت باشد ، مقدارعددی آن کاهش میابد.

مقدار عددی این پتانسیومترهای آنالوگی به حافظه های SMW26و SMW28 ارسال میگردد.

SMW26 برای پتانسیومتر شماره ۱ و SMW28برای پتانسیومتر شماره ۰ میباشد. این دو حافظه فقط قابل خواندن بوده و میتواند به عنوان مقدار از پیش تعیین شده برای تایمرها و کانترها و... استفاده شود.

۲.۴.۴:کانال های I/O بر روی ماژول CPU:

برروی تمامی ماژول های CPU تعدادی کانال های ورودی دیجیتال (DI) و تعدادی کانال های خروجی دیجیتال (DO) وجود دارد .آدرس این کانال ها ثابت بوده و غیر قابل تغییر میباشد.

۲.۴.۴.۱:*کانال های ورودی دیجیتال:*

کانال های ورودی دیجیتال در قسمت پایینی CPUقرار دارد . کانال های ورودی به گروه های ۸ تایی تقسیم میشوند .به عنوان مثال : I0.0~I0.7 در گروه اول ، I1.0~I1.0 در گروه دوم و.... قرار دارد . تمامی کانال های ورودی دیجیتال دارای یک LED بوده که نشان دهنده وضعیت هر کانال میباشد .

کانال های ورودی دیجیتال فقط به عنوان ورودی های دیجیتال معمولی استفاده نمیشوند بلکه میتوان از آنها به عنوان کانالهای ورودی کانتر سرعت بالا استفاده نمود.

مشخصات اصلی کانال های ورودی دیجیتال (به عنوان مثال CPU -K306 توضیح داده میشود):

دارای ۱۴ کانال ورودی دیجیتال که به دو گروه تقسیم میشوند ، گروه اول شامل ۸ کانال و گروه دوم شامل ۶ کانال میباشد .

دارای آدرس های ورودی : I0.0~I0.7 و I1.0~I1.5

Soure(Common cathode) / Sink(Common anode) به صورت اختیاری برای هر گروه

استفاده به صورت ورودی های معمولی و ورودی پالس سرعت بالا

ولتاژ کانال ورودی 24VDC، (ولتاژ موثر و تحریک کننده کانال ها 15~30Vمیباشد)

دارای ایزولیشین opto-electricalبین سیگنال های ورودی و نیز مدار داخلی

LED های نشان دهنده وضعیت کانال ها

۲.۴.۴.۲ کانال های خروجی دیجیتال:

کانال های خروجی دیجیتال در قسمت بالای ماژول CPUقرار گرفته است .کانال های خروجی نیز به گروه های ۴ تایی تقسیم میشوند .به عنوان مثال: Q0.0~Q0.3درگروه اول ، Q0.4~Q0.7 در گروه دوم و....

بر اساس نوع خروجی دو نوع CPU وجود دارد : برخی از CPUها دارای خروجی های ترانزیستوری و برخی از CPU ها دارای خروجی های رله ای میباشند.در حالت خروجی ترانزیستوری : کانال های Q0.0و Q0.1 هم میتوانند به عنوان کانال های خروجی دیجیتال معمولی و هم به عنوان کانال خروجی پالس سرعت بالا استفاده میگردد. هر کانال خروجی دارای LEDهایی هستند که نشان دهنده وضعیت خروجی ها میباشند.

مشخصات اصلی کانال های خروجی ترانزیستوری (به عنوان مثال CPU- K306 توضیح داده میشود):

دارای ۱۰ کانال خروجی ترانزیستوری که به سه گروه تقسیم میشوند.

ولتاژ تغذیه 24VDC

ولتاژ خروجی 24VDC ، ماکزیمم جریان خروجی هر کانال 750mA و به صورت Source میباشد .

دارای محافظت در برابر پلاریته معکوس در وردی تغذیه

دارای محافظت در برابر بارهای القایی

محافظت در برابر اتصال کوتاه (هنگامی که جریان خروجی در هر گروه از 3A تجاوز کند)

اجازه اتصال موازی خروجی ها در یک گروه

دارای ایزولیشن opto-electrical بین سیگنال های خروجی و مدار داخلی

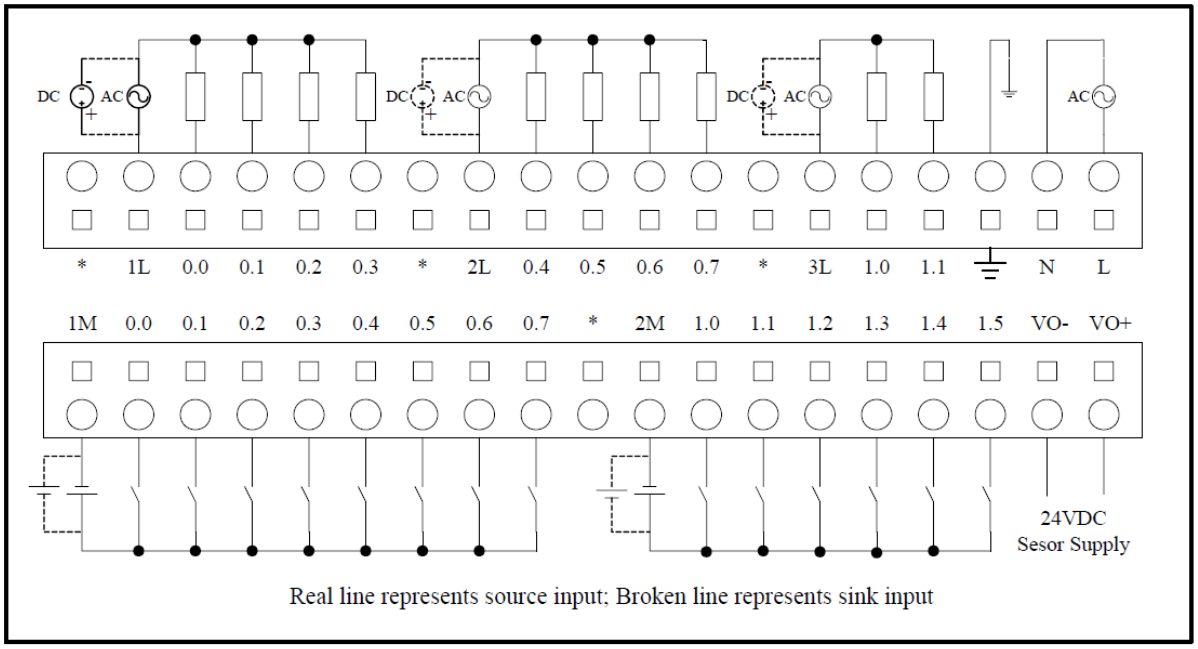
مشخصات اصلی کانال های خروجی رله ای (به عنوان مثال CPU- K306 توضیح داده میشود):

دارای ۱۰ کانال خروجی ترانزیستوری که به سه گروه تقسیم میشوند.

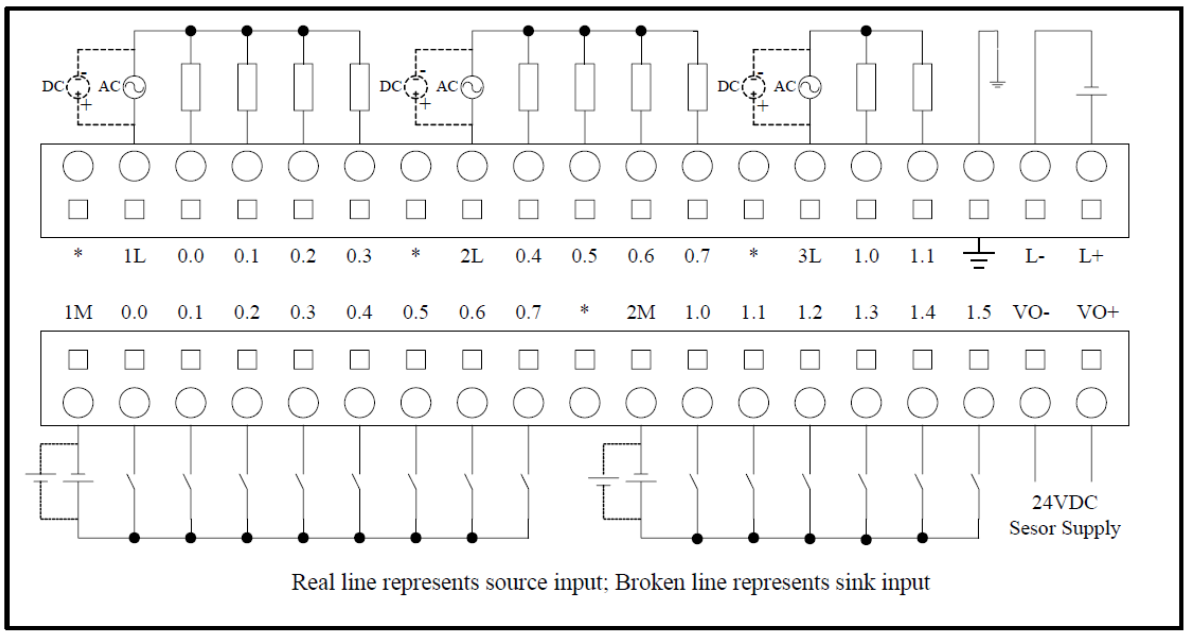
ولتاژ خروجی ماکزیمم 30VDC/270VAC

جریان خروجی هر کانال ماکزیمم 3A

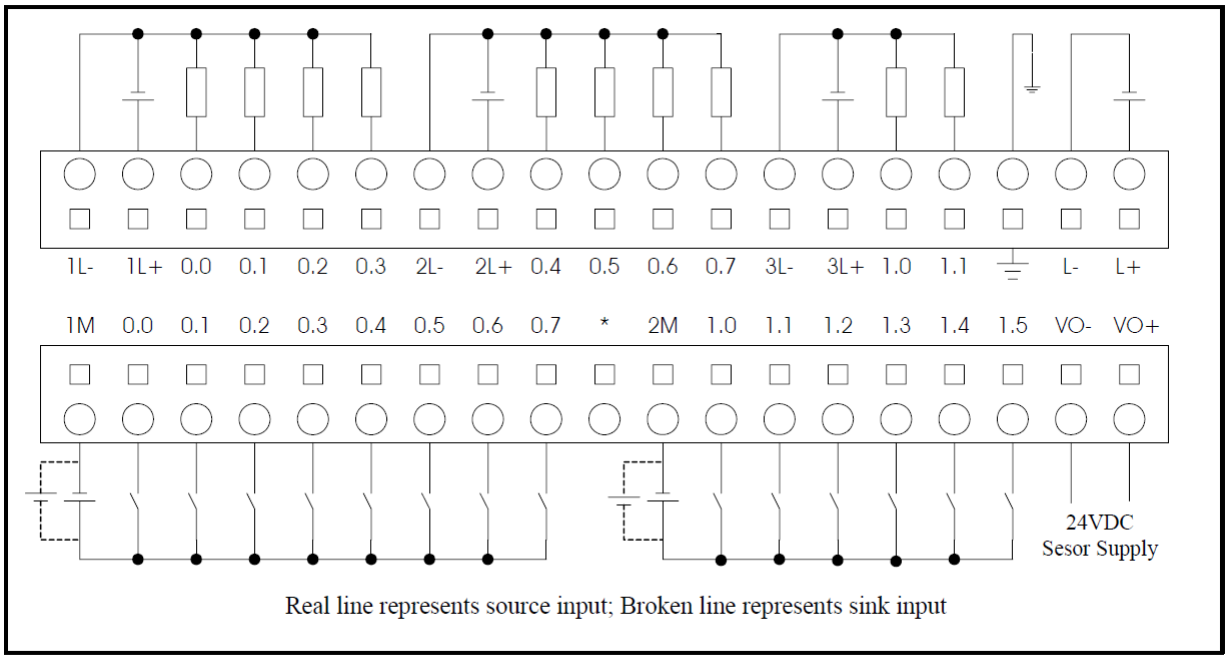
دیاگرام سیم بندی:



Wiring Diagram of Kinco-K306-24AR



Wiring Diagram of Kinco-K306-24DR



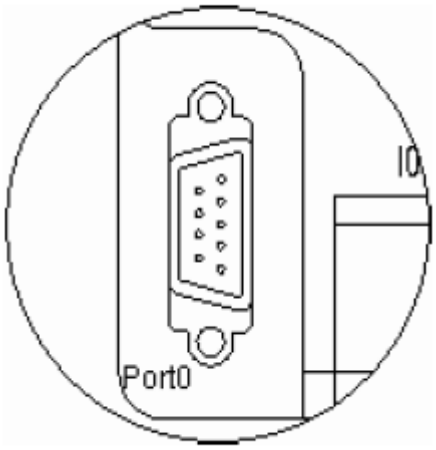
Wiring Diagram of Kinco-K306-24DT

۲.۵:پورت افزایشی :

در سمت راست ماژول CPU یک پورت افزایشی 16 pin به منظور اضافه نمودن ماژول های افزایشی به CPU وجود دارد .

توضیحات بیشتر در ادامه بیان شده است.

۲.۶:پورت ارتباطی :



CPU ها دارای یک پورت ارتباطی RS232 و یک پورت RS485میباشند.(CPU های K-304و K306 دارای یک پورت RS232و CPUهای k-306EXو K308 داری دو پورت RS232و RS485میباشند)

پورت ارتباطی که دارای یک کانکتور DB9 مادگی است در سمت راست ماژول CPUقرار دارد .از این پورت برای پروگرام کردن CPU و نیز ارتباط با سایر تجهیزات استفاده میگرد . این پورت ارتباط سریال ، پروتکل Modbus RTU(به عنوان slave) و نیز مد ارتباطی free –protocolرا نیز پشتیبانی میکند.

Pinout پورت های ارتباطی به صورت زیر میباشد:

RS232		
Signal	Description	Pin No.
GND	Signal ground	5
TxD	Transmit data	3
RxD	Receive data	2

RS485		
Signal	Description	Pin No.
A+	RS485+	7
B-	RS485-	8

۲.۷:سایر عملکردها:

CPUهای KINCO علاوه بر تمامی مواردی که در بالا به آن اشاره شد، عملکردهای خاص دیگری مانند شمارنده های سرعت بالا، خروجی قطار پالس سرعت بالا، وقفه و نیز ارتباط به صورت Free-protocol را نیز پشتیبانی میکند.

به عنوان مثال CPU306 دارای ۶ شمارنده سرعت بالا(HSC0~HSC5) میباشد که هر کدام از این شمارنده ها قابلیت دریافت ورودی با فرکانس 30Khzرا دارا هستند .هر یک از این شمارنده ها دارای مدهای کاری مختلفی هستند .باید در نظر داشت که مدهای کاری یکسان در شمارنده های مختلف عملکردی مشابه دارند. برای توضیحات بیشتر به بخش شمارنده های سرعت بالا در قسمت نرم افزار مراجعه نمایید.

CPU های KINCOدارای دو تولید کننده PTO/PWM میباشند که PTOتولید کننده قطار پالس در خروجی و PWM تولید کننده پالس با duty cycle متغیر میباشد .فرکانس خروجی هر یک از آنها تا 20Khz میباشد . برای توضیحات بیشتر به بخش تولید کننده PTO/PWMدر قسمت نرم افزار مراجعه نمایید.

همچنین CPU های KINCOوقفه های مختلفی که هر کدام مربوط به واقعه ای خاص در CPU میباشد را پشتیبانی میکند . برای توضیحات بیشتر به بخش وقفه ها در قسمت نرم افزار مراجعه نمایید.

CPU های KINCOقابلیت پشتیبانی از ارتباط به صورت free-protocolرا نیز دارند . برای توضیحات بیشتر به بخش دستورات ارتباطی در قسمت نرم افزار مراجعه نمایید.

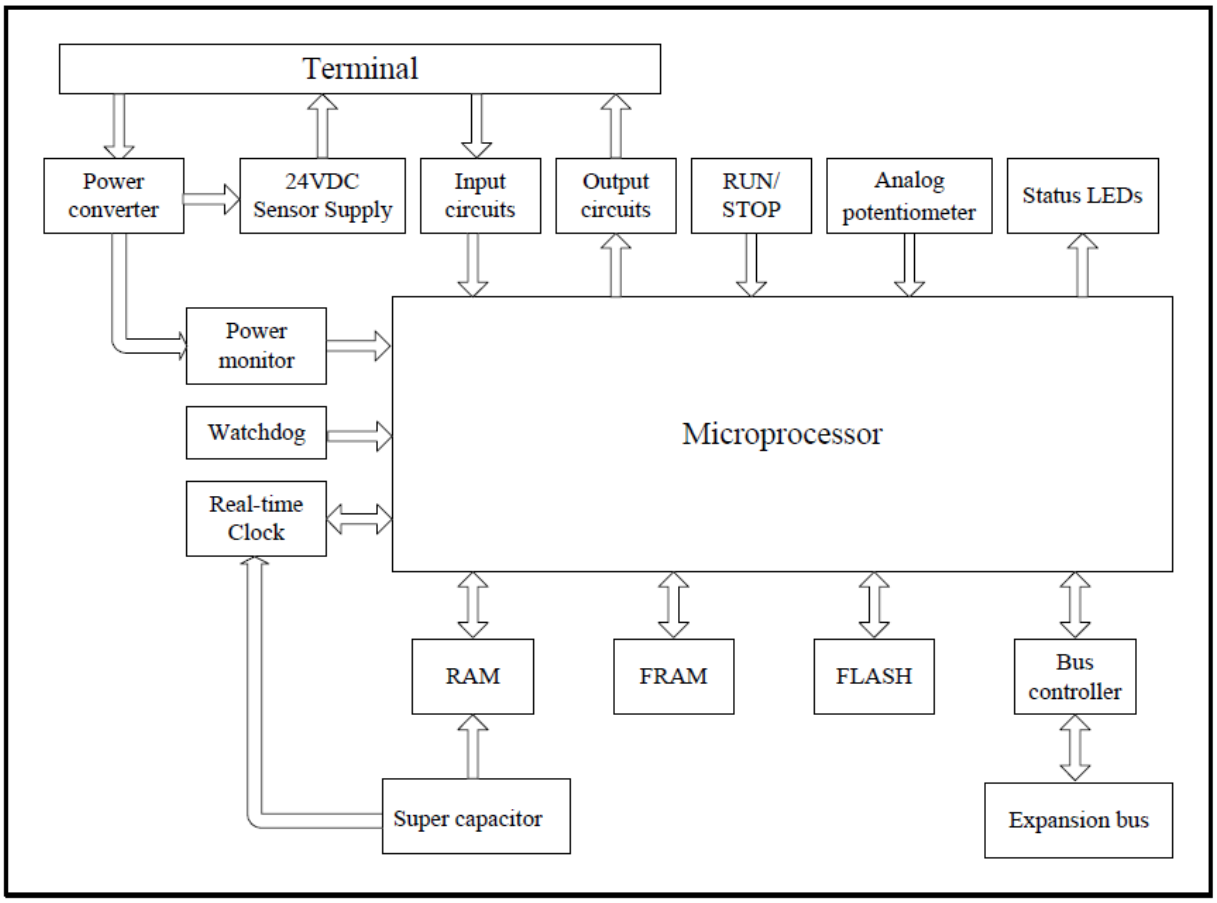
فصل سوم : سخت افزاری :

مقدمه:

CPUهای kinco از ترکیب یک میکرو پروسسور ۱۶ بیتی قوی ، RAM، حافظه FLASHبه منظور ذخیره سازی Firmware ، FRAM به منظور ذخیره سازی برنامه ، پاور ، مدارات ورودی ، مدارات خروجی ، RTCو ... تشکیل شده است که مجموعا ساختار یک PLC را تشکیل میدهند.

پس از آنکه کاربر برنامه را بر روی CPUدانلود نمود، CPU منطق برنامه را اجرا کرده و ورودی ها و تجهیزات خروجی را کنترل مینماید .

در تصویر زیر دیاگرام بلوک سخت افزاری یک cpuنشان داده شده است:



۳.۱:مشخصات CPU:

Feature	CPU304	CPU306	CPU308
Memory			
User program memory	E^2PROM, 8KB	FRAM, 8KB	FRAM, 32KB
RAM	64KB	64KB	128KB
Data retention	Super capacitor, 72 hours typical	Super capacitor, 72 hours typical	Li- battery, 2 month typical
I/O			
Built-in I/O channels	8*DI / 6*DO	14*DI / 10*DO	24*DI / 16*DO
DI image area	1 byte (8*DI)	8 bytes (64*DI)	32 bytes (256*DI)
DO image area	1 byte (8*DO)	8 bytes (64*DO)	32 bytes (256*DO)
AI image area	0	32 bytes (16*AI)	64 bytes (32*AI)
AO image area	0	32 bytes (16*AO)	64 bytes (32*AO)
Max.expansion modules	0	4	15
Anolog potentiometer	0	2, 10-bit resolution	2, 10-bit resolution
High-speed counters Single phase Two phase	2 counters total 2 at 20khz 2 at 10khz	6 counters total 6 at 30khz 4 at 20khz.	6 counters total 6 at 30khz 4 at 20khz.
High-speed pulse output	2 at 20khz	2 at 20khz	2 at 20khz
General			
Execution speed	Boolean instruction: 0.48μS Word instruction: <48μS Integer arithmetic instruction: <65μS Floating number arithmetic instruction: <150μS		
Timers	64 totally 1ms time-base: 4 10ms time-base: 16 100ms time-base: 44	128 totally 1ms time-base: 4 10ms time-base: 16 100ms time-base: 108	256 totally 1ms time-base: 4 10ms time-base: 16 100ms time-base: 236
Counters	64	128	256
Real-time clock	No	Yes, deviation less than 2 min/month@25°C	Yes, deviation less than 2 min/month@25°C

۳.۱.۱: مشخصات ورودی دیجیتال(DI):

Input type	Source/Sink
Rated input voltage	DC 24V (“1”, when DC15~30V)
Rated input current	4.1ma@24VDC
Max input voltage of logic 0	5V@0.7ma
Minimum input voltage of logic 1	15V@2.5ma
Input filter time delay	5ms
Isolation between input and internal circuit · Mode · Voltage	 Opto-electrical isolation 1500VAC/1 min
Status indicator	Green LED

۳.۱.۲:مشخصات خروجی DC 24V:

Output type	Source
Rated power supply voltage · Reverse polarity protection	DC 24V Yes
Rated output voltage	DC 24V
Output current per channel	Max 750ma@24VDC
Output leakage current	Max 0.5μA
Output impedance Output delay · off-to-on · on-to-off	Max 0.2Ω 0.3--5μs 5μs
Isolation between output and internal circuit · Mode · Voltage	 Opto-electrical isolation 1,500VAC/1 min
Inductive load protection	Yes
Short-circuit protection	Yes (When output current per group exceeds 3A)
Parallel connection of outputs	Yes (in the same group)
Status indication	Green LED

۳.۱.۳:مشخصات خروجی رله ای :

Output type	Relay

Load voltage	DC30V/AC250V
Output current per channel	3A (DC30V/AC250V)
Output current per group	Max 10A
Output off-to-on delay	Max 10ms
Output on-to-off delay	Max 5ms
Max. Switching rate · No load · Rated load	 12,000 times/min 100 times/min
Expected life of the contacts · Mechanical life (no-load) · Electrical life (rated load)	 20,000,000 times (1,200 tims/min) 100,000 times (6 times/min)
Isolation · Mode · Between coil and contact · Between contacts	 Relay 2000Vrms 1000Vrms
Status indication	Green LED

۳.۲:ماژول های ورودی دیجیتال:

این فصل به منظور آشنایی بیشتر با ماژول های ورودی دیجیتال در PLC های سری KINCO-K3 میباشد .در این فصل دیاگرام سخت افزاری و نیز سیم بندی کارت های دیجیتال و نیز اطلاعات فنی این کارت ها ارائه شده است . تمامی مدل های ماژول های دیجیتال ورودی به نام PM321 شناخته میشوند.

۳.۲.۱: DI8*24VDC (ماژول ورودی دیجیتال ۸ کاناله):

شماره مدل:K321-08Dx

این ماژول دارای ۸ کانال دیجیتال به همراه 8LED به منظور نشان دادن وضعیت این ورودی ها میباشد .این ماژول سیگنال های دیجیتال را از محیط دریافت کرده و آن را از طریق bus افزایشی در فضای حافظه I در CPU مینویسد.

مشخصات کلی :

دارای ۸ کانال ورودی دیجیتال میباشد که به دو گروه ۴ کاناله تقسیم شده اند.

ورودی Source (کاتد مشترک) / sink(آند مشترک) به صورت اختیاری در هر گروه

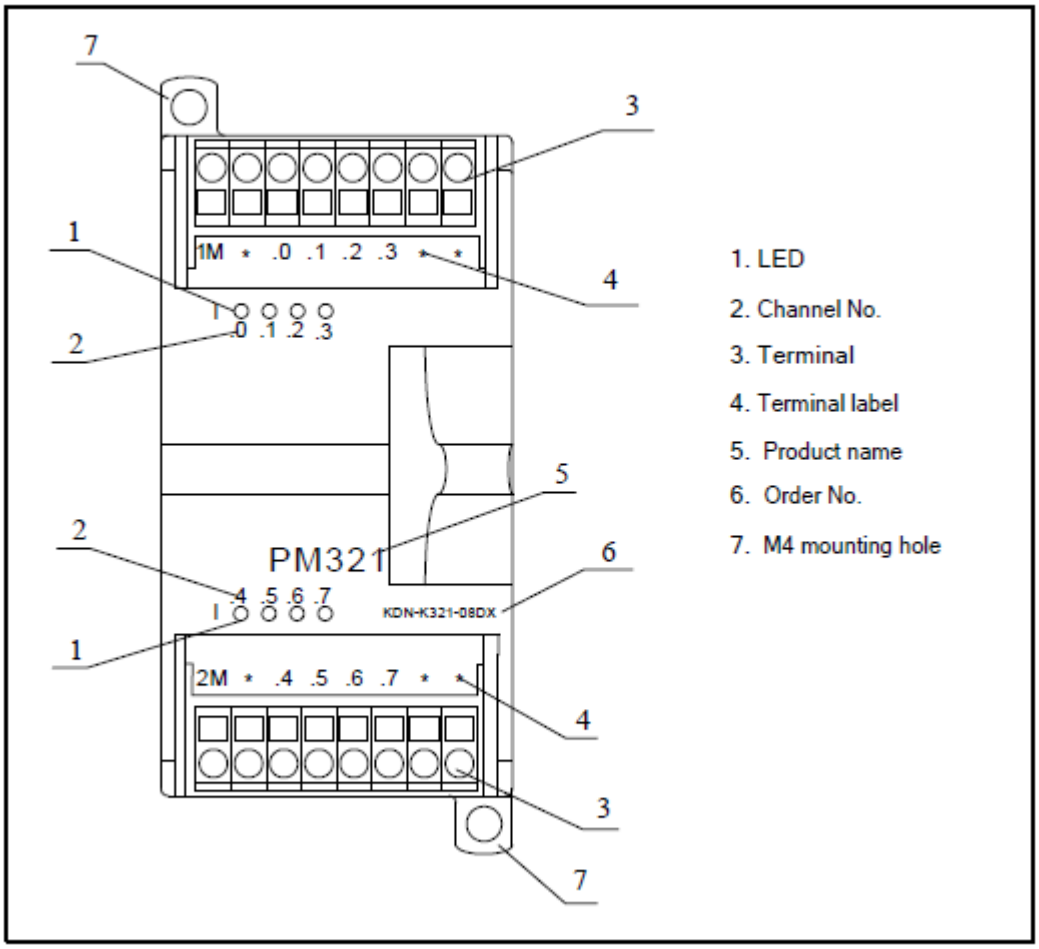
ولتاژ ورودی 24VDC (ولتاژ موثر برای تحریک ورودی 15~30V میباشد)

ایزولیشن opto-electricalبین سیگنال ورودی و مدار داخلی

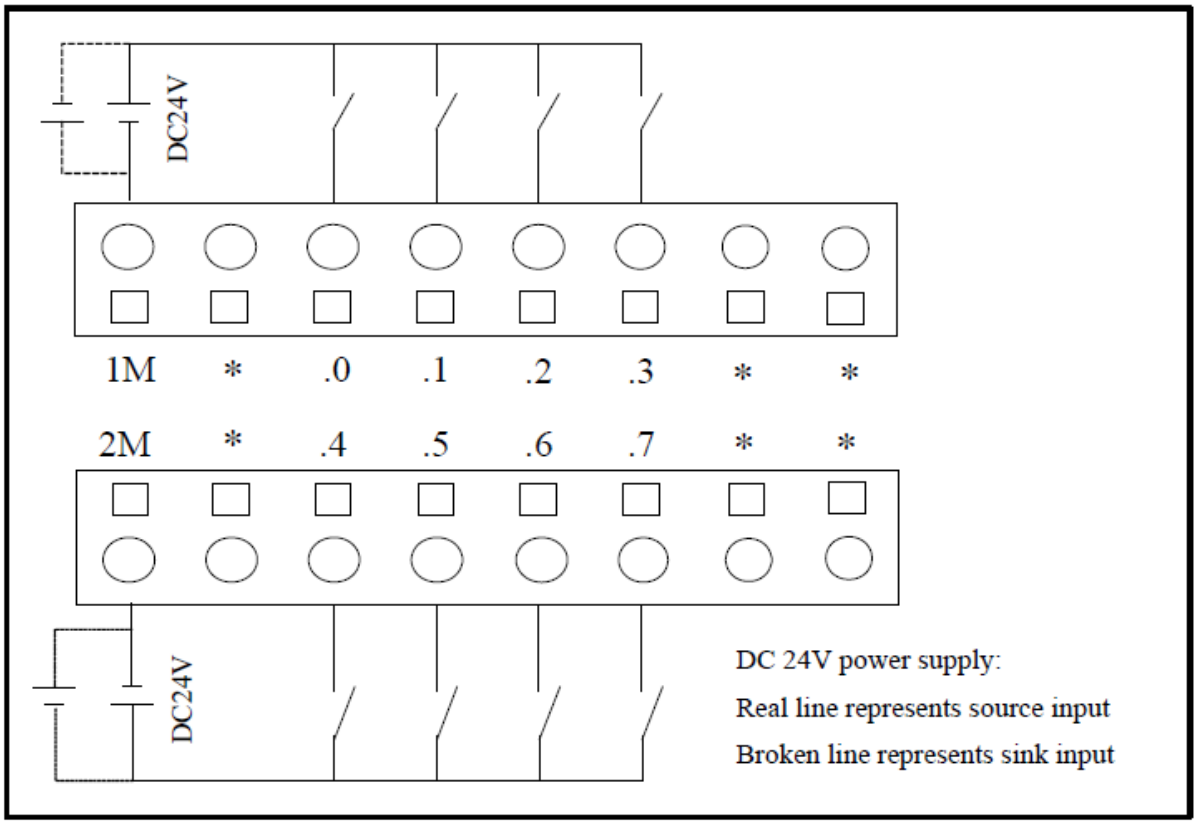
یک چراغ سبز LED برای هر کانال به منظور نمایش وضعیت هر کانال

طول ماژول 50mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:



دیاگرام سیم بندی این کارت مانند تصویر زیر میباشد:



مشخصات فنی این کارت مطابق با جدول زیر میباشد:

Electrical Data		
Number of channels	8 (4 channels/group)	
Input type	Source/Sink	
Rated input voltage	DC 24V (“1”, when DC15~30V)	
Rated input current	4.1ma@24VDC	
Max input voltage of logic 0	5V@0.7ma	
Minimum input voltage of logic 1	15V@2.5ma	
Input filter time delay	5ms	
Current consumption via expansion bus	5V	< 60ma
	24V	-
Isolation between input and internal circuit · Mode · Voltage	 Opto-electrical isolation 1500VAC/1 min	
Status indicator	Green LED	

Address occupied	
DI image area	1 byte
DO image area	-
Dimension and weight	
Dimension (L×W×H)	114×50×70mm
Net weight	125g

DI16*24VDC:۳.۲.۲ (ماژول ورودی ۱۶کاناله):

شماره مدل : K321-16Dx

این ماژول دارای 16 کانال دیجیتال به همراه 16LED به منظور نشان دادن وضعیت این ورودی ها میباشد .این ماژول سیگنال های دیجیتال را از محیط دریافت کرده و آن را از طریق bus افزایشی در فضای حافظه I در CPU مینویسد.

مشخصات کلی :

درارای 16 کانال ورودی دیجیتال میباشد که به دو گروه 8 کاناله تقسیم شده اند.

ورودی Source (کاتد مشترک) / sink(آند مشترک) به صورت اختیاری در هر گروه

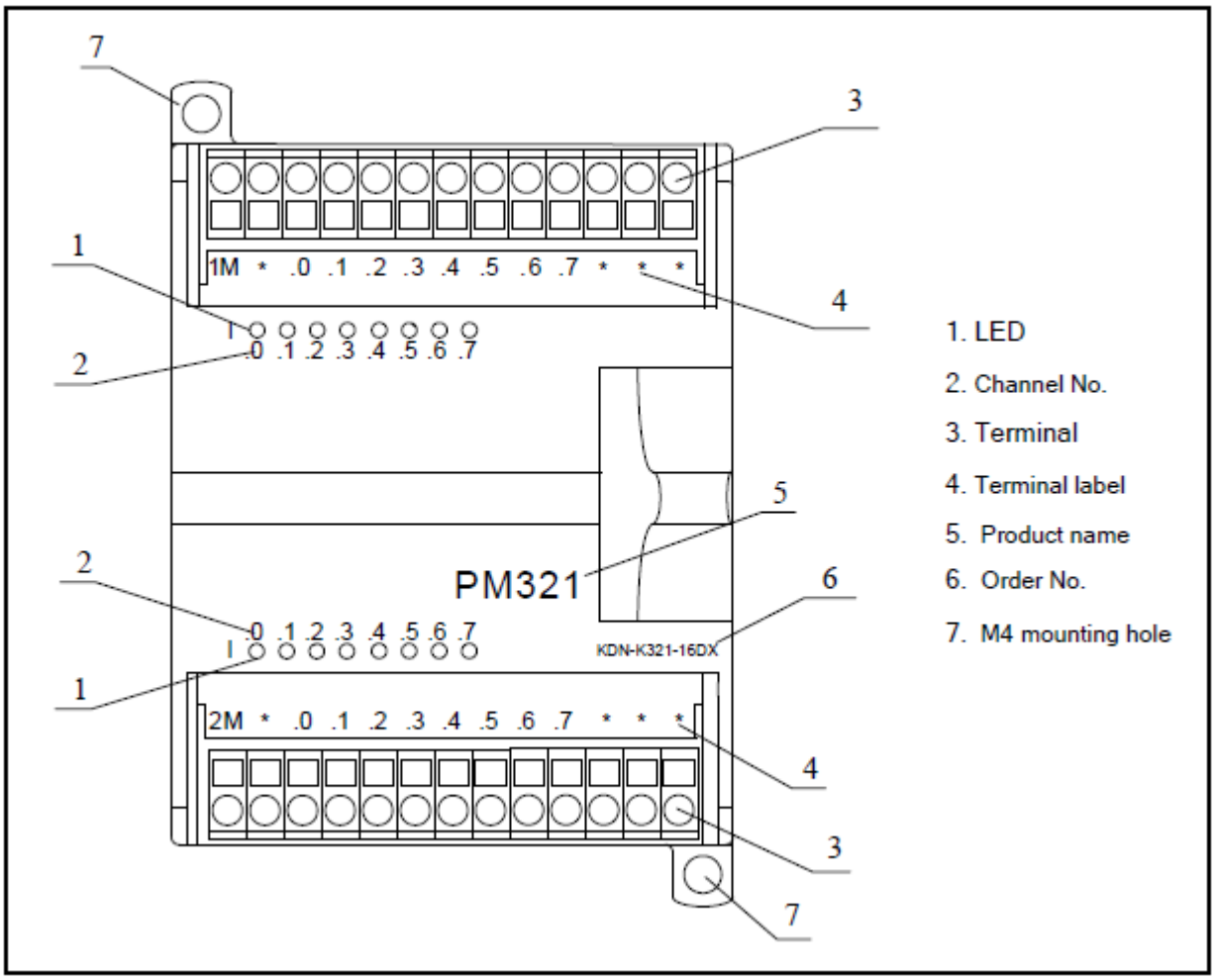
ولتاژ ورودی 24VDC (ولتاژ موثر برای تحریک ورودی 15~30V میباشد)

ایزولیشن opto-electricalبین سیگنال ورودی و مدار داخلی

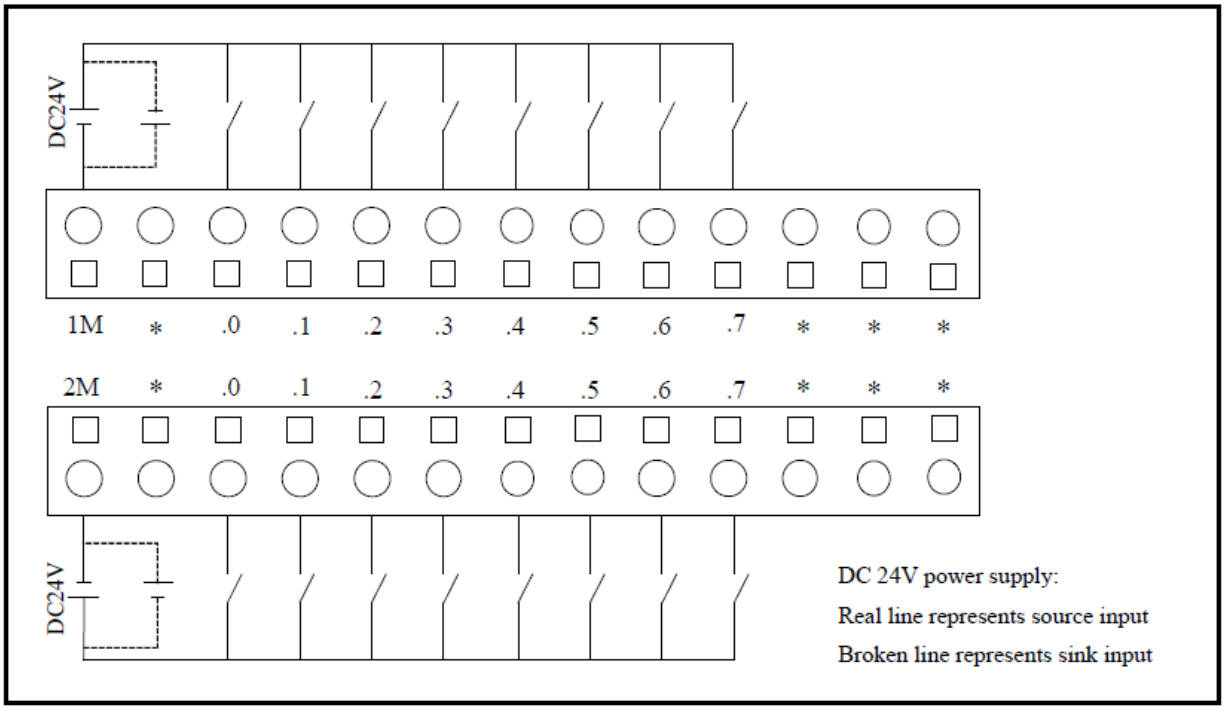
یک چراغ سبز LED برای هر کانال به منظور نمایش وضعیت هر کانال

طول ماژول 75mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:



دیاگرام سیم بندی این کارت مانند تصویر زیر میباشد:



مشخصات فنی این کارت مطابق با جدول زیر میباشد:

Electrical data		
Number of channels	16 (8 channels/group)	
Input type	Source/sink	
Rated input voltage	DC 24V (“1”, when DC15~30V)	
Rated input current	4.1ma@24VDC	
Max input voltage of “0”	5V@0.7ma	
Minimum input voltage of “1”	15V@2.5ma	
Input filter time delay	5ms	
Current consumption via expansion bus	5V	< 84mA
	24V	-
Isolation between input and internal circuit · Mode · Voltage	 Opto-electrical isolation 1500VAC/1 min	
Status indication	Green LED	
Address occupied		
DI image area	2 bytes	
DO image area	-	
Dimension and weight		
Dimension (L×W×H)	114×75×70mm	
Net weight	150g	

۳.۳: ماژول های خروجی دیجیتال(DO):

این فصل به منظور آشنایی بیشتر با ماژول های خروجی دیجیتال در PLC های سری KINCO-K3 میباشد .در این فصل دیاگرام سخت افزاری و نیز سیم بندی کارت های دیجیتال و نیز اطلاعات فنی این کارت ها ارائه شده است . تمامی مدل های ماژول های دیجیتال ورودی به نام PM322 شناخته میشوند.

۳.۳.۱:DO8*24VDC (ماژول خروجی دیجیتال ۸ کاناله):

شماره مدل : K322-08DT

این ماژول دارای ۸ کانال دیجیتال به همراه 8LED به منظور نشان دادن وضعیت این ورودی ها میباشد .این کانال ها سیگنال های کنترلی را از طریق busافزایشی دریافت کرده و به سیگنال های الکتریکی تبدیل میکند و از این طریق تجهیزات متصل به این کانالهارا کنترل میکند.

این ماژول نیاز به تغذیه 24VDC دارد.

مشخصات کلی :

درارای ۸کانال خروجی دیجیتال میباشد که به دو گروه ۴کاناله تقسیم شده اند.

تغذیه این کارت 24VDC میباشد.

ولتاژ خروجی کانال ها 24VDC میباشد .ماکزیمم جریان خروجی هر کانال 750mA و به صورت sourceمیباشد.

دارای محافظت در برابر پلاریته معکوس در تغذیه ورودی

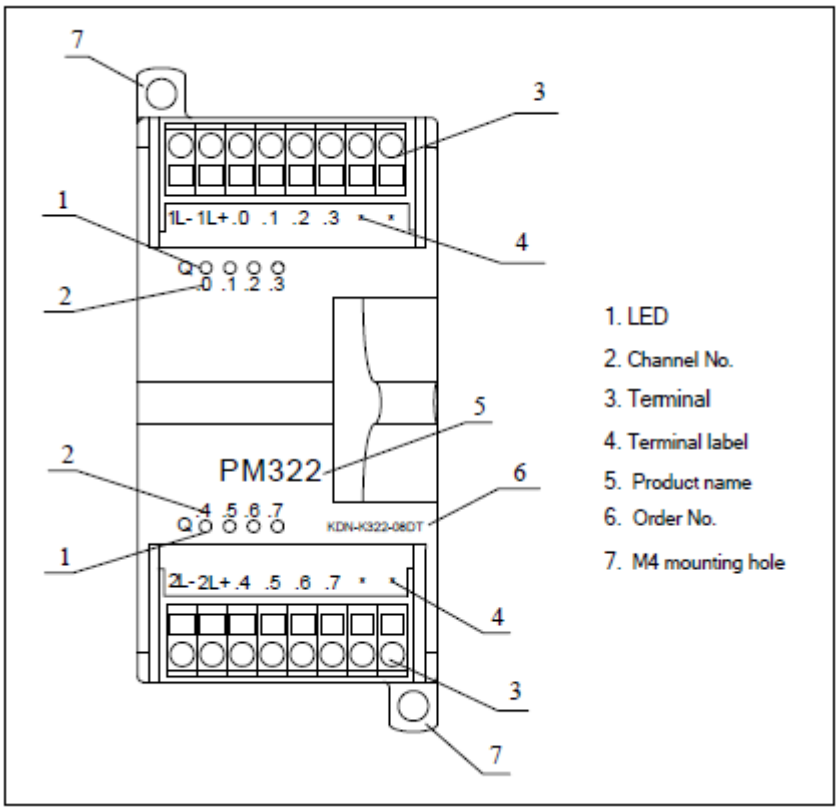
محافظت در برابر اتصال کوتاه (زمانی که جریان خروجی درهر گروه متجاوز از 3Aباشد)

اجازه اتصال موازی خروجی ها در یک گروه

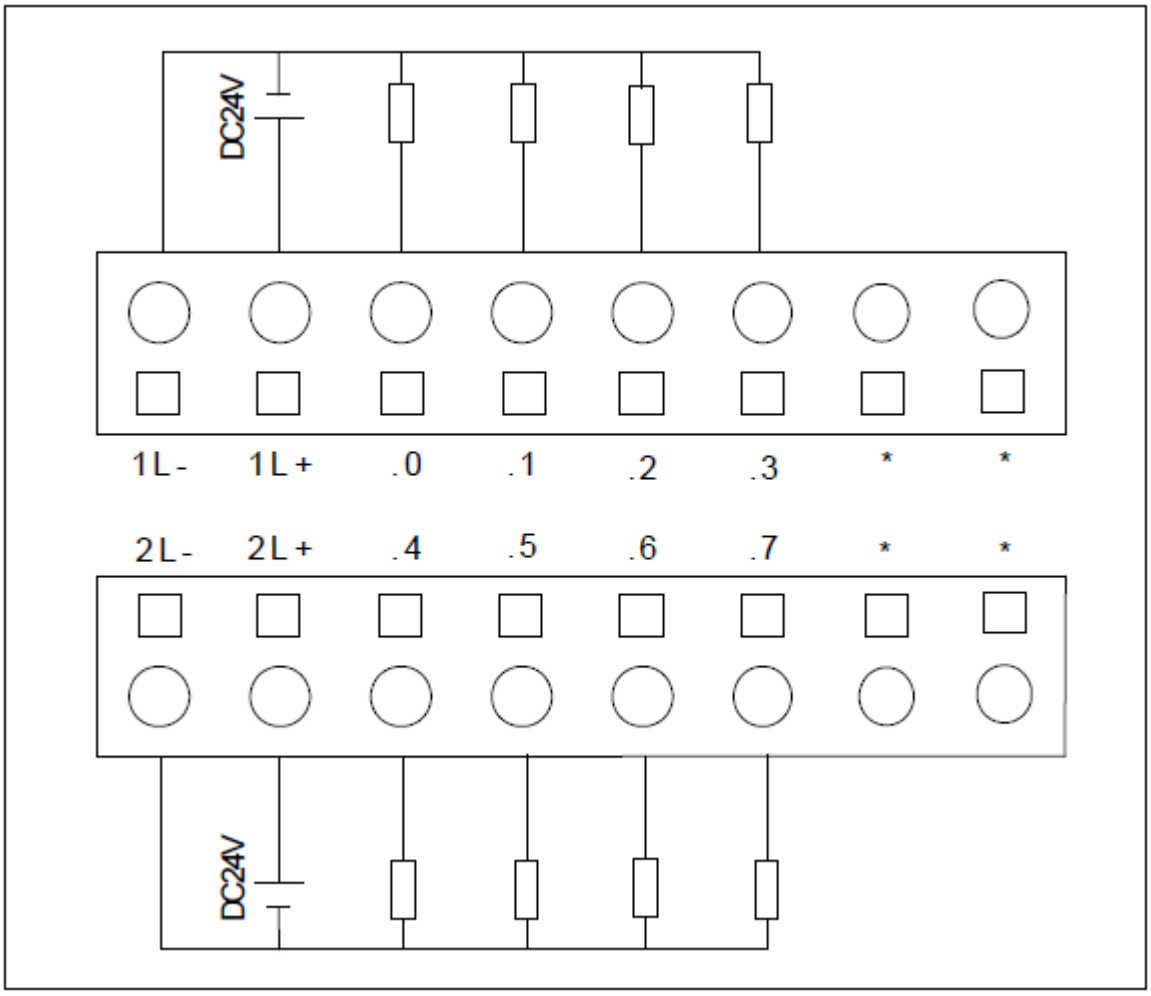
ایزولیشن opto-electricalبین سیگنال خروجی و مدار داخلی

طول ماژول 50mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:



دیاگرام سیم بندی این کارت مانند تصویر زیر میباشد:



مشخصات فنی این کارت مطابق با جدول زیر میباشد:

Electrical data		
Number of output channels	8 (4 channels/group)	
Output type	Source	
Rated power supply voltage · Reverse polarity protection	DC 24V Yes	
Rated output voltage	DC 24V	
Output current per channel	Max 750ma@24VDC	
Output leakage current	Max 0.5μA	
Output impedance	Max 0.2Ω	
Output delay · off-to-on · on-to-off	 0.3--5μs 5μs	
Current consumption via expansion bus	5V	<62.6ma
	24V	-
Isolation between output and internal circuit · Mode · Voltage	 Opto-electrical isolation 1,500VAC/1 min	
Inductive load protection	Yes	
Short-circuit protection	Yes (when output current per group exceeds 3A)	
Parallel connection of outputs	Yes (in the same group)	
Status indication	Green LED	
Address occupied		
DI image area	-	
DO image area	1 byte	
Dimension and weight		
Dimension (L×W×H)	114×50×70mm	
Net weight	125g	

۳.۳.۲:DO8*Relay(ماژول خروجی ۸ کاناله):

شماره مدل: K322-08XR

این ماژول دارای ۸ کانال دیجیتال به همراه 8LED به منظور نشان دادن وضعیت این ورودی ها میباشد .این کانال ها سیگنال های کنترلی را از طریق busافزایشی دریافت کرده و به سیگنال های الکتریکی تبدیل میکند و از طریق خروجی های رله ای تجهیزات متصل به این کانالهارا کنترل میکند.

مشخصات کلی :

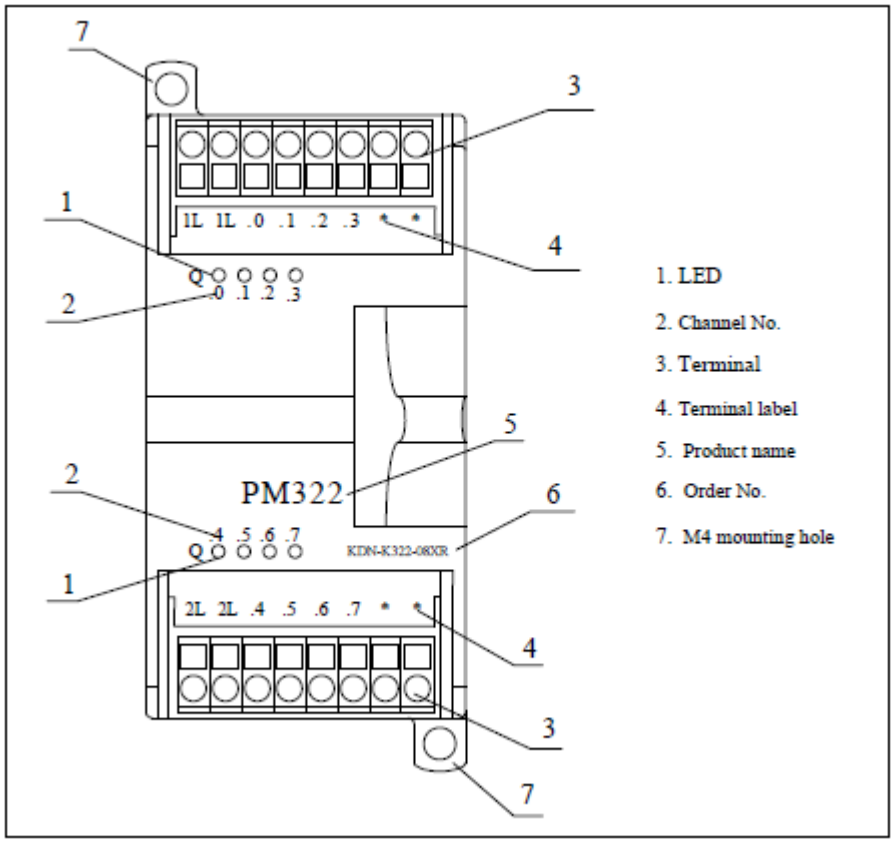
دارای ۸کانال خروجی دیجیتال میباشد که به دو گروه ۴کاناله تقسیم شده اند.

ولتاژ خروجی DC30V/AC250V

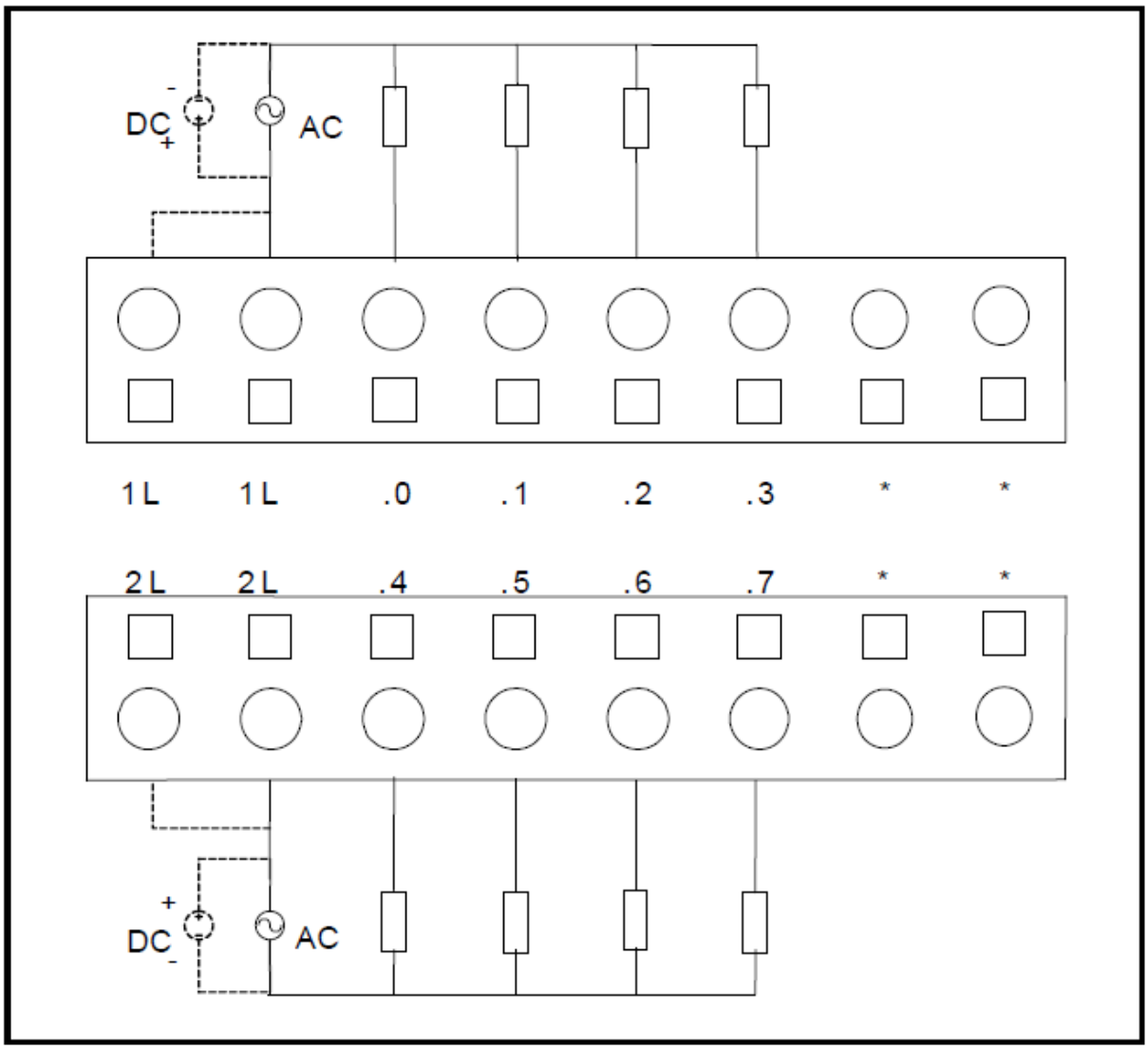
ماکزیمم جریان خروجی هر کانال 3A

طول ماژول 50mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:



دیاگرام سیم بندی این کارت مانند تصویر زیر میباشد:



مشخصات فنی این کارت مطابق با جدول زیر میباشد:

Electrical data		
Number of relay outputs	8 (4 channels/group)	
Load voltage	Max DC 30V/AC250V	
Output current per channel	Max 3A (DC 30V/AC250V)	
Output current per group	Max 10A	
Output off-to-on delay	Max 10ms	
Output on-to-off delay	Max 5ms	
Current consumption via expansion bus	5V	< 67.6 mA
	24V	< 42 mA
Max. Switching rate · No load · Rated load	 12,000 times/min 100 times/min	
Expected life of the contacts · Mechanical life (no-load) · Eectrical life (rated load)	 20,000,000 times (1200 times/min) 100,000 times (6 times/min)	
Isolation · Mode · Between coil and contact · Between contacts	 Relay 2000Vrms 1000Vrms	
Status indication	Green LED	
Address occupied		
DI image area	-	
DO image area	1 byte	
Dimension and weight		
Dimension (L×W×H)	114×50×70mm	
Net weight	150g	

ماژول های K322-16DT و K322-16XR نیز مشابه موارد رائه شده در بالا میباشد.

۳.۴: ماژول های دیجیتال ورودی / خروجی :

۳.۴.۱: DI4*DC24V , DO4 *DC24V :DI/DO

شماره مدل :K323-08DT

این ماژول دارای ۸ کانال میباشد که ۴ کانال آن به صورت ورودی ۲۴ ولت و ۴ کانال آن به صورت خروجی میباشد . هر کانال دارای یک LEDبرای نشان دادن وضعیت ورودی ها و خروجی ها میباشد.

مشخصات کلی:

درارای ۸ کانال میباشد که ۴ کانال به صورت ورودی دیجیتال ۲۴ ولت (به صورت یک گروه میباشند) و ۴ کانال به صورت خروجی دیجیتال ۲۴ ولت (به صورت یک گروه میباشند) است

ورودی Source (کاتد مشترک) / sink(آند مشترک) به صورت اختیاری در هر گروه

ولتاژ ورودی 24VDC (ولتاژ موثر برای تحریک ورودی 15~30V میباشد)

ایزولیشن opto-electricalبین سیگنال ورودی و مدار داخلی

ولتاژ خروجی کانال ها 24VDC میباشد .ماکزیمم جریان خروجی هر کانال 750mA و به صورت sourceمیباشد.

دارای محافظت در برابر پلاریته معکوس در تغذیه ورودی

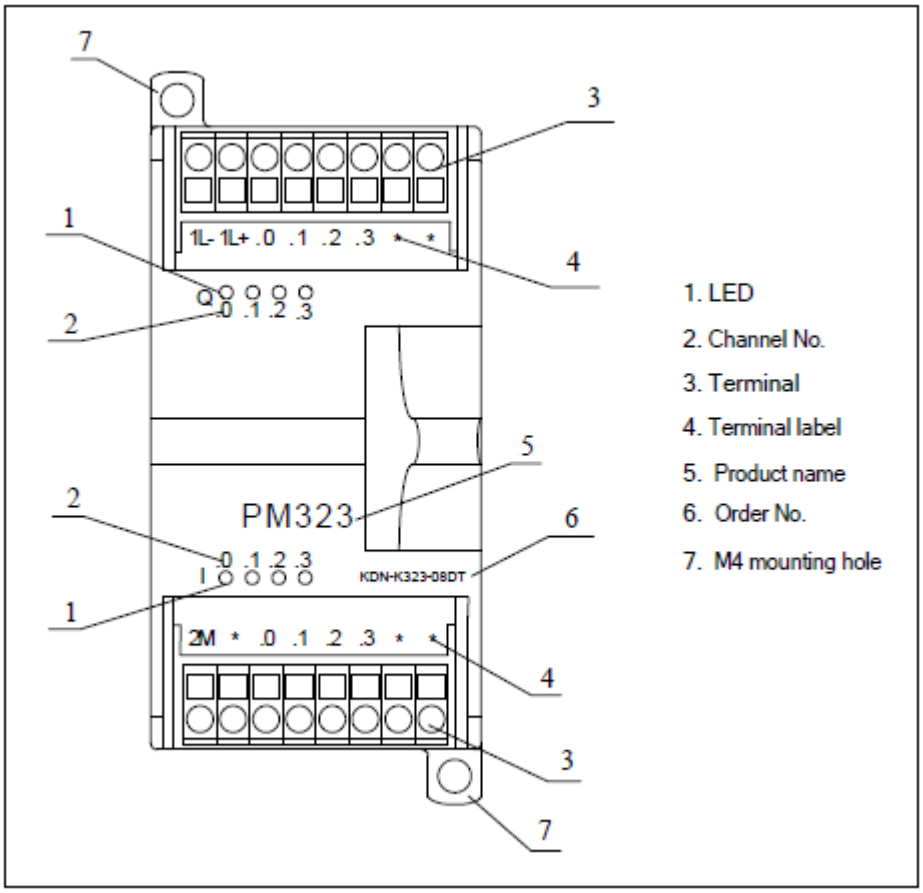
محافظت در برابر اتصال کوتاه (زمانی که جریان خروجی درهر گروه متجاوز از 3Aباشد)

اجازه اتصال موازی خروجی ها در یک گروه

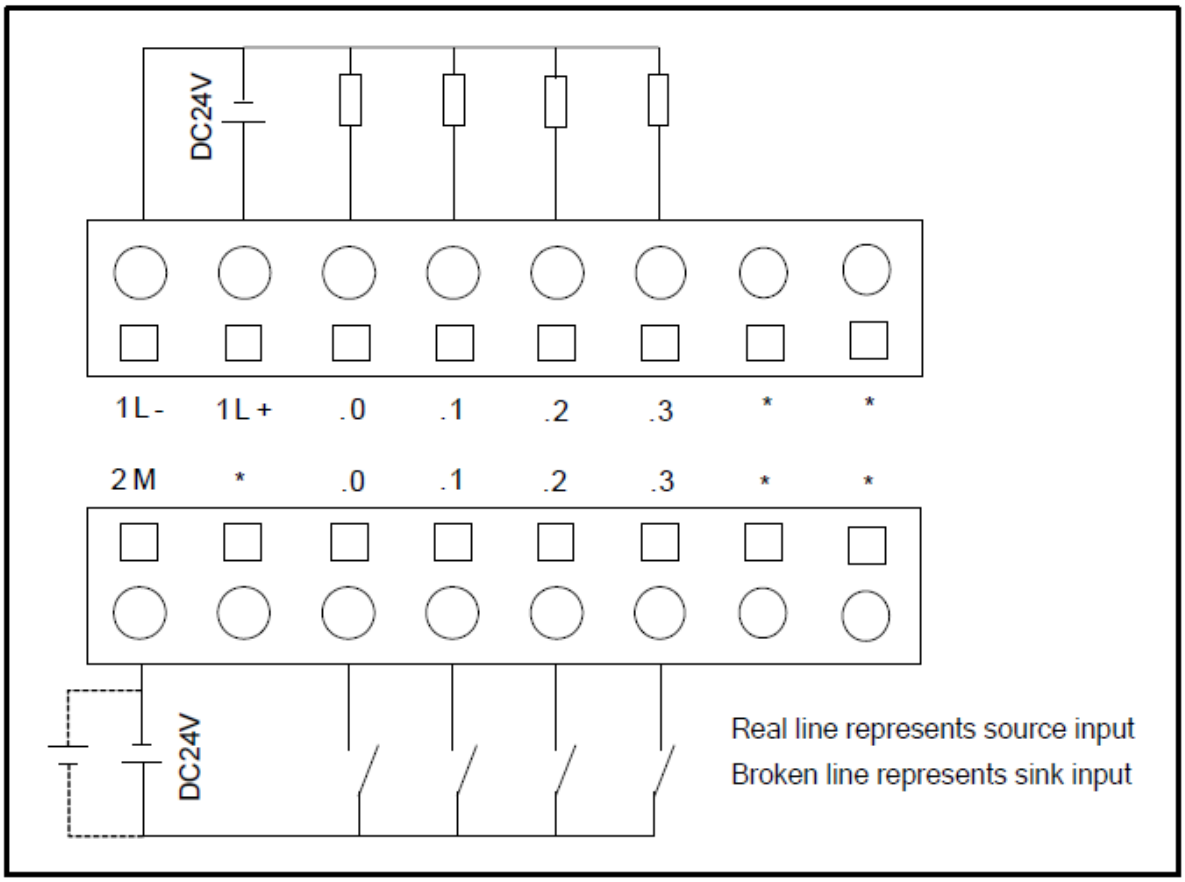
ایزولیشن opto-electricalبین سیگنال خروجی و مدار داخلی

طول ماژول 50mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:



دیاگرام سیم بندی این کارت به صورت زیر میباشد:



مشخصات فنی این ماژول مطابق با جدول زیر میباشد:

Electrical data	
Number of inputs	4 (4 channels/group)
Input type	Source/Sink
Rated input voltage	DC 24V (“1”, when DC15~30V)
Rated input current	4.1ma@24VDC
Max input voltage of logic 0	5V@0.7ma
Minimum input voltage of logic 1	15V@2.5ma
Input filter time delay	5ms
Isolation between input and internal circuit · Mode · Voltage	 Opto-electrical isolation 1500VAC/1 min

Number of output channels	4 (4 channels/group)
Output type	Source
Rated power supply voltage · Reverse polarity protection	DC 24V Yes
Rated output voltage	DC 24V
Output current per channel	Max 750ma@24VDC
Output leakage current	Max 0.5μA
Output impedance	Max 0.2Ω
Output delay · off-to-on · on-to-off	 0.3--5μs 5μs
Isolation between output and internal circuit · Mode · Voltage	 Opto-electrical isolation 1,500VAC/1 min
Inductive load protection	Yes
Short-circuit protection	Yes (when output current per group exceeds 3A)
Parallel connection of outputs	Yes (in the same group)

Current consumption via expansion bus	5V	< 65.8ma
	24V	-
Status indication	Green LED	
Address occupied		
DI image area	1 byte	
DO image area	1 byte	
Dimension and weight		
Dimension (L×W×H)	114×50×70mm	
Net weight	125g	

۳.۴.۲ DI4*DC24V , DO4 *Relay: DI/DO :

شماره مدل : K323-08DR

این ماژول دارای ۸ کانال میباشد که ۴ کانال آن به صورت ورودی ۲۴ ولت و ۴ کانال آن به صورت خروجی رله ای میباشد . هر کانال دارای یک LEDبرای نشان دادن وضعیت ورودی ها و خروجی ها میباشد.

مشخصات کلی:

درارای ۸ کانال میباشد که ۴ کانال به صورت ورودی دیجیتال ۲۴ ولت (به صورت یک گروه میباشند) و ۴ کانال به صورت خروجی دیجیتال رله ای (به صورت یک گروه میباشند) است

ورودی Source (کاتد مشترک) / sink(آند مشترک) به صورت اختیاری در هر گروه

ولتاژ ورودی 24VDC (ولتاژ موثر برای تحریک ورودی 15~30V میباشد)

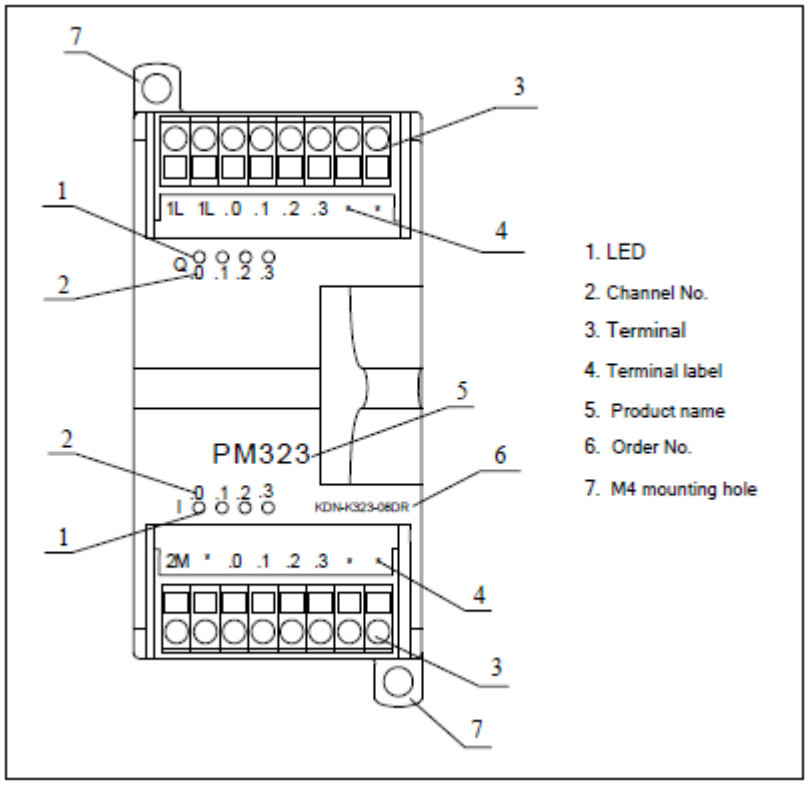
ایزولیشن opto-electricalبین سیگنال ورودی و مدار داخلی

ولتاژ خروجی DC30V/AC250V

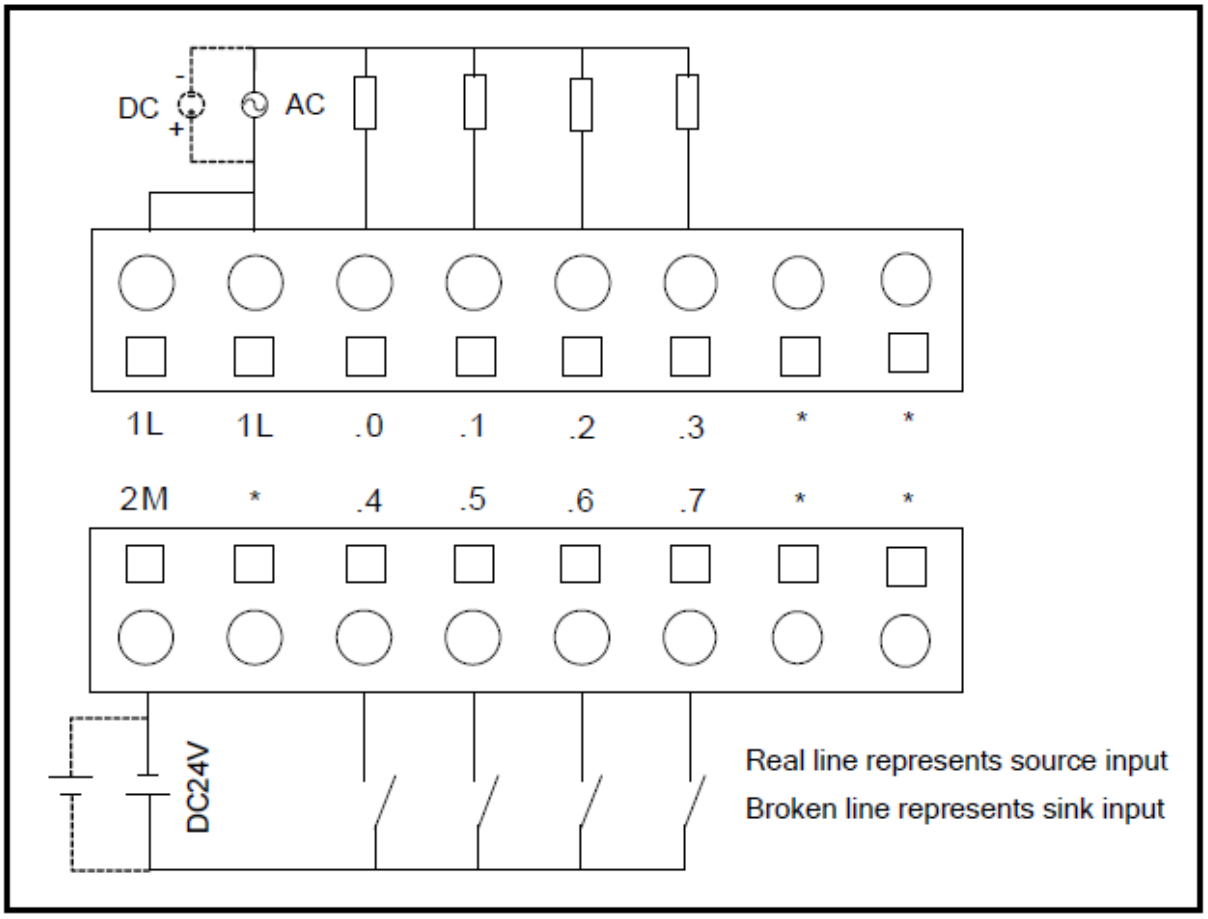
ماکزیمم جریان خروجی هر کانال 3A

طول ماژول 50mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:



دیاگرام سیم بندی این ماژول به صورت زیر میباشد:



مشخصات فنی این ماژول مطابق با جدول زیر میباشد:

Electrical Parameter		
Number of inputs	4 (4 channels/group)	
Input type	Source/Sink	
Rated input voltage	DC 24V (“1”, when DC15~30V)	
Rated input current	4.1ma@24VDC	
Max input voltage of logic 0	5V@0.7ma	
Minimum input voltage of logic 1	15V@2.5ma	
Input filter time delay	5ms	
Isolation between input and internal circuit · Mode · Voltage	 Opto-electrical isolation 1500VAC/1 min	
Number of relay outputs	4 (4 channels/group)	
Load voltage	Max DC 30V/AC250V	
Output current per channel	Max 3A (DC 30V/AC250V)	
Output current per group	Max 10A	
Output off-to-on delay	Max 10ms	
Output on-to-off delay	Max 5ms	
Max. Switching rate · No load · Rated load	 12,000 times/min 100 times/min	
Expected life of the contacts · Mechanical life (no-load) · Electrical life (rated load)	 20,000,000 times (1200 times/min) 100,000 times (6 times/min)	
Isolation · Mode · Between coil and contact · Between contacts	 Relay 2000Vrms 1000Vrms	
Current consumption via expansion bus	5V	< 67.4ma
	24V	< 22ma
Status indication	Green LED	

Address occupied	
DI image area	1 byte
DO image area	1 byte
Dimension and weight	
Dimension (L×W×H)	114×50×70mm
Net weight	145g

مشخصات ماژول های K323-16DT و K323-16DR مطابق با مطالب ارائه شده در بالا میباشد.

۳.۵: ماژول ورودی آنالوگ :

در این فصل دیاگرام سخت افزاری، دیاگرام سیم بندی و نیز اطلاعات فنی مربوط به ماژول های آنالوگی در PLC های KINCO-K3 توضیح داده میشود. تمامی ماژول های ورودی آنالوگ با نام PM331 شناخته میشوند.

۳.۵.۱: AI 4*IV (ورودی آنالوگی ۴ کاناله ، جریانی و ولتاژی) :

شماره مدل: K331-04IV

این ماژول دارای ۴ کانال آنالوگی برای اندازه گیری جریان وولتاژ میباشد. رنج اندازه گیری برای هر کانال به صورت : 1~5V ,-10~10V 4~20mA , 0~20mA, بوده و انتخاب جریانی و یا ولتاژی بودن و رنج اندازه گیری هر کانال بسته با سیگنال مورد نظر از طریق نرم افزار قابل انتخاب میباشد .روزلوشن ماژول های ورودی آنالوگی 16bit میباشد .

این ماژول در فضای آنالوگی AI ۸ بایت (برای هر کانال 2BYTE یا ۱۶ بیت را)در نظر میگیرد .سایر پارامترهای هر کانال از قبیل آدرس ، عملکرد و فیلتر و ... از طریق نرم افزار KincoBuilder قابل تنظیم میباشد . بر روی ماژول آنالوگی هرکانال دارای یک LEDقرمز رنگ میباشد .

توجه کنید که چنانچه از کانالی استفاده نمیشود ، باید پایه مثبت و منفی کانال ها اتصال کوتاه شوند.

مشخصات کلی:

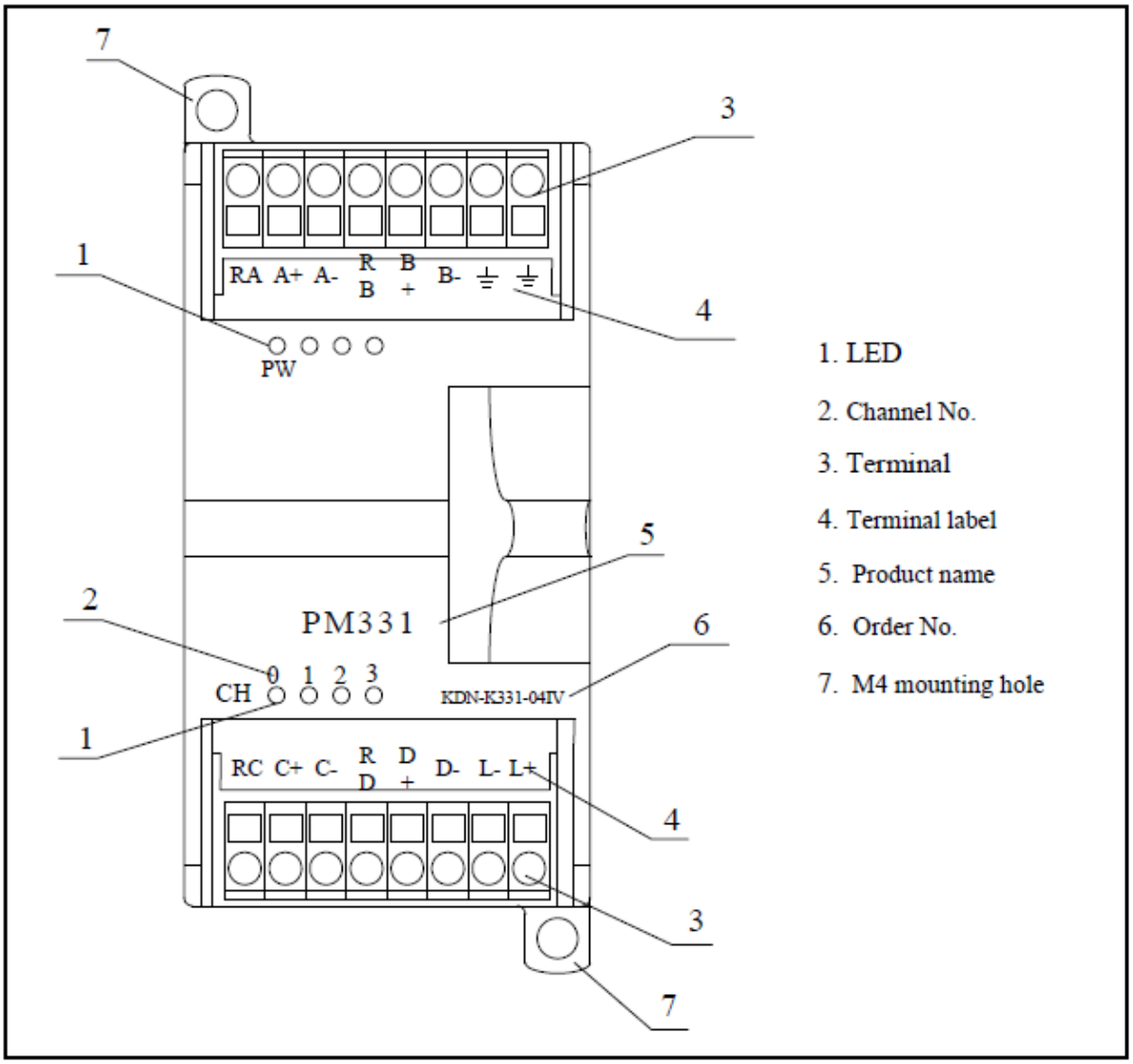
دارای ۴ کانال با قابلیت اندازه گیری سیگنال های 4~20mA ,1~5V, 0~20mA ,-10~10V

تنظیم پارامترهای هر کانال از طریق نرم افزار

دارای LEDقرمز رنگ برای مشخص شدن وضعیت هر کانال

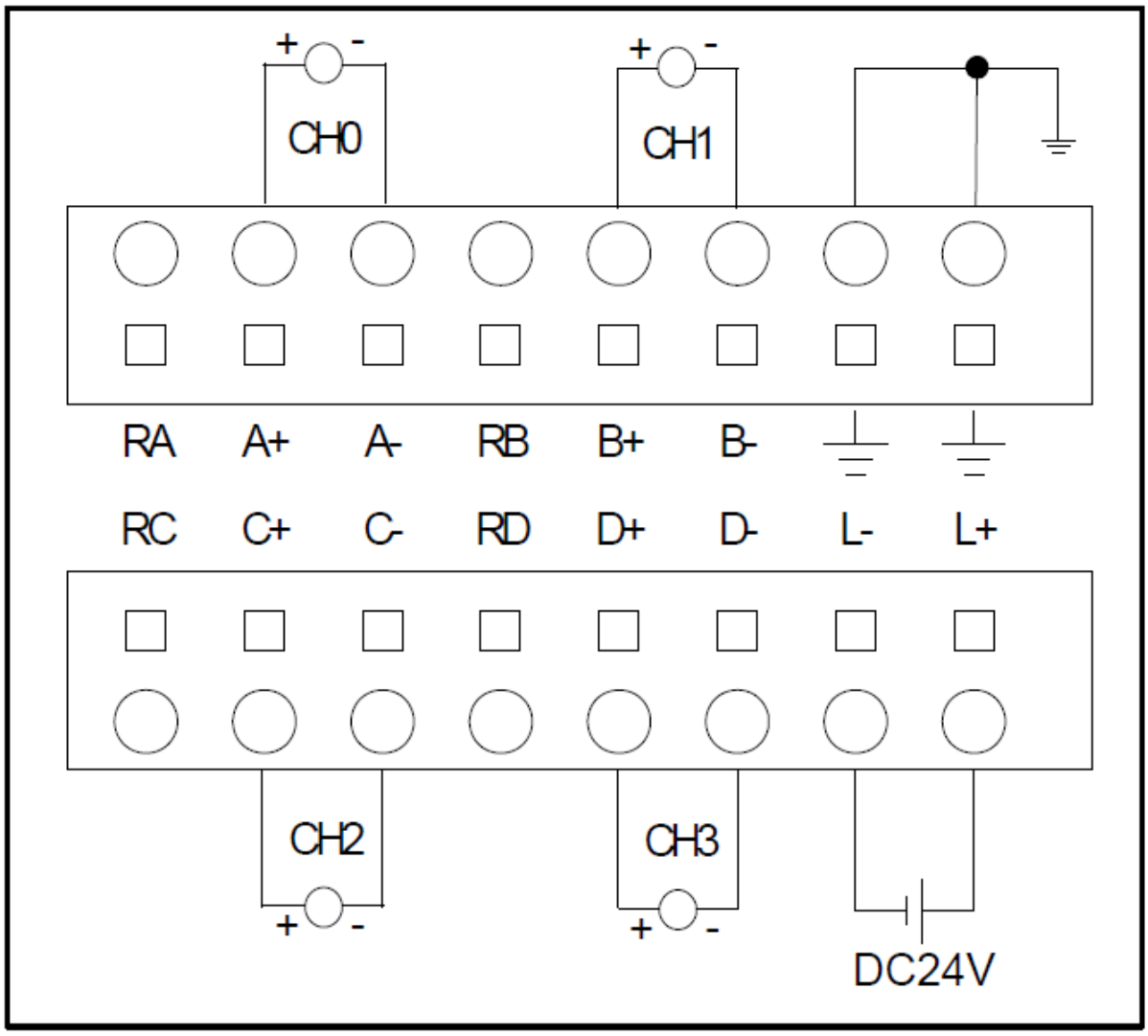
طول ماژول 50mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:

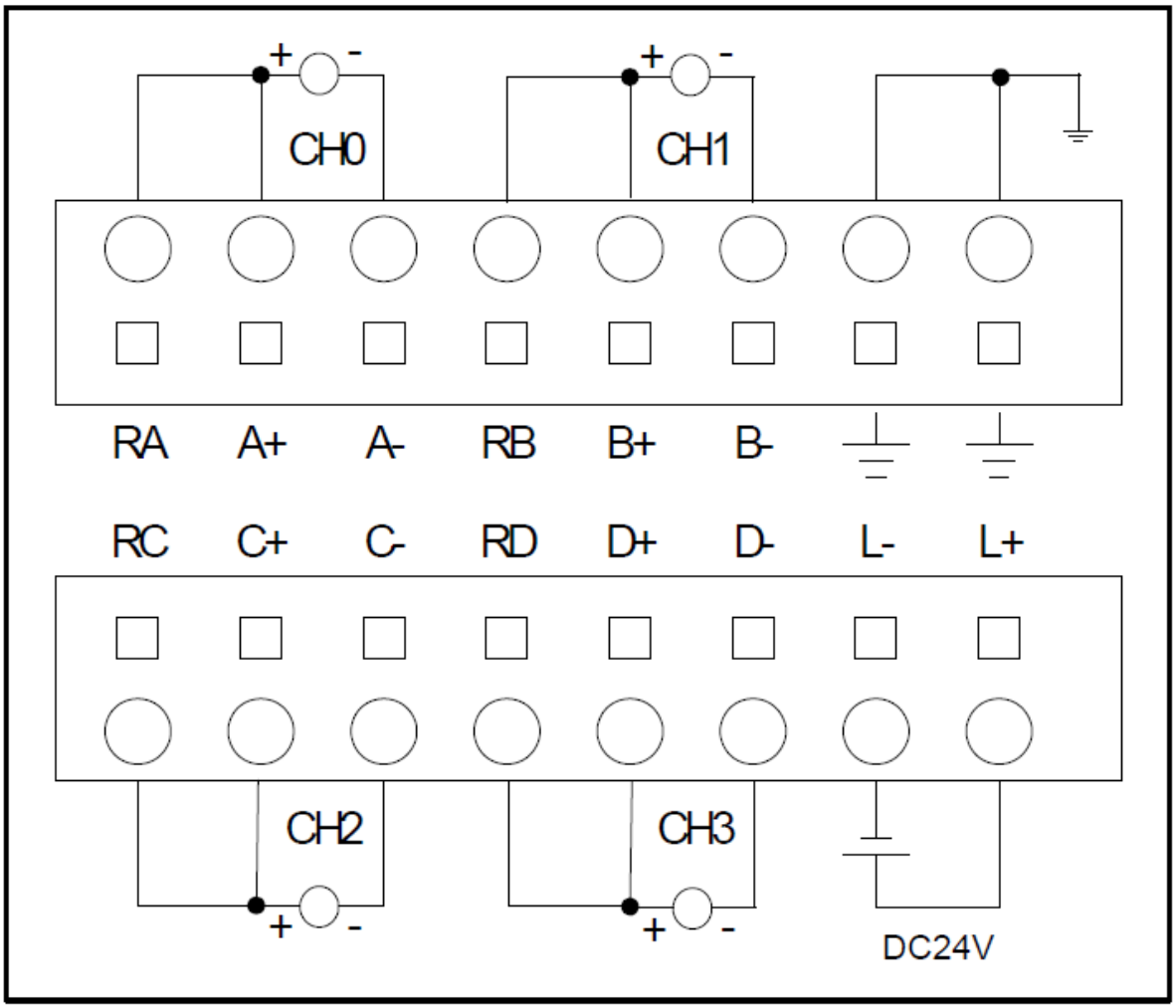


دیاگرام سیم بندی :

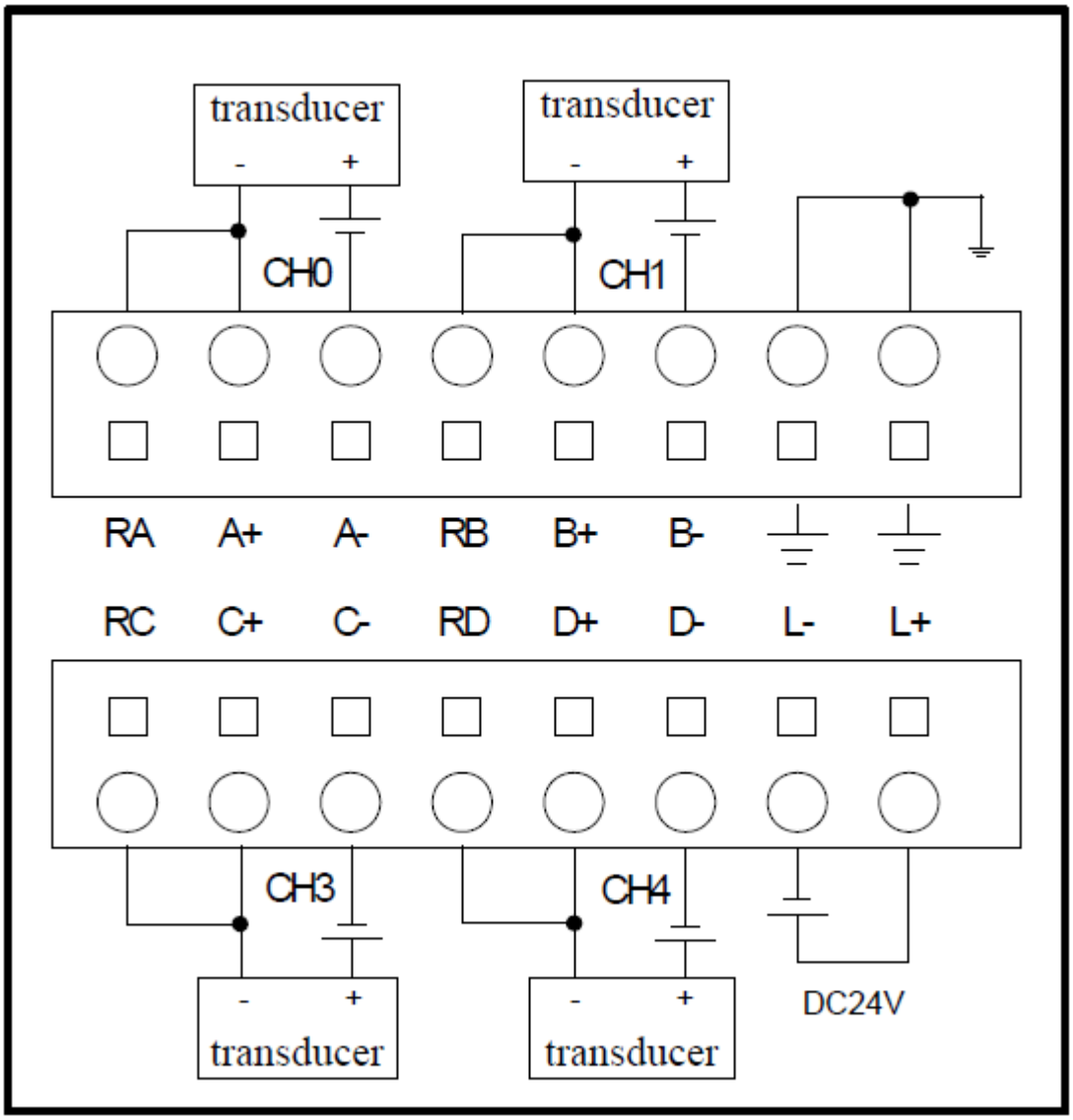
چنانچه سیگنال مورد نظر ولتاژی باشد (کانال ها به صورت ولتاژی تنظیم شده است) :



چنانچه سیگنال مورد نظر جریانی باشد (کانال ها به صورت جریانی تنظیم شده است) و ترنسیمتر مورد استفاده سه سیمه باشد :



چنانچه سیگنال مورد نظر جریانی باشد (کانال ها به صورت جریانی تنظیم شده است) و ترنسیمتر مورد استفاده دو سیمه باشد :



رنج اندازه گیری و مقادیر اندازه گیری شده توسط ماژول آنالوگی به صورت زیر میباشد:

Measurement Type	Measurement Range	Measured value	Remark
4~20ma(1)	0~20.4ma(3)	I×1000	If input signal exceeds the upper limit of measuring range, the measured value will be kept at 32767. If input signal exceeds the lower limit of measuring range, the measured value will be kept at -32767.
1~5V(2)	-10.2~10.2V(3)	V×1000	
0~20ma	0~20.4ma(3)	I×1000	
-10~10V	-10.2~10.2V(3)	V×1000	

همان طور که در جدول بالا ملاحظه میکنید چنانچه سیگنال به صورت جریانی و یا ولتاژی باشد ، مقدار اندازه گیری شده توسط ماژول آنالوگی در عدد ۱۰۰۰ ضرب شده و در فضای حافظه آنالوگی ذخیره میشود .

توجه شود که چنانچه سیگنال مورد نظر بیشتر از رنج های استاندارد (که در جدول بالا ارائه شده است) باشد ، ممکن است ماژول آنالوگی آسیب ببیند.

