التحكم المنطقي المبرمج

Programmable Logic Controller

Siemens S7-300/400

م/ حسن الشحات حسن

http://hassanheha.forumn.net

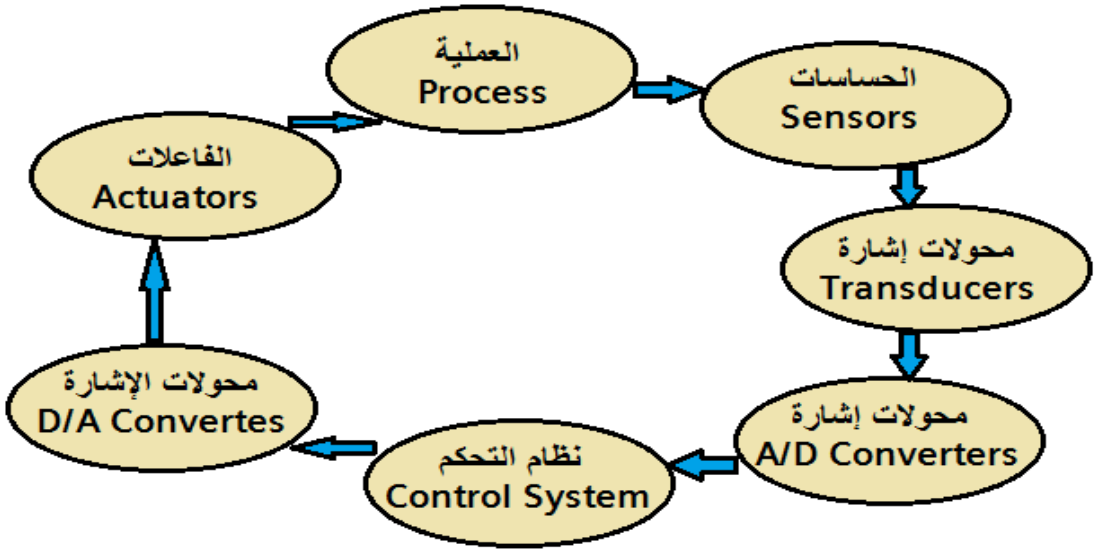
الفهرس

تابع الفهرس

نظام التحكم Control System

مقدمة:

إن تنفيذ أي عملية أو جزء من عملية مثل حركة محرك أو تشغيل عنصر تسخين أو حركة ذراع يستند إلى اسطوانة هوائية أو ضغط سائل أو غاز في أنبوب يتطلب أداة معينة تقوم بالتنفيذ بشكل يتناسب مع العملية وللتعرف على حالة تنفيذ العملية ووضعها قبل وأثناء وبعد التنفيذ يستلزم الأمر أجهزة استشعار أو حساسات تنقل الكميات الطبيعية Physical إلى إشارات كهربية مفهومة نستطيع التعامل معها ولتنفيذ عمل منهجي منظم فإن الأمر يتطلب منا صياغة لعمليات التحكم في الوظيفة نفسها وهذا يقودنا إلى شكل قياسي لمنظومة التحكم بأشكالها المختلفة



1

منظومة التحكم المغلقة تبدو كما في الشكل أمامنا مكونة من عناصر يتم تمثيلها بأشكال مختلفة:

العملية Process

العملية مثل تسخين جسم أو حركة محرك أو ضغط سائل أو غاز أو تغيير سرعة محرك إلى غير ذلك من احتياجات الصناعة أو الحياة اليومية وتوصيف العملية بشكل صحيح سوف يحدد بالتأكيد ذراعيها الفاعل Actuator والحساس Sensor للتعرف على الوضع الحالي للعملية

وماكينات وخطوط الإنتاج بالمصانع تحتوي على أنواع عديدة من العمليات التي تتطلب السيطرة عليها لتنفيذ عمليات التصنيع على النحو الأفضل