مشخصات فنی این ماژول مطابق با جدول زیر میباشد:

Electrical data		
Number of channels	4	
Measurement types	4~20ma, 1~5V, 0~20ma, ±10V	
Rated power supply	DC 24V, >=75ma	
Resolution (including sign)	16 bits	
Measurement accuracy	0.2% F.S.	
Conversion rate (per channel)	About 15 times/s	
Input impedance	Current mode: <250Ω Voltage mode: >4MΩ	
Current consumption via expansion bus	5V	< 44.2ma
	24V	-
Status indication	Red LED	
Address occupied		
AI image area	8 bytes (2 bytes per channel)	
AO image area	-	
Dimension and weight		
Dimension (L×W×H)	114×50×70mm	
Net weight	136g	

AI4*RD:۳.۵.۲ (کارت آنالوگی ورودی ۴ کاناله باورودی RTD):

شماره مدل : K331-04RD

این ماژول دارای ۴ کانال آنالوگ جهت اندازه گیری دما میباشد . هنگامی که از سنسورهای RTD برای اندازه گیری دما استفاده شود ، میتوان به صورت مستقیم از این ماژول استفاده نمود. این ماژول دو سنسور PT100و CU50 را پشتیبانی میکند .نوع سنسور برای هر کانال از طریق نرم افزار قابل تنظیم میشود .رزولوشن این ماژول ۱۶ بیت میباشد .

این ماژول در فضای آنالوگی AI، ۸ بایت (برای هر کانال 2BYTE یا ۱۶ بیت)را در نظر میگیرد .سایر پارامترهای هر کانال از قبیل آدرس ، عملکرد و فیلتر و ... از طریق نرم افزار KincoBuilder قابل تنظیم میباشد . بر روی ماژول آنالوگی هرکانال دارای یک LEDقرمز رنگ میباشد .

توجه کنید که کانال های استفاده نشده باید اتصال کوتاه شوند .

مشخصات کلی:

دارای ۴ کانال RTD(PT100 ,CU50) .ورودی میتواند به صورت ۲ سیمه و یا ۳ سیمه باشد .

رنج اندازه گیری در هر نوع سنسور به صورت زیر میباشد :

درجه سانتی گراد PT100 : -150~800

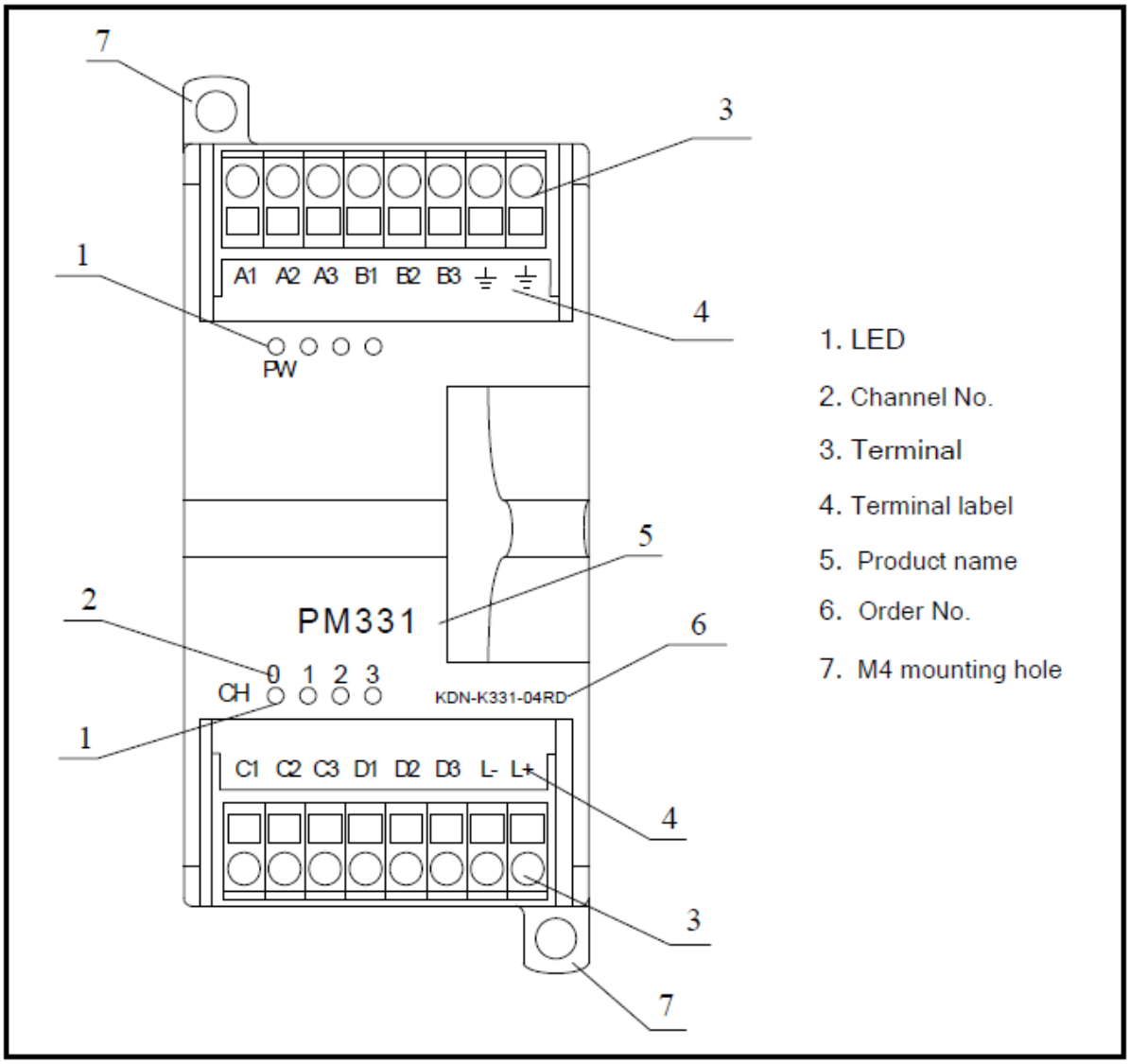
درجه سانتی گراد CU50: -50~150

تنظیم پارامترهای هر کانال از طریق نرم افزار

دارای LEDقرمز رنگ برای مشخص شدن وضعیت هر کانال

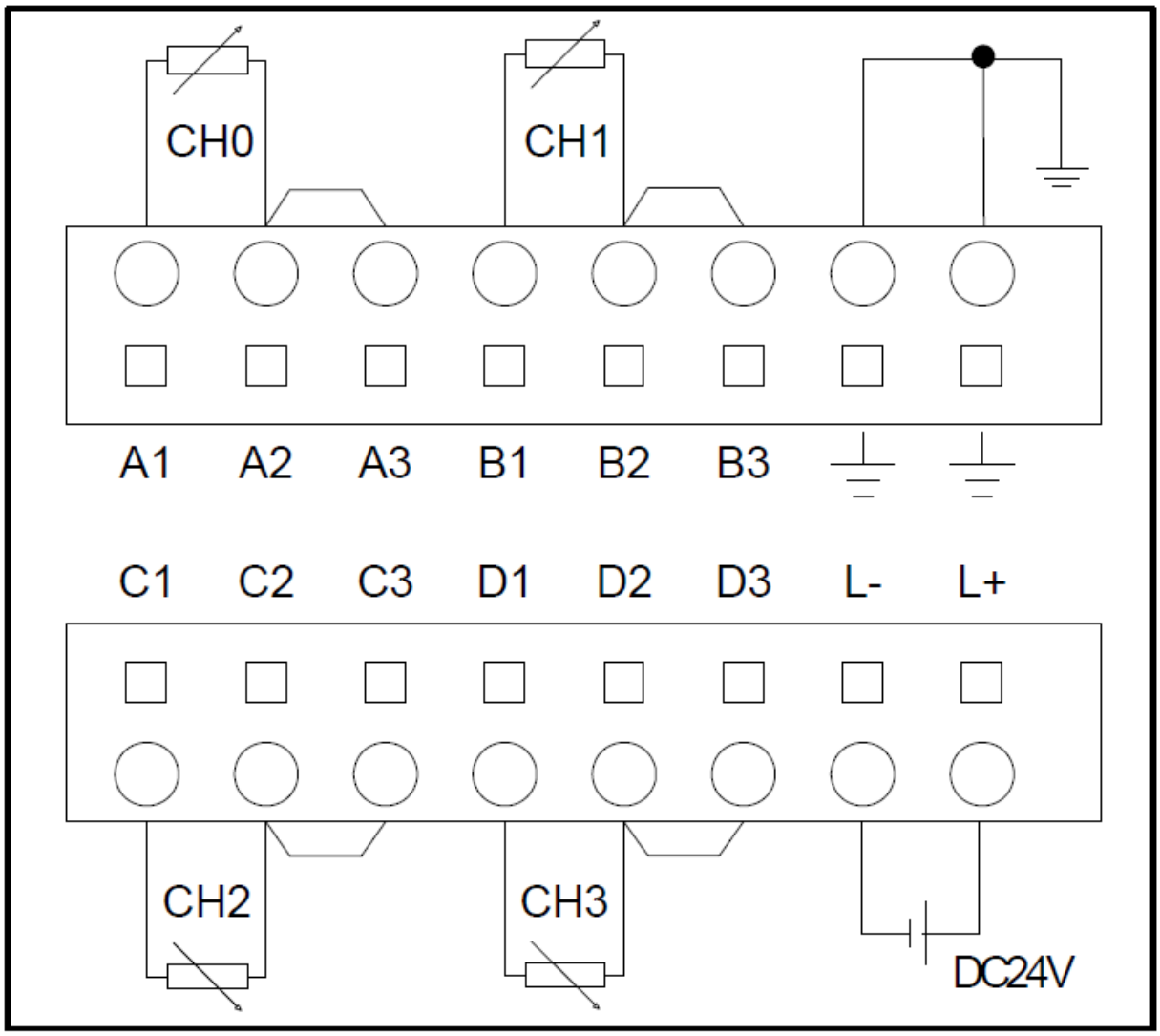
طول ماژول 50mm

سخت افزار ماژول ورودی مانند تصویر زیر میباشد:

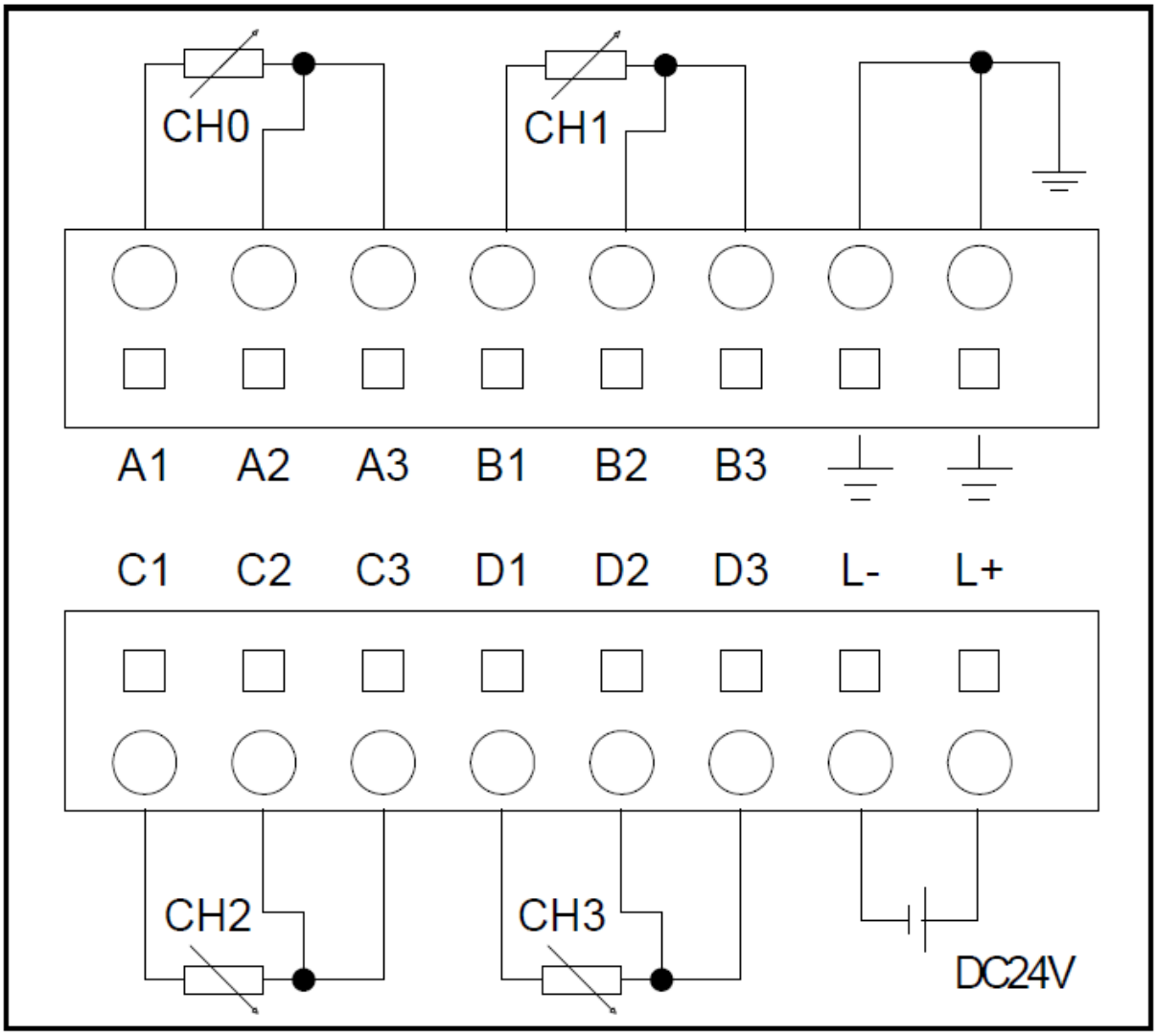


دیاگرام سیم بندی :

چنانچه اتصال به صورت ۲سیمه باشد :



چنانچه اتصال به صورت ۳سیمه باشد :



رنج اندازه گیری و مقادیر اندازه گیری شده توسط ماژول آنالوگی به صورت زیر میباشد:

Measurement Type	Measurement Range	Measured value	Remark
Pt100	-150~800°C	T×10	If input signal exceeds the upper limit of measuring range, the measured value will be kept at 32767. If input signal exceeds the lower limit of measuring range, the measured value will be kept at -32767. As long as the input signal overruns the measuring range, the red LED will light.
Cu50	-50~150°C	T×10	

همان طور که در جدول بالا ملاحظه میکنید زمانی که سیگنال توسط این ماژول اندازه گیری شود ، مقدار اندازه گیری شده توسط ماژول آنالوگی ، در عدد ۱۰ ضرب شده و در فضای حافظه آنالوگی ذخیره میشود .

مشخصات فنی این ماژول مطابق با جدول زیر میباشد:

Electrical data		
Number of channels	4	
Measurement types	Pt100: -150~800°C Cu50: -50~150°C	
Connection	2-wire or 3-wire	
Rated power supply	DC 24V, >=75ma	
Resolution (including sign)	16 bits	
Measurement accuracy	0.1% F.S.	
Conversion rate (per channel)	About 15 times/s	
Input impedance	>1MΩ	
Current consumption via expansion bus	5V	< 49.4ma
	24V	-
Status indication	Red LED	
Address occupied		
AI image area	8 bytes (2 bytes per channel)	
AO image area	-	
Dimension and weight		
Dimension (L×W×H)	114×50×70mm	
Net weight	132g	

۳.۶:ماژول خروجی آنالوگ :

در این فصل دیاگرام سخت افزاری، دیاگرام سیم بندی و نیز اطلاعات فنی مربوط به ماژول خروجی آنالوگی در PLC های KINCO-K3 توضیح داده میشود. ماژول خروجی آنالوگ با نام PM332شناخته میشود.

۳.۶.۱ AO 2*IV :

شماره مدل : K332-02IV

این ماژول دارای دو کانال خروجی آنالوگی بوده رنج خروجی این کانال ها به صورت 4~20mA ,0~20mA ,-10V~10V ,1~50V میباشد. چنانچه مقدار خروجی مشخص شده در برنامه از محدوده حد بالا /پایین خروجی تجاوز کند ، مقدار نهایی خروجی (به منظور جلوگیری از آسیب زده شدن به تجهیزات نصب شده به این خروجی ها) به صورت ثابت در حد بالا /پایین مجاز این خروجی باقی میماند.خروجی این کانال ها دارای رزولوشن ۱۲ بیت میباشد.

ماژول خروجی آنالوگ در فضای آنالوگی ۴ بایت (۲ بایت برای هر کانال) در نظر میگیرد. سایر پارامترهای هر کانال از قبیل آدرس ، عملکرد و... از طریق نرم افزار KincoBuilder قابل تنظیم میباشد .

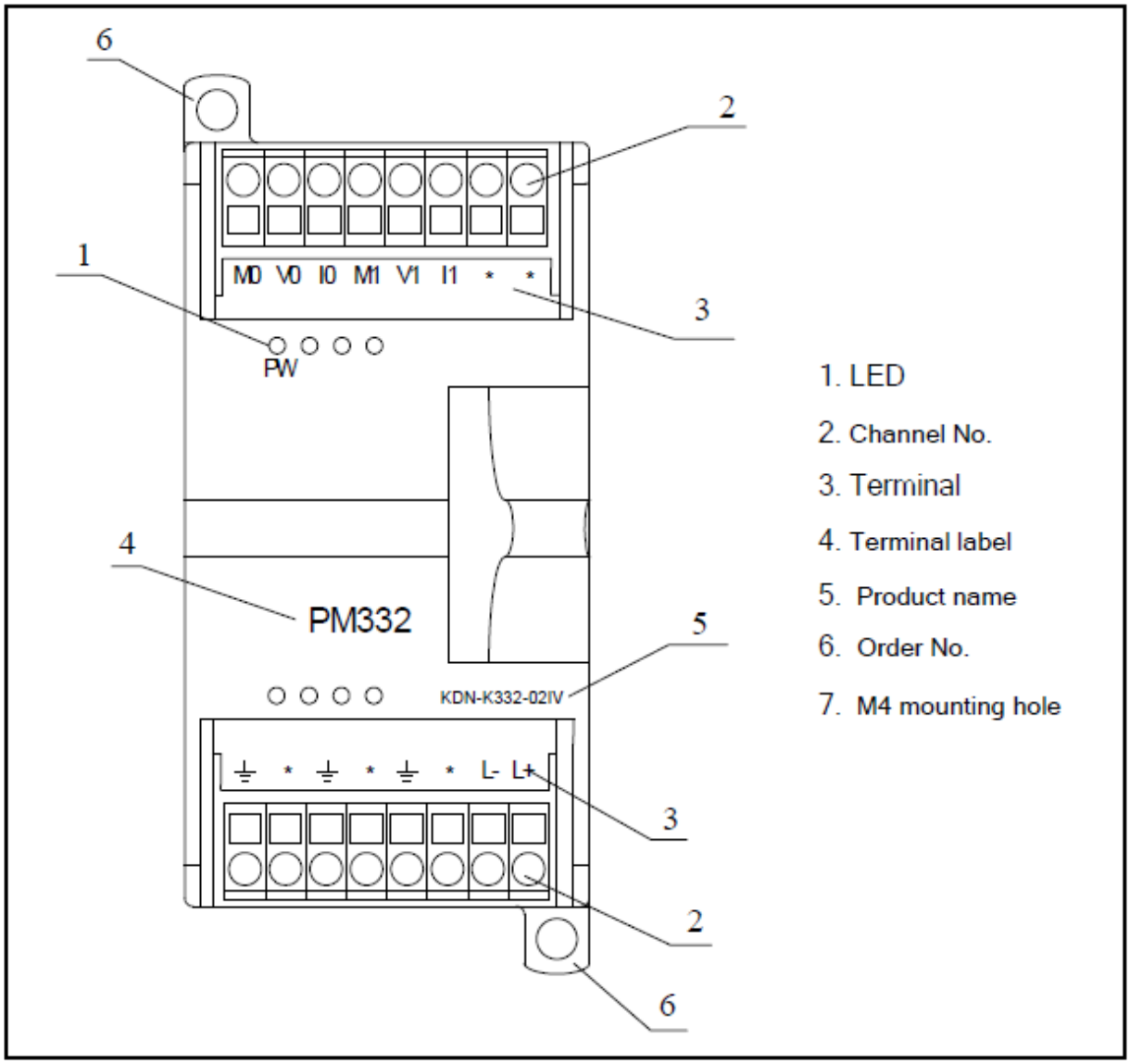
مشخصات کلی :

دارای دو کانال خروجی : 4~20mA , 1~5V , 0~20mA , -10~10V

تنظیم پارامترهای مربوط به هر کانال از طریق نرم افزار KincoBuilder

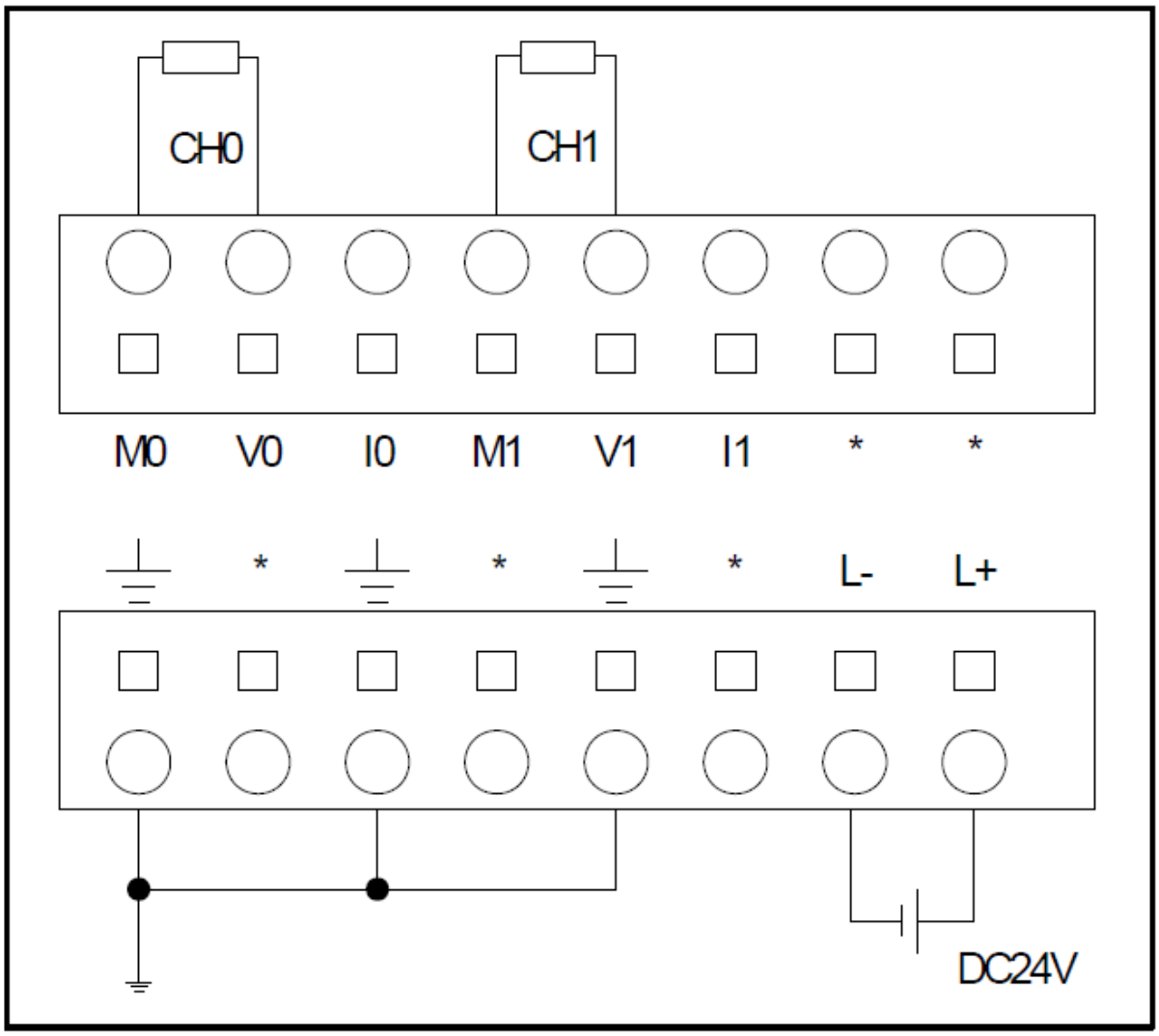
طول ماژول 50mm

سخت افزار ماژول خروجی مانند تصویر زیر میباشد:

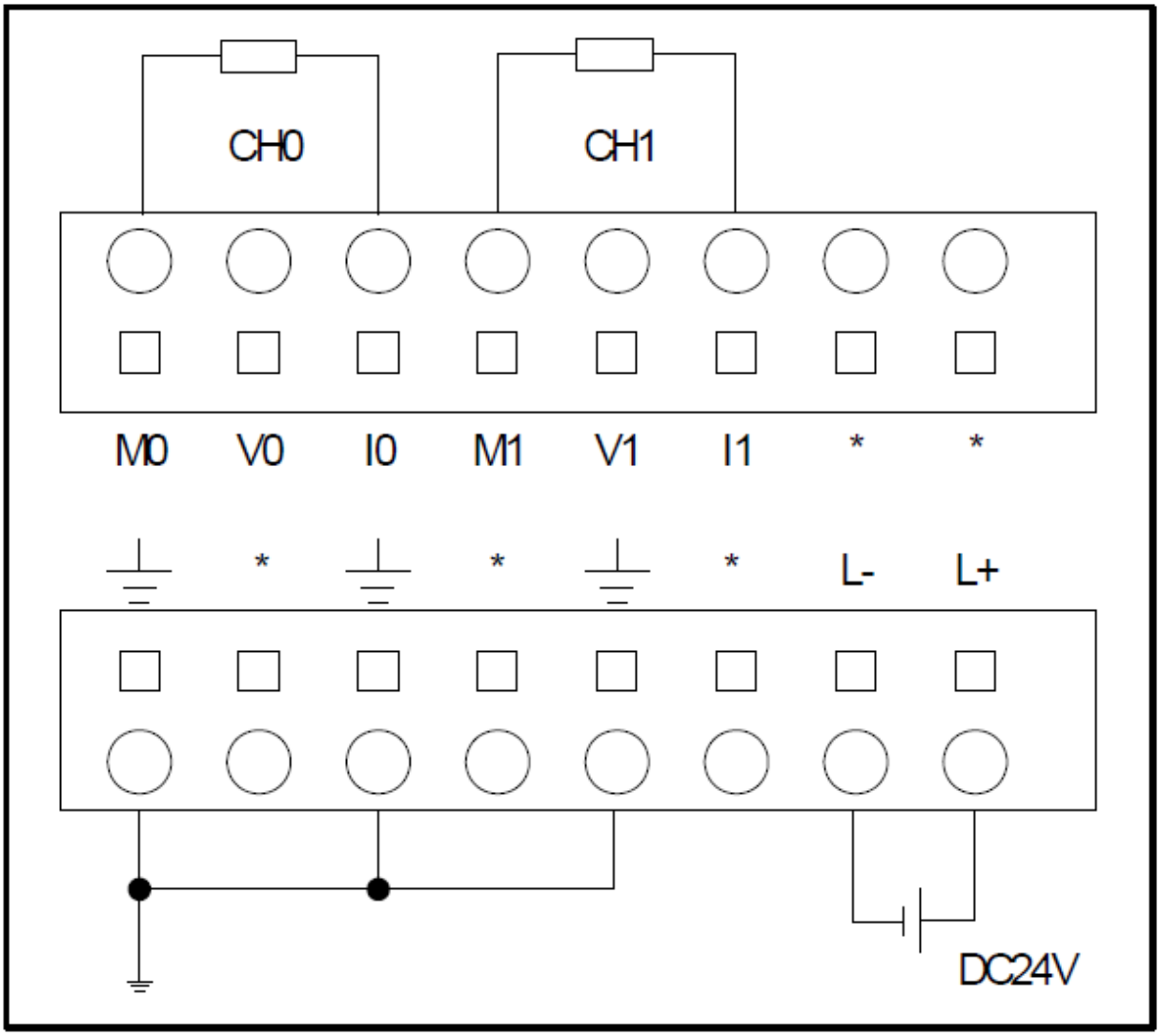


دیاگرام سیم بندی:

خروجی ولتاژی :



خروجی جریانی:



رنج خروجی و مقادیر خروجی ارائه شده :

همان طور که در کارت های ورودی آنالوگی توضیح داده شد ، سیگنال های آنالوگی که توسط کارت ورودی آنالوگ خوانده میشوند ،چنانچه از نوع جریانی / ولتاژی باشند در عدد ۱۰۰۰ و چنانچه از نوع RTD باشند در عدد ۱۰ ضرب شده و تحویل فضای حافظه آنالوگی میگردد.

به همین ترتیب زمانی که تمامی محاسبات انجام شد م و بخواهیم جریان / ولتاژی را در پایه خروجی آنالوگ (در ماژول افزایشی) داشته باشیم ، ابتدا مقدار مورد نظر را در عدد ۱۰۰۰ ضرب نموده و د رفضای حافظه خروجی آنالوگی(AQ) ذخیره کنید . CPUبه صورت اتوماتیک این مقدار را تقسیم بر ۱۰۰۰ نموده و تحویل پایه فیزیکی خروجی میدهد.به جدول زیر توجه نمایید:

Output Signal	Output Range	Output Value Representation	Remark
4~20ma	0~20.0ma	I×1000	If the output value specified in the user program exceeds the upper/lower limit of the output range, the actual output value will be kept at the upper/lower limit.
1~5V	0~5.1V	V×1000	
0~20ma	0~20.0ma	I×1000	
-10~10V	-10.2~10.2V	V×1000	

به عنوان مثال زمانی که بخواهیم ولتاژ ۲ ولت را در کانال ۰ خروجی آنالوگ داشته باشید، باید AQW0 =2000قرار دهید.

مشخصات فنی این ماژول مطابق با جدول زیر میباشد:

Electrical data		
Number of outputs	2	
Output signal	4~20ma, 1~5V, 0~20ma, ±10V	
Rated power supply	DC 24V	
Resolution (including sign)	12 bits	
Output Accuracy	0.5% F.S.	
Resistance load	Current mode: max. 500Ω Voltage mode: min. 1kΩ	
Current consumption via expansion bus	5V	< 41.9ma
	24V	-
Address occupied		
AI image area	-	
AO image area	4 bytes (2 bytes per channel)	
Dimension and weight		
Dimension (L×W×H)	114×50×70mm	
Net weight	125g	

۳.۷: ماژول ورودی / خروجی آنالوگ :

در این فصل دیاگرام سخت افزاری، دیاگرام سیم بندی و نیز اطلاعات فنی مربوط به ماژول ورودی/ خروجی آنالوگی در PLC های KINCO-K3 توضیح داده میشود. ماژول های ورودی /خروجی آنالوگ با نام PM333شناخته میشوند.

۳.۷.۱: AI2*IV , AO1*IV (ماژول ورودی/ خروجی آنالوگ):

شماره مدل: K333-03IV

این ماژول دارای دو کانال ورودی آنالوگ جریان / ولتاژ و نیز یک کانال خروجی آنالوگ جریان / ولتاژمیباشد. رنج این کانال ها هم در ورودی و هم در خروجی به صورت 4~20mA , 0~20mA , 1~5V ,-1~10Vمیباشد.

رزولوشن کانال های ورودی ۱۶ بیتی بوده و هر کانال دارای یک LEDقرمز رنگ برای نشان دادن وضعیت آنهاست .

در کانال خروجی چنانچه مقدار خروجی مشخص شده در برنامه از محدوده حد بالا /پایین خروجی تجاوز کند ، مقدار نهایی خروجی (به منظور جلوگیری از آسیب زده شدن به تجهیزات نصب شده به این خروجی ها) به صورت ثابت در حد بالا /پایین مجاز این خروجی باقی میماند.خروجی این کانال ها دارای رزولوشن ۱۲ بیت میباشد.

این ماژول در فضای حافظه ورودی آنالوگی ۴ بایت (۲ بایت برای هر کانال) و در فضای حافظه خروجی آنالوگی ۲ بایت در نظر میگیرد. سایر پارامترهای هر کانال از قبیل آدرس ، عملکرد ، فیلتر و... از طریق نرم افزار KincoBuilder قابل تنظیم میباشد .

توجه کنید که چنانچه از کانالی استفاده نمیشود ، باید پایه مثبت و منفی کانال ها اتصال کوتاه شوند.

مشخصات کلی :

دارای دو کانال ورودی آنالوگی در رنج 4~20mA , 0~20mA ,1~5V,-10~10V

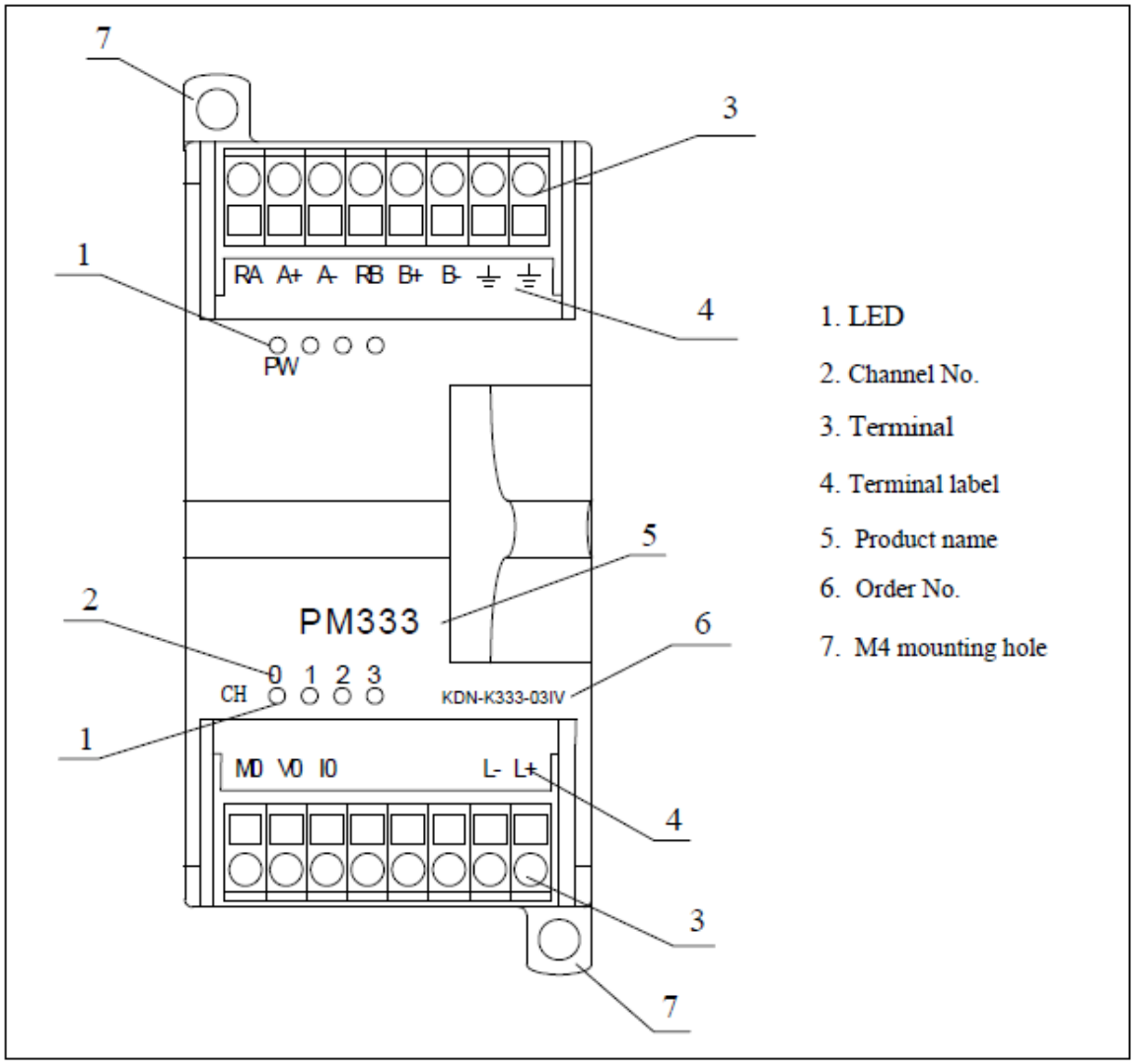
دارای LEDقرمز رنگ برای کانال های ورودی

دارای یک کانال خروجی در رنج 4~20mA , 0~20mA ,1~5V,-10~10V

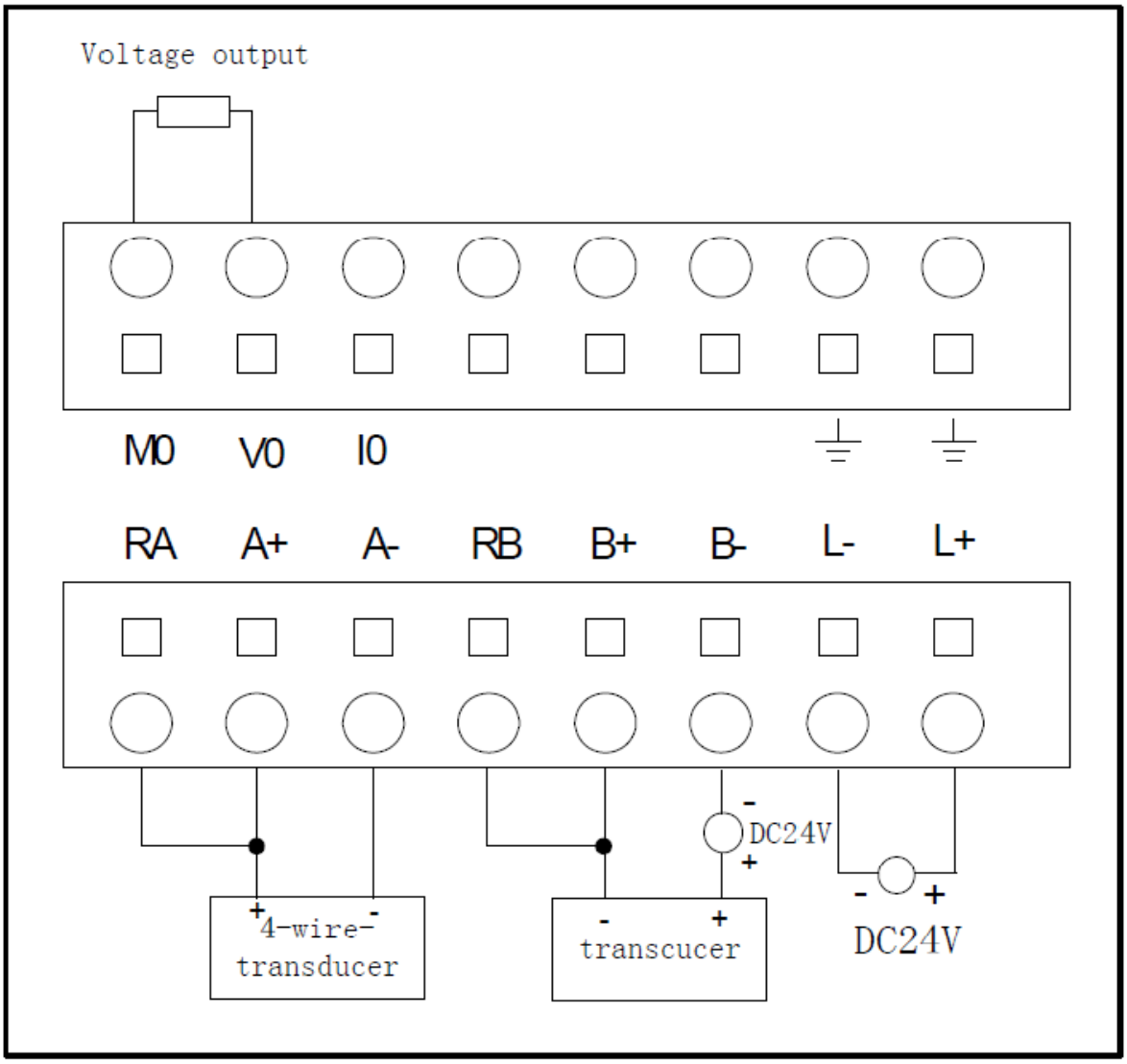
تنظیم پارامترهای مربوط به هر کانال از طریق نرم افزار KincoBuilder

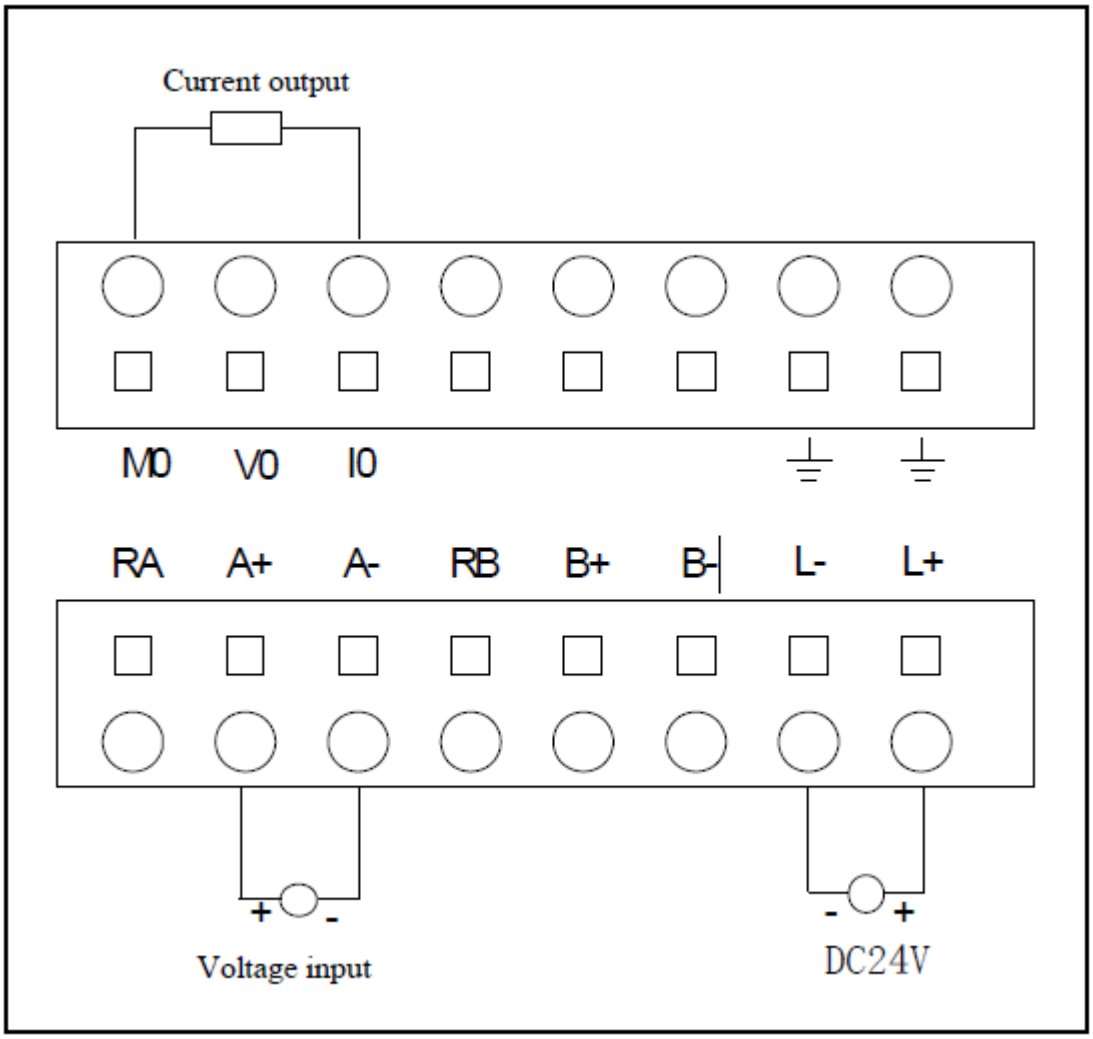
طول ماژول 50mm

سخت افزار این ماژول مانند تصویر زیر میباشد:



دیاگرام سیم بندی :


Voltage output
M0
V0
I0
RA
A+
A-
RB
B+
B-
L-
L+
DC24V
4-wire
transducer
transcucer
DC24V



مقادیر اندازه گیری شده و ارائه آنها در کانالهای ورودی مطابق با جدول زیر میباشد:

Measurement Type	Measurement Range	Measured value	Remark
4~20ma(1)	0~20.4ma(3)	I×1000	If input signal exceeds the upper limit of measuring range, the measured value will be kept at 32767.
1~5V(2)	-10.2~10.2V(3)	V×1000	
0~20ma	0~20.4ma(3)	I×1000	
-10~10V	-10.2~10.2V(3)	V×1000	If input signal exceeds the lower limit of measuring range, the measured value will be kept at -32767.

رنج خروجی و مقدار ارائه شده خروجی مطابق با جدول زیر میباشد:

Output Signal	Output Range	Output Value Representation	Remark
4~20ma	0~20.0ma	I×1000	If the output value specified in the user program exceeds the upper/lower limit of the output range, the actual output value will be kept at the upper/lower limit.
1~5V	0~5.1V	V×1000	
0~20ma	0~20.0ma	I×1000	
-10~10V	-10.2~10.2V	V×1000	

مشخصات فنی این ماژول مطابق با جدول زیر میباشد:

AI2*IV , AO2*IV:۳.۷.۲ (ماژول ورودی/ خروجی آنالوگ):

شماره مدل: K333-04IV

تمامی موارد ارائه شده در بالا در مورد این ماژول نیز صادق میباشد.

Electrical data		
Number of AI channels	2	
Measurement types	4~20ma, 1~5V, 0~20ma, ±10V	
Rated power supply	DC 24V, >=75ma	
Resolution (including sign)	16 bits	
Measurement accuracy	0.2% F.S.	
Conversion rate (per channel)	About 15 times/s	
Input impedance	Current mode: <250Ω Voltage mode: >4MΩ	
Status indication	Red LED	
Number of AO outputs	1	
Output signal	4~20ma, 1~5V, 0~20ma, ±10V	
Rated power supply	DC 24V	
Resolution (including sign)	12 bits	
Output Accuracy	0.5% F.S.	
Resistance load	Current mode: max. 500Ω Voltage mode: min. 1kΩ	
Current consumption via expansion bus	5V	< 44.2ma
	24V	-
Address occupied		
AI image area	4 bytes (2 bytes per channel)	
AO image area	2 bytes (2 bytes per channel)	
Dimension and weight		
Dimension (L×W×H)	114×50×70mm	
Net weight	136g	

فصل چهارم : آشنایی با نرم افزار Kinco Builder:

مقدمه:

IEC61131-3 تنها استاندارد جهانی برای برنامه نویسی در زمینه کنترل صنعتی میباشد و این استاندارد توسط بسیار ی از تولید کنندگان PLC پذیرفته شد ه است.

KincoBuilderنرم افزار برنامه نویسی PLC های کارخانه KINCO میباشد که کاربران به راحتی میتوانند با این نرم افزار ارتباط برقرار کرده و از آن استفاده نمایند .این نرم افزار به واسطه داشتن Function های قوی دارای کارایی بالایی میباشد. نرم افزار Kinco Builderنرم افزار ی مستقل و مطابق با استاندارد IEC61131-3امیباشد

نرم افزار Kinco Builder ازمشخصات زیر برخوردار میباشد:

مطابق با استاندارد IEC61131-3امیباشد.

دو زبان برنامه نویسی استاندارد را پشتیبانی میکند: زبان برنامه نویسی IL (Instruction List) واو زبان برنامه نویسی LD (Ladder)

برخورداری از دستورات Function ها و Function Blockهای قوی

پشتیبانی از برنامه نویسی ساختار یافته

پشتیبانی از وقفه ها و زیر برنامه های مختلف

پشتیبانی از متغیرها و نیز متغیر های سمبلیک

دارای محیط نرم افزاری کارآمد و کاربری آسان برای کاربران

پیکربندی سخت افزاری آسان که کاربر به راحتی میتواند پارامترهای مختلف سخت افزاری را از طریق نرم افزار تعیین نماید.

برای آشنایی بیشتر با چگونگی برنامه نویسی در نرم افزار KINCO Builder ابتدا باید با مفاهیم زیر بیشتر آشنا شویم :

۴.۱ مفاهیم :

۴.۱.۱ POU (Program Orgnization Unit)

در استاندارد IEC61131-3 ، بلوک هایی که در هر برنامه یا پروژه ای که ساخته میشوند ، POUنامیده میشوند.
POUدر اصل کوچک ترین بخش و در عین حال بخش مستقل در نرم افزار میباشد که شامل کدهای برنامه است . سه نوع POUدر استاندارد IEC61131-3 مشخص شده است :

:Program ۴.۱.۲

این نوع POUبیان گر mainبرنامه میباشد (main program) که در کنترلرها اجرا میشود. پارامترهای Programme میتواند هم شامل ورودی وهم شامل خروجی باشد.

:FUNCTION:۴.۱.۳

Function ها هم شامل ورودی میشوند وهم یک مقدار را به برنامه برمیگردانند. در واقع هر فانکشن دارای یک عملکرد خاص میباشد.فانکشن ها فاقد حافظه هستند .هر فانکشن دارای یک پایه فعال ساز(EN) میباشد. مانند move

:Function Block:۴.۱.۴

فانکشن بلاک ها دارای پارامترهای ورودی و خروجی و همچنین دارای متغیرهای استاتیکی میباشند که این متغیرهای استاتیکی میتواند وضعیت قبل را نگه دارد . تفاوت فانکشن بلاک ها با فانکشن ها در حافظه آنها میباشد . از جمله فانکشن بلاک ها میتوان تایمرها ،کانترها ،فلیپ فلاپها را نام برد .

۴.۲: انواع داده ها :

در PLCهای KINCOمانند سایر PLC ها داده ها انواع مختلف دارند که

هرد اده بنا بر نوع خود (که در ادامه به آن اشاره میشود) میتواند حجم مختلفی از فضاهای حافظه را اشغال نماید .علاوه براین انواع داده ها میتواند رنج های مختلفی از اعداد را در برگیرد . بنا براین تمامی متغیرهایی که در برنامه تعریف میشود باید یکی از انواع زیر باشد . انواع داده هایی که KINCO-K3 پشتیبانی مینماید مطابق با جدول زیر میباشد.

به جدول زیر توجه نمایید :

Keyword	Description	Size in Bits	Range of Values	Default Initial Value
BOOL	Boolean	1	true, false	false
BYTE	Bit string of length 8	8	0 ~ 255	0
WORD	Bit string of length 16	16	0 ~ 65,535	0
DWORD	Bit string of length 32	32	0 ~ 4,294,967,295	0
INT	Signed integer	16	$-2^{15} \sim (2^{15}-1)$	0
DINT	Signed Double integer	32	$-2^{31} \sim (2^{31}-1)$	0
REAL	Floating-point number, ANSI/IEEE 754--1985 standard format	32	$1.18*10^{-38} \sim 3.40*10^{38}$, $-3.40*10^{38} \sim -1.18*10^{-38}$	0.0

این انواع داده مطابق با استاندارد IEC61131-3 میباشد .

۴.۲.۲ BOOL(Boolean):

این نوع داده به صورت بیتی میباشد که مقدار آن یا صفر خواهد بود (False) یا ۱ (TRUE)

این نوع داده تنها یک بیت از فضای حافظه را اشغال مینماید.

۴.۲.۳ : BYTE

این نوع داده ۸ بیتی میباشد . به عبارت دیگر زمانی که داده به صورت BYTE میباشد در واقع این داده ۸ بیت از فضای حافظه را به خود اختصاص میدهد . مقداری که این نوع داده میتواند داشته باشد 0~255 میباشد .

۴.۲.۴ : WORD

این داده به صورت ۲ بایت میباشد .به عبارت دیگر داده با فرمت WORD، ۱۶ بیت یا ۲ بایت از فضای حافظه را به خود اختصاص میدهد . مقداری که این نوع داده میتواند داشته باشد 0~65535 میباشد.

۴.۲.۵ : DWORD

این نوع داده که به آن Double Word گفته میشود به صورت 2Word میباشد .به عبارت دیگر این نوع داده ۳۲ بیت یا ۴ بایت از فضای حافظه را اشغال مینماید . مقداری که این نوع داده میتواند داشته باشد 0~4,294,967,295 میباشد .

۴.۲.۶ : INT

این نوع داده به صورت ۲ بایت میباشد .به عبارت دیگر داده با فرمت INT، مانند فرمت WORD، ۱۶ بیت یا ۲ بایت از فضای حافظه را اشغال میکند .مقداری که این نوع داده میتواند داشته باشد - $-2^{15} \sim (2^{15}-1)$ میباشد .تفاوت داده با فرمت INT و داده با فرمت WORD در مقدار عددی است که میتوانند به خود اختصاص دهند .

۴.۲.۷ : DINT

این داده که به آن Double INT گفته میشود به صورت 2INT میباشد .به عبارت دیگر این نوع داده ۳۲ بیت یا ۴ بایت از فضای حافظه را اشغال مینماید .مقدار عددی که این نوع داده میتواند داشته باشد $-2^{31} \sim (2^{31}-1)$ میباشد.تفاوت این نوع داده با DWORD در مقدار عددی است که هر کدام از این داده ها میتوانند به خود اختصاص دهند.

۴.۲.۸ : REAL

این نوع داده به صورت ۴ بایت میباشد .به عبارت دیگر این نوع داده ۴ بایت یا ۳۲ بیت از فضای حافظه را به خود اختصاص میدهد .این داده از نظر مقدار عددی به صورت $1.18*10^{-38} \sim 3.40*10^{38}$, $-3.40*10^{38} \sim -1.18*10^{-38}$ میباشد .به عبارت دیگر در نرم افزر KINCO Builder اعداد اعشاری با فرمت REAL تعریف میشوند.

۴.۳:تعریف داده ها :

۴.۳.۱ CONSTANT (مقدار ثابت) :

این اصطلاح به صورت مستقیم بیان گر یک مقدارعددی ثابت در برنامه میباشد. هنگامی که یک داده در برنامه به صورت ثابت بوده و در تمام طول برنامه تغییر نمیابد آن داده را به صورت یک مقدار ثابت در برنامه تعریف میکنیم .

جدول زیر بیانگر نحوه مقدار دهی انواع داده ها در برنامه KINCO Builder میباشد.

Data Type	Format[1]	Range of value	Example
BOOL	true, false	true, false	false
BYTE	B#digits	B#0 ~ B#255	B#129
	B#2#binary digits		B#2#10010110
	B#8#octal digits		B#8#173
	B#16#hex digits		B#16#3E
WORD	W#digits	W#0 ~ W#65535	W#39675
	2#binary digits		2#100110011
	W#2#binary digits		W#2#110011
	8#octal digits		8#7432
	W#8#octal digits		8#174732
	16#hex digits		16#6A7D
	W#16#hex digits		W#16#9BFE
DWORD	DW#digits	DW#0 ~ DW#4294967295	DW#547321
	DW#2#binary digits		DW#2#10111
	DW#8#octal digits		DW#8#76543
	DW#16#hex digits		DW#16#FF7D
INT	Digits	-32768 ~ 32767	12345
	I#digits		I#-2345
	I#2#binary digits [2]		I#2#1111110
	I#8#octal digits [2]		I#8#16732
	I#16#hex digits[2]		I#16#7FFF
DINT	DI#digits	DI#-2147483647 ~ DI#2147483647	DI#8976540
	DI#2#binary digits[2]		DI#2#101111
	DI#8#octal digits[2]		DI#8#126732
	DI#16#hex digits [2]		DI#16#2A7FF
REAL	Digits with decimal point	$1.18*10^{-38} \sim 3.40*10^{38}$, $-3.40*10^{38} \sim -1.18*10^{-38}$	1.0, -243.456
	xEy		-2.3E-23

۴.۳.۲ متغیرها :

برخلاف تعریفی که در بخش قبل از مقادیر ثابت داشتیم ، متغیرها بیان گر مقادیری در برنامه هستند که محتوای آنها براساس شرایط و موقعیت های مختلف تغییر پذیر میباشد .به عبارت دیگر متغرها حافظه هایی هستند که در برگیرنده مقادیری میباشد که در طول برنامه ممکن است تغییر کند.متغیرها میتواند در برگیرنده ورودی ها ،خروجی ها و حافظه های داخلی PLC باشد.

متغیرها نیز مانند مقادیر ثابت باید با نوع داده ای که در برخواهند داشت تعیین گردند تا مشخص گردد که هر متغیر چه مقدار از فضای حافظه را اشغال میکند.همچنین مغییرها میتوانند فقط قابل خواندن یا قابل خواندن و نوشتن باشند.

۴.۴ چگونگی دسترسی به فضاهای حافظه در PLC های KINCO:

PLCهای KINCO انواع اطلاعات و داده ها را حافظه های مختلف ذخیره مینماید. برای سهولت کاربران – KINCO K3 دو روش جهت دسترسی به این حافظه ها فراهم نموده است .

این دو روش به شرح زیر میباشد :

۱- آدرس دهی مستقیم

۲- آدرس دهی غیر مستقیم

قبل از توضیح دو روش ذکر شده در بالا ابتدا باید با انواع فضاهای حافظه در PLCهای KINCO و مشخصات آنها آشنا شویم .

حافظه در KINCO –K3 به چندین فضا ی مختلف با کاربریهای مختلف تقسیم میشود که هریک از این فضاها دارای مشخصات خاص خود میباشند .

فضاهای حافظه:

به جدول زیر توجه نمایید :

۱-

I	
Description	DI (Digital Input) Image Area. The KINCO-K3 reads all the physical DI channels at the beginning of each scan cycle and writes these values to I area.
Access Mode	Can be accessed by bit, by byte, by word and by double word
Access Right	Read only
Others	Can be forced, and cannot be retentive

فضای حافظه DI: این فضا مخصوص به تصویر ورودی های دیجیتال میباشد . PLCهای KINCOدر هر سیکل برنامه ابتدا کانال های فیزیکی ورودی را خوانده وتصویری از مقدار ورودی های فیزیکی را درفضای I قرار میدهند .در طول برنامه ، از این فضا جهت انجام عملیات استفاده میگردد .در ابتدای هر سیکل این مقدار به روز رسانی میشود .این فضای حافظه فقط قابل خواندن میباشد و نمیتوان مقداری را در برنامه برروی آن نوشت . این فضا به صورت بیتی ،بایتی ، WORDو DWORD قابل دسترس میباشد.

-۲

Q	
Description	DO (Digital Output) Image Area. At the end of each scan cycle, the KINCO-K3 writes the values stored in Q area to the physical DO channels.
Access Mode	Can be accessed by bit, by byte, by word and by double word
Access Right	Read/write
Others	Can be forced, and cannot be retentive

فضای حافظه DQ: این فضا مخصوص به تصویر خروجی دیجیتال میباشد . PLCهای KINCOدر انتهای هر سیکل اسکن برنامه مقدار ذخیره شده در این فضای حافظه را در کانال های فیزیکی خروجی ذخیره مینماید . این فضای حافظه هم قابل خواندن و هم قابل نوشتن میباشد .این فضا به صورت بیتی ،بایتی ، WORD و DWORD قابل دسترس میباشد .

-۳

AI	
Description	AI (Analog Input) Image Area. The KINCO-K3 samples all the AI channels at the beginning of each scan cycle, and converts the analog input values (such as current or voltage) into 16-bit digital values and writes these values to AI area.

Access Mode	Can be accessed by word (the data type is INT)
Access Right	Read only
Others	Can be forced, and cannot be retentive

فضای حافظه AI(ورودی آنالوگ): این فضا مربوط به ورودی های آنالوگی میباشد. PLC های KINCO در ابتدای هر سیکل اسکن یک سمپل از کانال های فیزیکی میگیرد و این مقدار آنالوگی ورودی را (که با توجه به توضیحات فصل گذشته میتواند جریان یا ولتاژ باشد) به یک مقدار دیجیتال ۱۶ بیتی تبدیل میکند و آن را در فضای حافظه AI ذخیره مینماید. این حافظه فقط قابل خواندن بوده و به صورت INT قابل دسترس میباشد.

-۴

AQ	
Description	AO (Analog Output) Image Area. At the end of each scan cycle, The KINCO-K3 converts the 16-bit digital values stored in AQ area into field signal values and writes to AO channels.
Access Mode	Can be accessed by word (the data type is INT)
Access Right	Read/write
Others	Can be forced, and cannot be retentive

فضای حافظه AQ(خروجی آنالوگ) : این فضا مربوط به خروجی های آنالوگ میباشد .PLCهای KINCO در انتهای هر سیکل اسکن مقدار ۱۶ بیتی دیجیتال را که در فضای AQذخیره شده است به یک سیگنال (مقدار آنالوگ) تبدیل کرده و آن را در کانال آنالوگ خروجی بریزد. این فضا هم قابل خواندن و هم قابل نوشتن میباشد و به صورت INT قابل دسترس است .

-۵

V	
Description	Variable Area. It's relatively large and can be used to store a large quantity of data.
Access Mode	Can be accessed by bit, by byte, by word and by double word
Access Right	Read/write
Others	Can be forced, and can be retentive

فضای V(Variable): این فضا بزرگترین فضای حافظه در PLCهای KINCO میباشد که میتوان اطلاعات و داده های بسیاری را درآن ذخیره نمود .این حافظه قابل خواندن و نوشتن میباشد و با فرمت های بیتی ،بایت ، WORD و DWORD قابل دسترس میباشد.

-۶

M	
Description	Internal Memory Area. It can be used to store the internal status or other data. Compared with V area, M area can be accessed faster and more propitious to bit operation.
Access Mode	Can be accessed by bit, by byte, by word and by double word
Access Right	Read/write
Others	Can be forced

فضای M(فضای حافظه داخلی) : این حافظه جهت ذخیره کردن وضعیت های داخلی و داده های مختلف استفاده میگردد. در مقایسه با فضای حافظه V میتوان گفت که فضای Mسریع تر بوده و برای عملیات بیتی بسیار مناسب تر میباشد ،با توجه به این نکته که فضای حافظه M بسیار محدود تر است .

-۷

SM	
Description	System Memory Area. System data are stored here. You can read some SM addresses to evaluate the current system status, and you can modify some addresses to control some system functions.
Access Mode	Can be accessed by bit, by byte, by word and by double word
Access Right	Read/write
Others	Cannot be forced and cannot be retentive

فضای حافظه SM(حافظه سیستمی) : داده ها و اطلاعات سیستمی در این فضا ذخیره میشود . این حافظه قابل خواندن و قابل نوشتن میباشد و به صورت بیتی ،بایتی ، WORD و DWORD قابل دسترس است .

-۸

L	
Description	Local Variable Area. KincoBuilder shall assign memory locations in L area for all the local variables and input/output variables automatically. **You are not recommended to access L area directly.**
Access Mode	Can be accessed by bit, by byte, by word and by double word
Access Right	Read/write
Others	Cannot be forced and cannot be retentive

حافظه L(فضای حافظه محلی): نرم افزار KINCOBuilder یک فضای حافظه را در فضای L جهت متغیرهای محلی و متغیرهای ورودی /خروجی به صورت اتومات اختصاص داده است .این فضا قابل خواندن و نوشتن بوده و به صورت بیتی ،بایتی ، WORD و DWORD قابل دسترسی میباشد.

نکته : توصیه نمیشود که به صورت مستقیم از این حافظه استفاده نمایید .

-۹

HC	
Description	High-speed Counter Area. Used to store the current counting value of the high-speed counters.
Access Mode	Can be accessed by double word (the data type is DINT)
Access Right	Read only
Others	Cannot be forced, and cannot be retentive

فضای HC: این فضا مربوط به High speed counter میباشد .زمانی که در برنامه از High speed counter ها استفاده میگردد مقدار در حال شمارش در این فضا ذخیره میگردد. این حافظه فقط قابل خواند میباشد و نمیتوان مقداری را در آن نوشت .این حافظه تنها به صورت DINT قابل دسترس میباشد.

۴.۵ آدرس دهی:

۴.۵.۱ آدرس دهی مستقیم :

آدرس دهی مستقیم به این مفهوم میباشد که به منظور دسترسی به حافظه های مختلف متغیرهایی را به آنها اختصاص دهیم تا هر زمان که به آن حافظه نیاز پیدا کردیم متغیر را فراخوانی نماییم .

دو روش در آدرس دهی مستقیم وجود دارد:

بیان مستقیم متغیر: در استاندارد IEC61131-3 بیان مستقیم یک متغیر با فرمت و علامت "٪" مشخص میشود.برای این نوع آدرس دهی ابتدا علامت ٪ ،سپس فضای حافظه و بعد از آن نوع داده و شماره داده مورد نظر تعیین میگردد .به این صورت مشخص میشود که متغیر مورد نظر در کدام حافظه و به چه فرمتی از داده میباشد .به عنوان مثال "%QB7" بیان گر بایت هفتنم خروجی میباشد .

متغیرهای سمبلیک : در این روش میتوان به هریک از متغیرها (که در بالا به آن اشاره شد) میتوان یک نام اختصاص داد. از آن به بعد میتوان از سمبل مربوطه به جای آدرس مستقیم متغیر در برنامه استفاده نمود . سمبل ها ی هر متغیر را میتوان در جدول Global variable و نیز جدول بالای مربوط به هر POU تعیین کرد .

برای توضیحات بیشتر به بخش مربوط به جدول متغیر ها مراجعه نمایید.

جدول زیر بیان کننده نحوه آدرس دهی مستقیم فضاهای حافظه با فرمت داده های مختلف میباشد :

فضای حافظه I:

Bit Addressing	Format	**%I***x.y*
	Description	*x*: byte address of the variable *y*: bit number, i.e. bit of byte. Its range is 0 ~ 7.
	Data type	BOOL
	Example	%I0.0 %I0.7 %I5.6
Byte Addressing	Format	**%IB***x*
	Description	*x*: byte address of the variable
	Data type	BYTE
	Example	%IB0 %IB1 %IB5
Word Addressing	Format	**%IW***x*
	Description	*x*: starting byte address of the variable. Since the size of WORD is 2 bytes, x must be an even number.
	Data type	WORD, INT
	Example	%IW0 %IW2 %IW4
Double word Addressing	Format	**%ID***x*
	Description	*x*: starting byte address of the variable. Since the size of DWORD is 4 bytes, *x* must be an even number.
	Data type	DWORD, DINT
	Example	%ID0 %ID4

فضای حافظه Q:

Bit	Format	**%Q***x.y*

Addressing	Description	*x*: byte address of the variable *y*: bit number, i.e. bit of byte. Its range is 0 ~ 7.
	Data type	BOOL
	Example	%Q0.0 %Q0.7 %Q5.6
Byte Addressing	Format	**%QB***x*
	Description	*x*: byte address of the variable
	Data type	BYTE
	Example	%QB0 %QB1 %QB4
Word Addressing	Format	**%QW***x*
	Description	*x*: starting byte address of the variable. Since the size of WORD is 2 bytes, *x* must be an even number.
	Data type	WORD, INT
	Example	%QW0 %QW2 %QW4
Double word Addressing	Format	**%QD**x
	Description	x: starting byte address of the variable. Since the size of DWORD is 4 bytes, *x* must be an even number.
	Data type	DWORD, DINT
	Example	%QD0 %QD4 %QD12

فضای ورودی آنالوگ (AI):

Word Addressing	Format	**%AIW***x*
	Description	*x*: starting byte address of the variable. Since the size of INT is 2 bytes, *x* must be an even number.
	Data type	INT
	Example	%AIW0 %AIW2 %AIW12

فضای خروجی آنالوگ (AQ):

Word Addressing	Format	**%AQW***x*
	Description	*x*: starting byte address of the variable. Since the size of INT is 2 bytes, *x* must be an even number.
	Data type	INT
	Example	%AQW0 %AQW2 %AQW12

فضای حافظه M:

Bit Addressing	Format	**%M***x.y*
	Description	*x*: byte address of the variable *y*: bit number, i.e. bit of byte. Its range is 0 ~ 7.
	Data type	BOOL
	Example	%M0.0 %M0.7 %M5.6
Byte Addressing	Format	**%MB***x*
	Description	*x*: byte address of the variable
	Data type	BYTE
	Example	%MB0 %MB1 %MB10
Word Addressing	Format	**%MW***x*
	Description	*x*: starting byte address of the variable. Since the size of WORD is 2 bytes, *x* must be an even number.
	Data type	WORD, INT
	Example	%MW0 %MW2 %MW12
Double word Addressing	Format	**%MD***x*
	Description	*x*: starting byte address of the variable. Since the size of DWORD is 4 bytes, *x* must be an even number.
	Data type	DWORD, DINT
	Example	%MD0 %MD4 %MD12

فضای حافظه V:

Bit Addressing	Format	**%V***x.y*
	Description	*x*: byte address of the variable *y*: bit number, i.e. bit of byte. Its range is 0 ~ 7.
	Data type	BOOL
	Example	%V0.0 %V0.7 %V5.6
Byte Addressing	Format	**%VB***x*
	Description	*x*: byte address of the variable
	Data type	BYTE
	Example	%VB0 %VB1 %VB10
Word Addressing	Format	**%VW***x*
	Description	*x*: starting byte address of the variable. Since the size of WORD is 2 bytes, *x* must be an even number.
	Data type	WORD, INT
	Example	%VW0 %VW2 %VW12
Double word Addressing	Format	**%VD***x*
	Description	*x*: starting byte address of the variable. Since the size of DWORD is 4 bytes, *x* must be an even number.
	Data type	DWORD, DINT
	Example	%VD0 %VD4 %VD12
REAL Addressing	Format	**%VR***x*
	Description	*x*: starting byte address of the variable. Since the size of REAL is 4 bytes, *x* must be an even number.
	Data type	REAL
	Example	%VR0 %VR4 %VR1200

فضای حافظه SM:

Bit Addressing	Format	**%SM***x.y*
	Description	*x*: byte address of the variable *y*: bit number, i.e. bit of byte. Its range is 0 ~ 7.
	Data type	BOOL
	Example	%SM0.0 %SM0.7 %SM5.6
Byte Addressing	Format	**%SMB***x*
	Description	*x*: byte address of the variable
	Data type	BYTE
	Example	%SMB0 %SMB1 %SMB10
Word Addressing	Format	**%SMW***x*
	Description	*x*: starting byte address of the variable. Since the size of WORD is 2 bytes, *x* must be an even number.
	Data type	WORD, INT
	Example	%SMW0 %SMW2 %SMW12
Double word Addressing	Format	**%SMD***x*
	Description	*x*: starting byte address of the variable. Since the size of DWORD is 4 bytes, *x* must be an even number.
	Data type	DWORD, DINT
	Example	%SMD0 %SMD4 %SMD12

فضای حافظه L:

Bit Addressing	Format	**%L***x.y*
	Description	*x*: byte address of the variable *y*: bit number, i.e. bit of byte. Its range is 0 ~ 7.
	Data type	BOOL
	Example	%L0.0 %L0.7 %L5.6
Byte Addressing	Format	**%LB***x*
	Description	*x*: byte address of the variable
	Data type	BYTE
	Example	%LB0 %LB1 %LB10
Word Addressing	Format	**%LW***x*
	Description	*x*: starting byte address of the variable. Since the size of WORD is 2 bytes, *x* must be an even number.
	Data type	WORD, INT
	Example	%LW0 %LW2 %LW12
Double word Addressing	Format	**%LD***x*
	Description	*x*: starting byte address of the variable. Since the size of DWORD is 4 bytes, *x* must be an even number.
	Data type	DWORD, DINT, REAL
	Example	%LD0 %LD4 %LD12

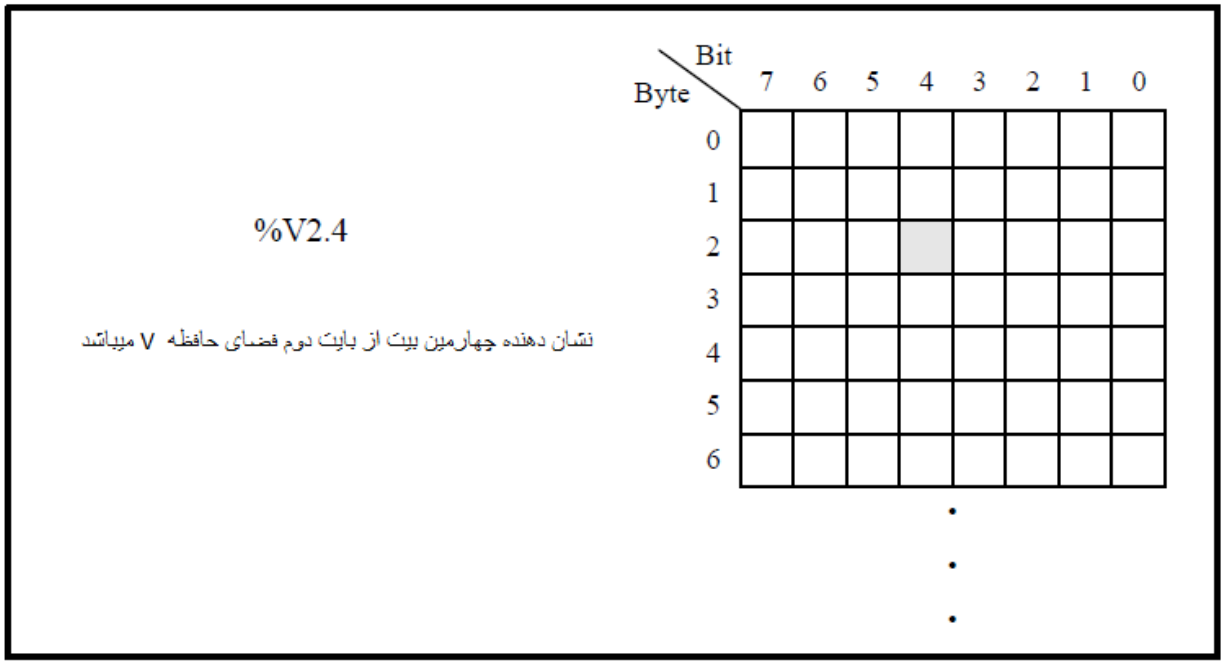
فضای حافظه HC:

Double word Addressing	Format	**%HC***x*
	Description	*x*: the high-speed counter number
	Data type	DINT
	Example	%HC0 %HC1

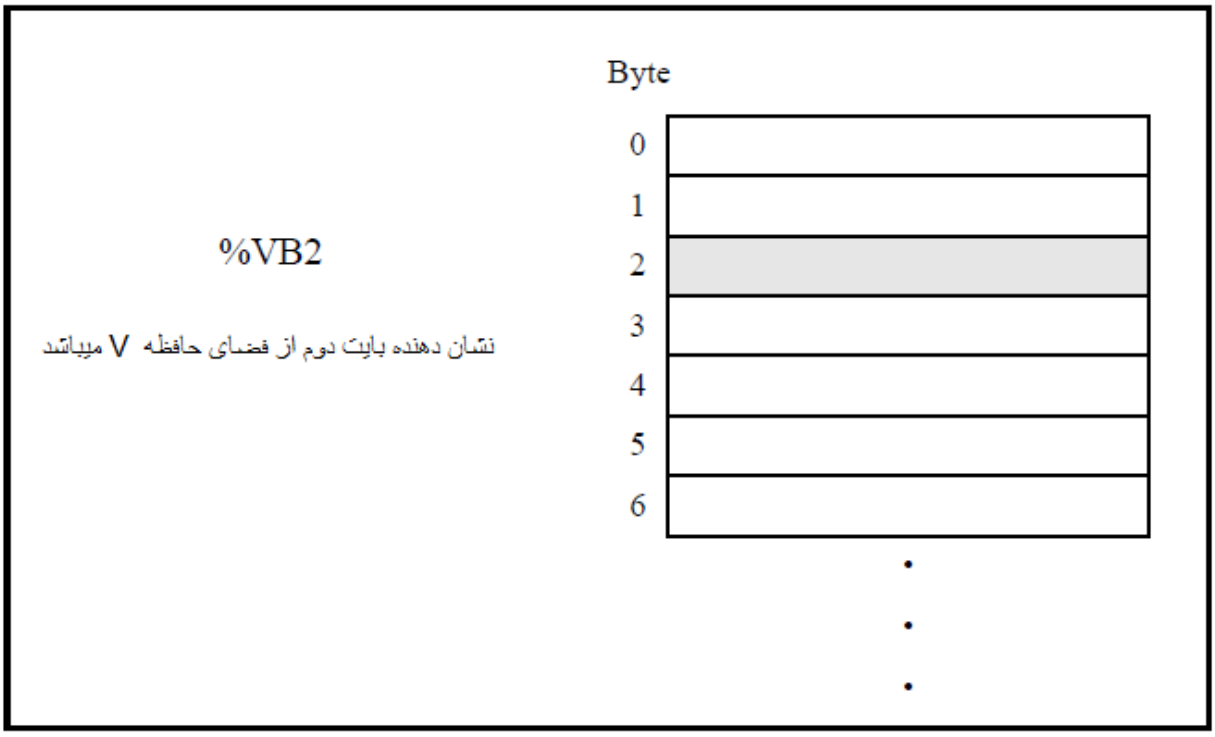
برای درک بهتر مفهوم آدرس دهی مستقیم به تصاویر زیر توجه نمایید :

در نقشه های زیر به عنوان مثال فضای حافظه V در نظر گرفته شد ه است و مفهوم هریک از آدرس دهی ها بیان گردیده است :

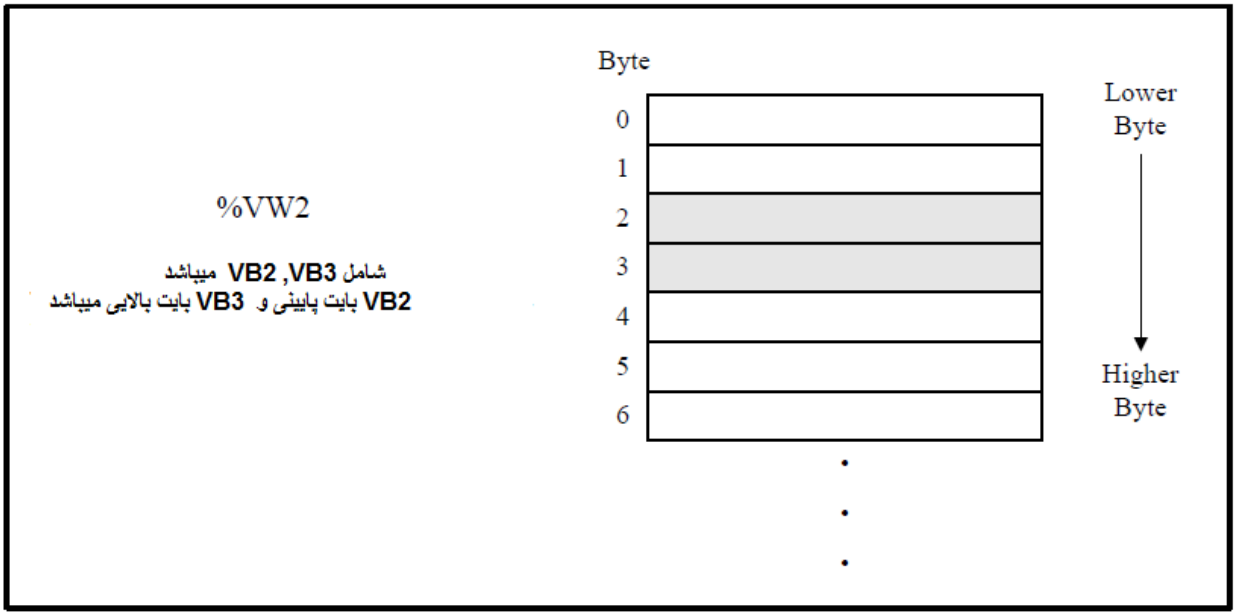
آدرس دهی بیتی :



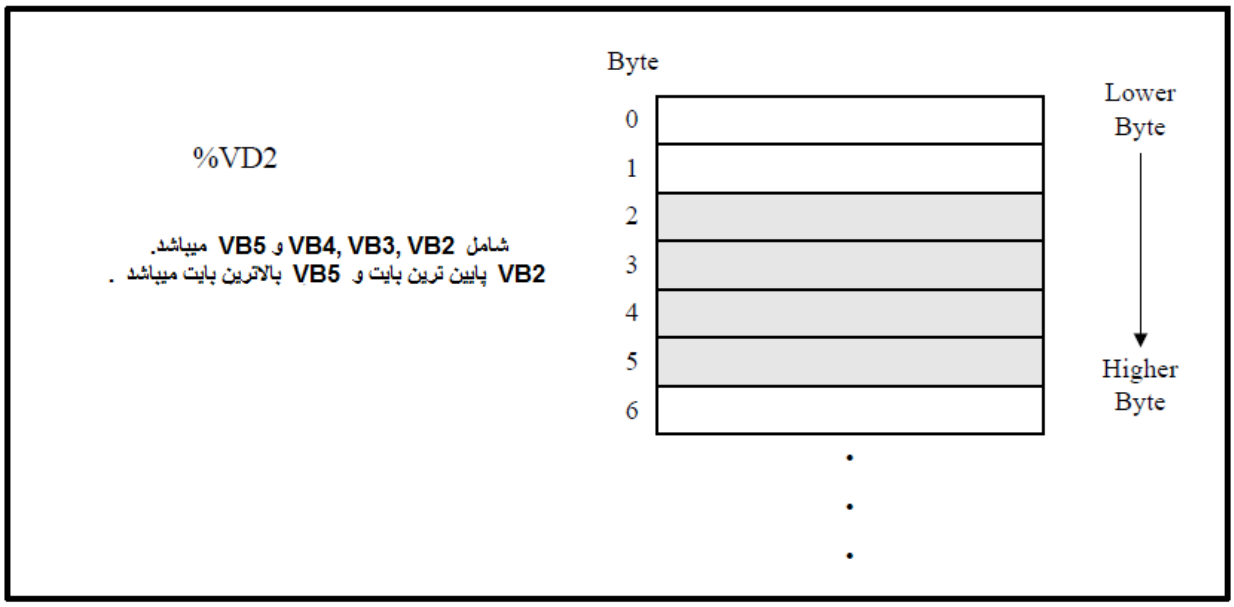
۲-آدرس دهی بایتی :



۳-آدرس دهی به فرمت WORD:



آدرس دهی به فرمت Double WORD:



۴.۵.۲ آدرس دهی غیر مستقیم :

Pointer یک متغیر با فرمت Double WORD میباشد که آدرس فیزیکی یک حافظه در آن ذخیره میشود در روش آدرس دهی غیر مستقیم از یک Pointer برای دسترسی به محتوای آدرسی که در آن pointer ذخیره شده است ،استفاده میشود.

Pointerها تنها به حافظه V دسترسی دارند.

نکته ۱: این روش آدرس دهی تنها در زبان IL قابل استفاده میباشد.

نکته۲: تنها cPUهای مدل K306EXو K308 میتواند روش آدرس دهی غیرمستقیم را پشتیبانی کند.

نحوه ایجاد یک pointer:

Pointer یک متغیر ۳۲ بیتی میباشد که محتوای آن میتواند با عملیاتی مانند ADD, SUB و غیره تغییر کند.زمانی که مقدار pointer کاهش یا افزایش میابد ،آدرسی که به آن اشاره دارد نیز افزایش یا کاهش میابد. زمانی که مقدار یک pointer را تغییر میدهید باید نوع داده متغیری که pointer به آن اشاره دارد دقت نمایید.

چنانچه pointer به یک متغیر بایتی اشاره کند،میتوان مقدار آن را با یک عدد integer تغییر داد.

چنانچه pointer به یک متغیر INT یا WORD اشاره کند، میتوان مقدار آن را با ضریبی از ۲ تغییر داد.

چنانچه pointer به یک متغیر DINT ,DWORD و یا REAL اشاره کند، میتوان مقدار آن را با ضریبی از ۴ تغییر داد.

به مثال زیر توجه نمایید:

```
LD      %SM0.0
MOVE    &VW0, %VD200      (*  ایجاد یکpointer (VD200) که به آدرس VW0 اشاره دارد  *)
MOVE    *VD200, %VW50     (*  اختصاص مقدارVW0 به VW50 . VD200 به VW0 اشاره دارد .  *)

ADD     DI#2, %VD200      (*  مقدارVD200 دو واحد اضافه میگردد. بنا براین VD200 به VW2 اشاره دارد .  *)
MOVE    *VD200, %VW52     (*  مقدار VW2 را بهVW52 اختصاص میدهد  *)
```

۴.۶: رنج آدرس های حافظه در CPU های مختلف :

PLCهای kinco از چندین سری تشکیل میشوند که رنج آدرس های حافظه در آنها متفاوت میباشد. جدول زیر رنج آدرسهای حافظه را در CPU های مختلف بیان میکند :

		CPU304	CPU304EX, CPU306	CPU306EX, CPU308
I	Size	2 bytes	8 bytes	32 bytes
	Bit address	%I0.0 --- %I1.7	%I0.0 --- %I7.7	%I0.0 --- %I31.7
	Byte address	%IB0、IB1	%IB0 --- %IB7	%IB0 --- %IB31
	Word address	%IW0	%IW0 ---% IW6	%IW0 ---% IW30
	Double-word address	———	%ID0 --- %ID4	%ID0 --- %ID28
Q	Size	2 bytes	8 bytes	32 bytes
	Bit address	%Q0.0 --- %Q0.7	%Q0.0 --- %Q7.7	%Q31.0 --- %Q31.7
	Byte address	%QB0	%QB0 --- %QB7	%QB0 --- %QB31
	Word address	———	%QW0 --- %QW6	%QW0 --- %QW30
	Double-word address	———	%QD0 --- %QD4	%QD0 --- %QD28
AI	Size	0	32 bytes	64 bytes
	Word address	———	%AIW0 --- %AIW30	%AIW0 --- %AIW62
AQ	Size	0	32 bytes	64 bytes
	Word address	———	%AQW0 -- %AQW30	%AQW0 -- %AQW62
HC	Size	8 bytes	24 bytes	
	Word address	%HC0，%HC1	%HC0 --- %HC5	
V	Size	2048 bytes	4096 bytes	
	Bit address	%V0.0 --- %V2047.7	%V0.0 ---%V4095.7	
	Byte address	%VB0 --- %VB2047	%VB0 --- %VB4095	
	Word address	%VW0 --- %VW2046	%VW0 --- %VW4094	
	Double-word address	%VD0 --- %VD2044	%VD0 --- %VD4092	
	REAL address	%VR0 --- %VR2044	%VR0 --- %VR4092	
M	Size	32 bytes		
	Bit address	%M0.0 --- %M31.7		
	Byte address	%MB0 --- %MB31		
	Word address	%MW0 --- %MW30		

	Double-word address	%MD0 --- %MD28
SM	Size	300 bytes
	Bit address	%SM0.0 --- %SM299.7
	Byte address	%SMB0 --- %SMB299
	Word address	%SMW0 --- %SMW298
	Double-word address	%SMD0 --- %SMD296
L	Size	272 bytes
	Bit address	%L0.0 --- %L271.7
	Byte address	%LB0 --- %LB271
	Word address	%LW0 --- %LW270
	Double-word address	%LD0 --- %LD268

: Function Block

فانکشن بلاک های استاندارد در IEC61131-3به شرح زیرمیباشد:

تایمرها :

TP تایمر پالس

TON: تایمر تاخیری در روشن کردن

TOF: تایمر تاخیری در خاموش کردن

کانترها:

CTU : شمارنده رو به بالا

CTD: شمارنده رو به پایین

CTUD: شمارنده رو به بالا و پایین

فلیپ فلاپ ها:

RS: که در آن دستور RESET برتری دارد.

SR : که در آن دستور SETبرتری دارد.

آشکار ساز های لبه :

R_TRIG:آشکار ساز لبه بالا رونده

F_TRIG: آشکار ساز لبه پایین رونده

هر فانکشن بلاک دارای یک instance میباشد .

فضای حافظه instance در فانکشن بلاک ها :

هر فانکشن بلاک دارای یک فضای حافظه مشخص برای ذخیره سازی instance مربوط به آن فانکشن بلاک میباشد.

به منظور توضیحات بیشتر به جداول زیر توجه فرمایید :

فضای حافظه تایمر :

T	
Description	Timer Memory Area, where instances of TON, TOF and TP can be allocated. It's used to store the status bits and current values of all the timer instances.
Access mode	Directly access the status bit and current value of a timer instance
Access right	Read only
Others	Can not be retentive, and can not be forced

فضای حافظه تایمر جایی است که instanceهای TON، TOF وTP در آن قرار دارد .این حافظه جهت ذخیره سازی بیت های وضعیت هر تایمر استفاده میشود .این حافظه فقط قابل خواندن میباشد.

فضای حافظه کانتر:

C	
Description	Counter Memory Area, where the instances of CTU, CTD and CTUD can be allocated. It's used to store the status bits and current values of all the counter instances.
Access mode	Directly access the status bit and current value of a counter instance
Access right	Read-only
Others	Can be retentive, and can not be forced

فضای حافظه کانترها فضایی است که instance های CTU، CTD و CTUD در آن قرار دارد.از این حافظه جهت ذخیره سازی بیت های وضعیت هر کانتر استفاده میگردد .این حافظه فقط قابل خواندن میباشد.

فضای حافظه RS:

RS	
Description	RS Bistable Area, where instances of RS can be allocated. It's used for storing the status bits for all the RS instances.

Access Mode	Directly access the status of the RS instances
Access Rights	Read-only
Others	Can not be retentive, and can not be forced

این فضای حافظه فلیپ فلاپ RS میباشد و برای ذخیره سازی بیت وضعیت فلیپ فلاپ RSمورد استفاده قرار میگیرد .این حافظه فقط قابل خواندن میباشد.

فضای حافظه SR:

SR	
Description	SR Bistable Area, where instances of SRcan be allocated. It's used for storing the status for all the SR instances.
Access Mode	Directly access the status bit of the SR instances
Access Rights	Read-only
Others	Can not be retentive, and can not be forced

این فضای حافظه فلیپ فلاپ SR میباشد و برای ذخیره سازی بیت وضعیت فلیپ فلاپ SRمورد استفاده قرار میگیرد .این حافظه فقط قابل خواندن میباشد.

تعداد فانکشن بلاک ها در CPU های مختلف متفاوت میباشد . جدول زیر بیان گر تعداد انواع فانکشن بلاک ها در CPUهای مختلف میباشد.

		CPU304	CPU304EX, CPU306	CPU306EX, CPU308
T	Amount	64	128	256

	Range	T0 --- T63	T0 --- T127	T0 --- T255
	Resolution	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T63: 100ms	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T127: 100ms	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T255: 100ms
	Max timing	32767* Resolution	32767* Resolution	32767* Resolution
C	Amount	64	128	256
	Range	C0 --- C63	C0 --- C127	C0 --- C255
	Max counting value	32767	32767	32767
RS	Amount	———		32
	Range	———		RS0 --- RS31
SR	Amount	———		32
	Range	———		SR0 --- SR31

فصل پنجم :چگونگی استفاده از نرم افزار Kinco Builder:

مقدمه:

دراین فصل با چگونگی نصب برنامه ، پروگرام کردن (برنامه نویسی) ، ارتباط با کامپیوتر و اجرای برنامه آشنا خواهید شد .هدف این فصل ارائه جزئیات بیشتر جهت استفاده بهتر و راحت تر کاربران میباشد

قابل نصب در کامپیوترهای شخصی میباشد. حداقل مشخصات لازم کامپیوتر جهت kinco Kinco-builder نرم افزار

نصب برنامه به شرح زیر میباشد:

- CPU: 133 MHz or higher
- Hard disk: at least 10M bytes of free space
- RAM: 32M or more
- Keyboard, mouse, a serial communication port
- 256-color VGA or higher, 1024*768,
- Operating system: English version Windows NT 4.0 (or later version)/Windows 2000/Windows XP

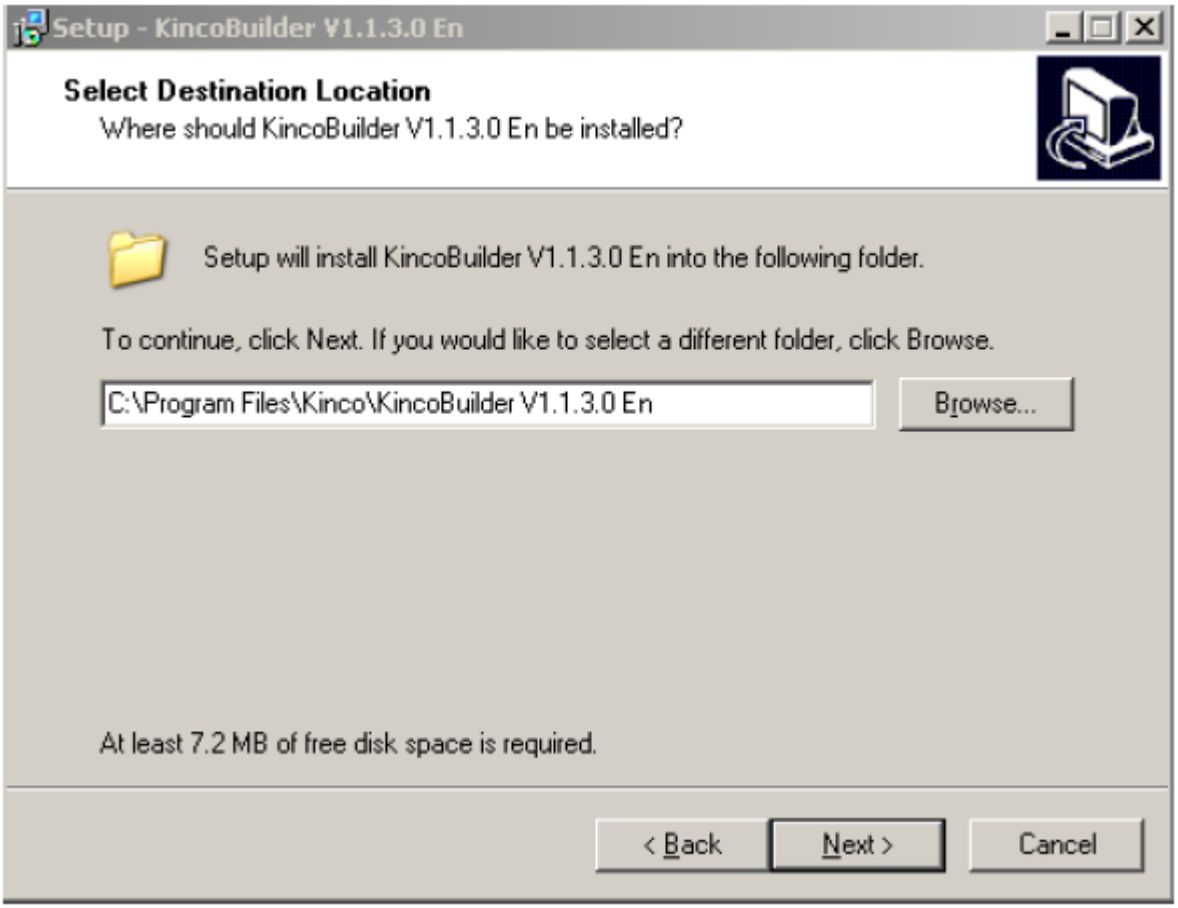
۵.۱: نصب و غیر فعال کردن نرم افزار(install/uninstall) :

۵.۱.۱: نصب نرم افزار KINCO Builder:

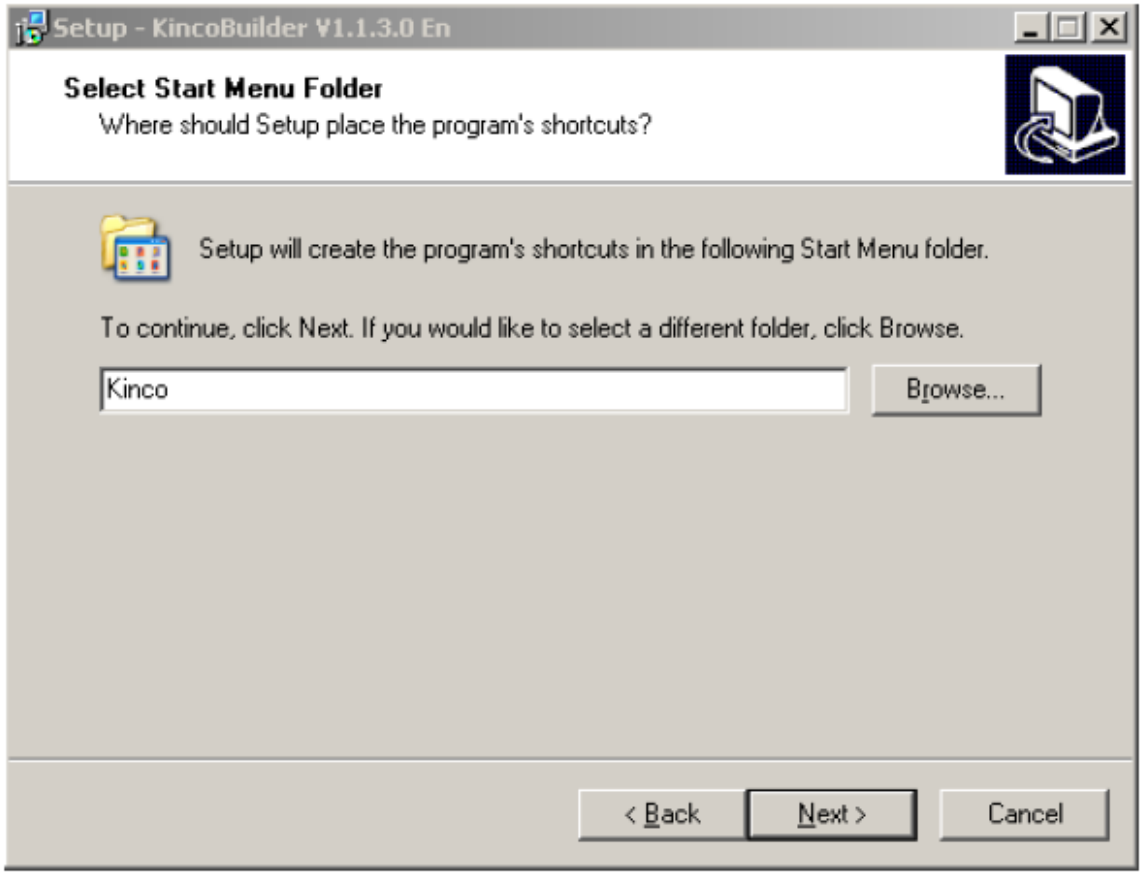
برای نصب برنامه چنانچه version قدیمی تری از برنامه روی کامپیوتر نصب باشد ابتدا باید uninstallشود. در تمام مراحل نصب برنامه با کلیک بر روی گزینه cancleمیتوان ازادامه نصب خارج شد .

a: اجرای kincobuilder Vxxxx_setup.exe (xxxxبیانگر versionبرنامه میباشد.) تا صفحه setup wizardمانند تصویر زیر روی صفحه نمایش نمایان شود

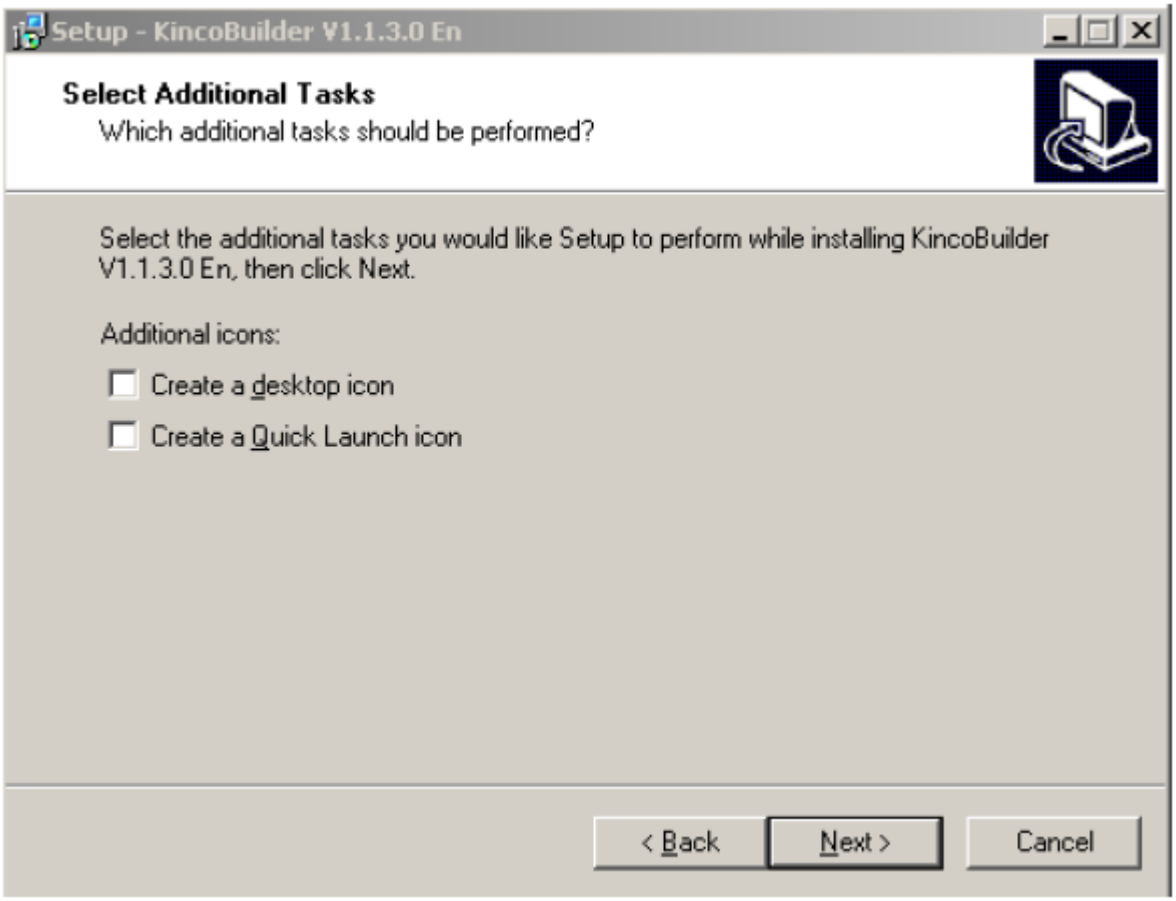
b:برای ادامه مراحل روی گزینه next کلیک کنید ومیتوانید مسیرنصب برنامه را انتخاب کنید.



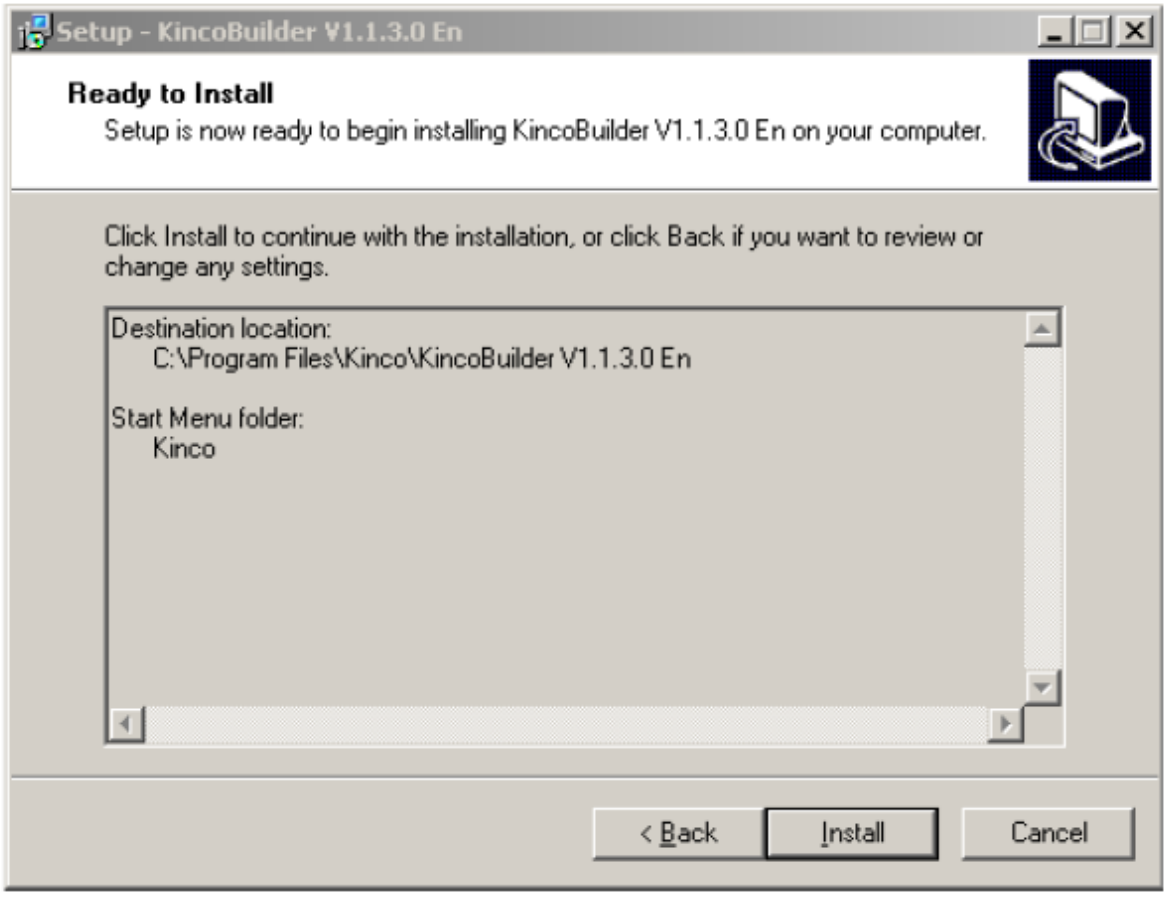
C:روی گزینه next کلیک کنید تا فایلی جهت ذخیره شدن shortcut در نظر گرفته شود،فولدر در نظرگرفته شده به صورت پیش فرض به نام kinco میباشد.



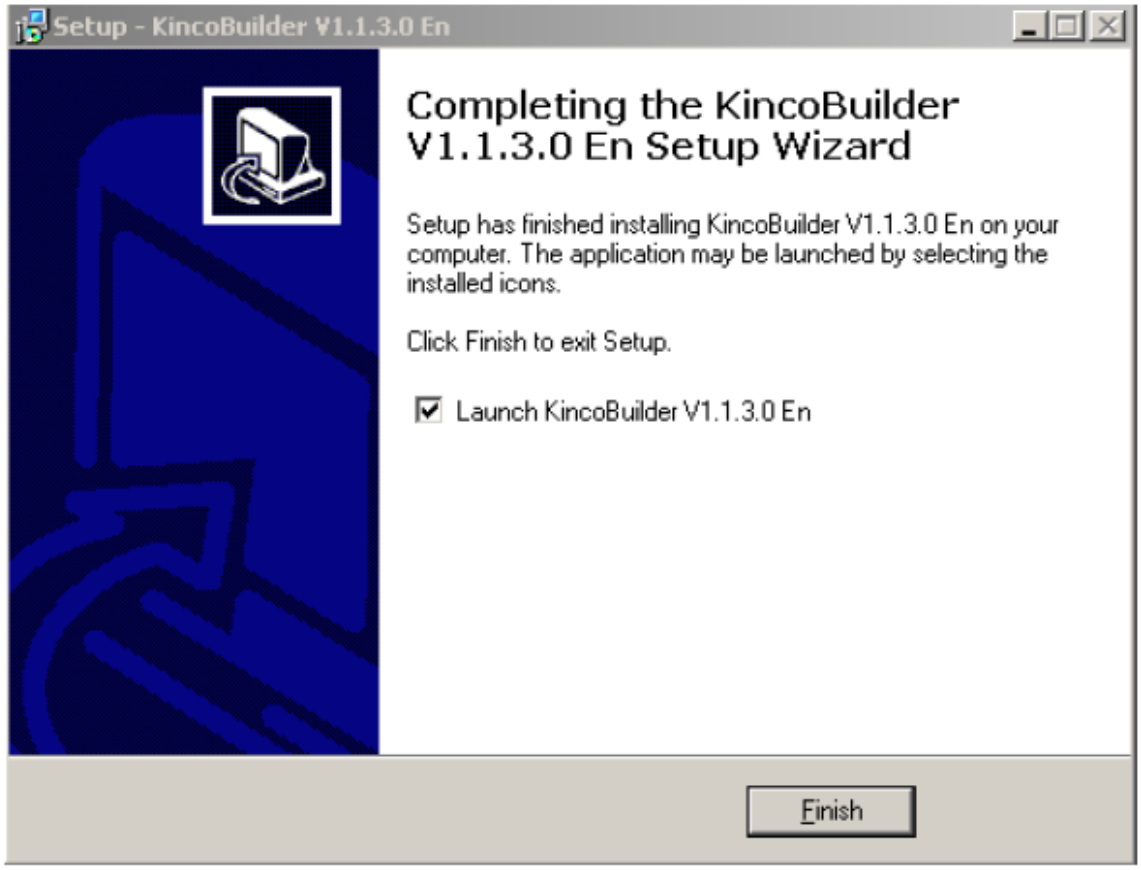
d: روی گزینه next کلیک کنید میتوان برای نمایش icon روی صفحه desktop گزینه اول را انتخاب کرد .



e: روی گزینه next کلیک کنید .برنامه جهت نصب آماده است .



f: روی گزینه install کلیک کنید .برنامه به طور کامل روی سیستم نصب میشود .



g: سپس روی برای تمام شدن مراحل نصب روی گزینه finish کلیک کنید .

چنانچه همزمان گزینه launch kincobuilder را انتخاب کنید ، برنامه فورا آغاز میشود .

۵.۲ Uninstall: کردن نرم افزار :

قبل از آنکه برنامه را uninstall نمایید ،ابتدا از برنامه خارج شوید .

دو روش برای uninstall کردن برنامه وجود دارد :

روش اول : روی منوی start کلیک کنید ، از روی program برنامه kinco را پیدا کرده و روی گزینه uninstall kincobuilder کلیک کنید .(start /all program /kinco/uninstall kincobuilder) فایل kincobuilder به صورت خودکار حذف میشود .

روش دوم : از منوی start و قسمت setting وارد control panel شوید .روی گزینه add or remove program دو بار کلیک کنید .سپس برنامه kincobuilderVx.x.x.x را انتخاب کنید .روی گزینه remove کلیک کنید .برنامه پاک میشود .

(start/setting/control panel/add or remove program/remove)

۵.۳ چگونگی باز کردن و خارج شدن از برنامه :

: چگونگی باز کردن برنامه : دوروش برای بازکردن برنامه (شروع برنامه) وجود دارد .

a:روی منوی start کلیک کنید . programرا انتخاب کنید . از روی kincoگزینه kincobuilderرا انتخاب کنید (start/program/kinco/kincobuilder).

b:چنانچه در مراحل نصب desktop iconرا ایجاد شده باشد روی نشانه  دو بار کلیک کنید .

سه روش برای خارج شدن (بستن)برنامه وجود دارد :

a: در برنامه گزینه Fileرا باز کرده و گزینه exitرا انتخاب کنید .

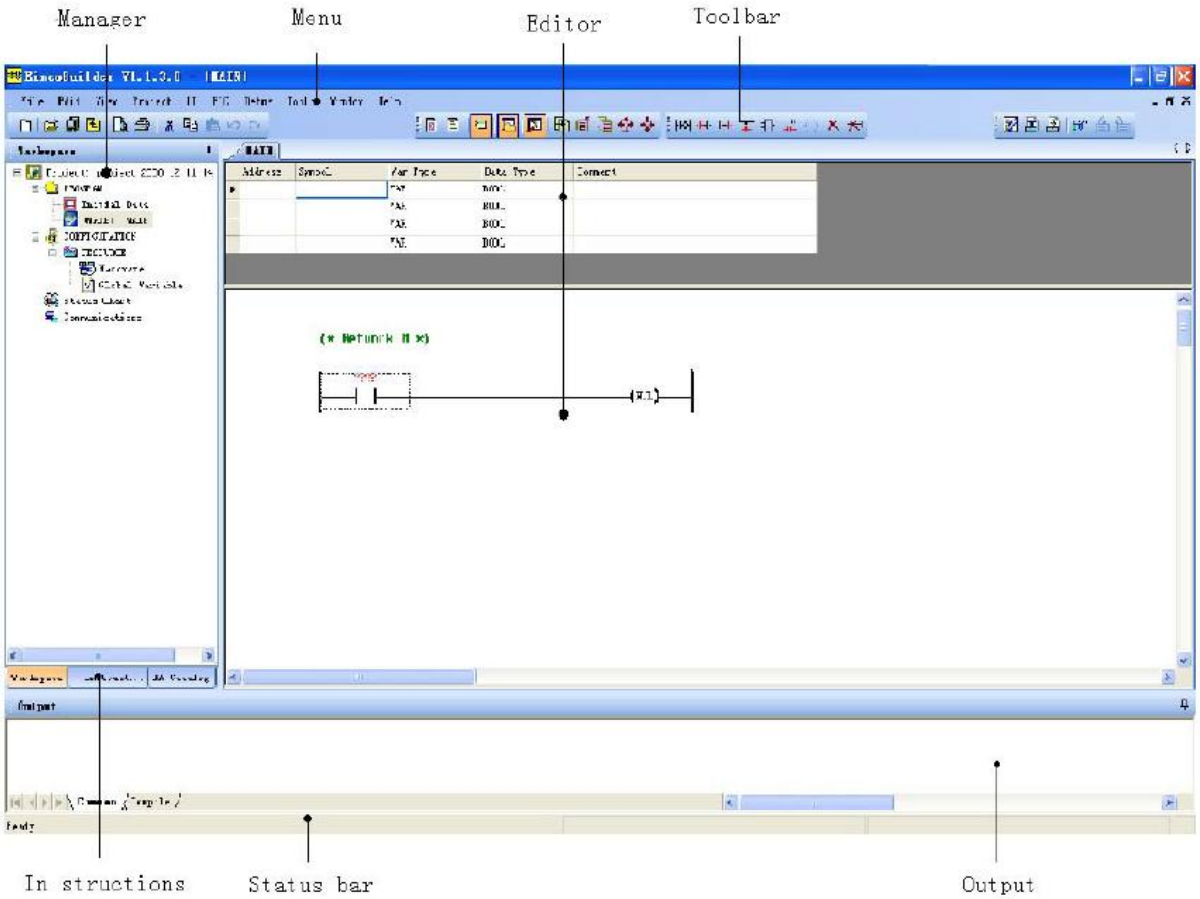
b: میتوان از راه کوتاه Alt+F4 استفاده کرد .

c:میتوان با استفاده از نشانه  در قسمت بالا سمت راست صفحه main از برنامه خارج شد .

نحوه کاربا نرم افزار kincobuilder:

بعد از نصب نرم افزار kinco builderبا کلیک بر روی آیکون مربوطه میتوان وارد محیط برنامه نویسی شد .

۵.٤:معرفی بخش های نرم افزار:



Menu: شامل تمامی دستورات کاربردی در نرم افزار kinco builderمیباشد .

نوارابزار Toolbar: شامل سمبلهایی است که به تناوب دربرنامه استفاده میشود و این بخش امکان دسترسی آسان را به این دستورات فراهم مینماید .

Statusbar:این بخش شامل وضعیت اطلاعات و پیغام هایی است که توسط کامپیوتر برای اجرای برنامه داده میشود .

Manager: این بخش شامل نمایش درختی برای هر پروژه میباشد .بخش های این صفحه عبارتند از :
Program,Hardware,Global Variable,....این بخش ها در درک ساختمان پروژه به کاربر کمک مینماید .
manager پروژه یک ابزار راحت برای سازماندهی برنامه و مدیریت پروژه میباشد.

با راست کلیک کردن برروی هر کدام از این بخش ها (شاخه ها) میتوان به منوی مربوط به آن بخش دسترسی پیدا کرد .

Editor: این بخش شامل جدول متغیرها (variable table) و editorبرنامه (IL یا LD) میباشد . برنامه مربوط به پروژه در بخش Program Editorنوشته شده و متغیرهای محلی ، پارامترهای ورودی و خروجی در Variable Table مشخص میگردند .

Instructions: درنوار Instructions دستورات به صورت درختی نمایش داده شده است .این بخش شامل تمام دستوراتی است که در زبان IL و LD استفاده میشود .

HW catalog: این قسمت مربوط به سخت افزار میباشد .در این قسمت میتوان سخت افزار مورد نیاز جهت پروژه مورد نظر را انتخاب و پیکر بندی کرد.

Output: این صفحه بیانگر چندین مشخصه میباشد .نوارهایی که در پایین صفحه Outputقراردارد به ترتیب :

نوار common: این بخش بیانگر یک سری اطلاعات درباره آخرین عملیات انجام شده میباشد .

نوار Compile: این بخش بیانگر اطلاعات مربوط به کامپایل برنامه میباشد .

استفاده از kinco-Builderجهت ایجاد برنامه برای کاربرد های مورد نظر:

برای استفاده از نرم افزار KINCOBuilder و ایجاد یک پروژه ابتدا باید اجزای اصلی یک برنامه را بشناسیم .

تمام قسمت های یک پروژه در جدول زیر توضیح داده شده است ،مواردی که روبروی آنها optionalنوشته شده است به این مفهوم میباشد که قسمت مورد نظر برای برنامه به صورت اختیاری میباشد و میتوان در پروژه ازآن صرف نظر کرد .

PROGRAM	Initial Data (Optional)	You can assign initial numerical values to BYTE, WORD, DWORD, INT, DINT and REAL variables in V area. The CPU module processes the Initial Data once at power on and then starts the scan cycle.
	Main Program	It is the execution entry of the user program. The CPU module executes it once per scan cycle. Only 1 Main Program exists in a project.

	Interrupt routines (Optional)	They are interrupt service routines used to process the specific interrupt events. They are not invoked by the main program. You attach an interrupt routine to a predefined interrupt event, and the CPU module executes this routine only on each occurrence of the interrupt event. At most 16 interrupt routines are allowed in a project.
	Subroutines (Optional)	The subroutines can only be executed when they are invoked by the main program or interrupt routines. Subroutines are helpful to better structure the user program. They are reusable, and you can write the control logic once in a subroutine and invoke it as many times as needed. Formal input/output parameters can be used in the subroutines. At most 16 subroutines are allowable in a project.
CONFIGURATION	Hardware	Here you can configure the KINCO-K3 modules used in your control system, including their addresses, function parameters, etc. The CPU module shall process the hardware configuration once at power on and then execute other tasks.
	Global variables (Optional)	Here you can declare the global variables required in the project.

همان طور که در جدول بالا مشاهده میکنید یک پروژه از بخش های مختلفی تشکیل شده است که در ادامه به توضیح این بخش ها میپردازیم :

۵.٤.۱بخش manager:

قسمت managerدر هر پروژه از بخش های زیر تشکیل میشود:

۵.۴.۲ PROGRAM :

این بخش از زیر بخش های زیر تشکیل میشود:

۱-Initial Data: این قسمت در یک پروژه اختیاری میباشد و زمانی از آن استفاده میگردد که بخواهیم در ابتدای برنامه ،زمانی که CPU روشن شده و اولین سیکل برنامه در حال اجرا میباشد حافظه Vرا مقداردهی نماییم .به عبارت دیگر برای مقداردهی اولیه برخی از حافظه ها از این قسمت استفاده مینماییم .باید توجه داشت که داده ها در این قسمت میتواند به فرمت های بایت ، WORD، DWORD، INT، DINTو REALباشد

۲- Main: در هر برنامه یک mainاصلی وجود دارد . CPUدر هر سیکل اسکن یک بار mainبرنامه را اجرا میکند.این قسمت اصلی ترین بخش برنامه میباشد.

چنانچه در برنامه وقفه و یا زیر برنامه هم داشته باشیم بخش های زیر به قسمت PROGRAM اضافه میگردند.

۳- interrupt Routine (روتین وقفه):

این قسمت به صورت اختیاری و بنا بر نیاز برنامه ،در یک پروژه ایجاد میگردد. این روتین برنامه ای است که کاربر درآن مشخص میکند که زمانی که واقعه خاصی (واقعه مربوط به روتین)اتفاق افتاد اجرا شود. به عبارت دیگر درهنگام استفاده از این بخش روتین وقفه به یک واقعه خاص انتصاب داده میشود، هنگامی که واقعه اتفاق میافتد CPUروتین مربوطه را اجرا مینماید.

۴-Subroutine (زیر برنامه):

زیر برنامه تنها زمانی اجرا میشود که در main ویا یک روتین وقفه فراخوانی شده و شرط مربوط به اجرای آن برقرار باشد. زیر برنامه برای بهبود ساختار یک پروژه بسیار مفید است . در هر زیر برنامه میتوان برنامه ای کنترلی مورد نظر را نوشت و هر زمان که نیاز باشد فراخوانی نمود. این قسمت نیز اختیاری و بنا بر نیاز برنامه میباشد. در ادامه به صورت کامل به شرح آن میپردازیم.

۵.۴.۳ CONFIGURATION (پیکر بندی):

این قسمت شامل دو بخش عمده HARDWAREو Global Variable میباشد.

1-Hardware(سخت افزار) : در این بخش سخت افزار مورد نیاز که در سیستم کنترلی به کار رفته است و مورد نیاز پروژه میباشد را انتخاب کرده و آدرس ها و سایر پارامترهای مربوطه را میتوان تنظیم نمود .زمانی که برق سیستم وصل میشود CPUابتدا یک باز پیکر بندی سخت افزار را تحلیل کرده و سپس بقیه عملیات را انجام میدهد .

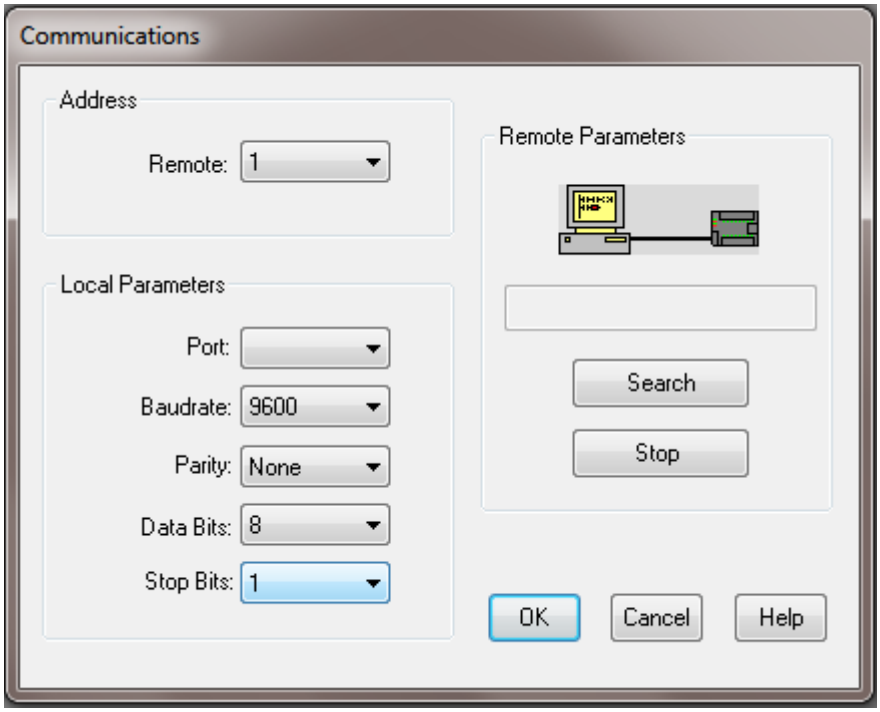
2- Global Variable(متغیرهای عمومی) : استفاده از این بخش نیز اختیاری و بنا به نیاز در برنامه میباشد .در این بخش میتوان به متغیرهای مورد استفاده در برنامه یک نام و سمبل اختصاص داده و در سرتاسر برنامه از آن سمبل (به جای استفاده از متغیر مربوطه) استفاده کرد .

۵.۴.۴ :Status chart

در این بخش با استفاده از یک جدول میتوان مقدار تمامی متغیر های مورد نظر را در حین اجرای برنامه مشاهده نمود . به عبارتی دیگر با استفاده از این جدول مقادیرتمامی متغیرهای مورد نظر را میتوان در قالب یک جدول مانیتور کرد.همچنین با استفاده از این جدول میتوان مقادیر مورد نظر را به حافظه های مشخص شده در این قسمت وارد (با استفاده از forceکردن)وارد نمود.

۵.۴.۵ Communication: :

با استفاده از این قسمت میتوان ارتباط بین PC(کامپیوتر) و PLC را چک نمود .این قسمت شامل تنظیمات پارامترهای ارتباطی میباشد.



۵.۴.۶:ذخیره کردن پروژه:

محل ذخیره شدن فایل های یک پروژه :

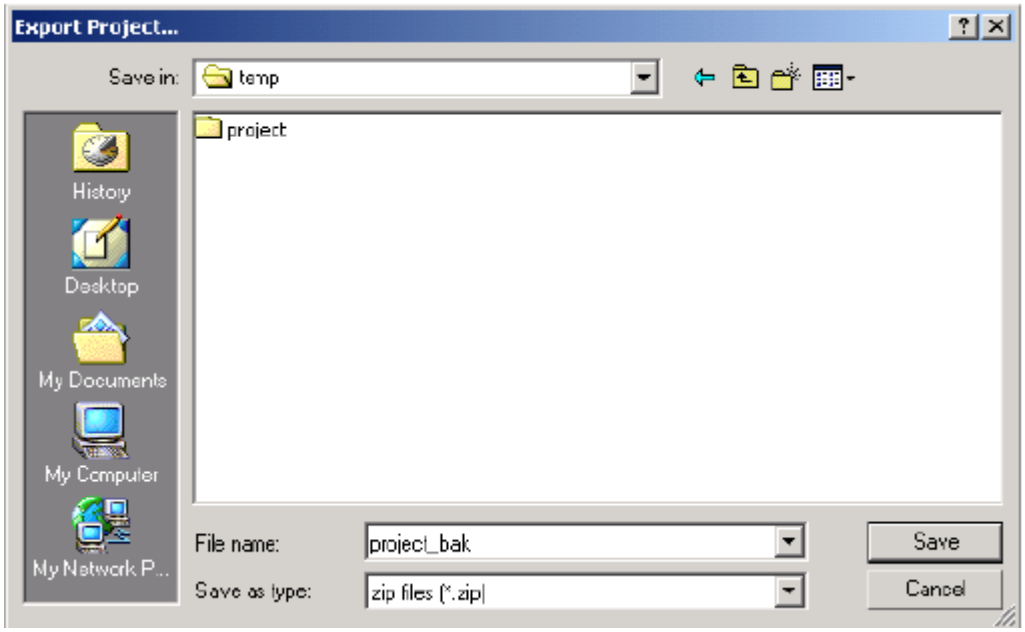
زمانی که یک پروژه تولید میشود نرم افزار Kinco-Builder ابتدا از شما میخواهد که مسیری را برای ذخیره کردن فایل پروژه تعیین کنید .سپس یک فایل خالی با پسوند "kpr." در این مسیر تولید میشود .سپس یک فولدر با همین نام در این مسیر ایجاد میشود .از این فولدر برای ذخیره شدن تمام فایل های برنامه،متغیرهای برنامه و فایل های موقتی برنامه استفاده میشود .به عنوان مثال زمانی که پروژه ای به نام Example در مسیر C:\temp تولید میشود مسیر فایل پروژه به صورت C:\temp\example.kpr و مابقی فایل ها در فولدر C:\temp\example قرار میگیرد.

۵.۴.۷ Importing and exporting Project: :

Export: این گزینه امکان Compressکردن فایل های مربوط به پروژه در یک فایل به صورت فایل zip را به کاربر میدهد .

برای Exportکردن : از قسمت منوی برنامه Fileراباز کرده و گزینه Export را انتخاب کنید .

صفحه ای مانند تصویر زیر نمایش داده میشود

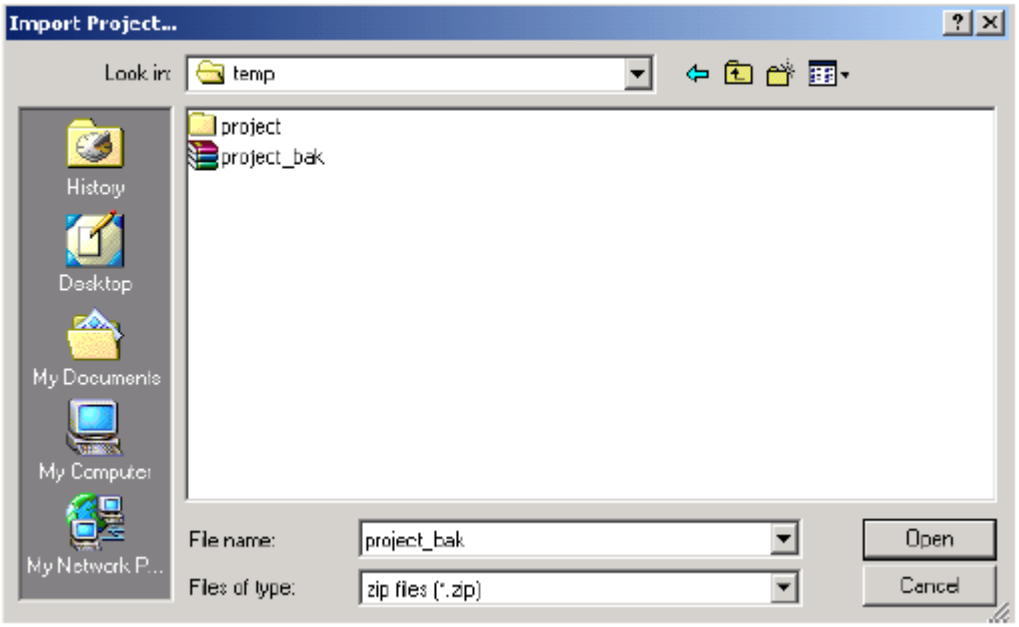


مسیر ذخیره فایل و نام فایل را انتخاب کرده سپس گزینه saveراانتخاب کنید .

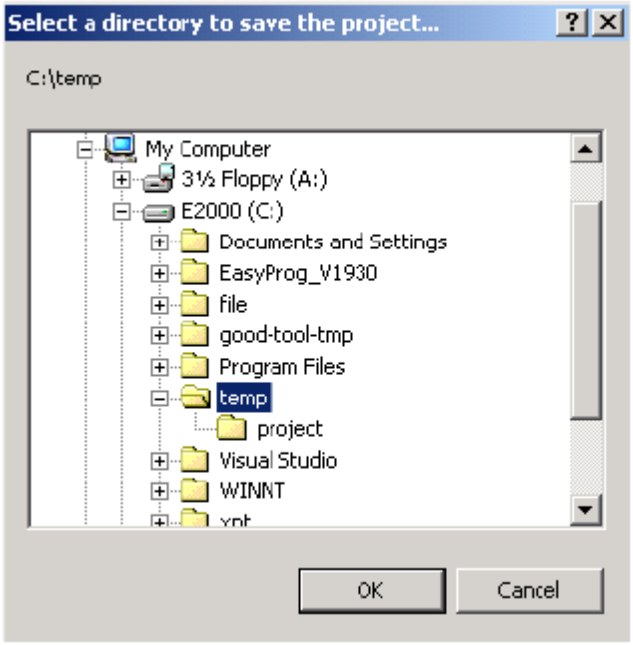
Import:

این امکان را به کاربرمیدهد که فایلی توسط Export به صورت zip درآمده است ، باز نمایید .

برای Importکردن : از قسمت منو Fileرا بازکرده و گزینه Import را انتخاب نمایید .صفحه ای مانند تصویر نشان داده شده در زیر نمایش داده میشود



فایل مورد نظر را انتخاب وگزینه openرا انتخاب کنید .پنجره ای مانند تصویر نشان داده شده در زیر باز میشود



پوشه موردنظر را انتخاب کرده و okکنید . فایل ذخیره شده در این پوشه باز میشود .

۵.۴.۸:چگونگی ارتباط Kinco-K3 با کامپیوتر:

در ماژول CPU دو پورت سریال (RS 232یاRS485) برای ارتباط PLC با تجهیزات دیگر وجود دارد .در این قسمت چگونگی ارتباط ماژول CPU (با پورت RS232) با کامپیوتر و برنامه نویسی PLC توسط نرم افزار توضیح داده شده است .

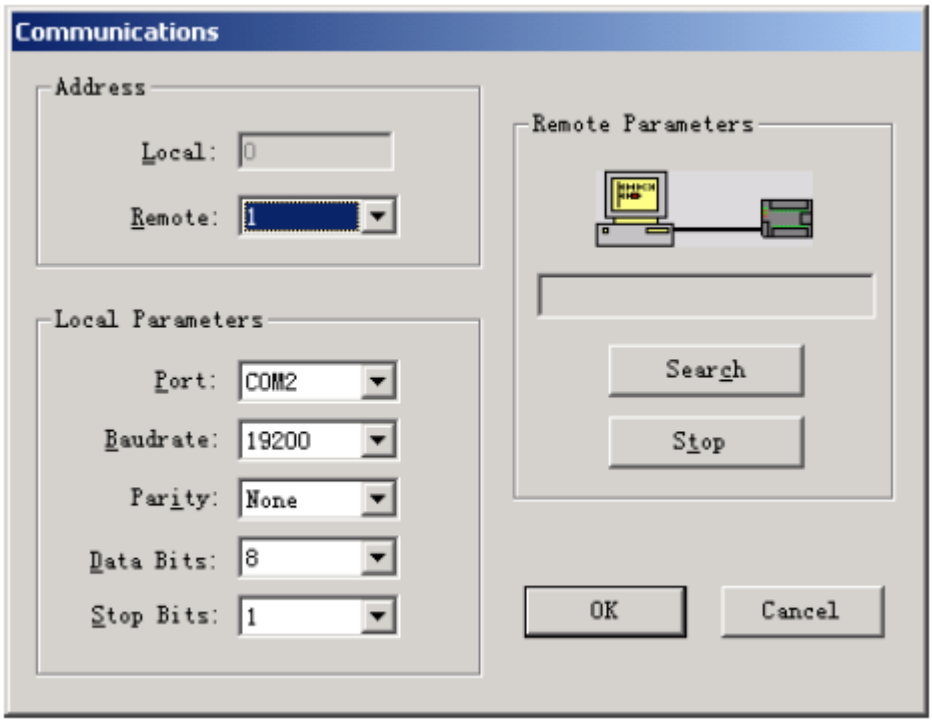
۱):توسط نرم افزار Kinco-Builderیک پروژه جدید بسازید و یا پروژه ای را که از قبل ساخته اید باز نمایید .

پورت سریال کامپیوتر را توسط کابل مناسب به CPUماژول متصل کنید .

۲)پارامترهای مربوط به ارتباط پورت سریال کامپیوتر را پیکره بندی نمایید .

a)از قسمت منو Toolsرا انتخاب نمایید و سپس روی گزینه communicationکلیک کنید (میتوانید در صفحه Manager روی گزینه communicationدو بار کلیک کنید و یا روی گزینه communicationدر همین صفحه راست کلیک کرده و گزینه open را انتخاب کنید .)

پنجره ای مانند تصویر نشان داده شده در پایین نمایش داده میشود



b) شماره PLCمورد نظر(موقعیت PLC)را در قسمت Remoteوارد نمایید .

در قسمت Portشماره پورت کام که در کامپیوتر استفاده شده است را وارد نمایید .در قسمت های دیگر پارامترهای پورت انتخابی comرا بر اساس پورت CPU وارد نمایید .(Boudrate,Parity,Data Bits,Stop Bits).سپس دکمه okرا زده تا تنظیمات فوق ذخیره شود .

چنانچه پارامترهای ارتباطی پورت CPU را نمیدانید میتوانید به دو طریق زیر میتوانید این اطلاعات را بدست آورید :

پورت مورد استفاده در کامپیوتر را انتخاب نموده و دکمه search را بزنید ، تا نرم افزار نرم افزار پارامترهای cpu ایی که online میباشد را به صورت اتوماتیک بیابد. این عملیات ممکن است چند ثانیه تا چند دقیقه طول بکشد . در صورتی که عملیات search با موفقیت انجام شد نرم افزار kincobuilderبه صورت اتوماتیک پارامترهای مناسب برای کامپیوتر را تنظیم میکند.

تغذیه CPUرا قطع کرده و وضعیت عملکردی CPUرا در مد STOP قرار دهید. سپس تغذیه را وصل نمایید. در این صورت پورت cpu از پارامترهای ارتباطی سریال که به طور پیش فرض برای آن تنظیم شده است استفاده میکند. این پارامترها به صورت زیر میباشد :

Station number :1

Baudrate :19200

Parity :none

Data bits:8

Stop bit :1

شما میتوانید پورت سریال کامپیوتر خود را مطابق با این پارامترها تنظیم نمایید.

۳) پس از این که پارامترهای ارتباطی پورت COMکامپیوتر را پیکره بندی کردید اکنون میتوانید Kinco-K3را پروگرام کنید .

چگونگی اصلاح پارامترهای ارتباطی کامپیوتر :

پس از آنکه CPU ماژول را به کامپوتر متصل نمودید ، پارامترهای ارتباطی آنرا اصلاح و تغییر دهید.

۱) ابتدا پنجره "HARDWARE"را به یکی از روش های زیر را باز کنید.

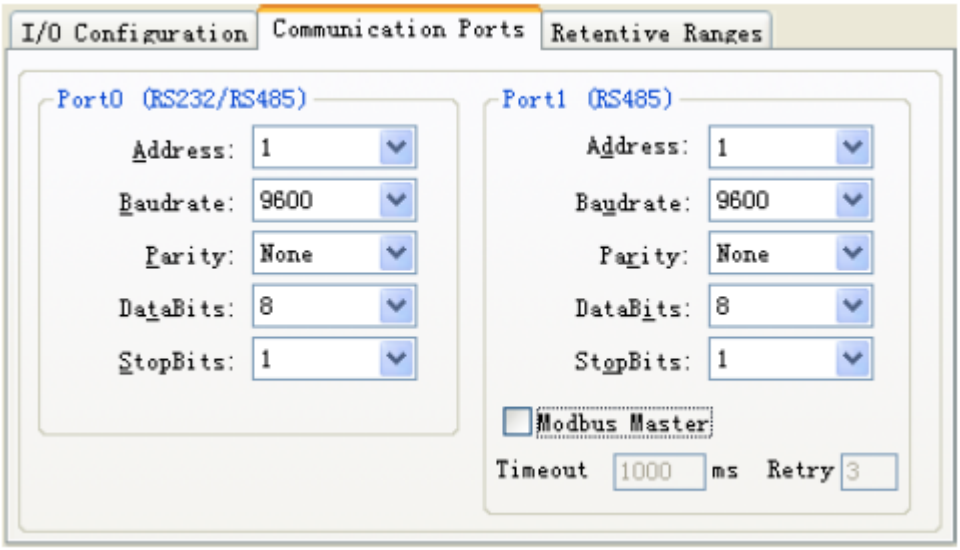
روی گزینه HARDWARE در قسمت Managerدو بار کلیک نمایید .

روی گزینه HARDWAREراست کلیک نمایید و گزینه openرا انتخاب کنید.

قسمت بالای پنجره Hardware(جدول بالا)لیست جزئیات مازول PLc را به صورت یک جدول نشان میدهد که به آن جدول پیکره بندی میگوییم .جدول پیکره بندی ،پیکره بندی واقعی را بیان میکند و شما باید به این نکته دقت نمایید که پیکر بندی این جدول دقیقا مشابه چیدمان قطعات شما در سخت افزار باشد.

قسمت پایین پنجره Hardware تمام پرامترهای ماژول انتخابی در جدول پیکره بندی را نشان میدهد که آن را پنجره پارامترها مینامیم .

ماژول CPU را در جدول پیکره بندی انتخاب نمایید و سپس نوار Communication port(پورت های ارتباطی) را در پنجره پارامترها انتخاب کنید .اکنون شما میتوانید پارامترهای ارتباطی را مانند تصویر نشان داده شده در پایین اصلاح کنید.



۴)پس از آنکه پارامترها را اصلاح نمودید ،باید آنها را درمازول CPUدانلود کنید .

نکته : پارامترهای پیکره بندی بی تاثیر خواهند بود مگر انکه آنهارا دانلود کنید .

۵.۴.۹:مراحل گام به گام ساخت یک پروژه جدید :

نام پروژه : example

kinco-K306-24DT Cpu module :PLC

Control ligic: Toggle Q0.0.....Q0.7 .برای ساختمان بهتر از دو POU استفاده نموده ایم یک subroutine به نام Demo برای تسخیص کنترل های منطقی و یک Mainبرنامه که در آن Demo فراخوانی میشود .

ابتدا برنامه Kinco –Builderرا باز نمایید .

چنانچه لازم است به روش زیر پیش فرض هایی که در kinco-Builder استفاده شده است را اصلاح نمایید :

از نوار Tools گزینه Options را انتخاب کنید .پنجره ای باز میشود که در آن میتوانید بعضی از پیش فرضها را پیکره بندی نمایید .مانند زبان برنامه نویسی و ..این پیش فرض ها به صورت اتوماتیک ذخیره شده اند ،بنابراین فقط نیاز است یک بارآنها را قبل از تغییرات دیگر پیکره بندی نمایید .

۳)نحوه ایجاد پروژه جدید (تولید پروژه به یکی از روشهای زیر) :

از منوی اصلی برنامه ،روی گزینه Fileکلیک کنید (از نشانه  در نوارTool barنیز میتوان استفاده کرد) و new projectرا انتخاب نمایید .در صفحه ای که ایجاد میشود نام پروژه و فایلی را که برای ذخیره کردن پروژه در نظر میگیرید را انتخاب نمایید ،سپس روی گزینه saveکلیک کنید . بنابراین پروژه جدید ایجاد میشود .

برای این نمونه مسیر D:\tempو نام Example را انتخاب نمایید.

۴)تنظیمات و پیکره بندی سخت افزار : پس ازتولید پروژه نرم افزار به صورت اتوماتیک یک CPUپیش فرض در پنجره Options قرار میدهد .

تنظیمات سخت افزاری پروژه را میتوان در هر زمانی تغییرداد ، چون این تنظیمات و پیکره بندی سخت افزاری جزء لازم در هر پروژه میباشد لذا توصیه میشود که در ابتدا این تظیمات به صورت کامل انجام شود .

مراحل انجام تنظیمات سخت افزاری :

۱-بازکردن صفحهHardware(میتوانید پنجره hardwareرا به یکی از روشهای زیر بازکنید) :

a- روی گزینه Hardwareدر Manager دوبار کلیک کنید .

b-روی گزینه Hardware در Manager کلیک راست کنید .گزینه openرا انتخاب کنید .

برای مراحل توضیح داده شده به قسمت چگونگی اصلاح پارامترهای ارتباطی CPU مراجعه نمایید .

در این مثال یک Kinco-K306-24DT ماژول را با پارامترهای پیش فرض ان مورد استفاده قرار میگیرد .

۵)تولید برنامه نمونه :

زبان برنامه : نرم افزار kinco-builder دو نوع زبان برنامه نویسی LD(Ladder Diagram) و IL(Instruction List) را پشنیبانی میکند .

میتوانیدبا بازکردن گزینه projectدر قسمت Menuو انتخاب گزینه IL ویا LD زبان برنامه نویسی خود را عوض کنید .

(برای این مثال برنامه اصلی به نام "Main" و یک subroutineبه نام "Demo " که به زبان Ld نوشته میشود)

a) Mainبرنامه(Main Program) :زمانی که یک پروژه جدیدتوسط نرم افزار kinco-builder ایجاد شد ،اتوماتیک و به صورت همزمان یک Mainخالی برای برنامه نویسی ایجاد میکند که به صورت "Main"در قسمت Managerنمایش داده میشود .

b) ایجاد subroutine:

برای ایجاد یک subroutineمیتوان به سه روش عمل کرد :

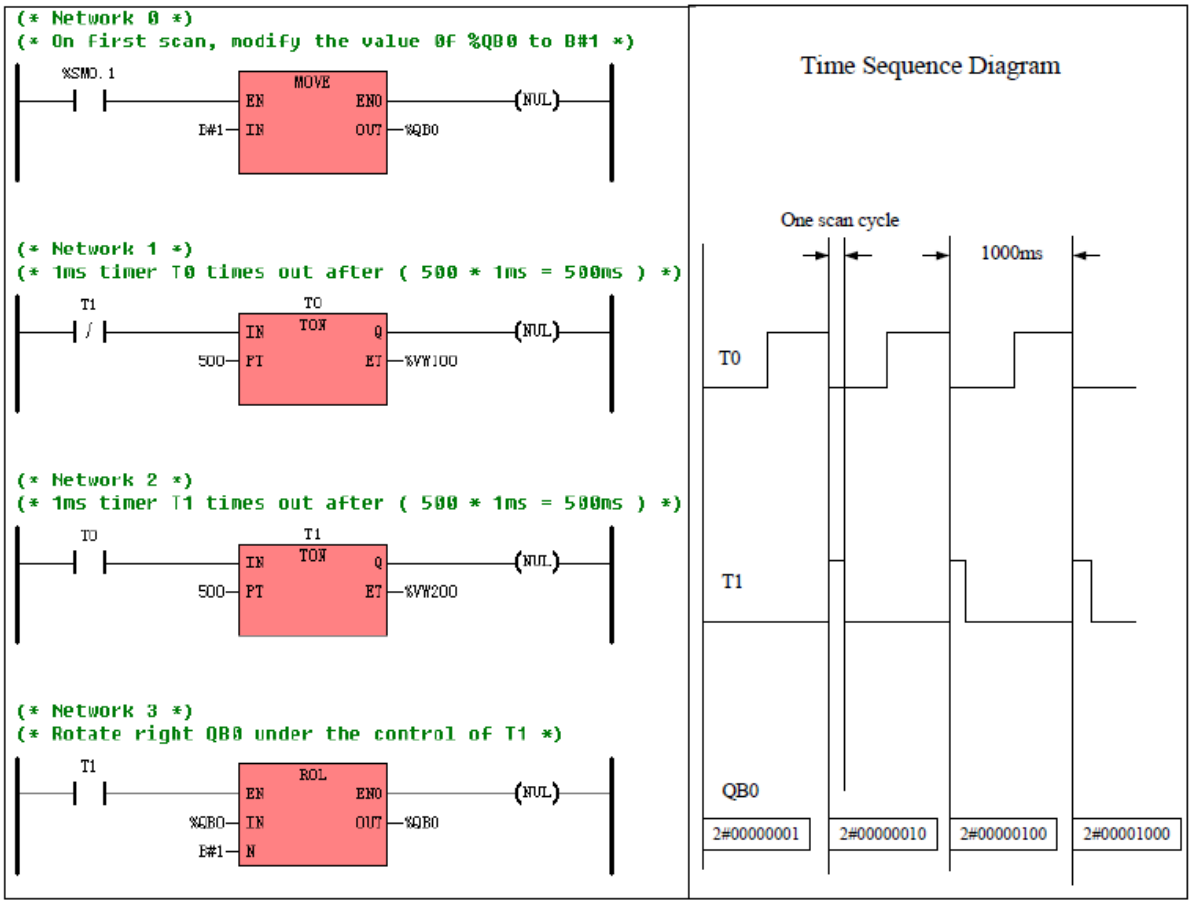
از منوی برنامه projectرا انتخاب نموده و روی گزینه new subroutineکلیک کنید .(project/new sunroutine)

از روی نوار toolbar روی نشانه  کلیک کنید .

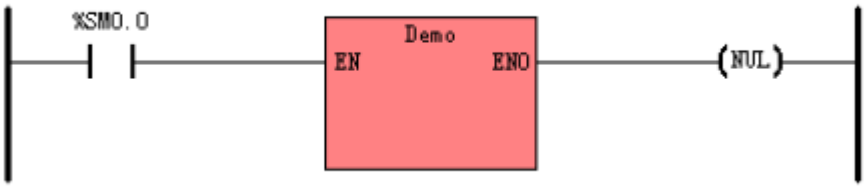
روی گزینه PROGRAMدر صفحه Manager راست کلیک کنید وگزینه New sunroutineرا انتخاب نمایید .

به این ترتیب یک Subroutineجدید ایجاد میشود که به صورت تعریف شده در نرم افزار به نام " SBR_0"میباشد .حال میتوانید دستورات مورد نظر را (مانند تصویر نمایش داده شده) در این Subroutineوارد نمایید .

پس از وارد نمودن دستورا ت میتوان نام Subroutineرا تغییرداد .برای این منظور ابتدا صفحه Subroutineرا ببندید ،سپس روی گزینه SBR00 در صفحه Managerراست کلیک نمایید و گزینه Renameرا برای انجام تغییر به نام مورد نظر (به عنوان مثال Demo) انتخاب کنید .ویا در همین مسیر (پس از راست کلیک روی گزینه SBR00) گزینه Propertyرا انتخاب نمایید .



C: Mainبرنامه : پس از اتمام 'Demo' Subroutine نیاز هست که به Main برنامه برگردیم و این دستوررا به برنامه اضافه کنیم .(مانند تصویر زیر)



همان طور که در تصویر بالا مشاهده میشود ،زیربرنامه ایجاد شده باید در mainبرنامه فراخوانی شود تا قابل اجرا باشد.

Compile:5.4.10 کردن برنامه :

پس از آنکه پروژه به صورت کامل انجام شد این نیاز وجود دارد که برنامه compileشود .هنگامی که پروژه compileمیشود نرم افزارابتدا آخرین تغییرات پروژه را به صورت اتوماتیک ذخیره مینماید .

برای compileکردن برنامه میتوان به صورت زیر عمل کرد :

۱-از صفحه منو روی گزینه PLCکلیک کنید ،سپس گزینه Compile Allرا انتخاب کنید .

۲-روی نشانه  در نوار Tool Barکلیک کنید .

۳-میتوان با فشردن کلید F7برنامه را Compileکرد .

پیغام های مربوط به copileبرنامه در نوار compile در صفحه خروجی قرار میگیرد .

چنانچه در برنامه errorوجود داشته باشد برای پیدا کردن code و خط مورد نظر مربوط به errorمیتوان روی پیغام error در صفحه compile دو بار کلیک کنید . باید تغییرات لازم را برای برطرف کردن error انجام دهید ، تا زمانی که پروژه با موفقیت کامپایل شود.

Download:5.4.11کردن برنامه :

چنانچه لازم باشد میتوانید پارامترهای ارتباطی پورت سریال کامپیوتر خود را در پنجره communication به صورت مناسب تنظیم کنید.

میتوان به سه روش برنامه را بر روی PLCدانلود کرد :

۱-از قسمت Menu گزینه PLCرا انتخاب نمایید و روی Downloadکلیک کنید . (PLC/Download)

۲- روی نشانه  در نوار Tool Barکلیک کنید .

۳-میتوان با فشردن کلید F8برنامه را دانلود کرد .

چنانچه CPU module در مد RUNباشد ،پیغامی روی صفحه نمایش داده میشود مبنی بر این که این مد را به مد STOPتغییر دهید .برای این که در مد STOPقرار بگیرید روی گزینه Yes کلیک نمایید.

پس ازاین که برنامه دانلود شد CPU moduleدر مد RUNقرار میگیرد و LEDهای نشانگر وضعیت Q0.0.....Q0.7 به ترتیب و به صورت چرخشی خاموش و روشن میشوند .

بنابراین اولین پروژه kinco-k3کامل شده است .

توجه داشته باشید :

میتوانید اجرای برنامه را به صورت همزمان بر روی کامپیوتر ملاحظه کنید .(برنامه را به صورت online،مانیتور کنید)برای این منظور باید بر روی deibug از قسمت منو کلیک کنید .سپس گزینه monitor را انتخاب کنید و یا میتوانید بر روی نشانه  در قسمت Tool Barکلیک کنید . در این صورت نرم افزار تمامی مقادیر متغیر هایی که در برنامه وجود دارد را نمایش خواهد داد .

برای تغییر وضعیت به مد STOPمیتوان کلید های عملگر را در وضعیت STOPقرار داد ویا بر روی Debugدر قسمت Menuکلیک کرده و گزینه STOPرا انتخاب کنید .

فصل ششم :آشنایی با بخش های دیگر نرم افزار KincoBuilder:

در این بخش به توضیح بخش های مختلف این نرم افزار میپردازیم .

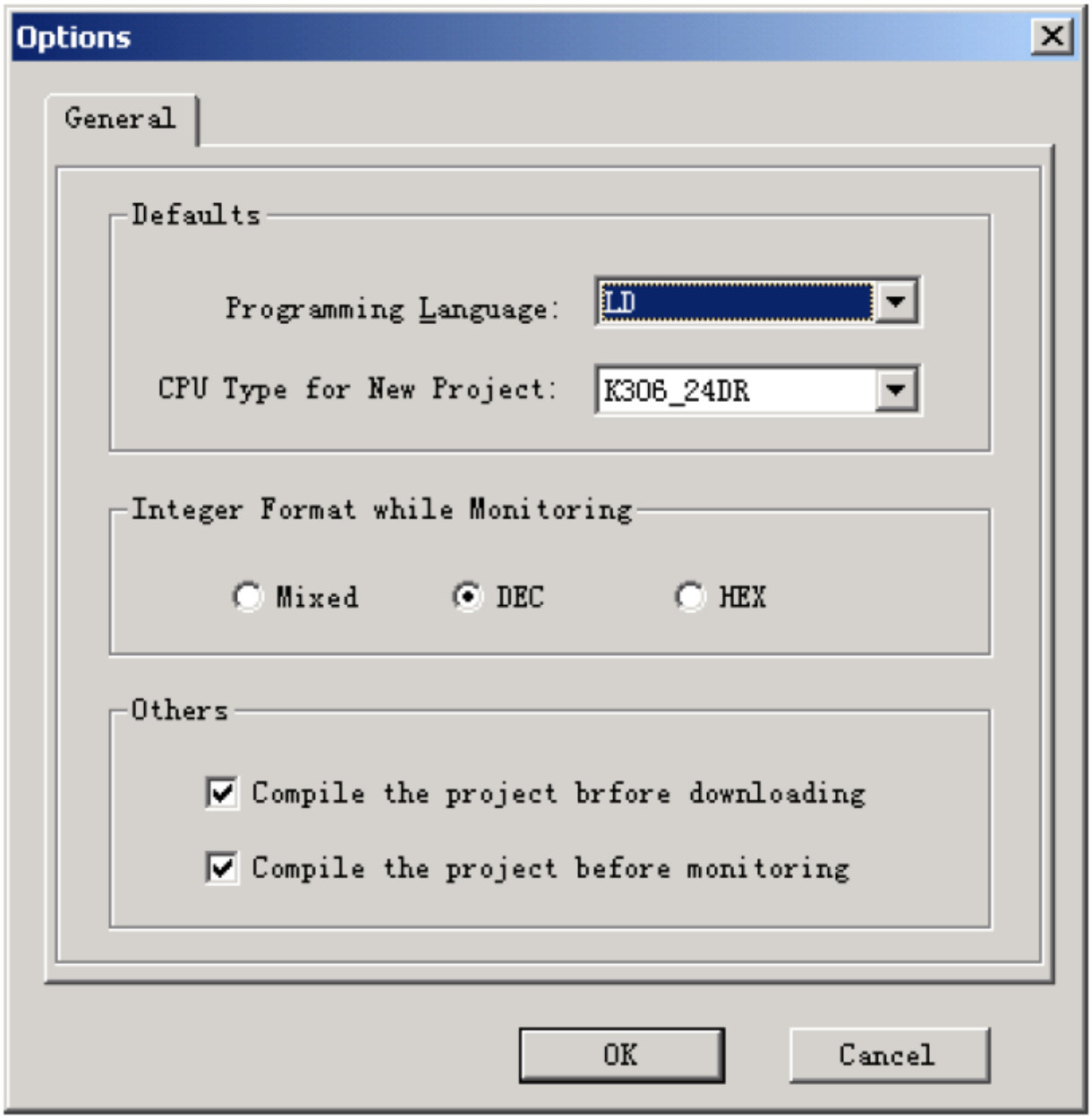
با توجه به مطالب مختصری که در قسمت قبل توصیح داده شد ، این بخش در درک بهتر و جامع تر از نرم افزار KincoBuilderبه شما کمک خواهد کرد.

۶.۱:پیکره بندی بخش های عمومی نرم افزار:

همواره در هر پروژه نیاز میباشد که برخی از بخش های نرم افزار را قبل از شروع هر کاری مطابق با نیازهای پروژه پیکره بندی نمایید .مانند زبان برنامه نویسی ،نوع CPU متناسب با پروژه مورد نظر. شما تنظیمات مربوط به این پارامترها را تنها یک بار در ابتدای برنامه انجام داده ،نرم افزار KincoBuilder این تغییرات را به صورت اتوماتیک ذخیره مینماید .

۶.۱.۱:منوی OPTION:

در قسمت منوی نرم افزار برروی گزینه TOOLS کلیک کرده و گزینه OPTIONS را انتخاب کنید .یک پنجره مانند تصویر زیر باز میشود:



Defaults

Programming Language: در این بخش میتوانید زبان برنامه نویسی را که میخواهید نرم افزار همیشه به صورت پیش فرض در نظر بگیرد (LD یا IL) انتخاب نمایید .

CPU Type for New Project: در این قسمت میتوانید نوع CPUرا که میخواهید زمانی که پروژه جدید ایجاد مینمایید ، نرم افزار به صورت پیش فرض در نظر بگیرد انتخاب نمایید.

Integer Format While Monitoring:

در این قسمت میتوان مشخص کرد که در زمان مانیتور کردن مقادیر integer با چه فرمتی نمایش داده شود .

Mixed: با انتخاب این گزینه مقادیر INT وDINT در هنگام مانیتورینگ به صورت Decimal نمایش داده میشود و مقادیر BYTE ، WORD و DWORD به فرمت Hexadecimal نمایش داده میشوند.

DEC: با انتخاب این گزینه تمام مقادیر به صورت DEC نمایش داده میشود .

HEX: با انتخاب این گزینه تمام مقادی به صورت Hexadecimal نمایش داده میشود .

:Others

:Compile the project before downloading

با انتخاب این گزینه نرم افزار قبل از دانلود برنامه بر روی CPUبه صورت اتوماتیک برنامه جاری را کامپایل مینماید .

:Compile the project before monitoring

با انتخاب این گزینه نرم افزار قبل از مانیتورینگ به صورت اتوماتیک برنامه را کامپایل میکند .

پیکره بندی سخت افزار :

در این بخش نحوه تنظیمات و پیکره بندی سخت افزار را توضیح خواهیم داد .

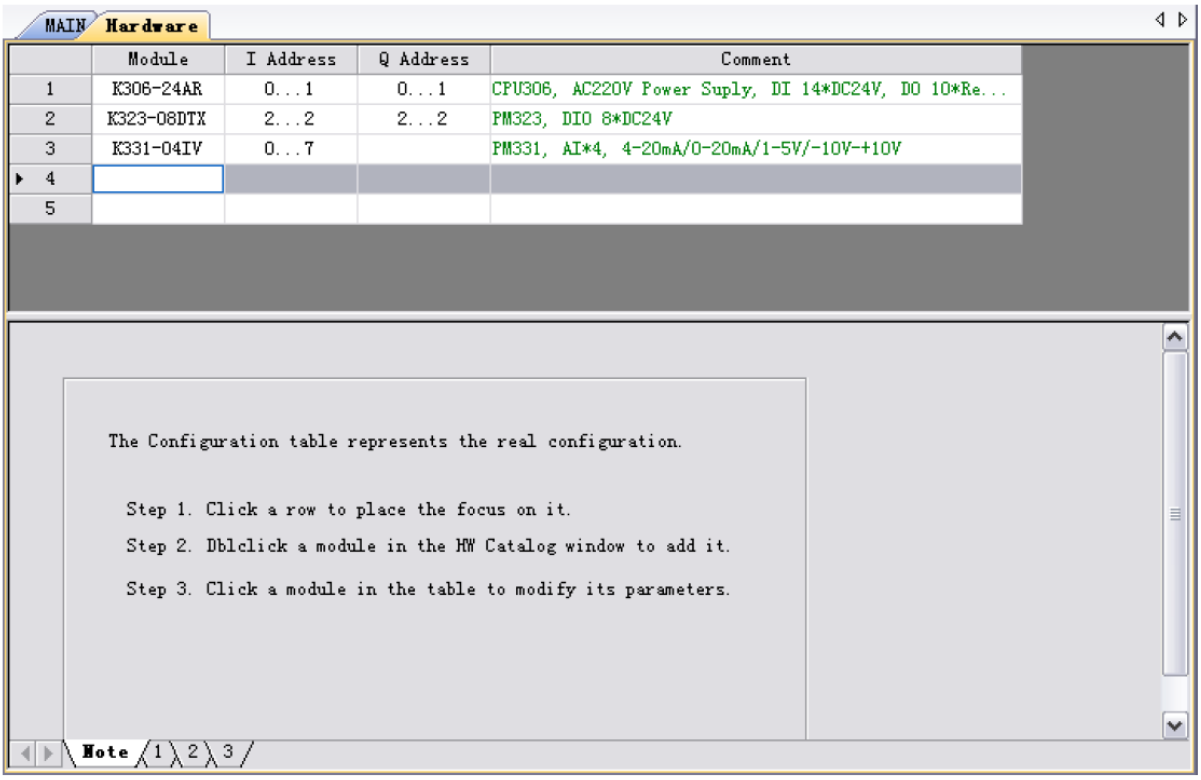
در یک پروژه همواره توصیه میشود قبل از انجام هر کاری ابتدا تنظیمات مربوط به سخت افزار انجام شود. زمانی که یک پروژه جدید ایجاد میشود نرم افزار به صورت اتوماتیک CPU از پیش تعیین شده را (همان طور که در بخش قبل توضیح داده شد میتوان مدل این CPUپیش فرض را در منوی Tools /Options تغییر داد) به صورت پیش فرض در نظر میگیرد .

برای آنکه بتوانید سخت افزار مناسب برای پروژه مورد نظر را تنظیم کنید باید با توجه به مطالب بیان شده در پایین عمل نمایید :

Hardware:6.2 :

پنجره Hardware(سخت افزار) در نرم افزار KincoBuilder مانند تصویر پایین از دو بخش تشکیل شده است :

(میتوان با دو بار کلیک کردن بر روی گزینه Hardware در قسمت Manager این جدول را مشاهده نمود)



۶.۲.۱:جدول پیکره بندی :

قسمت بالای پنجره سخت افزار جدولی قرار دارد که لیستی از جزئیات و مشخصات CPU و کارت های افزایشی را نشان میدهد به این جدول، جدول پیکربندی نامیده میشود .این جدول پیکر بندی واقعی سخت افزار را نشان میدهد ،به عبارت دیگر چیدمان وپیکره بندی سخت افزار در این جدول باید عینا مطابق با آنچه در سیستم کنترلی انجام میدهید باشد .

۶.۲.۲: جدول پارامترها :

در قسمت پایین پنجره سخت افزار جدول پارامترها قرار دارد . زمانی که هر یک از ماژول را در جدول پیکره بندی انتخاب نمایید جدول پارامترهای مربوط به آن ماژول در پایین صفحه نمایش داده میشود .

تنظیمات مربوط به جدول پارارمترها باید بر روی CPUدانلود شود .

۶.۳:چگونگی باز کردن پنجره سخت افزار :

همان طور که قبلا توضیح داده شده بود بادوبار کلیک کردن بر روی گزینه Hardware در پنجره Manager و یا با راست کلیک بر روی گزینه Hardwareو انتخاب Open میتوان پنجره سخت افزار را مشاهده کرد .

۶.۳.۱:اضافه کردن و حذف کردن ماژول :

اضافه کردن ماژول :

زمانی که میخواهید یک ماژول را در برنامه اضافه نمایید در جدول پیکره بندی روی سطری را که میخواهید ماژول مورد نظر در آن قرار گیرد کلیک کرده (به عبارت دیگر سطری را که میخواهید ماژول مورد نظر در آن قرار گیرد انتخاب میکنید) .اگر ماژولی در آن سطر قرار گرفته بود ابتدا باید حذف گردیده تا بتوان ماژول جدید را جایگزین کرد .

سپس از قسمت HW Catalog بر روی ماژول مورد نظر دو بار کلیک نمایید .به این ترتیب ماژول مورد نظر در سطری که انتخاب نمودید قرار میگیرد

باید توجه داشته باشید که در سطر ۱ تنها CPU میتوان ماژول CPUرا قرار داد و در بقیه سطرها ماژول های افزایشی قرار میگیرد .بنابراین قرار دادن ماژول های افزایشی در سطر ۱ و یا قرار دادن ماژول CPU در سطر های دیگر غیر ممکن میباشد .

همان طور که میدانیم ماکزیمم تعداد کانال های ورودی و خروجی که میتوان برای هریک از CPU ها با استفاده از ماژول های افزایشی ایجاد نمود محدود میباشد . چنانچه تعداد این کانال های ورودی و خروجی با استفاده از کارتهای افزایشی به حد ماکزیممم مجاز برسد ،نرم افزار به صورت اتوماتیک اجازه اضافه کردن ماژول افزایشی دیگری رادر جدول پیکر بندی به کاربر نمیدهد.

حذف کردن ماژول :

برای حذف کردن ماژول از جدول پیکره بندی میتوان به دو طریق عمل کرد :

برروی ماژولی که باید از جدول حذف شود کلیک کرد و سپس با استفاده از دکمه Del آن را خذف کرد .

بر روی ماژول مورد نظر راست کلیک کرده و گزینه Removeرا انتخاب کنید .

۶.۴:تنظیمات پارامترهای هر ماژول در جدول پارامترها :

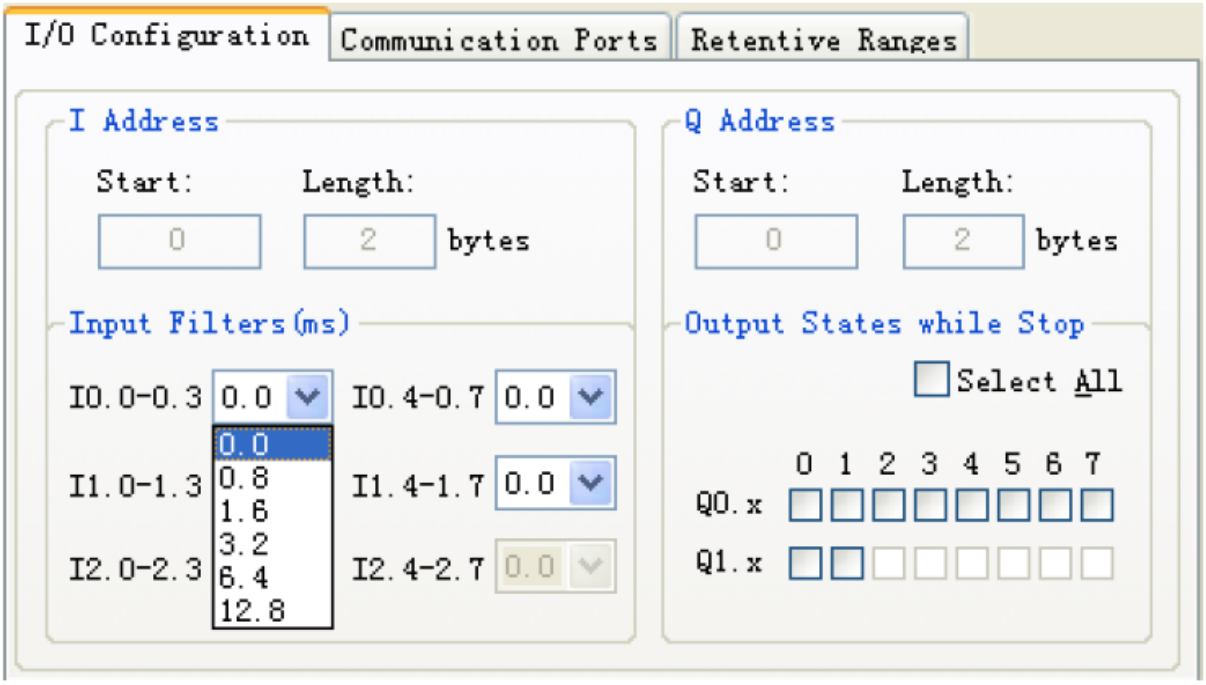
زمانی که ماژول های مورد نظر را درجدول پیکره بندی قرار دادید میتوان پارامترهای هر یک را در جدول پارامترها تعیین کرده و تغییرات لازم را اعمال نمود .

در جدول پارامترها بر روی ماژول CPU کلیک کرده و آن را انتخاب نمایید ، سپس پنجره پارامترهای مربوط به CPU را متوانید در پایین مشاهده کنید .در این پنجره میتوانید تنظیمات مورد نظر برای پارامترهای هر ماژول را مشخص کنید.

پارامترهای CPU:

۱-نوار I/O Configuration:

در این قسمت کاربر میتواند پارامترهای مربوط به ورودی و خروجی های CPU را تنظیم نماید .به تصویر زیر توجه نمایید :



INPUT: در این قسمت کاربر میتواند تنظیمات مربوط به کانال های ورودی دیجیتال روی ماژول CPU را انجام دهد .

I Address: در این قسمت آدرس بایت شروع کانال های ورودی دیجیتال را در فضای I مشخص میکند که همیشه این آدرس ۰ میباشد .

Input Filter: در این قسمت کاربر تاخیر زمانی مورد نظر را برای کانالهای ورودی دیجیتال تعیین میکند که براساس میلی ثانیه میباشد .این تاخیر زمانی برای نویز های ورودی موثر میباشد .هنگامی که وضعیت در ورودی تغییر میکند و این تغییر پس از گذشت مدت زمان مشخص شده در این قسمت همچنان باقی بماند به عنوان ورودی پذیرفته میشود.

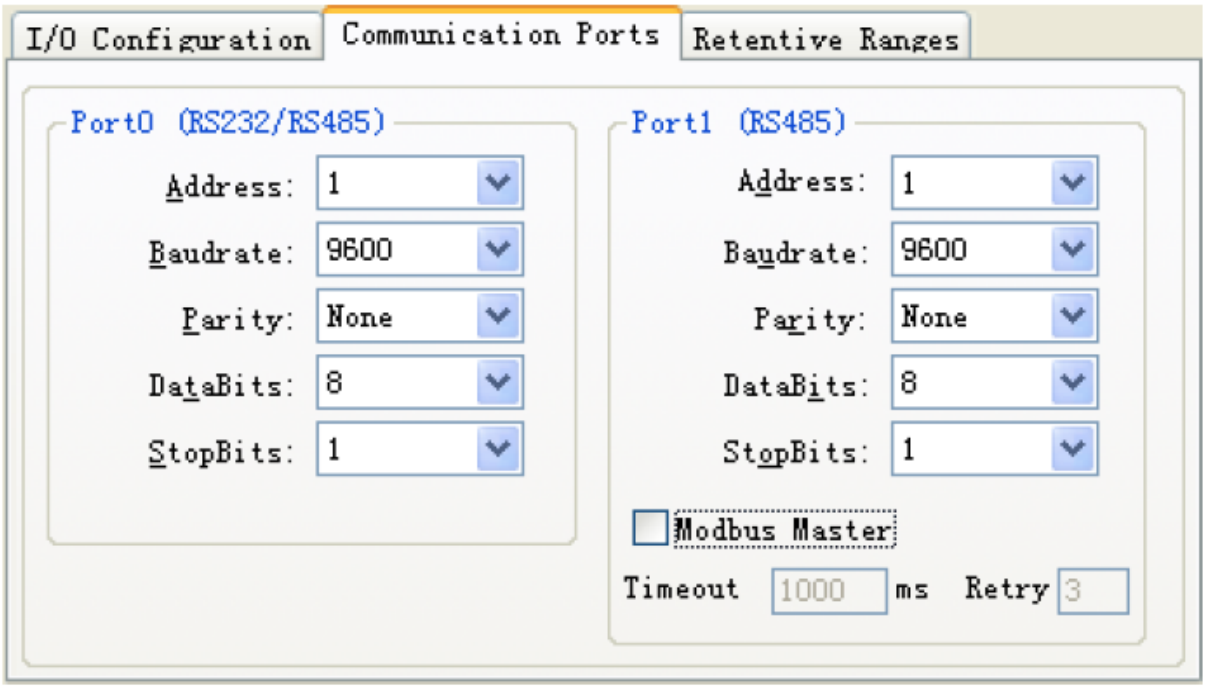
Output: در این قسمت کاربر میتواند تنظیمات مربوط به کانال های خروجی دیجیتال روی ماژول CPUرا انجام دهد .

Q Address: در این قسمت آدرس بایت شروع کانال های خروجی دیجیتال را در فضای Q مشخص میکند که همیشه این آدرس ۰ میباشد .

Output States While Stop: در این قسمت کاربر میتواند وضعیت خروجی را در زمانی که CPUدر مد STOP قرار میگیرد مشخص نماید .زمانی که تیک گزینه هریک از Q ها زده شود آن خروجی در زمانی که CPU در مد STOPقرار میگیرد وضعیت ۱ خواهد داشت .

۲-نوار Communication Port:

در این قسمت کاربر میتواند تنظیمات مربوط به پورت های سریال CPU(پورت ۰ و پورت ۱) را انجام دهد .



PORT 0:

این قسمت تنظیمات مربوط به پورت ۰ (که این پورت مربوط به پروگرام کردن CPU میباشد) است .

پورت ۰ ، پورت سریال RS232 میباشد .

Address: در این قسمت آدرس مورد نظر پورت ۰ انتخاب میشود .همچنین این آدرس میتواند شماره Modbus RTU slave نیز باشد . این آدرس در شبکه به صورت یک آدرس انحصاری میباشد.

Baudrate: در این قسمت baudrate مورد نظر را میتوان انتخاب کرد (1200,2400,4800,9600,19200,38400 bps)

Parity: در این قسمت parityمشخص میشود(NO,odd ,even)

DataBits:در این قسمت میتوان تعداد بیت های هر بایت را در ارسال و دریافت مشخص کرد(۸)

STOPBits: این قسمت مربوط به Stop bitبوده که ۱ میباشد .

:PORT 1

پورت ۱ ، RS485بوده و این قسمت تنظیمات مربوط به پورت ۱ میباشد .برخی از CPUها دارای یک پورت سریال(PORT 0) و برخی دیگر د ارای دو پورت (PORT 1 ,PORT 0)میباشند.

Modbus Master: چنانچه قسمت Modbus Master انتخاب شده باشد ، این پورت به عنوان Modbus masterدر برنامه عمل میکند.

Timeout:در این قسمت مدت زمان Timeout را برای Modbus master تعیین میکنید.

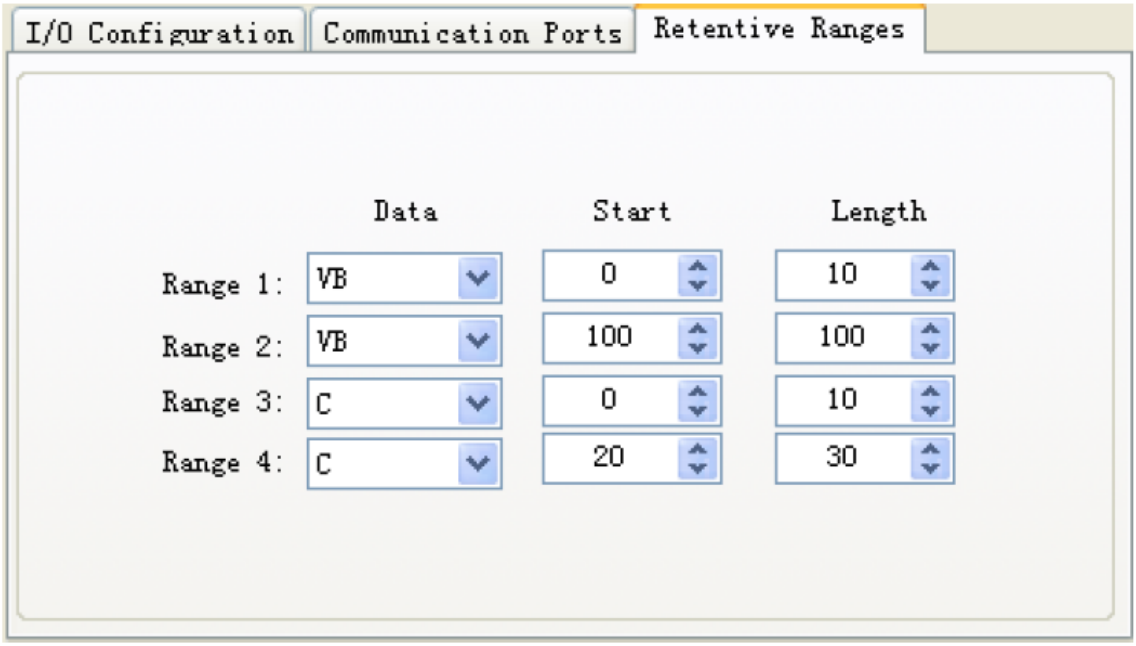
Retry: در این قسمت تعداد دفعات ارتباط مجدد را مشخص میکنید .چنانچه masterیک فریم نادرست از slave دریافت نماید ، مجددا به مقدار Retry times برای ارتباط مجدد با تلاش میکند.

سایر پارامترها مشابه پارامترهای پورت ۰ میباشد .

نوار Retentive Range:

در این قسمت شما میتوانید ۴ رنج از حافظه RAM را که میخواهید اطلاعات آن با قطع تغذیه CPU حفظ شود را مشخص نمایید.

در این حالت هنگامی که تغذیه CPU قطع میشود اطلاعاتی که در حافظه RAM قرار دارد توسط یک خازن داخلی نگهداری میشود . تنها حافظه هایی که در این قسمت تعیین میشوند در برگشت مجدد تغذیه بدون تغییر باقی میمانند.



Range 1(رنج ۱): در این قسمت میتوانید حافظه هایی از فضای V ویا کانتر به عنوان رنج اول انتخاب نمایید .در فضای کانتر فقط مقدار درحال شمارش میتواند retentive شود.

Start: در این قسمت آدرس بایت شروع در رنج اول تعیین میگردد .

Length: در این قسمت طول رنج ۱ (واحد آن بایت است) تعیین میگردد .

Range 4 ,Range 3 ,Range 2 نیز مشابه توضیحات بالا میباشند.

در تصویر بالا به عنوان مثال داده های حافظه های :

رنج ۱: VB0~VB9

رنج ۲:VB100~VB199

رنج ۳: C0~C9

رنج ۴ : C20~C49

پس از قطع تغذیه و وصل مجدد آن بدون تغییر باقی میمانند.

پارامترهای ماژول ورودی دیجیتال:

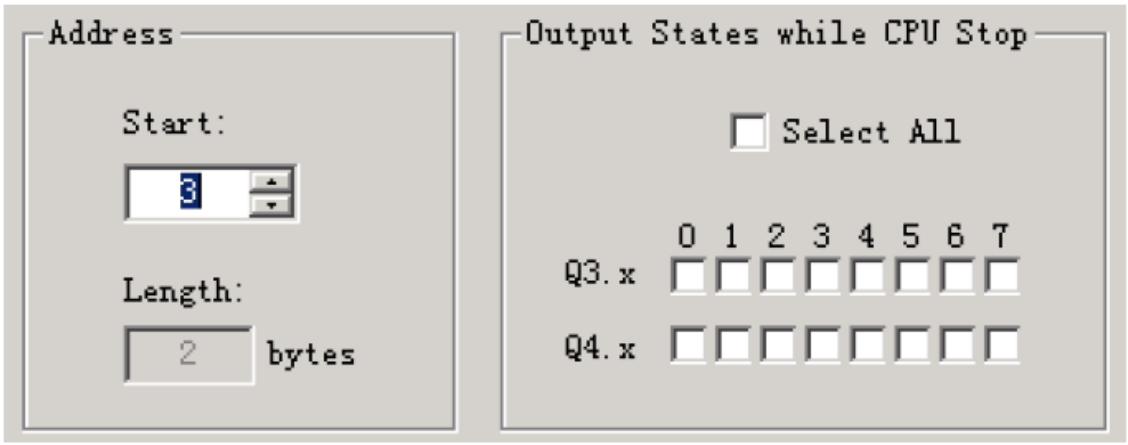
Adrress:

Start: در این قسمت میتوان آدرس بایت شروع ماژول ورودی دیجیتال را در فضای حافظه I مشخص کرد .آدرس برای کانالهای این ماژول بر اساس آدرس شروع میباشد.

Length: این قسمت طول رنج آدرس ماژول میباشد .این قسمت ثابت بوده و بستگی به تعداد کانال های دیجیتال کارت انتخابی دارد .

همان طور که در تصویر بالا نشان داده شده است کارت انتخابی دارای ۸ کانال ورودی میباشد که آدرس شروع آن IB3 میباشد .بنابراین آدرس کانال های این کارت I3.0~i3.7 خواهد بود.

پارامترهای ماژول خروجی دیجیتال:



Adrress:

Start: در این قسمت میتوان آدرس بایت شروع ماژول خروجی دیجیتال را در فضای حافظه Q مشخص کرد .آدرس برای کانالهای این ماژول بر اساس آدرس شروع میباشد.

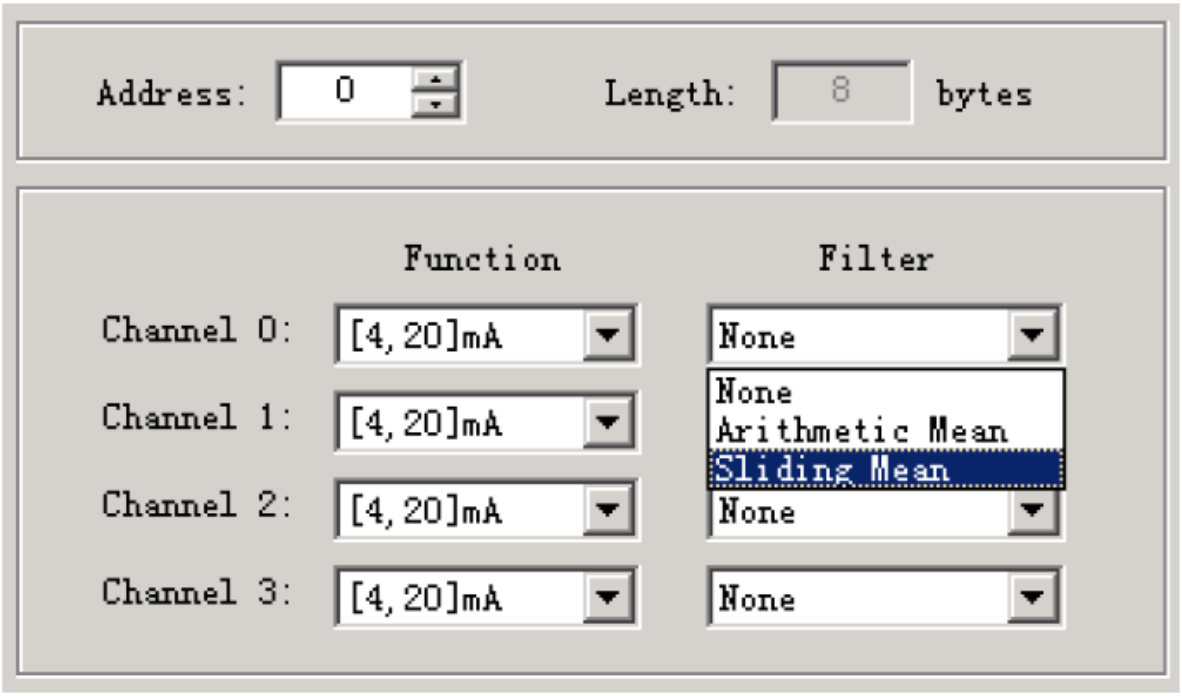
Length: این قسمت طول رنج آدرس ماژول میباشد .این قسمت ثابت بوده و بستگی به تعداد کانال های دیجیتال کارت انتخابی دارد .

همان طور که در تصویر بالا نشان داده شده است کارت انتخابی دارای ۱۶ کانال خروجی میباشد که آدرس شروع آن QB3 میباشد .بنابراین آدرس کانال های این کارت Q3.0~Q4.7خواهد بود.

Output State while CPU Stop:

در این قسمت میتوان وضعیت هر کانال خروجی دیجیتال را در حالتی که CPUدر وضعیت Stop قرار دارد تعیین کرد . با انتخاب هرکدام از این خانه ها کانال دیجیتال مربوطه در زمانی که CPU در وضعیت stopقرار میگیرد وضعیت ۱ خواهد داشت .

پارامترهای ماژول ورودی آنالوگ:



:Adrress

Start: در این قسمت میتوان آدرس بایت شروع (آدرس اولین کانال) ماژول ورودی آنالوگ را در فضای حافظه AI مشخص کرد .آدرس بقیه کانالهای این ماژول بر اساس آدرس شروع میباشد.هر آدرس ۲ بایت را اشغال مینماید .باید دقت داشته باشید که مقداری که در این قسمت قرار میگیرد باید یک عدد زوج باشد .