الفاعلات Actuators

الفاعلات هي عناصر المنظومة التي تقوم بتنفيذ العملية فعليا بتحويل الطاقة الكهربية إلى إحدى صور الطاقة المختلفة سواء طاقة حركية أو حرارية أو ضوئية أو ضغط أو سريان إلى غير ذلك من أشكال الطاقة وللفاعلات أشكال كثيرة مثل المحركات الكهربية والسخانات وطلمبات الضغط وغيرها سواء كانت وسائل بسيطة أو معقدة




الحساسات Sensors

وهي العنصرالذي يقوم بمراقبة تنفيذ العملية بقياس الكمية التي يتم التحكم فيها ويحولها إلى شكل كهربي مفهوم بشكل مباشر أو عن طريق محول إشارة Transducer وهناك لكل العمليات حساسات مختلفة فهناك حساسات للحرارة وهناك حساسات للضغط وحساسات للسرعة وحساسات للسريان وحساسات مستوى وحساسات للمسافة إلى غير ذلك من الأشكال المختلفة للتنفيذ وتوجد بشكليها التماثلي والرقمي

فالحساسات التماثلية Analog sensors تعطي إشارة متصلة تأخذ شكلا من الأشكال القياسية والتي قد تكون فولت أو مللي فولت أو مللي أمبير أو مقاومة أو بشكل آخر من الأشكال المتعارف عليها أما الحساسات الرقمية فتعطي حالة فقط وفي شكل نقطة توصيل إما مفتوحة وتغلق حال التفعيل NO أو مغلقة وتفتح حال التوصيل NC وفي كل الأحوال تستخدم الحساسات لإعطائنا موقف تنفيذ العملية طبقا للتصميم


حساس حرارة


حساس ضغط


حساس تقاربي Proximity


حساس حد Limit switch


مولد تاكو للسرعة


حساس مستوى - عوامة


حساس سريان


خلية وزن Load cell

أمثلة لبعض أنواع الحساسات في عالم الصناعة

محولات الإشارة Transducers

تقوم محولات الإشارة كوسيط بين الحساس بنظريته الفيزيائية وبين نظام التحكم الكهربي حيث يقوم بتحويل الإشارة المحسوسة إلى إشارة كهربية قياسية يمكن استخدامها في أنظمة التحكم وفي معظم الأحيان يكون جزءا من الحساس نفسه

محولات الإشارة A/D Converters

في الأنظمة الرقمية أو التي تتعامل مع إشارات تناظرية Analog تحتاج تلك الأنظمة لتحويل قيمة الإشارة إلى رقم لتتم عليه العمليات وبالتالي نحتاج إلى تلك المحولات لتعطينا القيمة المناظرة للإشارة لنقوم عليها بالحسابات داخل المنظومة.

محولات الإشارة D/A Converters

كذلك الحال عند الحاجة إلى تحويل قيمة رقمية إلى إشارة كهربية في نظم التحكم فسوف نحتاج إلى هذا النوع من المحولات

نظام التحكم Control System

نظام التحكم هو الذي يستقبل المدخلات عبر الحساسات ووسائل الإدخال وربطها بشكل حسابي أو منطقي ليعطي مخرجات معينة تتحكم في شكل أداء الفاعلات لتنفيذ العملية المطلوبة

ويمكن أن يتم تنفيذه بطرق عديدة لكنها كلها لا تخرج عن الحسابات المنطقية والرقمية بأشكالها المختلفة

ولكي ننتقل إلى كيفية تمثيل نظم التحكم العنصر الرئيسي والذي سيكون محور حديثنا في الفترة القادمة نبدأ بالتعرف على العناصر التي سنستخدمها في بناء منظومات التحكم وتشغيل العمليات المختلفة

3

أولا : عناصر الحماية

ماذا نحمي وبماذا نحمي؟ سؤال مهم قبل أن نتنقل للحديث التفصيلي عن الحماية فنحن نتحدث عن دوائر كهربية وأحمال كهربية فلدينا مصدركهربي ووسائل نقل ومنظومات تحكم وأحمال وفي كل من تلك المراحل نجد جزءا من الحماية لابد أن نعيره الاهتمام

المصدر الكهربي له خصائص تتمثل في فرق الجهد الكهربي Electric voltage وتردد التيار الكهربي المصدر وترتيب فازات المصدر Phase sequence وتيار الحمل المسموح به Load current والشوشرة على شكل الموجة الكهربية لمصدر التغذية Noise ومعامل القدرة Power factor