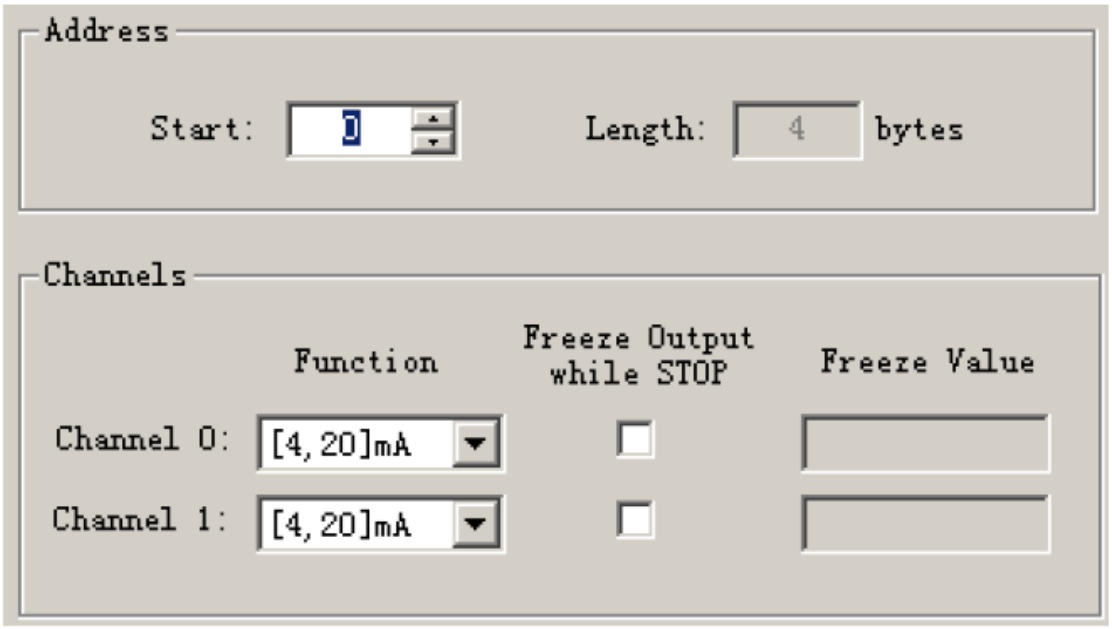
Length: این قسمت طول رنج آدرس ماژول میباشد .این قسمت ثابت بوده و بستگی به تعداد کانال ها ی آنالوگی کارت انتخابی دارد .

همان طور که در تصویر بالا مشاهده مینمایید کارت انتخابی دارای ۴ کانال ورودی آنالوگی میباشد که آدرس شروع آن AIW0 میباشد .بنابراین آدرس سایر کانالها به صورت AIW2,AIW4,AIW6خواهد بود.

FUNCTION: همان طور که در تصویر دیده میشود میتوان این کارت ها را به صورت جریانی و یا ولتاژی تنظیم نمود .در این قسمت با توجه به سیگنال آنالوگ ورودی نوع کارت انتخاب میشود (بنابراین انتخاب جریانی یا ولتاژِ بودن کارت از طریق نرم افزار انجام میشود)

FILTER:در این قسمت میتوان یک فیلتر نرم افزاری برای ورودی های آنالوگی در نظر گرفت.

پارامترهای ماژول خروجی آنالوگ:



Adrress:

Start: در این قسمت میتوان آدرس بایت شروع (آدرس اولین کانال) ماژول خروجی آنالوگ را در فضای حافظه AQ مشخص کرد .آدرس بقیه کانالهای این ماژول بر اساس آدرس شروع میباشد.هر آدرس ۲ بایت را اشغال مینماید .باید دقت داشته باشید که مقداری که در این قسمت قرار میگیرد باید یک عدد زوج باشد .

Length: این قسمت طول رنج آدرس ماژول میباشد .این قسمت ثابت بوده و بستگی به تعداد کانال ها ی خروجی آنالوگی کارت انتخابی دارد .

همان طور که در تصویر بالا مشاهده میکنید کارت انتخابی دارای دو کانال آنالوگی خروجی با آدرس شروع AQW0 میباشد .بنابراین آدرس کانال دیگر AQW2 میباشد.

FUNCTION:در این قسمت میتوان نوع سیگنال خروجی آنالوگی (جریانی یا ولتاژی) را انتخاب کرد.

FREEZE OUTPUT WHILE STOP:با انتخاب این گزینه میتوان مقدار مشخصی را برای خروجی آنالوگ در زمانی که CPU در وضعیت STOPقراردارد ، تعیین کرد .

FREEZE VALUE: در این قسمت میتوان مقدار ثابتی را که میخواهیم خروجی آنالوگی در زمانی که CPU درحالت STOP داشته باشد تعیین کرد .

۶.۵:جدول مقدار اولیه (Initial Data Table):

در این جدول میتوانید یک مقدار عددی اولیه رابرای متغیرهای Byte ,WORD,DWORD,INT,DINT,REAL در فضای حافظه v اختصاص داد.

زمانی که تغذیه cpuبرقرار میشود ، ماژول CPU ابتدا مقادیر اولیه تعیین شده در این جدول را تحلیل کرده و سپس سیکل اسکن را شروع میکند .

Initial Data

	Address	Value	Value	Value	Value
1	%VB0	B#1	B#2		
2	%VW10	2	3	4	
3	%VD100	DI#100	DI#200	DI#2000	DI#2456
4					
5					

نحوه باز کردن و پارامتردهی جدول INITIAL DATA:

با دوبار کلیک کردن بر روی [INITIAL DATA] در پنجره Manager

بر روی [INITIAL DATA] راست کلیک کرده و گزینه open را انتخاب کنید .

همان طور که در تصویر بالا مشاهده میکنید جدول فوق دارای ۵ ستون میباشد :

یک ستون Address و چهار ستون Value

Address: در این ستون آدرس حافظه مورد نظر در فضای V مشخص میگردد.

Value: در این ستون ها میتوان مقادیر مختلف قرار داد . مقدار value در ستون دوم به حافظه با آدرسی که در address مشخص شده است ، اختصاص داده میشود .و مقادیر valueدر ستون های سوم ، چهارم و پنجم به سایر حافظه هایی که به ترتیب بعد از حافظه Addressقرار دارند اختصاص میبابند.

به عنوان مثال همان طور که در تصویر بالا مشاهده میکنید ، مقدار B#1به حافظه VB0اختصاص داده شده است .همچنین مقدار B#2به حافظه VB1اختصاص میابد.

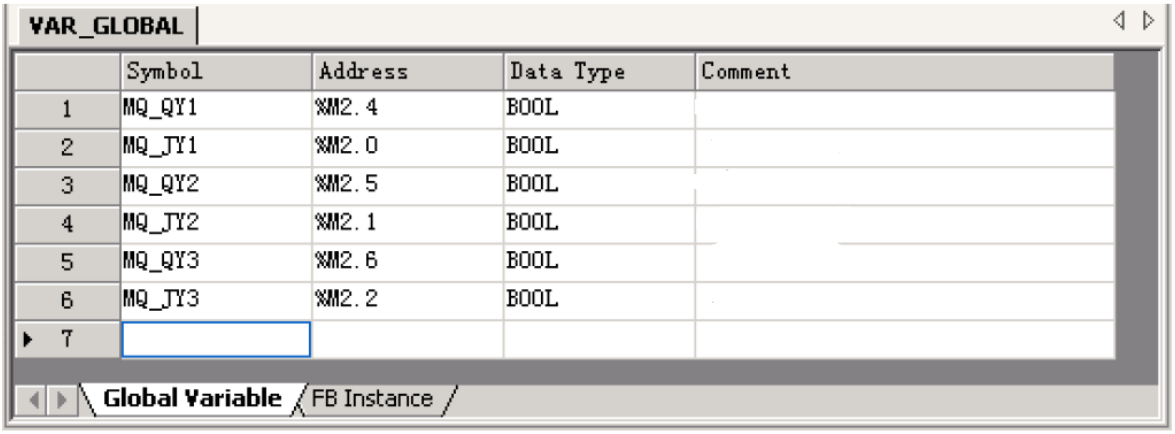
همچنین در ردیف دوم مقادیر 2,3,4 به ترتیب به حافظه های VW10,VW12,VW14 اختصاص میابد.

جدل متغیر های عمومی(Global variable Table) :

این جدول شامل دو بخش میباشد : Global variable tab و FB instance tab

۶.۶: جدول Global variable :

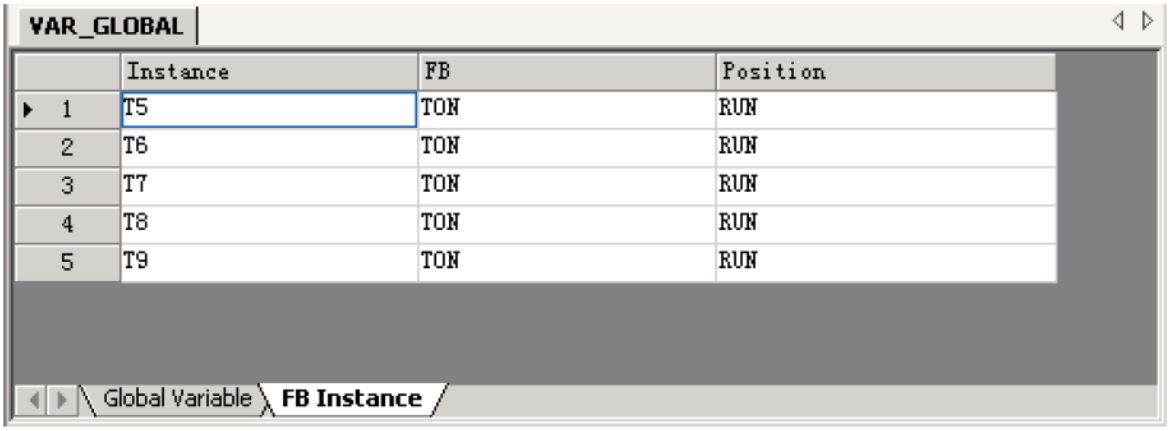
با استفاده از این قسمت میتوان برای تمامی متغیر های تعریف شده در برنامه یک سمبل تعریف کرد و در تمام طول برنامه از آن سمبل استفاده نمود.


VAR_GLOBAL

	Symbol	Address	Data Type	Comment
1	MQ_QY1	%M2.4	BOOL	
2	MQ_JY1	%M2.0	BOOL	
3	MQ_QY2	%M2.5	BOOL	
4	MQ_JY2	%M2.1	BOOL	
5	MQ_QY3	%M2.6	BOOL	
6	MQ_JY3	%M2.2	BOOL	
7				

Global Variable | FB Instance

:FB Instance tab


VAR_GLOBAL

	Instance	FB	Position
1	T5	TON	RUN
2	T6	TON	RUN
3	T7	TON	RUN
4	T8	TON	RUN
5	T9	TON	RUN

Global Variable | FB Instance

زمانی که در برنامه از Function block ها استفاده شود این دستورات توسط نرم افزار به صورت اتوماتیک شناسایی شده و اطلاعات آنها در این قسمت وارد میشود . اطلاعات این بخش از جدول غیرقابل تغییر میباشد .

باز کردن جدول Global variable:

۱)با دو بار کلیک کردن بر روی [Global Variable] در پنجره Manager

۲)با راست کلیک بر روی [Global Variable]و انتخاب گزینه open

۳)در منوی Project انتخاب گزینه Global variable

قسمت Global variable tab دارای ۴ ستون میباشد :

Symbol: در این قسمت نام سمبل مورد نظر را میتوان نوشت .

Address: در این قسمت آدرس حافظه مورد نظر را میتوان وارد کرد .

Data Type: میتوان در این قسمت نوع و فرمت مورد نظر را انتخاب کرد . (نرم افزار با انتخاب آدرس مورد نظر این قسمت را به صورت اتوماتیک ایجاد مینماید)

Comment: در این قسمت میتوان توضیحات مربوط به آن داده وسمبل را نوشت .

پس از انتخاب سمبل خاص برای یک آدرس مشخص در تمامی برنامه میتوان به جای استفاده از آدرس مورد نظر سمبل آن را به کار برد .

۶.۷:جدول Cross Reference:

این جدول تمامی متغیرهای مورد استفاده در پروژه و نیز قسمتی از برنامه که در آن به کار رفته است و همچنین شبکه مربوط به آن و چگونگی دسترسی به آن (read ویا write شدن) نشان داده میشود .

این جدول هنگامی که میخواهید بدانید چه آدرسی ، در کجای برنامه و با چه سمبلی به کار رفته است بسیار مفید میباشد.

اطلاعات این جدول پس از اولین کامپایل ایجاد شده و پس از هر کامپایل به صورت اتوماتیک به روز رسانی میشود.

Cross Reference

Index	Address	Symbol	POU	Position	Read/Write
0	%M1.3		MAIN	Network 0	Read
1	%M1.3		MAIN	Network 1	Read
2	%M1.4		MAIN	Network 0	Read
3	%M1.4		MAIN	Network 1	Read
4	%M10.0		MAIN	Network 0	Write
5	%M10.0		MAIN	Network 1	Write

Cross Reference

Address: تمامی آدرس حافظه هایی را که در برنامه استفاده شده است را نشان میدهد.

Symbol: سمبل مربوط به هر آدرس را نشان میدهد.

POU: قسمتی از برنامه را که آدرس مورد نظر در آن به کار رفته است نشان میدهد.

Position:شبکه و خطی را که آدرس مورد نظر در آنجا استفاده شده است نشان میدهد.

Read/Write: نشان میدهد که حافظه درموقعیت نشان داده شده خوانده شده ویا نوشته شده است .

باز کردن جدول Cross reference:

از منوی Projectبا انتخاب گزینه Cross reference

با کلیک بر روی آیکون  در قسمت Toolbar

با کمک Alt +C

جدول۶.۸: Status chart

با کمک این جدول میتوان متغیر هایی را که در برنامه استفاده شده است ، پس از دانلود برنامه بر روی PLC مانیتور و یا Force کرد.

Status Chart

	Address	Symbol	Format	Current Value	New Value
1	%M1.3		BOOL	2#0	
2	%M1.4		BOOL	2#0	
3	%VW4		Signed	100	
4	%VW6		Hexadecimal	W#16#64	
5					

Address: در این قسمت آدرس متغیری را که قرار است مانیتور و Force شود را وارد میگردد.

Symbol: در این قسمت سمبل آدرس مورد نظر نشان داده میشود.

Format: در این قسمت فرمت مقدار جاری و مقدار جدید انتخاب میشود.

Current value: مقدار فعلی آ درس مورد نظر را نشان میدهد.

New value: در این قسمت میتوان مقدار مورد نظری را که میخواهید در زمان مانیتورینگ به آدرس مورد نظر Force کنید وارد نمایید.

نحوه باز کردن Status chart:

بر روی گزینه status chartدر پنجره manager میتوان جدول را باز کرد.

با راست کلیک بر روی status chart و انتخاب گزینه open میتوان جدول را باز نمود.

از منوی debug و انتخاب گزینه status chartمیتوان جدول را باز کرد.

۶.۹ Password :

در PLCهای KINCO به منظور ایجاد محدودیت در دسترسی به برنامه میتوان از password استفاده نمود . این رمز میتواند شامل حروف ویا رقم باشد .ماکزیمم تعداد این رمز میتواند ۸ بیت باشد .

به طور کلی در plc های kinco ۳ سطح دسترسی وجود دارد :

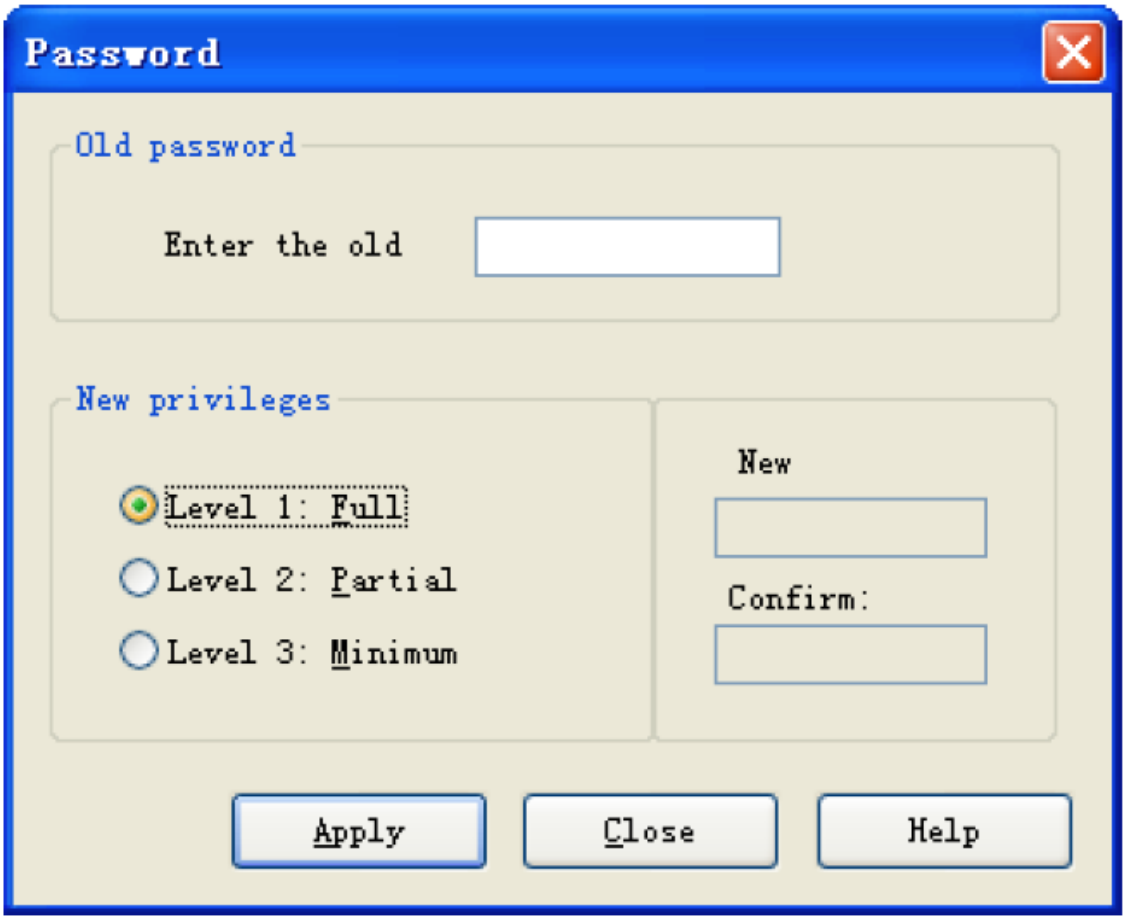
سطح ۱: دسترسی کامل میباشد . در این سطح هیچ رمزی وجود ندارد .

سطح ۲:در این سطح فقط در زمان دانلود برنامه بر روی CPUنیاز به وارد کردن رمز میباشد

سطح ۳: در این سطح کاربر هم برای دانلود و هم برای آپلود کردن برنامه نیاز به رمز دارد

چگونگی تغییر رمز :

از منوی PLCمیتوان پنجره passwordرا باز نمود .



Old password:چنانچه CPU قبلا دارای رمز باشد در این قسمت باید رمز قبلی وارد شود .

New Privileges:

در این قسمت میتوان سطح دسترسی مورد نظر را انتخاب نمود.

New password: در این قسمت میتوان رمز جدید را وارد نمود.

Confirm: در این قسمت مجددا رمز جدید را وارد نمایید .

سپس گزینه Applyرا انتخاب کنید . تنظیمات و رمز جدید بر روی CPU متصل به کامپیوتر تنظیم میشود .

چنانچه رمز انتخابی فراموش شود ، به منظور استفاده مجدد از CPUباید آن را clearکنید .

برای این کار در قسمت منوی PLCگزینه Clearرا انتخاب نمایید . با این کار رمز پاک شده . باید توجه داشت که با انتخا ب این گزینه برنامه و کل تنظیمات ماژول CPU پاک خواهد شد و CPUبه تنظیمات اولیه کارخانه باز میگردد.

فصل هفتم :آشنایی با نرم افزار و برنامه نویسی PLC KINCO K3 :

خلاصه :

در این بخش تمامی دستورات به همراه توضیحات کامل و کاربردی بیان شده است و دستورات به دو زبان برنامه نویسی IL و LD (زبان های برنامه نویسی قابل استفاده در PLC های KINCO) توضیح داده میشود.

تمامی دستورات KINCO – K3 بر مبنای استاندارد IEC-61131-3 در برنامه نویسی میباشد.

برنامه نویسی PLC های KINCO به دو زبان LD(Ladder) و IL(instruction list) انجام میشود:

برنامه نویسی IL : یک زبان برنامه نویسی سطح پایین میباشد که بسیار مشابه زبان اسمبلی است .IL به زبان ماشین نزدیک میباشد و زبان بسیار موثر و کارآمدی است و برای کاربران با سابقه و حرفه ای مناسب میباشد .

برنامه نویسی IL زبان برنامه نویسی خطی میباشد .هر برنامه ILشامل زنجیره ای از دستورات است و هر دستور در یک خط جدید قرار میگیرد و شامل یک عملگر میباشد ، وجود عملوند اختیاری میباشدو با یک ویرگول یا فاصله از یکدیگر جدا میشوند.میتوان در ادامه هر خط برنامه یک نوشته به عنوان کامنت وارد کرد که با پرانتز و علامت صورت مشخص میشود.

در پایین فرمت کلی برنامه IL نشان داده شده است :

```
label:
Operator    Operands          (* Comment *)
```

:Lable

این قسمت به صورت اختیاری میباشد و زمانی استفاده میشود که از دستور jump استفاده نماییم .دستور jump زمانی استفاده میشود که بخواهید به خط مشخصی از برنامه پرش کنید.در این صورت lableمشخص کننده خط مورد نظر میباشد که میخواهیم به آن پرش کنیم و در جلوی خط قرار میگیرد.

Operator :عملگر

Operand:عملوند های مختلف در بخش دستورات توضیح داده خواهد شد .

:Comment

این قسمت به صورت اختیاری میباشد .تنها یک کامنت میتواند در هر خط قرار گیرد.

CR: در IL زبان یک آکومولاتور(انباره) عمومی به نام CR (Curren Result) را فراهم میکند که نتیجه جاری عملیات منطقی در آن قرار میگیرد. در زبان IL زمانی دستوری اجرا میشود که مقدار CR یک باشد.

تمامی دستورات در نرم افزار Kinco Builder بر اساس تاثیر آنها بر روی CR گروه بندی میشوند.جدول زیر این گروه بندی را نمایش میدهد. این گروه بندی در استاندارد IEC61131-3 مشخص نشده است و این تقسیم بندی در سیستم های برنامه نویسی دیگر میتواند متفاوت باشد.

مثال	تاثیر برروی CR	گروه
LD , LDN	باعث ایجاد CRمیگردد	C
دستورات Bit logic، دستورات مقایسه ای	مقدار CRرا برابر با حاصل عملیات قرار میدهد	P
ST , R, S ,JMP	تاثیری بر روی CR ندارد	U

NETWORK (شبکه) :

در نرم افزار Kinco Builder یک برنامه از بخش های زیر تشکیل میشود:

الگوی کلی و نام برنامه

بخش معرفی متغیرها

بخش کدها (Code part) که شامل دستورات میشود.

شبکه به عنوان اساسی ترین بخش در قسمت برنامه به شمار میرود و هر Code part شامل تعدادی شبکه میباشد.

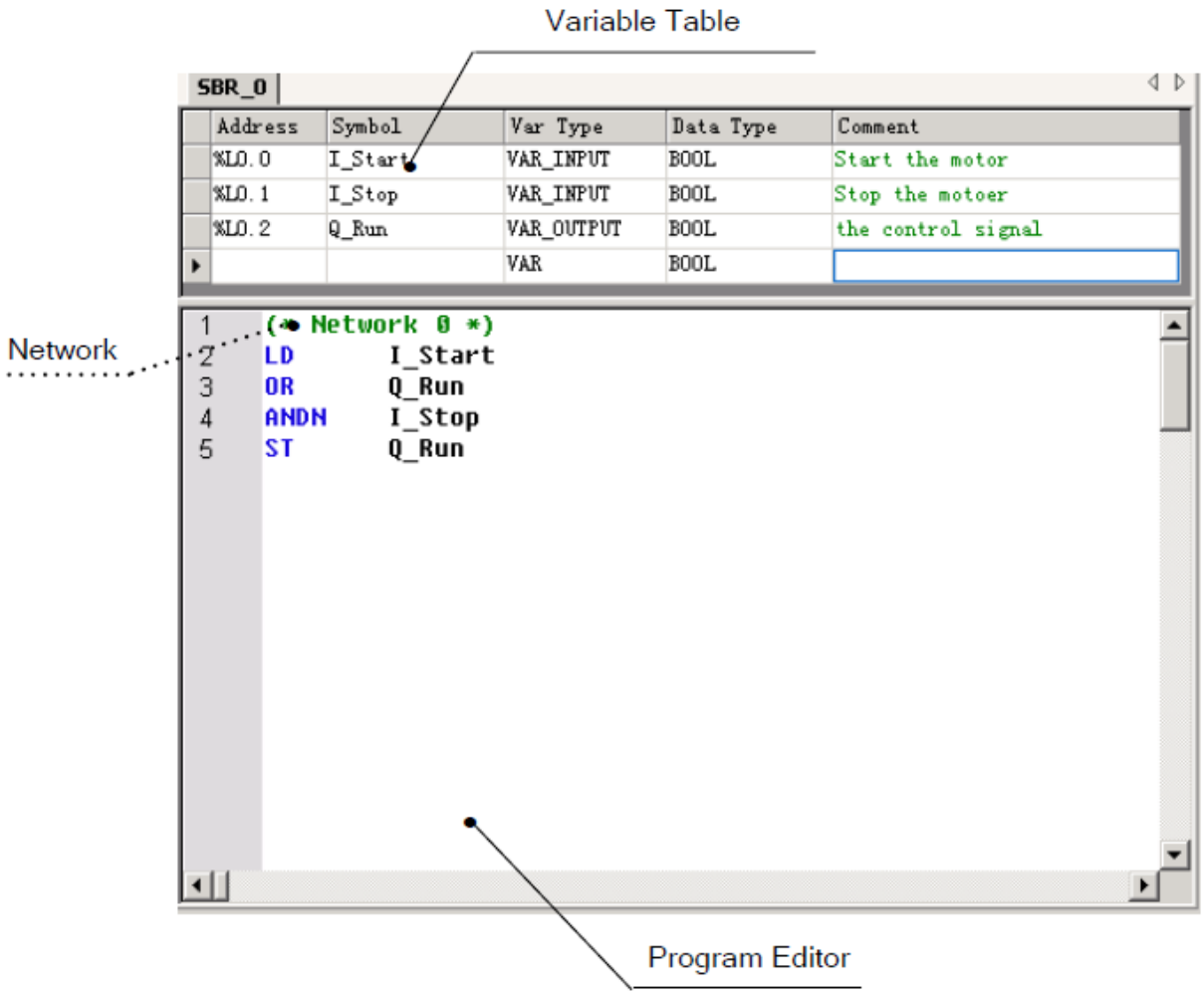
به طور کلی هر شبکه شامل بخش های زیر میباشد:

لیبل

Comment (توضیحات)

دستورات

صفحه EDITOR در زبان IL به صورت زیر میباشد:



IL Editor از دو بخش تشکیل میشود :

جدول متغیرها (Variable Table) : در این جدول میتوان متغیرها با سمبل های مختلف را تعریف کرد.

Program Editor: فضایی است که در آن میتوان برنامه نوشت و برنامه را تغییر داد .

نحوه ایجاد شبکه دربرنامه به زبان IL:

برای وارد کردن شبکه در برنامه میتوان به یکی از دو روش زیر عمل کرد :

میتوان با زدن کلیدهای Ctrl+Qیک شبکه به برنامه اضافه کرد .

میتوان روی صفحه program Editorراست کلیک کرد و گزینه Insert Networkرا انتخاب نمود .

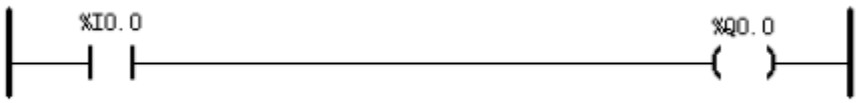
فرمت مجاز دستورات در شبکه (در زبان IL):

یک شبکه میتواند شامل تعدادی از دستورات باشد . همان طور که در بالا توضیح داده شد دستورات میتوانند به سه گروه C,P ,U تقسیم شوند . شروع یک شبکه باید با دستوری از گروه Cو پایان آن باید با دستوری از گروه P,U باشد . به مثال زیر توجه نمایید :

```
(* NETWORK 0 *)
LDN     %M0.0
TON     T0, 1000        (*Start T0 with the output of T1, timing: 1000*1ms *
ST      %M0.1
LD      %M0.1
TON     T1, 1000        (*Start T1 with the output of T0, timing: 1000*1ms *
ST      %M0.0

LD      %M0.1
ST      %Q0.0           (* Output square wave with 2s period at %Q0.0 *)
```

برنامه نویسی LD: برنامه نویسی LD(Ladder diagram) زبان برنامه نویسی گرافیکی متداول در برنامه نویسی PLC ها میباشد.این برنامه نویسی بر مبنای منطق نردبانی میباشد.در این نوع برنامه نویسی میتوان از سمبل های استاندارد در برنامه نویسی Ladder استفاده کرد و نسبت به برنامه نویسی ILبسیار ساده تر میباشد.

در تصویر پایین یک بخش کوچک از یک برنامه به صورت LD نشان داده شده است :



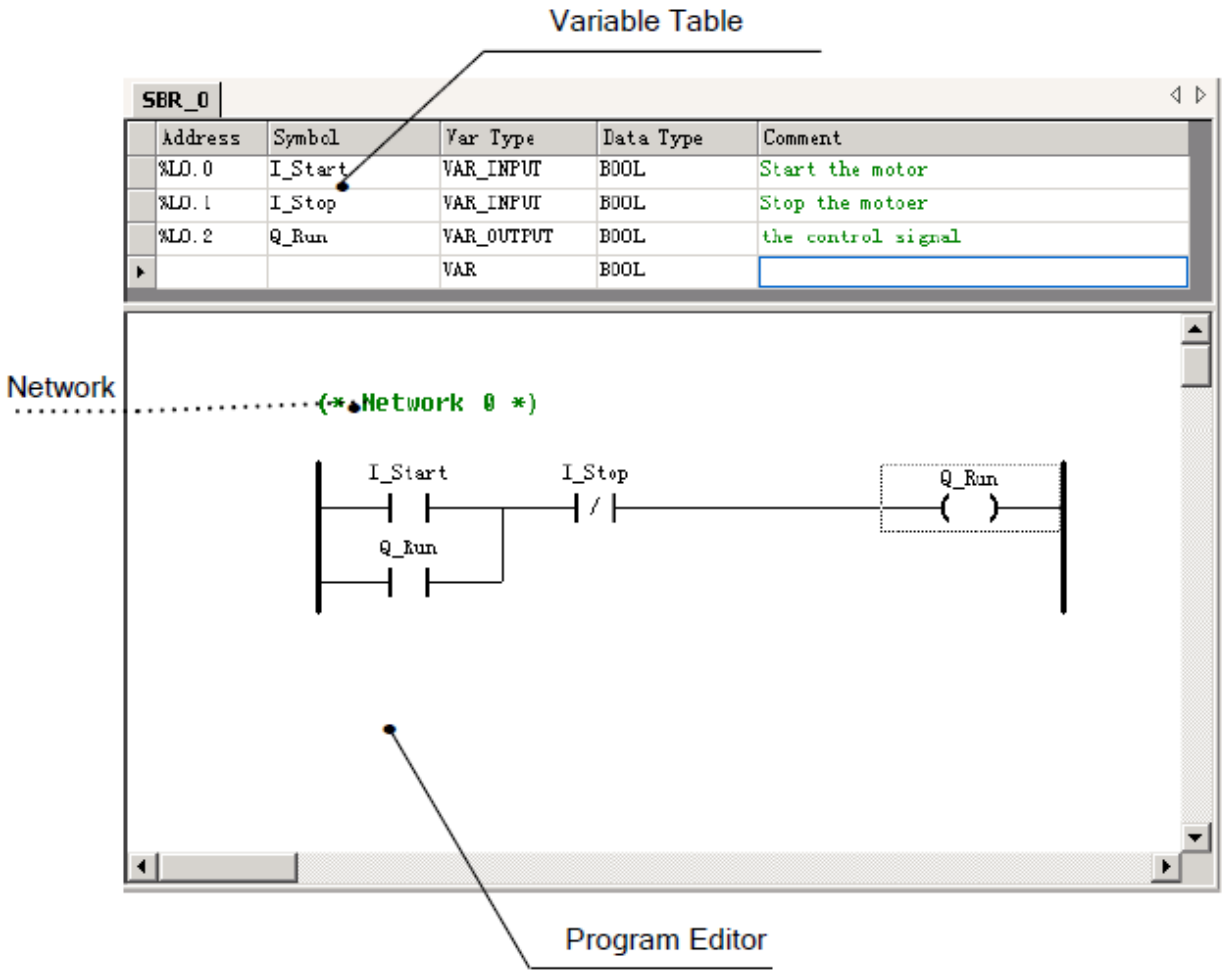
نحوه برنامه نویسی در زبان LD نیز مانند زبان IL به صورت شبکه ای میباشد . برای نوشتن برنامه به زبان LDو ایجاد یک شبکه منطقی کاربر باید از نشانه های گرافیکی استاندارد وهممچنین چیدمان مناسب برای این نشانه ها استفاده نماید.

Function: در زبان LD توابع به صورت بلوک های مستطیلی شکل وجود دارند که دارای تعدادی ورودی و خروجی های مخصوص به خود بوده و عملکرد خاصی دارند.

در اکثر بلوک ها یک بیت به نام EN در ورودی ویک بیت در خروجی به نام ENO وجود دارد .بیت EN برای کنترل عملکرد تابع استفاده میشود .چنانچه EN در حالت True باشد (وضعیت ۱ داشته باشد) تابع فعال شده (در حالت اجرا قرار میگیرد) و بیت خروجی ENO نیز در حالت True (وضعیت ۱) قرار میگیرد و چنانچه ENورودی در وضعیت False قرار گیرد تابع غیر فعال میباشد و ENO نیز FALSE میشود .

زمانی که یک پروژه جدید تعریف میشود LD Editor آماده برای برنامه نویسی میباشد و همچنین زمانی که یک برنامه LDنوشته شده را باز میکنید LD EDITOR آماده میباشد .

صفحه EDITOR در زبان LD به صورت زیر میباشد:



محدودیت های برنامه به زبان LD:

ماکزیمم تعداد شبکه ای در این زبان ۲۰۰ شبکه میباشد .شما میتوانید پنجره PROGRAM EDITOR را به چندخانه تقسیم کرد .درون این تقسیم یک شبکه میتواند به اندازه ۳۲ خانه افقی و ماکزیمم ۱۶ خانه عمودی توسعه یابد .بنابراین ماکزیمم المان افقی در یک شبکه به صورت زیر میباشد : چنانچه در آن فقط اتصالات (CONTACT) و COIL باشد تا ۳۱اتصال و ۱ COIL.چنانچه فقط شامل بلوک های دستورات باشد تا ۱۲ بلوک ،۱ اتصال و یک خروجی (COIL) و همچنین در یک شبکه تعداد شاخه ها در اتصالات موازی نباید از ۱۶ تا تجاوز کند .اتصال موازی ۲ بلوک دستوری مستقل یا بیشتر ممنوع میباشد .

عملکردهای معمولی :

با کلیک کردن بر روی هر یک از المان ها یک صفحه مستطیلی اطراف بلوک ایجاد میگردد .با دو بار کلیک کردن بر روی المان مورد نظر پنجره مربوط به تنظیمات آن باز میشود .در این فضا میتوانید پارامترهای مربوط به آن بلوک را به دلخواه تنظیم نمایید ،همچنین با راست کلیک کردن بر روی بلوک مورد نظر یک منو باز میشود که میتوانید بنا به نیاز از هر کدام از گزینه های آن استفاده نمایید .

همچنین میتوانید از کلیدهای UP,DOWN ,LEFT,RIGHTبرای تغییر FOCUS و ازکلید ENTER برای انتخاب فضای پارامترهای المان و واردکردن پارامترهای آن و ازکلید DELبرای حذف کردن بلوکی که برروی آن کلیک شده است (انتخاب شده) استفاده کرد .

مراحل برنامه نویسی در زبان LD:

نحوه ایجاد شبکه در زبان LD:

به یکی از چهار روش زیر میتوان یک شبکه در برنامه اضافه نمود:

1) در قسمت منو برروی LD کلیک کرده و گزینه Networkرا انتخاب کنید .

۲) در قسمت toolbar برروی نشانه  **کلیک نمایید**

۳) میتوان از کلیدهای Ctrl+Wاستفاده نمود.

۴) میتوان بر روی بلوک انتخابی راست کلیک کرد و گزینه NETWORKرا انتخاب نمود .در این صورت یک شبکه بعد از شبکه بلوک انتخابی ایجاد میشود.

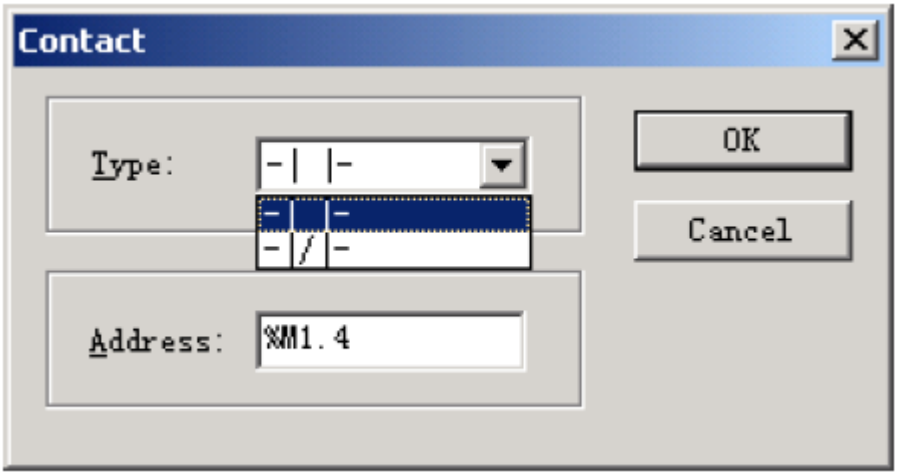
با روش های بالا میتوان یک شبکه در برنامه وارد کرد :

با دوبار کلیک کردن برروی Network lable پنجره ای برای نوشتن توضیحات(comment) دلخواه باز میشود .میتوان در این پنجره توضیحات مربوط به شبکه را وارد نمود .

پس از اضافه نمودن هر دستوری متغیرهای آن درابتدا به صورت علامت سوال قرمز (????) مشخص میشوند که این نشاندهنده آن است که متغیرهای مربوط به دستور (بلوک) تعریف نشده است .

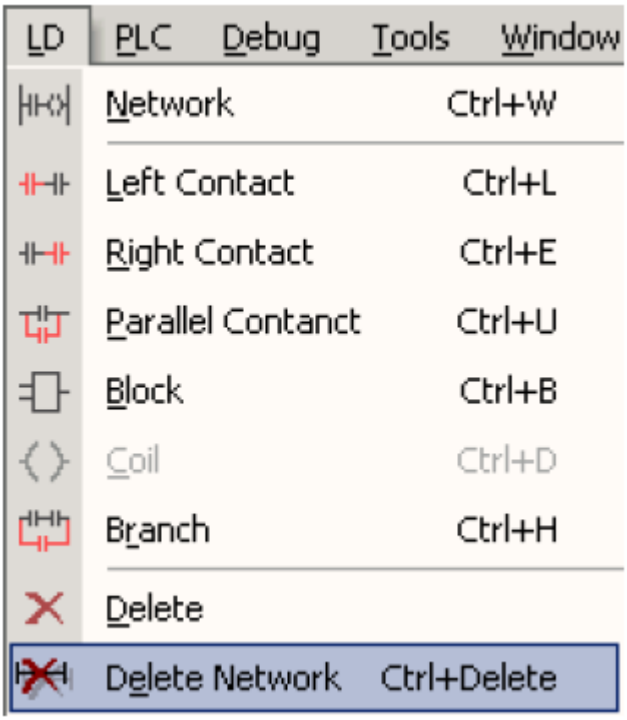
ابتدا باید قبل از کامپایل کردن برنامه این متغیرها را تعریف نمود .با کلیک کردن بر روی متغیر یک پنجره بازمیشود که مربوط به فضای متغیرمیباشد .میتوانید در این فضا متغیر دلخوا ه و یا عدد ثابت مورد نظر را وارد نمایید .همچنین میتوانید از کلید enter برروی المانی که روی آن focus کرده اید در تعیین فضای متغیرها استفاده نمایید .چنانچه آدرس مستقیم را درقسمت متغیروارد کردید نیازی به وارد کردن علامت ٪ قبل از آن نیست .

به علاوه با دوبار کلیک کردن بر روی یک متغیر، پنجره مربوط به آن باز شده و میتوانید نوع آن و پارامترها را تغییر و اصلاح نمایید .تصویر زیرپنجره مربوط به یک اتصال میباشد



با کلیک برروی یک بلوک و یا المان انتخابی آن را به عنوان مرجع درنظرگرفته و حالا میتوانید با روش های زیر المان های مختلفی را نسبت به المان انتخابی اضافه نمایید.

میتوان با راست کلیک کردن برروی هر المان از تمام گزینه های موجود در لیست (مانند تصویر زیر)استفاده کرد:



Left contact: این گزینه یک اتصال در سمت چپ المان انتخابی ایجاد مینماید .

Right contact: یک contactدرقسمت راست المان انتخابی ایجاد میشود .

Parallel contact: یک contactبه صورت موازی به المان انتخابی اضافه میشود .

Block: یک بلوک (function/FB/subroutine) به صورت سریال با المان انتخابی به شبکه اضافه میگردد .

Coil: چنانچه خروجی به عنوان مرجع انتخاب شود این گزینه فعال است و یک خروجی به صورت موازی با خروجی انتخابی به شبکه اضافه میکند .

Branch:با کمک این گزینه میتوان یک شاخه به صورت موازی با المان های دیگر ترسیم کرد .

Delete: المانی که به عنوان مرجع درنظر گرفته شده است را حذف مینماید .

Delete Network:شبکه مربوط به المان مرجع را به طورکامل حذف میکند

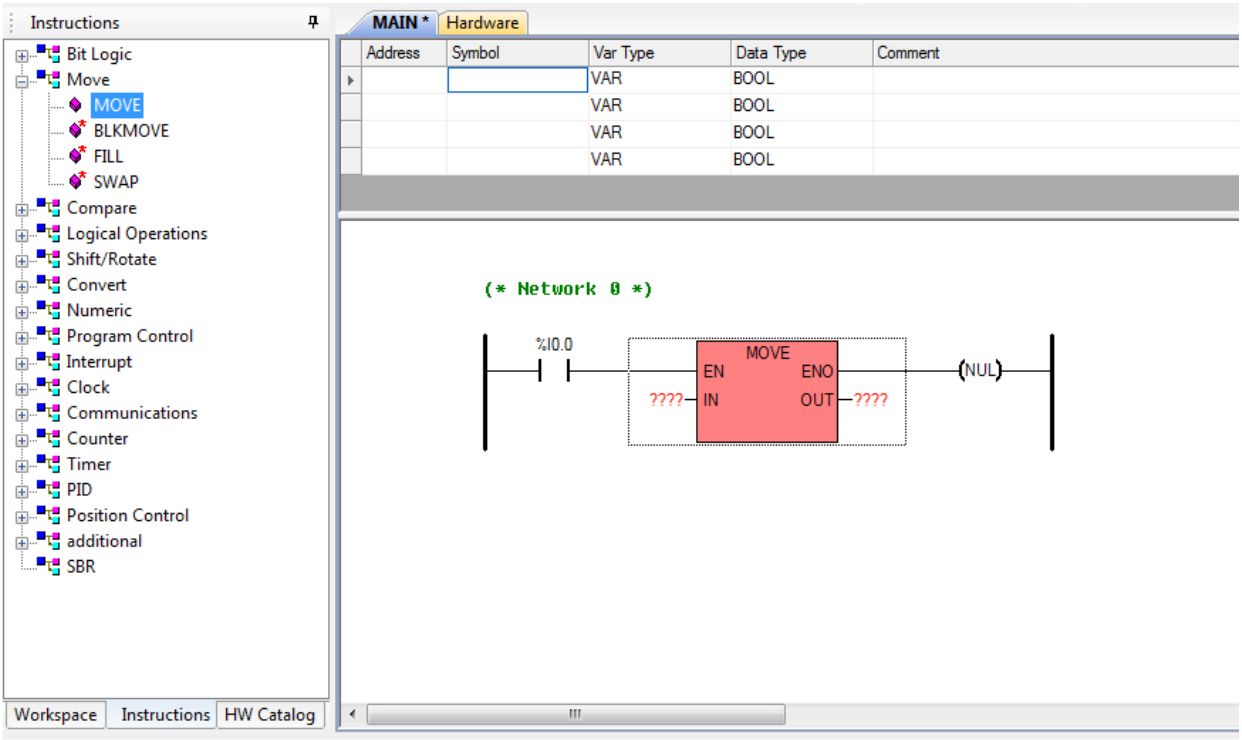
برای تمامی کارهای توضیح داده شده در بالا میتوان از نشانه های موجود در Toolbarاستفاده نمود :

به تصویر نشان داده شده در پایین توجه نمایید :

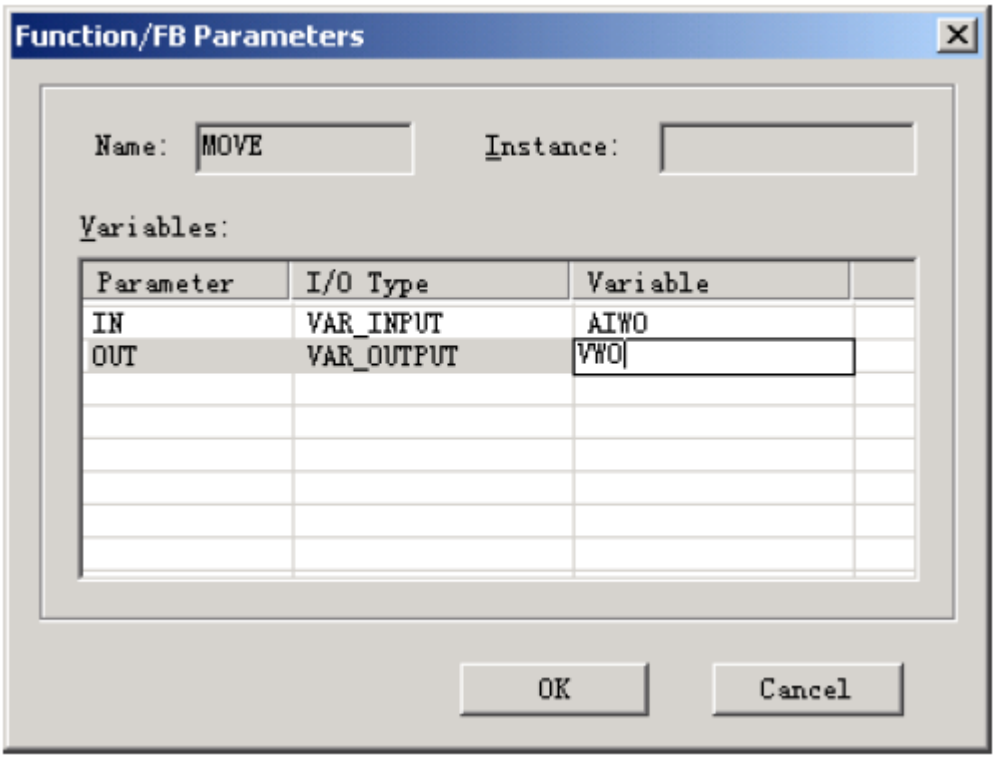
وارد کردن دستورات در برنامه :

با انتخاب نوار instructions (مطابق تصویر نشان داده شده در پایین) میتوان به دستورات دسترسی پیدا کرد .برای وارد نمودن دستورات دلخوا ه روی آن در نوار instructions دو بار کلیک نمایید .

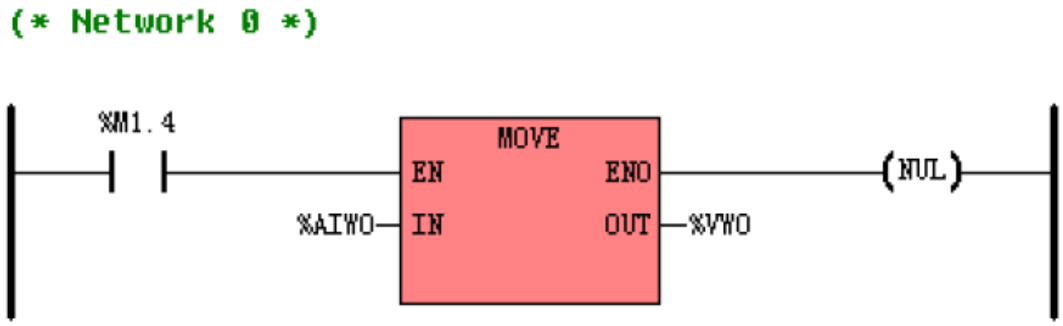
به عنوان مثال د رتصویر زیر بلوک Moveبه شبکه اضافه شده است :



برای تعیین متغییرها در هر بلوک میتوان با کلیک بر روی ورودی و یا خروجی متغیر مورد نظر را وارد کرد و یا میتوان با استفاده از ماوس ویا کلید Enter به متغیرها ی المان جدید (به منظور ایجاد تغییر و اصلاحات لازم) دسترسی پیدا کرد .به علاوه با دوبار کلیک بر روی المان در برنامه میتوان به پنجره پارامترها دسترسی پیدا کرد و اصلاحات لازم را اعمال نمود .

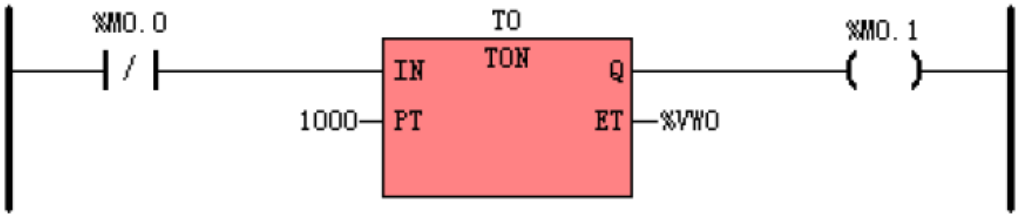


میتوان از طریق این جدول ورودی ها و خروجی های مورد نظر را برای هر دستور وارد نمود .

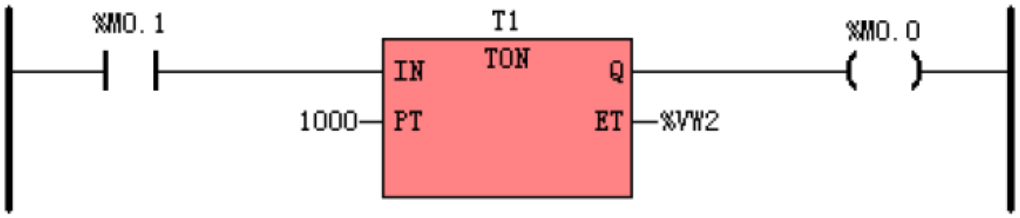


پس از آنکه یک شبکه به صورت کامل انجام شد میتوان مراحل اضافه کردن و اصلاح شبکه جدید را انجام داد تا برنامه(POU) به صورت کامل تمام شود.به عنوان تصویر زیر نشان دهنده یک برنامه در زبان LD میباشد :

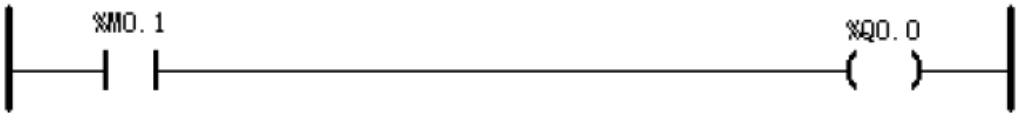
```
(* Network 0 *)
(* Start T0 with the output of T1, timing: 1000*1ms *)
```




```
(* Network 1 *)
(* Start T1 with the output of T0, timing: 1000*1ms *)
```



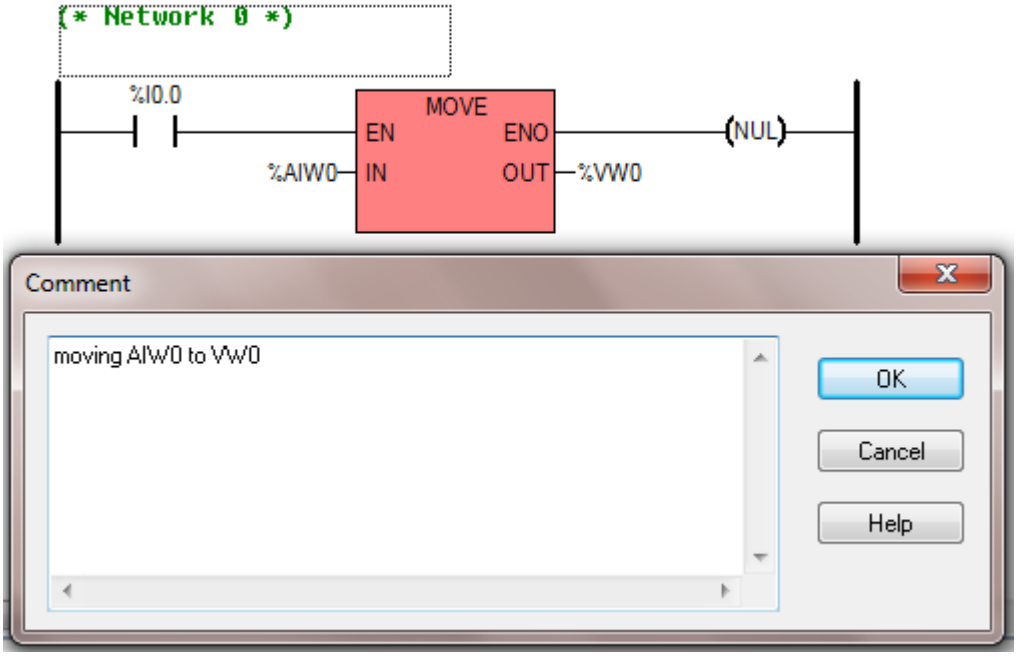

```
(* Network 2 *)
(* Output square wave with 2s period at %Q0.0 *)
```



زمانی که میخواهیم یک شبکه جدید ایجاد نماییم ،چنانچه lable شبکه فعلی که انتخاب شده به عنوان مرجع در نظر گرفته شود ، شبکه ایجاد شده در بالای شبکه مرجع ایجاد میشود و در غیر این صورت شبکه جدید در زیر شبکه مرجع ایجاد میگردد.

در اینجا مقصود از شبکه فعلی شبکه ای است که المان انتخاب شده در آن قرار دارد .

نکته : همان طور که در بالا اشاره شد برای وارد نمودن توضیحات در ابتدای هر شبکه میتوان بر نام شبکه دو بار کلیک کرده ،صفحه ای مانند تصویر زیر باز میشود ، میتوان توضیحات مربوط به آن شبکه را وارد کرد .



در این مرحله میتوان با باز کردن Debug از قسمت منو وانتخاب گزینه Monitor وارد مد مانیتورینگ شد .دراین مرحله وضعیت متغیرهای برنامه در صفحه LD Editorنمایش داده میشود و نمیتوان تغییری در این مرحله در برنامه ایجاد کرد .

فصل هشتم :آشنایی با دستورات Kinco-K3:

مقدمه

در این بخش عملکرد تمامی دستورات به صورت کامل بیان خواهد شد . نحوه استفاده از این دستورات هم به زبان IL و هم به زبان LD توضیح داده میشود.

قبل از توضیحات دستورات لازم میباشد با مفاهیمی که در ادامه به آن اشاره میشود آشنا شوید.

۸.۱ EN، ENO و CR

در زبان LD، عملوند هایی به نام EN و ENO وجوددارد که این عملوندها برای تمامی دستورات دارای عملکردی مشابه میباشند . EN(Enable) به صورت یک ورودی بیتی (Bool) برای فعال سازی بیشتر دستورات میباشد .زمانی که مقدار آن ۱ شود ، دستور مورد نظر اجرا خواهد شد . ENO(Enable out) به صورت یک خروجی بیتی (Bool) در بیشتر بلاک ها میباشد .هنگامی که EN یک شده و دستور مورد نظر اجرا گردد ، مقدار ENO یک خواهد شد .

در زبان IL، عملوندی به نام CR(Current result) وجود دارد که پس ازاجرای هر دستور refreshمیشود و میتواند به عنوان شرط اجرا و یا یک عملوند در دستور بعدی مورد استفاده قرار گیرد.

۸.۲-دستورات منطقی (Bit Logic Instructions):

۸.۲.۱ اتصالات استاندارد(standard contact) :

	Name	Usage	Group	
LD	Normally open contact	*bit* ⊣ ⊢		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	Normally closed contact	*bit* ⊣/⊢		
IL	LD	LD *bit*	C	
	LDN	LDN *bit*		
	AND	AND *bit*	P	
	OR	OR *bit*		
	ANDN	ANDN *bit*		
	ORN	ORN *bit*		

Operand	Input/Output	Data Type	Acceptable Memory Areas
bit	Input	BOOL	I, Q, V, M, SM, L, T, C, RS, SR, constant

LD : از این اتصالات جهت کنترل فرامین وانتقال اطلاعات استفاده میگردد. این اتصالات به صورت بیتی (Boolean) میباشد.

NO(Normally Open) :. زمانی که مقدار یک بیت ۱ میشود بدان معناست که کلید Normally Openبسته(On) شده است، بنابراین سیگنال توسط خط رابط افقی به المان بعدی انتقال میابد . به عبارت دیگر زمانی که ولتاژ به این تیغه برسد (مقدار بیت مورد نظر یک منطقی باشد) این اتصال فعال بوده و سیگنال را هدایت میکند.و زمانی که ولتاژی وجود نداشته باشد (مقدار بیت موردنظر ۰ منطقی باشد) غیر فعال بوده و اتصال قطع میباشد

NC(Normally Closed) : زمانی که مقدار بیت ۰ میشود بدان معناست که کلید Normally Closedبسته (on) میباشد، بنابراین سیگنال توسط خط رابط افقی به المان بعدی انتقال میابد .به عبارت دیگرزمانی که ولتاژی وجود نداشته باشد(مقداربیت مورد نظر ۰منطقی باشد) این اتصال فعال بوده و سیگنال را هدایت میکند و زمانی که ولتاژ به این تیغه برسد(مقدار بیت مورد نظر یک منطقی باشد) این تیغه غیر فعال بوده و اتصال قطع میباشد.

در واقع NC برعکس NOعمل میکند.

IL:

اتصالات normally open در برنامه نویسی IL توسط دستورات LD ,AND,OR نمایش داده میشود.

دستور LD بیت مورد نظررا خوانده و مقدار CRرا مساوی با آن مقدار قرار میدهد .

دستور AND جهت ANDکردن بیت مورد نظربا CR به کار میرود و مقدار CR را برابر با نتیجه عملیات قرار میدهد.

دستور ORبیت مورد نظر را با CR، ORکرده و مقدار CR را مساوی نتیجه حاصل از این عملیات قرار میدهد.

اتصالات Normally cosed در برنامه نویسی IL توسط دستورات LDN ,ANDN,ORN بیان میشود.

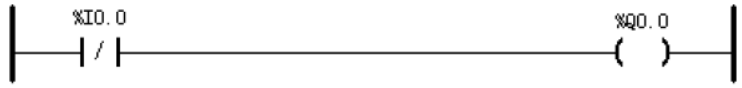
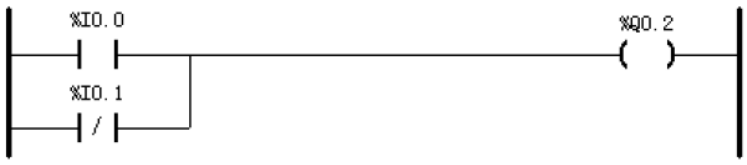
دستور LDN مقدار NOT بیت مورد نظر را در CRذخیره مینماید .

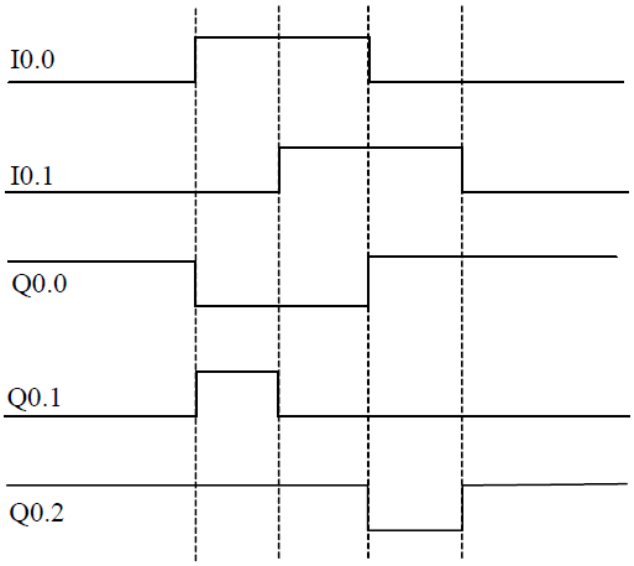
دستور ANDN، جهت ANDکردن NOTمنطقی بیت مورد نظر با CRمیباشد و درنهایت مقدار CR را برابر با نتیجه عملیات قرار میدهد.

دستور ORN ، جهت ORکردن NOTمنطقی بیت مورد نظر با CRمیباشد و در نهایت مقدار CRرا برابر با نتیجه عملیات قرار میدهد.

همان طو که درجدول بالا نشان داده شده است این دستورات در تمام CPUها وجود دارد.

جهت درک بهتر مفاهیم اتصالات به مثال زیر توجه فرمایید :

LD	IL
	LDN %I0.0 ST %Q0.0
	LD %I0.0 ANDN %I0.1 ST %Q0.1
	LD %I0.0 ORN %I0.1 ST %Q0.2



۸.۲.۲-اتصالات لحظه ای (IMMEDIATE CONTACT) :

	Name	Usage	Group	
LD	Normally open immediate contact	*bit*		CPU304 CPU304EX CPU306 CPU306EX CPU308
	Normally closed immediate contact	*bit*		
IL	LDI	LDI *bit*	C	
	LDNI	LDNI *bit*		
	ANDI	ANDI *bit*	P	
	ORI	ORI *bit*		
	ANDNI	ANDNI *bit*		
	ORNI	ORNI *bit*		

Operand	Input/Output	Data Type	Acceptable Memory Areas
bit	Input	BOOL	I (CPU body)

زمانی که دستورات لحظه ای اجرا میشوند ، مقدار فیزیکی کانال های ورودی در هر لحظه به صورت فوری بدست می آید، اما فضای حافظه مربوط به تصویر ورودی ها به روز نمیشود.در حقیقت این اتصالات برخلاف اتصالات معمولی برای به روز شدن نیاز به سیکل اسکن ندارند و در اصل این اتصالات هر لحظه مطابق با تغییرات جدید به سرعت تغییر

میابند. به عبارت دیگر در این گونه اتصالات با پایه فیزیکی ورودی سرو کار خواهید داشت(در صورتی که در اتصالات استاندارد CPU از فضای تصویر ورودی که در هر سیکل اسکن به روز میشود استفاده میکند) .دستورات لحظه ای فقط در مورد کانال های دیجیتال ورودی مورد استفاده قرار میگیرند.چنانچه از این نوع اتصالات برای ورودی دیجیتال استفاده شود تنظیمات فیلتر که در قسمت hardware در نرم افزار برای ورودی های معمول انجام میشد ، بی تاثیر خواهد بود (برای درک بهتر توضیحات بخش فیلتر در قسمت hardwareدر نرم افزار مطالعه شود)

LD : normally open: هنگامی که مقدار منطقی ورودی فیزیکی به ۱تغییر کند، اپن اتصال سریعا بسته شده و سیگنال را هدایت میکند.

Normally closed: هنگامی که مقدار منطقی ورودی فیزیکی به صفر تغییر کند، این اتصال سریعا بسته شده و سیگنال را هدایت میکند

IL :

در این زبان اتصال لحظه ای Normally open با دستورات LDI , ANDI,ORI ارائه میشود.

LDI سریعا مقدار فیزیکی ورودی را خوانده و مقدار CR را برابر با آن قرار میدهد.

ANDI دستوری است که مقدار فیزیکی ورودی را با مقدار CR، AND کرده و در نهایت مقدار CR را برابر با حاصل این عملیات قرار میدهد.

ORI: دستوری است که مقدار فیزیکی ورودی را با مقدار CR، OR کرده و درنهایت مقدار CR را برابر با حاصل این عملیات قرار میدهد.

اتصالات لحظه ای Normaly closed با دستورات LDNI ,ANDNI ,ORNI بیان میشود.عملکرد مشابهی با توضیحات ارائه شده در بالا دارد با این تفاوت که NOT منطقی عملیات فوق صورت میپذیرد.

همان طور که در جدول بالا مشاهده میکنید این دستورات تنها در مدل های K306EX و K308وجود دارد.

۸.۲.۳ خروجی (Coil) :

	Name	Usage	Group	
LD	Coil	bit —( )—		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	Negated Coil	bit —(/)—		
	Set Coil	bit —(S)—		
	Reset Coil	bit —(R)—		
	Null coil	—(NUL)—		
IL	ST	ST bit	U	
	STN	STN bit		
	R	R bit		
	S	S bit		

Operand	Input/Output	Data Type	Acceptable Memory Areas
bit	Output	BOOL	Q, V, M, SM, L

Coil در اصل یک خروجی میباشد که در پایان هر خط از برنامه یا یک شبکه قرار میگیرد و در اصل بیان گر نتیجه اجرای دستورات قبل از خود به صورت بیتی میباشد.

: LD

دستور Coil:

این دستور در اصل سیگنال ارسالی از سمت چپ را دررجیستر تصویر خروجی مینویسد . چنانچه دستورات قبل از خروجی اجرا شده و و یا شرط مورد نظر در برنامه برقرار باشد مقداراین بیت ۱ میباشدو درغیر این صورت مقدار آن ۰ منطقی میباشد .این خروجی به صورت بیتی تعریف میشود.مقدار این خروجی مطابق با مقدار فضای تصویر خروجی خواهد بود .

: Negated coil

این دستور در اصل معکوس سیگنال ارسالی از سمت چپ را در رجیستر تصویر خروجی مینویسد.در صورتی که این سیگنال (نتیجه عملیات قبل) مقدار ۱ منطقی باشد ،مقدار این بیت صفر و چنانچه مقدار سیگنال ارسالی از سمت چپ صفر باشد مقدار این بیت ۱ خواهد بود.

: Set Coil

در این دستور چنانچه مقدار سیگنال ارسالی از سمت چپ ۰ منطقی باشد مقدار فضای تصویر خروجی بدون تغییر باقی میماند (چنانچه مقدار اولیه ۰ را داشته باشد ، ۰ باقی میماند) و درصورتی که این سیگنال مقدار منطقی ۱ شود ، در فضای تصویر خروجی مقدار ۱ نوشته میشود و مقدار بیت برابر با ۱ خواهد بود . پس از قطع مجدد سیگنال ارسالی از سمت چپ همان طور که در بالا توضیح داده شد این بیت بدون تغییر باقی خواهد ماند و همچنان مقدار ۱ را نگه میدارد.

Reset Coil:

در این دستور چنانچه مقدار سیگنال ارسالی از سمت چپ ۰ باشد این دستور اجرا نشده و مقدار فضای تصویر خروجی بدون تغییر باقی میماند ، در صورتی که سیگنال ارسالی از سمت چپ مقدار ۱ شود ،رجیستر تصویر خروجی ۰ شده و درنهایت این بیت ۰ میشود.

Null Coil : در حقیقت عملکرد خاصی ندارد و تنها بیان گر پایان یک شبکه میباشد .در شبکه هایی که خروجی عملیات مداخله ای در برنامه ندارد و نیاز به استفاده از آن در قسمت های دیگر برنامه ندارید از این خروجی استفاده نمایید.

IL:

خروجی ها در زبان IL به صورت دستورات ST,STN,R,S ارائه میشوند.

دستورات ST مقدار CR را در رجیستر تصویر خروجی مینویسد .

دستور STN مقدار معکوس CR را در رجیستر تصویر خروجی مینویسد.

دستور S: این دستور پس از اشدن مقدار CR ، رجیستر تصویر خروجی را برابر با ۱ میکند و در غیر این صورت این رجیستر را بدون تغییر حفظ مینماید .

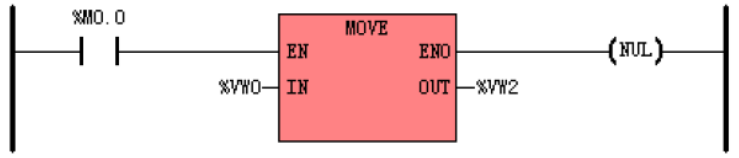
دستور R: این دستور پس از ۱ شدن مقدار CR ، رجیستر تصویر خروجی را برابر با ۰ میکند و در غیر این صورت این رجیستر را بدون تغییر حفظ مینماید.

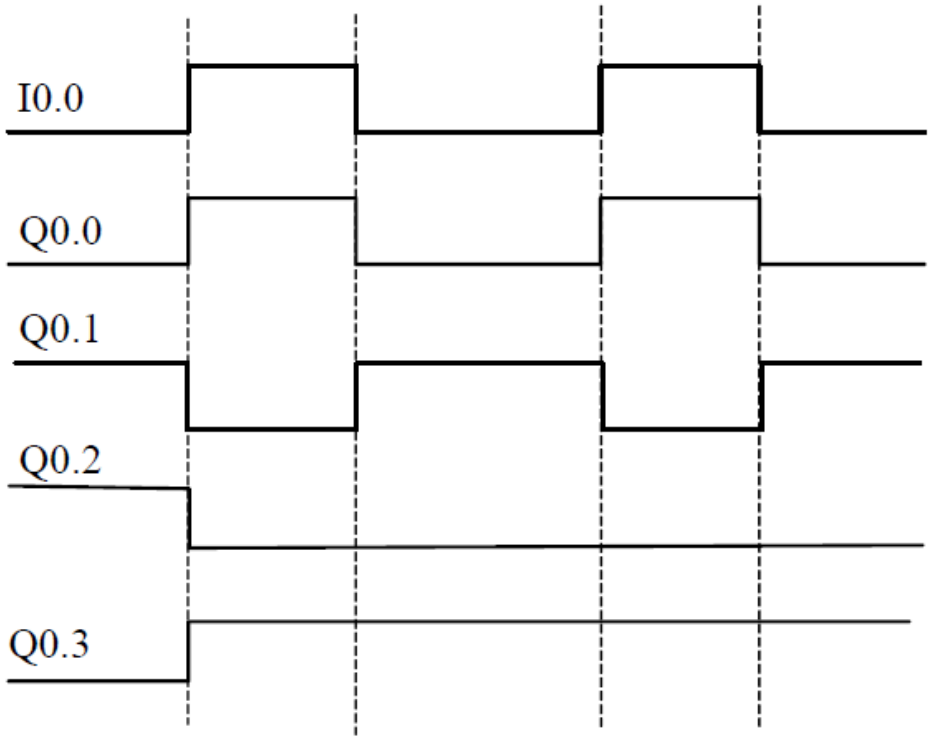
دستور های ST,STN,S,R بر روی مقدار CR تاثیری نخواهند داشت .

همان طور که در جدول بالا مشاهده میکنید این دستورات در تمام CPU ها وجود دارد.

در مثال زیر با فعال شدن I0.0(اشدن این بیت) مقدار Q0.0 ,Q0.1,Q0.3برابر با یک خواهد شد و مقدار Q0.2، برابر با صفرخواهد شد. پس از قطع سیگنال I0.0، مقدار Q0.0,Q0.1 صفرخواهد شد ولی Q0.3و Q0.4 بدون تغییر باقی خواهند ماند .به عبارت دیگر با یک بار یک شدن I0.0، Q0.2 برای همیشه غیر فعال و Q0.3 برای همیشه فعال خواهد ماند (مگر آن که در قسمت دیگری از برنامه مجدد Setو یا Resetشوند)

مثال دوم کاربرد Null در برنامه میباشد که همان طور که توضیح داده شد برای نشان دادن پایان یک شبکه به کار میرود.

LD	IL
	LD %I0.0 ST %Q0.0 STN %Q0.1 R %Q0.2 S %Q0.3
	LD %M0.0 MOVE %VW0, %VW2



۸.۲.۴خروجی لحظه ای :

	Name	Usage	Group	
LD	Immediate Coil	*bit* —(I)—		☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	Set Immediate Coil	*bit* —(SI)—		
	Reset Immediate Coil	*bit* —(RI)—		
IL	STI	STI *bit*	U	
	RI	RI *bit*		
	SI	SI *bit*		

Operand	Input/Output	Data Type	Acceptable Memory Areas
bit	Output	BOOL	Q (CPU body)

این دستورات فقط برای کانالهای خروجی دیجیتال روی CPU (DO) قابل استفاده میباشد.

هنگامی که دستورات خروجی لحظه ای اجرا میشود ، سیگنال ارسالی از سمت چپ سریعا هم در خروجی های فیزیکی و هم در رجیستر تصویر خروجی مربوط به آن نوشته میشود (در صورتی که در خروجی های استاندارد این مقدار فقط در رجیستر تصویر خروجی نوشته میشد و خروجی فیزیکی تا پایان سیکل اسکن برنامه تغییر نمیکرد)

تمامی دستورات در این قسمت مانند تمامی دستورات خروجی های استاندارد میباشد با این تفاوت که در این قسمت هم خروجی فیزیکی و هم رجیستر تصویر خروجی به سرعت تغییر خواهند کرد .

.LD:

Reset immidiate Coil: زمانی که این دستور اجرا میشود ، چنانچه نتیجه حاصل از عملیات سمت چپ ۱ باشد، هم خروجی فیزیکی وثبات خروجی (رجیستر تصویر خروجی) وابسته به آن سریعا ۰میشوند . در غیر این صورت خروجی بدون تغییر باقی میماند .

set immidiate Coil : زمانی که این دستور اجرا میشود ، چنانچه نتیجه حاصل از عملیات سمت چپ ۱ باشد ،هم خروجی فیزیکی وثبات خروجی (رجیستر تصویر خروجی) وابسته به آن سریعا امیشوند . در غیر این صورت خروجی بدون تغییر باقی میماند .

:IL

خروجی لحظه ای توسط دستورات SI,RI,STI نشان داده میشود .

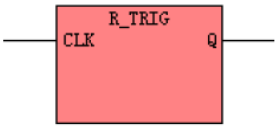
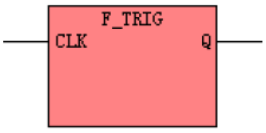
دستور STI: زمانی که این دستور اجرا میشود ، مقدار CR سریعا در خروجی فیزیکی و در ثبات خروجی ریخته میشود .

دستور RI:زمانی که این دستور اجرا میشود ، چنانچه ۱CR باشد ، هم خروجی فیزیکی و هم ثبات خروجی سریعا ۰ میشوند .در غیر این صورت این مقادیر بدون تغییر باقی میماند

دستور SI: زمانی که این دستور اجرا میشود ، چنانچه ۱CR باشد ، هم خروجی فیزیکی و هم ثبات خروجی سریعا ۱ میشوند .در غیر این صورت این مقادیر بدون تغییر باقی میماند.

این دستورات تاثیری در مقدار CRندارند ..

۸.۲.۵: آشکار ساز لبه :

	Name	Usage	Group	
LD	Rising edge detector			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	Falling edge detector			
IL	R_TRIG	R_TRIG	P	
	F_TRIG	F_TRIG		

Operands	Input/Output	Data Type	Acceptable Memory Areas
CLK (LD)	Input	BOOL	Power flow
Q (LD)	Output	BOOL	Power flow

دستورات آشکار ساز لبه R-Trig(آشکار ساز لبه بالارونده) و F-Trig (آشکار ساز لبه پایین رونده) میباشد .

دستور R-Trigحساس به لبه بالا رونده میباشد .هنگامی که ورودی از حالت صفر به یک تغییر وضعیت میدهد (از حالت قطع به وصل تبدیل میشود) خروجی دقیقا در لحظه تغییر این وضعیت برای یک لحظه (به اندازه سیک اسکن) یک خواهد شد و مجددا به حات صفر برمیگردد.

دستور F-Trigحساس به لبه پایین رونده میباشد .هنگامی که ورودی از حالت یک به صفر تغییر وضعیت میدهد (از حالت وصل به قطع تبدیل میشود) خروجی دقیقا در لحظه تغییروضعیت برای یک لحظه (به اندازه یک سیکل اسکن) یک خواهد شد و مجددا به حالت صفر برمیگردد.

خروجی این دستورات در هر بار ایجاد تغییر وضعیت در ورودی (قطع به وصل یا برعکس) یک لحظه یک میشود.

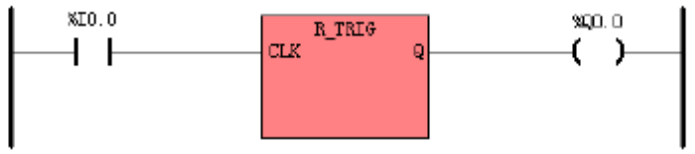
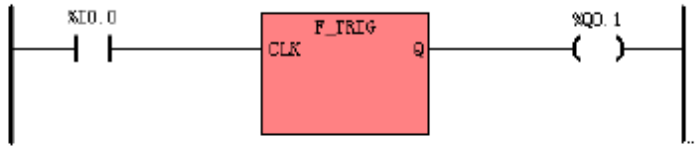
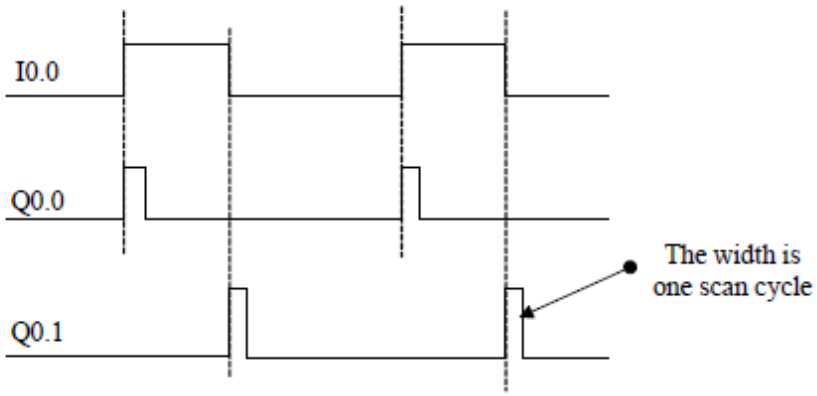
قابل ذکر است که برخی از دستورات حساس به لبه هستند و برای اجرا شدن نیاز به یک فعال سازی آنی دارند . از خروجی دستورات R-Trig و F-trig برای فعال سازی این دستورات استفاده میشود.

به طور کلی از این دستورات برای آشکار سازی لبه استفاده میشود.

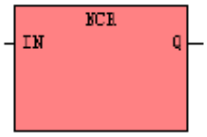
LD :در زبان LD این دستورات آشکار ساز لبه ورودی (CLK) میباشد .

IL:در زبان IL این دستورات آشکار ساز لبه CR میباشد.

همان طور که در جدول بالا مشاهده میکنید این دستورات در تمام CPU های KINCO وجود دارد.

LD	IL
	LD %I0.0 R_TRIG ST %Q0.0
	LD %I0.0 F_TRIG ST %Q0.1
Time Sequence Diagram	
	

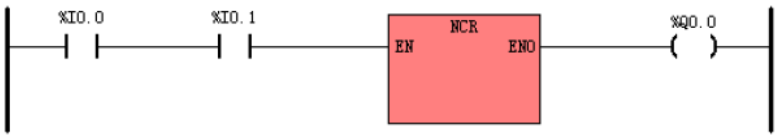
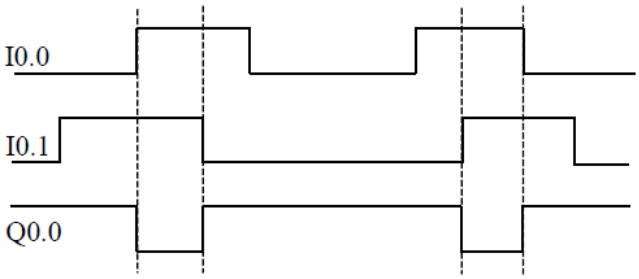
8.2.6: دستور معکوس کننده(NCR):

	Name	Usage	Group	
LD	NCR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	NCR	NCR	P	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BOOL	Power flow
Q	Output	BOOL	Power flow

LD: این دستور معکوس کننده ورودی EN میباشد .چنانچه سیگنال ورودی در پایه EN ، صفر باشد خروجی دستور ۱ و چنانچه ورودی در پایه ENیک باشد خروجی دستور 0 خواهد بود

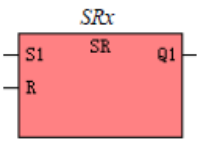
IL: این دستور معکوس کننده CRمیباشد .چنانچه مقدار CR ، صفر باشد خروجی دستور ۱ و چنانچه مقدار CR یک باشد خروجی دستور ، 0 خواهد بود.

LD	IL
	LD %I0.0 AND %I0.1 NCR ST %Q0.0
Time Sequence Diagram	
	

8.2.7: فلیپ فلاپها :

در نرم افزار kincobuilder دو نوع فلیپ فلاپ وجود دارد .فلیپ فلاپ RS ,SR

فلیپ فلاپ SR:

	Name	Usage	Group	
LD	SR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SR	LD *S1* SR *SRx, R*	P	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
SRx	-	SR instance	SR
S1	Input	BOOL	Power flow
R	Input	BOOL	I, Q, V, M, SM, L, T, C, RS, SR
Q1	Output	BOOL	Power flow

دستور SR دارای دو پایه ورودی به نام های(SET) S1و(RESET) Rو یک پایه خروجی به نام Q1 میباشد .SRx بیت بیان کننده وضعیت فلیپ فلاپ وx نشان دهنده شماره فلیپ فلاپ مورد استفاده میباشد.

در این دستور اولویت اجرا با ورودی setمیباشد.به عبارت دیگر چنانچه ورودی SET(S1) و ورودی RESET (R) به طور همزمان دارای مقدار ۱ منطقی باشد اولویت با ورودی SET (S1) میباشد و هم مقدار خروجی Q1 و هم مقدار SRx یک خواهد بود. همان طور که در جدول زیر مشاهده میشود چنانچه ورودی SETو RESETبه طور همزمان دارای مقدار ۰ باشد خروجی وضعیت قبل را حفظ مینماید.

جدول زیر بیان کننده وضعیت خروجی فلیپ فلاپ در حالت ورودی های مختلف است :

S1	*R*	*Q1, SRx*
0	0	Previous value
0	1	0
1	0	1
1	1	1

RS :

	Name	Usage	Group	
LD	RS			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	RS	LD *S* RS *RSx, R1*	P	

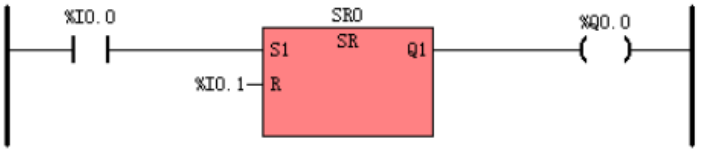
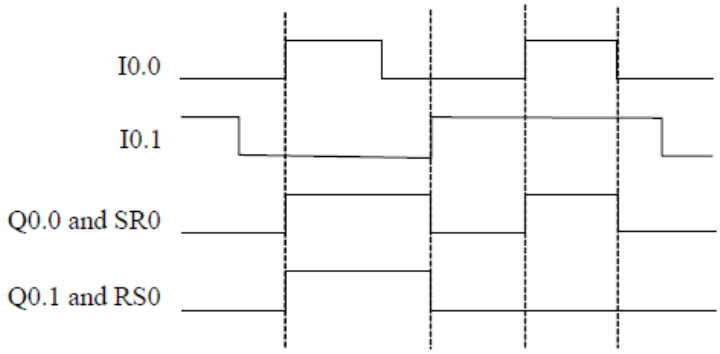
Parameter	Input/Output	Data Type	Acceptable Memory Areas
RSx	-	RS instance	RS
S	Input	BOOL	Power flow
R1	Input	BOOL	I, Q, V, M, SM, L, T, C, RS, SR
Q1	Output	BOOL	Power flow

دستور RS دارای دو پایه ورودی به نام های (S)SET و (R1)Reset و یک خروجی به نام Q1 میباشد .RSx بیت بیان کننده وضعیت فلیپ فلاپ و xنشان دهنده شماره tفلیپ فلاپ مورد استفاده میباشد.

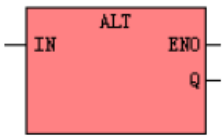
عملکرد این دستور دقیقا مشابه با دستور SR بوده با این تفاوت که در این دستور اولویت اجرا با ورودی RESET(R1) خواهد بود .به عبارت دیگر چنانچه مقدار پایه های ورودی R1 و Sبه طور همزمان برابر با یک منطقی باشد اولویت اجرا با ورودی RESET(R1) خواهد بود وخروجی صفرخواهد بود. در این دستور نیز مانند دستور SR چنانچه ورودیهای SERو RESETبه طور همزمان دارای مقدار صفر باشند خروجی وضعیت قبل را حفظ مینماید.

R1	*S*	*Q1, SRx*
0	0	Previous value
0	1	1
1	0	0
1	1	0

قابل ذکر میباشد که این دو دستور (همان طور که در جدول مشخصات دستور بیان شده است) تنها در CPU های K306EX ,K308 وجود دارند .

LD	IL
(* Network 0 *)  	(* Network 0 *) LD %I0.0 SR SR0, %I0.1 ST %Q0.0
(* Network 1 *)  	(* Network 1 *) LD %I0.0 RS RS0, %I0.1 ST %Q0.1
Time Sequence Diagram	
 	

۸.۲.۸ ALT : (متناوب کننده) :

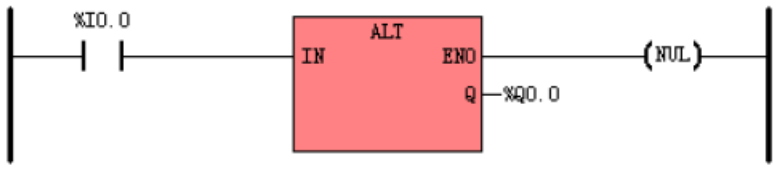
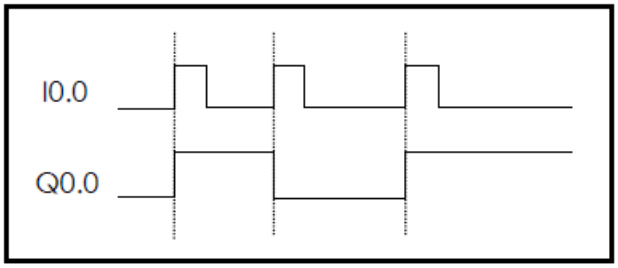
	Name	Usage	Group	
LD	ALT	 		☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	ALT	ALT *Q*	U	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
IN (LD)	Input	BOOL	Power flow
Q	Output	BOOL	Q, V, M, SM, L

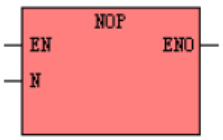
این دستور دارای یک ورودی (که با IN) مشخص شده است و یک خروجی (که با Q مشخص شده است) میباشد . (پایه ENOدر قسمت مفاهیم توضیح داده شده است)

ابن دستور به این صورت عمل میکند که با هر لبه بالارونده در پایه ورودی IN باعث تغییر وضعیت درخروجی میگردد.به این صورت که با فعال شدن ورودی INخروجی نیز فعال شده و با غیر فعال شدن وردی خروجی به همان صورت فعال باقی میماند ،به محض فعال شدن مجدد ورودی (با لبه بالارونده ورودی IN) خروجی به حالت صفر تغییر وضعیت میدهد.به عبارت دیگر شاید بتوان این دستور را نوعی تولید کننده پالس معرفی کرد.

قابل ذکر است که ورودی و خروجی این دستور هر دو ازنوع BOOLEAN میباشند .این دستور تنها در CPUهای مدل K306EXو K308وجود دارد.

LD	IL
	LD %I0.0 ALT %Q0.0
Time Sequence Diagram	
	

NOP : ۸.۲.۹

	Name	Usage	Group	CPU304 / CPU304EX / CPU306 / CPU306EX / CPU308
LD	NOP			
IL	NOP	NOP *N*	U	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
N	Input	INT	Constant (Positive)

این دستور در اصل کار خاصی انجام نمیدهد و فقط برای ایجاد یک وقفه زمانی درروند اجرای دو دستور پشت سرهم استفاده میشود واز نظر عملیاتی تاثیری در برنامه نخواهد داشت .ورودی N به صورت یک عدد integerمثبت میباشد.

این دستور تنها در CPUهای مدل K306EXو K308وجود دارد.

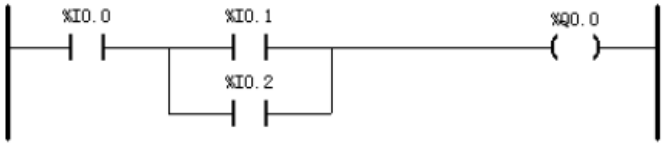
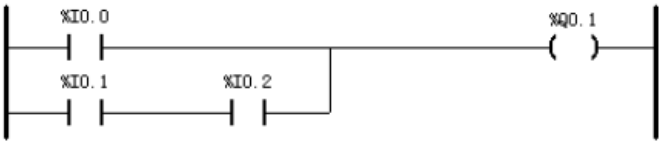
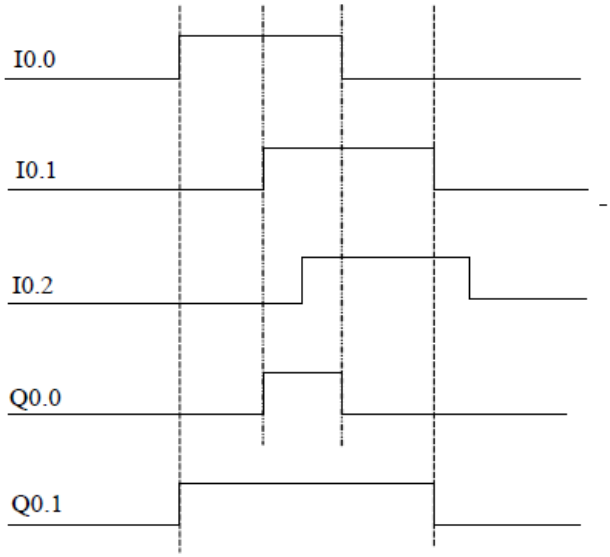
۸.۲.۱۰ Bracket modifier:

	Name	Usage	Group	CPU304 / CPU304EX / CPU306 / CPU306EX / CPU308
IL	AND(	AND(	U	
	OR(	OR(		
	)	)	P	

این دستور تنها در زبان IL وجود دارد .

که به صورت OR(....) , (.....) AND میباشد. در یک برنامه به زبان IL قبل از اجرای عبارت بین دو پرانتز ابتدا CR به صورت موقف ذخیره میگردد.سپس عبارت بین دو پرانتز اجرا میشود و نتیجه این عملیات با CRذخیره شده به صورت موقت ANDو یا ORمیشود.در نهایت مقدار CR برابربا نتیجه این عملیات میگردد.

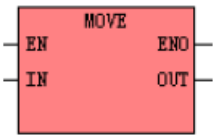
به دلیل عدم وجود این دستور در زبان LDاین عملیات در زبان LDبه صورت دیگری انجام میشود.در تصویر زیر یک نمونه برنامه که در آن از این دستور استفاده شده و معادل آن در زبان LD بیان شده است :

LD	IL
	LD %I0.0 AND(LD %I0.1 OR %I0.2) ST %Q0.0
	LD %I0.0 OR(LD %I0.1 AND %I0.2) ST %Q0.1
Timing Diagram	
	

۸.۳:

دستورات انتقال :

۸.۳.۱ :MOVE

	Name	Usage	Group	
LD	MOVE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	MOVE	MOVE *IN, OUT*	U	

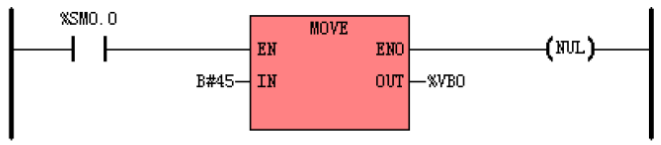
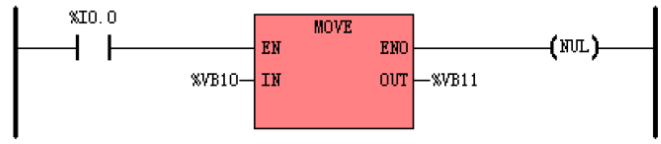
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, WORD, DWORD, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant, pointer
OUT	Output	BYTE, WORD, DWORD, INT, DINT, REAL	Q, M, V, L, SM, AQ, pointer

از

این دستور برای انتقال وجابه جایی داده ها بین خانه های حافظه استفاده میشود. همان طور که در تصویر بالا مشاهده میکنید این دستور دارای یک پایه ورودی IN ویک پایه خروجی OUT میباشد . این دستور قابلیت انتقال اطلاعات با فرمت های مختلف (فرمت های ذکر شده درستون Data type) را در فضاهای مختلف حافظه (مطابق فضاهای عنوان شده در تصویربالا در ستونAcceptable Memory Area)را داراست .شایان توجه است که در این دستور فرمت داده در پایه ورودی و در پایه خروجی باید یکی باشد .به عنوان مثال نمیتوان یک داده از نوع بایت را از یک قسمت از حافظه به فضای دیگری از حافظه به فرمت DWORD انتقال داد . بنابراین باید به این نکته توجه شود که نوع داده در ورودی و نوع داده در خروجی یکسان باشد .

LD :همان طور که در قسمت مفاهیم توضیح داده شد زمانی این دستور اجرا خواهد شد که ENیک باشد

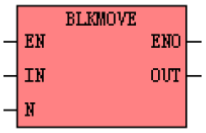
IL: زمانی این دستور اجرا میشود که مقدار CRیک باشد .این دستور تاثیری بر مقدار CR نخواهد داشت.

LD		%SM0.0 is always ON, therefore the MOVE is always executed: B#45 is assigned to %VB0.
		If %I0.0 is 0, the MOVE is not executed. If %I0.0 is 1, the value of %VB10 is assigned to %VB11.

IL	LD %SM0.0 (* The CR is created with %SM0.0 *) MOVE B#45, %VB0 (* B#45 is assigned to %VB0 *)
	LD %I0.0 (* The CR is created with %I0.0 *) MOVE %VB10, %VB11 (* If the CR is 1, the value of %VB10 is assigned to %VB11. *) (*Otherwise, this statement is not executed, %VB11 remains unchanged *)

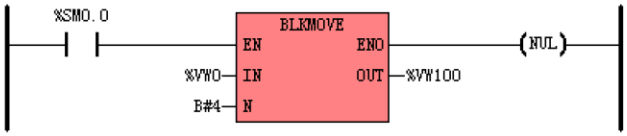
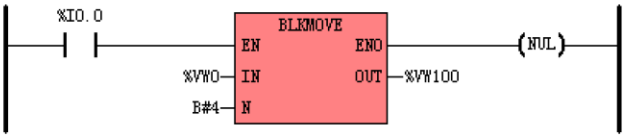
همان طور که در مثال بالا مشاهده مینمایید به محض ۱ شدن ورودی EN مقدار موجود در پایه IN به حافظه خروجی OUT منتقل شده و مقدار ENO میشود.

8.3.2 :BLKMOVE

	Name	Usage	Group	
LD	BLKMOVE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	BLKMOVE	BLKMOVE *IN*, *OUT*, *N*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, WORD, DWORD, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC
N	Input	BYTE	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD, INT, DINT, REAL	Q, M, V, L, SM, AQ

این دستور در اصل انتقال به صورت بلوکی میباشد .به این صورت که چنانچه EN (CR در زبان IL) مقدار یک را داشته باشد این دستور به تعداد Nمتغیر را به صورت پشت سرهم از خانه های حافظه که با آدرس مشخص شده در IN شروع میشود را به N حافظه از خروجی که با آدرس مشخص شده در OUT شروع میشود انتقال میدهد .

LD		%SM0.0 is always ON, therefore the BLKMOVE is always executed: the data in %VW0 - %VW6 are moved into %VW100 - %VW106.
		If %I0.0 is 1, the data in %VW0 - %VW6 are moved into %VW100 - %VW106. Otherwise, the BLKMOVE is not executed.

IL	LD %SM0.0 (* The CR is created with the% SM0.0 *) BLKMOVE %VW0, %VW100, B#4 (* The data in VW0 - VW6 are moved into %VW100 - %VW106 *)
	LD %I0.0 (* The CR is created with %I0.0 *) BLKMOVE %VW0, %VW100, B#4 (* If the CR is 0, this statement isn't executed *) (* If the CR is 1, the data in %VW0 - %VW6 are moved into %VW100 - %VW106 *)

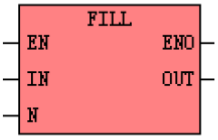
همان طور که در مثال بالا میبینید عدد قرار داده شده در پایه N،۴ و حافظه مشخص شده در پایه IN، VW0 و خروجی OUT، VW100میباشد .پس این دستور 4 متغیر به فرمت WORDرا از آدرس های VW0 ,VW2,VW4,VW6 به ترتیب به خانه های VW100 ,VW102,VW104,VW106 انتقال میدهد.

The following is an example:

VW0	VW2	VW4	VW6
0	10	20	30

VW100	VW102	VW104	VW106
0	10	20	30

:FILL:8.3.3

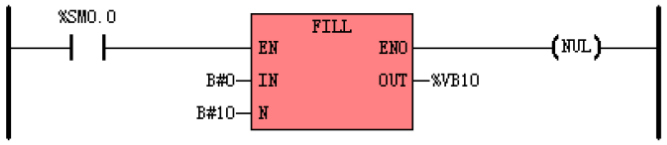
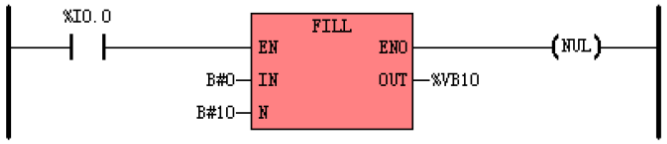
	Name	Usage	Group	
LD	FILL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	FILL	FILL *IN*, *OUT*, *N*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE	Constant
N	Input	BYTE	constant
OUT	Output	BYTE	M, V, L

این دستور تعداد N بایت از حافظه را که با آدرس OUTآغاز میشود را با مقدار ثابت که در ورودی INقرار دارد پر میکند.

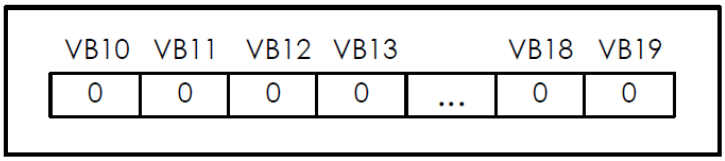
نوع متغیرها در این دستور به صورت بایتی میباشد .

مثال:

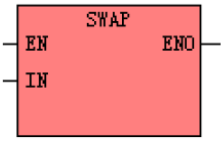
LD		%SM0.0 is always ON, therefore the FILL is always executed: 10 variables from %VB10 to %VB19 are all set to B#0.
		If %I0.0 is 0, the FILL is not executed. If %I0.0 is 1, 10 variables from %VB10 to %VB19 are all set to B#0.

IL	LD %SM0.0	(* The CR is created with %SM0.0 *)
	FILL B#0, %VB10, B#10	(* 10 variables from %VB10 to %VB19 are all set to B#0 *)
	LD %I0.0	(* The CR is created with %I0.0 *)
	FILL B#0, %VB10, B#10	(* If the CR is 1, 10 variables from %VB10 to %VB19 are all set to B#0 *) (* If the CR is 0, this statement is not executed *)

همان طور که د رمثال بالا مشاهده میشود به محض ۱ شدن ورودی EN حافظه های VB10,VB11,....VB19 با مقدار ۰ پر خواهد شد .



8.3.4 SWAP:

	Name	Usage	Group	
LD	SWAP			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SWAP	SWAP *IN*	U	

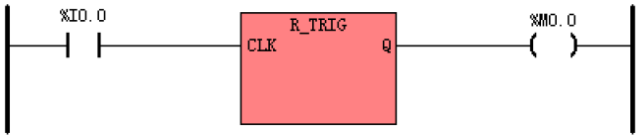
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input/Output	WORD、DWORD	Q, M, V, L, SM

این دستور بایت پرارزش و بایت کم ارزش را در ورودی INچنانچه ورودی به صورت WORD تعریف شده باشد ، جابه جا میکند . چنانچه ورودی به صورت DWORD تعریف شده باشد این دستور WORD پرارزش و کم ارزش را در ورودی IN جا به جا میکند .

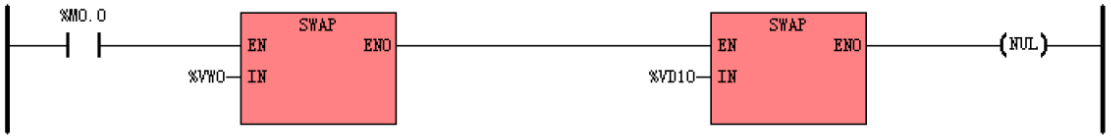
به مثال زیر توجه نمایید:

LD

```
(* Network 0 *)
(* On the rising edge of %I0.0, the following program
   exchanges the most significant byte with the least significant byte of %VW0
   and exchanges the most significant word with the least significant word of %VD10. *)
```



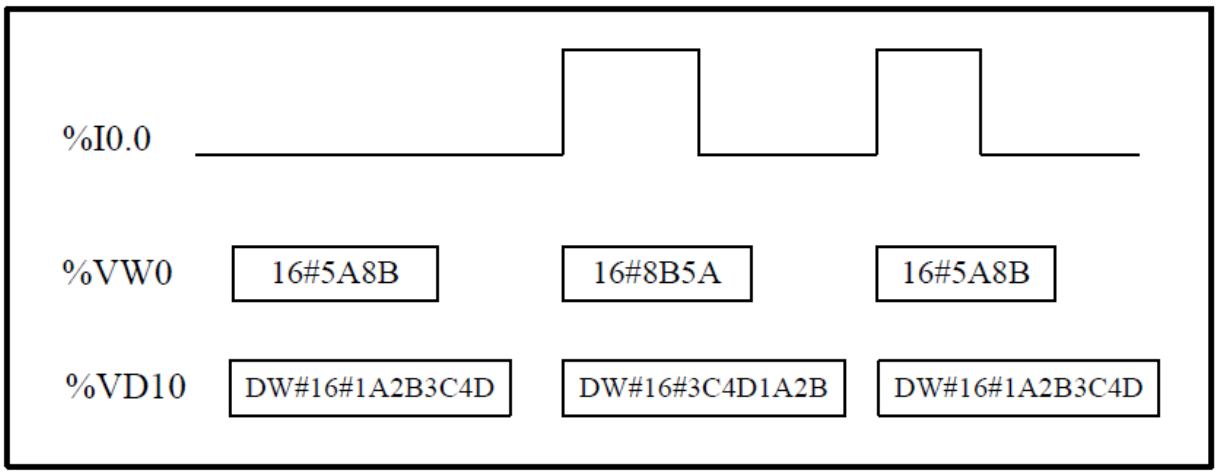

```
(* Network 1 *)
```



IL

```
(* Network 0 *)
LD      %I0.0
R_TRIG          (* On the rising edge of %I0.0, *)
SWAP    %VW0    (* the most significant byte with the least significant byte of %VW0 are exchanged, *)
SWAP    %VD10   (* and the most significant word with the least significant word of %VD10 are exchanged. *)
```

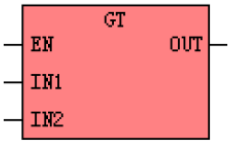
نتیجه اجرای مثال فوق به صورت زیر میباشد:



۸.۴ :دستورات مقایسه ای :

در کلیه دستورات این بخش باید متغیرها (ورودی ها) هم فرمت باشند.

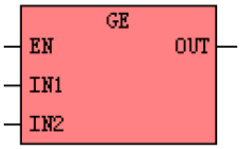
۸.۴.۱ :GT(بزرگتر از):

	Name	Usage	Group	
LD	GT			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	GT	GT *IN1, IN2*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant, pointer
IN2	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant, pointer
OUT (LD)	Output	BOOL	Power flow

در این دستور به محض یک شدن پایه EN ورودی های IN1 , IN2 با یکدیگر مقایسه میشوند .چنانچه IN1>IN2 باشد خروجی OUT یک خواهد شد .

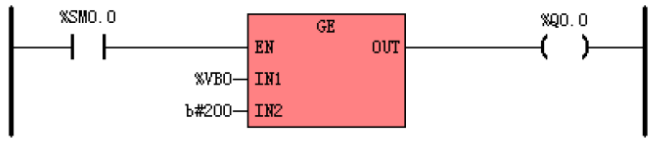
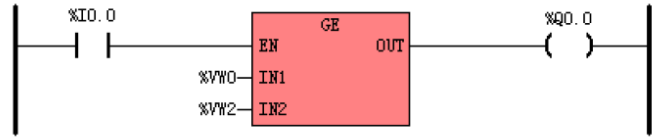
8.4.2 GE: بزرگتر یا مساوی)

	Name	Usage	Group	
LD	GE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	GE	GE *IN1*, *IN2*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
IN2	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
OUT (LD)	Output	BOOL	Power flow

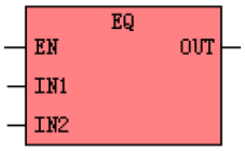
در این دستور به محض یک شدن پایه EN ، چنانچه IN1≥IN2 باشد خروجی یک خواهد شد .

مثال:

LD		SM0.0 is always ON, therefore *GE* is always executed: if VB0 is greater than or equal to B#200, Q0.0 is set equal to 1, otherwise Q0.0 is set equal to 0.
		If I0.0 is 1: if VW0 is greater than or equal to VW2, Q0.0 is set to be 1, otherwise Q0.0 is set to be 0. If I0.0 is 0: *GE* is not executed, and Q0.0 is set to be 0.

IL	LD %SM0.0 (* CR is created with SM0.0 *) GE %VB0, B#200 (* If VB0 is greater than or equal to B#200, CR is set to be 1, otherwise CR is set to be 0 *) ST %Q0.0 (* Q0.0 is set equal to CR *)
	LD %I0.0 (* CR is created with I0.0 *) GE %VW0, %VW2 (* If CR is 1: if VW0 is greater than or equal to VW2, CR is set to be 1, *) (* otherwise CR is set to be 0; If CR is 0: GE is not executed, CR remains 0 *) ST %Q0.0 (* Q0.0 is set equal to CR *)

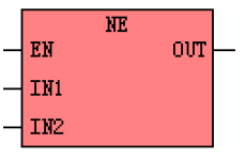
۸.۴.۳ EQ : (مساوی):

	Name	Usage	Group	
LD	EQ			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	EQ	EQ *IN1*, *IN2*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
IN2	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
OUT (LD)	Output	BOOL	Power flow

در این دستور به محض یک شدن EN چنانچه IN1=IN2 باشد مقدار خروجی OUT یک خواهد شد .

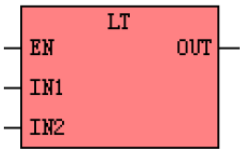
8.4.4 NE (نا مساوی):

	Name	Usage	Group	
LD	NE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	NE	NE *IN1*, *IN2*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
IN2	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
OUT (LD)	Output	BOOL	Power flow

در این دستور با فعال شدن پایه EN چنانچه IN2 ≠ IN1 باشد خروجی فعال خواهد شد .

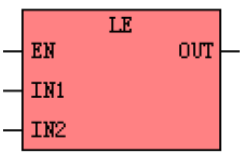
۸.۴.۵ LT(کوکتر از):

	Name	Usage	Group	
LD	LT			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	LT	LT *IN1*, *IN2*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
IN2	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
OUT (LD)	Output	BOOL	Power flow

در این دستور با فال شدن پایه EN،چنانچه IN1< IN2 باشد ،خروجی فعال خواهد شد .

8.4.6 LE(کوچکتر یا مساوی):

	Name	Usage	Group	
LD	LE			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	LE	LE *IN1*, *IN2*	P	

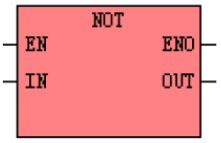
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
IN2	Input	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, constant
OUT (LD)	Output	BOOL	Power flow

در این دستور با فعال شدن پایه EN، چنانچه IN1≤IN2 باشد ، خروجی فعال خواهد شد .

8.5 : دستورات منطقی :

در این دستورات ورودی ها و خروجی ها باید هم فرمت باشند.

۸.۵.۱ :NOT

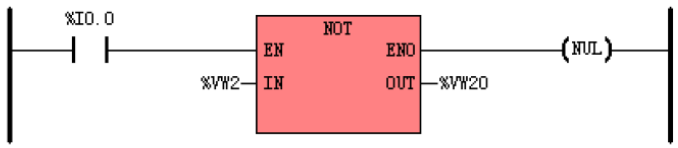
	Name	Usage	Group	
LD	NOT			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	NOT	NOT *OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD:در این دستور با فعال شدن EN، بیت های متغیر در ورودی IN نظیر به نظیر معکوس شده و نیجه در حافظه مشخص شده در خروجی OUT دخیره میگردد.

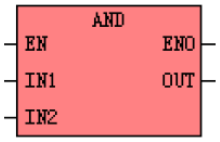
IL: چنانچه مقدار CRیک باشد ، بیتهای OUT نظیر به نظیر معکوس شده و در OUTذخیره میگردد.این دستور تاثیری بر مقدار CRنخواهد داشت .

مثال:

LD		If I0.0 is 0: *NOT* is not executed. If I0.0 is 1: *NOT* inverts each bit of VW2 and assigns the result to VW20.

IL	LD %I0.0 (* CR is created with I0.0 *) NOT %VW20 (* If CR is 1: *NOT* inverts each bit of VW20 and still stores the result in VW20 *) (* If CR is 0: *NOT* instruction is not executed *)
Result	For the LD example, if *NOT* instruction is executed, the result will be as the following: Address VW2 Value W#16#5555 Address VW20 Value W#16#AAAA

AND:8.5.2

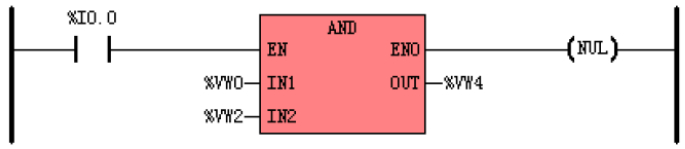
	Name	Usage	Group	
LD	AND			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	AND	AND *IN*, *OUT*	U	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
IN2	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: با فعال شدن EN، بیتهای ورودی های IN1وIN2 نظیر به نظیر با یکدیگر AND شده و خروجی د ر OUTذخیره میگردد.

IL: در صورتی که CRیک باشد ، بیتهای INو OUT نظیر به نظیر با یکدیگر ANDشده و خروجی در OUTذخیره میگردد.این دستور تاثیری بر مقدار CR نخواهد داشت .

مثال:

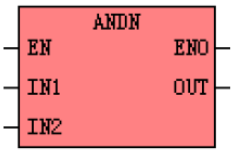
LD		If I0.0 is 0: *AND* is not executed. If I0.0 is 1: The *AND* instruction ANDs the corresponding bits of VW0 and VW2, and assigns the result to VW4.
IL	LD %I0.0 (* CR is created with I0.0 *) AND %VW0, %VW2 (* If CR is 1: The *AND* instruction ANDs the corresponding bits of VW0 and VW2, *) (* and still stores the result in VW2 *) (* If CR is 0: The *AND* instruction is not executed *)	

For the LD example, if *AND* instruction is executed, the result will be as the following:

Address	VW0	VW2
Value	W#16#129B	W#16#960F

Address	VW4
Value	W#16#120B

8.5.3: ANDN

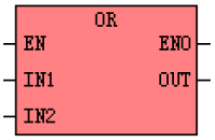
	Name	Usage	Group	
LD	ANDN			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ANDN	ANDN *IN, OUT*	U	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
IN2	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: با فعال شدن EN ، بیتهای IN 1 وIN2 نظیر به نظیر با یکدیگر ANDشده ، بیتهای نتیجه معکوس شده و در OUT ذخیره میگردد.

IL: چنانچه CR یک باشد ، بیتهای IN و OUTنظیر به نظیر با یکدیگر AND شده ، بیتهای نتیجه معکوس شده و در OUT ذخیره میگردد.این دستور تاثیری بر مقدار CR نخواهد داشت .

8.5.4: OR

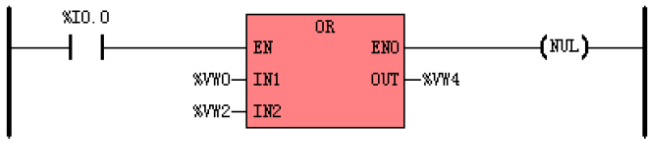
	Name	Usage	Group	
LD	OR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	OR	OR *IN, OUT*	U	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
IN2	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: با فعال شدن EN، بیتهای IN1و IN2نظیر به نظیر با یکدیگر ORشده و نتیجه د رOUT ذخیره میگردد.

IL:چنانچه مقدار CRیک باشد ، بیتهای IN وOUT نظیر به نظیر با یکدیگر ORشده و نتیجه در OUT ذخیره میگردد . این دستور تاثیری بر مقدار CRنخواهد داشت .

مثال:

LD		If I0.0 is 0: *OR* is not executed. If I0.0 is 1: The *OR* instruction ORs the corresponding bits of VW0 and VW2, and assigns the result to VW4.

IL	LD	%I0.0	(* CR is created with I0.0 *)
	OR	%VW0, %VW2	(* If CR is 1: The *OR* instruction ORs the corresponding bits of VW0 and VW2, *) (* and still stores the result in VW2 *) (* If CR is 0: The *OR* instruction is not executed *)

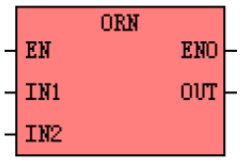
نتیجه به صورت زیرخواهد بود :

For the LD example, if *OR* instruction is executed, the result will be as the following:

Address	VW0	VW2
Value	W#16#5555	W#16#AAAA

Address	VW4
Value	W#16#FFFF

8.5.5 ORN :

	Name	Usage	Group	
LD	ORN			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ORN	ORN *IN, OUT*	U	

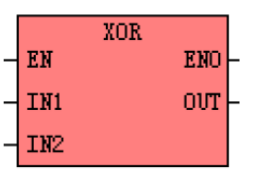
Parameter	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
IN2	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: با فعال شدن EN ، بیتهای ورودی های IN1و IN2 نظیر به نظیر با یکدیگر OR شده و بیتهای نتیجه نظیر به نظیرمعکوس میگردد و حاصل در خروجی OUTذخیره میگردد.

IL: چنانچه مقدار CR یک باشد ، بیتهای IN و OUT نظیر به نظیر با یکدیگر ORشده ،بیتهای نتیجه یک به یک معکوس شده و حاصل در OUT ذخیره میگردد .

این دستور تاثیری بر مقدار CR نخواهد داشت .

8.5.6 XOR :

	Name	Usage	Group	
LD	XOR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	XOR	XOR *IN, OUT*	U	

Parameter	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
IN2	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD:

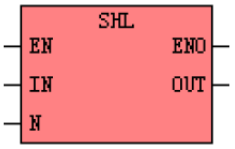
با فعال شدن EN، بیتهای ورودی IN1و IN2 نظیر به نظیر با یکدیگر XOR شده و نتیجه در OUT ذخیره میگردد.

IL: چنانچه مقدار CR یک باشد ، بیتهای INو OUTنظیر به نظیر با یکدیگر XOR شده و نتیجه در OUT ذخیره میگردد.این دستور تاثیری بر مقدار CR نخواهد داشت .

8.6: دستورات شیفت و چرخش :

این دستورات جهت جابه جایی و نیز چرخش بیت های رجیسترها به سمت چپ یا راست میباشد . در این دستورات ورودی IN و OUT باید هم فرمت باشند.

8.6.1: SHL(شیفت به سمت چپ) :

	Name	Usage	Group	
LD	SHL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHL	SHL *OUT, N*	U	

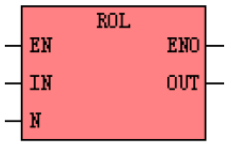
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
N	Input	BYTE	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: با هر بار فعال شدن EN، مقدار ورودی IN به تعداد Nبیت به سمت چپ شیفت داده میشود .

مثال تست شود

IL: چنانچه مقدار CR یک باشد ، این دستور مقدار out را به تعداد N بیت به سمت چپ شیفت میدهدو جای هر بیتی که به سمت چپ شیفت پیدا میکند با صفر پر میشود .مقدار نهایی در out ذخیره میگردد .

8.6.2 ROL (چرخش چپ):

	Name	Usage	Group	
LD	ROL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ROL	ROL *OUT, N*	U	

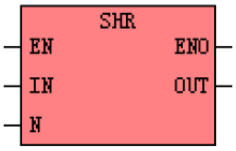
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
N	Input	BYTE	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: چنانچه مقدار EN یک باشد ، این دستور مقدار ورودی IN را به تعداد Nبیت از سمت چپ به سمت راست گردش میدهد . به عبارت دیگر بیت های با ارزش بالایی (MSB) جای بیت های بی ارزش (LSB) قرار میگیرند و نتیجه در خروجی OUT ذخیره میگردد . دستور چک شود

IL:چنانچه مقدار CR یک باشد ، این دستور مقدار outرا به تعداد Nبیت از سمت چپ به سمت راست گردش میدهد .به عبارت دیگر بیت های با ارزش بالایی (MSB) جای بیت های بی ارزش (LSB) قرار میگیرند و نتیجه مجددا در خروجی OUT ذخیره میگردد .

مثال تست شود

8.6.3 SHR (شیفت به سمت راست):

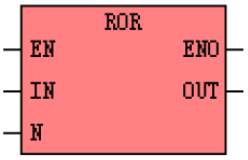
	Name	Usage	Group	
LD	SHR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHR	SHR *OUT, N*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
N	Input	BYTE	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: با هر بار فعال شدن EN، مقدار ورودی IN به تعداد Nبیت به سمت راست شیفت داده میشود .

IL: چنانچه مقدار CR یک باشد ، این دستور مقدار out را به تعداد N بیت به سمت راست شیفت میدهدو جای هر بیتی که به سمت راست شیفت پیدا میکند با صفر پر میشود .مقدار نهایی در out ذخیره میگردد .

۸.۶.۴ :ROR(چرخش راست):

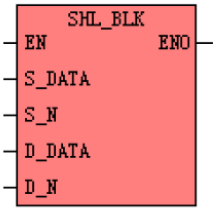
	Name	Usage	Group	
LD	ROR			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	ROR	ROR *OUT, N*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, WORD, DWORD	I, Q, M, V, L, SM, constant
N	Input	BYTE	I, Q, M, V, L, SM, constant
OUT	Output	BYTE, WORD, DWORD	Q, M, V, L, SM

LD: چنانچه مقدار EN یک باشد ، این دستور مقدار ورودی IN را به تعداد Nبیت از سمت راست به سمت چپ گردش میدهد . به عبارت دیگر بیت های با ارزش بالایی (LSB) جای بیت های بی ارزش (MSB) قرار میگیرند و نتیجه در خروجی OUT ذخیره میگردد . دستور چک شود

IL:چنانچه مقدار CR یک باشد ، این دستور مقدار outرا به تعداد Nبیت از سمت راست به سمت چپ گردش میدهد .به عبارت دیگر بیت های با ارزش بالایی (LSB) جای بیت های بی ارزش (MSB) قرار میگیرند و نتیجه مجددا در خروجی OUT ذخیره میگردد

:SHL_BLK:8.6.5

	Name	Usage	Group	
LD	SHL_BLK			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHL_BLK	SHL_BLK *S_DATA, S_N, D_DATA, D_N*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
S_DATA	Input	BOOL	I, Q, M, V, L
S_N	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constant, Pointer
D_DATA	Input/Output	BOOL	Q, M, V, L
D_N	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constant, Pointer

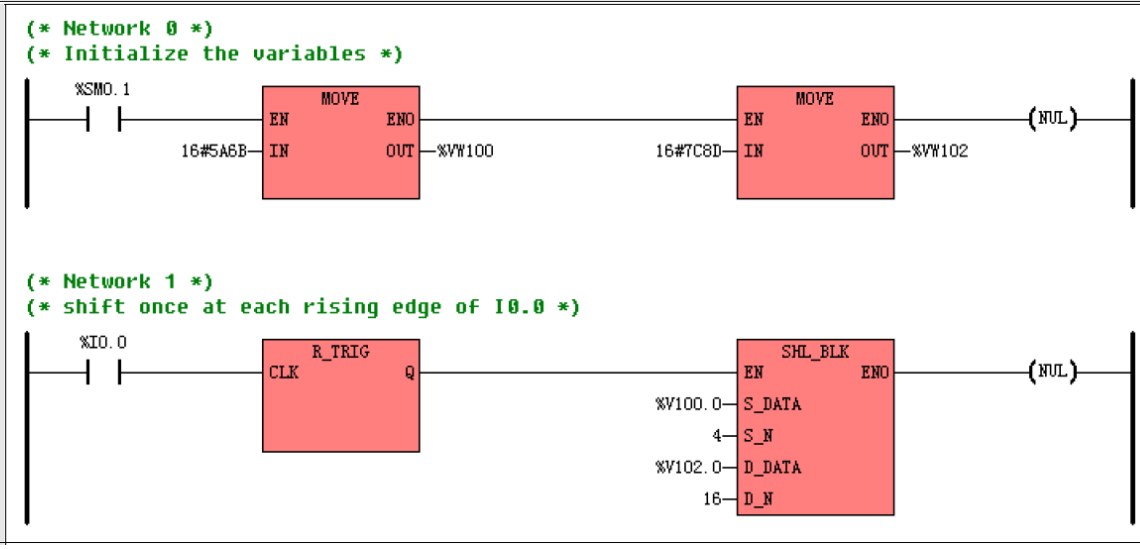
این دستور مانند دستور SHL عمل میکند با این تفاوت که یک بلوک از داده ها را همزمان انتقال میدهد .این دستور به تعداد D_N بیت پست سرهم و پیوسته را که با آدرس D_DATA آغاز میشود را به سمت چپ انتقال داده و جای آن را با S_Nبیت متوالی که با آدرس S_DATA آغاز میشود پر میگردد .

LD: چنانچه مقدار EN یک باشد این دستوراجرا میگردد .

IL: چنانچه مقدار CR یک باشد این دستور اجرا میگردد . این دستور تاثیری برمقدار CRنخواهد داشت .

دستور و مثال چک شود .

LD



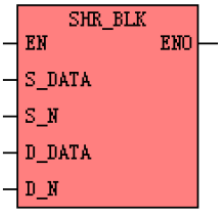
IL	(* Network 0 *)
	(*Initialize the variables*)
	LD %SM0.1
	MOVE 16#5A6B, %VW100
	MOVE 16#7C8D, %VW102
	(* Network 1 *)
	(*shift once at each rising edge of I0.0*)
	LD %I0.0
	R_TRIG
	SHL_BLK %V100.0, 4, %V102.0, 16

Result

The result is shown as the following:

	VW102				VW100			
	V103.7			V102.0	V101.7			V100.0
Initial value	0111	1100	1000	1101	0101	1010	0110	1011
After the 1st execution	1100	1000	1101	1011				
After the 2nd execution	1000	1101	1011	1011				
After the 3rd execution	1101	1011	1011	1011				

8.6.6 SHR_BLK:

	Name	Usage	Group	
LD	SHR_BLK			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SHR_BLK	SHR_BLK *S_DATA, S_N, D_DATA, D_N*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
S_DATA	Input	BOOL	I, Q, M, V, L
S_N	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constant, Pointer
D_DATA	Input/Output	BOOL	Q, M, V, L
D_N	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constant, Pointer

این دستور مانند دستور SHR عمل میکند با این تفاوت که یک بلوک از داده ها را همزمان انتقال میدهد .این دستور به تعداد D_N بیت پست سرهم و پیوسته را که با آدرس D_DATA آغاز میشود را به سمت راست انتقال داده و جای آن را با S_Nبیت متوالی که با آدرس S_DATA آغاز میشود پر میگردد .

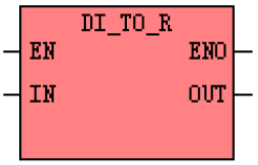
LD: چنانچه مقدار EN یک باشد این دستوراجرا میگردد .

IL: چنانچه مقدار CR یک باشد این دستور اجرا میگردد . این دستور تاثیری برمقدار CRنخواهد داشت .

8.7: دستورات تبدیل :

این دستورات انواع داده ها را به یکدیگر تبدیل میکنند.به عبارت دیگر با استفاده از این دستورات میتوان یک نوع داده را به نوع دیگری تبدیل کرد.

8.7.1: DI_TO_R (DINT To Real) :

	Name	Usage	Group	
LD	DI_TO_R			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	DI_TO_R	DI_TO_R *IN, OUT*	U	

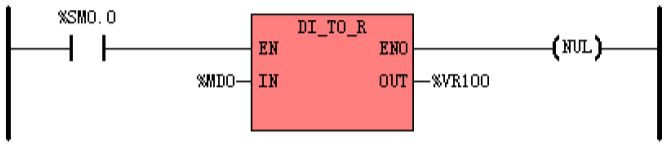
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	DINT	I, Q, M, V, L, SM, HC, constant
OUT	Output	REAL	V, L

این دستورداده با فرمت DINT را در ورودی به داده با فرمت Real تبدیل میکند و نتیجه در خروجی OUT ذخیره میگردد.

LD:چنانچه مقدار EN یک باشد این دستور اجرا میگردد .

IL: چنانچه مقدار CRیک باشد این دستور اجرا میگردد .این دستور تاثیری بر مقدار CR نخواهد گذاشت .

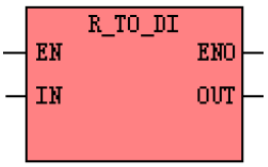
به مثال زیر توجه نمایید:

LD		SM0.0 is always ON, therefore *DI_TO_R* is always executed: The value of MD0 is converted to a REAL value that is assigned to VR100.
IL	LD %SM0.0 (* CR is created SM0.0 *) DI_TO_R %MD0, %VR100 (* The value of MD0 is converted to a REAL value that is assigned to VR100 *)	

نتیجه به صورت زیر خواهد بود :

MD0	VR100
DI#123	123.0
DI#-9876	-9876.0

8.7.2 R_TO_DI (REAL TO DINT):

	Name	Usage	Group	
LD	R_TO_DI			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	R_TO_DI	R_TO_DI *IN, OUT*	U	

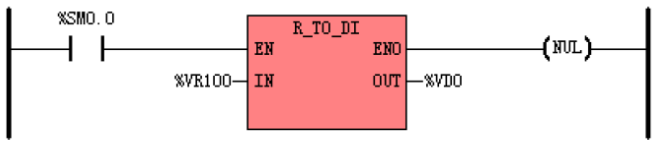
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, constant
OUT	Output	DINT	M, V, L, SM

این دستور داده با فرمت REAL را در ورودی به داده با فرمت DINTتبدیل کرده و در خروجی OUT ذخیره مینماید . در این دستور در اصل مقدار اعشار موجود در ورودی حذف میشود.

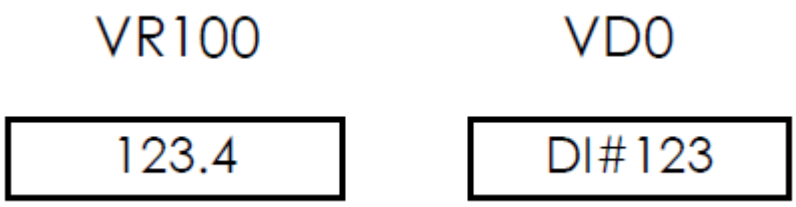
LD:چنانچه مقدار EN یک باشد این دستور اجرا میگردد .

IL: چنانچه مقدار CRیک باشد این دستور اجرا میگردد .این دستور تاثیری بر مقدار CR نخواهد گذاشت .

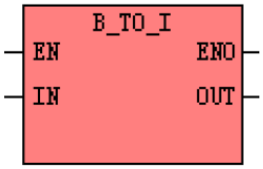
به مثال زیر توجه نمایید:

LD		SM0.0 is always ON, therefore *R_TO_DI* is always executed: The value of VD4000 is converted to a DINT value that is assigned to VD0.
IL	LD %SM0.0 (* CR is created SM0.0 *) R_TO_DI %VD4000, %VD0 (* The value of VD4000 is converted to a DINT value that is assigned to VD0 *)	

نتیجه به صورت زیر خواهد شد :



B_TO_INT:8.7.3 (BYTE TO INT)

	Name	Usage	Group	
LD	B_TO_I			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	B_TO_I	B_TO_I *IN*, *OUT*	U	

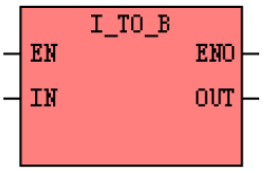
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE	I, Q, M, V, L, SM, Constant
OUT	Output	INT	Q, M, V, L, SM, AQ

این دستور داده با فرمت بایت را در ورودی به داده با فرمت INT تبدیل کرده و نتیجه را در خروجی ذخیره میکند .

LD:چنانچه مقدار EN یک باشد این دستور اجرا میگردد .

IL: چنانچه مقدار CRیک باشد این دستور اجرا میگردد .این دستور تاثیری بر مقدار CR نخواهد گذاشت .

I_TO B:8.7.4 (INT TO BYTE):

	Name	Usage	Group	
LD	I_TO_B			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_B	I_TO_B *IN*, *OUT*	U	

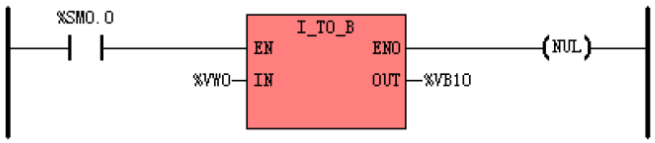
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	INT	I, Q, M, V, L, SM, AI, AQ, T, C, Constant
OUT	Output	BYTE	Q, M, V, L, SM

این دستور داده با فرمت INT در ورودی را به داده با فرمت BYTE تبدیل کرده و درخروجی OUT ذخیره میکند.

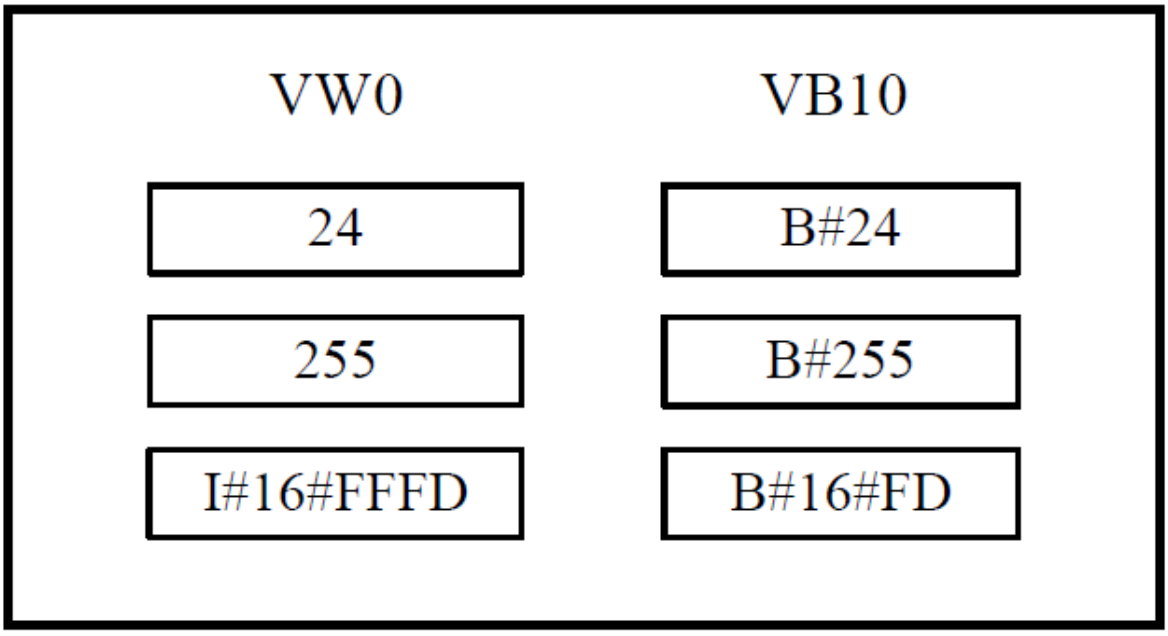
نکته ای که در این دستور باید مورد توجه قرار گیرد مقدار عددی ماکزیممی است که هر کدام از داده های بایت و INT میتواند بپذیرد . به عبارت دیگر چنانچه داده ورودی که فرمت آن به صورت INT میباشد بیشتر از مقدار عددی ۲۵۵ (که حد ماکزیمم مجاز در داده با فرمت بایتی است) باشد ، این دستور ۸ بیت پایین مقدار ورودی را در خروجی ذخیره میکند و بقیه مقدار ورودی از بین میرود.

به عبارت دیگر این دستور در اصل بایت پایین ورودی با فرمت INT را در خروجی ذخیره میکند .

برای درک بهترمطلب بالا به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so I_TO_B is always be executed: assigns VB0 (the least byte of VW0) to VB10.
IL	LD %SM0.0 I_TO_B %VW0, %VB10	

نتیجه به صورت زیر میباشد:



8.7.5: DI_TO_I (DINT TO INT):

	Name	Usage	Group	
LD	DI_TO_I	DI_TO_I EN ENO IN OUT		☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	DI_TO_I	DI_TO_I *IN, OUT*	U	

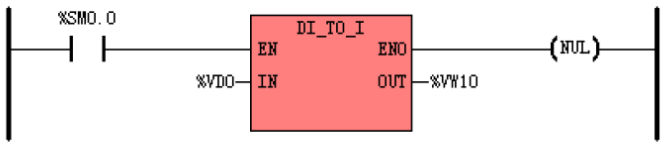
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	DINT	I, Q, M, V, L, SM, HC, Constant
OUT	Output	INT	Q, M, V, L, SM, AQ

این دستور داده با فرمت DINT در ورودی را به فرمت INT تبدیل کرده و نتیجه را در خروجی OUT ذخیره مینماید .

در این دستور نیز باید مقدار ماکزیممی که هر متغیر میتواند بپذیرد توجه شود . برای توضیحات بیشتر به دستور بالا مراجعه نمایید.

به عبارت دیگر این دستور در اصل دو بایت پایینی ورودی با فرمت DINT را در خروجی ذخیره میکند.

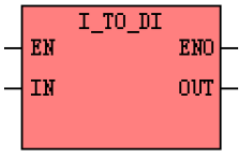
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so DI_TO_I is always be executed: assigns VW0 (the least word of VD0) to VW10.
IL	LD %SM0.0 DI_TO_I %VD0, %VW10	

نتیجه به صورت زیر خواهد شد :

VD0	VW10
DI#12345	12345
DI#-234	-234
DI#16#7A8B9C1D	I#16#9C1D

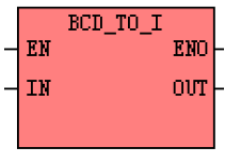
8.7.6 I_TO_DI (INT TO DINT):

	Name	Usage	Group	
LD	I_TO_DI			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_DI	I_TO_DI *IN, OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	INT	I, Q, M, V, L, SM, AI, AQ, T, C, Constant
OUT	Output	DINT	Q, M, V, L, SM

این دستور داده با فرمت INTدر ورودی را به داده با فرمت DINT تبدیل کرده و نتیجه را در خروجی ذخیره میکند.

8.7.7 BCD_TO_I (BCD TO INT)

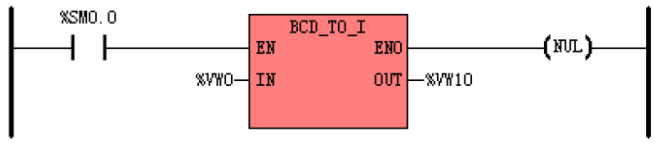
	Name	Usage	Group	
LD	BCD_TO_I			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	BCD_TO_I	BCD_TO_I *IN, OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	WORD	I, Q, M, V, L, SM, Constant
OUT	Output	INT	Q, M, V, L, SM, AQ

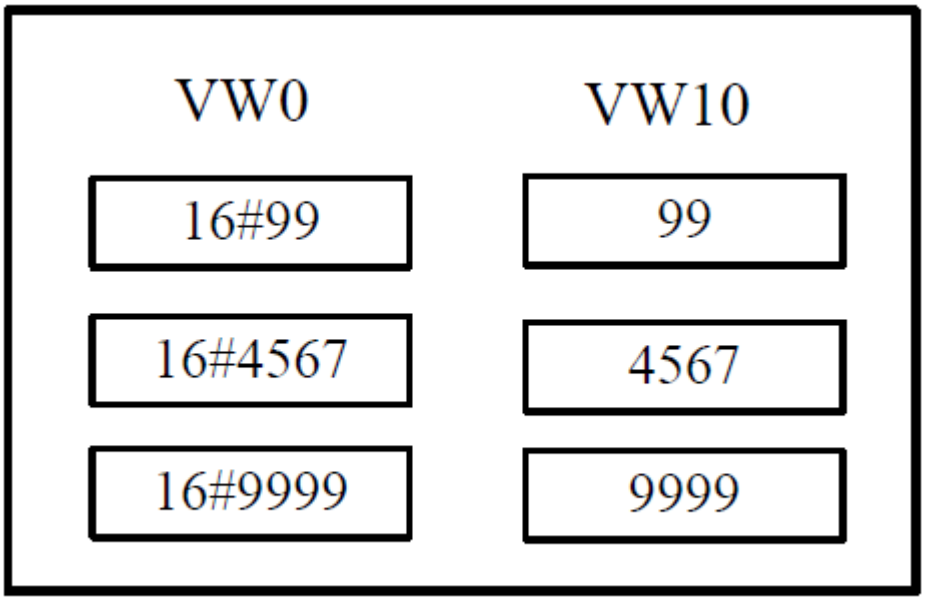
این دستور داده با فرمت BCD (Binary Coded Decimal) را به داده با فرمت INTتبدیل کرده و نتیجه را درخروجی ذخیره میکند.

مقدار BCD مجازی که در ورودی میتواند قرار بگیرد در رنج ۰ تا ۹۹۹۹ میباشد .

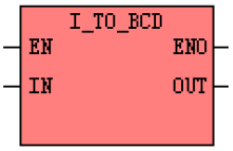
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so BCD_TO_I is always be executed: converts VW0 from BCD to an integer and assigns it to VW10.
IL	LD %SM0.0 BCD_TO_I %VW0, %VW10	

نتیجه به صورت زیر خواهد بود :



8.7.8: I_TO_BCD :

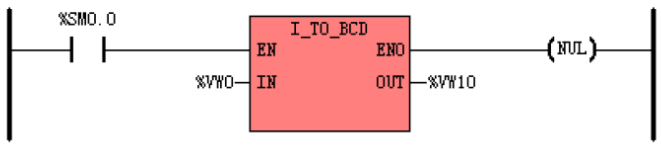
	Name	Usage	Group	
LD	I_TO_BCD			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_BCD	I_TO_BCD *IN*, *OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	INT	I, Q, M, V, L, SM, AI, AQ, T, C, Constant
OUT	Output	WORD	Q, M, V, L, SM

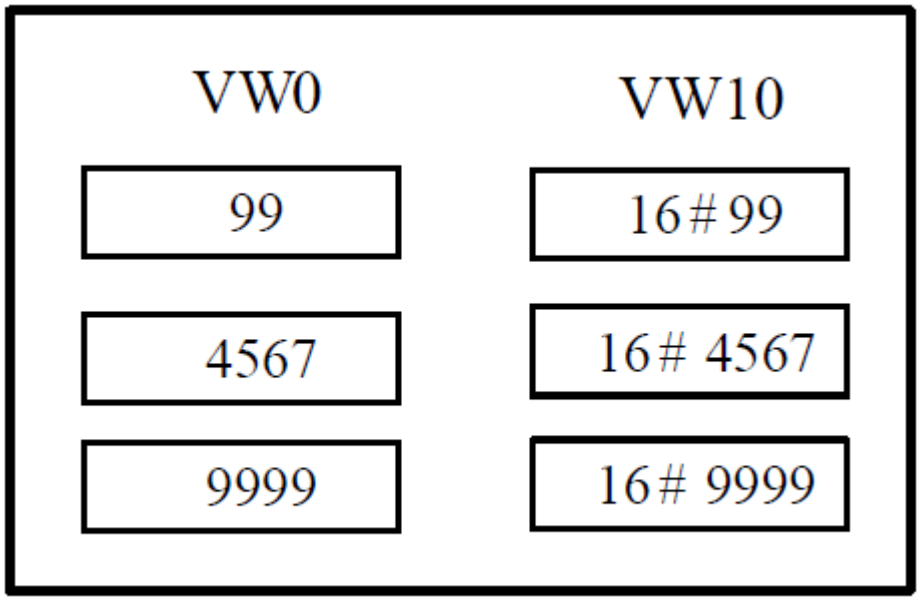
این دستور داده با فرمت INT در ورودی را به داده با فرمت BCD تبدیل کرده و نتیجه را در خروجی OUT ذخیره میکند.

ماکزیمم مقدار مجاز در ورودی در رنج ۰ تا ۹۹۹۹ میباشد.(چک شود) مثال هم چک شود)

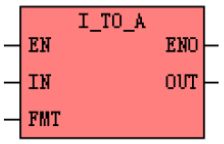
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so I_TO_ BCD is always be executed: converts VW0 to a BCD value and assigns it to VW10.
IL	LD %SM0.0 I_TO_ BCD %VW0, %VW10	

نتیجه به صورت زیر خواهد شد :



8.7.9 I_ TO _A (INT TO ASCII):

	Name	Usage	Group	
LD	I_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	I_TO_A	I_TO_A *IN, OUT, FMT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	INT	I, Q, M, V, L, SM, AI, AQ, T, C, Constant
FMT	Input	BYTE	I, Q, M, V, L, SM
OUT	Output	BYTE	Q, M, V, L, SM

این دستور داده با فرمت INTدر ورودی را به یک رشته ASCII تبدیل میکند. فرمت این رشته بر اساس FMTمیباشد و نتیجه در بافر خروجی که با آدرس OUT آغاز میشود ذخیره میگردد.

نتیجه تبدیل یک مقدار مثبت شامل هیچ علامتی نمیباشد ولی نتیجه تبدیل یک مقدار منفی به همراه یک علامت (-) خواهد بود .

آدرسی که در خروجی OUTقرار میگیرد نشان دهنده آدرس شروع بافر خروجی میباشد . از این آدرس به تعداد ۸ بایت پشت سرهم برای ذخیره نتیجه در حافظه مشخص شده در نظر گرفته میشود .در این بافر خروجی بایت های خالی با فاصله (space) که کد اسکی آن ۳۲ میباشد پر میشوند.

FMT به منظور فرمت دادن به این رشته مورد استفاده قرار میگیرد .الگوی FMT در پایین توضیح داده شده است :

MSB LSB

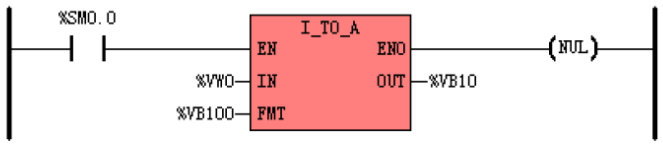
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

(1) *nnn* --- This field specifies the number of digits of the decimal part.
Its available rang is 0 to 5. 0 stands for no decimal part.

(2) *c* --- This field specifies the separator between the whole number and the fraction:
0 for a decimal point (whose ASCII is 46), and 1 for a comma(whose ASCII is 44).

(3) The upper 4 bits must be zero.

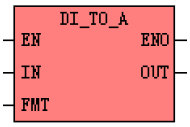
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so the instruction I_TO_A is always executed: converts the value of VW0 to a string, and format the string and put the result to a buffer beginning with VB10.
IL	LD %SM0.0 I_TO_A %VW0, %VB10, %VB100	

نتیجه به صورت زیر خواهد شد :

VB100	VW0	Result							
		VB10							VB17
B#3	12	32	32	32	48	46	48	49	50
		' '	' '	' '	'0'	'.'	'0'	'1'	'2'
	-23456	32	45	50	51	46	52	53	54
		' '	'-'	'2'	'3'	'.'	'4'	'5'	'6'

8.7.10 DI_T_A (DINT TO ASCII):

	Name	Usage	Group	
LD	DI_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	DI_TO_A	DI_TO_A *IN, OUT, FMT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	DINT	I, Q, M, V, L, SM, HC, Constants
FMT	Input	BYTE	I, Q, M, V, L, SM
OUT	Output	BYTE	Q, M, V, L, SM

این دستور داده با فرمت DINTدر ورودی را به یک رشته اسکی تبدیل میکند. فرمت این رشته بر اساس FMTمیباشد و نتیجه در بافر خروجی که با آدرس OUT آغاز میشود ذخیره میگردد. نتیجه تبدیل یک مقدار مثبت شامل هیچ علامتی نمیباشد ولی نتیجه تبدیل یک مقدار منفی به همراه یک علامت (-) خواهد بود . آدرسی که در خروجی OUTقرار میگیرد نشان دهنده آدرس شروع بافر خروجی میباشد . از این آدرس به تعداد۱۲ بایت پشت سرهم برای ذخیره نتیجه در حافظه مشخص شده در نظر گرفته میشود.

دربافر خروجی بایت های خالی با فاصله (space) که کد اسکی آن ۳۲ میباشد پر میشوند.

FMT به منظور فرمت دادن به این رشته مورد استفاده قرار میگیرد .الگوی FMT در پایین توضیح داده شده است :

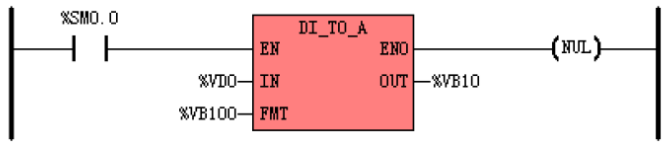
MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

(1) *nnn* --- This field specifies the number of digits of the decimal part.
Its available rang is 0 to 5. 0 stands for no decimal part.

(2) *c* --- This field specifies the separator between the whole number and the fraction:
0 for a decimal point (whose ASCII is 46), and 1 for a comma(whose ASCII is 44).

(3) The upper 4 bits must be zero.

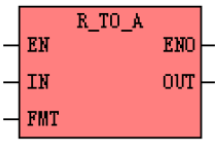
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so the instruction DI_TO_A is always executed: Convert the value of VD0 to a string, and format the string and put it to a buffer beginning with VB10.
IL	LD %SM0.0 DI_TO_A %VD0, %VB10, %VB100	

نتیجه به صورت زیر خواهد بود:

VB 100	VD 0	Result											
		VB10											VB21
B# 3	DI#12	32	32	32	32	32	32	32	48	46	48	49	50
		‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘0’	‘.’	‘0’	‘1’	‘2’
	DI#123456	32	32	32	32	45	49	50	51	46	52	53	54
		‘ ’	‘ ’	‘ ’	‘ ’	‘-’	‘1’	‘2’	‘3’	‘.’	‘4’	‘5’	‘6’

8.7.11 R_TO_A (REAL TO ASCII):

	Name	Usage	Group	
LD	R_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	R_TO_A	R_TO_A *IN, OUT, FMT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, Constants
FMT	Input	BYTE	I, Q, M, V, L, SM
OUT	Output	BYTE	Q, M, V, L, SM

این دستور داده با فرمت REAL در ورودی را به یک رشته اسکی تبدیل میکند. فرمت این رشته بر اساس FMT میباشد و نتیجه در بافر خروجی که با آدرس OUT آغاز میشود ذخیره میگردد. نتیجه تبدیل یک مقدار مثبت شامل هیچ علامتی نمیباشد ولی نتیجه تبدیل یک مقدار منفی به همراه یک علامت (-) خواهد بود .

چنانچه تعداد رقم های اعشار ورودی از مقدار nnn در FMT(که مشخص کننده تعداد رقم اعشار در رشته اسکی میباشد) بزرگتر بود ابتدا ورودی اصطلاحا گرد شده و سپس عملیات تبدیل انجام میشود . چنانچه تعداد رقم اعشار ورودی کمتر از مقدار nnn باشد بقیه رقم

های اعشار باقی مانده صفر در نظر گرفته خواهد شد . آدرسی که در خروجی OUTقرار میگیرد نشان دهنده آدرس شروع بافر خروجی میباشد که سایز آن از طریق FMT مشخص میشود. دربافر خروجی بایت های خالی با فاصله (space) که کد اسکی آن ۳۲ میباشد پر میشوند. الگوی FMT در پایین توضیح داده شده است :

MSB LSB

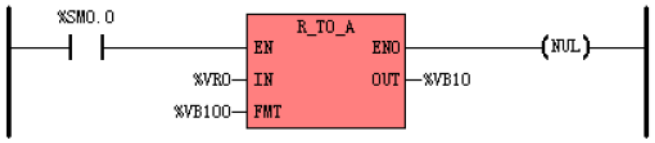
7	6	5	4	3	2	1	0
s	s	s	s	c	n	n	n

(1) *nnn* --- This field specifies the number of digits of the decimal part.
Its available rang is 0 to 5. 0 stands for no decimal part.

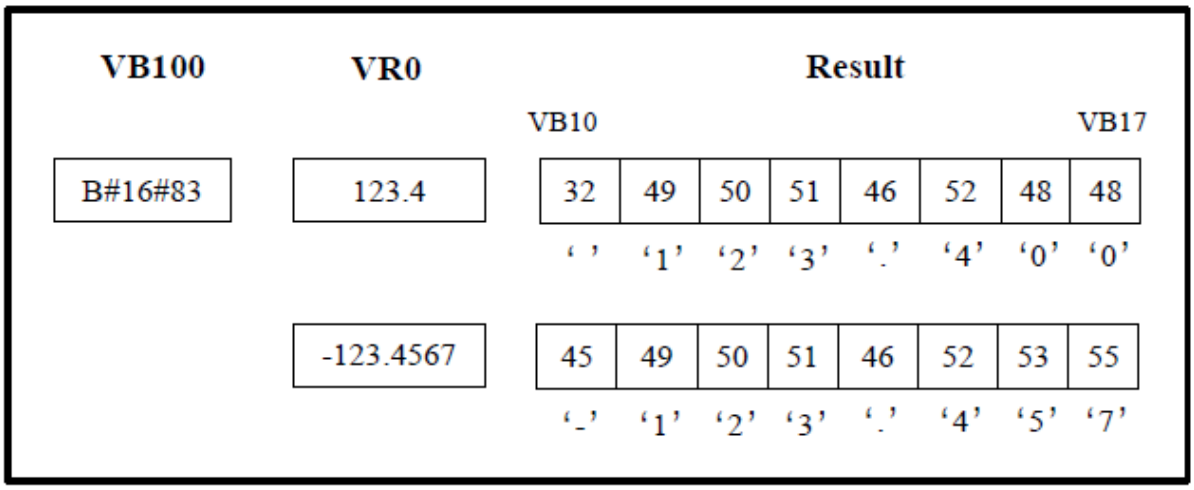
(2) *c* --- This field specifies the separator between the whole number and the fraction:
0 for a decimal point (whose ASCII is 46), and 1 for a comma(whose ASCII is 44).

(3) *ssss* --- This field specifies the size of the buffer.
Its available rang is 3 to 15, and it must be greater than *nnn*.

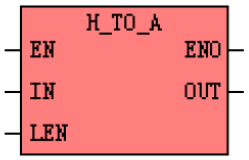
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so the instruction DI_TO_A is always executed: Convert the value of VR0 to a string, and format the string and put it to a buffer beginning with VB10.
IL	LD %SM0.0 R_TO_A %VR0, %VB10, %VB100	

نتیجه به صورت زیر خواهد بود :



8.7.12 H_TO _A (Hexadecimal TO ASCII):

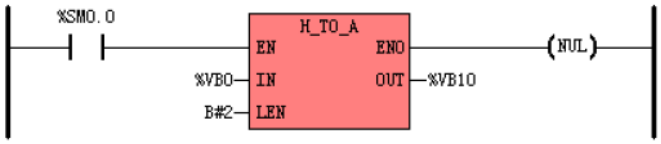
	Name	Usage	Group	
LD	H_TO_A			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	H_TO_A	R_TO_A *IN, OUT, LEN*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE	I, Q, M, V, L, SM
LEN	Input	BYTE	I, Q, M, V, L, SM, Constants
OUT	Output	BYTE	Q, M, V, L, SM

این دستور به تعداد LENبایت از ورودی را که با آدرس مشخص شده در IN آغاز میشود و به صورت Hwxadecimal میباشد به یک رشته کد اسکی تبدیل میکند و نتیجه را در بافر خروجی که با آدرس OUTشروع میشود ذخیره میکند.

نکته : همان طور که میدانید هر ۴ رقم باینری یک رقم هگزادسیمال میباشد .بنابراین هر بایت به صورت ۲ رقم هگزادسیمال خواهد بود .سایز بافر خروجی به صورت LEN *2 بایت میباشد .

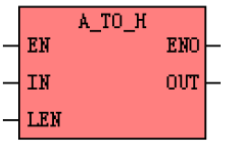
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so H_TO_A is always executed: converts 2-bytes hexadecimal digits, beginning with VB0, to a string and put the result into the buffer which occupies 4 continuous bytes beginning with VB10.
IL	LD %SM0.0 H_TO_A %VB0, %VB10, B#2	

نتیجه به صورت زیر میباشد:

VB0	VB1	Result			
		VB10			VB13
B#16#1A	B#16#2B	49	65	50	66
		'1'	'A'	'2'	'B'
B#16#7C	B#16#8D	55	67	56	68
		'7'	'C'	'8'	'D'

8.7.13: A_TO_H (ASCII TO Hexadecimal):

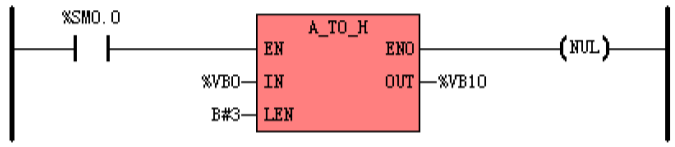
	Name	Usage	Group	
LD	A_TO_H			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	A_TO_H	A_TO_H *IN, OUT, LEN*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE	I, Q, M, V, L, SM
LEN	Input	BYTE	I, Q, M, V, L, SM, Constants
OUT	Output	BYTE	Q, M, V, L, SM

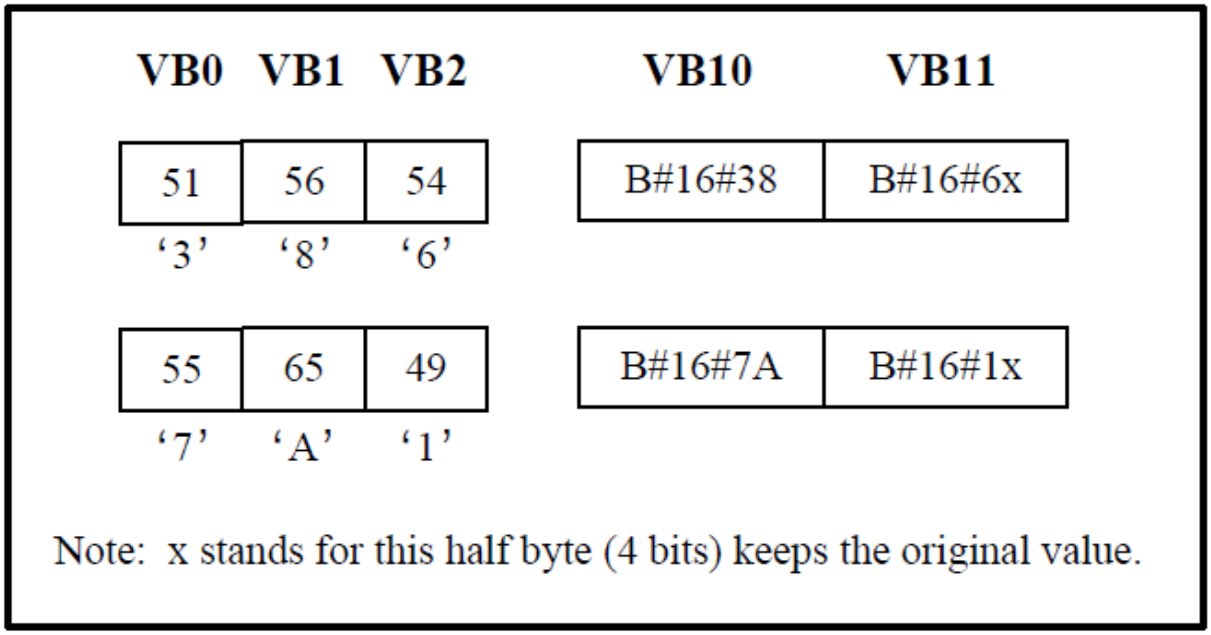
این دستور به تعداد LEN کاراکتر اسکی را که آدرس آن با ورودی INآغاز میشود به رقم های هگزادسیمال تبدیل کرده و نتیجه را دربافر خروجی که با آدرس OUT آغاز میشود ذخیره میکند.

نکته : همان طور که میدانید هر ۴ رقم باینری یک رقم هگزادسیمال میباشد.بنابراین هر بایت ورودی که یک کاراکتر اسکی در آن قرار دارد به اندازه ۴ رقم باینری از فضای حافظه بافر خروجی (نیم بایت) را اشغال میکند .

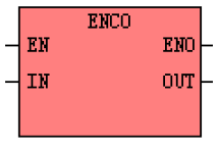
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so A_TO_H is always executed: converts the 3-bytes ASCII string, beginning with VB0, to hexadecimal digits, and put the result into the Output Buffer beginning with VB100.
IL	LD %SM0.0 A_TO_H %VB0, %VB10, B#3	

نتیجه به صورت زیر خواهد بود :



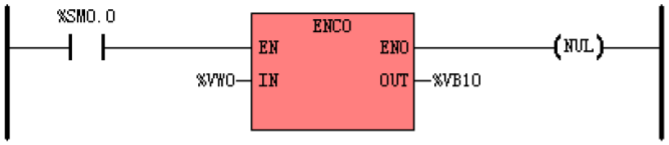
8.7.14 : ENCO (ENCODING):

	Name	Usage	Group	
LD	ENCO			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	ENCO	ENCO *IN, OUT*	U	

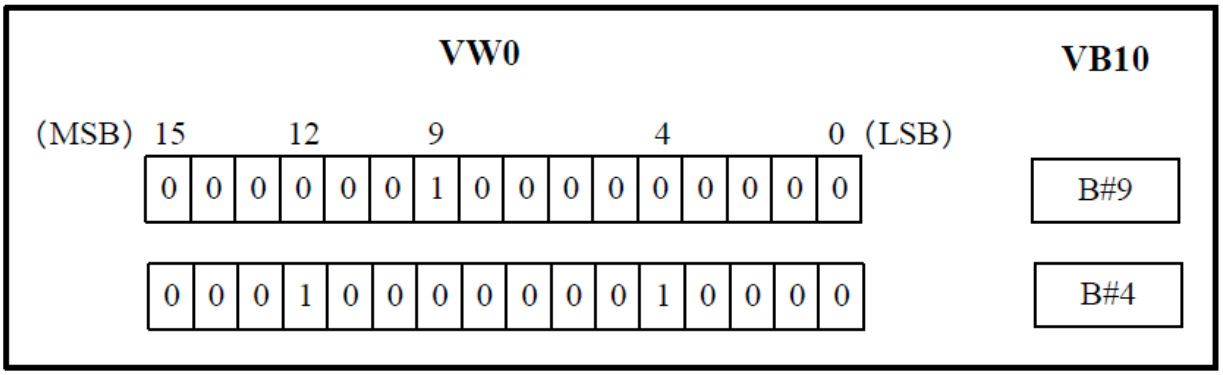
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	WORD	I, Q, M, V, L, SM, Constant
OUT	Output	BYTE	Q, M, V, L, SM

این دستور ورودی که به صورت WORDمیباشد را از پایین ترین بیت چک کرده و شماره اولین بیتی را که مقدار آن یک میباشد در خروجی OUT ذخیره میکند.

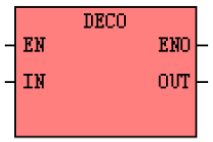
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so ENCO is always executed: writes the bit number of the first bit equal to 1 into VB10.
IL	LD %SM0.0 ENCO %VW0, %VB10	

نتیجه به صورت زیر میباشد:



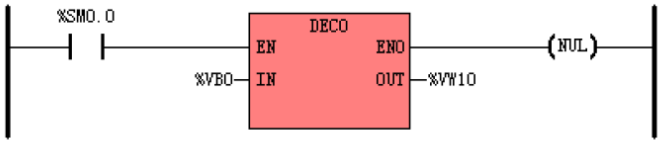
8.7.15: DECO (DECODING)

	Name	Usage	Group	
LD	DECO			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	DECO	DECO *IN*, *OUT*	U	

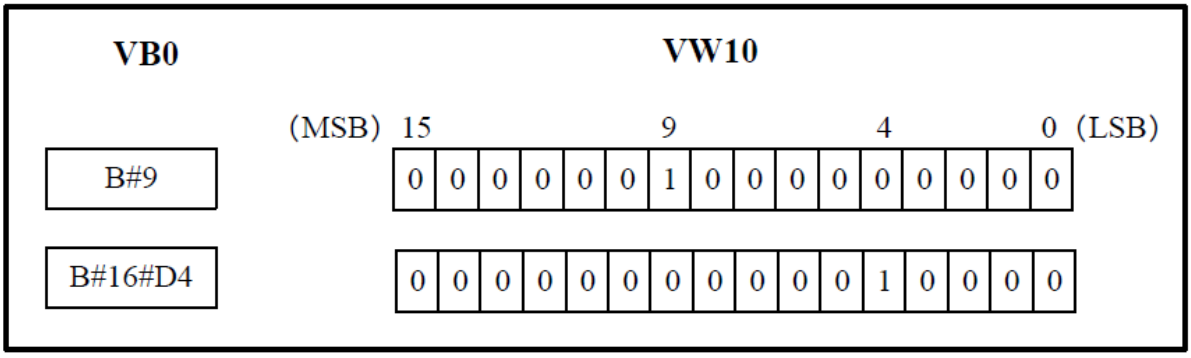
Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE	I, Q, M, V, L, SM, Constant
OUT	Output	WORD	Q, M, V, L, SM

این دستور شماره خانه ای از خروجی را که معادل چهار بیت پایینی از ورودی میباشد یک (SET) میکند. به عبارت دیگر شماره بیتی از خروجی یک میشود که برابر با عددی است که از چهار بیت پایینی ورودی دریافت میشود . در این دستور به غیر از بیت ست شده بقیه بیت ها دارای مقدار صفر میباشند.

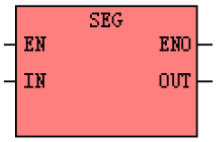
به مثال زیر توجه نمایید:

LD		SM0.0 is always 1, so DECO is always executed: sets the bit in VW10 which corresponds to the bit number represented by the least significant "nibble" of VB0.
IL	LD %SM0.0 DECO %VB0, %VW10	

نتیجه به صورت زیر میباشد :



8.7.16: SEG (7-Segment Display)

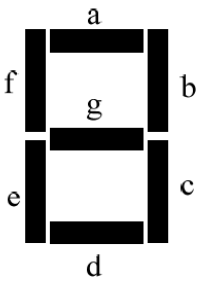
	Name	Usage	Group	
LD	SEG			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SEG	SEG *IN, OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE	I, Q, M, V, L, SM, Constant
OUT	Output	BYTE	Q, M, V, L, SM

این دستوربا توجه به ۴ بیت پایینی ورودی IN یک الگو بیتی برای سون سگمنت ایجاد مینماید و سپس نتیجه را درخروجی OUTقرار میدهد.

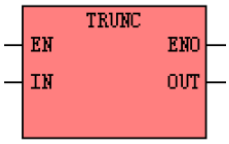
الگوی ذکرشده به شکل زیر میباشد:

IN (LSD)	Display	*OUT* (- g f e d c b a)
0	0	0 0 1 1 1 1 1 1
1	1	0 0 0 0 0 1 1 0
2	2	0 1 0 1 1 0 1 1
3	3	0 1 0 0 1 1 1 1
4	4	0 1 1 0 0 1 1 0
5	5	0 1 1 0 1 1 0 1
6	6	0 1 1 1 1 1 0 1
7	7	0 0 0 0 0 1 1 1



IN (LSD)	Display	*OUT* (- g f e d c b a)
8	8	0 1 1 1 1 1 1 1
9	9	0 1 1 0 0 1 1 1
A	A	0 1 1 1 0 1 1 1
B	B	0 1 1 1 1 1 0 0
C	C	0 0 1 1 1 0 0 1
D	D	0 1 0 1 1 1 1 0
E	E	0 1 1 1 1 0 0 1
F	F	0 1 1 1 0 0 0 1

8.7.17 :TRUNC

	Name	Usage	Group	
LD	TRUNC			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	TRUNC	TRUNC *IN*, *OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, Constant
OUT	Output	DINT	M, V, L, SM

این دستور مقدار REALدر ورودی را به یک مقدار DINT تبدیل کرده و نتیجه را درخروجی ذخیره میکند.

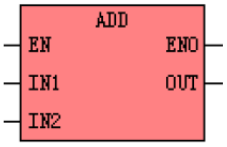
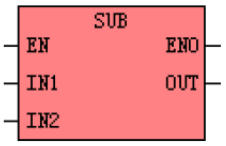
این دستور مقدار اعشار ورودی را حذف میکند.

تفاوت با دستور R – TO DIبررسی شود

8.8: دستورات ریاضی:

این دستورات عملیات ریاضی و محاسباتی انجام میدهند . نکته مهم و حائز اهمیت در این دستورات این است که حتما فرمت داده در ورودی ها و خروجی باید یکسان باشد .

8.8.1: ADD,SUB(جمع و تفریق):

	Name	Usage	Group	
LD	ADD			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	SUB			
IL	ADD	ADD *IN1*, *OUT*	U	
	SUB	SUB *IN1*, *OUT*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, constant
IN2	Input	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, constant
OUT	Output	INT, DINT, REAL	Q, AQ, M, V, L, SM

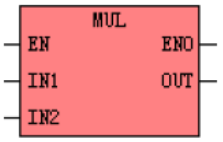
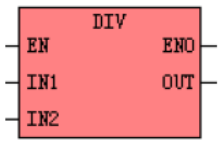
ADD: این دستور ورودی های IN1و IN 2 را با هم جمع کرده و نتیجه را در خروجی OUT ذخیره میکند.

$$OUT=IN1+IN2$$

SUB: این دستور IN2 را از IN1 کم کرده و نتیجه را در خروجی OUT ذخیره مینماید .

$$OUT=IN1-IN2$$

8.8.2: MUL ,DIV(ضرب و تقسیم) :

	Name	Usage	Group	
LD	MUL			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	DIV			
IL	MUL	MUL *IN1*, *OUT*	U	
	DIV	DIV *IN1*, *OUT*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, constant
IN2	Input	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, constant
OUT	Output	INT, DINT, REAL	Q, AQ, M, V, L, SM

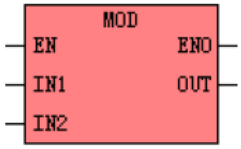
MUL:این دستور IN1و IN2را در هم ضرب کرده و نتیجه را در خروجی OUT ذخیره مینماید.

$$OUT=IN1\times IN2$$

DIV: این دستور IN1را بر IN2 تقسیم کرده و نتیجه را در خروجی OUT ذخیره مینماید.

$$OUT=IN1\div IN2$$

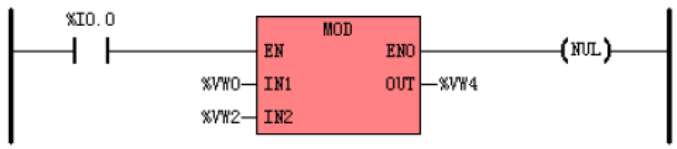
8.8.3 MOD: (باقیمانده تقسیم)

	Name	Usage	Group	
LD	MOD			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	MOD	MOD *IN1*, *OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN1	Input	BYTE, INT, DINT	I, Q, AI, AQ, M, V, L, SM, T, C, HC, constant
IN2	Input	BYTE, INT, DINT	I, Q, AI, AQ, M, V, L, SM, T, C, HC, constant
OUT	Output	BYTE, INT, DINT	Q, AQ, M, V, L, SM

این دستور IN1را بر IN2تقسیم کرده و باقی مانده این تقسیم را در خروجی OUTذخیره میکند.

به مثال زیر توجه کنید :

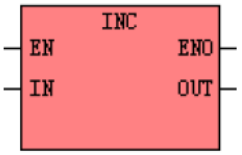
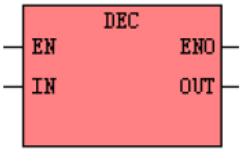
LD		If I0.0 is 0: *MOD* is not executed. If I0.0 is 1: VW0 is divided by VW2, and the remainder is assigned to VW4.
IL	LD %I0.0 (* CR is created with I0.0 *) MOD %VW0, %VW4 (* If CR is 1: VW4 is divided by VW0, and the remainder is still stored in VW4 *) (* If CR is 0: this instruction is not executed *)	

نتیجه به صورت زیر خواهد بود :

VW0	VW2
8	3

VW4
2

: INC ,DEC:8.8.4

	Name	Usage	Group	
LD	INC			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	DEC			
IL	INC	INC *OUT*	U	
	DEC	DEC *OUT*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	BYTE, INT, DINT	I, Q, AI, AQ, M, V, L, SM, T, C, HC, constant
OUT	Output	BYTE, INT, DINT	Q, AQ, M, V, L, SM

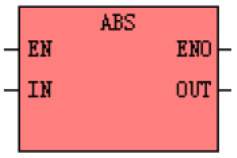
INC: با فعال شدن پایه EN، مقدار ورودی یک واحد اضافه شده و در خروجی ذخیره میگردد.

$$OUT = IN + 1$$

DEC: با فعال شدن پایه EN، مقداروروی یک واحد کاهش یافته و در خروجی ذخیره میگردد.

$$OUT = IN - 1$$

8.8.5: ABS(قدر مطلق) :

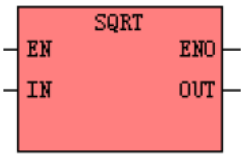
	Name	Usage	Group	
LD	ABS			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	ABS	ABS *IN*, *OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	INT, DINT, REAL	I, Q, V, M, L, SM, T, C, AI, AQ, HC, Constant, Pointer
OUT	Output	INT, DINT, REAL	Q, V, M, L, SM, AQ, Pointer

این دستور قدر مطلق مقدار ورودی را محاسبه کرده و نتیجه را در خروجی ذخیره میکند .

$$OUT = |IN|$$

8.8.6: SQRT(جذر):

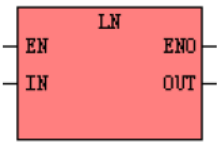
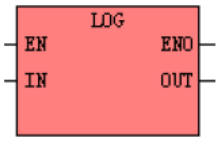
	Name	Usage	Group	
LD	SQRT			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SQRT	SQRT *IN*, *OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, Constant, Pointer
OUT	Output	REAL	V, L, Pointer

این دستور جذر ورودی IN را محاسبه کرده و نتیجه را در خروجی OUT ذخیره میکند.

$$OUT = \sqrt{IN}$$

8.8.7 LOG , LN:

	Name	Usage	Group	
LD	LN			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	LOG			
IL	LN	LN *IN, OUT*	U	
	LOG	LOG *IN, OUT*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, Constant, Pointer
OUT	Output	REAL	V, L, Pointer

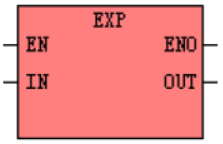
LN: این دستور لگاریتم ورودی INدر در مبنای e (عدد نپر) محاسبه کرده و نتیجه را در خروجی ذخیره مینماید .

$$OUT = \log_e(IN)$$

LOG: این دستور LOG ورودی IN (لگاریتم در مبنای ۱۰) را محاسبه کرده و نتیجه را در خروجی ذخیره میکند.

$$OUT = \log_{10}(IN)$$

8.8.8: EXP (به توان e):

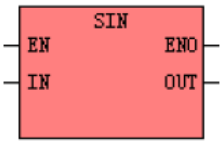
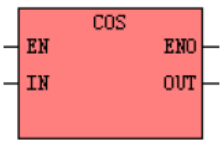
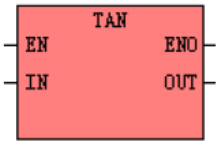
	Name	Usage	Group	
LD	EXP			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	EXP	EXP *IN, OUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, Constant, Pointer
OUT	Output	REAL	V, L, Pointer

این دستور مقدار e به توان IN را محاسبه کرده و نتیجه را در خروجی ذخیره میکند.

$$OUT = e^{IN}$$

8.8.9: SIN ,COS ,TAN :

	Name	Usage	Group	
LD	SIN			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	COS			
	TAN			
IL	SIN	SIN *IN, OUT*	U	
	COS	COS *IN, OUT*		
	TAN	TAN *IN, OUT*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, Constant, Pointer
OUT	Output	REAL	V, L, Pointer

SIN: این دستور مقدار سینوس ورودی IN(را که به صورت رادیان در نظر گرفته شده است) محاسبه کرده و نتیجه را در OUTذخیره میکند.

$$OUT = \text{SIN}\ (IN)$$

COS: این دستور مقدار کسینوس ورودی IN(را که به صورت رادیان در نظر گرفته شده است) محاسبه کرده و نتیجه را در OUTذخیره میکند.

$$OUT = \text{COS}\ (IN)$$

TAN: این دستور مقدار تانژانت ورودی IN(را که به صورت رادیان در نظر گرفته شده است) محاسبه کرده و نتیجه را در OUTذخیره میکند.

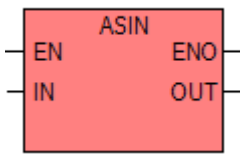
$$OUT = \text{TAN}\ (IN)$$

:ASIN(arc-sine) ,ACOS(arc-cosine),ATAN (arc-tangent):8.8.10

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN	Input	REAL	V, L, Constant, Pointer
OUT	Output	REAL	V, L, Pointer

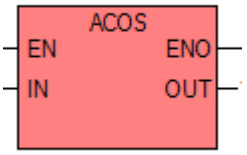
ASIN: این دستور مقدار arc-sine ورودی IN را محاسبه کرده و نتیجه را در خروجی ذخیره میکند.

$$OUT = \text{ARCSIN}\ (IN).$$



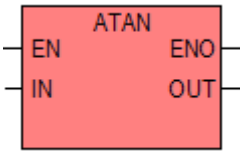
ACOS: این دستور مقدار arc-cosine ورودی IN را محاسبه کرده و نتیجه را در خروجی ذخیره میکند.

$$OUT = \text{ARCCOS}\ (IN).$$



ATAN: این دستور مقدار arc-tangent ورودی IN را محاسبه کرده و نتیجه را در خروجی ذخیره میکند.

$$OUT = \text{ARCTAN}\ (IN)$$



8.9: دستورات کنترلی:

8.9.1: دستورات JMP و LBL(دستورات پرش) :

	Name	Usage	Group	
LD	LBL	lbl —(LBL)—		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	JMP	lbl —(JMP)—		
	JMPC	lbl —(JMPC)—		
	JMPCN	lbl —(JMPCN)—		
IL	LBL	*lbl*:	U	
	JMP	JMP *lbl*		
	JMPC	JMPC *lbl*		
	JMPCN	JMPCN *lbl*		

Operand	Description
lbl	Valid identifier

از این دستورات میتوان برای پرش از بخشی از برنامه به بخشی دیگر استفاده نمود

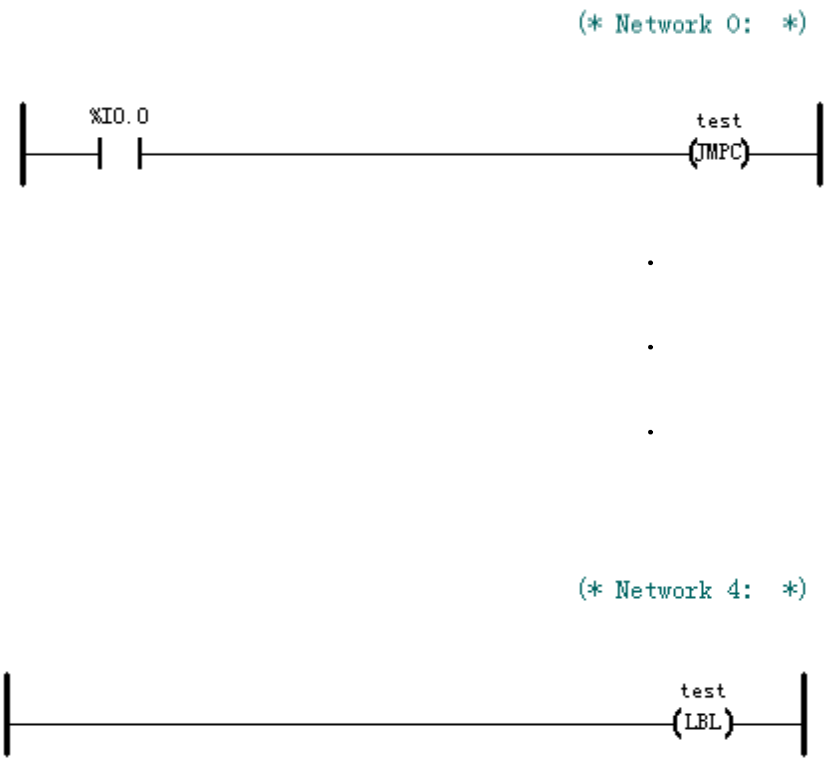
به وسلیه دستور lbl برای یک موقعیت از برنامه یک لیبل تعریف میشود و با استفاده از آن میتوان مشخص نمود که پرش به کدام قسمت از برنامه انجام شود .به عبارت دیگر LBL مشخص کننده مقصد پرش میباشد .

JMP: با استفاده از این دستور اجرای برنامه به صورت اتومات وبدون هیچ قید و شرطی از یک شبکه به شبکه دیگر منتقل میشود .شبکه مقصد توسط LBL مشخص شده است . (به شبکه ای که توسط LBLمشخص شده است پرش میکند)

JMPC: این دستور زمانی اجرای برنامه را از یک شبکه به شبکه مقصد (که توسط LBLمشخص شده است) هدایت میکند که سیگنال ارسالی از سمت چپ برقرار باشد (به صورت TRUE یا یک باشد) . به عبارت دیگر در این دستور پرش با شرط برقرار شدن سیگنال ارسالی از سمت چپ میباشد .

JMPCN:این دستور زمانی اجرای برنامه را از یک شبکه به شبکه مقصد (که توسط LBLمشخص شده است)هدایت میکند که سیگنال ارسالی از سمت چپ برقرار نباشد (به صورت FALSE یا صفر باشد).به عبارت دیگر پرش زمانی صورت میگیرد که بیت مربوط در سطر قبلی صفر باشد.

به عنوان مثال:



8.9.2: دستورات RETURN(دستورات بازگشت):

	Name	Usage	Group	
LD	RETC	—(RETC)—		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	RETCN	—(RETCN)—		
IL	RETC	RETC	U	
	RETCN	RETCN		

:RETC

LD: از این دستور جهت پایان دادن به اجرای زیر برنامه یا روتین وقفه و بازگشت برنامه به محل فراخوانی آنها میباشد. این دستور زمانی اجرا میگردد که سیگنال ارسالی از سمت چپ ۱(TRUE) باشد.

IL: این دستور در زبان IL زمانی اجرا میگردد که CR یک باشد .

:RETCN

LD : از این دستور جهت پایان دادن به اجرای زیر برنامه یا روتین وقفه و بازگشت برنامه به محل فراخوانی آنها میباشد. این دستور زمانی اجرا میگردد که سیگنال ارسالی از سمت چپ ۰(FALSE) باشد.

IL: این دستور در زبان IL زمانی اجرا میگردد که CR صفر باشد .

۸.۹.۳: فراخوانی زیر برنامه:

تعریف زیر برنامه (Subroutine):

	Name	Usage	Group	
LD	CAL	NAME EN ENO IN1 OUT1 IN_OUT1		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	CAL	CAL *NAME*, actual parameter 1, actual parameter 2, …	U	

زیر برنامه بخش کوچکی از برنامه میباشد که بنا به شرایط خاص در قسمت های مختلف یک پروژه فراخوانی میگردد.

لزوم استفاده از زیر برنامه :

در یک پروژه با توجه به حجم بالا ی دستورات و نیز عدم نیاز به اجرای تمامی بخش ها در هر سیکل کاری ، برنامه را به بخش های کوچک تری تقسیم میکنیم . با این کار سرعت اجرای برنامه و انجام پروسه سریع تر میگردد و نیز چنانچه در قسمتی از برنامه مشکلی وجود داشته باشد و یا نیاز به تغییراتی باشد میتوان به راحتی این تغییرات را اعمال نمود .

به دلیل این که تنها قسمتی از برنامه که در هر سیکل کاری اسکن شده و اجرا میگردد main برنامه میباشد بنابراین چنانچه بخواهیم یک زیر برنامه با توجه به شرایط خاصی اجرا شود باید آن را در قسمت mainفراخوانی کنیم . (چنانچه زیر برنامه در قسمت main فراخوانی نشود هرگز اجرا نخواهد شد)

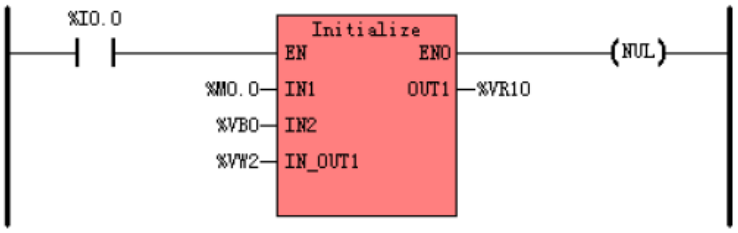
نحوه ایجاد زیر برنامه و نیز تغییر نام دادن آن در بالا توضیح داده شده است .

پس از ایجاد زیر برنامه در قسمت instructions /SBR میتوان زیر برنامه ایجاد شده را مشاهده کرد . حال میتوان در mainبرنامه با توجه به شرط مورد نظر این زیر برنامه را مانند یک دستور فراخوانی کرد .