ففرق الجهد الكهربي يلزم الحماية ضد ارتفاعه عن حد معين أو انخفاضه عن حد معين Over/Under voltage

والتردد يجب الحماية ضد ارتفاعه أو انخفاضه بنسبة معينة عن القيمة الطبيعية 50 هرتز مثلا ±2

وترتيب الفازات يجب أن يكون بالترتيب R ثم S ثم T ويكون ثابتا لنفس المصدر على الدوام

كما يجب أن يكون هناك حماية ضد قصر الدائرة Short Circuit وكذلك الحمل الزائد عن المقنن بالنسبة للدائرة Overload

كما يجب توفير وسيلة لرفع معامل القدرة Power factor لتكون أكبر من 0.95

ويجب كذلك معالجة الشوشرة خاصة تلك التي يسببها Harmonics بسبب استخدام الأحمال التي تحتوي على ملفات ومكثفات في نفس الوقت

كل هذه العوامل يتم توفير وسائل حماية للمنظومة وللمصدر

فارتفاع الحمل وقصر الدائرة له أكثر من وسيلة للحماية ضده مثل الفيوزات والقواطع الحرارية والأوفرلود (الحمل الزائد)

فيوزات

قاطع Circuit breaker

قاطع حراري

حماية أوفرلود

وزيادة أو نقصان فرق الجهد الكهربي يوجد ريلاي للفولت Voltage relay يوفر الحماية ضده حيث يتم ضبطه في نطاق معين للتشغيل الصحيح

وترتيب الفازات أيضا يوجد ريلاي يسمى Phase sequence relay للحماية ضد اختلاف ترتيب الفازات

كذلك للتردد أيضا ريلاي يضمن التشغيل داخل نطاق معين للتردد

أما حماية معامل القدرة فتتم عن طريق إضافة أحمال سعوية Capacitive load لتعويض فارق معامل القدرة ولها حساباتها الخاصة ولها أجهزة التحكم الخاصة بها

والشوشرة في حالة تأثيرها على العملية أو خشية التأثير يتم إضافة فلاتر Filters كهربية لامتصاص تلك الشوشرة قبل الاستخدام

جهاز تحكم في معامل قدرة

أحمال مكثفات معادلة لمعامل القدرة

وحدات فلتر

ثانيا : عناصر التوصيل/الفصل

هناك أشكال عديدة من عناصر التوصيل والفصل منها المفاتيح بأشكالها والريليهات بأشكالها والكونتاكتورات بأشكالها والحساسات الرقمية بأشكالها وسوف نستعرض معا بعضا من أشكال تلك العناصر وصور تواجدها في عالم الصناعة

المفاتيح

مفاتيح الضغط Push Buttons

هو مفتاح مزود بسوسته حيث عند الضغط عليه يقوم بالتوصيل وعند تحريره يفصل مرة أخرى وهذا في حالة نقطة التوصيل حال عدم الضغط مفتوحة Normally open (NO) أما لو كانت نقطة التوصيل حال عدم الضغط مغلقة Normally closed (NC) فعند الضغط يتم الفصل وعند تحرير المفتاح يعود للتوصيل مرة أخرى

مفتاح الاختيار Selector switch

وهو مفتاح للاختيار بين أوضاع للتشغيل وضعين أو أكثر ويمكن أن يكون سوسته فيكون مثل Push button تماما أو العادي حيث في كل وضع يكون هناك وضع وحيد للتوصيل أو الفصل من أوضاع التشغيل حسب اختيار نقاط التوصيل عليه NO أو NC

دواسة القدم Foot/Pedal Switch

وتستخدم في المناطق التي تعمل باستخدام القدم لسهولة الوصول لها بالقدم أو انشغال اليدين وتعمل بنفس الطريقة للمفاتيح

مفتاح توقف مفاجئ Emergency stop switch

يستخدم مفتاح التوقف المفاجئ لفصل عمليات التشغيل تماما في الماكينات كوسيلة ظاهرة سهل الوصول إليها للمشغل