LD

Main program:

```
(* Network 0 *)
(* call the subroutine 'Initialize' *)
```



The Local Variable Table of the subroutine 'Initialize':

Address	Symbol	Var Type	Data Type	Comment
%L0.0	IN1	VAR_INPUT	BOOL	
%LB16	IN2	VAR_INPUT	BYTE	
%LW22	IN_OUT1	VAR_IN_OUT	INT	
%LD18	OUT1	VAR_OUTPUT	REAL	

IL

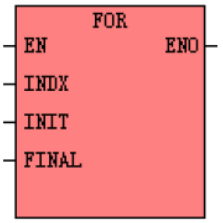
Main Program:

```
(* Network 0 *)
(*call the subroutine 'Initialize'*)
LD      %I0.0
CAL     Initialize, %M0.0, %VB0, %VW2, %VR10
```

The Local Variable Table of the subroutine 'Initialize':

Address	Symbol	Var Type	Data Type	Comment
%L0.0	IN1	VAR_INPUT	BOOL	
%LB16	IN2	VAR_INPUT	BYTE	
%LW22	IN_OUT1	VAR_IN_OUT	INT	
%LD18	OUT1	VAR_OUTPUT	REAL	

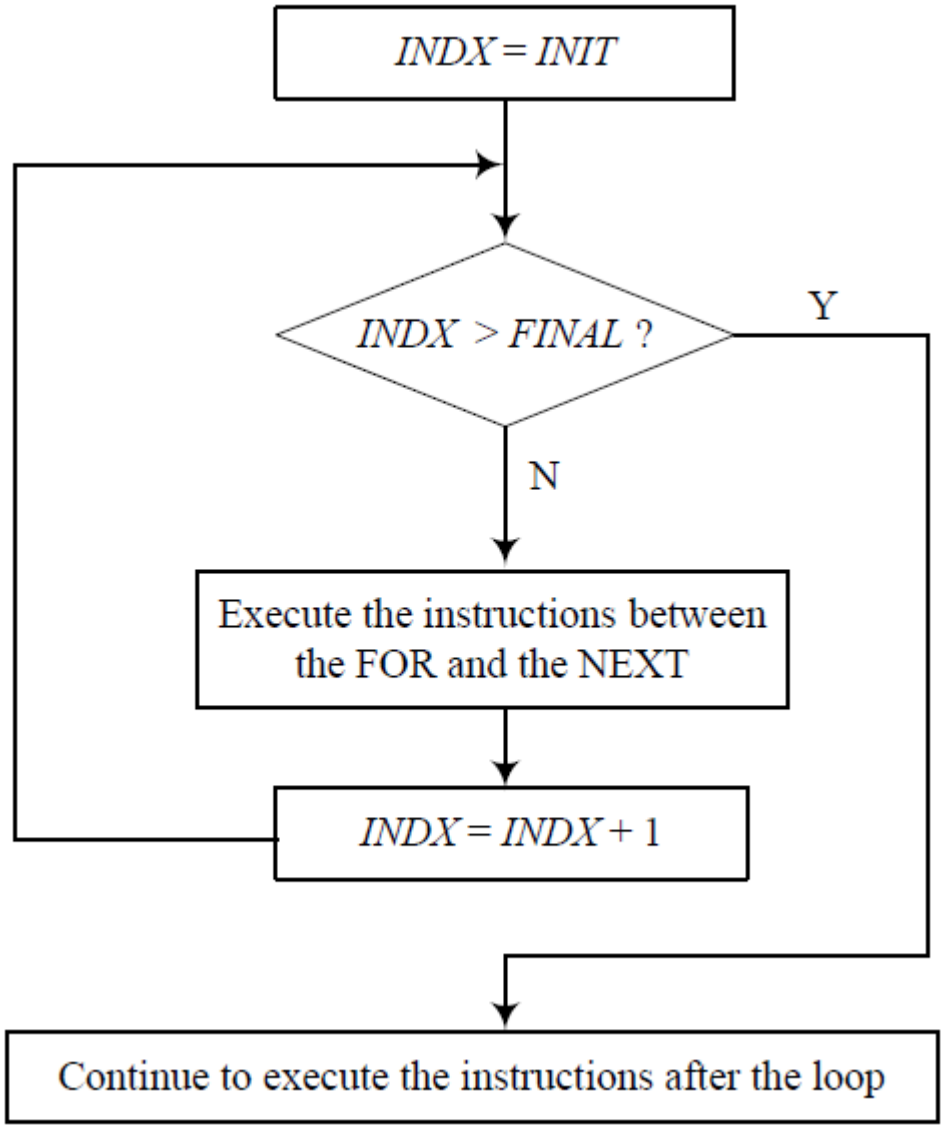
8.9.4: FOR/NEXT

	Name	Usage	Group	
LD	FOR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
	NEXT			
IL	FOR	FOR *INDX, INIT, FINAL*	U	
	NEXT	NEXT		

Operands	Input/Output	Data Type	Acceptable Memory Areas
INDX	Input	INT	M, V, L, SM
INIT	Input	INT	M, V, L, SM, T, C, Constant
FINAL	Output	INT	M, V, L, SM, T, C, Constant

با استفاده از این دستور میتوان یک حلقه را به تعداد مشخص شده ای تکرار نمود. این تعداد در قسمت INDX تعیین میشود. مقدار شروع در INIT و مقدار نهایی در FINAL مشخص میگردد. NEXT نشان دهنده پایان حلقه میباشد . هر دستور FOR نیاز به یک دستور NEXT دارد و این دو دستور حتما باید با یکدیگر به کار رود. چنانچه یک حلقه FOR/NEXT در درون یک حلقه دیگر باشد به آن حلقه تودرتو گفته میشود. میتوان از ۸ حلقه تو در تو استفاده نمود.

روند اجرای برنامه در حلقه FOR/NEXT به صورت زیر میباشد:



زمانی که از دستور FOR/NEXT استفاده میکنید باید موارد زیر در نظر گرفته شود:

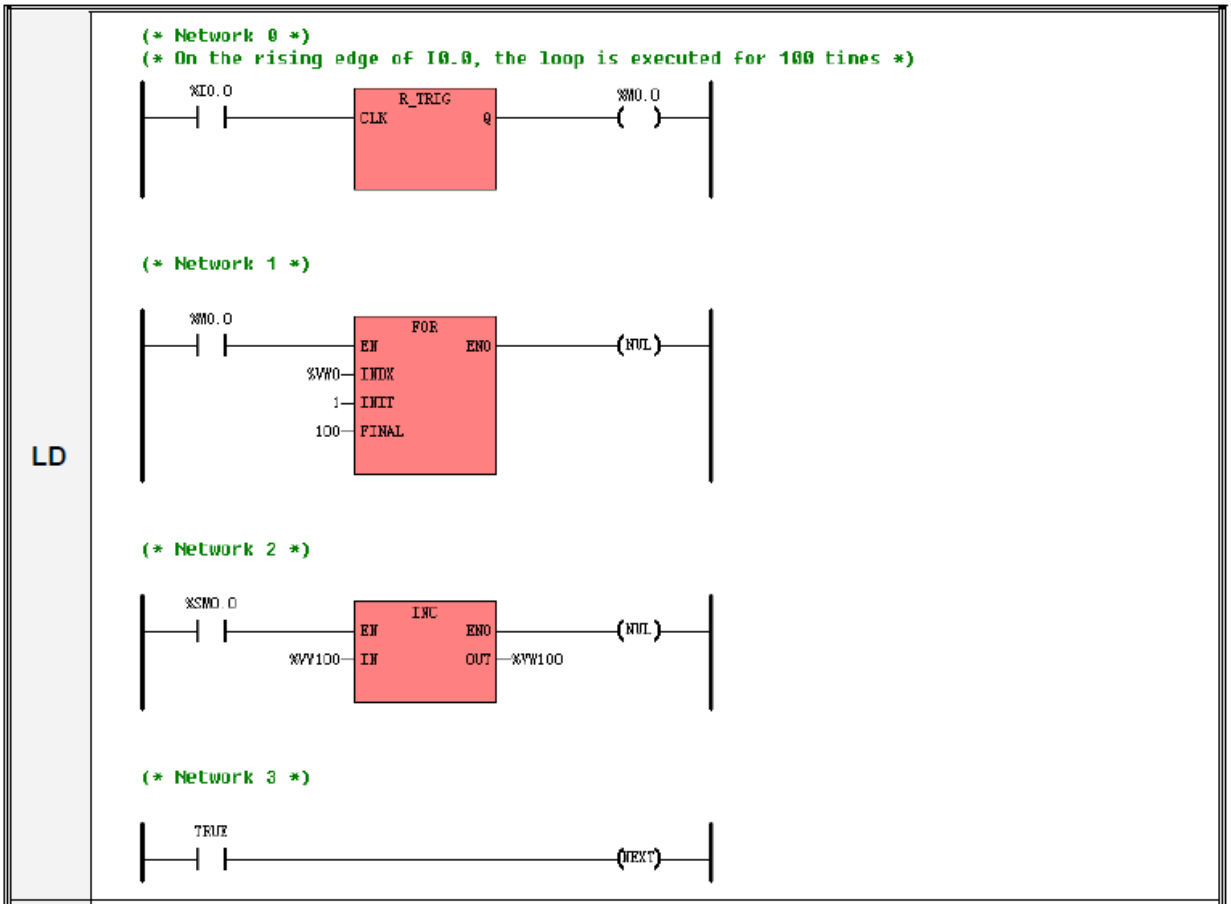
۱)دستور FOR باید دومین دستور در شبکه باشد.

۲) دستور NEXT باید منحصرا در یک شبکه قرار گیرد.

۳) میتوان مقدار FINAL را برای تغییر دادن شرایط اتمام حلقه ازدرون شبکه تغییر داد .

به مثال زیر توجه نمایید:

LD



:END :8.9.5

	Name	Usage	Group	
LD	END	—(END)—		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	END	END	U	

این دستوررا فقط میتوان در mainبرنامه به منظور پایان دادن به سیکل اسکن جاری به کار برد . باید توجه داشت که نرم افزار kincoBuilder در پایان هر سیکل اسکن به صورت اتوماتیک این دستور را فراخوانی میکند.

LD: این دستور زمانی اجرا میشود که سیگنال ارسالی از چپ مقدار ۱ ،(TRUE) باشد .

IL: این دستور زمانی اجرا میشود که مقدار CR یک باشد .

:STOP :8.9.6

	Name	Usage	Group	
LD	STOP	—(STOP)—		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	STOP	STOP	U	

این دستور اجرای برنامه را متوقف کرده و CPUرا از حالت RUNبه حالت STOP میبرد.

LD: این دستور زمانی اجرا میشود که سیگنال ارسالی از چپ مقدار ۱ ،(TRUE) باشد .

IL: این دستور زمانی اجرا میشود که مقدار CR یک باشد .

8.10:وقفه :

هدف استفاده از تکنیک وقفه افزایش بهره وری اجرا در KINCO-K3 میباشد . به عبارت دیگر با استفاده از روتین وقفه میتوان به برخی از واقعه های داخلی و خارجی از پیش تعیین شده به سرعت پاسخ گفت . KINCO-K3 ، ۱۰ واقعه خاص را که هر کدام با یک شماره خاص مشخص شده است ، پشتیبانی میکند.چنانچه بخواهید یک وقفه را فعال نمایید از دستور ATCH به منظور انتصاب واقعه خاص (که با شماره خاص مشخص میشود) به روتین وقفه ای (این روتین با یک نام مشخص میشود) که میخواهیم با رو دادن آن واقعه خاص اجرا شود، استفاده کرد .

میتوان از دستور DTCHجهت غیر فعال کردن روتین وقفه استفاده کرد .این دستور انتساب واقعه به روتین خاص خودش را از بین میبرد .

۸.۱۰.۱:چگونگی عملکرد KINCO-K3 در زمان اجرای روتین وقفه :

روتین وقفه با اتفاق افتادن واقعه مربوطه یک بار اجرا میشود .هنگامی که آخرین دستور از روتین وقفه اجرا شد ، اجرای برنامه به main باز میگردد.

همچنین برای خارج شدن از روتین وقفه و بازگشت به mainبرنامه میتوان از دستورات RETCو RETC نیز استفاده نمود.

8.10.2:اولویت در اجرای وقفه :

واقعه (EVENT) های مختلف برای اجرا دارای سطح اولویت های مختلف میباشند. هنگامی که واقعه های مختلف اتفاق می افتد ، براساس اولویتشان در صف اجرا قرار میگیرند .

چنانچه واقعه های مختلف به صورت همزمان اتفاق بیفتند براساس اولویت اجرا و نیز زمان وقوعشان پاسخ داده میشوند :

هنگامی که این واقعه ها دارای اولویت های یکسانی باشند روتین واقعه ای که زودتر اتفاق افتاده است ، زودتر اجرا میشود .

هنگامی که واقعه هایی که اتفاق می افتد دارای اولویت اجرای متفاوت باشندروتین واقعه با اولویت بالاتر زودتر اجرا میشود.

نکته :

۱)در یک زمان تنها یک روتین وقفه میتواند اجرا شود.

۲)زمانی که یک روتین وقفه در حال اجرا باشد نمیتواند توسط یک روتین وقفه دیگر متوقف شود .هنگامی که در زمان اجرای یک روتین وقفه واقعه دیگری اتفاق بیفتد در صف اجرا قرار میگیرد .

8.10.3 : انواع واقعه هایی (event) که KINCO-K3 پشتیبانی میکند:

وقفه ارتباطی (Communication Port interrupt):

این وقفه دارای بالاترین اولویت میباشد .

این وقفه برای ارتباط free protocol استفاده میشود. وقفه ارسال و دریافت به کاربر این امکان را میدهد که ارتباط را به صورت کامل کنترل نماید . برای شرح بیشتر به بخش دستورات ارسال و دریافت(transmit and receive instructions) مراجعه نمایید .

وقفه I/O :

این وقفه در اولویت متوسط قرار دارد .

این وقفه شامل وقفه های لبه بالارونده /پایین رونده ، HSC و نیز PTO میباشد .

وقفه های لبه بالارونده /پایین رونده میتواند فقط توسط ۴ کانال ورودی (I0.0~I0.3) دریافت شود .به عبارت دیگر چنانچه بخواهیم از وقفه لبه بالارونده /پایین رونده استفاده کنید باید از کانال های I0.0~ I0.3 بر روی CPU استفاده نمایید .هر کدام از این کانال ها میتواند این تغییر وضعیت سیگنال ورودی را دریافت کرده و PLC سریعا به این تغییر وضعیت پاسخ می دهد .

وقفه HSC زمانی رخ میدهد که مقدار در حال شمارش (CV) به مقدار از پیش تعیین شده(PV) برسد . در این حالت یا جهت شمارش تغییر میابد و یا کانتر به صورت خارجی RESET میشود . هر کدام از این وقفه ها این امکان را میدهد که PLC به واقعه High speed counter در لحظه سریعا پاسخ گوید .

وقفه های PTO زمانی که تعداد مشخص شده پالس خروجی کامل شد سریعا رخ میدهد.

نوعی استفاده معمول از این نوع وقفه در کنترل استپر موتور ها (STEPPER MOTOR) میباشد .

وقفه های زمانی :

این نوع از وقفه از پایین ترین اولویت برخوردار است .

این وقفه ها شامل وقفه های زمانی و وقفه های تایمر های T2 وT3 میباشد.

وقفه زمانی به صورت دوره ای و در فواصل زمانی معین (که بر اساس میلی ثانیه) میباشد اتفاق می افتد . این وقفه ها میتوانند برای انجام عملیاتی که باید در فواصل زمانی خاصی تکرار شوند استفاده میگردند.

وقفه تایمر ها زمانی که مقدار جاری (CV) در تایمر های T2و T3 به مقدار از پیش تعیین شده (PV) میرسد، به سرعت اتفاق می افتد . از این وقفه میتوان برای پاسخ زمانی سریع در پایان اتمام یک فاصله زمانی تعیین شده استفاده کرد .

8.10.4 جدول واقعه های وقفه :

همان طور که در بالا اشاره شد هر کدام از واقعه ها دارای یک شماره میباشند . در برنامه برای فعال سازی هر کدام از وقفه ها باید از شماره واقعه مربوط به آن استفاده گردد.

Event No.	Description	Type	Priority
32	PORT 1: XMT complete	Communication Port Interrupts	Highest
31	PORT 1: RCV complete		
30	PORT 0: XMT complete		
29	PORT 0: RCV complete		
28	PTO 0 complete	I/O Interrupts	
27	PTO 1 complete		
26	I0.0, Falling edge		
25	I0.0, Rising edge		
24	I0.1, Falling edge		
23	I0.1, Rising edge		
22	I0.2, Falling edge		
21	I0.2, Rising edge		
20	I0.3, Falling edge		
19	I0.3, Rising edge		
18	HSC0 CV=PV		
17	HSC0 direction changed		
16	HSC0 external reset		
15	HSC1 CV=PV		
14	HSC1 direction changed		
13	HSC1 external reset		
12	HSC2 CV=PV		
11	HSC2 direction changed		
10	HSC2 external reset		
9	HSC3 CV=PV		
8	HSC4 CV=PV		
7	HSC4 direction changed		
6	HSC4 external reset		
5	HSC5 CV=PV		

4	Timed interrupt 1. Its period is specified in SMW24, unit: ms, range: 1~65535ms.	Time Interrupts	
3	Timed interrupt 0. Its period is specified in SMW22, unit: ms, range: 1~65535ms.		
2	Timer T3 ET=PT		
1	Timer T2 ET=PT		Lowest

8.10.5:فعال ساز وقفه(ENI :Enable Interrupt) ، غیر فعال کننده وقفه (Disable Interrupt):

	Name	Usage	Group	
LD	ENI	—(ENI)—		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	DISI	—(DISI)—		
IL	ENI	ENI	U	
	DISI	DISI		

دستور ENI به صورت عمومی پردازش تمامی واقعه های وقفه انتساب یافته (ATTACHشده) را فعال میکند.

دستور DISI به صورت عمومی پردازش تمامی واقعه های وقفه را غیر فعال میکند.

زمانی که CPU در مد RUNقرار دارد وقفه های فعال شده به صورت پیش فرض اجرا میگردد.

LD: این دستور زمانی اجرا میشود که سیگنال ارسالی از سمت چپ ۱ باشد (TRUE) ، در غیر این صورت اجرا نمیشود .

IL:این دستورزما نی اجرا میشود که مقدار CR یک باشد ، در غیر این صورت این دستور اجرا نمیشود .

8.10.6:دستورات ATCH و DTCH:

	Name	Usage	Group	
LD	ATCH	ATCH EN ENO INT EVENT		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	DTCH	DTCH EN ENO EVENT		
IL	ATCH	ATCH *INT, EVENT*	U	
	DTCH	DTCH *EVENT*		

Operands	Input/Output	Data Type	Description
INT	Input		The name of an existing interrupt routine
EVENT	Input	INT	Constant, an interrupt event No.

ATCH: این دستور واقعه وقفه را (که شماره آن در پایه EVENT مشخص شده است)به روتین وقفه ای که نام آن در پایه INT مشخص میشود انتساب داده و واقعه وقفه را فعال میکند.پس از اجرای این دستور در زمان وقوع واقعه ، روتین وقفه به صورت اتوماتیک فراخوانی میشود .

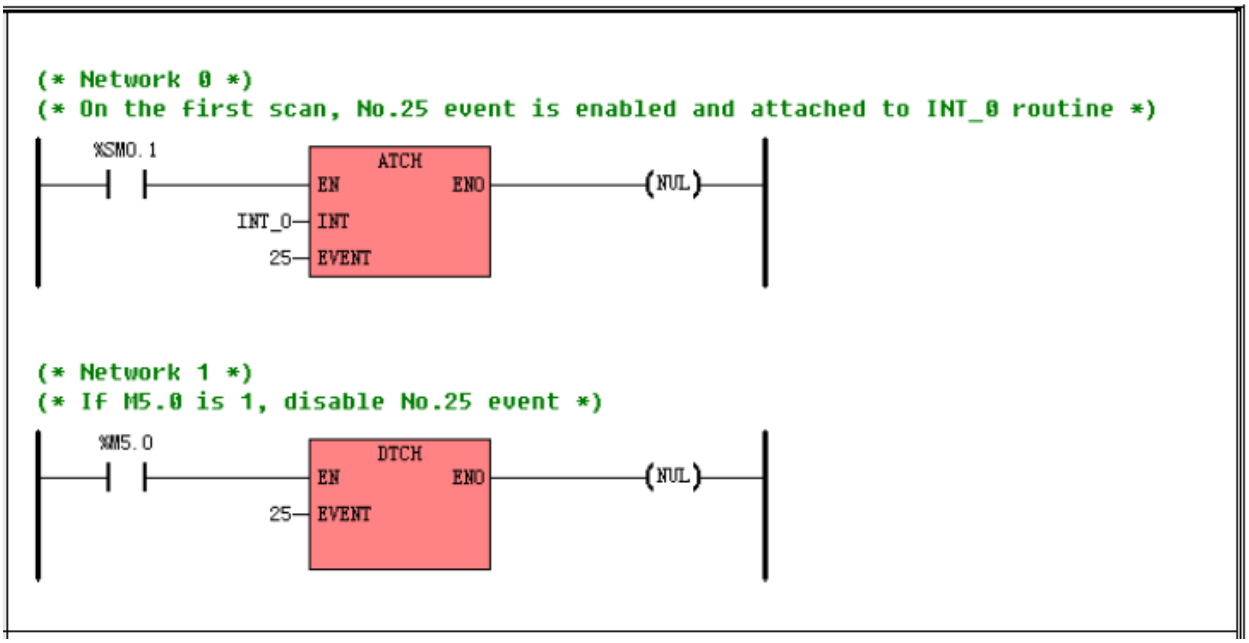
میتوان چندین واقعه وقفه را به یک روتین وقفه انتساب داد اما یک واقعه وقفه را تنها میتوان به یک روتین وقفه نسبت داد و نمیتوان یک واقعه را به چندین روتین وقفه نسبت داد .

DTCH:این دستور اتصال بین واقعه وقفه (که شماره آن در پایه EVENT مشخص میشود) و روتین مربوطه را قطع کرده و واقعه وقفه را غیر فعال میکند .

LD: این دستور چنانچه مقدار EN، یک باشد اجرا میگردد.

IL: این دستور چنانچه مقدار CR، یک باشد اجرا میگردد.

به مثال زیر توجه نمایید :



RTC:8.11 (Real Time Clock):

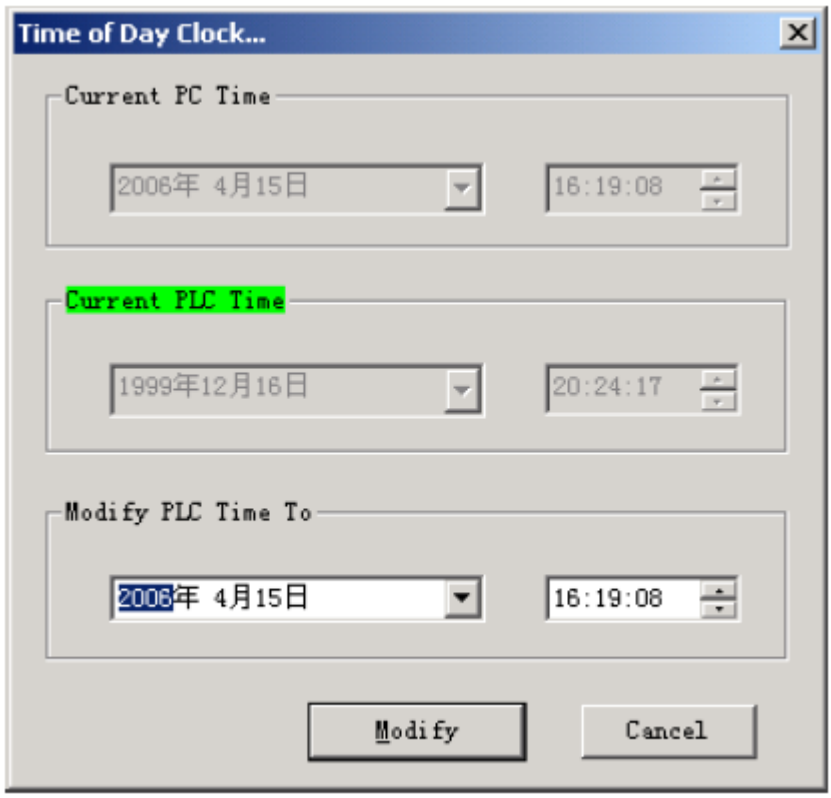
RTC در ماژول PLC یک زمان سنج داخلی به منظور تعیین زمان و تاریخ میباشد .زمان /تاریخ سنج داخلی ، زمان و تاریخ را با فرمت BCD ذخیره کرده و این تنظیمات به صورت اتوماتیک و با استفاده از یک خازن Backup انجام میشود . در دمای کاری معمول ، مدت زمان شارژ این خازن ۷۲ ساعت میباشد .

8.11.1 تنظیم RTC به صورت Online:

کاربر میبایست زمان و تاریخ RTC را با زمان و تاریخ فعلی تنظیم نماید . قبل از انجام این تنظیمات ممکن است مقدار RTC یک مقدار اتفاقی و نادرست باشد .

برای انجام تنظیمات RTC باید به روش زیر عمل نمود :

زمانی که CPU به کامپیوتر وصل میباشد در نرم افزار KincoBuilder از منوی PLC گزینه Time of Day Clock... کلیک کرده ، پنجره مربوط به تنظیمات RTC مانند تصویر زیر نمایش داده میشود :



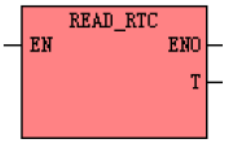
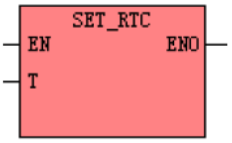
Current PC Time: نشان دهنده زمان و تاریخ فعلی کامپیوتر میباشد .

Current PLC Time: نشان دهنده زمان و تاریخ فعلی RTC، ماژول PLC میباشد که به صورت online به کامپیوتر متصل است .

Modify PLC Time To: میتوان زمان و تاریخ مورد نظر برای RTC را در این قسمت وارد نمود.

Modify: با کلیک بر روی این دکمه تاریخ و زمانی را که وارد نمودید در ماژول PLC نوشته میشود .بنابراین RTC با زمان و تاریخ مورد نظر تنظیم میشود .

8.11.2: READ_RTC ، SET_RTC

	Name	Usage	Group	
LD	READ_RTC			☐ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	SET_RTC			
IL	READ_RTC	READ_RTC *T*	U	
	SET_RTC	SET_RTC *T*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
T	Input (SET_RTC)	BYTE	V
	Output (READ_RTC)		

دستور READ_RTC به منظور خواندن تاریخ و زمان فعلی از RTC استفاده میشود .

این دستور زمان و تاریخ خوانده شده را در یک بافر ۸ بایتی که با آدرس مشخص شده در پایه Tشروع میشود ذخیره مینماید .

دستور SET_RTC به منظور نوشتن تاریخ و زمان که در بافر ۸ بایتی که با آدرس Tشروع میشود ، در RTC استفاده میگردد.

نحوه ذخیره زمان و تاریخ در این بافر ۸ بایتی مطابق با جدول ارائه شده در پایین میباشد :

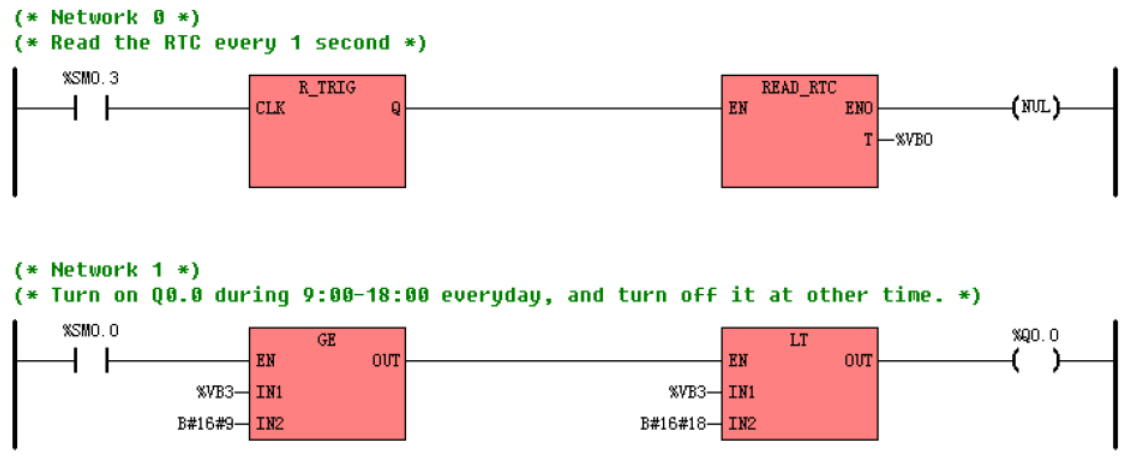
نکته : تمامی مقادیر با فرمت BCDمیباشد .

V Byte	Meaning	Remark
T	Week	Range: 1~7, thereof 1 represents Monday, 7 represents Sunday.
T+1	Second	Range: 0~59
T+2	Minute	Range: 0~59
T+3	Hour	Range: 0~23
T+4	Day	Range: 1~31
T+5	Month	Range: 1~12
T+6	Year	Range: 0~99
T+7	Century	Fixed as 20, BCD coding, hereinafter the same.

به مثال زیر توجه نمایید :

در مثال ارائه شده خروجی Q0.0 هر روز در فاصله زمانی ساعت ۹ تا ۱۸ روشن بوده و بقیه ساعات روز خاموش میبا شد :

LD



IL

```
(* Network 0 *)
(*Read the RTC every 1 second*)
LD          %SM0.3
R_TRIG
READ_RTC %VB0
(* Network 1 *)
(*Turn on Q0.0 during 9:00-18:00 everyday, and turn off it at other time.*)
LD          %SM0.0
GE          %VB3, B#16#9
LT          %VB3, B#16#18
ST          %Q0.0
```

8.12: دستورات ارتباطی :

از این دستورات جهت ارتباط به صورت free protocol استفاده میگردد .مد ارتباطی free protocol به برنامه شما این اجازه را میدهد که به صورت کامل پورت های ارتباطی CPU را کنترل نماید.کاربر میتواند از این مد به منظور اجرای پروتکل های ارتباطی تعریف شده توسط کاربر برای ارتباط با تمامی تجهیزات هوشمند استفاده نماید .این مدهردو پروتکل ASCII و باینری را پشتیبانی میکند. CPUدارای یک و یا دو پورت ارتباطی میباشد . پس از آنکه دستورات ارتباطی اجرا شد مد ارتباط به صورت freeprotocol فعال میگردد.

کاربر میتواند پارامترهای ارتباطی رامانند:,Baudrate,parityو غیره را برای هر پورت در پنجره hardware تنظیم نماید.

8.12.1: دستورات XMT و RCV:

	Name	Usage	Influence	
LD	XMT	XMT EN ENO TBL PORT		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	RCV	RCV EN ENO TBL PORT		
IL	XMT	XMT *TBL, PORT*	U	
	RCV	RCV *TBL, PORT*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
TBL	Input	BYTE	I, Q, M, V, L, SM
PORT	Input	INT	Constant (0 or 1)

XMT: این دستور جهت ارسال د اده های ذخیره شده در بافر از طریق پورت (مشخص شده در پایهPORT) د رمد freeprotocol استفاده میشود.

آدرس شروع بافردیتا در پایه TBL تعیین میگردد.

بایت اول تعیین کننده تعداد بایت های ارسالی میباشد و بایتهای بعدی داده خواهد بود .چنانچه SM87.1=1 باشد زمانی که CPU آخرین کاراکتر بافر دیتا را ارسال نمود ، واقعه وقفه XMT- Complete به صورت اتوماتیک رخ میدهد .(شماره واقعه مربوط به پورت ۰ ، ۳۰ و شماره واقعه مربوط به پورت ۱، ۳۲ میباشد)

چنانچه تعداد بایتهای ارسالی ۰ تنظیم شود ، دستور XMT هیچ عملیاتی انجام نمیدهد ، بنابراین واقعه وقفه نیز رخ نمیدهد.

RCV: این دستور جهت دریافت داده از طریق پورت مشخص شده در پایه PORT در مد ارتباط free protocol استفاده میگردد.داده های دریافت شده در بافر دیتا ذخیره میگردد.بافر دیتا با آدرس TBL شروع شده و اولین بایت این بافر تعیین کننده تعداد بایت های دریافتی میباشد .کاربر باید وضعیت شروع و پایان عملیات دریافت را مشخص کند.

چنانچه SM87.1=1 باشد هنگامی که در CPU عملیات دریافت کامل شد ، به صورت اتوماتیک واقعه وقفه RCV-complete رخ میدهد (شماره واقعه مربوط به پورت ۰، ۲۹ و شماره واقعه مربوط به پورت ۱، ۳۱ میباشد)

LD: اجرای دستورات XMT, RCV بستگی به وضعیت EN دارد.

IL: اجرای دستورات XMT, RCV بستگی به وضعیت CR دارد.این دستورات تاثیری بر مقدار CR نخواهند داشت .

رجیسترهای وضعیت و رجیسترهای کنترلی در فضای حافظه SM برای ارتباط free protocol:

برای ارتباط free protocol در کنار دستورات XMT و RCV ، تعدادی رجیسترهای وضعیت و رجیسترهای کنترلی درفضای حافظه SM وجود دارد .

کاربر میتواند به منظور مشخص کردن وضعیت ارتباط و یا کنترل ارتباط در برنامه این رجیسترها را خوانده و یا مقداری در آنها وارد نماید .

جداول زیر به صورت خلاصه بایتهای وضعیت و WORD های کنترلی را توضیح میدهد:

SMB86 --- Receive Status Register

Bit (read-only)		Status	Description
PORT 0	PORT 1		
SM86.0		1	A parity error is detected, but receive shall not be terminated.
SM86.1		1	Receive was terminated because of receiving the maximum character number. (see SMB94)
SM86.2		1	Receive was terminated because of receiving a character Overtime. (See SMW92)
SM86.3		1	Receive was terminated because of System Overtime.
SM86.4		-	Reserved.
SM86.5		1	Receive was terminated because of receiving the user-defined End character (see SMB89).
SM86.6		1	Receive was terminated because of the errors in the parameters or missing the Start or End condition.
SM86.7		1	Receive was terminated because of the user disable command (See SM87.7)

SMB87 --- Receive Control Register

Bit		Status	Description
PORT 0	PORT 1		
SM87.0		-	Reserved.
SM87.1		0	Disenable XMT-complete and RCV-complete interrupts.
		1	Enable XMT-complete and RCV-complete interrupts.
SM87.2		0	Ignore SMW92.
		1	Terminate receive if the time in SMW92 is exceeded while receiving a character.
SM87.3		-	Reserved.
SM87.4		0	Ignore SMW90.
		1	Turn to effective receive if the time interval in SMW90 is exceeded.
SM87.5		0	Ignore SMB89.
		1	Enable the user-defined End character in SMB89.

SM87.6		0	Ignore SMB88.
		1	Enable the user-defined Start character in SMB88
SM87.7		0	Disenable RCV function. This condition prevails over any other conditions.
		1	Enable RCV function.

Other Control Registers

PORT 0	PORT 1	Description
SMB88		To store the user-defined receive Start character. After executing the *RCV* instruction, the CPU turns into effective receive state when the Start character is received, and the previously received data will be rejected. CPU takes the Start character as the first effective byte received. SM87.6 should be set to be 1 to enable SMB88.
SMB89		To store the user-defined receive End character. The CPU will take this character as the last effective byte received. When the character is received, the CPU will immediately terminate receive disregarding any other End conditions. SM87.5 should be set to be 1 to enable SMB89.
SMW90		To store the user-defined receive Ready time (Range: 1~60,000ms). After executing the *RCV* instruction and passing through this time interval, the CPU will automatically turn into effective receive state disregarding whether the Start character is received or not. Thereafter, the data received shall be effective. SM87.4 should be set to be 1 to enable SMW90.
SMW92		To store the user-defined receiving a character Overtime (Range: 1~60,000ms). After executing the *RCV* instruction and turning into effective receive state, if no character is received within this time interval, the CPU will terminate receive disregarding any other End condition. SM87.2 should be set to be 1 to enable SMW92.

SMW94		To store the maximum number of characters to be received (1~255). The CPU will immediately terminate receive as soon as the maximum effective characters are received disregarding any other End conditions. If this value is set to be 0, the *RCV* instruction will return directly.

در مد ارتباطی free protocol یک زمان اضافی برای دریافت به صورت پیش فرض وجود دارد (۹۰ ثانیه) این زمان مقدار اضافه زمان به صورت زیر عمل میکند:

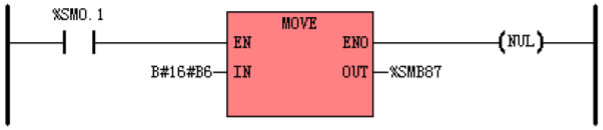
پس از اجرای RCV ، CPU چنانچه در این مدت زمان داده ای دریافت نکند بلافاصله عملیات دریافت را پایان میدهد . به علاوه هنگامی که CPU شروع به دریافت داده های موثر میکند ، ابتدا از مقدار موجود در SMW92 استفاده میکند، چنانچه مقداری در SMW92 نباشد مقدار زمان سیستمی جای گزین این مقدار میشود .

مثالی که درادامه ارائه میشود به منظور نشان دادن عملکرد مد ارتباطی free protocolمیباشد .در این مثال CPU یک رشته کاراکتر دریافت میکند.به عنوان کاراکتر آخر return را دریافت میکند.چنانچه عملیات دریافت به صورت عادی کامل شد، داده های دریافتی ارسال میشود و سپس عملیات دریافت مجدد شروع میشود .و درصورتی که دریافت به صورت غیر عادی کامل شود (به دلایلی مانند خطا در ارتباط ، Time outو....) داده های دریافتی در نظر گرفته نشده و عملیات دریافت دوباره آغاز میشود:

MAIN Program:

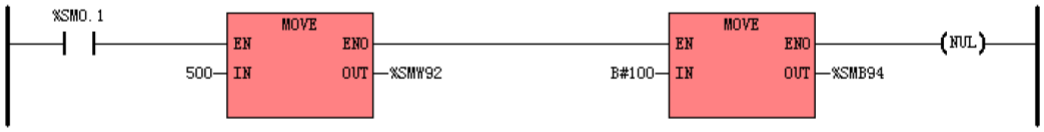
LD

```
(* Network 0 *)
(* The following program is to initialize free-protocol communication.
At first,configure the Start and End conditions of the effective Receive state. *)
```

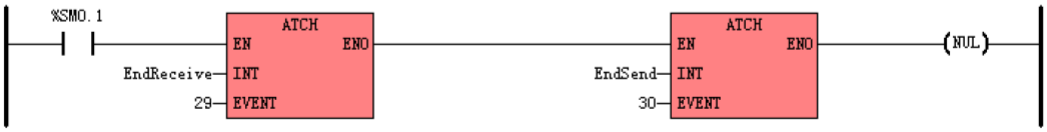



```
(* Network 1 *)
(* The receive Ready time is set to be 10ms,
The receive End character is set to be RETURN character whose ASCII is 13. *)
```

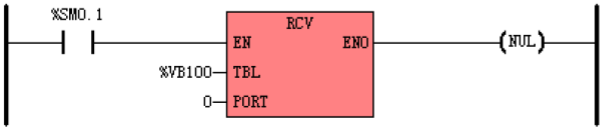

```
(* Network 2 *)
(* The receiving a character Overtime is set to be 500ms,
The maximum number of characters to be received is set to be 100. *)
```




```
(* Network 3 *)
(* Attach the RCV-complete event to the EndReceiver routine,
Attach the XMT-complete event to the EndSendroutine *)
```



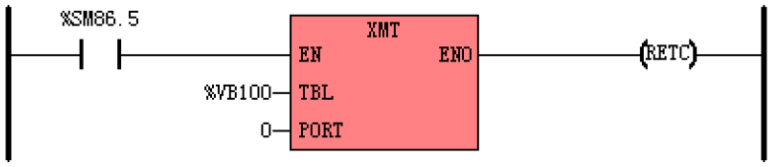

```
(* Network 4 *)
(* Start the Receive task once on the first scan. *)
```



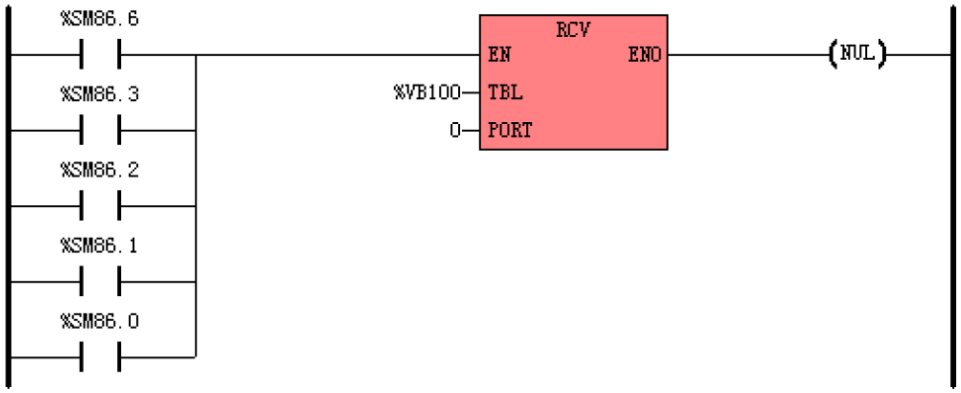
LD

EndReceive (INT00): The RCV-complete interrupt routine

```
(* Network 0 *)
(* If receiving the receive End character,
then transmit bach the data received and return. *)
```

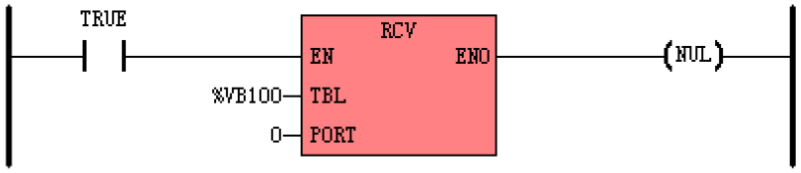



```
(* Network 1 *)
(* if receive is completed abnormally, then restart receive. *)
```



EndSend (INT01): XMT-complete interrupt routine

```
(* Network 0 *)
(* Restart receive after the transmition is completed. *)
```



IL

MAIN Program:

```
(* Network 0 *)
(* The following program is to initialize free-protocol communication. *)
(* At first, configure the Start and End conditions of the effective Receive state. *)
LD      %SM0.1
MOVE    B#16#B6, %SMB87

(* Network 1 *)
(* The receive Ready time is set to be 10ms, *)
(* The receive End character is set to be RETURN character whose ASCII is 13. *)
LD      %SM0.1
MOVE    10, %SMW90
MOVE    B#16#D, %SMB89

(* Network 2 *)
(* The receiving a character Overtime is set to be 500ms, *)
(* The maximum number of characters to be received is set to be 100. *)
LD      %SM0.1
MOVE    500, %SMW92
MOVE    B#100, %SMB94

(* Network 3 *)
(* Attach the RCV-complete event to the EndReceiver routine, *)
(* Attach the XMT-complete event to the EndSendroutine *)
LD      %SM0.1
ATCH    EndReceive, 29
ATCH    EndSend, 30

(* Network 4 *)
(* Start the Receive task once on the first scan. *)
LD      %SM0.1
RCV     %VB100, 0
```

EndReive (INT00): The RCV-complete interrupt routine

```
(* Network 0 *)
(* If receiving the receive End character, then transmit bach the data received and return. *)
LD      %SM86.5
XMT     %VB100, 0
RETC
(* Network 1 *)
(* if receive is completed abnormally, then restart receive. *)
LD      %SM86.6
OR      %SM86.3
OR      %SM86.2
OR      %SM86.1
OR      %SM86.0
RCV     %VB100, 0
```

EndSend (INT01): XMT-complete interrupt routine

```
(* Network 0 *)
(* Restart receive after the transmition is completed. *)
LD      TRUE
RCV     %VB100, 0
```

8.12.2 دستورات Modbus RTU Master:

پروتکل Modbus RTU به صورت بسیار گسترده در صنعت مورد استفاده قرار میگیرد . با استفاده از این دستورات میتوان از Kinco-K3 به عنوان Modbus RTU master در شبکه استفاده نمود.

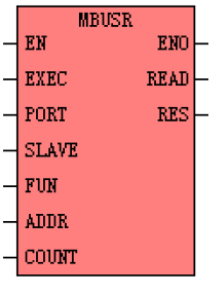
این دستورات تنها توسط پورت RS485(پورت ۱) قابل اجرا میباشند .به عبارت دیگر در این شبکه تنها میتوان از پورت ارتباطی RS485 استفاده نمود.

برای برنامه نویسی Modbus RTU master باید مطابق مراحل زیر عمل نمود:

تنظیم پارامترهای ارتباطی PORT 1 در پنجره Hardware

فراخوانی د ستورات MBUSRو MBUSW در برنامه

8.12.2.1. دستور MBUSR (Modbus RTU master read):

	Name	Usage	Group	
LD	MBUSR			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	MBUSR	MBUSR *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, READ, RES*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
EXEC	Input	BOOL	I, Q, V, M, L, SM, RS, SR
PORT	Input	INT	Constant (1)
SLAVE	Input	BYTE	I, Q, M, V, L, SM, Constant
FUN	Input	INT	Constant (MODBUS function code)
ADDR	Input	INT	I, Q, M, V, L, SM, AI, AQ, Constant
COUNT	Input	INT	I, Q, M, V, L, SM, AI, AQ, Constant
READ	Output	BOOL, WORD, INT	Q, M, V, L, SM, AQ
RES	Output	BYTE	Q, M, V, L, SM

از این د ستور به منظور خواندن داده از slaveهای موجود در سیستم استفاده میگردد.

دراین دستور پارامتر PORTمشخص کننده پورت ارتباطی مورد استفاده میباشد که همیشه به صورت ثابت ۱ میباشد .

پارامتر SLAVE بیان گر آدرس slave مقصد است که میتواند در رنج 1~31 باشد .

پارامتر FUN بیان گر Function code مورد استفاده میباشد .

Function codeهایی که در این دستور از آن ها استفاده میشود به صورت زیر میبا شد:

فانکشن کد ۱ : خواندن وضعیت خروجی های دیجیتال(DO)

فانکشن کد ۲:خواندن وضعیت ورودی های دیجیتال(DI)

فانکشن کد ۳:خواندن وضعیت خروجی های آنالوگی (AO)

فانکشن کد۴:خواندن وضعیت ورودی های آنالوگی (AI)

پارامتر ADDR بیانگر آدرس شروع رجیستر Modbus میباشد که باید خوانده شود.

COUNTبیانگر تعداد رجیسترهایی است که خوانده میشود (ماکزیمم ۳۲)

لبه بالارونده در پایه EXEC برای شروع و آغاز ارتباط استفاده میشود .

زمانی که دستور MBUSRاجرا شد برای یک بار در لبه بالارونده EXEC ارتباط برقرار میشود. پیام Modbus RTU بر اساس پارامترهای SLAVE,COUNT, ADDR,FUN ایجاد شده و سپس ارسال میگرددو CPU منتظر پاسخ از slave میماند.هنگامی که پیام پاسخ دریافت شد بررسی CRC، شماره slaveو function code تعیین میکند که آیا پیام درست بوده است یا نه .چنانچه پیام درست باشد اطلاعات و دیتای درست در بافری که با آدرس پایه READ آغاز میگردد، نوشته میشود.

در غیر این صورت پیام دریافتی اصطلاحا دور ریخته میشود .

پارامتر READ در این دستور مشخص کننده آدرس شروع بافری میباشد که اطلاعات دریافتی در آن ذخیره میگردد.

باید توجه شود که نوع داده REAd باید متناسب با نوع داده function code باشد . به عبارت دیگر چنانچه function code، ۱ و یا ۲ باشد Read نیز باید به صورت یک حافظه با نوع داده Boolدر نظر گرفته شود و چنانچه function code ، ۳ و یا ۴ باشد Read باید به صورت یک حافظه با نوع داده INT یا WORD در نظر گرفته شود.

ذخیره سازی وضعیت ارتباطی و اطلاعات نادرست در اجرای فعلی در حافظه ای که در پایه RES تعیین میگردد، انجام میشود .بنابراین این حافظه فقط قابل خواندن میباشد(یک حافظه فقط خواندنی)

بیت های مربوط به حافظه RES در تصویر پایین توضیح داده شده است :

MSB LSB

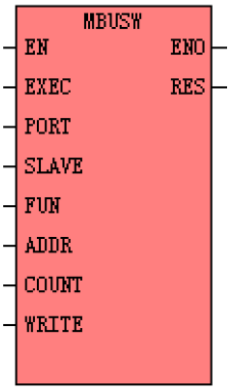
7	6	5	4	3	2	1	0

Bit 7 --- Indicates whether the communication has been finished or not: 0 = not finished, 1 = finished.
Bit 6 --- Reserved。
Bit 5 --- Illegal *SLAVE*.
Bit 4 --- Illegal *COUNT*.
Bit 3 --- Illegal *ADDR*.
Bit 2 --- 1 = The specified port is busy.
Bit 1 --- 1 = Time out
Bit 0 --- 1 = The received message is wrong because of CRC error, frame error, etc.

LD: در صورتی که EN، ابشد این دستور اجرا میشود .

IL:در صورتی که CR، ابشد این دستور اجرا میشود . این دستور تاثیری بر مقدار CR نخواهد داشت .

8.12.2.2 دستور MBUSW (Modbus RTU master Write):

	Name	Usage	Group	
LD	MBUSW			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	MBUSW	MBUSW *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, READ, RES*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
EXEC	Input	BOOL	I, Q, V, M, L, SM, RS, SR
PORT	Input	INT	Constant (1)
SLAVE	Input	BYTE	I, Q, M, V, L, SM, Constant
FUN	Input	INT	Constant (MODBUS function code)
ADDR	Input	INT	I, Q, M, V, L, SM, AI, AQ, Constant
COUNT	Input	INT	I, Q, M, V, L, SM, AI, AQ, Constant
WRITE	Input	BOOL, WORD, INT	I, Q, RS, SR, V, M, L, SM, T, C, AI, AQ
RES	Output	BYTE	Q, M, V, L, SM

از این دستور برای نوشتن اطلاعات برروی slaveها استفاده میگردد.

پارامتر PORT مشخص کننده پورت ارتباطی مورد استفاده میباشد که همیشه به صورت ثابت ۱ خواهد بود .

پارامتر SLAVE مشخص کننده آدرس SLAVE های مورد نظر میباشد که در رنج 1~31 میباشد.

پارامتر FUN مشخص کننده Function code مورد استفاده میباشد .

Function codeهایی که در این دستور از آن ها استفاده میشود به صورت زیر میبا شد:

فانکشن کد ۵ : برای نوشتن بر روی خروجی های دیجیتال(DO)

فانکشن کد ۶:برای نوشتن بر روی خروجی های آنالوگی (AO)

فانکشن کد ۱۵:برای نوشتن بر روی تعدادی خروجی دیجیتال (DOs)

فانکشن کد۱۶: برای نوشتن بر روی تعدادی خروجی آنالوگ (AOs)

پارامتر ADDRمشخص کننده آدرس شروع رجیستر Modbusمیباشد که باید نوشته شود.

COUNT مشخص کننده تعداد رجیستر هاست (ماکزیمم ۳۲)

WRITEمشخص کننده آدرس شروع بافری است که داده هایی که در slave نوشته میشود را ذخیره میکند.

باید توجه شود که نوع داده WRITE باید متناسب با نوع داده function code باشد . به عبارت دیگر چنانچه function code، 5 و یا 15 باشد WRITE نیز باید به صورت یک حافظه با نوع داده Boolدر نظر گرفته شود و چنانچه function code ، 6 و یا 16 باشد WRITE باید به صورت یک حافظه با نوع داده INT یا WORD در نظر گرفته شود.

لبه بالارونده EXEC به منظور شروع ارتباط استفاده میگردد. هنگامی که دستور MBUSW اجرا شد ارتباط برای یک بار با لبه بالارونده در EXECبرقرار میشود . پیام Modbus RTU بر اساس پارامترهای WRITE، COUNT، ADDR ، FUN و SLAVE ایجاد میشود ،سپس این پیام ارسال میگردد و CPU منتظر پاسخ از slaveمیماند.

هنگامی که پیام slave دریافت شد CRC، شماره slave و function code به منظور بررسی این که آیا پیام درست میباشد یا نه چک میگردد .

ذخیره سازی وضعیت ارتباطی و اطلاعات نادرست در اجرای فعلی در حافظه ای که در پایه RES تعیین میگردد، انجام میشود .بنابراین این حافظه فقط قابل خواندن میباشد(یک حافظه فقط خواندنی)

بیت های مربوط به حافظه RES در تصویر پایین توضیح داده شده است :

MSB LSB

7	6	5	4	3	2	1	0

Bit 7 --- Indicates whether the communication has been finished or not: 0 = not finished, 1 = finished.
Bit 6 --- Reserved。
Bit 5 --- Illegal *SLAVE*.
Bit 4 --- Illegal *COUNT*.
Bit 3 --- Illegal *ADDR*.
Bit 2 --- 1 = The specified port is busy.
Bit 1 --- 1 = Time out
Bit 0 --- 1 = The received message is wrong because of CRC error, frame error, etc.

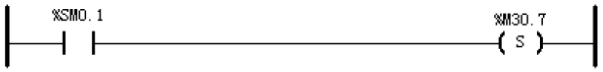
LD: در صورتی که EN، اباشد این دستور اجرا میشود .

IL:در صورتی که CR، اباشد این دستور اجرا میشود . این دستور تاثیری بر مقدار CR نخواهد داشت .

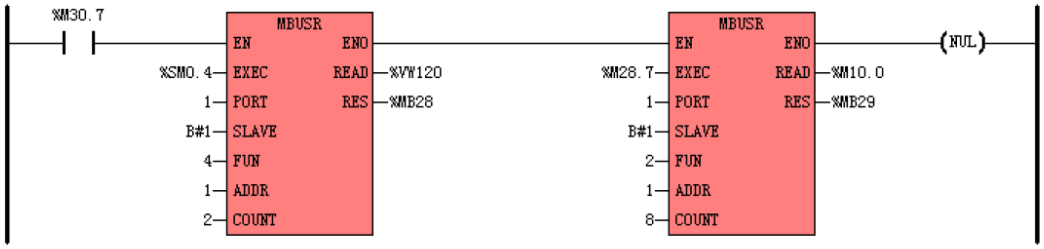
مثال:

LD

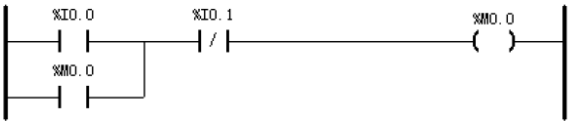
```
(* Network 0 *)
(* M30.7 indicates whether the MBUSW has finished communicating or not *)
```



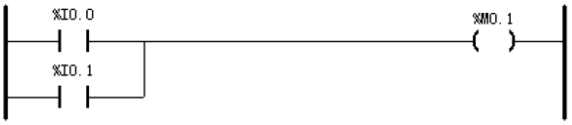

```
(* Network 1 *)
(* If PORT1 is free currently, then MBUSR will be executed:
Every 2 seconds, reads data from slave 1.
Firstly, reads No.1 and No.2 AI registers, then reads No.1-No.8 DI registers. *)
```



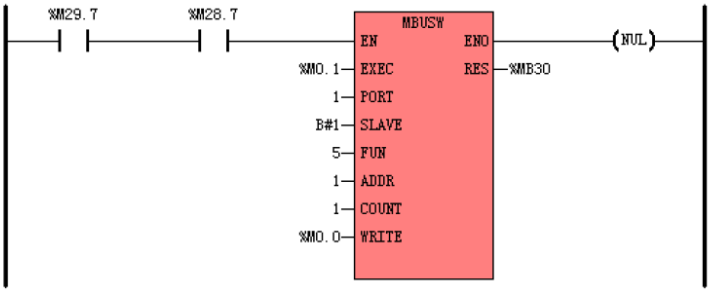

```
(* Network 2 *)
```




```
(* Network 3 *)
```




```
(* Network 4 *)
(* If PORT1 is free currently, then MBUSW will be executed:
Once I0.0 or I0.1 is on, then immediately writes the value of M0.0
into No.1 DO register of the slave 1. *)
```



IL	(* Network 0 *) (* M30.7 indicates whether the MBUSW has finished communicating or not*) LD %SM0.1 S %M30.7 (* Network 1 *) (* If PORT1 is free currently, then MBUSR will be executed: *) (* Every 2 seconds, reads data from slave 1. *) (* Firstly, reads No.1 and No.2 AI registers, then reads No.1-No.8 DI registers.*) LD %M30.7 MBUSR %SM0.4, 1, B#1, 4, 1, 2, %VW120, %MB28 MBUSR %M28.7, 1, B#1, 2, 1, 8, %M10.0, %MB29 (* Network 2 *) LD %I0.0 OR %M0.0 ANDN %I0.1 ST %M0.0 (* Network 3 *) LD %I0.0 OR %I0.1 ST %M0.1 (* Network 4 *) (* If PORT1 is free currently, then MBUSW will be executed: *) (* Once I0.0 or I0.1 is on, then immediately writes the value of M0.0 *) (* into No.1 DO register of the slave 1.*) LD %M29.7 AND %M28.7 MBUSW %M0.1, 1, B#1, 5, 1, 1, %M0.0, %MB30

8.13 شمارنده (COUNTER):

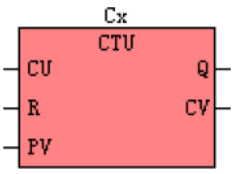
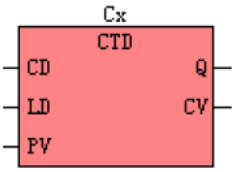
8.13.1 شمارنده رو به بالا(CTU) ،شمارنده رو به پایین (CTD):

شمارنده ها یکی از FUNCTION BLOCK های تعریف شده در استاندارد IEC61131-3 میباشد که بر سه نوع هستند :

CTU: UP COUNTER

CTD: DOWN COUNTER

CTUD: UP – DOWN COUNTER

	Name	Usage	Group	
LD	CTU			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	CTD			
IL	CTU	CTU *Cx, R, PV*	P	
	CTD	CTD *Cx, LD, PV*		

Operands	Input/Output	Data Type	Acceptable Memory Areas
Cx	-	Counter instance	C
CU	Input	BOOL	Power flow
R	Input	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
CD	Input	BOOL	Power flow
LD	Input	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
PV	Input	INT	I, Q, M, V, L, SM, AI, AQ, constant
Q	Output	BOOL	Power flow
CV	Output	INT	Q, M, V, L, SM, AQ

:CTU

LD: شمارنده CTU با هر لبه بالارونده در ورودی CU یک واحد رو به بالا میشمرد و مقدار شمارش شده را در CV (که مقدار اولیه آن ۰ بوده است) ذخیره میکند. هنگامی که مقدار CV مساوی و بزرگتراز PV(که قبلا توسط کاربر مشخص شده است) شد خروجی Q و نیز بیت وضعیت Cx یک خواهد شد. شمارنده همچنان به شمارش خود (با هر لبه بالارونده در ورودی) ادامه میدهد و خروجی همچنان ۱ باقی میماندتا زمانی که پایه R یک شو د .در این صورت CV مجددا صفر شده (خروجی Q هم صفر میشود) و شمارش از ابتدا آغاز میشود.

پایه CU: این پایه ورودی مربوط به پالسی میباشد که باید شمارش شود.

R: پایه RESETمیباشد(با یک شدن این پایه مقدار در حال شمارش شمارنده(CV) مجددا ۰ میشود.)

PV: PRESET VALUE(این مقدار توسط کاربر مشخص و تعیین میشود)

CV: CURRENT VALUE(شمارنده مقدار در حال شمارش خودرا در این رجیستر ذخیره مینماید،مقدار آن در ابتدای اجرا ۰ میباشد)

Q: خروجی شمارنده میباشد.

IL: شمارنده CTU با هر لبه بالارونده در CR یک واحد رو به بالا میشمرد و مقدار شمارش شده را در CV (که مقدار اولیه آن ۰ بوده است) ذخیره میکند. هنگامی که مقدار CV مساوی و بزرگتراز PV(که قبلا توسط کاربر مشخص شده است) شد خروجی Q و نیز بیت وضعیت Cx یک خواهد شد. شمارنده همچنان به شمارش خود (با هر لبه بالارونده در ورودی) ادامه میدهد و خروجی همچنان ۱ باقی میماندتا زمانی که پایه R یک شو د .در این صورت CV مجددا صفر شده (خروجی Q هم صفر میشود) و شمارش از ابتدا آغاز میشود.پس از هر سیکل اسکن مقدار CR برابر با مقدار بیت وضعیت Cx میباشد .

:CTD

LD: شمارنده CTD با هر لبه پایین رونده در ورودی CD یک واحد رو به پایین میشمرد و یک واحد ازمقدار CV(که در ابتدا برابر با مقدار PVمیباشد) کم میکند.(در این شمارنده زمانی که CPUروشن میشود ، مقدار PV در CV ذخیره میگردد) .زمانی که مقدار CVبرابر با صفر شد ، خروجی Q و Cx یک خواهد شد. این مقدار صفر میماند تا زمانی که پایه LD یک شود .زمانی که این پایه یک شد مجددا مقدار PVدرCV بارگذاری میشود. (هرزمان که پایه LDفعال شود صرف نظر از این که CV به چه مقدار عددی رسیده است ، مقدار PV درCV بارگذاری میشود)

پایه CD: این پایه ورودی مربوط به پالسی میباشد که باید شمارش شود.

LD: پایه LOAD INPUT میباشد که عملکردی مانند پایه RESET در شمارنده CTUدارد (با یک شدن این پایه مقدار در حال شمارش شمارنده(CV) مجددا با مقدار PVبرابر میشود.)

PV: PRESET VALUE(این مقدار توسط کاربر مشخص و تعیین میشود)

CV: CURRENT VALUE(شمارنده مقدار در حال شمارش خودرا در این رجیستر ذخیره مینماید،مقدار آن در ابتدای اجرا ۰ میباشد)

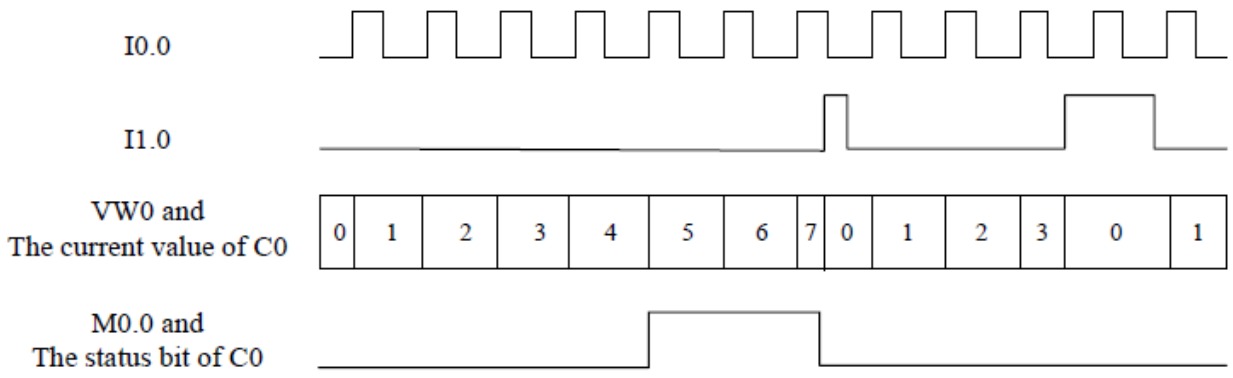
Q: خروجی شمارنده میباشد.

IL: شمارنده CTD با هر لبه پایین رونده CR یک واحد رو به پایین میشمرد و یک واحد ازمقدار CV(که در ابتدا برابر با مقدار PVمیباشد) کم میکند.(در این شمارنده زمانی که CPUروشن میشود ، مقدار PV در CV ذخیره میگردد) .زمانی که مقدار CVبرابر با صفر شد ، خروجی Q و Cx یک خواهد شد. این مقدار صفر میماند تا زمانی که پایه LD یک شود .زمانی که این پایه یک شد مجددا مقدار PVدرCV بارگذاری میشود. (هرزمان که پایه LDفعال

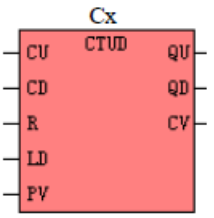
شود صرف نظر از این که CV به چه مقدار عددی رسیده است ، مقدار PV درCV بارگذاری میشود). .پس از هر سیکل اسکن مقدار CRبرابر با مقدار بیت وضعیت Cx میباشد .

به مثال زیر توجه نمایید :

LD	IL
(* Network 0: *) %I0.0 — C0 CTU (CU, R %I1.0, PV 5, Q, CV %VW0) — %M0.0 (* Network 1: *) %I0.1 — C1 CTD (CD, LD %I1.1, PV 5, Q, CV %VW2) — %M0.1	(* NETWORK 0 *) LD %I0.0 CTU C0, %I1.0, 5 ST %M0.0 (* NETWORK 1 *) LD %I0.1 CTD C1, %I1.1, 5 ST %M0.1



CTUD:8.13.2 (UP-DOWN COUNTER):

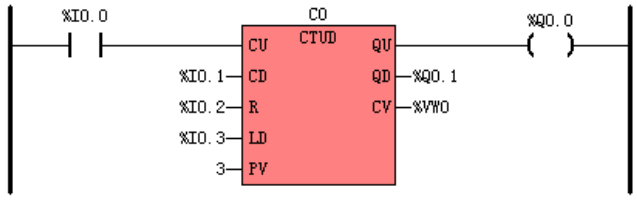
	Name	Usage	Group	
LD	CTUD			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	CTUD	CTUD *Cx, CD, R, LD, PV, QD*	P	

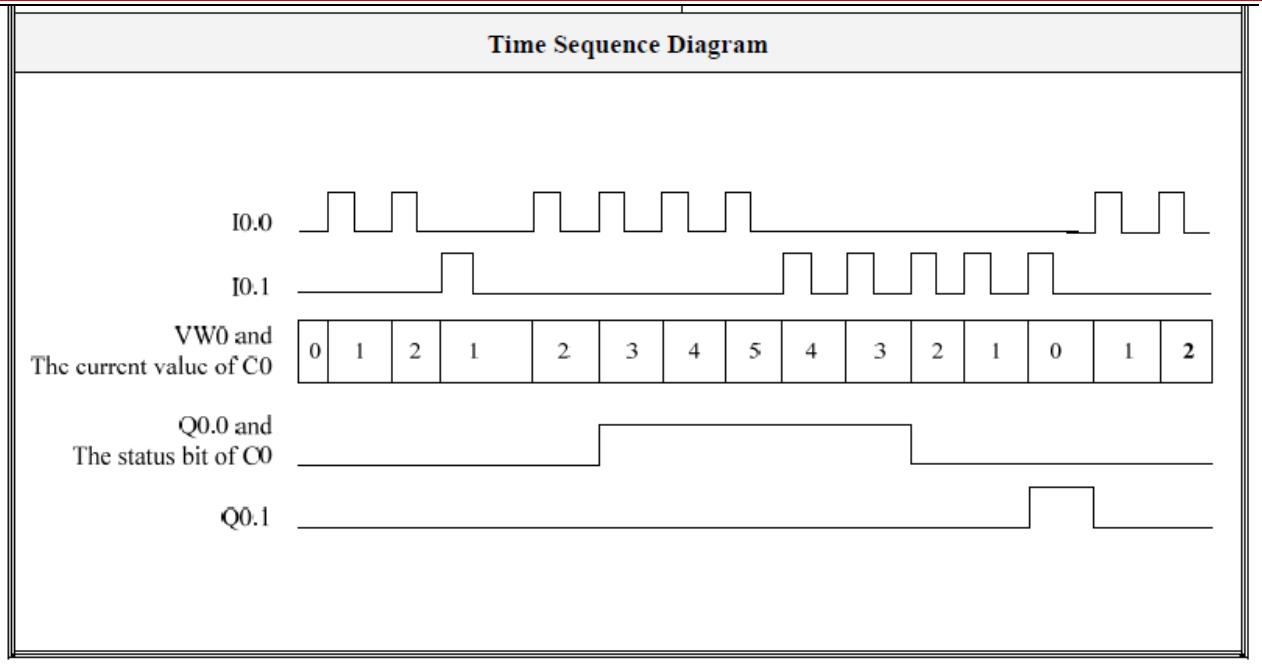
Operands	Input/Output	Data Type	Acceptable Memory Areas
Cx	-	Counter instance	C
CU	Input	BOOL	Power flow
CD	Input	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
R	Input	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
LD	Input	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
PV	Input	INT	I, Q, M, V, L, SM, AI, AQ, constant
QU	Output	BOOL	Power flow
QD	Output	BOOL	Q, M, V, L, SM
CV	Output	INT	Q, M, V, L, SM, AQ

این شمارنده عملکردی ترکیبی از شمارنده رو به بالا و شمارنده رو به پایین دارد.

این شمارنده بالبه بالارونده در پایه CU یک واحد رو به بالا و با لبه بالا رونده در پایه CD یک واحد رو به پایین میشمرد. مقدار در حال شمارش این شمارنده نیز در ذخیره میگردد. زمانی که مقدارV C مساوی یا بزرگتر از CVشود خروجی QU و نیز بیت وضعیت Cx یک خواهد شد و زمانی که CV برابر با صفر شد مقدار QD برابر یک خواهد شد.هنگامی که پایه R فعال شود مقدار صفر را در CV و زمانی که LD فعال شود مقدار PV در آن بارگذاری خواهد شد . زمانی که این دو پایه همزمان یک شوند اولویت با پایه Rبوده و مقدار CV صفر خواهد شد .

به مثال زیر توجه نمایید :

LD	IL
	LD %I0.0 CTUD C1, %I0.1, %I0.2, %I0.3, 4 ,%Q0.1 ST %Q0.0



8.13.3: شمارنده های سرعت بالا(High speed counter):

این دستورات برای شمارش پالس های ورودی با سرعت بالا که نمیتوانند با سرعت اسکن CPU کنترل شوند (مانند پالس های دریافتی از یک انکودر) به کار میرود .

	Name	Usage	Group	
LD	HDEF	HDEF EN ENO HSC MODE		☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
	HSC	HSC EN ENO N		
IL	HDEF	HDEF *HSC, MODE*	U	
	HSC	HSC *N*		

Operands	Input/Output	Data Type	Description
HSC	Input	INT constant (0~5)	HSC number
MODE	Input	INT constant (0~11)	Operations mode
N	Input	INT constant (0~5)	HSC number

:HDEF

دستور HDEF(High Speed Counter Definition) تعیین کننده شماره high speed counter مورد استفاده و نیز مد کاری مورد نظر میباشد .شماره high speed counter در پایه HSC و مد کاری مورد نظر در پایه MODE تعیین میگردد.این دستور برای تمامی high speed counterها استفاده میگردد. هر یک از high speed counter ها میتواند در یکی از ۱۲ مد کاری (0~11) پیکربندی شود (باید توجه داشته باشید که تمامی high speed counter ها نمیتوانند تمامی مدهای کاری را پشتیبانی کنند، در بخش بعدی به آن اشاره خواهد شد .) .مد کاری در یک high speed counter بیان کننده مشخصات ورودی clock، جهت شمارش ، startو reset میباشد .

HSC:

دستور HSC ، High speed counter مورد نظررا که شماره آن در پایه N مشخص شده است بر اساس مقادیر رجیسترهای SMمربوطه پیکربندی و راه اندازی مینماید .

8.13.3.1: high speed counter هایی پشتیبانی شده در Kinco-K3:

Feature	CPU304	CPU306
High-speed counters	2 counters (HSC0 and HSC1)	6 counters (HSC0 to HSC5)
Single phase	2 at 20KHz	6 at 30KHz
Two phase	2 at 10KHz	4 at 20KHz.

HSC 3و HSC 5 دارای یک مد کاری ، HSC 4 و HSC0 دارای ۷ مد کاری و HSC1 و HSC2 دارای ۱۲ مد کاری میباشند . تمامی High speed counter ها در مدهای کاری یکسان دارای عملکردی مشابه و یکسان هستند.

هر ورودی در high speed counter مطابق الگوی زیر عمل میکند:

هنگامی که ورودی reset، TRUE باشد مقدار شمارش جاری (Current Value) پاک خواهد شد تا زمانی که مجددا ورودی resetغیر فعال (false) شود.

هنگامی که ورودی Start، TRUE باشد به کانتر اجازه شمارش داده میشود .زمانی که این ورودی false باشد مقدار شمارش جاری (current Value) به صورت ثابت باقی مانده و تغییر نمیکند و ورودی Clockهم در نظر گرفته نمیشود .

هنگامی که ورودی reset، TRUE و ورودی Start، false باشد ، ورودی reset در نظر گرفته نشده و مقدار Current Value بدون تغییر باقی میماند .زمانی که ورودی start و resetهردو به صورت همزمان TRUE باشند ، مقدار Current Value پاک خواهد شد .

در کانتر تک فاز (Single phase Coubter) با کنترل جهت خارجی چنان چه ورودی جهت ، TRUE باشد کانتر رو به بال میشمارد و چنانچه ورودی جهت false باشد کانتر رو به پایین میشمارد .

8.13.3.2. مد های کاری و ورودی های high speed counter:

در جدولی که در پایین مشاهده مینمایید مدهای کاری هر یک از high speed counter ها و نیز ورودی های قابل استفاده در هر مد کاری ارائه شده است .

HSC 0				
Mode	**Description**	**I0.1**	**I0.0**	**I0.5**
0	Single-phase up/down counter with internal direction control	Clock		
1			Reset	
2			Reset	Start
3	Single-phase up/down counter with external direction control	Clock		Direction
4			Reset	Direction
6	Two-phase counter with up/down clock inputs	Clock Up	Clock Down	
9	A/B phase quadrature counter	Clock B	Clock A	

HSC 1					
Mode	**Description**	**I0.3**	**I0.7**	**I1.2**	**I1.3**
0	Single-phase up/down counter with internal direction control			Clock	
1		Reset			
2		Reset	Start		
3	Single-phase up/down counter with external direction control			Clock	Direction
4		Reset			Direction
5		Reset	Start		Direction
6	Two-phase counter with up/down clock inputs			Clock Down	Clock Up
7		Reset			
8		Reset	Start		
9	A/B phase quadrature counter			Clock B	Clock A
10		Reset			
11		Reset	Start		

HSC 2					
Mode	**Description**	**I0.6**	**I1.1**	**I1.4**	**I1.5**
0	Single-phase up/down counter with internal direction control			Clock	
1		Reset			
2		Reset	Start		
3	Single-phase up/down counter with external direction control			Clock	Direction
4		Reset			Direction
5		Reset	Start		Direction
6	Two-phase counter with up/down clock inputs			Clock Down	Clock Up
7		Reset			
8		Reset	Start		
9	A/B phase quadrature counter			Clock B	Clock A
10		Reset			
11		Reset	Start		

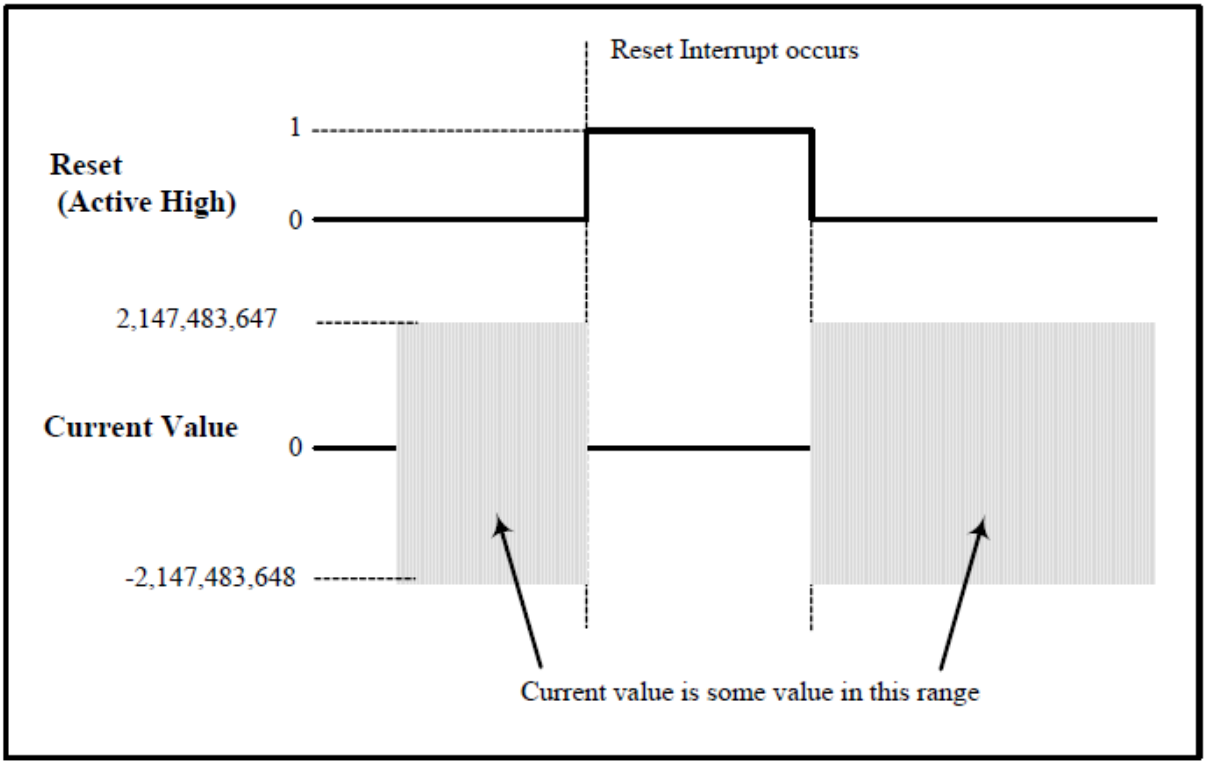
HSC 3		
Mode	**Description**	**I0.0**
0	Single-phase up/down counter with internal direction control	Clock

HSC 4				
Mode	**Description**	**I0.2**	**I1.0**	**I1.1**
0	Single-phase up/down counter with internal direction control	Clock		
1			Reset	
2			Reset	Start
3	Single-phase up/down counter with external direction control	Clock		Direction
4			Reset	Direction
6	Two-phase counter with up/down clock inputs	Clock Down	Clock Up	
9	A/B phase quadrature counter	Clock B	Clock A	

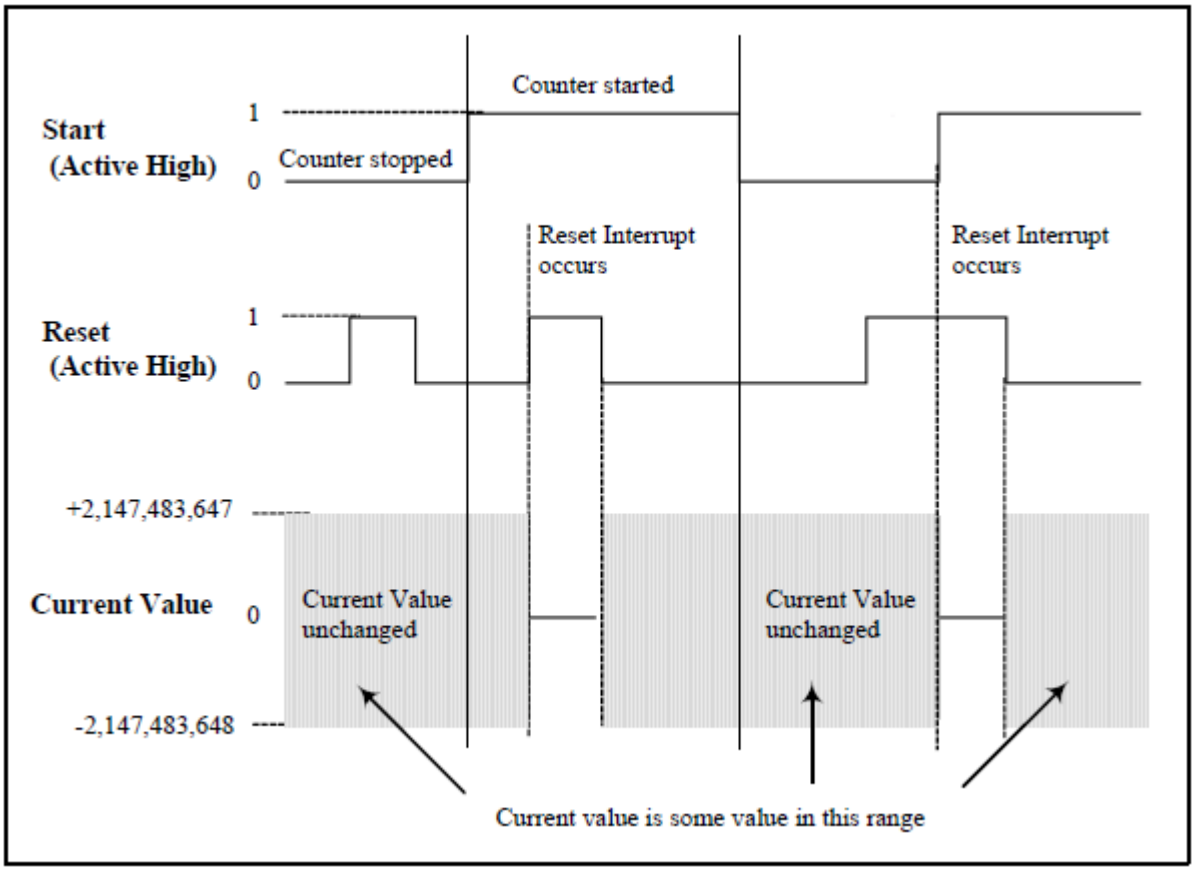
HSC 5		
Mode	**Description**	**I0.3**
0	Single-phase up/down counter with internal direction control	Clock

به منظور درک بهتر مفهو م high speed counter و مطالب ارائه شده در بالا به دیاگرام هایی که در ادامه ارائه شده است توجه نمایید :

Resetو Start:



Time Sequence with Reset and without Start



Time Sequence with Reset and Start

HSC2,HSC1,HSC0 و HSC4 دارای سه بیت کنترلی جهت تنظیمات و پیکربندی

شمارنده های سرعت بالای مربوطه میباشند .شرح این بیت های کنترلی در جدول زیر ارائه شده است :

HSC0	HSC1	HSC2	HSC4	Description
SM37.0	SM47.0	SM57.0	SM147.0	Control bit for active level of Reset: 0 = Active High; 1 = Active Low
SM37.1	SM47.1	SM57.1	SM147.1	Control bit for active level of Start: 0 = Active High; 1 = Active Low
SM37.2	SM47.2	SM57.2	SM147.2	Control bit for counting rate of quadrature counter: 0 = 4x counting rate; 1 = 1x counting rate

باید در نظر داشت که قبل از اجرای دستور HSC این بیت های کنترلی باید مطابق وضعیت دلخواه پیکر بندی شوند . در غیر این صورت شمارنده وضعیت از پیش تعیین شده را برای مد کاری انتخابی در نظر میگرید . وضعیت از پیش تعیین شده به شرح زیر میباشد :

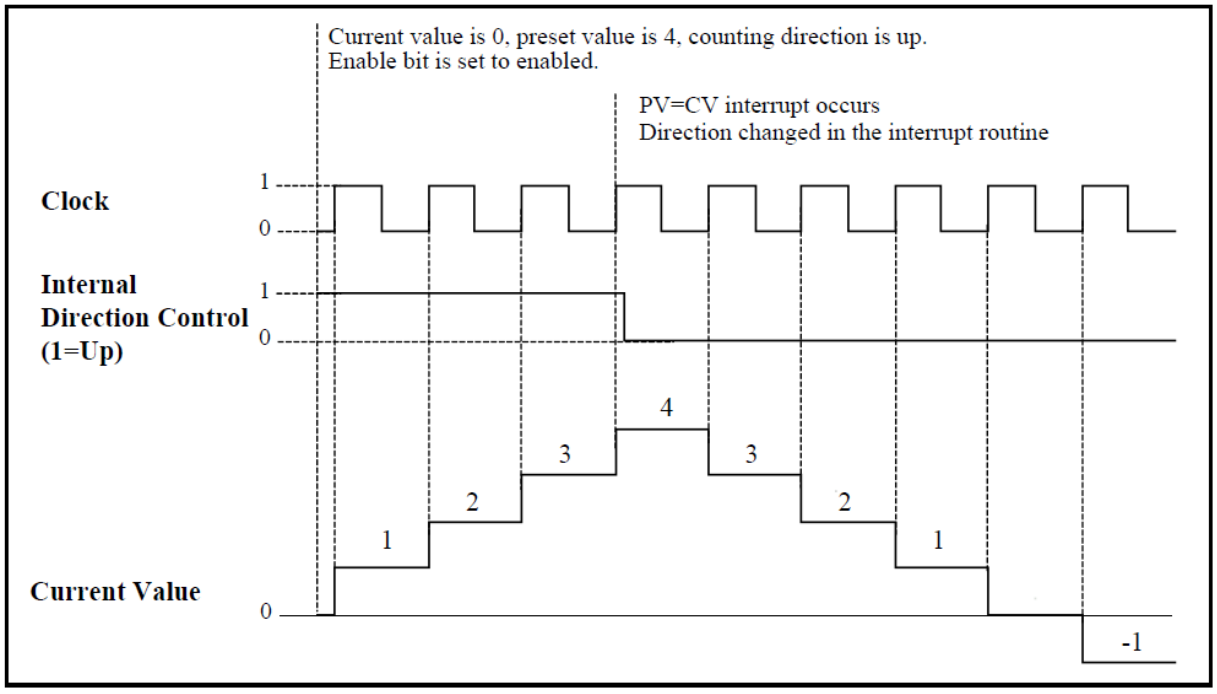
ورودی Start: active high

ورودی Reset: active high

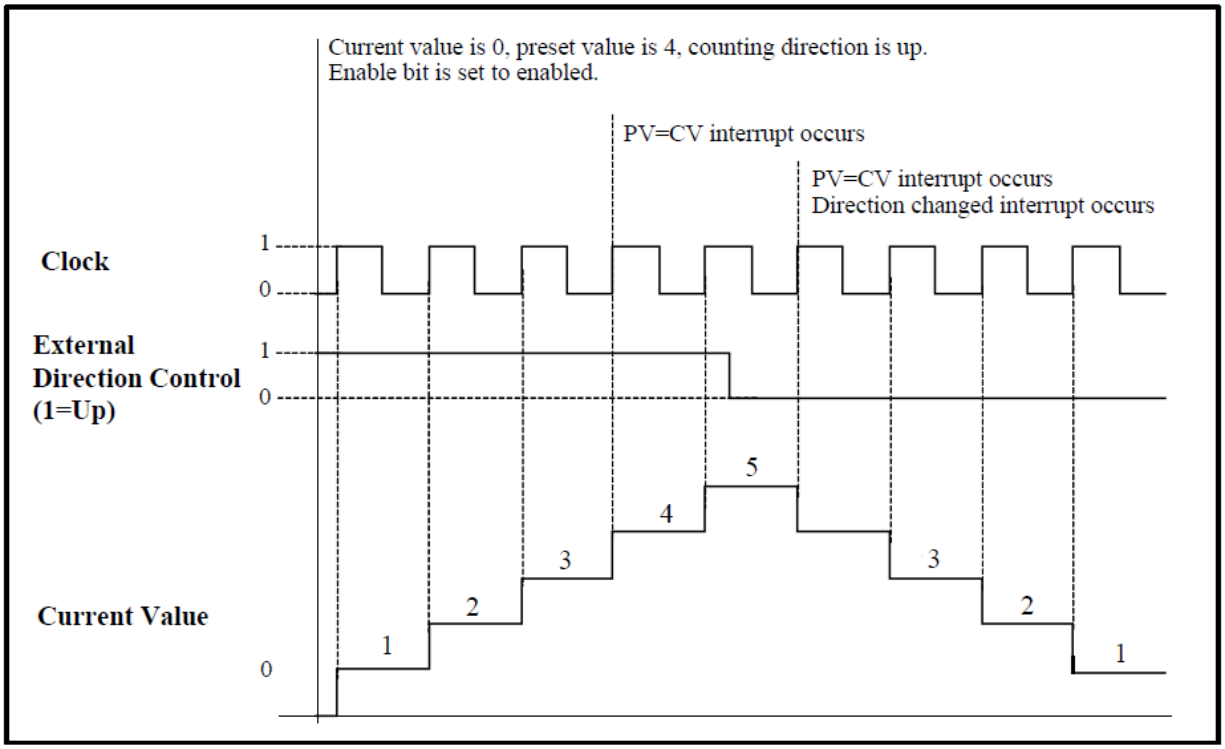
Quadrature rate: 1x counting rate

زمانی که دستور HSC اجرا شد ، پیکر بندی شمارنده قابل تغییر نمیباشد .

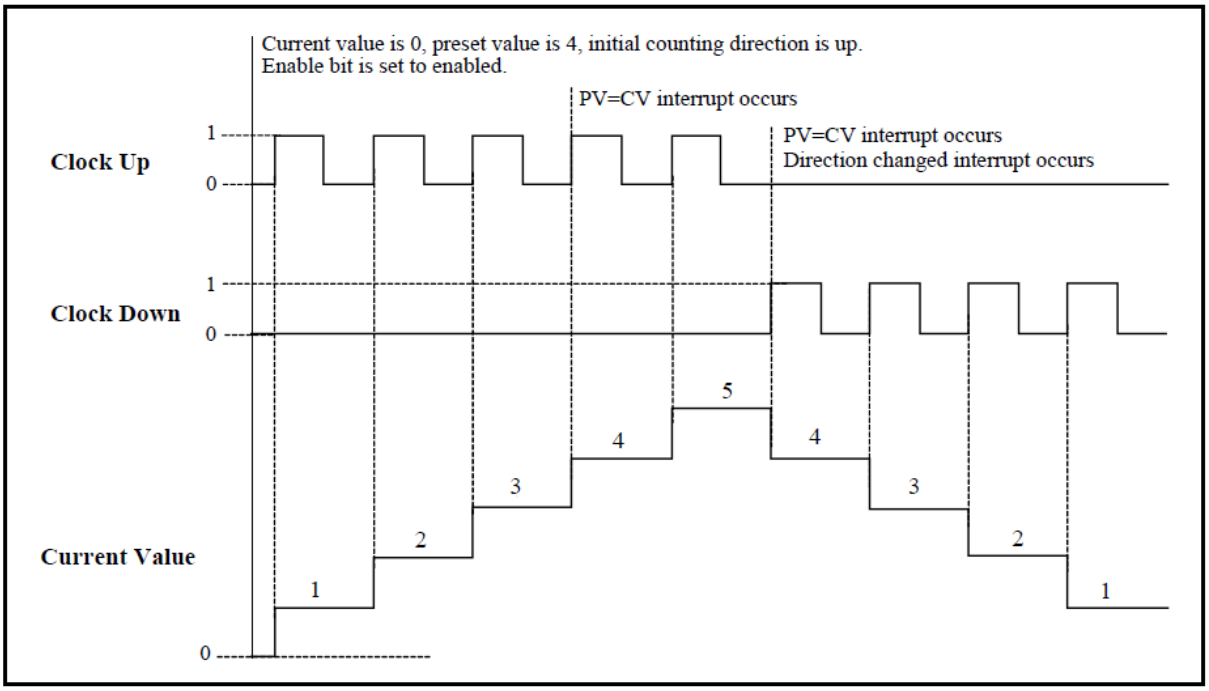
دیاگرام های زمانی که در پایین ارائه شده است چگونگی عملکرد هر شمارنده را در مدهای کاری مختلف نشان میدهد :



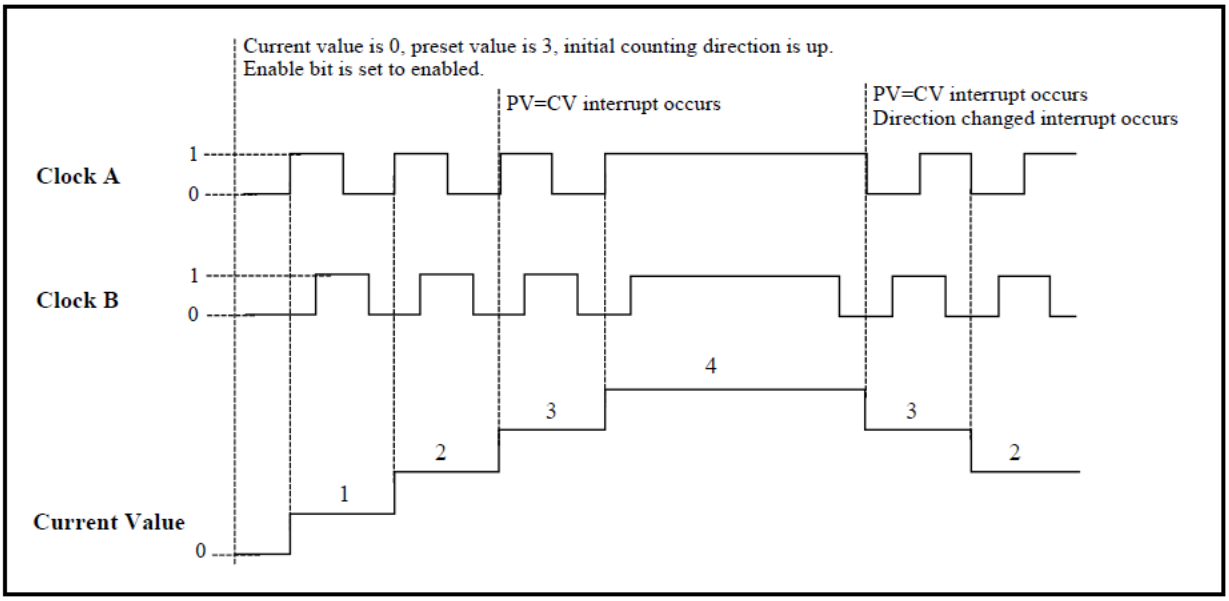
Time Sequence of Mode 0, 1 or 2



Time Sequence of Mode 3, 4 or 5



Time Sequence of Mode 6, 7 or 8



Time Sequence of Mode 9, 10 or 11 (Quadrature, 1x rate)

8.13.3.3. پیکر بندی بایت های کنترلی:

پارامترهای دینامیکی شمارنده سرعت بالا تنها زمانی که شمارنده و مد کاری آن تعیین شده باشد ، میتوانند برنامه نویسی شوند.