مفتاح نهاية مشوار Limit switch

يستخدم مفتاح نهاية المشوار في الإحساس بنهاية حركة معينة حيث يثبت على الجزء الثابت أو الجزء المتحرك ويقوم بالتفصيل أو الفصل عند الالتقاء

مفتاح ضغط Pressure switch

يستخدم مفتاح الضغط للإحساس بقيمة معينة للضغط والتوصيل أو الفصل بناء على نقطة التوصيل

مفتاح حراري Thermostatic switch

يستخدم المفتاح الحراري للإحساس بالحرارة والتوصيل أو الفصل عند درجة حرارة معينة

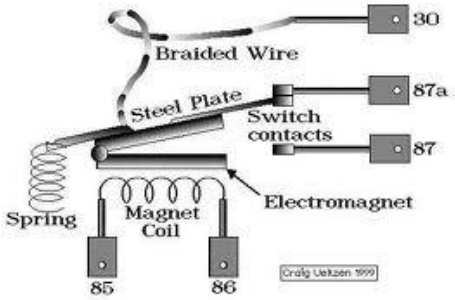
مفتاح السريان Flow switch

يستخدم حساس السريان للإحساس بسريان مائع (سائل/غاز) في مكان ما ويقوم بالتوصيل أو الفصل بناء على حالة السريان

وهناك حالات أخرى عديدة من المفاتيح ترتبط بنوع معين من التأثيرات الميكانيكية أو الطبيعية وتقوم بالتوصيل أو الفصل بناء على ذلك

الريليهات Relays

الريلاي هو وسيلة كهربية للتحكم في إشارة كهربية عن طريق إشارة كهربية أخرى مع العزل الكامل بين الإشارتين حيث يتم توصيل الإشارة الحاكمة على ملف الريلاي وتوصيل الإشارة المتحكم فيها عن طريق نقطة توصيل للتوصيل أو الفصل



يعمل الريلاي بتوصيل جهد التشغيل على ملف التشغيل وبناء عليه يتم التوصيل أو الفصل بين نقاط التوصيل

شكل من أشكال الريليهات المستخدمة على نطاق واسع في عالم التحكم الآلي

شكل من أشكال الريليهات التي يتم تثبيتها على الكروت الإلكترونية

وهناك أنواع أخرى من الريليهات مرتبطة بوظيفة معينة مثل ريلاي مؤقت وريلاي تيار وريلاي مستوى وغيرها حسب الوظيفة المرتبط بها تماما مثل الحساسات حيث يقوم بالتوصيل بناء على شرط معين مرتبط بشئ فيزيائي

6

ريلاي مؤقت Timer Relay

يستخدم للتوصيل أو الفصل بناء على شرط مرتبط بالوقت سواء تأخير توصيل أو تأخير فصل أو توليد نبضات على فترات قابلة للضبط

ريلاي تيار Current Relay

يستخدم حيث يتم ضبطه على تيار حمل معين ولو كان أقل منه يفصل أو يوصل نقطة تستخدم في دوائر التحكم

ريلاي حالة صلبة Solid State Relay

يستخدم للتوصيل والفصل في الحالات التي تستدعي تردد توصيل وفصل عالي

ريلاي مستوى Level Relay

ويستخدم للتحكم في مستوى سائل بناء على توصيل طرف إحساس بالسائل أو أكثر وبناء على ذلك يقوم بتوصيل أو فصل نقطة توصيل

ريلاي عداد Counter Relay

ويستخدم للقيام بعمليات العد والتوصيل أو الفصل بناء على شرط الوصول إلى عدد معين للعداد

ريلاي فولت Voltage Relay

ويستخدم للتوصيل أو الفصل بناء على مقارنة قيمة معينة سواء فوق أو تحت تلك القيمة أو نطاق بين قيمتين

الكونتاكتورات Contactors

بينما تستخدم الريليهات في عمليات التحكم والإشارات الضعيفة بينما الكونتاكتور والذي يماثل تماما الريلاي في تشغيله إلا أنه يستخدم بجانب عمليات التحكم في توصيل القدرات الكهربية الكبيرة للأجهزة المختلفة منها أيضا أنواع

كونتاكتور عادي Contactor

ويتكون من ملف تشغيل يتم توصيفه بجهد التشغيل الكهربي ونقاط توصيل أساسية لتوصيل القدرة الكهربية إضافة إلى نقاط مساعدة تستخدم في التحكم

كونتاكتور زئبق Mercury Contactor

وهو يماثل الكونتاكتور العادي في الملف إلا أن التوصيل يكون عن طريق موصل زئبقي ويستخدم في الحالات التي تستدعي توصيل وفصل سريع مثل دوائر السخانات

وتستخدم منظومات التحكم العادية والبسيطة في معظمها مكونات بسيطة من تلك التي ذكرناها سواء مفاتيح أو ريليهات أو كونتاكتورات بالإضافة إلى عناصر الحماية التي ذكرناها سابقا

ويفضل دائما الفصل والعزل بين دوائر القدرة الكهربية ودوائر التحكم وحمايتها كذلك وإن أمكن أن نفصلها أيضا في المكان يكون أفضل

ويمكن تنفيذ عملية الفصل والعزل في دوائر التحكم البسيطة عن طريق المحولات والتي تستخدم لفصل دوائر الملف الثانوي عن دوائر الملف الابتدائي ومن الأفضل في حالة استخدام دائرة تحكم بجهد كهربي 230 فولت أن يتم استخدام محول لتغذية دوائر التحكم 230 فولت بدلا من استخدام L,N من المصدر الرئيسي وفي مثل تلك الحالة يؤمن تماما في حالات الشورت مع الأرضي حدوث تلف في عناصر دائرة التحكم

هذا وسوف نكمل باقي عناصر منظومة التحكم في الأجزاء التالية إن شاء الله

المحركات الكهربية Electric Motors

المحركات الكهربية هي أحد أهم العناصر في عالم الصناعة بل ربما في الحياة العملية اليوم بما تمثله من أدوار كثيرة في معظم الماكينات والأدوات المستخدمة في الحياة اليومية وهي العنصر الفاعل الأهم Actuator من عناصر منظومة التحكم الآلي السبعة (العملية Process - نظام التحكم Control system – الحساسات Sensors – المحولات Transducers - الفاعلات Actuators – المحولات الرقمية التناظرية D/A Converters - المحولات التناظرية الرقمية A/D Converters) فالمحركات أحد الفاعلات Actuators والتي تقوم بأداء معظم أشكال الحركة بأنواعها الطولية والدورانية والترددية بإمكانياته الهائلة.

وللمحرك باعتبار التغذية الكهربية إليه فرعان رئيسيان هما محركات التيار المتردد AC Motors ومحركات التيار المستمر DC Motors

أولا : محركات التيار المتردد AC Motors

لو تحدثنا عن كل فرع منها على حده فسوف نبدأ بمحركات التيار المتردد AC Motors حيث تنقسم أيضا باعتبار عدد فازات التغذية إلى أحادي الفاز Single phase motor وثلاثي الفاز Three phase motor ولم ندخل في عملنا كثيرا في تفاصيل تكوين المحرك إذ نبحث هنا عن عمليات التحكم في المحرك أكثر من مهمة البحث عن أشكاله وأصنافه وأكثر ما يوجد في عالم الصناعة حاليا تحت مسمى محرك حثي Induction Motor وأكثر المحركات انتشارا في عالم الصناعة هي محركات ذات القفص السنجابي.

وحسب مواصفات وخصائص المحركات يمكن تنفيذ العديد من عمليات التحكم فيها لتنفيذ العديد من العمليات الصناعية ومنها العمليات الآتية :

8

- التشغيل والإيقاف
- عكس اتجاه الدوران
- التحكم في سرعة الدوران
- التحكم في عزم الدوران