برای هر شمارنده یک بایت کنترلی اختصاص داده شده است که عملکردی که در پایین بیان شده است را بر عهده دارند:

فعال و غیر فعال سازی HSC

کنترل جهت شمارش (که مختص مد کاری ۰، ۱ و ۲ میباشد) یا جهت شمارش اولیه برای تمام مدها .

بار گذاری مقدار جاری در حال شمارش

بارگذاری مقدار از پیش تعیین شده

بایت کنترلی مربوط به مقدار جاری و مقدار از پیش تعیین شده قبل از اجرای دستور HSC باید بار گذاری شود.

در جدولی ارائه شده است ، بیت های مربوط به بایت کنترلی هر شمارنده را توضیح داده شده است :

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Description
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	Counting direction: 0 = Up; 1 = Down
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	Write the counting direction to the HSC: 0 = No; 1 = Yes
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	Write the new preset value to the HSC: 0 = No; 1 = Yes
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	Write the new current value to the HSC: 0 = No; 1 = Yes
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	Enable the HSC: 0 = Disable; 1 = Enable

پیکر بندی مقدار جاری و مقدار از پیش تعیین شده :

هر شمارنده سرعت بالا دارای یک مقدار جاری ۳۲ بیتی(Current value) و یک مقدار ازپیش تعیین شده (Preset value) ۳۲ بیتی میباشد .هم مقدار جاری و هم مقدار از پیش تعیین شده به صورت یک داده Double INT علامت دار میباشند .به منظور نوشتن مقدار جاری جدید و مقدار از پیش تعیین شده در شمارنده سرعت بالا ابتدا باید بایت کنترلی و بایت های SM (که مقدار جاری و از پیش تعیین شده در آن ذخیره میگردد) پیکر بندی شوند و سپس باید دستور HSC اجرا شود تا این مقادیر بتواند در شمارنده سرعت بالا نوشته شود . جدولی که د رادامه می بینید بایت هایی از حافظه SMرا که مقدار جاری جدید و مقدار از پیش تعیین شده در آن ذخیره میگردد را نشان میدهد :

	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
New current value	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
New preset value	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

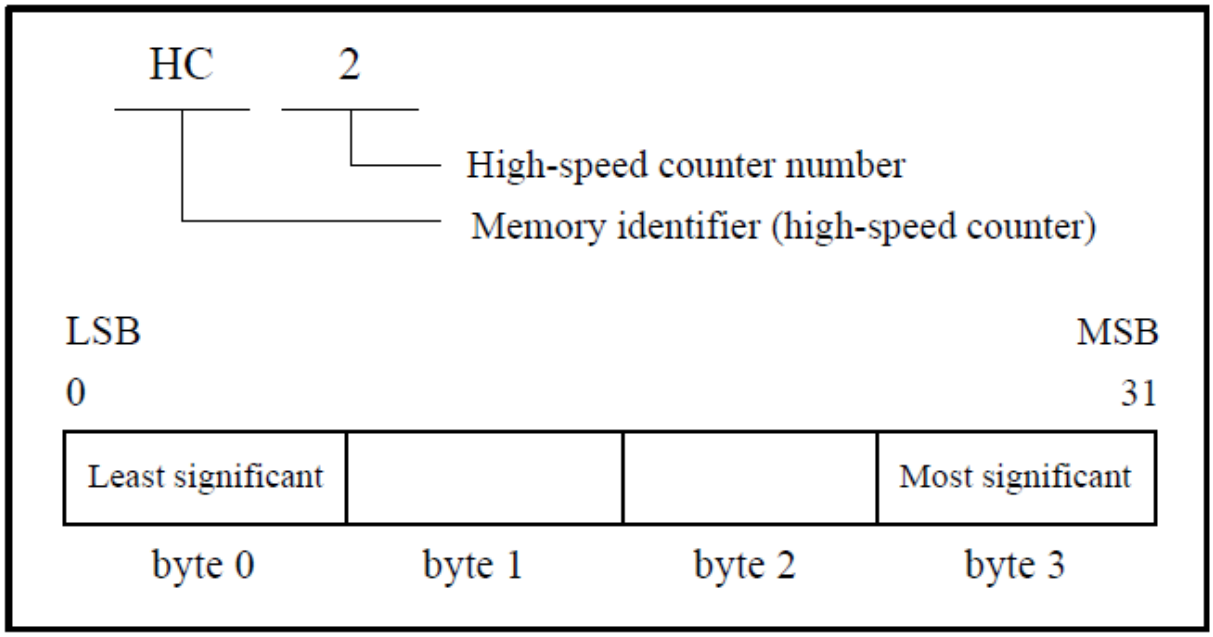
دسترسی به مقدار در حال شمارش جاری در شمارنده های سرعت بالا:

مقدار در حال شمارش در شمارنده های سرعت بالا به صورت یک داده با فرمت

Double INT بوده و فقط قابل خواندن میباشد (نمیتوان در آن چیزی نوشت) .

این مقدار در شمارنده های سرعت بالا با استفاده از حافظه HC و نیز شماره مربوط به شمارنده قابل دسترس میباشد .به عنوان مثال مقدار درحال شمارش HSC0(شمارنده سرعت بالای ۰) در حافظه HC0 ذخیره میگردد و با فراخوانی این

حافظه میتوان به مقدار در حال شمارش دسترسی پیدا کرد .



اختصاص دادن وقفه :

تمام مدهای کاری وقفه PV=CV(برابری مقدار در حال شمارش با مقدار از پیش تعیین شده) را پشتیبانی میکنند.

مد ی که از پایه ورودی RESETخارجی استفاده میکند ، وقفه RESET(RESET INTERRUPT) را پشتیبانی کرده و نیز مدی که از پایه ورودی کنترل جهت خارجی استفاده میکند، وقفه تغییر جهت(Direction Changed interrupt) را پشتیبانی میکند.

هر کدام از این وقفه ها میتوانند به صورت مجزا فعال و یا غیر فعال شوند . برای جزئیات بیشتر به بخش 6.10.3 انواع وقفه هایی که توسط kinco –k3پشتیبانی میشود مراجعه نمایید .

8.13.3.4 بایت های وضعیت :

هر شمارنده سرعت بالا در فضای حافظه SM دارای یک بایت وضعیت میباشد که برخی از بیت های آن بیان گر جهت شمارش فعلی و نیز برابر و یا بزرگ تر بودن مقدار در حال شمارش نسبت به مقدار از پیش تعیین شده میباشد .جدول زیر نشان دهنده این بیت های وضعیت میباشد :

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Description
SM36.0	SM46.0	SM56.0	SM136.0	SM146.0	SM156.0	Reserved
SM36.1	SM46.1	SM56.1	SM136.1	SM146.1	SM156.1	Reserved
SM36.2	SM46.2	SM56.2	SM136.2	SM146.2	SM156.2	Reserved
SM36.3	SM46.3	SM56.3	SM136.3	SM146.3	SM156.3	Reserved
SM36.4	SM46.4	SM56.4	SM136.4	SM146.4	SM156.4	Reserved
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	Current counting direction: 0 = Down; 1= Up
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	Current value equal to preset value: 0 = Not equal; 1 = Equal
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	Current value greater than preset value: 0 = Not greater than; 1 = Greater than

8.13.3.5 برنامه نویسی شمارنده های سرعت بالا :

میتوان شمارنده های سرعت بالا را مطابق الگوی ارائه شده برنامه نویسی کرد :

اختصاص بایت کنترلی

اختصاص مقدار در حال شمارش (مقدار شروع شمارش) و مقدار از پیش تعیین شده

اختصاص روتین وقفه با استفاده از دستو ر ATCH(استفاده از وقفه به صورت اختیاری و با توجه به پروژه و نظر برنامه نویس میباشد)

مشخص کردن شمارنده و مد کاری مورد نظر با استفاده از دستور HDEF

نکته : دستور HDEF میتواند برای هر شمارنده فقط یک بار و پس از آن که CPU در مد RUN قرار گرفت ، اجرا شود.

راه اندازی شمارنده سرعت بالا با استفاده از دستورHSC :

د رادامه به عنوان مثال نحوه پارامتر دهی و عملکرد HSC0 توضیح داده خواهد شد :

نکته : توصیه میشود که در استفاده از شمارنده های سرعت بالا یک زیر برنامه شامل دستور HDEFو نیز سایر پارامتردهی ها ی دستورات ایجاد نموده و سپس این زیر برنامه را در Main برنامه با استفاده از SM0.1فراخوانی نمایید .(SM0.1 در زمانی که CPU در مد RUNقرار میگیرد یک بار ۱ شده و در بقیه زمان ها ۰ میباشد . به دلیل آن که پارامتردهی شمارنده های سرعت بالا فقط یک بار در برنامه انجام میشود و نیز به منظور کاهش زمان اسکن CPU از SM0.1 استفاده میشود)

استفاده از HSC:

مثالی که در ادامه بیان شده است برای مد کاری شماره ۹ است .

۱)برای پارامتردهی زیر برنامه ،مقدار مورد نظر (وضعیت کنترلی دلخواه) را در رجیستر SMB37 بار گذاری نمایید .

به عنوان مثال : SMB37=b#16#FC

این مقدار بیان گر :

فعال سازی HSC0

نوشتن مقدار جاری جدید در HSC0

نوشتن مقدار از پیش تعیین شده جدید در HSC0

تنظیم جهت شمارش به صورت بالا شمار

تنظیم ورودی Start و Reset به صورت active high (با سطح یک فعال شوند)

2)بار گذاری مقدار جاری(Current Value) مورد نظر در SMD38(چنانچه ۰ در آن بارگذاری شود ، SMD38پاک میشود)

۳)بارگذاری مقدار از پیش تعیین شده(Preset Value) مورد نظر در SMD42

۴) انتساب واقعه PV=CV (Event شماره ۱۸) به روتین وقفه مورد نظر به منظور پاسخ گویی سریع در زمانی که مقدار در حال شمارش با مقدار از پیش تعیین شده برابر شود . (در نظر گرفتن وقفه به صورت اختیاری میباشد)

۵) انتساب واقعه تغییر جهت (Event شماره ۱۷) به روتین وقفه مورد نظر به منظور پاسخ گویی سریع به واقعه تغییر جهت . (در نظر گرفتن وقفه به صورت اختیاری میباشد)

۶) انتساب واقعه Reset خارجی (Event شماره ۱۶) به روتین وقفه مورد نظر به منظور پاسخ گویی سریع به واقعه Reset خارجی . (در نظر گرفتن وقفه به صورت اختیاری میباشد)

۷)اجرای دستور HDEF .ورودی HSC را ۰ و ورودی Mode را ۹ تنظیم کنید .

۸)اجرای دستور HSC به منظور پیکر بندی و راه اندازی HSC0.

تغییر جهت شمارش برای مدهای ۰، ۱و ۲:

در ادامه چگونگی تغییر جهت HSC0 برای مدهای کاری ۰ ، ۱ و ۲ توضیح داده میشود:

۱)بار گذاری مقدار مورد نظر در رجیستر کنترلی SMB37

به عنوان مثال : SMB37=b#16#90

فعال سازی شمارنده

تنظیم جهت شمارش جدید به صورت پایین شمار

۲)اجرای دستور HSC به منظور پیکر بندی و راه اندازی HSC0.

بارگذاری مقدار فعلی جدید در تمام مدهای کاری:

در ادامه چگونگی تغییر مقدار فعلی (Current value) HSC0 برای تمام مدهای کاری توضیح داده میشود:

بار گذاری مقدار مورد نظر در رجیستر کنترلی

به عنوان مثال :SMB37=b#16#C0

فعال سازی شمارنده

اجازه نوشتن مقدار فعلی جدید در HSC0

۲)بارگذاری مقدار مورد نظر در SMD38

۳)اجرای دستور HSC به منظور پیکر بندی و راه اندازی HSC0.

غیر فعال کردن شمارنده سرعت بالا در تمام مدهای کاری:

در ادامه چگونگی غیر فعال کردن HSC0 توضیح داده شده است :

بار گذاری مقدار مورد نظر در رجیستر کنترلی

به عنوان مثال : SMB37=b#16#00

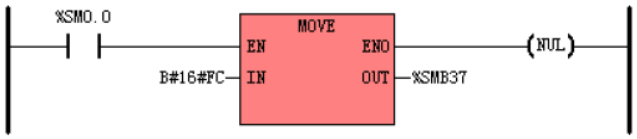
غیر فعال کننده شمارنده

۲) اجرای دستور HSC به منظورغیر فعال کردن HSC0.

LD

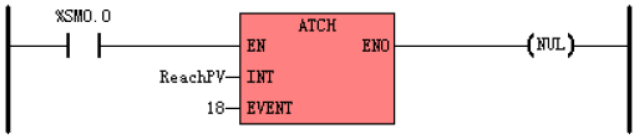
The initialization subroutine: Initialize

```
(* Network 0 *)
(* 1x counting rate; Enable HSC0; Allow updating current value and preset value;
   Up-counter; Set the start input and the reset input to be active high *)
```

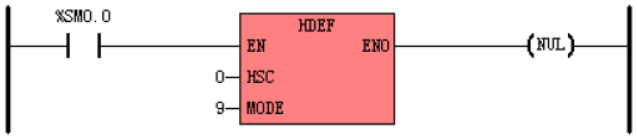



```
(* Network 1 *)
(* Set the new current value and new preset value *)
```

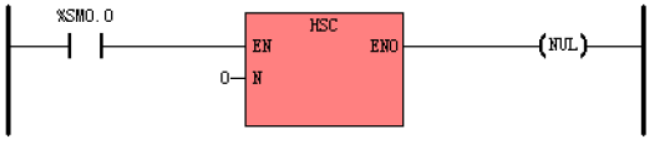

```
(* Network 2 *)
(* Attach the CV = PV event (event 18) to ReachPV interrupt routine *)
```




```
(* Network 3 *)
(* Define HSC0 to be in mode 9 *)
```



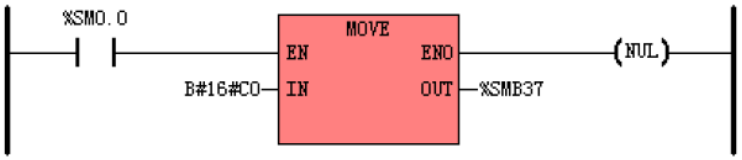

```
(* Network 4 *)
(* Configure and start HSC0 *)
```



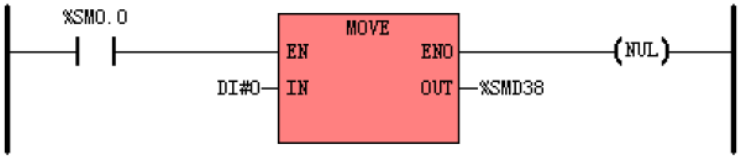
LD

The interrupt routine: ReachPV

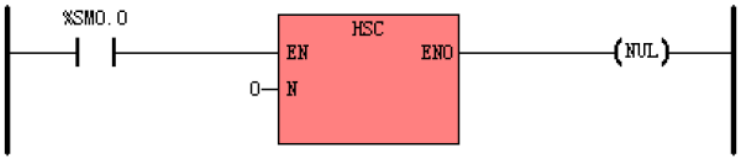
```
(* Network 0 *)
(* Allow updating current value *)
```




```
(* Network 1 *)
(* Set the new current value to be 0 to re-count *)
```

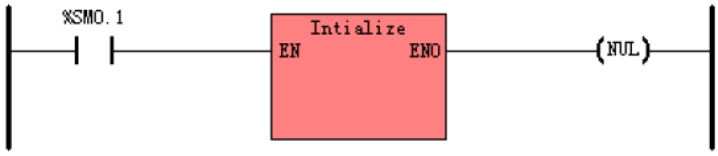



```
(* Network 2 *)
(* Configure and restart HSC0 *)
```



Main program:

```
(* Network 0 *)
(* Call Initialize subroutine *)
```



The initialization subroutine: Initialize

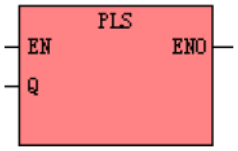
IL

```
(* Network 0 *)
(* 1x counting rate; Enable HSC0; Allow updating current value AND preset value; *)
(* Up-counter; Set the start input and the reset input to be active high *)
LD      %SM0.0
MOVE    B#16#FC, %SMB37
(* Network 1 *)
(*Set the new current value and new preset value*)
LD      %SM0.0
MOVE    DI#0, %SMD38
MOVE    DI#100, %SMD42
(* Network 2 *)
(*Attach the CV = PV event (event 18) to ReachPV interrupt routine*)
LD      %SM0.0
ATCH    ReachPV, 18
(* Network 3 *)
(*Define HSC0 to be in mode 9*)
LD      %SM0.0
HDEF    0, 9
(* Network 4 *)
(*Configure and start HSC0*)
LD      %SM0.0
HSC     0
```

IL	**The interrupt routine: ReachPV**
	
	(* Network 0 *)
	(*Allow updating current value*)
	LD %SM0.0
	MOVE B#16#C0, %SMB37
	(* Network 1 *)
	(*Set the new current value to be 0 to re-count*)
	LD %SM0.0
	MOVE DI#0, %SMD38
	(* Network 2 *)
	(*Configure and restart HSC0*)
	LD %SM0.0
	HSC 0
	
	Main program:
	
	(* Network 0 *)
	(*Call Initialize subroutine*)
	LD %SM0.1
	CAL Intialize

8.13.4 دستورات خروجی پالس سرعت بالا:

در این قسمت منظور از خروجی پالس سرعت بالا ، به معنی خروجی قطار پالس (PTO) ویا پالس با پهنای باند متغیر (PWM) میباشد .

	Name	Usage	Group	
LD	PLS			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	PLS	PLS *Q*	U	

Operands	Input/Output	Data Type	Description
Q	Input	INT constant (0 or 1)	Assign pulse output channel: 0 represents output through Q0.0; 1 represents output through Q0.1.

دستور PLS به منظور بارگذاری تنظیمات مربوط به PTO/PWM مشخص شده در پایه Q از رجیستر های SM مشخص شده و نیز راه اندازی تولید کننده PTO/PWM بر اساس آن ، به کار میرود .

LD: زمانی که EN، یک باشد دستور PLSاجرا خواهد شد.

IL: زمانی که CR یک باشد دستور PLSاجرا خواهد شد. این دستور تاثیری بر مقدار CR نخواهد داشت .

8.13.4.1: خروجی قطار پالس که توسط KINCO-K3 پشتیبانی میشود:

Kinco_K3 ، ۲ تولید کننده پالس PWM/PTO دارد که میتواندیا به عنوان خروجی قطار پالس سرعت بالا ویا به عنوان PWMبه کار رود .فرکانس پالس های خروجی تا 20KHZمیباشد .بنا براین تولید کننده ای که به Q0.0اختصاص داده شده است PWM0ویا PTO0 و تولید کننده ای که به Q0.1 اختصاص داده شده است PWM1و یا PTO1 نامیده میشود. زمانی که Q0.0 یا Q0.1 با عملکرد PTO ویا PWMبه کار میروند، تولید کننده PTO/PWM خروجی را کنترل کرده و نمیتوان از خروجی های Q0.0وQ0.1 به عنوان خروجی های معمول استفاده کرد .زمانی که این تولید کننده ها ی غیر فعال باشند میتوان از خروجی های Q0.0وQ0.1 به عنوان خروجی های معمول استفاده کرد .

برای هرکدام از تولید کننده های PTO/PWM تعدادی رجیستر در فضای حافظه SMدر نظرگرفته شده است که این حافظه ها شامل :

یک بایت کنترلی (۸ بیت) ، cycle time، مقدار پهنای پالس (داده با فرمت unsigned integer) و تعداد پالس (داده با فرمت unsigned integer) میباشد .

هنگامی که این رجیستر ها بر اساس مقادیر مورد نظر پیکربندی شوند ، با اجرای دستور PLS عملکرد مورد نظر اجرا خواهد شد .مقدار default این رجیسترها (بیت های کنترلی ، cycle time ،پهنای پالس و تعداد پالس) در ماژول CPU، ۰ در نظر گرفته میشود.

نکته : توجه داشته باشید که چنانچه Q0.0وQ0.1 خروجی رله ای میباشند ، ا زآنها به عنوان تولید کننده PTO/PWM استفاده ننمایید .

PWM(Pulse-Width Modulation):

PWMیک خروجی پالس پیوسته با سیکل زمانی (Cycle Time) ثابت و duty cycle متغیر ایجاد مینماید .شما میتوانید سیکل زمانی و پهنای باند را کنترل نمایید .

سیکل زمانی و پهنای باند هردو میتوانند بر اساس میلی ثانیه ویا میکرو ثانیه در نظر گرفته شوند .رنج مجاز سیکل زمانی به صورت 50~65535 µs یا 2~65535 ms میباشد .

رنج مجاز پهنای باند به صورت 0~65535 µs یا 0~65535 ms میباشد.

چنانچه پهنای باند بزرگ تراز سیکل زمانی باشد ، Duty cycle به صورت اتوماتیک 100% در نظر گرفته شده و خروجی به صورت پیوسته on خواهد شد .

چنانچه پهنای باند ۰ باشد ، Duty cycle، ۰ در نظر گرفته شده و خروجی ۰ خواهد شد .

شما میتوانید از دو روشی که در ادامه ارائه شده است جهت به روز رسانی (update کردن) مشخصات شکل موج PWM استفاده نمایید:

Synchronous Update:

Synchronous Update زمانی استفاده میشود که نیازی به تغییر پایه زمانی (Time base) (µsو یا ms) نمیباشد .

با Synchronous Update ،تغییرات مشخصات شکل موج در محدوده سیکل قرار میگیرد .

عملکرد یک PWM معمولی تغییر پهنای پالس است به صورتی که سیکل زمانی(Cycle Time) را ثابت نگه میدارد . بنابراین نیازی به تغییر ندارد .

ASynchronous Update:

زمانی که نیاز باشد Time base در PWM تغییریابد میتوان از روش Asynchronous Update میتوان استفاده کرد .

ASynchronous Update ممکن است از عملکرد PWMبه صورت آنی جلوگیری کرده و منجر به تولید شکل موج PWM غیر همزمان میگردد .این میتواند باعث ایجاد لرزش های نامطلوب در قطعات کنترلی شود .بنابراین توصیه

میشود برای استفاده از Update Synchronous PWM، time base مناسبی را برای تمام سیکل های زمانی انتخاب نمایید .

زمانی که دستور PLS اجرا شود، بیت ها ی کنترلی SM77.4وSM67.4 بیان گر روش update استفاده شده میباشد . در زمانی که Time base تغییر میکند ، ASynchronous Update بدون در نظر گرفتن بیت های کنترلی update رخ میدهد .

PTO(Pulse Train Output):

PTOتولید کننده یک خروجی موج مربعی با duty cycle ۵۰٪ میباشد که میتوان سیکل زمانی (که بر اساس میلی ثانیه و یا میکرو ثانیه میباشد) و نیز تعداد پالسهای خروجی را کنترل نمود . مانند شکل موج PWM، در PTO نیز رنج سیکل زمانی 50~65535 µs یا 2~65535 ms میباشد . دراین حالت چنانچه سیکل زمانی به صورت یک عدد فرد در نظر گرفته شود (به عنوان مثال 35 ms) ممکن است مقداری اعوجاج در duty cycle رخ دهد .

رنج مجاز برای تعداد پالس های خروجی به صورت 1~4,294,967,295 میباشد .

چنانچه تعداد پالس ۰ تعریف شود ، تعداد پالس ها به صورت پیش فرض ۱ در نظر گرفته میشود .

دستور PTO میتواد یک قطار پالس تولید نماید ، همچنین میتواند با استفاده از یک پروفایل (profile pulse) میتواند مجموعه ای از پالس های متعدد را تولید نماید .در این حالت قطار پالس جدید بلافاصله پس از اتمام قطار پالس فعلی تولید میگردد.

Single_segment Pipelining:

در حالت Single_segment Pipelining برای تولید قطار پالس بعدی باید رجیستر های SM مربوطه را آپدیت کرد .

هنگامی که PTO اولیه شروع شد ، رجیسترهای SM مربوطه باید بلافاصله بر اساس مشخصات مورد نیاز موج مربعی دوم تغییر یابد و سپس دستور PLS مجددا اجرا گردد.

تنظیمات و پیکربندی قطار پالس دوم در یک pipeline(خط لوله) نگهداری میشود تا زمانی که قطار پالس اول تکمیل شود . سپس خروجی قطار پالس دوم بلافاصله آغازمیشود و مجددا پیکر بندی pipeline عوض شده و پیکر بندی قطار پالس بعدی در pipeline قرار میگیرد .این روند برای قطار پالس بعدی مجددا تکرار میشود .

زمان گذر بین قطارهای پالس بسیار ملایم و نامحسوس میباشد به جز درموارد زیر:

Time base تغییر یابد ویا قطار پالس فعلی به اتمام رسیده وCPU تنظیمات مربوط به قطار پالس جدید را با اجرای مجدد دستور PLS دریافت نکرده باشد .

Multi_Segment Pipelining:

در روش Multi_Segment Pipelining ، CPU به صورت اتوماتیک تنظیمات هر بخش از قطار پالس را از یک جدول مشخصات که در حافظه V (Variable) قرار دارد ، میخواند .

در این حالت Time base، در SMB67(برای PTO0) و در SMB167(برای PTO1) ذخیره میشود . offset برای آدرس شروع جدول در حافظه V در حافظه های SMW168(برای PTO0) و در SMW178(برای PTO1) ذخیره میگردد. همچنین time base میتواند بر اساس میلی ثانیه و میکرو ثانیه تعریف شود و برای تمام مقادیر cycle در جدول پروفایل مربوطه اعمال خواهد شد و نمیتواند در طول اجرای profile تغییر کند.

سپس PLS را به منظور شروع عملیات اجرا نمایید.

طول هر بخش (segment) ۸ بایت است که شامل Cycle time(16bit,1Word) ، مقدار افزایش cycle time(16 bit ,INT) و تعداد پالس (32 bit, DWORD)می باشد .

جدول زیر فرمت جدول پروفایل را توضیح میدهد:

Byte offset[1]	Length	Segment	Description
0	16-bit	-	The number of segments (1 to 64)
1	16-bit	1	Initial cycle time (2 to 65535 times of the time base)
3	16-bit		Cycle time increment for each pulse (-32768 to 32767 times of the time base)
5	32-bit		Pulse count (1 to 4,294,967,295)
9	16-bit	2	Initial cycle time (2 to 65535 times of the time base)
11	16-bit		Cycle time increment for each pulse (-32768 to 32767 times of the time base)
13	32-bit		Pulse count (1 to 4,294,967,295)
…	…	…	…

آدرس شروع جدول پروفایل باید یک آدرس فرد در فضای حافظه V باشد.

مقدار cycle Time میتواند به صورت اتوماتیک بر اساس مقدار افزایش Cycle value تعیین شده برای هر پالس کاهش یا افزایش یابد .

مقدار افزایش مثبت ، cycle time را افزایش داده و مقدار افزایشی منفی ، cycle time را کاهش میدهد. مقدار عددی ۰ ، cycle time را بدون تغییر نگه میدارد .

8.13.4.2.پیکر بندی و کنترل عملکرد PTO/PWM:

برای هر تولید کننده PTO/PWM تعدادی رجیستر ویژه در فضای حافظه SM در نظر گرفته شده است که وظیفه ذخیره تنظیمات پیکربندی و نیز نشان دادن وضعیت ها را برعهده دارند.

مشخصات موج های PTO/PWM میتواند با تغییر رجیسترهای SM مربوط به آن تغییر یابد و سپس دستور PLS اجرا شود .

جدول زیر تمامی رجیستر های کنترلی را با جزئیات توضیح میدهد :

Q0.0	Q0.1	Control bits
SM67.0	SM77.0	(PTO/PWM) Whether to update the cycle time: 0 = not update; 1 = update
SM67.1	SM77.1	(PWM) Whether to update pulse width time: 0 = not update; 1 = update
SM67.2	SM77.2	(PTO) Wheter to update the pulse count: 0 = not update; 1 = update
SM67.3	SM77.3	(PTO/PWM) Time base: 0 = 1μs; 1 = 1ms
SM67.4	SM77.4	(PWM) Update method: 0 = asynchronous update; 1 = synchronous update
SM67.5	SM77.5	(PTO) Single or multiple segment operation: 0 = single; 1 = multiple
SM67.6	SM77.6	Select PTO or PWM mode: 0 = PTO; 1 = PWM
SM67.7	SM77.7	(PTO/PWM) Enable: 0 = disable; 1 = enable
Q0.0	**Q0.1**	**Other registers**
SMW68	SMW78	(PTO/PWM) Cycle time value, Range: 2 to 65535
SMW70	SMW80	(PWM) Pulse width value, Range: 0 to 65535
SMD72	SMD82	(PTO) Pulse count value, Range: 1 to 4,294,967,295
SMB166	SMB176	The number of the segments in progress For multi-segment PTO operation only
SMW168	SMW178	The starting location of the profile table (byte offset from V0) For multi-segment PTO operation only

جدول زیر بیت های وضعیت مربوط به تولید کننده های PTO/PWM را توضیح میدهد :

Q0.0	Q0.1	Status Bits
SM66.4	SM76.4	PTO profile terminated due to increment calculation error: 0 = no error; 1 = terminated
SM66.5	SM76.5	PTO profile terminated due to user command: 0 = not terminated; 1 = terminated
SM66.6	SM76.6	PTO pipeline overflow/underflow 0= no; 1 = overflow/underflow
SM66.7	SM76.7	PTO idle 0 = in progress; 1 = iddle

بیت های SM76.7یا SM66.7 نشان دهنده کامل شدن خروجی قطار پالس میباشد ، به علاوه به محض کامل شدن قطار پالس ، روتین وقفه مربوط به آن فراخوانی میشود .

چنانچه عملکرد Multi –segment استفاده شده باشد ، روتین وقفه به محض کامل شدن جدول پروفایل فراخوانی میشود .

8.13.4.3 محاسبه جدول پروفایل :

عملکرد multi segment pipelining برای کاربردهای مختلف بسیار مفید میباشد به ویژه کنترل استپ موتورها .

به عنوان مثال: میتوان از این نوع عملکرد برای کنترل یک استپ موتور براساس یک جدول پروفایل که شامل یک شتاب ، سرعت ثابت و یک شتاب منفی است و یا یک جدول پروفایل کامل شده شامل ۶۴ بخش که هر بخش به یک شتاب ، سرعت ثابت و یک شتاب منفی مربوط میشود استفاده کرد .

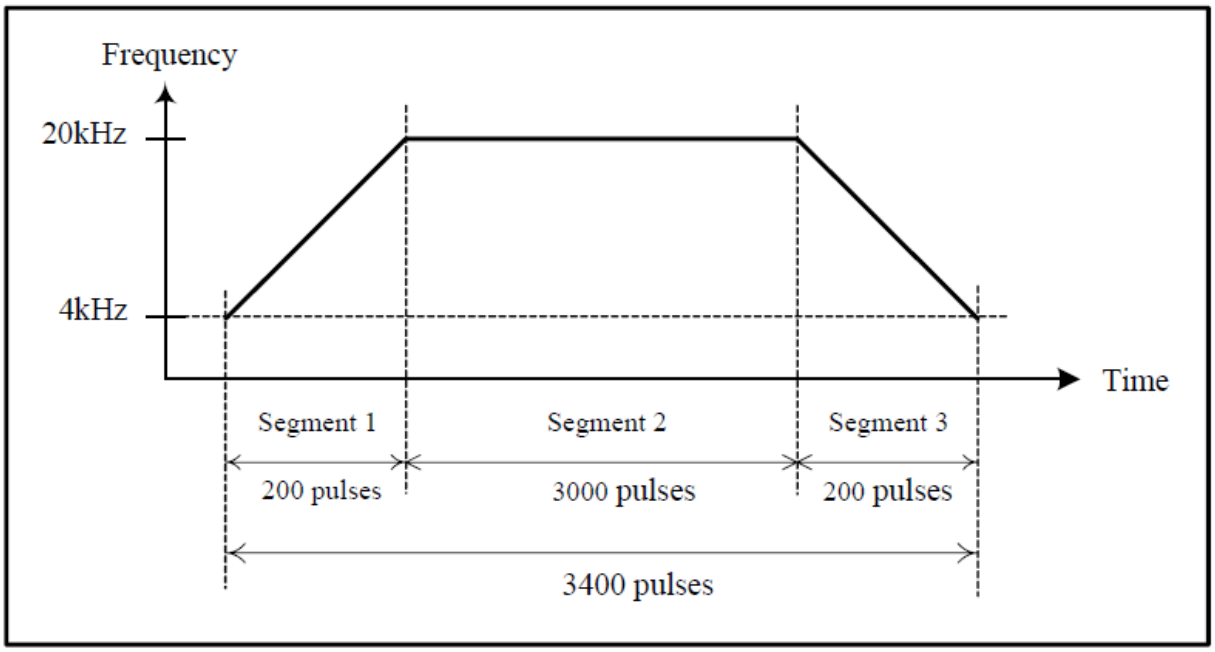
در ادامه یک مثال خاص جهت کنترل استپ موتور بیان شده است که چگونگی محاسبات مقادیر جدول پروفایل multi-segment را نشان میدهد .این جدول پروفایل شامل سه بخش میباشد :

بخش اول :شتاب مثبت استپ موتور

بخش دوم : عملکرد استپ موتور در یک سرعت ثابت

بخش سوم : شتاب منفی استپ موتور

به دیاگرام زیر توجه نمایید:



Sample Frequency/Time Diagram

در این مثال :

در بخش اول فرکانس خروجی در طی ۲۰۰ پالس از مقدار 4KHz به 20KHz افزایش میابد .

در بخش دوم : فرکانس خروجی به تعداد ۳۰۰۰ پالس در مقدار 20KHz ثابت باقی میماند.

دربخش سوم :فرکانس خروجی در طی ۲۰۰ پالس از مقدار 20KHz به 4KHz کاهش میابد.

به دلیل این که در جدول پروفایل به جای فرکانس از cycle Time استفاده میشود کاربر باید فرکانس را به مقدار cycle time تبدیل نماید .بنا براین cycle time اولیه و نهایی 20µs بوده و حداقل cycle time(بسته به ماکزیمم فرکانس) 50µs میباشد .

فرمول هایی که در ادامه به آن اشاره شده است میتواند برای محاسبه مقدار افزایشی cycle time برای هر بخش مورد استفاده قرار گیرد :

مقدار افزایشی cycle timeبرای هر بخش = (ETseg – ITseg)/Qseg

ETseg= مقدار نهایی cycle timeبرای بخش مربوطه

ITseg= مقدار اولیه cycle time برای بخش مربوطه

Qseg=تعداد پالسها در بخش مربوطه

با استفاده از فرمول ارائه شده در بالا میتوان مقدار افزایشی cycle time را در مثال بیان شده محاسبه کرد :

بخش اول : مقدار افزایشی cycle time= -1

بخش دوم :مقدار افزایش cycle time=۰

بخش سوم :مقدار افزایشی cycle time=۱

محاسبه جدول پروفایل در فضای حافظه V که با آدرس VB701آغاز میشود :

Byte Offset	Value	Comment	
VB701	3	The number of segments	
VW702	250	Initial cycle time	Segment 1
VW704	-1	Cycle time increment	
VD706	200	The number of pulses	
VW710	50	Initial cycle time	Segment 2
VW712	0	Cycle time increment	
VD714	3000	The number of pulses	
VW718	50	Initial cycle time	Segment 3
VW720	1	Cycle time increment	
VD722	200	The number of pulses	

انتقال بین بخش ها بسیار پر اهمیت و مهم میباشد .یک انتقال مناسب به این نکته وابسته میباشد که cycle time نهایی بخش قبلی به اضافه مقدار افزایشی cycle time برابر با مقدار اولیه cycle timeبخش بعدی گردد .

8.13.4.4 عملکرد PTO

در ادامه به منظور آشنایی با چگونگی پیکربندی و عملکرد تولید کننده PTO/PWM در قالب یک مثال، PTO0 توضیح داده شده است :

پارامتر دهی PTO(در حالت عملکرد Single-segment):

ابتدا یک زیر برنامه که شامل دستورات پارامتردهی میباشد را به وسیله بیت SM0.1 فراخوانی نمایید .(زمانی که از SM0.1استفاده میشود این زیر برنامه تنها یک بار فراخوانی و اجرا میشود و این باعث کاهش در زمان اسکن و عملکرد بهتر ساختار برنامه میشود) .

در ادامه مراحل چگونگی پیکربندی و تنظیمات PTO0 در زیربرنامه مربوط به پارامتردهی بیان شده است :

۱)بارگذاری وضعیت کنترلی مورد نظر در SMB67

به عنوان مثال : SMB67=B#16#85

مقدار فوق بیان گرموارد زیر میباشد :

فعال سازی تابع PTO/PWM

انتخاب عملکرد PTO

انتخاب 1µs به عنوان time base

اجازه Update شدن مقدار cycle time و تعداد پالس

بارگذاری cycle time در SMW68

بارگذاری تعداد پالس در SMD72

انتساب وقفه کامل شدن PTO0(وقفه شماره ۲۸) به روتین وقفه مربوطه به منظورپاسخ سریع به وقفه کامل شدن PTO0(به صورت اختیاری با صلاحدید برنامه نویس)

اجرای دستور PLSبه منظور پیکربندی و راه اندازی PTO0

تغییر cycle time در PTO(در حالت عملکرد single-segment):

د رادامه مراحل تغییر cycle timeدر PTO توضیح داده شده است :

بارگذاری وضعیت کنترلی مورد نظر در SMB67

به عنوان مثال :SMB67=B#16#81

مقدار فوق بیان گر موارد زیر میباشد:

فعال سازی تابع PTO/PWM

انتخاب عملکرد PTO

انتخاب 1µs به عنوان time base

اجازه Update شدن مقدار cycle time

بارگذاری cycle time در SMW68

اجرای دستور PLSبه منظور پیکربندی و راه اندازی PTO0

هنگامی که PTOفعال عملیات خود را به پایان رسانید ، یک شکل موج جدید PTO با cycle time جدید (updateشده) تولید خواهد شد .

تغییر تعداد پالس در PTO(در حالت عملکرد single-segment):

د رادامه مراحل تغییر تعداد پالس در PTO توضیح داده شده است :

بارگذاری وضعیت کنترلی مورد نظر در SMB67

به عنوان مثال :SMB67=B#16#84

مقدار فوق بیان گر موارد زیر میباشد:

فعال سازی تابع PTO/PWM

انتخاب عملکرد PTO

انتخاب 1µs به عنوان time base

اجازه Update شدن تعداد پالس

۲)بارگذاری تعداد پالس در SMD72

3) اجرای دستور PLSبه منظور پیکربندی و راه اندازی PTO0

هنگامی که PTOفعال عملیات خود را به پایان رسانید ، شکل موج جدید PTO با تعداد پالس جدید (updateشده) تولید خواهد شد .

تغییر cycle time و تعداد پالس در PTO (در حالت عملکرد single-segment):

د رادامه مراحل تغییر تعداد پالس و cycle time در PTO توضیح داده شده است :

1) بارگذاری وضعیت کنترلی مورد نظردر SMB67

به عنوان مثال: SMB67=B#16#85

مقدار فوق بیان گر موارد زیر میباشد:

فعال سازی تابع PTO/PWM

انتخاب عملکرد PTO

انتخاب 1µs به عنوان time base

اجازه Update شدن تعداد پالس و cycle time

بارگذاری cycle time در SMW68

بارگذاری تعداد پالس در SMD72

اجرای دستور PLSبه منظور پیکربندی و راه اندازی PTO0

هنگامی که PTOفعال عملیات خود را به پایان رسانید ، یک شکل موج جدید PTO با cycle time و تعداد پالس جدید (updateشده) تولید خواهد شد .

پارامتر دهی PTO(در حالت عملکرد Multi - segment):

ابتدا یک زیر برنامه که شامل دستورات پارامتردهی میباشد را به وسیله بیت SM0.1 فراخوانی نمایید .(زمانی که از SM0.1استفاده میشود این زیر برنامه تنها یک بار فراخوانی و اجرا میشود و این باعث کاهش در زمان اسکن و عملکرد بهتر ساختار برنامه میشود) .

در ادامه مراحل چگونگی پیکربندی و تنظیمات PTO0 در زیربرنامه مربوط به پارامتردهی بیان شده است :

بارگذاری وضعیت کنترلی مورد نظر در SMB67

به عنوان مثال :SMB67=B#16#A0

مقدار فوق بیان گر موارد زیر میباشد:

فعال سازی تابع PTO/PWM

انتخاب عملکرد PTO

انتخاب مد عملکردی Multi-segment

انتخاب 1µs به عنوان time base

۲) بارگذاری یک عدد فرد به عنوان موقعیت شروع جدول پروفایل

۳)استفاده از فضای حافظه V جهت پیکربندی جدول پروفایل

۴)انتساب وقفه کامل شدن PTO0(وقفه شماره ۲۸) به روتین وقفه مربوطه به منظورپاسخ سریع به وقفه کامل شدن PTO0(به صورت اختیاری با صلاحدید برنامه نویس)

۵)اجرای دستور PLSبه منظور پیکربندی و راه اندازی PTO0

8.13.4.5 عملکرد PWM:

در ادامه به عنوان مثال PWM0 مطرح میشود که در آن با چگونگی پیکربندی و عملکرد تولید کننده PTO/PWM در برنامه آشنا خواهید شد .

پارامتردهی خروجی PWM:

ابتدا یک زیر برنامه که شامل دستورات پارامتردهی میباشد را به وسیله بیت SM0.1 فراخوانی نمایید .(زمانی که از SM0.1استفاده میشود این زیر برنامه تنها یک بار فراخوانی و اجرا میشود و این باعث کاهش در زمان اسکن و عملکرد بهتر ساختار برنامه میشود).

در ادامه مراحل چگونگی پیکربندی و تنظیمات PTO0 در زیربرنامه مربوط به پارامتردهی بیان شده است :

۱)بارگذاری وضعیت کنترلی مورد نظردر SMB67

به عنوان مثال :SMB67=B#16#D3

مقدار فوق بیان گر موارد زیر میباشد:

فعال سازی تابع PTO/PWM

انتخاب عملکرد PWM

انتخاب 1µs به عنوان time base

اجازه Update شدن مقدار عرض پالس و cycle time

انتخاب روش synchronous update

بارگذاری مقدار cycle timeدر SMW68

بارگذاری مقدار عرض پالس در SMW70

اجرای دستور PLSبه منظور پیکربندی و راه اندازی PWM0

تغییر پهنای باند (Pulse width) برای خروجی PWM:

مراحل زیر چگونگی تغییر پهنای باند خروجی PWM را توضیح میدهد :

تصور نمایید که SMB67 با مقدارB#16#D2 ویا B#16#DA بارگذاری شده است :

بارگذاری مقدار عرض پالس در (16 bit) در SMW70

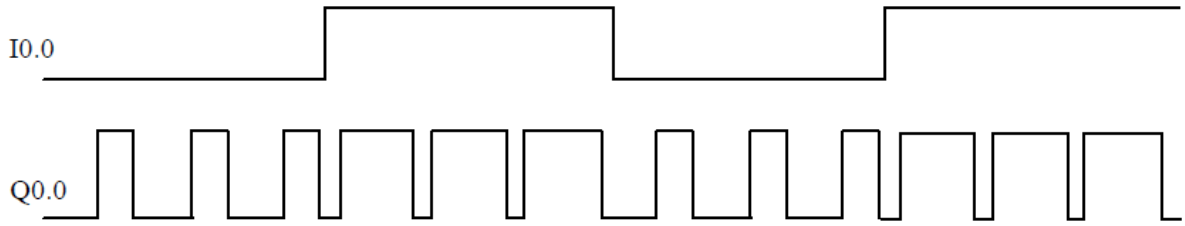
اجرای دستور PLSبه منظور پیکربندی و راه اندازی PWM0

به مثال زیر توجه نمایید:

در این مثال از خروجی PWM1(خروجی از طریق Q0.1میباشد) استفاده شده است :

چنانچه I0.0، false (صفر) باشد ، مقدار Duty cycle به مقدار 40% تغییر یافته و چنانچه I0.0 ،True (یک باشد) مقدار Duty cycleبه مقدار 80% تغییر میابد.

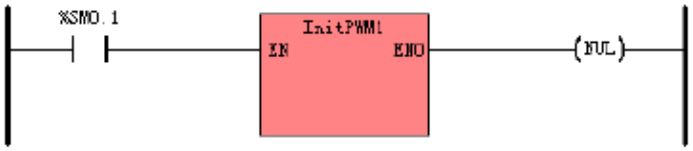
به دیاگرام زیر توجه نمایید:



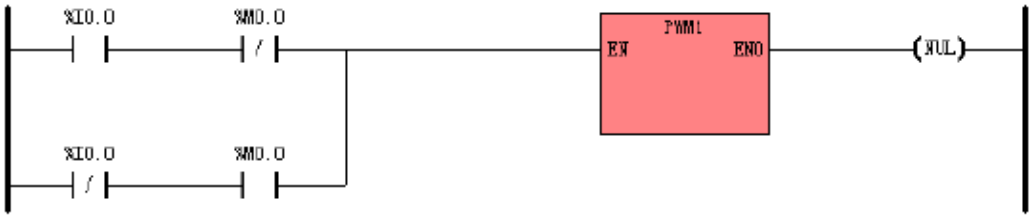
MAIN Program:

LD

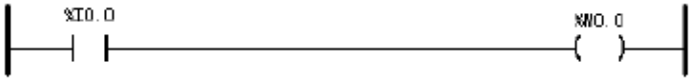
```
(* Network 0 *)
(* Use SM0.1 to call subroutine InitPWM1 to initialize PWM1 *)
```




```
(* Network 1 *)
(* If the status of I0.0 changes,
subroutine PWM1 shall be called to change the pulse width. *)
```



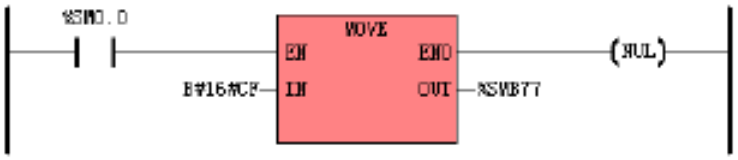

```
(* Network 2 *)
```



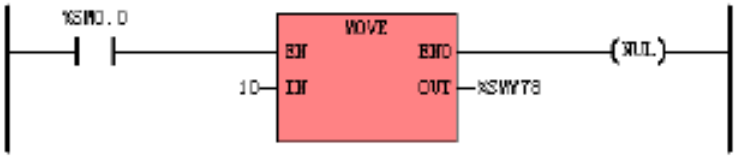
Subroutine *InitPWM1*:

LD

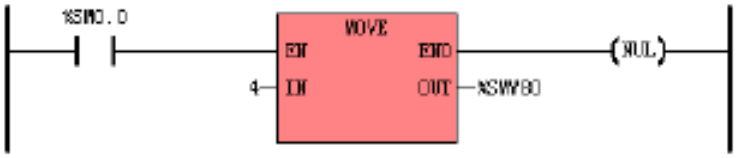
```
(* Network 0 *)
(* Select PWM1; Select 1ms as the time base;
Allow updating the cycle time value and the pulth width *)
```



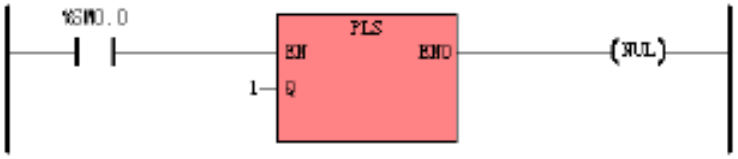

```
(* Network 1 *)
(* Set the cycle time of PWM1 to be 10ms *)
```




```
(* Network 2 *)
(* Set the pulse width of PWM1 to be 4ms *)
```



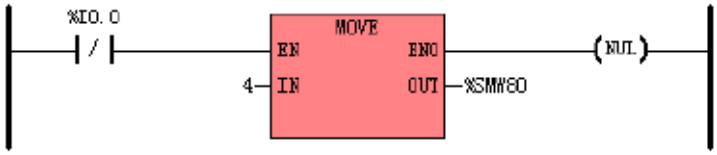

```
(* Network 3 *)
(* Execute PWM1 *)
```



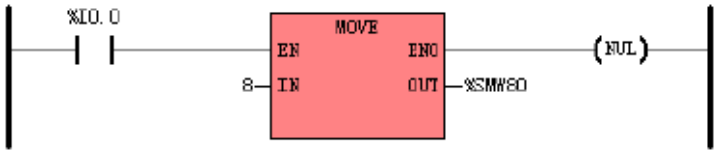
Subroutine *PWM1*:

LD

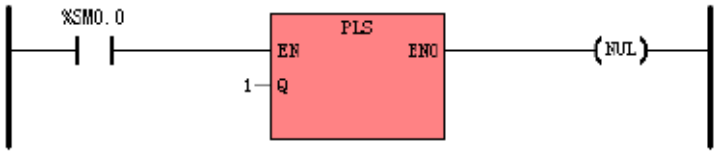
```
(* Network 0 *)
(* If I0.0 is false, the pulse width of PWM1 is set to be 4ms *)
```




```
(* Network 1 *)
(* If I0.0 is true, the pulse width of PWM1 is set to be 8ms *)
```




```
(* Network 2 *)
(* Execute PWM1 *)
```



IL	**MAIN Program:** (* Network 0 *) (* Use SM0.1 to call subroutine InitPWM1 to initialize PWM1 *) LD %SM0.1 CAL InitPWM1 (* Network 1 *) (* If the status of I0.0 changes, subroutine PWM1 shall be called to change the pulse width. *) LD %I0.0 ANDN %M0.0 OR(LDN %I0.0 AND %M0.0) CAL PWM1 (* Network 2 *) LD %I0.0 ST %M0.0

IL

Subroutin *InitPWM1*:

```
(* Network 0 *)
(*Select PWM1; Select 1ms as the time base; Allow updating the cycle time value and the pulth width*)
LD      %SM0.0
MOVE    B#16#CF, %SMB77
(* Network 1 *)
(*Set the cycle time of PWM1 to be 10ms*)
LD      %SM0.0
MOVE    10, %SMW78
(* Network 2 *)
(*Set the pulse width of PWM1 to be 4ms*)
LD      %SM0.0
MOVE    4, %SMW80
(* Network 3 *)
(*Execute PWM1*)
LD      %SM0.0
PLS     1
```

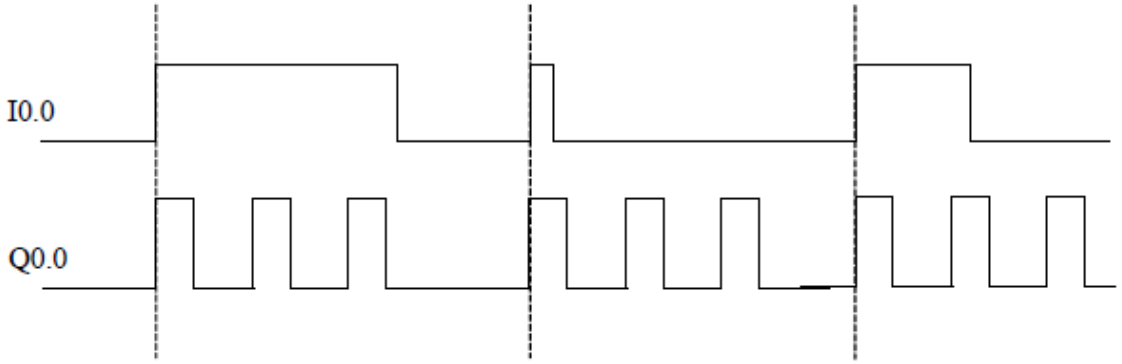
Subroutin *PWM1*:

```
(* Network 0 *)
(*If I0.0 is false, the pulse width of PWM1 is set to be 4ms*)
LDN     %I0.0
MOVE    4, %SMW80
(* Network 1 *)
(*If I0.0 is true, the pulse width of PWM1 is set to be 8ms*)
LD      %I0.0
MOVE    8, %SMW80
(* Network 2 *)
(*Execute PWM1*)
LD      %SM0.0
PLS     1
```

عملکرد PTO(مد single – segment):

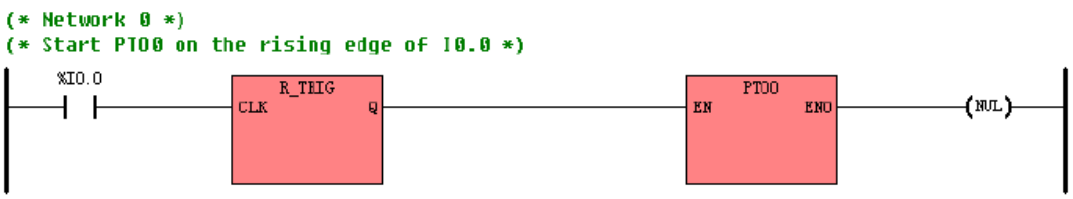
در این مثال از PTO0(خروجی از طریق Q0.0میباشد) استفاده شده است :

با هر لبه بالارونده I0.0 ، PTO0 راه اندازی و ۳ پالس تولید مینماید .به دیاگرام زیر توجه نمایید:


I0.0
Q0.0

LD

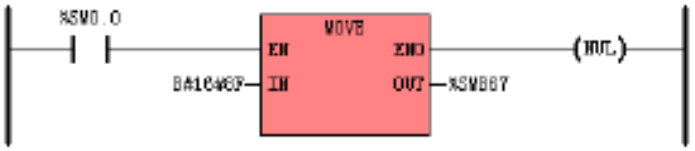
MAIN Program:



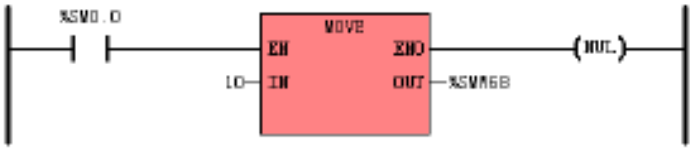
LD

Subprogram *PTO0*:

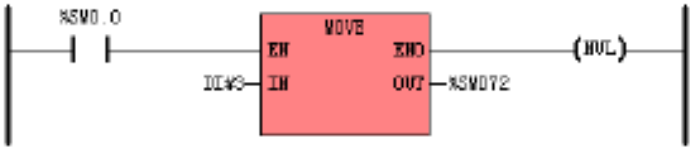
```
(* Network 0 *)
(* Select a single-segment operation for PTO0;
Select 1ms as the time base; Allow updating the cycle time and the pulse count *)
```



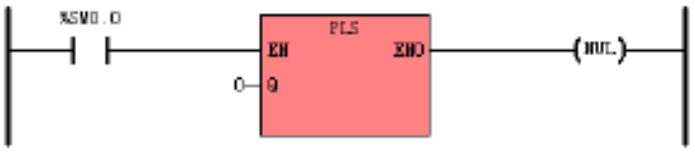

```
(* Network 1 *)
(* Set the cycle time to be 10ms *)
```




```
(* Network 2 *)
(* Set the pulse count to be 3 pulses *)
```




```
(* Network 3 *)
(* Execute PTO0 *)
```



IL

MAIN Pogram:

```
(* Network 0 *)
(* Start PTO0 on the rising edge of I0.0 *)
LD      %I0.0
R_TRIG
CAL     PTO0
```

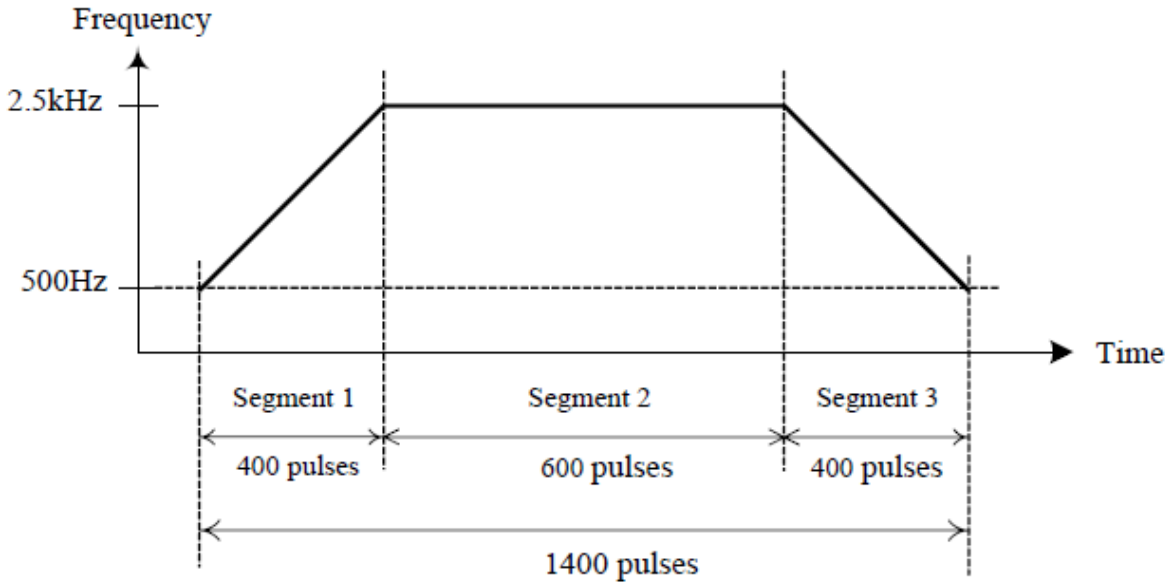
Subprogram PTO0:

```
(* Network 0 *)
(* Select a single-segment operation for PTO0; *)
(* Select 1ms as the time base; Allow updating the cycle time and the pulse count *)
LD      %SM0.0
MOVE    B#16#8F, %SMB67
(* Network 1 *)
(* Set the cycle time to be 10ms *)
LD      %SM0.0
MOVE    10, %SMW68
(* Network 2 *)
(* Set the pulse count to be 3 pulses *)
LD      %SM0.0
MOVE    DI#3, %SMD72
(* Network 3 *)
(* Execute PTO0 *)
LD      %SM0.0
PLS     0
```

عملکرد PTO(Multi-segment):

در این مثال از PTO0(خروجی از طریق Q0.0میباشد) استفاده شده است :

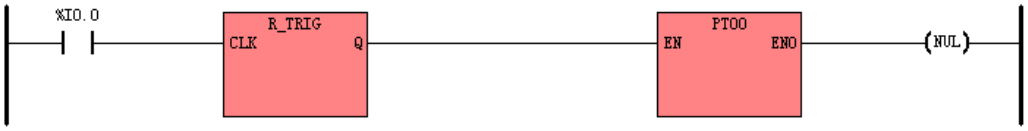
با لبه بالارونده I0.0، PTO0 راه اندازی میشود.با توجه به دیاگرام زیر میتوان جدول پروفایل multi-segment را محاسبه کرد .


Frequency
2.5kHz
500Hz
Time
Segment 1
Segment 2
Segment 3
400 pulses
600 pulses
400 pulses
1400 pulses

LD

MAIN Program:

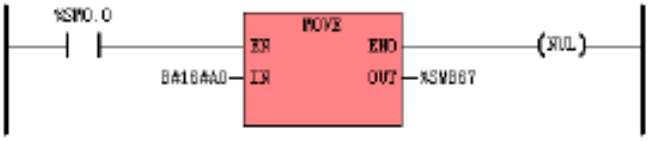
```
(* Network 0 *)
(* Start PTO0 on the rising edge of I0.0 *)
```



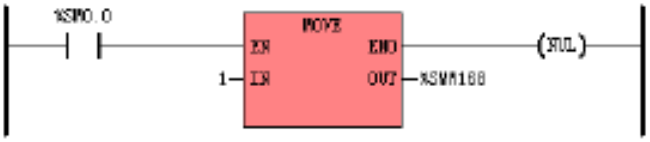
LD

Subroutine *PTO0*:

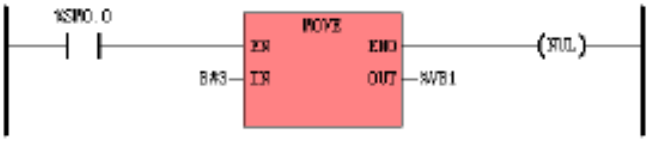
```
(* Network 0 *)
(* Enable PTO0; Select multi-segment operation; Set the time base to be 1us *)
```



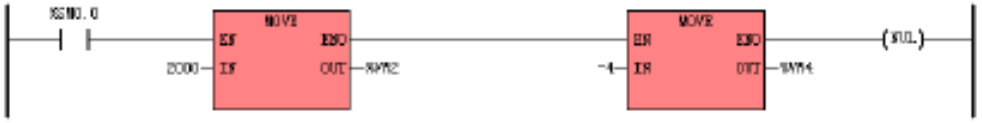

```
(* Network 1 *)
(* Use VB1 as the staring position of the profile table *)
```




```
(* Network 2 *)
(* Set the number of segments to be 3 *)
```




```
(* Network 3 *)
(* Segment 1: Set the initial cycle time to 2000us, set the cycle time increment to -4us *)
```

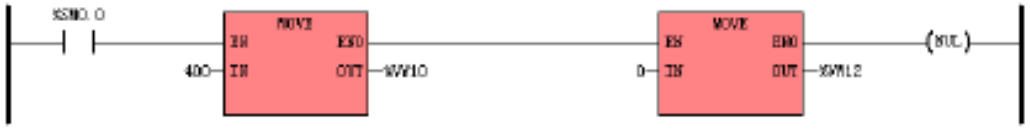



```
(* Network 4 *)
(* Segment 1: Set the number of pulses to 400 *)
```

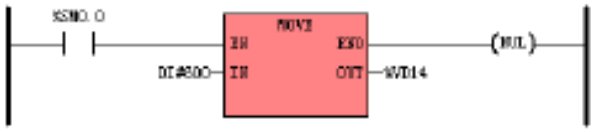
LD

Subroutine *PTO0*: (Continued)

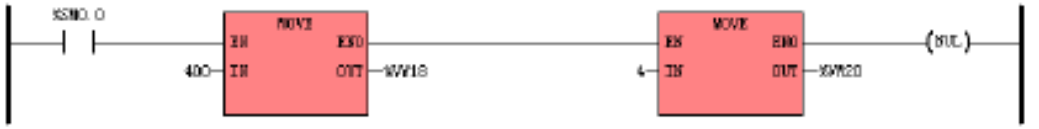
```
(* Network 5 *)
(* Segment 2: Set the initial cycle time to 400us, set the cycle time increment to 0 *)
```



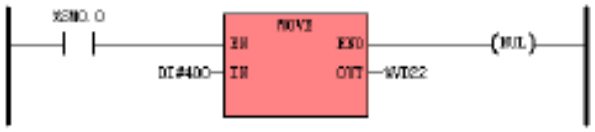

```
(* Network 6 *)
(* Segment 2: Set the number of pulses to 600 *)
```




```
(* Network 7 *)
(* Segment 3: Set the initial cycle time to 400us, set the cycle time increment to 4us *)
```




```
(* Network 8 *)
(* Segment 3: Set the number of pulses to 400 *)
```




```
(* Network 9 *)
(* Execut PT00 *)
```

IL

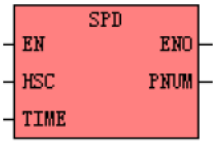
MAIN Program:

```
(* Network 0 *)
LD      %I0.0
R_TRIG
CAL     PTO0            (* Start PTO0 on the rising edge of I0.0 *)
```

Subroutine PTO0:

```
(* NETWORK 0 *)
LD      %SM0.0
MOVE    B#16#A0, %SMB67     (* Enable PTO0; Select multi-segment operation; Set the time base to be 1us *)
MOVE    1, %SMW168          (* Use VB1 as the staring position of the profile table *)
MOVE    B#16#03, %VB1       (* Set the number of segments to be 3 *)

(* Segment 1 *)
MOVE    2000, %VW2          (* Set the initial cycle time to 2000us *)
MOVE    -4, %VW4            (* Set the cycle time increment to -4us *)
MOVE    DI#400, %VD6        (* Set the number of pulses to 400 *)

(* Segment 2 *)
MOVE    400, %VW10          (* Set the initial cycle time to 400us *)
MOVE    0, %VW12            (* Set the cycle time increment to 0 *)
MOVE    DI#600, %VD14       (* Set the number of pulses to 600 *)

(* Segment 3 *)
MOVE    400, %VW18          (* Set the initial cycle time to 400us *)
MOVE    4, %VW20            (* Set the cycle time increment to 4us *)
MOVE    DI#400, %VD22       (* Set the number of pulses to 400 *)

PLS     0                   (* Execute PTO0 *)
```

SPD.8.13.5

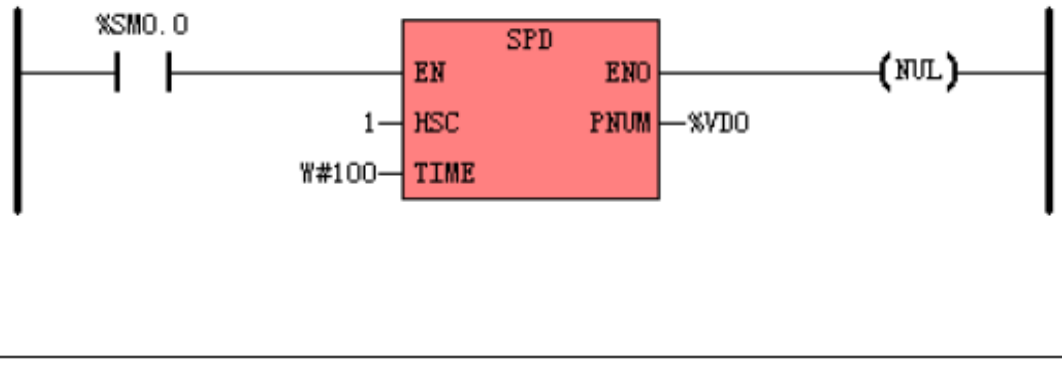
	Name	Usage	Group	
LD	SPD			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	SPD	SPD *HSC, TIME, PNUM*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
HSC	Input	INT	Constant (0-5, the number of a HSC)
TIME	Input	WORD	I, Q, M, V, L, SM, Constant
PNUM	Output	DINT	Q, M, V, L, SM

این دستور تعداد پالس دریافت شده از یک شمارنده سرعت بالای مشخص (که شماره آن در پایه HSC تعیین میشود) را در بازه زمانی تعیین شده در پایه TIME (بر اساس ms میباشد) محاسبه و نتیجه را در PNUM ذخیره میکند.

به مثال زیر توجه نمایید:

LD



IL

```
LD    %SM0.0
SPD   1, W#100, %VD0
```

در مثال بالا با توجه به این که SM0.0همیشه on میباشد ، بنابراین تابع SPD پالس های دریافتی از شمارنده HSC1 در هر ۱۰۰ میلی ثانیه شمارش کرده و نتیجه را در VD0 ذخیره میکند.

8.14 تایمر ها (Timers):

تایمر یکی از فانکشن بلاک هایی است که در استاندارد IEC61131-3 تعریف شده است

و در مجموع بر سه نوع TON , TOF,TP میباشند.

8.14.1 دقت زمانی (Resolutaion) تایمرها :

سه نوع دقت زمانی برای تایمرها وجود دارد که با توجه به شماره تایمر و جدول زیر میتوان آن را تعیین کرد :

	CPU304	CPU306
Resolution	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T63: 100ms	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T127: 100ms
Max timing	32767* Resolution	32767* Resolution

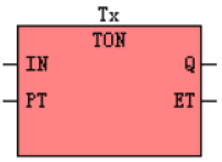
مقدار از پیش تعیین شده و مقداردر حال شمارش نهایی از حاصل ضرب رزولوشن (دقت زمانی) تایمرها د رمقدار PT و ET بدست می آید .

به عبارت دیگر : زمانی که میخواهید مقدار PT را تعیین نمایید باید زمان مورد نظر را بر دقت زمانی تقسیم کرده تا مقدار PT مشخص شود.

PT=(زمان مورد نظر) / دقت تایمر)

به عنوان مثال :مقدار ۱۰۰ در دقت اندازه گیری 1 ms (T0~T3) مقدار ۱۰۰۰ میلی ثانیه را میدهد.

8.14.2 TON(تایمر با تاخیر در روشن شدن ، on-delay timer):

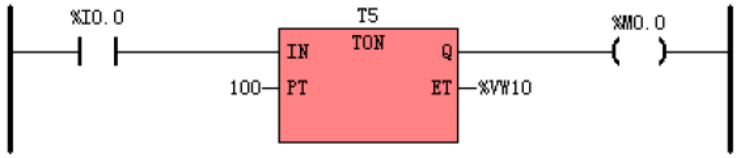
	Name	Usage	Group	
LD	TON			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	TON	TON *Tx, PT*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
Tx	-	Timer instance	T
IN	Input	BOOL	Power flow
PT	Input	INT	I, AI, AQ, M, V, L, SM, constant
Q	Output	BOOL	Power flow
ET	Output	INT	Q, M, V, L, SM, AQ

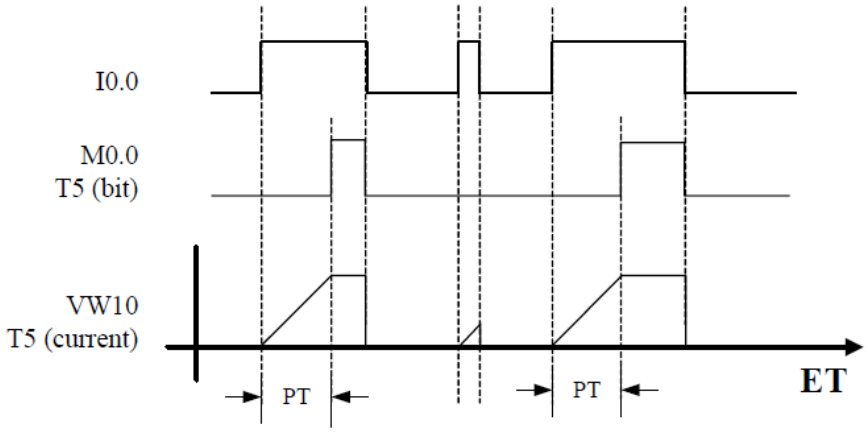
LD: عملکرد این تایمر به این صورت میباشد با فعال شدن پایه ورودی IN (با لبه بالارونده)تایمر Tx فعال میشود .هنگامی که زمان سپری شده ET مساوی یا بزرگتر از PT (مقدار زمان تعیین شده) شود خروجی Q و بیت وضعیت Tx یک خواهد شد .بنا براین تایمر به اندازه "دقت زمانی تایمر مربوطه × PT = میلی ثانیه "در روشن خروجی تاخیر ایجاد میکند.هنگامی که پایه ورودی IN،false(غیر فعال) شود ، بیت وضعیت Tx و نیز خروجی Qبلافاصله ۰ خواهد شد .ضمنا مقدار در حال شمارش نیز صفرخواهد شد .

IL: با فعال شدن CR (با لبه بالارونده) تایمر Tx فعال میشود .هنگامی که زمان سپری شده مساوی یا بزرگتر از PT (مقدار زمان تعیین شده) شود بیت وضعیت Tx یک خواهد شد .هنگامی که CR،false(غیر فعال) شود ، بیت وضعیت Tx بلافاصله ۰ خواهد شد .ضمنا مقدار در حال شمارش نیز صفرخواهد شد . پس از هر سیکل اسکن مقدار CR همان مقداربیت وضعیت Tx خواهد بود .

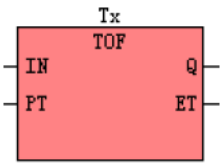
به مثال زیر توجه نمایید:

LD	IL
(* Network 0 *) (* T5 is an instance of TON, and its preset time is 1000ms (100*10ms) *)  	(* NETWORK 0 *) LD %I0.0 TON T5, 100 ST %M0.0

Time Sequence Diagram



TOF:8.14.3(تایمر با تاخیر در خاموش شدن، off-delay timer) :

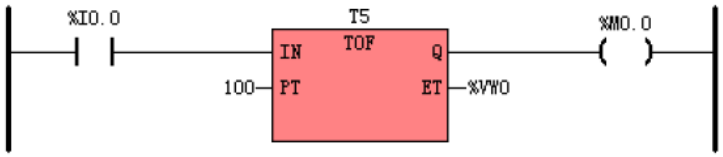
	Name	Usage	Group	
LD	TOF			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	TOF	TOF *Tx, PT*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
Tx	-	Timer instance	T
IN	Input	BOOL	Power flow
PT	Input	INT	I, AI, AQ, M, V, L, SM, constant
Q	Output	BOOL	Power flow
ET	Output	INT	Q, M, V, L, SM, AQ

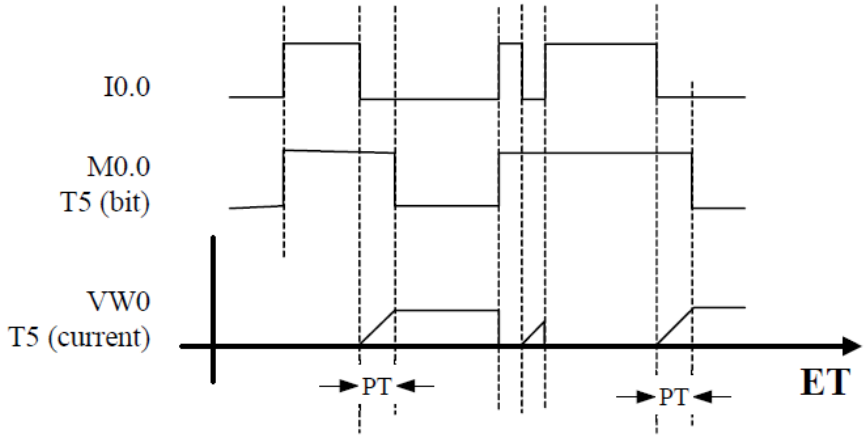
LD: عملکرد این تایمر به این صورت میباشد با غیرفعال شدن پایه ورودی IN (با لبه پایین رونده)تایمر Tx فعال میشود .هنگامی که زمان سپری شده ET مساوی یا بزرگتر از PT (مقدار زمان تعیین شده) شود خروجی Q و بیت وضعیت Tx ، 0 خواهد شد .بنا براین تایمر به اندازه "دقت زمانی تایمر مربوطه × PT = میلی ثانیه "در خاموش شدن خروجی تاخیر ایجاد میکند.هنگامی که پایه ورودی IN،True (فعال) شود ، بیت وضعیت Tx و نیز خروجی Qبلافاصله 1 خواهد شد .ضمنا مقدار در حال شمارش نیز صفرخواهد شد .

IL: با غیر فعال شدن CR (با لبه پایین رونده) تایمر Tx فعال میشود .هنگامی که زمان سپری شده مساوی یا بزرگتر از PT (مقدار زمان تعیین شده) شود بیت وضعیت Tx ، ۰ خواهد شد .هنگامی که CR،TRUE(فعال) شود ، بیت وضعیت Tx بلافاصله ،۱ خواهد شد .ضمنا مقدار در حال شمارش نیز صفرخواهد شد .پس از هر سیکل اسکن مقدار CR همان مقداربیت وضعیت Tx خواهد بود .

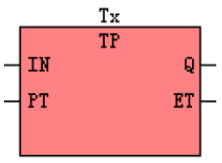
به مثال زیر توجه نمایید:

LD	IL
(* Network 0 *) (* T5 is an instance of TOF, and its preset time is 1000ms (100*10ms) *)  	(* NETWORK 0 *) LD %I0.0 TOF T5, 100 ST %M0.0

Time Sequence Diagram



8.14.4 TP (Pulse Timer):

	Name	Usage	Group	
LD	TP			☑ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308
IL	TP	TP *Tx, PT*	P	

Operands	Input/Output	Data Type	Acceptable Memory Areas
Tx	-	Timer instance	T
IN	Input	BOOL	Power flow
PT	Input	INT	I, AI, AQ, M, V, L, SM, constant
Q	Output	BOOL	Power flow
ET	Output	INT	Q, M, V, L, SM, AQ

LD:عملکرد این تایمر به این صورت میباشد که با لبه بالارنده پایه ورودی IN تایمر مربوطه شروع به شمارش میکند و خروجی Q و بیت وضعیت Tx نیز یک (True) میگردد .

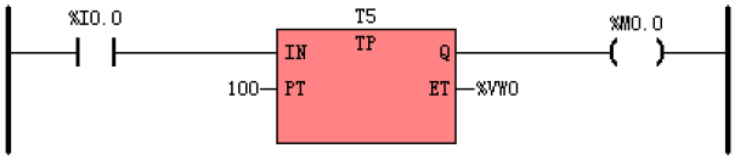
خروجی Q و بیت وضعیت Tx به مدت زمانی "دقت زمانی تایمر مربوطه × PT = میلی ثانیه" در وضعیت TRUE(یک) باقی میماند و به محض آنکه مقدار ETبه مقدار PT رسید خروجی و بیت وضعیت Tx ، false(صفر) میگردد.

IL: با لبه بالارنده CR تایمر مربوطه شروع به شمارش میکند و بیت وضعیت Tx نیز یک (True) میگردد .

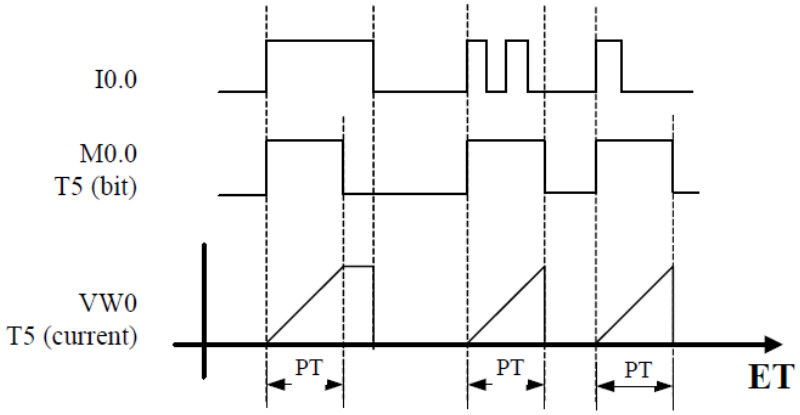
بیت وضعیت Tx به مدت زمانی "دقت زمانی تایمر مربوطه × PT = میلی ثانیه" در وضعیت TRUE(یک) باقی میماند و به محض آنکه مقدار ETبه مقدار PT رسید ، بیت وضعیت Tx ، false(صفر) میگردد. پس از هر سیکل اسکن مقدار CR همان مقداریبت وضعیت Tx خواهد بود .

به مثال زیر توجه نمایید:

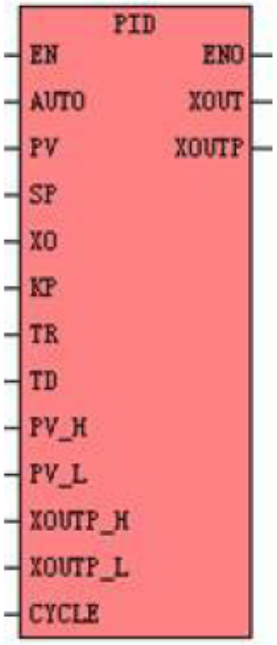
LD	IL
(* Network 0 *) (* T5 is an instance of TP, and its preset time is 1000ms (100*10ms) *)	(* NETWORK 0 *) LD %I0.0 TP T5, 100 ST %M0.0



Time Sequence Diagram



:PID

	Name	Usage	Group	
LD	PID			☐ CPU304 ☑ CPU304EX ☑ CPU306 ☑ CPU306EX ☑ CPU308 ☑ CPU406 ☑ CPU408
IL	PID	PID AUTO, PV, SP, XO, KP, TR, TD, PV_H, PV_L, XOUTP_H, XOUTP_L, CYCLE, XOUT, XOUTP	U	

Operands	IN/OUT	Data Type	Memory Areas	Comment
AUTO	INPUT	BOOL	I,Q,V,M,SM,L,T,C	Manual/Auto. 0=Manual, 1=Auto.
PV	INPUT	INT	AI,V,M,L	Process Variable
SP	INPUT	INT	V,M,L	Setpoint
XO	INPUT	REAL	V,L	Manual value, range [0.0, 1.0]
KP	INPUT	REAL	V,L	Proportionality constant
TR	INPUT	REAL	V,L	Reset time, which determines the time response of the integrator. (Unit: s)
TD	INPUT	REAL	V,L	Derivative time, which determines the time response of the derivative unit. (Unit: s)
PV_H	INPUT	INT	V,L	The upper limit value of PV
PV_L	INPUT	INT	V,L	The lower limit value of PV
XOUTP_H	INPUT	INT	V,L	The upper limit value of XOUTP
XOUTP_L	INPUT	INT	V,L	The lower limit value of XOUTP
CYCLE	INPUT	DINT	V,M,L	Sampling period. (Unit: ms)
XOUT	OUTPUT	REAL:	V,L	Manipulated Value, range [0.0, 1.0].
XOUTP	OUTPUT	INT	AQ,V,M,L	Manipulated Value Peripheral. This value is the normalizing result of XOUT.

میتوان در یک CPU، از ۸ حلقه PID استفاده نمود.

LD: چنانچه EN، ۱باشد این دستور اجرا میشود.

IL: چنانچه CR، ۱باشد این دستور اجرا میشود.این دستور تاثیری بر مقدار CR نخواهد داشت .

:MANUAL/AUTO

میتوان از دردستور PID ، در دو مد کاری دستی (MANUAL) و اتوماتیک (AUTO) استفاده نمود.

چنانچه ورودی AUTO، ۰ باشد ،PID در مد دستی (MANUAL) بوده و مقدار ورودی XO مستقیما به عنوان خروجی XOUTقرار میگیرد.

چنانچه ورودی AUTO، ۱ باشد ،PID در مد اتوماتیک (AUTOMATIC) بوده ،سپس محاسبات PID براساس ورودی ها انجام شده و نتیجه نهایی XOUTتنظیم میگردد.

همچنین میتوان به وسیله ورودیAUTO بین مد AUTOو MANUALسوئیچ کرد.

نرمالیزه کردن PV وSP :

PV وSP به عنوان ورودی با فرمت INT میباشند.در الگوریتم PID ، به مقادیر با فرمت اعشاری بین 0.0 و 1.0 نیاز میباشد .بنابراین باید عملیات نرمالیزه کردن انجام شود.

KINCO-K3به صورت اتوماتیک و براساس PV,SP,PV-H ,PV-L عملیات نرمالیزه کردن را انجام میدهد.

نرمالیزه کردن به صورت زیر میباشد:

مقدار نرمالیزه K*PV+b=PV

مقدار نرمالیزه SP =K*SP+b

به عنوان مثال : شما میخواهید فشار را در مقدار مورد نظر 25Mpa کنترل نمایید.در این پروسه از یک ترانسمیتر فشار استفاده میگردد که فشار در بازه 0~40Mpa را اندازه گیری کرده و خروجی آن جریان 4~20mA میباشد .خروجی این ترانسمیتر به کانال شماره ۱ از ماژول ورودی آنالوگ متصل میگردد.

میتوان ورودی های PID را مطابق جدول زیرتنظیم نمود :

<1cmn lang="EN-US">	Actual Parameter	Comment
PV	AIW0	AIW0 can be set as PV because of their linear relation.
SP	14000	14mA. Because 14mA means the real pressure value 25MPa.
PV_L	4000	The lower limit value of the transformer's output
PV_H	20000	The upper limit value of the transformer's output

مقادیر خروجی:

مقادیر خروجی در PID، XOUTو XOUTP میباشد.

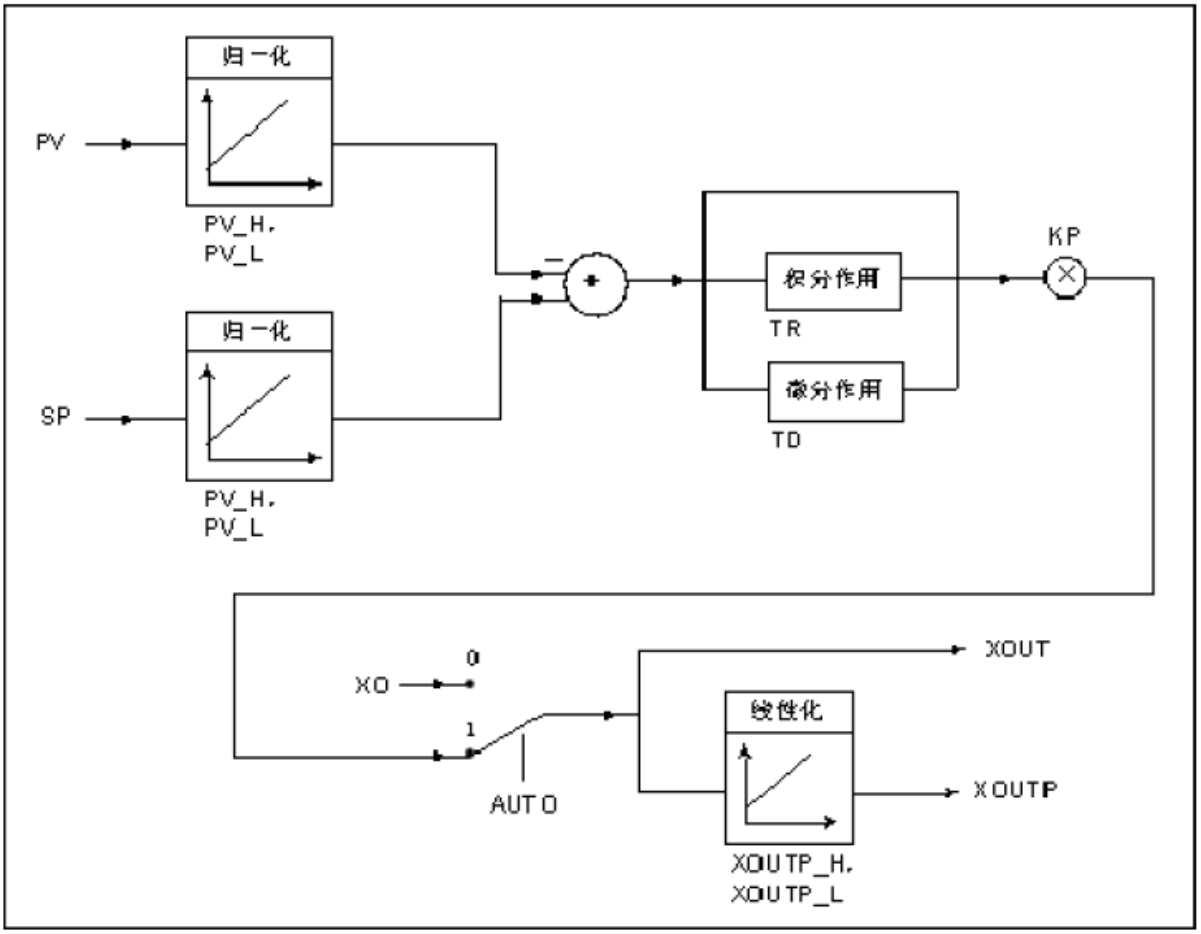
XOUT یک مقدار بین 0.0و1.0 بوده (که نمایان گر مقدار 0.0% و100% میباشد)

XOUTP یک مقدار INT بوده و نتیجه نرمالیزه شدن XOUTبر اساس XOUTP-H و XOUTP-L میباشد.

XOUTP=(XOUTP_H – XOUTP_L)* XOUT+XOUTP_L

این فرمت برای ارسال XOUTP به کانال خروجی آنالوگ (AO) مناسب میباشد .

دیاگرام PID به صورت زیر میباشد:


归一化
PV
PV_H.
PV_L
归一化
SP
PV_H.
PV_L
积分作用
TR
微分作用
TD
KP
XO
0
1
AUTO
XOUT
线性化
XOUTP
XOUTP_H.
XOUTP_L

IL

```
(* Network 0 *)

(* At first, enter the actual parameters *)

LD      %SM0.0

MOVE    7200, %VW0      (* SP *)



MOVE    4000, %VW2      (* PV_L *)

MOVE    20000, %VW4     (* PV_H *)



MOVE    4000, %VW6      (* XOUTP_L *)

MOVE    20000, %VW8     (* XOUTP_H *)



(* Network 1 *)

(* Execute PID *)

LD      %SM0.0

PID     %M0.0, %AIW0, %VW0, %VR100, %VR104, %VR108, %VR112, %VW2, %VW4, %VW6,
%VW8, %VD10, %VR116, %AQW0
```

8.16: کنترل موقعیت (Position control):

PLC های KINCO دارای دو کانال خروجی پالس سرعت بالا (Q0.0 و Q0.1) میباشند، که میتوان از این دو خروجی جهت کنترل موقعیت دو محور استفاده نمود.

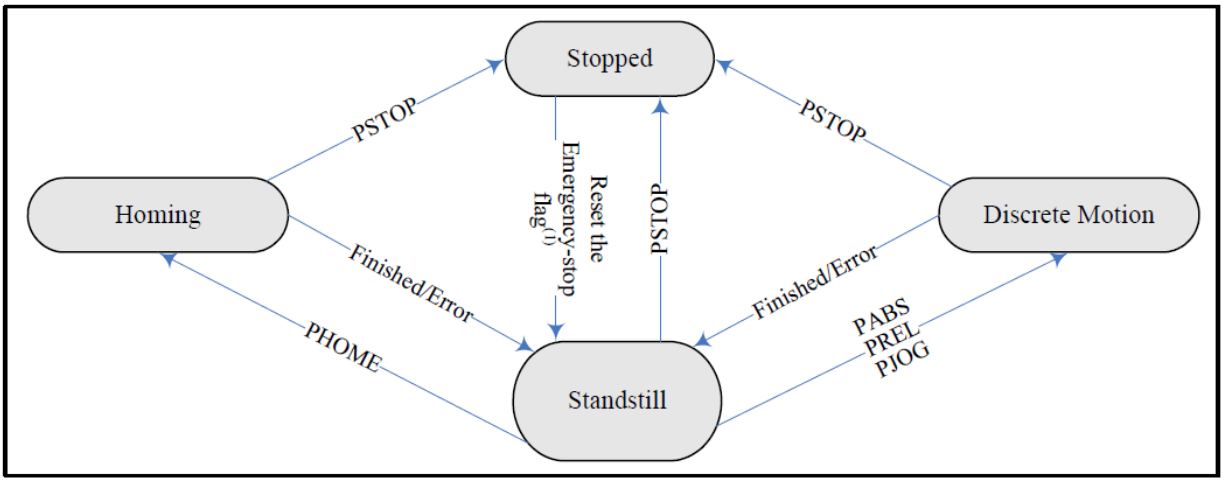
در قسمت قبل (6.13.3) دستورات مربوط به خروجی پالس سرعت بالا ، استفاده از PTO/PWM و نیز دستور PLS به صورت کامل توضیح داده شده است .در این قسمت دستورات مربوط به کنترل موقعیت (Position Control) که کاربرد دیگری از عملکرد خروجی پالس سرعت بالا میباشد توضیح داده شده است .

در مقایسه با دستور PLS، دستورات Position Control برای کنترل نمودن موقعیت مناسب تر میباشد .

در این دستورات نیز مشابه با دستور PLS فرکانس پالس خروجی ماکزیمم 20KHZ میباشد.

دیاگرام زیر برمبنای یک محور میباشد و عملکرد و رفتار محور را در بالاترین سطح در زمانی که دستورات کنترل موقعیت فعال میباشد ، نشان میدهد.

محور همیشه در یکی از موقعیت های مشخص شده میباشد (مطابق دیاگرام پایین).



رجیسترهای SM201.7/SM231.7 بیتهای emergency –stop هستند .زمانی که دستور PSTOP اجرا میشود این بیتها به صورت اتوماتیک ۱ خواهد شد .در ادامه توضیحات بیشتر ارائه خواهد شد .

8.16.1 متغیر های وابسته:

8.16.1.1 کانال خروجی جهت :

برای دستورات کنترل موقعیت در PLCهای KINCO-K3، برای هر کانال خروجی پالس (پالس سرعت بالا) یک کانال خروجی جهت و نیز یک بیت کنترلی در فضای حافظه SM به منظور فعال سازی جهت خروجی در نظر گرفته شده است .

به جدول زیر توجه نمایید:

High-speed Pulse Output Channel	Q0.0	Q0.1
Direction output channel	Q0.2	Q0.3
Direction control bit	SM201.3	SM231.3

کانال جهت به منظور ایجاد کردن یک سیگنال جهت به کار میرود.(این سیگنال جهت به منظور کنترل جهت موتورهای الکتریکی استفاده میشود)، ۰ به معنی چرخش به سمت جلو و ۱ به معنی چرخش به سمت عقب میباشد .

بیت کنترل جهت به منظور فعال و با غیر فعال کردن کانال خروجی جهت مربوطه استفاده میشود .

بیت کنترلی جهت از بالاترین اولویت برخوردار میباشد ، چنانچه غیر فعال باشد ، زمان اجرای یک دستور کنترل موقعیت ، سیگنال جهتی ایجاد نمیشود و کانال خروجی جهت مربوط به آن میتواند به عنوان یک خروجی دیجیتال معمولی (Digital output) استفاده میگردد.

8.16.1.2: رجیسترهای کنترلی و رجیسترهای وضعیت:

Kinco-k3برای دستورات کنترل موقعیت ،برای هر کانال خروجی سرعت بالا یک بایت کنترلی در نظر گرفته است . این بایت کنترلی به منظور ذخیره سازی پیکربندی این کانال های خروجی سرعت بالا میباشد.

رجیسترهای وضعیت نیز برای ذخیره مقدار جاری(تعداد پالس های خروجی که به صورت یک داده DINT میباشد) در نظر گرفته شده است .در چرخش به سمت جلو مقدار جاری افزایش یافته و در چرخش به سمت عقب این مقدار کاهش میابد.

جدول زیر تمامی این رجیسترها را با جزئیات توضیح میدهد.

توجه داشته باشید زمانی که دستور کنترل موقعیت پایان میابد مقدار جاری به صورت اتوماتیک پاک نخواهد شد و کاربر باید آن را در برنامه پاک نماید .

جدول زیر بایت کنترلی و نیز مقدار جاری را توضیح میدهد:

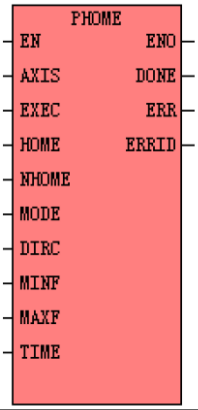
Q0.0	Q0.1	Description
SM201.7	SM231.7	Emergency-Stop flag. If this bit is 1, no position control instructions can be executed. When executing the PSTOP instruction, this bit is set to 1 automatically, and it must be reset by your program.
SM201.0~SM201.2	SM201.0~SM201.2	Reserved
SM201.3	SM231.3	Direction control bit. 1 --- Disable the direction output channel. 0 --- Enable the direction output channel.
SM201.0~SM201.2	SM201.0~SM201.2	Reserved
Q0.0	**Q0.1**	**Description**
SMD212	SMD242	The current value

8.16.1.3 error identification(شناسایی خطا) :

در زمان اجرای دستورات کنترل موقعیت ممکن است خطاهایی ایجاد گردد. در این حالت CPU خطا را شناسایی کرده و خطا را در پارامتر ERRID دستور مینویسد.جدول زیر این کدهای خطا را توضیح میدهد:

Error Code	Description
0	No error
1	The value of *AXIS* is not 0 or 1.
2	The value of *MINF* is larger than the value of *MAXF*.
3	The value of *MINF* is less than the allowed lowest frequency (20Hz).
4	The value of *TIME* (accelerating / decelerating time) doesn't match the value of *MINF* and *MAXF*.

8.16.2 :PHOME

	Name	Usage	Group	
LD	PHOME			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PHOME	PHOME *AXIS*, *EXEC*, *HOME*, *NHOME*, *MODE*, *DIRC*, *MINF*, *MAXF*, *TIME*, *DONE*, *ERR*, *ERRID*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
AXIS	Input	INT	Constant (0 or 1)
EXEC	Input	BOOL	I, Q, V, M, L, SM, RS, SR
HOME	Input	BOOL	I, Q, V, M, L, SM, RS, SR
NHOME	Input	BOOL	I, Q, V, M, L, SM, RS, SR
MODE	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constant
DIRC	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constant
MINF	Input	WORD	I, Q, M, V, L, SM, Constant
MAXF	Input	WORD	I, Q, M, V, L, SM, Constant
TIME	Input	WORD	I, Q, M, V, L, SM, Constant

DONE	Output	BOOL	Q, M, V, L, SM
ERR	Output	BOOL	Q, M, V, L, SM
ERRID	Output	BYTE	Q, M, V, L, SM

جدول زیر تمامی عملوندهای این دستور را به صورت کامل توضیح میدهد:

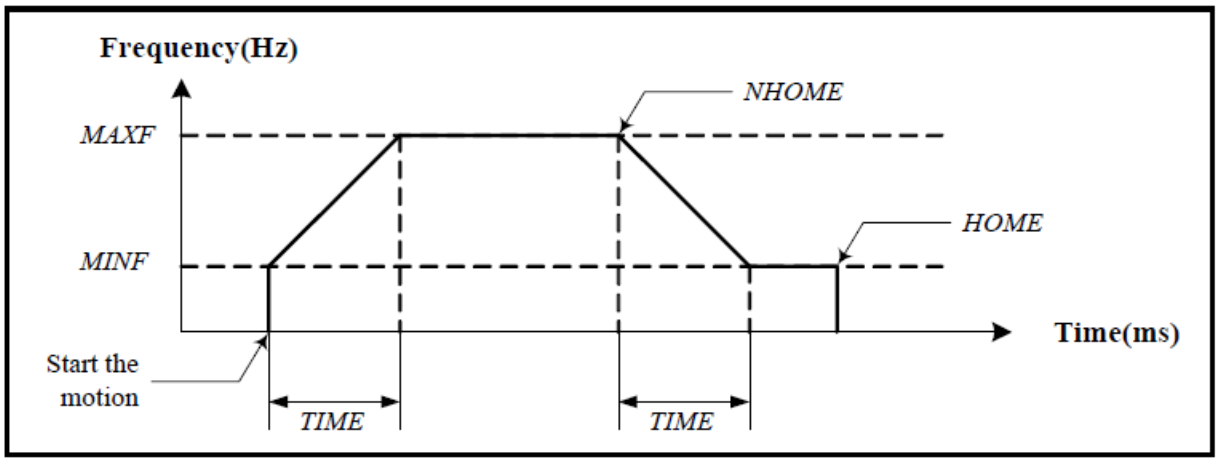
Operands	Description
AXIS	The high-speed output channel, 0 means Q0.0, 1 means Q0.1.
EXEC	If *EN* is 1, the *EXEC* starts the 'search home' motion on the rising edge.
HOME	The home signal from the home sensor
NHOME	The near home signal from the near home sensor
MODE	Specifies the homing mode: 0 means that the home signal and the near home signal are all used; 1 means that only the home signal is used.
DIRC	Specifies the rotating direction of the electric motor: 0 means rotating forwards; 1 means rotating backwards. Please refer to 6.16.2.1 The direction output channel for more information.
MINF	Specifies the initial speed (i.e., the initial frequency) of the pulse train output. Unit: Hz. Note: the value of *MINF* must be equal to or less than 2KHz.
MAXF	Specifies the highest speed (i.e., the highest frequency) of the pulse train output. Unit: Hz. The available range of *MAXF* is 20Hz ~ 20KHz. *MAXF* must be larger than or equal to *MINF*.
TIME	Specifies the acceleration/deceleration time. Unit: ms. In the position control instructions, the acceleration time is the same as the deceleration time. The acceleration time is the time for the speed accelerating from *MINF* to *MAXF*. The deceleration time is the time for the speed decelerating from *MAXF* to *MINF*.
DONE	Indicates that the instruction has finished successfully. 0 = not finished; 1 = finished.
ERR	Indicates that error has occurred during the execution. 0 = no error; 1 = an error has occured.
ERRID	Error identification. If the ERR is 1, the ERRID describes the error's detailed information. Please refer to 6.16.2.3 The error identification .

این دستور محور را با استفاده از دو سیگنال NHOME و HOME کنترل مینماید.

پایه MODEمشخص کنند homin mode میباشد. هنگامی که عملیات search home اجرا میگردد چنانچه DIRC به صورت ۰ تنظیم شده باشد (چرخش به سمت جلو) مقدار جاری یعنی مقدار رجیسترهای SMD212/SMD242

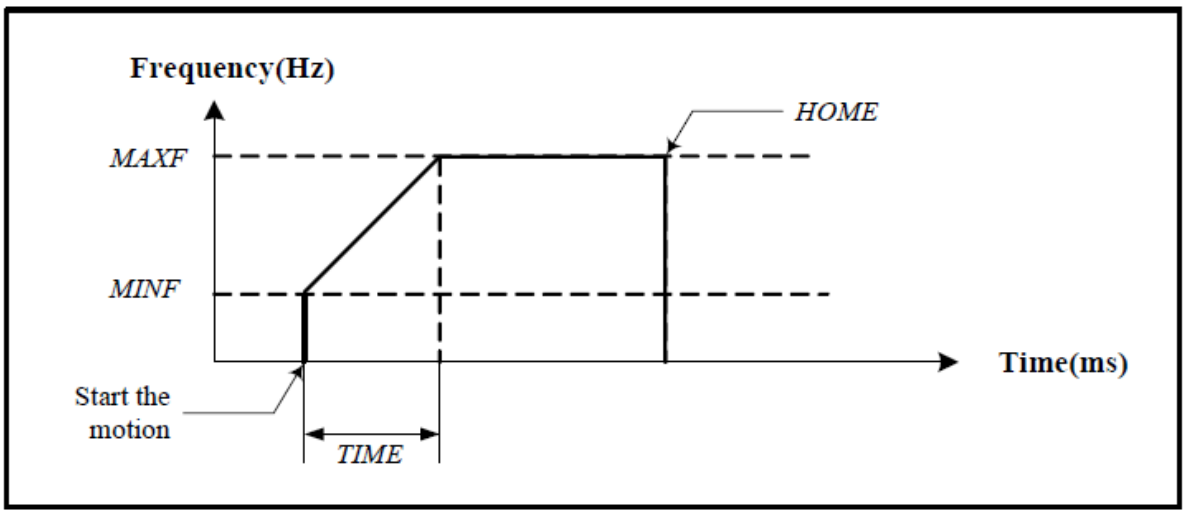
افزایش میابد . در صورتی که DIRC به صورت ۱ تنظیم شده باشد (چرخش به سمت عقب) مقدار جاری (SMD212/SMD242) کاهش خواهد یافت .

چنان چه پایه MODE ، ۰ باشد از هر دوسیگنال HOMEو NHOME استفاده میشود.به محض یک شدن سیگنال NHOME دستور PHOME محور را به سمت کاهش سرعت کنترل میکند و به محض آن که سیگنال HOME یک شود سرعت ۰ خواهد شد .دیاگرام زمانی در این حالت به صورت زیر میباشد:



چنان چه پایه MODE، ۱ باشد تنها از سیگنال HOME استفاده میگردد .در این حالت به محض آن که سیگنال HOME یک شود محور از حرکت می ایستد (سرعت ۰ میگردد)

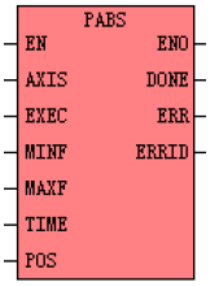
دیاگرام زمانی در این حالت به صورت زیر میباشد:



LD: چنانچه EN یک باشد این دستور اجرا خواهد شد .

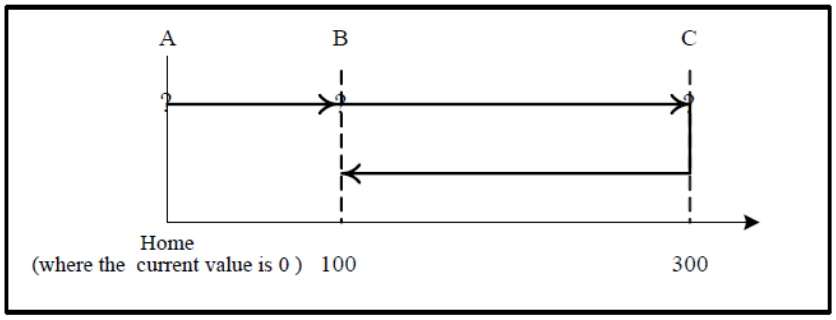
IL: چنانچه CR یک باشد این دستور اجرا خواهد شد . این دستور تاثیری بر مقدار CR نخواهد داشت .

8.16.3: PABS (حرکت مطلق):

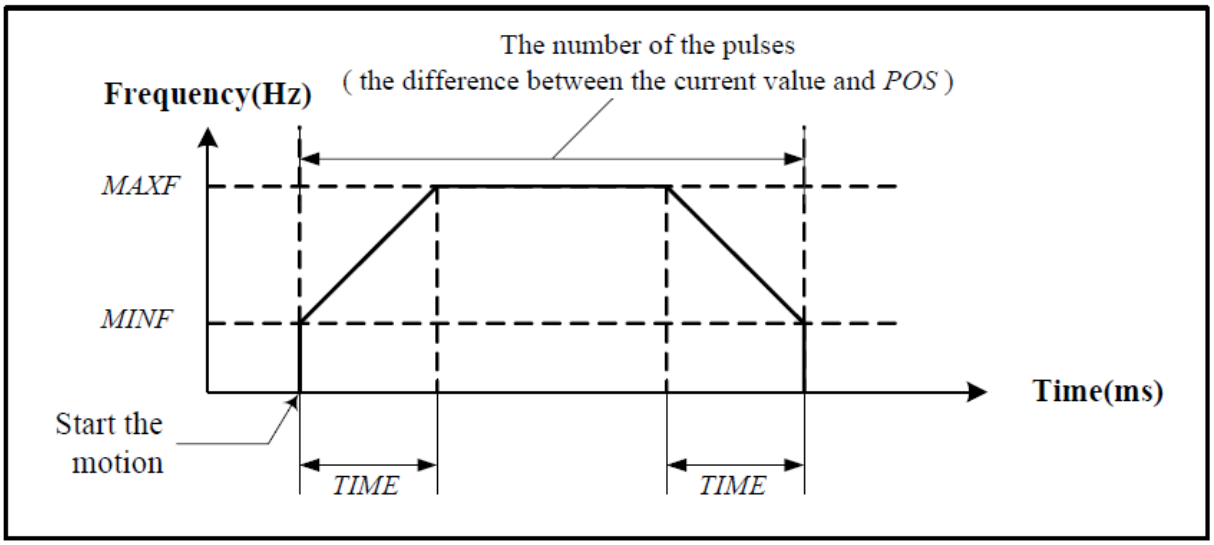
	Name	Usage	Group	
LD	PABS			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PABS	PABS *AXIS*, *EXEC*, *MINF*, *MAXF*, *TIME*, *POS*, *DONE*, *ERR*, *ERRID*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
AXIS	Input	INT	Constant (0 or 1)
EXEC	Input	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Input	WORD	I, Q, M, V, L, SM, Constant
MAXF	Input	WORD	I, Q, M, V, L, SM, Constant
TIME	Input	WORD	I, Q, M, V, L, SM, Constant
POS	Input	DINT	I, Q, M, V, L, SM, HC, Constant
DONE	Output	BOOL	Q, M, V, L, SM
ERR	Output	BOOL	Q, M, V, L, SM
ERRID	Output	BYTE	Q, M, V, L, SM

جدول زیر تمامی عملوند ها را به صورت کامل توضیح میدهد:

Operands	Description
AXIS	The high-speed output channel, 0 means Q0.0, 1 means Q0.1.
EXEC	If *EN* is 1, the *EXEC* starts the absolute motion on the rising edge.
MINF	Specifies the initial speed (i.e., the initial frequency) of the pulse train output. Unit: Hz. Note: the value of *MINF* must be equal to or less than 2KHz.
MAXF	Specifies the highest speed (i.e., the highest frequency) of the pulse train output. Unit: Hz. The available range of *MAXF* is 20Hz ~ 20KHz. *MAXF* must be larger than or equal to *MINF*.
TIME	Specifies the acceleration/deceleration time. Unit: ms. In the position control instructions, the acceleration time is the same as the deceleration time. The acceleration time is the time for the speed accelerating from *MINF* to *MAXF*. The deceleration time is the time for the speed decelerating from *MAXF* to *MINF*.
POS	Specifies the target value. It is represented with the number of pulses between the home positon, where the current value is 0, and the target position. As shown in the following figure, if the object is moved from A to B, the *POS* should be set as '100'; If it is moved from B to C, the *POS* should be set as '300'; If it is moved from C to B, the *POS* should be set as '100'. 
DONE	Indicates that the instruction has finished successfully. 0 = not finished; 1 = finished.
ERR	Indicates that error has occurred during the execution. 0 = no error; 1 = an error has occured.
ERRID	Error identification. If the ERR is 1, the ERRID describes the error's detailed information. Please refer to 6.16.2.3 The error identification .

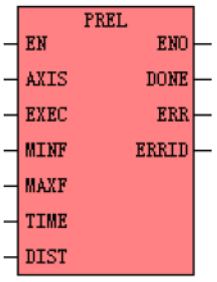
این دستور یک محور (AXIS) را به منظور حرکت به سمت یک موقعیت مشخص (POS) کنترل می نماید .چنانچه بیت کنترل جهت (SM201.3 /SM231.3) ۰ باشد دستور PABS سیگنال خروجی جهت را در کانال خروجی جهت مربوطه (Q0.2 /Q0.3) ایجاد مینماید .چنانچه مقدار نهایی از مقدار جاری بزرگ تر باشد ، خروجی جهت گردش به جلو ایجاد میگرد .سپس مقدار جاری (رجیستر های SMD212/SMD242) افزایش میابد. چنان چه مقدار نهایی از مقدار جاری کوچکتر باشد خروجی جهت گردش به عقب ایجاد میگردد، سپس مقدار جاری کاهش میابد. دیاگرام زمانی به صورت زیر خواهد بود :



LD: چنانچه EN یک باشد این دستور اجرا خواهد شد .

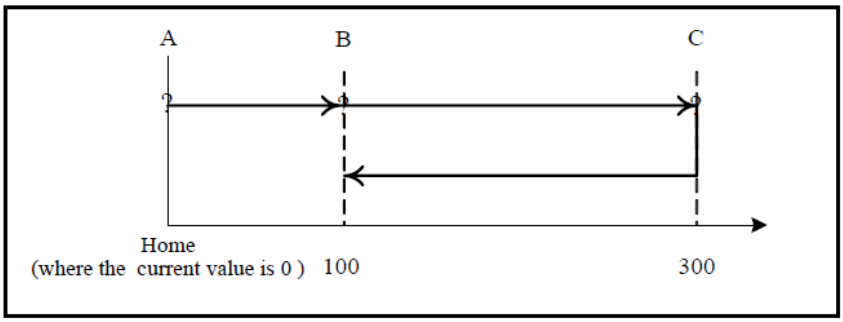
IL: چنانچه CR یک باشد این دستور اجرا خواهد شد . این دستور تاثیری بر مقدار CR نخواهد داشت .

8.16.4: PREL (حرکت نسبی):

	Name	Usage	Group	
LD	PREL			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PREL	PREL *AXIS*, *EXEC*, *MINF*, *MAXF*, *TIME*, *DIST*, *DONE*, *ERR*, *ERRID*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
AXIS	Input	INT	Constant (0 or 1)
EXEC	Input	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Input	WORD	I, Q, M, V, L, SM, Constant
MAXF	Input	WORD	I, Q, M, V, L, SM, Constant
TIME	Input	WORD	I, Q, M, V, L, SM, Constant
DIST	Input	DINT	I, Q, M, V, L, SM, HC, Constant
DONE	Output	BOOL	Q, M, V, L, SM
ERR	Output	BOOL	Q, M, V, L, SM
ERRID	Output	BYTE	Q, M, V, L, SM

جدول زیر تمامی عملوند ها را به صورت کامل توضیح میدهد:

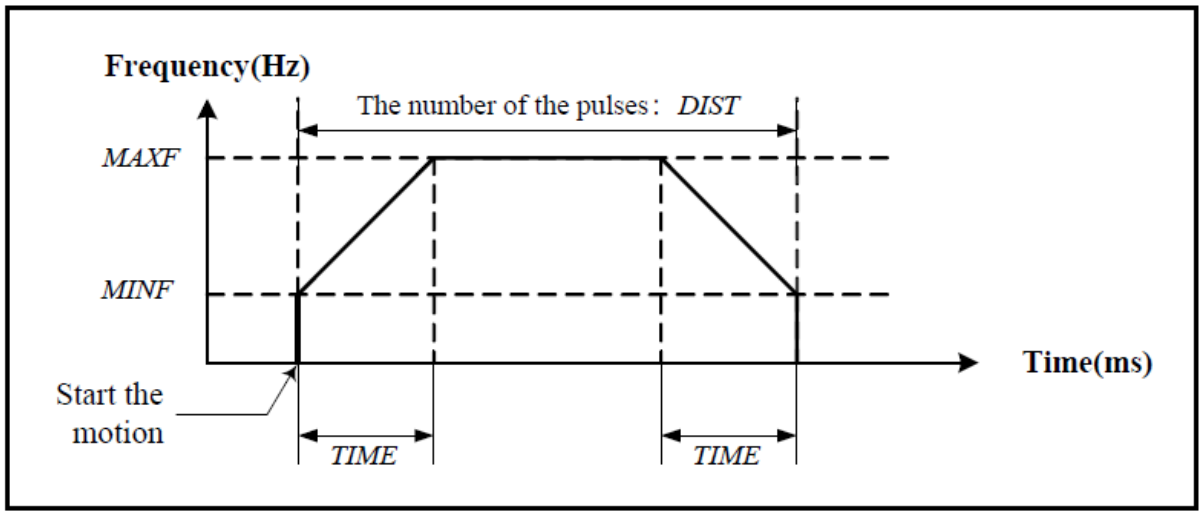
Operands	Description
AXIS	The high-speed output channel, 0 means Q0.0, 1 means Q0.1.
EXEC	If *EN* is 1, the *EXEC* starts the relative motion on the rising edge.
MINF	Specifies the initial speed (i.e., the initial frequency) of the pulse train output. Unit: Hz. Note: the value of *MINF* must be equal to or less than 2KHz.
MAXF	Specifies the highest speed (i.e., the highest frequency) of the pulse train output. Unit: Hz. The available range of *MAXF* is 20Hz ~ 20KHz. *MAXF* must be larger than or equal to *MINF*.
TIME	Specifies the acceleration/deceleration time. Unit: ms. In the position control instructions, the acceleration time is the same as the deceleration time. The acceleration time is the time for the speed accelerating from *MINF* to *MAXF*. The deceleration time is the time for the speed decelerating from *MAXF* to *MINF*.
DIST	Specifies the target distance. It is represented with the number of pulses between the current positon and the target position. As shown in the following figure, if the object is moved from A to B, the *DIST* should be set as '100'; If it is moved from B to C, the *DIST* should be set as '200'; If it is moved from C to B, the *DIST* should be set as '-200'. 
DONE	Indicates that the instruction has finished successfully. 0 = not finished; 1 = finished.
ERR	Indicates that error has occurred during the execution. 0 = no error; 1 = an error has occured.
ERRID	Error identification. If the ERR is 1, the ERRID describes the error's detailed information. Please refer to 6.16.2.3 The error identification .

این دستور محور(AXIS) را برای اجرای یک حرکت با فاصله مشخص که به مقدار جاری در زمان اجرا وابسته است ، کنترل مینماید .

در صورتی که بیت کنترل جهت (SM201.3/SM231.3) به صورت ۰ تنظیم شده باشد دستور PREL سیگنال خروجی جهت را درکانال خروجی جهت مربوطه (Q0.2/Q0.3) تولید مینماید.

چنانچه DISTمثبت باشد خروجی جهت چرخش به سمت جلو ایجاد میگردد ، سپس مقدار جاری (رجیسترهای SMD212 /SMD242) افزایش میابد . در صورتی که DIST منفی باشد ، خروجی جهت چرخش به سمت عقب تولید شده ، سپس مقدار جاری کاهش میابد.

به دیاگرام زمانی زیر توجه نمایید:



LD: چنانچه EN یک باشد این دستور اجرا خواهد شد .

IL: چنانچه CR یک باشد این دستور اجرا خواهد شد . این دستور تاثیری بر مقدار CR نخواهد داشت .

8.16.5: PJOG

	Name	Usage	Group	
LD	PJOG	PJOG EN ENO AXIS DONE EXEC ERR MINF ERRID DIRC		☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PJOG	PJOG *AXIS, EXEC, MINF, DIRC, DONE, ERR, ERRID*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
AXIS	Input	INT	Constant (0 or 1)
EXEC	Input	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Input	WORD	I, Q, M, V, L, SM, Constant
DIRC	Input	INT	I, Q, M, V, L, SM, AI, AQ, T, C, Constant
DONE	Output	BOOL	Q, M, V, L, SM
ERR	Output	BOOL	Q, M, V, L, SM
ERRID	Output	BYTE	Q, M, V, L, SM

جدول زیر تمامی عملوند ها را به صورت کامل توضیح میدهد:

Operands	Description
AXIS	The high-speed output channel, 0 means Q0.0, 1 means Q0.1.
EXEC	If *EN* is 1, the *EXEC* starts the jog motion on the rising edge.
MINF	Specifies the speed (i.e., the initial frequency) of the pulse train output. Unit: Hz.
DIRC	Specifies the the direction of the electric motors: 0 means rotating forwards, and 1 means rotating backwards.
DONE	Indicates that the instruction has finished successfully. 0 = not finished; 1 = finished.
ERR	Indicates that error has occurred during the execution. 0 = no error; 1 = an error has occured.
ERRID	Error identification. If the ERR is 1, the ERRID describes the error's detailed information. Please refer to 6.16.2.3 The error identification .

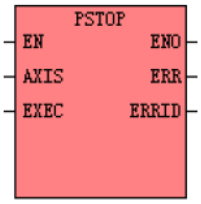
این دستورر محور (Axis) را برای اجرای یک حرکت آهسته کنترل مینماید (با ایجاد خروجی قطار پالس پشت سرهم که فرکانس آن به اندازه MINF میباشد) .

در صورتی که بیت کنترل جهت (SM201.3/SM231.3) به صورت ۰ تنظیم شده باشد دستور PJOGسیگنال خروجی جهت را در کانال خروجی جهت مربوطه (Q0.2/Q0.3) تولید میکند. چنانچه DIRC ، ۰ باشد (چرخش به سمت جلو) مقدار جاری (SMD212/SMD242) افزایش یافته و چنانچه DIRC ، ۱ باشد (چرخش به سمت عقب) مقدار جاری کاهش میابد.

LD: چنانچه EN یک باشد این دستور اجرا خواهد شد .

IL: چنانچه CR یک باشد این دستور اجرا خواهد شد . این دستور تاثیری بر مقدار CR نخواهد داشت .

8.16.6 PSTOP (STOP):

	Name	Usage	Group	
LD	PSTOP			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	PSTOP	PSTOP *AXIS, EXEC, ERR, ERRID*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
AXIS	Input	INT	Constant (0 or 1)
EXEC	Input	BOOL	I, Q, V, M, L, SM, RS, SR
ERR	Output	BOOL	Q, M, V, L, SM
ERRID	Output	BYTE	Q, M, V, L, SM

جدول زیر تمامی عملوند ها را به صورت کامل توضیح میدهد:

Operands	Description
AXIS	The high-speed output channel, 0 means Q0.0, 1 means Q0.1.
EXEC	If *EN* is 1, the *EXEC* stops the current motion on the rising edge.
ERR	Indicates that error has occurred during the execution. 0 = no error; 1 = an error has occured.
ERRID	Error identification. If the ERR is 1, the ERRID describes the error's detailed information.

	Please refer to 6.16.2.3 The error identification .

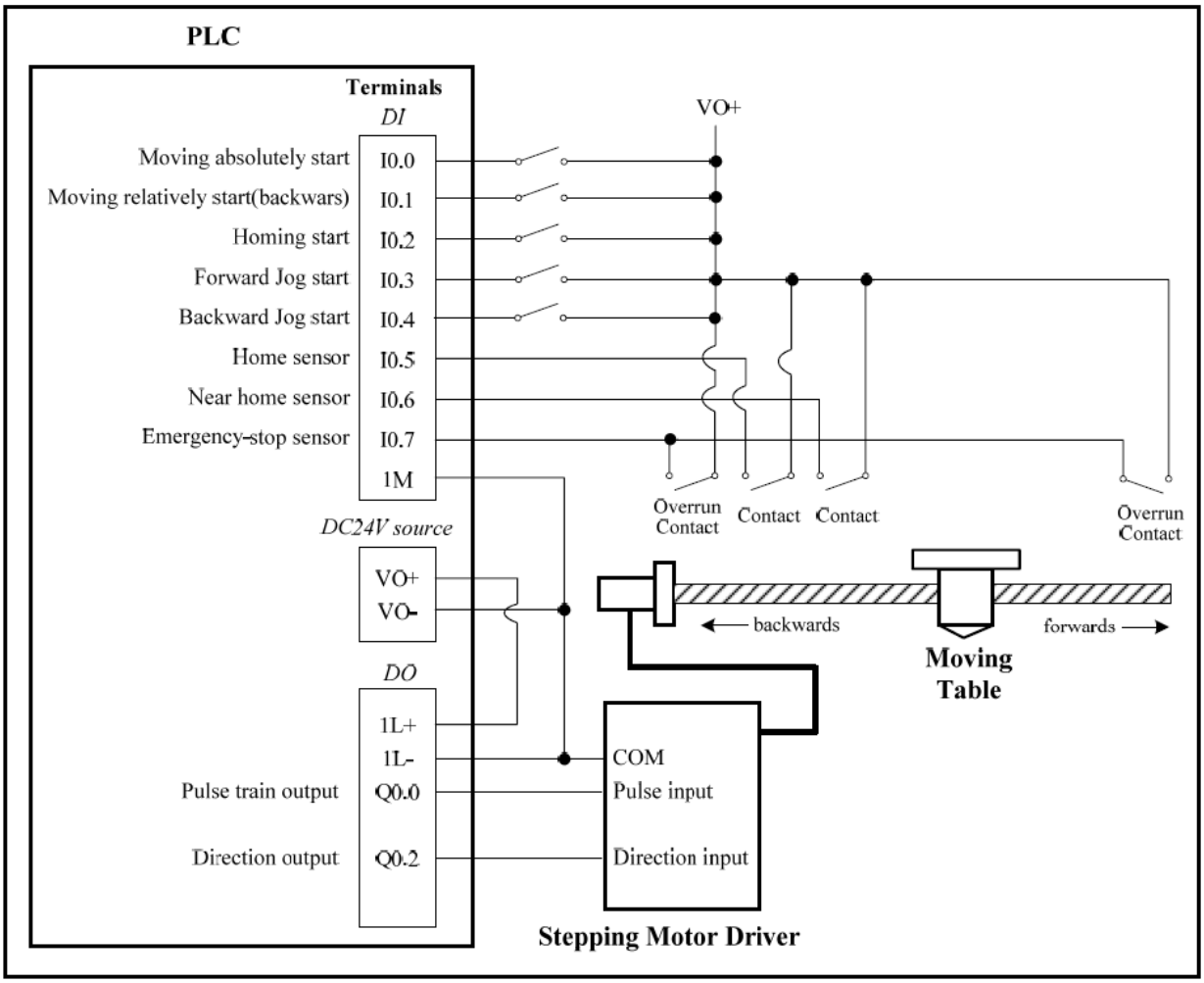
این دستور حرکت محور را متوقف میکند . در همان لحظه فلگ توقف اضطراری (Emergency-stop flag) که همان SM201.7/SM231.7 میباشد ، ۱ میشود .تازمانی که این بیت ها در برنامه reset نشوند هیچ دستور کنترل موقعیت دیگری نمیتواند اجرا گردد.

LD: چنانچه EN یک باشد این دستور اجرا خواهد شد .

IL: چنانچه CR یک باشد این دستور اجرا خواهد شد . این دستور تاثیری بر مقدار CR نخواهد داشت .

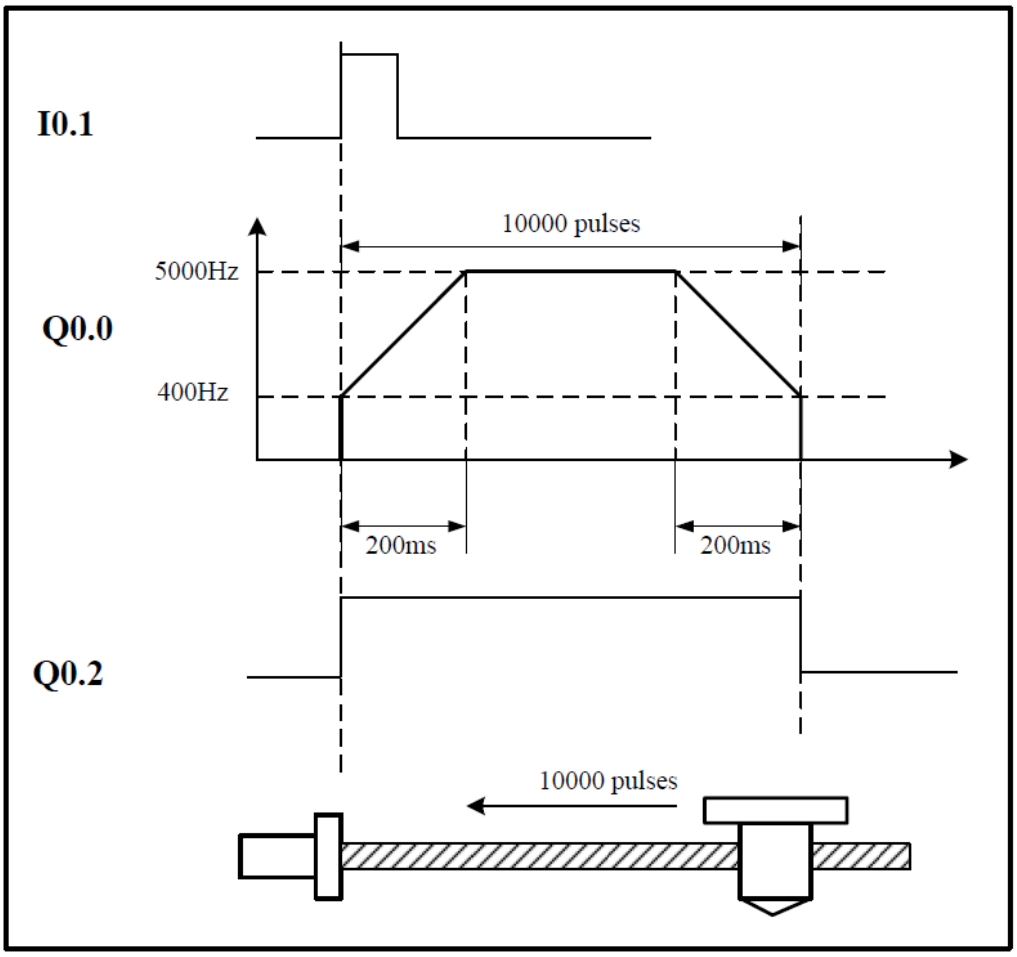
سیستمی که در ادامه به عنوان مثال مطرح میشود جهت توضیح چگونگی استفاده از دستورات PHOME , PABS ,PREL,PJOG,PSTOP میباشد :

:WIRING

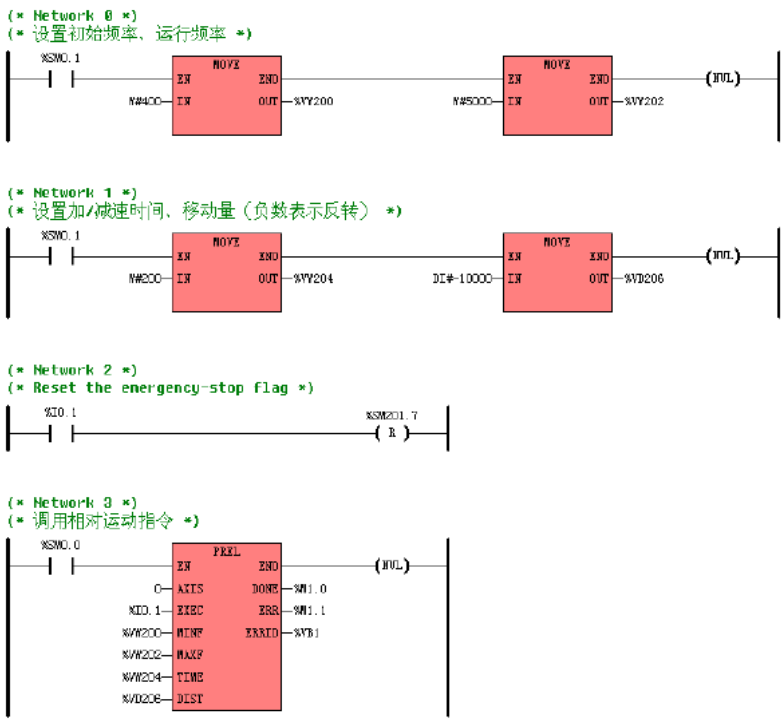


حرکت نسبی: I0.1 به منظور آغاز حرکت نسبی (به صورت عقبگرد) استفاده شده است .

Time Sequence Diagram



LD

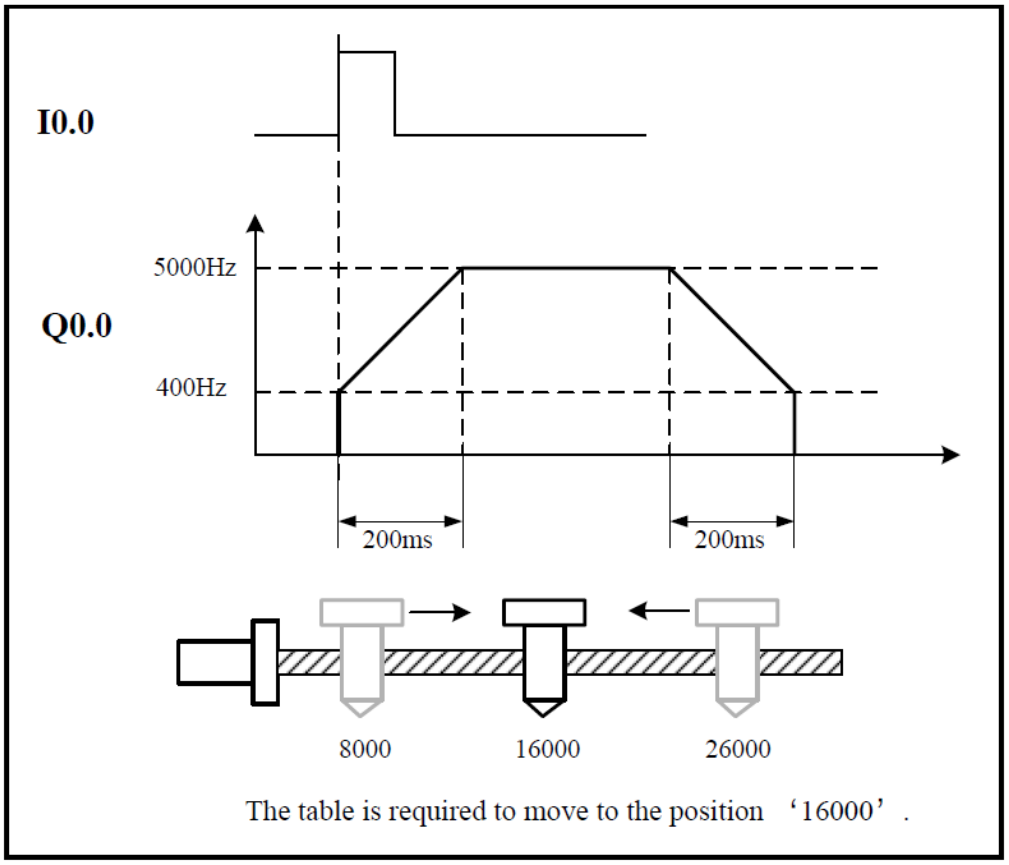


IL

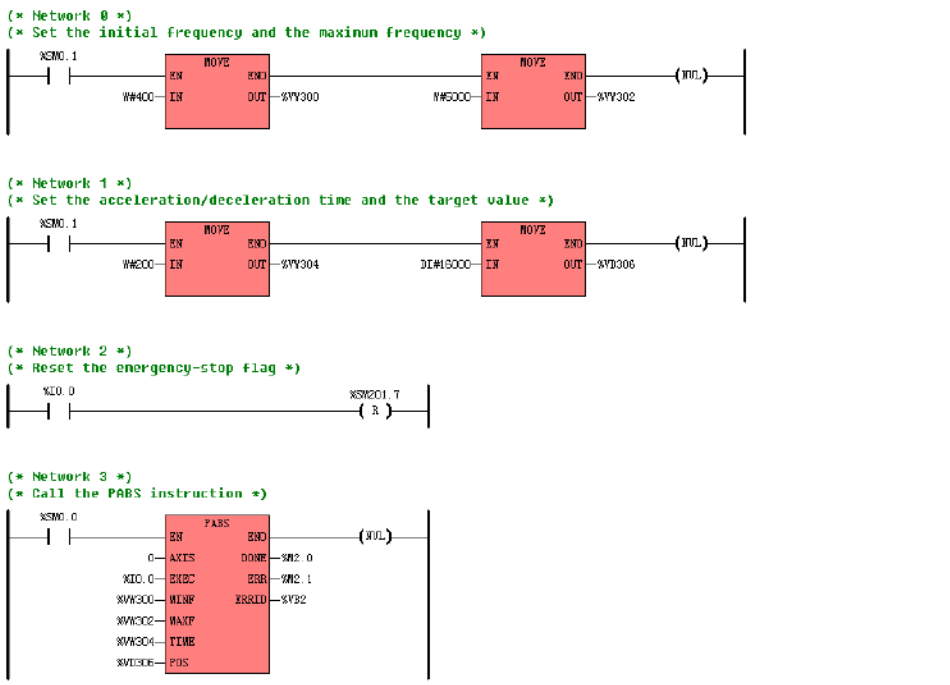
```
(* Network 0 *)
(*Set the initial frequency and the maximum frequency*)
LD      %SM0.1
MOVE    W#400, %VW200
MOVE    W#5000, %VW202
(* Network 1 *)
(*Set the acceleration/deceleration time and the distance*)
LD      %SM0.1
MOVE    W#200, %VW204
MOVE    DI#-10000, %VD206
(* Network 2 *)
(*Reset the emergency-stop flag*)
LD      %I0.1
R       %SM201.7
(* Network 3 *)
(*Call the PREL instruction*)
LD      %SM0.0
PREL    0, %I0.1, %VW200, %VW202, %VW204, %VD206, %M1.0, %M1.1, %VB1
```

حرکت مطلق : I0.0به منظور شروع حرکت مطلق استفاده شده است.

Time Sequence Diagram



LD



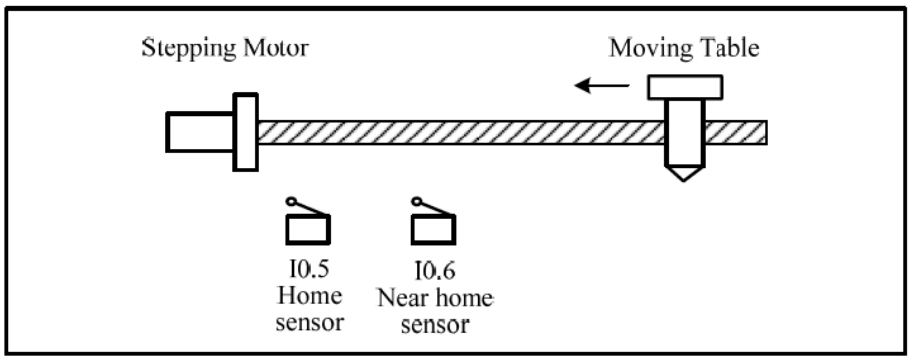
IL

```
(* Network 0 *)
(*Set the initial frequency and the maximum frequency*)
LD      %SM0.1
MOVE    W#400, %VW300
MOVE    W#5000, %VW302
(* Network 1 *)
(*Set the acceleration/deceleration time and the target value*)
LD      %SM0.1
MOVE    W#200, %VW304
MOVE    DI#16000, %VD306
(* Network 2 *)
(*Reset the emergency-stop flag*)
LD      %I0.0
R       %SM201.7
(* Network 3 *)
(*Call the PABS instruction*)
LD      %SM0.0
PABS    0, %I0.0, %VW300, %VW302, %VW304, %VD306, %M2.0, %M2.1, %VB2
```

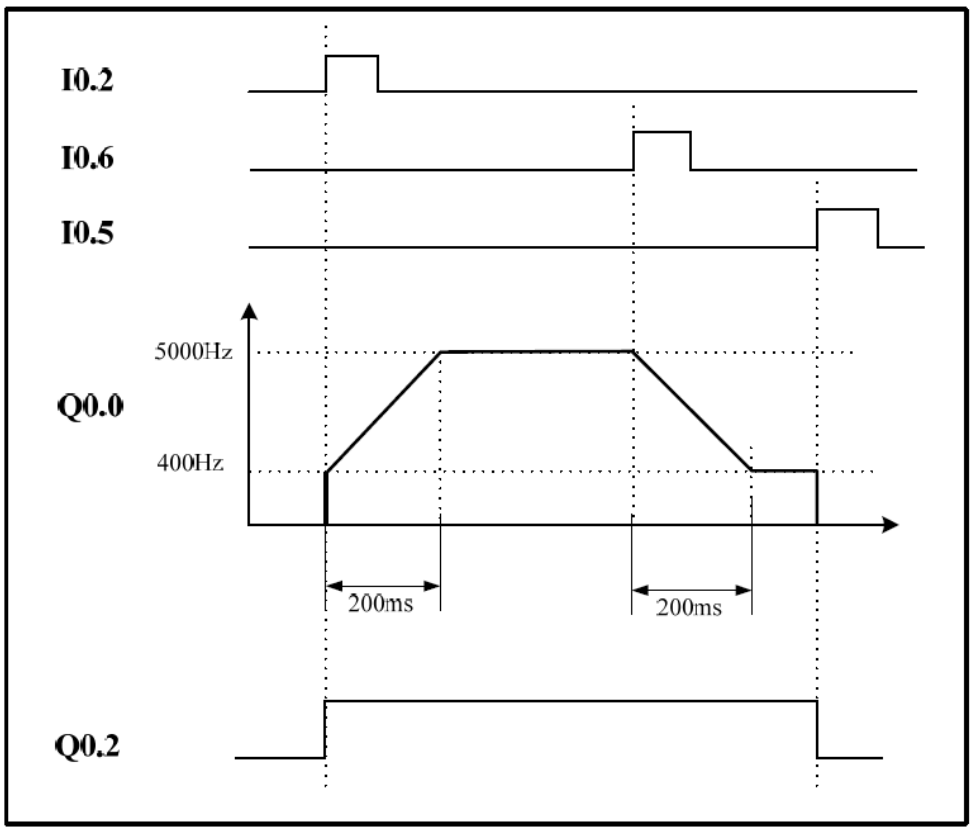
HOME: I0.2 به منظور آغاز شروع برگشت به موقعیت HOME استفاده شده است .

Time Sequence Diagram

Supposing that the moving is in the following initial status:



During the motion, Q0.2 is 1 because of moving backwards。



IL

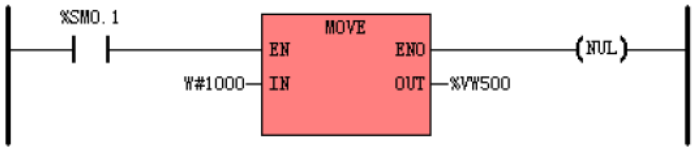
```
(* Network 0 *)
(*use both the home and the near home input; move backwards*)
LD      %SM0.1
MOVE    0, %VW396
MOVE    1, %VW398
(* Network 1 *)
(*set the initial frequency, maximum frequency and acceleration/deceleration time*)
LD      %SM0.1
MOVE    W#400, %VW400
MOVE    W#5000, %VW402
MOVE    W#200, %VW404
(* Network 2 *)
(*Reset the emergency-stop flag*)
LD      %I0.2
R       %SM201.7
(* Network 3 *)
LD      %SM0.0
PHOME   0, %I0.2, %I0.5, %I0.6, %VW396, %VW398, %VW400, %VW402, %VW404, %M3.0, %M3.1, %VB3
```

JOG : I0.3به منظور شروع حرکت به سمت جلو و I0.4 به منظور شروع حرکت به سمت عقب استفاده شده است :

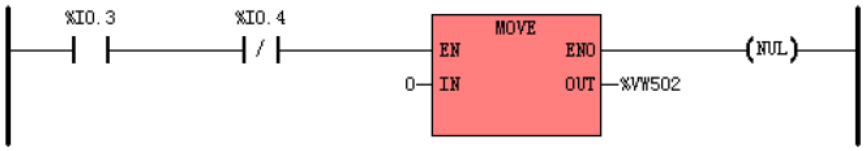
چنانچه I0.3وI0.4 هردو ۱ باشند جهت اخیر دنبال میگردد.

LD

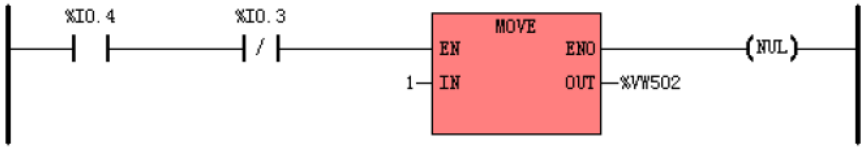
```
(* Network 0 *)
(* Set the frequency of PTO *)
```




```
(* Network 1 *)
(* Set the direction *)
```

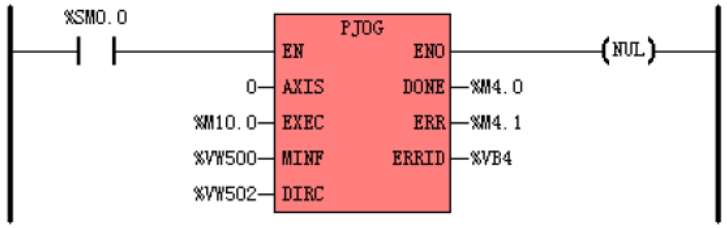



```
(* Network 2 *)
```



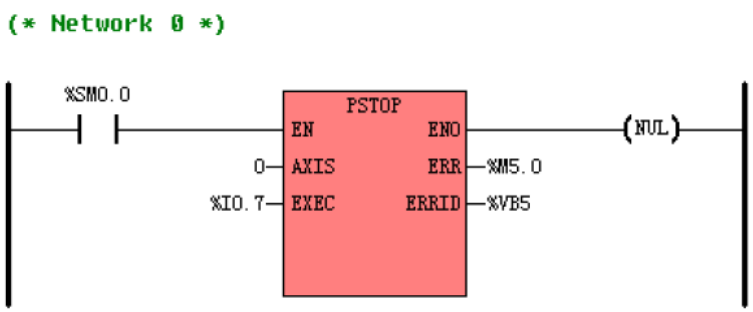

```
(* Network 3 *)
(* Jog *)
```


```
(* Network 4 *)
```



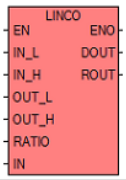
IL	(* Network 0 *)
	(*Set the frequency of PTO*)
	LD %SM0.1
	MOVE W#1000, %VW500
	(* Network 1 *)
	(*Set the direction*)
	LD %I0.3
	ANDN %I0.4
	MOVE 0, %VW502
	(* Network 2 *)
	LD %I0.4
	ANDN %I0.3
	MOVE 1, %VW502
	(* Network 3 *)
	(*Jog*)
	LD %I0.3
	OR %I0.4
	ST %M10.0
	R %SM201.7
	(* Network 4 *)
	LD %SM0.0
	PJOG 0, %M10.0, %VW500, %VW502, %M4.0, %M4.1, %VB4

:PSTOP

LD	
IL	(* Network 0 *) LD %SM0.0 PSTOP 0, %I0.7, %M5.0, %VB5

8.17 دستورات دیگر :

8.17.1 LINCO :

	Name	Usage	Group	
LD	LINCO			☐ CPU304 ☐ CPU304EX ☐ CPU306 ☑ CPU306EX ☑ CPU308
IL	LINCO	LINCO *IN_L, IN_H, OUT_L, OUT_H, RATIO, IN, DOUT, ROUT*	U	

Operands	Input/Output	Data Type	Acceptable Memory Areas
IN_L	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constants
IN_H	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Constants
OUT_L	Input	REAL	V, L, Constants
OUT_H	Input	REAL	V, L, Constants
RATIO	Input	REAL	Constants
IN	Input	INT	I, Q, V, M, L, SM, T, C, AI, AQ
DOUT	Output	DINT	Q, M, V, L, SM
ROUT	Input	REAL	V, L

از این دستور جهت scsle بندی استفاده می گردد.توجه داشته باشید که پایه های IN_L ,IN_H ,OUT_L ,OUT_H میتوانند هم به صورت عدد ثابت و هم به صورت متغیر تعریف شوند.

این دستور ورودی IN را بر اساس یک رابطه خطی مشخص محاسبه کرده و سپس نتیجه حاصل را در ROUT و DOUT ذخیره مینماید .این رابطه خطی براساس روش دو نقطه (که تشکیل یک خط میدهند) میباشد که این دو نقطه همان (IN_L ,OUT_L) و (IN_H , OUT_H) خواهد بود .

عملکرد تابع LINCO را میتوان بر اساس فرمول ارائه شده در پایین توضیح داد:

$$ROUT = RATIO * (k*IN + b)$$

$$DOUT = TRUNC(ROUT)$$

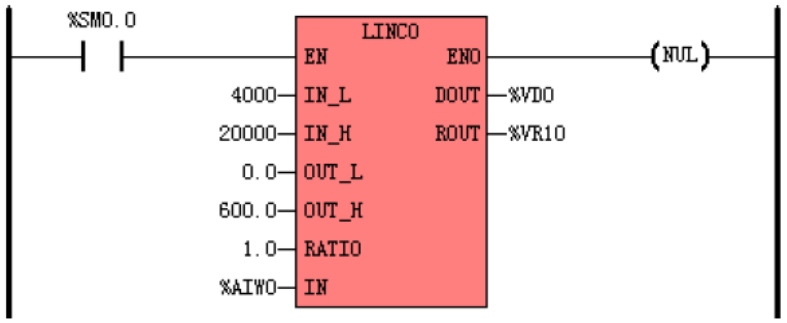
$$k = \frac{OUT\_H - OUT\_L}{IN\_H - IN\_L} \quad , \quad b = OUT\_L - k \times IN\_L$$

LD: چنانچه EN یک باشد این دستور اجرا خواهد شد .

IL: چنانچه CR یک باشد این دستور اجرا خواهد شد . این دستور تاثیری بر مقدار CR نخواهد داشت .

مثال: تصور کنید که برای اندازه گیری دما یک ترنسمیتر با رنج 0~600 درجه سانتی گراد داشته باشید و خروجی این ترنسمیتر یک سیگنال جریان در رنج 4mA ~20m A باشد .خروجی این ترنسمیتر به ورودی آنالوگی (به عنوان مثال AIW 0) متصل خواهد بود . در این حالت برای بدست آوردن دمای واقعی باید سیگنال جریان

دریافتی در کانال آنالوگی را به دما تبدیل نمایید .

LD	
IL	LD %SM0.0 LINCO 4000, 20000, 0.0, 600.0, 1.0, %AIW0, %VD0, %VR10

نکته : همان طور که در قسمت های قبل گفته شد ، KINCO-K3 ورودی آنالوگ (جریانی – ولتاژی) را خوانده و آن را به صورت ۱۰۰۰ X در حافظه آنالوگی ذخیره مینماید .

پیوست: عملکرد برخی رجیسترهای حافظه SM:

پس از هر سیکل اسکن ، firmware داخلی cpuهای kinco داده های سیستمی ذخیره شده در فضای حافظه SM را به روز میکنند.میتوان برای ارزیابی وضعیت فعلی سیستم برخی از این حافظه ها را خواندو نیز میتوان برای کنترل های عملکرد سیستمی در برخی از این حافظه های خاص نوشت.

SMB0 –1:

این بایت در هر سیکل اسکن توسط CPU به روز میشود. بیت های این بایت از حافظه فقط قابل خواندن بوده و کاربر میتواند در مواقع مورد نیاز از این بیت ها استفاده نماید .

SM Bit	Description
SM0.0	Always ON
SM0.1	ON during the first scan cycle only. Usually used for some initializations.
SM0.2	If the data in RAM is lost, this bit is ON during the first scan cycle, and later cleared to FALSE.
SM0.3	Provide a pulse train (50% duty cycle) with a cycle time of 1s.
SM0.4	Provide a pulse train (50% duty cycle) with a cycle time of 2s..
SM0.5	Provide a pulse train (50% duty cycle) with a cycle time of 4s.
SM0.6	Provide a pulse train (50% duty cycle) with a cycle time of 60s.
SM0.7	Reserved

2- SMW 22 و SMW24 :

SMW22 به منظور ذخیره سازی مقدار CYCLE TIME در وقفه زمانی ۰ (EVENT 3) استفاده میشود.رنج آن 1~65535 و واحد آن میلی ثانیه میباشد. در صورتی که مقدار ۰ در آن ذخیره شده باشد وقفه ۰ غیر فعال میباشد . به صورت پیش فرض مقدار این حافظه ۰ میباشد .

SMW24 به منظور ذخیره سازی مقدار CYCLE TIME در وقفه زمانی 1 (EVENT 4) استفاده میشود.رنج آن 1~65535 و واحد آن میلی ثانیه میباشد. در صورتی که مقدار ۰ در آن ذخیره شده باشد وقفه 1 غیر فعال میباشد . به صورت پیش فرض مقدار این حافظه ۰ میباشد .

3- SMW26وSMW28 : این دو حافظه برای ذخیره سازی مقدار آنالوگی دو پتانسومتر(که بر روی ماژول CPU قرار دارد)میباشد . SMW26برای پتانسیومتر شماره ۱ و SMW28برای پتانسیومتر ۰ میباشد .این دو حافظه فقط قابل خواندن میباشند.

4- SMB31وSMW32:

CPU 304EX،CPU 304 و CPU 306 برای BACKUP داده به صورت دائم استفاده میکنند.توضیحات بیشتر در پیوست C میباشد.

پیوست C: ذخیره اطلاعات به صورت دائم (Permanent Data Backup):

داده های ذخیره شده در رنج خاصی از فضای حافظه V میتوانند در حافظه FRAMذخیره شده و به صورت دائم باقی بمانند. برای این منظور میتوان از حافظه های SMB31وSMW32 استفاده کرد .

رنج فضای حافظه برای ذخیره سازی دائم :

جدول زیر لیست رنجی از فضای حافظه Vرا که میتواند در حافظه FRAM ذخیره شود نشان میدهد .برای ذخیره سازی دائم میتوان این فضا ها را فراخوانی کرد .

	CPU304	CPU304EX, CPU306, CPU306EX and CPU308
Length	128 bytes	255 bytes
Range	VB1648~VB1775	VB3648~VB3902

چگونگی ذخیره سازی داده ها به صورت دائم :

۲.۱ : CPU306 EX وCPU 308 :

دراین مدل از CPUها فقط کافی است که داده های مورد نظر را در حافظه های دائمی ذخیره نمایید . در این صورت این داده ها به صورت اتوماتیک در حافظه FRAM ذخیره شده و به صورت همیشگی باقی میمانند.

به مثال زیر توجه نمایید:

```
(*NETWORK  0*)
LD      %SM0.0
MOVE    %AIW0, %VW3648        (* store the value of AIW0 permanently *)
SPD     1, W#1000, %VD4000    (* calculate the frequency of the pulse train from HSC1 *)
                              (* and store the frequency permanently *)
```

۲.۲: CPU 304، CPU 304EX وCPU 306:

زمانی که از این مدل CPU ها استفاده میکنید برای ذخیره دائم اطلاعات باید به صورت زیر عمل نمایید:

۱)نوشتن وذخیره داده ها در حافظه های دائم (که در جدول بالا به آن اشاره شد)

۲) استفاده از SMB31وSMW32 به منظور انتقال این داده ها از فضاهای حافظه اشاره شده به FRAM

2.2.1: SM31.0، SM31.1وSM31.7:

SM31.1	SM31.0	Description
0	0	Save a BYTE (8-bit) value
0	1	Save a BYTE (8-bit) value
1	0	Save a WORD (16-bit) value
1	1	Save a DWORD (32-bit) value

SM31.7	Description
0	Enable writing into FRAM
1	Disenable writing into FRAM

۲.۲.۲: SMW32

آدرس داده در فضای حافظه V در این رجیستر ذخیره میشود .این مقدار به عنوان یک offset از حافظه VB0در نظر گرفته میشود.

2.2.3:نوشتن در FRAM:

فرمت نوشتن در FRAMبه صورت زیر میباشد:

MOVE *offset*, %SMW32

OFFبه صورت یک OFF SET از حافظه VB0میباشد.(یک عدد INT میباشد) و آدرس داده در فضای حافظه V راکه بایدبه صورت دائم ذخیره گردد را مشخص میکند.

به عنوان مثال اگر بخواهید VB3600را در FRAM ذخیره کنید مقدار OFFSETباید ۳۶۰۰ باشد.توجه داشته باشید که CPU در پایان هر سیکل اسکن این دستور را برای ذخیره داده در FRAM اجرا میکند.

به مثال زیر توجه نمایید:

```
(* NETWORK   0 *)
(* Write VB3649, VW3650, VD3652 into FRAM under the control of M0.0*)
 LDN     %M0.0              (* If M0.0 is 0 *)
 MOVE    B#0, SMB31         (* Disenable writing into FRAM *)

(* NETWORK 1 *)
 LD      %M0.0              (* If M0.0 is 1 *)
 MOVE    B#2#10000001, SMB31   (* To save 1 byte *)
 MOVE    3649, %SMW32       (* Save VB3649 to FRAM*)
 MOVE    B#2#10000010, SMB31   (* To save 1 word (2 bytes) *)
 MOVE    3650, %SMW32       (* Save VW3650 to FRAM *)
 MOVE    B#2#10000011, SMB31   (* To save 1 double-word (4 bytes) *)
 MOVE    3652, %SMW32       (* Save VD3652 to FRAM *)
```

فصل نهم معرفی KINCO K5 :

نسل جدید PLC های کارخانه Kinco سری Kinco K5 میباشد .این خانواده از PLC ها ، نسل به روز شده سری Kinco K3 میباشند.

این نسل از PLC ها نسبت به سری K3، از لحاظ کارایی سخت افزاری و نرم افزاری بهبود یافته و از لحاظ ظاهر و نیز نحوه برنامه نویسی تقریبا مشابه با سری K3 میباشد .

نرم افزاری که میتوانید برای برنامه نویسی این نسل از PLC ها استفاده نمایید ، KincoBuilder 1.5.0.0 میباشد.

برای آشنایی با جزئیات بیشتر در بخش برنامه نویسی میتوانید از help نرم افزار کمک بگیرید .

در ادامه توضیحات مختصری در ارتباط با سخت افزار Kinco K5 ارائه میگردد.

ماژول K5:

قانون نام گذاری در سری K5 دقیقا مطابق با قانون نام گذاری سری K3 میباشد .شایان ذکر میباشد که ماژول های سری K5 و ماژول های سری K3 تطابقی با هم نداشته و کاربر نمیتواند از ماژول های سری K3 برای CPU سری K5 استفاده نماید .به عبارت دیگر ماژول های سری K3 قابل اتصال و استفاده در CPU های سری K5 نمیباشندو چنانچه نیاز به استفاده از ماژول های افزایشی باشد ، لزوما" باید از ماژول های سری K5 استفاده نمود.

نسل جدید K5 در مقایسه با سری K3 دارای کارایی و عملکرد بهتر و نیز فانکشن های بیشتری میباشد.

لطفا" به جدول مقایسه سری های K5 و K3 توجه نمایید :

	I/O	RS232	RS485	Max Modules	RTC	K5
K304	DI*8 , DO*6	1	--	--	--	1. 10 times faster than K3 2. Strengen other detais
K504		1	--	--		

K304EX	DI*8 , DO*6	1	--	4	yes	1.10 times faster than K3 2. Strengen other detais 3.The max pulse output 200KHz 4. Support canopen communication via K541 module.
K504EX		1	1	2		
K306	DI*14, DO*10	1	--	4	yes	
K306EX		1	1	15	yes	
K506		1	2	6	yes	
K308	DI*24, DO*16	1	1	15	yes	
K508		1	2	6	yes	
K506EA	DI*14, DO*10, AI*4 , AO*2	1	2	6	yes	

تغییر CPU K3 به CPU K5:

چنانچه قصد جایگزینی CPU K3 با نسل جدید CPU K5 را داشته باشید ، در نظر گرفتن نکات زیردر برنامه الزامی میباشد:

پیکربندی سخت افزاری (Hardware configuration):

در ورژن جدید نرم افزار KincoBuilder لسیتی از سخت افزار های سری K5 وجود دارد . پروژه قدیمی را باز کرده و مدل CPUرا از K3 CPU به K5 CPU و نیز ماژول های افزایشی را از سری K3 به سری K5 تغییر دهید.سپس برنامه به برنامه جدید سری K5 تغییر میابد. در برنامه جدید کاربر میتواند از فانکشن های جدید که مختص سری K5 میباشد استفاده کند.

نکته :

ماژول هایی که در جدول پیکربندی سخت افزاری انتخاب میشود باید مطابق با سخت افزاری که در پروژه مورد استفاده قرار میگیرد باشد ، در غیر این صورت برنامه دانلود نمیشود.

ماژول های K3 و K5تطابقی با یکدیگر ندارند ، لذا نمیتوان از ماژول های سری K3 برای CPU K5 استفاده نمود.

برنامه PLC:

به دلیل آنکه CPU K3 و CPU K5 به منظور برنامه نویسی از یک نرم افزار (KincoBuilder) استفاده میکنند ، بنابراین دستورات این دو سری از PLC ها کاملا" با یکدیگر مطابقت دارند. شایان ذکر است با توجه به موارد گفته شد تفاوت های مختصری با یکدیگر دارند که در ادامه به آنها اشاره خواهد شد .

لطفا" در هنگام برنامه نویسی موارد زیر را در نظر بگیرید:

دستورات موقعیت (Position instruction):

ماکزیمم فرکانس پالس خروجی در سری K5 ، 200KHZ میباشد.بنابراین ماکزیمم فرکانس (MAXF) در دستورات موقعیت باید به صورت DWORD در نظر گرفته شود (در سری K3 به صورت WORD در نظر گرفته میشد)

به علاوه نکته ای که در بالا بدان اشاره شد ، ریست کردن مقدار جاری (" Reset Current Value") نیز در رجیستر های کنترلی اضافه گردیده است .

دستورتولید پالس (PLS instruction):

دستور PLS تنها برای تولید PWM مورد استفاده قرار میگیرد. چنانچه بخواهید از تابع PTO برای ارسال پالس استفاده کنید ، ازدستورات موقعیت استفاده نمایید.

شمارنده های سرعت بالا(HSC):

K5 تنها دارای دو کانال شمارنده سرعت بالا میباشد: HSC0 و HSC1 . در صورتی که سری K3 دارای ۶ شمارنده سرعت بالا (HSC0~HSC5) میباشد.

وقفه تایمر(Timer Interrupt) :

Time base برای وقفه تایمر در سری K5 ، 0.1 ms میباشد. رجیسترهای interrupt cycle SMD12(برای وقفه تایمر شماره ۰) و SMD16(برای وقفه تایمر شماره ۱) میباشد.

تعدادی فانکشن جدید برای سری K5 وجود دارد : communication PORT 2، فضای بیشتر حافظه M، فضای بیشتر برای حافظه های ماندگارو...

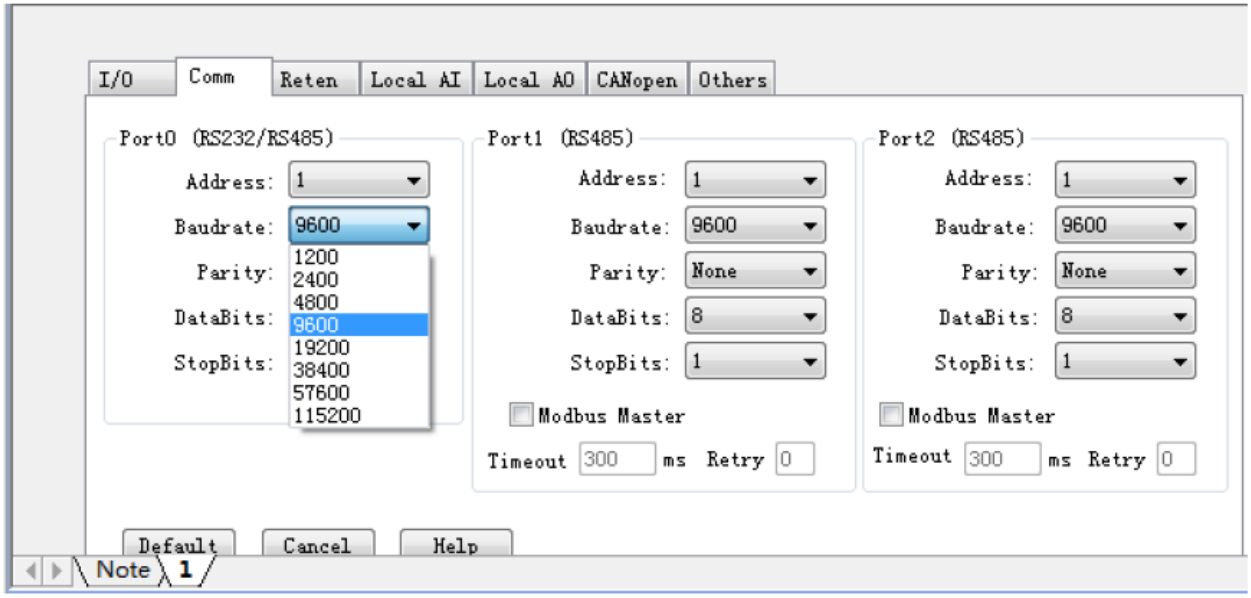
پورت ارتباطی در K5:

K506، K506EA، K508 دارای سه پورت ارتباطی هستند: Port 0 ,Port 1, Port 2

Port 0 پورت سریال RS232 ، Port 1 و Port 2 پورت سریال RS485 میباشد.

تمامی پارامترهای پورت های ارتباطی در PLC در قسمت Hardware >> Comm قابل تنظیم میباشد . برای پورت ۰ در K5 Baudrate جدید اضافه شده است : 57.6 K 115.2 K .

به علاوه Baudrate جدید ۱۲۰۰ نیز به دو پورت دیگر (Port 1, Port 0) اضافه شده است .



پورت ۲ (Port 2) کاملا" مشابه با پورت ۱ (Port 1) میباشد. این پورت دستورات ارتباطی (XMT ,RCV) و دستورات Modbus master (MBUSR ,MBUSW) را پشتیبانی میکند.

رجیسترهای کنترلی و وضعیت پورت ۲ ، SMB286 ~SMB294 میباشد.برای اطلاعات بیشتر به فایل help نرم افزار KincoBuilder مراجعه نمایید.

فضای حافظه در K5:

فضای حافظه M:

فضای حافظه M در K5 CPU به 1024 byte ارتقاء یافته است. میتوانید از این فضا به صورت بیتی ، بایتی ، word و dword استفاده کنید . جزئیات بیشتر مطابق با جدول زیر میباشد:

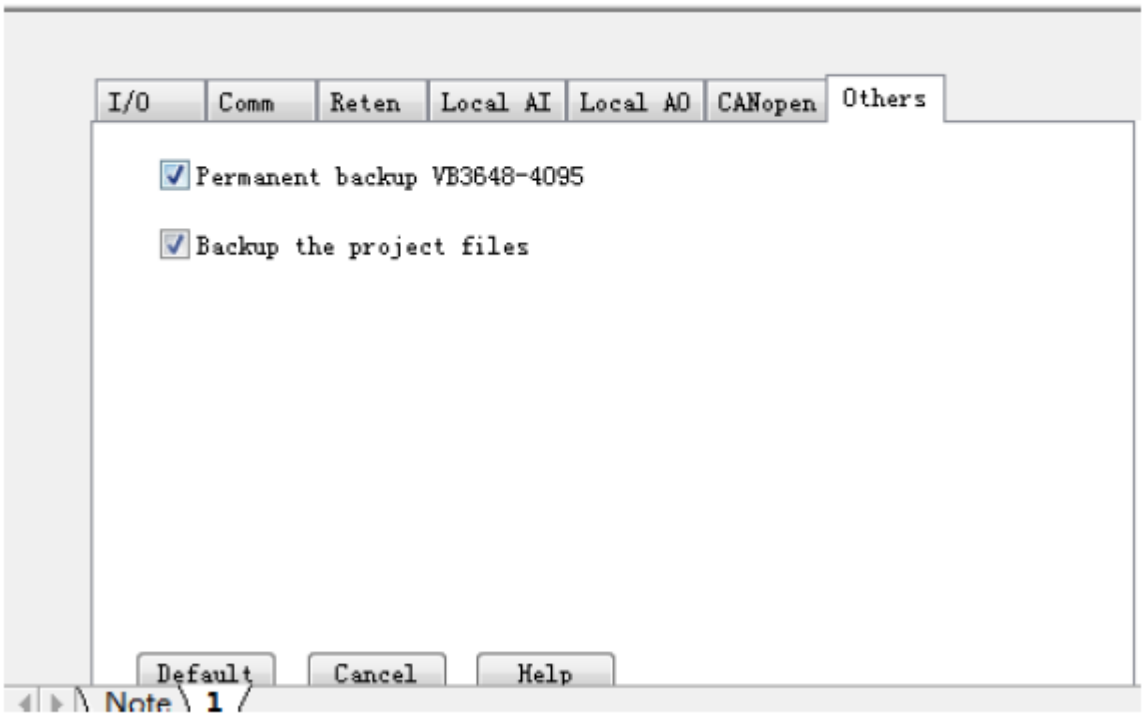
		1024 Bytes
M	Bit Address	%M0.0 --- %M1023.7
	Byte Address	%MB0 --- %MB1023
	Word Address	%MW0 --- %MW1022
	DWord Address	%MD0 --- %MD1020

آدرس مربوط به مدباس فضای M در رنج 320~8511 میباشد .۳۲۰ به معنی M0.0 و ۸۵۱۱ به معنی M1023.7 میباشد.

Date backup :

۴۸۸ رجیستر(VB3648 ~VB4095) به عنوان رجیستر های ماندگار جهت ذخیره اطلاعات به صورت دائم وجود دارد. داده های PLC در این رجیسترها به صورت ماندگار ذخیره میگردند.

نکته : به صورت پیش فرض ، فضای حافظه ذخیره دائم در PLC های K 5 نیز مشابه با K3 بوده واین فضا VB3902 ~VB 3648 میباشد. چنانچه کاربر نیاز به استفاده از فضای حافظه دائم بیشتری داشته باشد(VB3648~VB4095) ، باید تنظیمات لازم را در قسمت Hardware>>Others انجام دهید . چنانچه کاربرتنظیمات لازم را در قسمت hardware انجام ندهد ، داده ها به صورت دائم در فضای حافظه VB3903 ~VB4095 ذخیره نمیگردد.



C) کارکرد جدید در فضای حافظه SM:

در سری K5، تعدادی رجیسترهای سیستمی جدید و عملکردهای جدید دررجیسترهای سیستمی اضافه شده است .

برای اطلاعات بیشتر به جدول زیر توجه نمایید:

SM	Function
SMB0	
SM0.2	Read Only. If the reten data in RAM is lost, this bit is set to 1 in the first scan of CPU, then it is reset to 0
SMB2	
SM2.0	Read&Write, its value depends how CPU work if fatal error happens. 0：If fatal error happens, PLC runs in independent safe sub system. 1：If fatal error happens, reset PLC directly.
SM2.1	Read&Write, Set the working mode of the AI and AO on CPU itself. 0：Set AI/AO chnnels in normal working mode. 1：Set AI/AO chnnels in adjusting mode.
SMB6	
SMB6	Read Only, store the latest scan time of PLC, unit：ms.
SMW10	
SMW10	Read Only, store the voltage value of back-up battery, unit：0.01V。
SMD12,SMD16	
SMD12	Read&Write.Set the cycle of Timer Interrupt 0, unit: 0.1ms
SMD16	Read&Write.Set the cycle of Timer Interrupt 1, unit: 0.1ms
The cycle of Timer Interrupt 0 and 1 are set in SMW22 and SWM24 in K3, these two are not used in K5.	
SMB286-SMB294	
SMB286-SMB294	Read&Write. Cntrol and status registers in free communication of Port2, The detail functions are the similar as port0 and port1 in K3, please refere to the corresponding pages in help file of Kinco Builder

I/O پورت در K5:

ورودی شمارنده سرعت بالا:

K5 دارای دو شمارنده سرعت بالا میباشد. HSC0 و HSC1 . مقدار در حال شمارش این شمارنده ها در حافظه های HC0 و HC1 ذخیره میگردند. ماکزیمم فرکانس شمارش در این شمارنده ها در مد تک فاز 60KHz و در مد دو فاز 20KHz میباشد. نحوه برنامه

نویسی شمارنده های سرعت بالا در سری K5 دقیقا" مشابه با سری K3 میباشد. در ادامه تعدادی از اختلافات در ورودی های سخت افزاری بیان شده است . به جدول زیر توجه نمایید:

HSC 0				
Mode	**Description**	**I0.1**	**I0.0**	**I0.5**
0	Single-phase up/down counter with internal direction control Direction Control: SM37.3	Clock		
1			Reset	
2			Reset	Start
3	Single-phase up/down counter with external direction control	Clock		Direction
4			Reset	Direction
6	Two-phase counter with up/down clock inputs	Clock (Down)	Clock (Up)	
9	A/B phase quadrature counter	Clock (A)	Clock (B)	

HSC1					
Mode	**Description**	**I0.4**	**I0.6**	**I0.3**	**I0.2**
0	Single-phase up/down counter with internal direction control Direction Control: SM47.3			Clock	
1		Reset			
2		Reset	Start		
3	Single-phase up/down counter with external direction control			Clock	Direction
4		Reset			Direction
5		Reset	Start		Direction
6	Two-phase counter with up/down clock inputs			Clock (Down)	Clock (Up)
7		Reset			
8		Reset	Start		
9	A/B phase quadrature counter			Clock(A)	Clock(B)
10		Reset			
11		Reset	Start		

خروجی سرعت بالا:

CPU K5 دارای دو کانال خروجی سرعت بالا Q0.0 و Q0.1 میباشد(کاملا" مشابه با CPU K3).

ماکزیمم فرکانس خروجی سرعت بالا در سری K5 تا 200KHz میباشد. تنها تفاوتی که در سری K5 وجود دارد این است که از دستور PLS تنها میتوان در PWM استفاده نمود .(از این دستور نمیتوان در PTO استفاده کرد)

چنانچه کاربر بخواهد با استفاده از K5 پالس های سرعت بالا ارسال نماید ، لطفا" از دستورات موقعیت (PHOME ,PABS,PREAL,...) استفاده نمایید . دستورات موقعیت مشابه با K3 میباشد .

سیگنال کنترلی جهت موتور:

در دستورات موقعیت برای هر کانال خروجی سرعت بالا یک بیت کنترلی سیگنال وجود دارد که برای کنترل جهت موتوراستفاده میشود . چنانچه این بیت صفر باشد ، حرکت به سمت جلو بوده و چنانچه یک باشد ،جهت حرکت معکوس خواهد بود . همچنین بیت های فعال ساز برای کنترل جهت (رجیستر های سیستمی) وجود دارد که این بیت ها جهت فعال و یا غیر فعال کردن سیگنال جهت استفاده شود .به جدول زیر توجه نمایید :

Pulse Output Channel	**Q0.0**	**Q0.1**
Direction Oupput Channel	Q0.2	Q0.3
Direction Enable Control	SM201.3	SM231.3

بیت های فعال ساز برای کنترل جهت دارای اولویت بالایی میباشند . چنانچه بیت های فعال ساز یک باشند،دستور موقعیت سیگنال کنترل جهت خروجی نخواهد داشت و خروجی های مربوطه (Q0.2 و Q0.3) به عنوان خروجی های معمولی استفاده میشوند.

رجیستر های کنترلی و رجیسترهای وضعیت در دستورات موقعیت :

در دستورات موقعیت برای هر خروجی سرعت بالا رجیسترهای کنترلی و وضعیت وجود دارد . به جدول زیر توجه نمایید :

Q0.0	Q0.1	Description
SM201.7	SM231.7	Read&Write. Bit of emergency stop, if this bit is 1, the motor is in emergency stop state, no position instructions will be executed. When the PSTOP instruction is executed, this bit is set to 1 automatically. User needs to set it to 0 before he want to execute other position instructions.
SM201.6	SM231.6	Read&Write.Reset the current value or not 1 --- Reset current value in SMD212 or SMD242 0 --- Holdon the current value in SMD212 or SMD242
SM201.5~SM201.4	SM231.4~SM231.5	Reserved
SM201.3	SM231.3	Read&Write.Direction Control Enable 1 --- Disable direction output channel, direction ouput channel can beusedas normal DO. 0 --- Enable direction output channel。
SM201.0~SM201.2	SM231.0~SM231.2	Reserved
Q0.0	**Q0.1**	**Description**
SMD212	SMD242	Read Only, the current value (increase in forward rotating, decrease in reverse rotating), it means how many pulse have been output.

در PLCهای K5 پارامترهای MAXF به صورت DWORDمیباشد (این پارامتر در PLC های K3 به صورت WORD میباشد)

هنگامی که خطایی در دستورات موقعیت اتفاق میوفتد ، کد خطا در پارامتر ERRID وجود دارد. جزئیات بیشتر در جدول زیر بیان شده است .

Error Code	Description
0	No error
1	The initial pulse cycle exceeds the time of each segment
2	The initial speed (*MINF*) exceeds the highest pulse speed (400KHz).
3	The initial speed(*MINF*) is lower than the lowest pulse speed(125Hz)
4	The pulse number in acceleration and deceleration progress exceeds the total number of pulse.
5	The initial speed (*MINF*) exceeds the highest speed(MAXF)

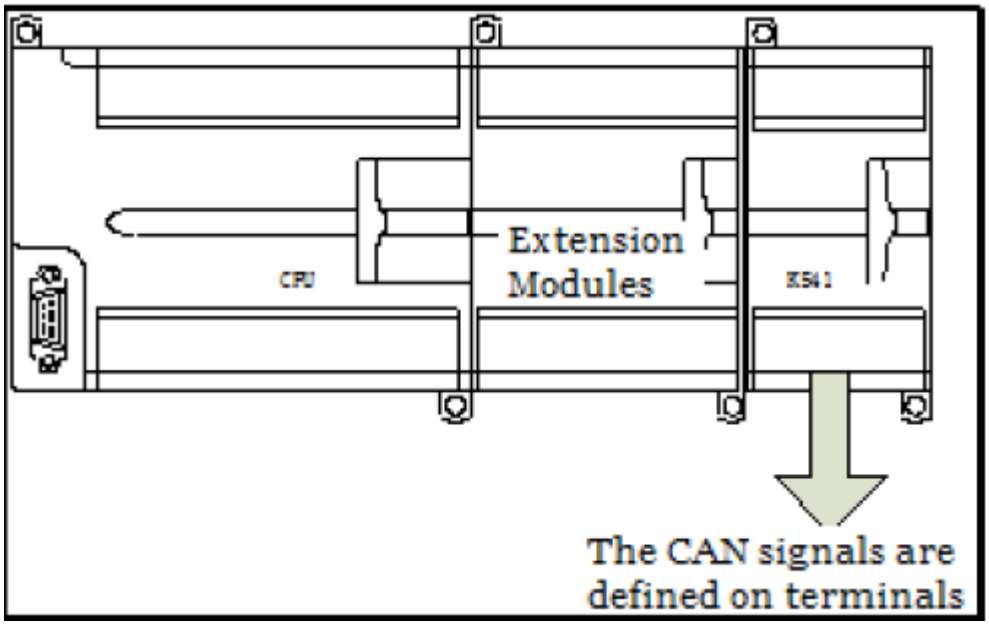
:CANopen Communication Function

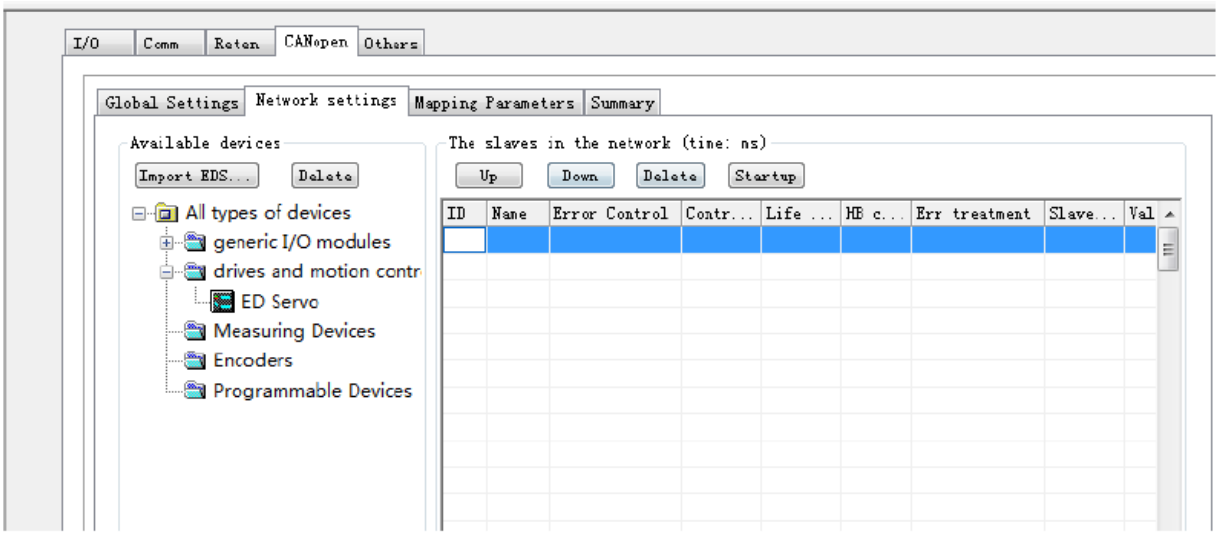
در سری های K5، ماژول ارتباطی CANOpen (K541) اضافه گردیده است .تمامی مدل های K5 به استثنای K504 میتواند ارتباط CANOpen master را با استفاده از ماژول K541 پشتیبانی کند.

ماژول K541 ، baudrate استاندارد 10K~1M را پشتیبانی میکند.

چگونگی استفاده از K541:

ماژول K541 باید در انتهای تمامی ماژول های افزایشی قرار گیرد. کاربر جهت استفاده از این ماژول نیازی به انتخاب ماژول K541 درجدول Hardware در نرم افزار KincoBuilder نخواهد داشت. تنها باید پارامترهای CANopen را در قسمت Hardware<<CANopen پیکربندی کرد. کاربر میتواند در برنامه از عملکردCANopen master و یا عملکرد CANopen free communication استفاده نماید.

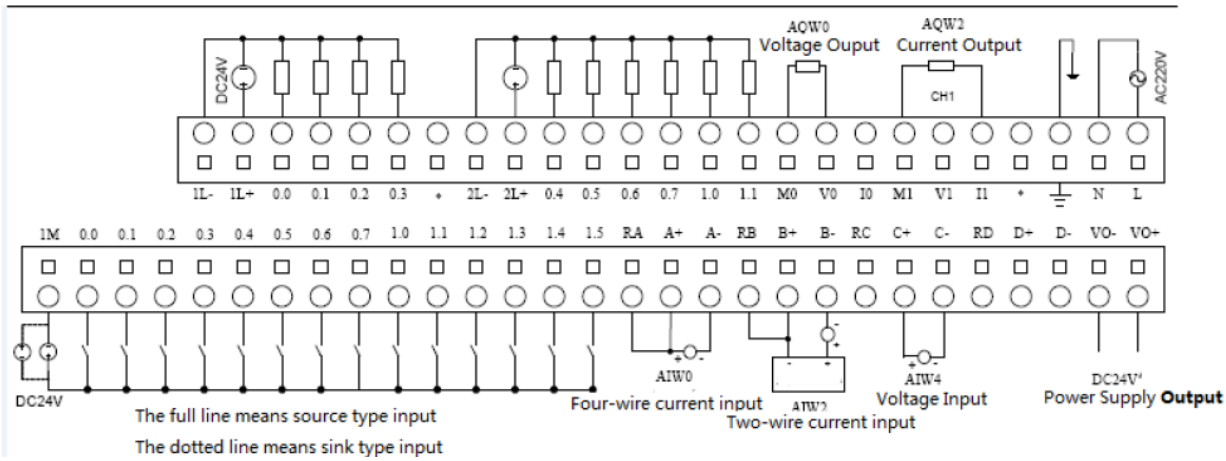




سیم بندی K5 و اتصالات سخت افزاری:

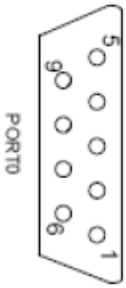
در تصویر زیر نمونه ای از نقشه سیم بندی که مربوط به CPU K506EA-30AT میباشد ، نشان داده شده است .

سیم بندی سایر مدل ها نیز مشابه با نقشه زیرخواهد بود .



در تصویر زیر pinout پورت های ارتباطی نیز نشان داده شده است . این pinout در پورت های ارتباطی برای تمامی مدلها یکسان خواهد بود .

	Pin	Define	Descr
PORT1	3	B	RS485 -
	2	A	RS485 +
PORT2	8	B	RS485 -
	7	A	RS485 +



	Pin	Define	Descr
PORT0	5	GND	SG
	3	TxD	Send
	2	RxD	Receive

به طور کلی مشخصات CPU های سری K5 مطابق با جدول زیر میباشد:

Parameter	CPU504	CPU504EX	CPU506	CPU506EA	CPU508
I/O and communication port					
Built-in digital points	8*DI / 6*DO		14*DI / 10*DO	14*DI / 10*DO	24*DI / 16*DO
Built-in analog points	--	--	--	4*AI / 2*AO	--
Number of connectable expansion modules	--	2	6	6	6

Communication ports	1 RS232	2, PORT0: RS232; PORT1: RS485	3, PORT0: RS23; PORT1/PORT2: RS485
	PORT0 supports programming, Modbus RTU (master), free protocol; PORT1/PORT2 support programming, Modbus RTU (master/slave), free protocol;		
High-speed counter Single-phase Dual-phase	2, with a maximum frequency of 30KHz 2, with a maximum frequency of 10KHz	2, with a maximum frequency of 60KHz 2, with a maximum frequency of 20KHz	
Pulse output	2, with a maximum frequency of 20KHz	2, with a maximum frequency of 200KHz	
Memory area			
User program memory	Maximum 1K steps	Maximum 4K steps	
user data memory	M area 1KB; V area 4KB.		
DI mapping area	1 bytes (8*DI)	5 bytes (40*DI)	32 bytes (256*DI)
DO mapping area	1 bytes (8*DO)	5 bytes (40*DO)	32 bytes (256*DO)
AI mapping area	--	16 bytes (8*DO)	64 bytes (32*DO)
AO mapping area	--	8 bytes (4*DO)	64 bytes (32*DO)
Data backup characteristic	E^2PROM, 256 bytes	FRAM,448 bytes	
Data retention	--	4K bytes. Lithium battery, 3 years at room temperature.	

characteristic		
Other		
Timer	256 1ms time base:4 10ms time base:16 100ms time base:236	
Timer interruption	2 time base: 0.1ms.	
Counter	256	
Real-time clock	--	Yes, with an error not greater than 2 minutes/month under temperature of 25℃.
DC24V output supply	300mA, short circuit protection	500mA, short circuit protection