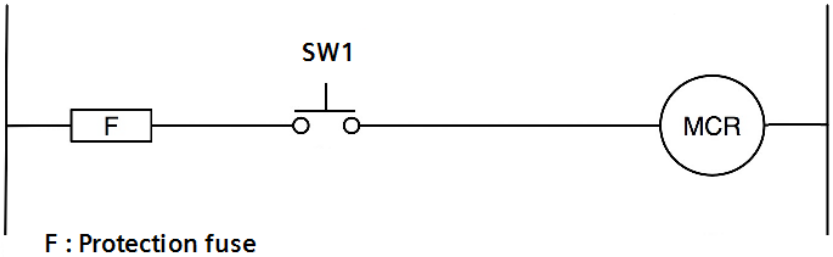
وهذا كما ذكرنا طبقا لخصائص ومواصفات المحرك

- فبتوصيل القدرة الكهربية للمحرك يتم التشغيل وبفصلها يتم الإيقاف
- وبتغيير ترتيب فازات المحرك الثلاثي يمكن عكس اتجاه الحركة وبعكس أطراف ملفات التقويم في المحركات الأحادية ذات ملفات التقويم يمكن عكس اتجاه الدوران
- وبتغيير التردد يمكن تغيير سرعة دوران المحرك
- وبتغيير فرق الجهد والتيار الكهربي يمكن التحكم في عزم الدوران للمحرك

أمثلة على التحكم في تشغيل محركات التيار المتردد (تشغيل/إيقاف/عكس اتجاه الحركة) :

مثال 1 :

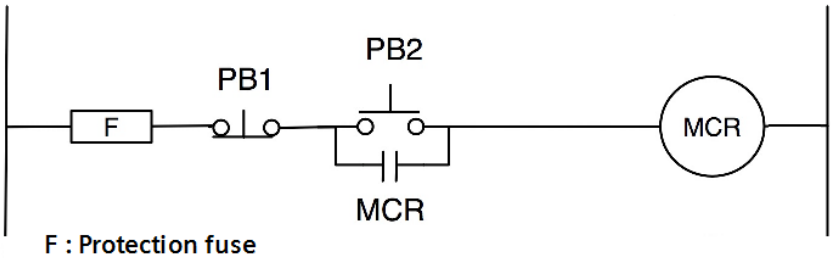
تشغيل وإيقاف محرك باستخدام مفتاح واحد من النوع ON/OFF Selector وكونتاكتور لتوصيل القدرة الكهربية وحماية أوفرلود للحماية من ارتفاع تيار الحمل



F : Protection fuse
SW1 : Selector Switch ON/OFF
MCR : Power Contactor

مثال 2 :

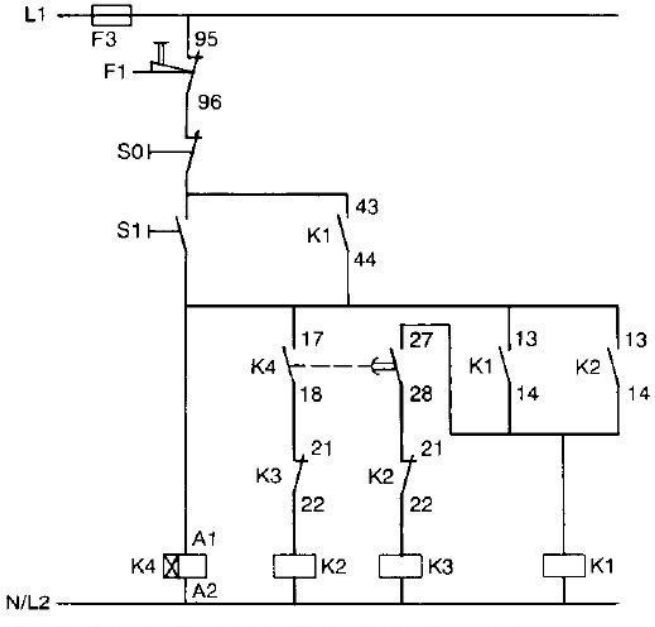
تشغيل وإيقاف محرك باستخدام عدد 2 مفتاح Push Button الأول للتشغيل وتكون النقطة فيه NO والآخر للإيقاف وتكون النقطة فيه NC مع كونتاكتور مناسب للحمل الكهربي وحماية أوفرلود ضد ارتفاع تيار الحمل



F : Protection fuse
PB1 : Stop Push Button
PB2 : Start Push Button
MCR : Power Contactor

مثال 3 :

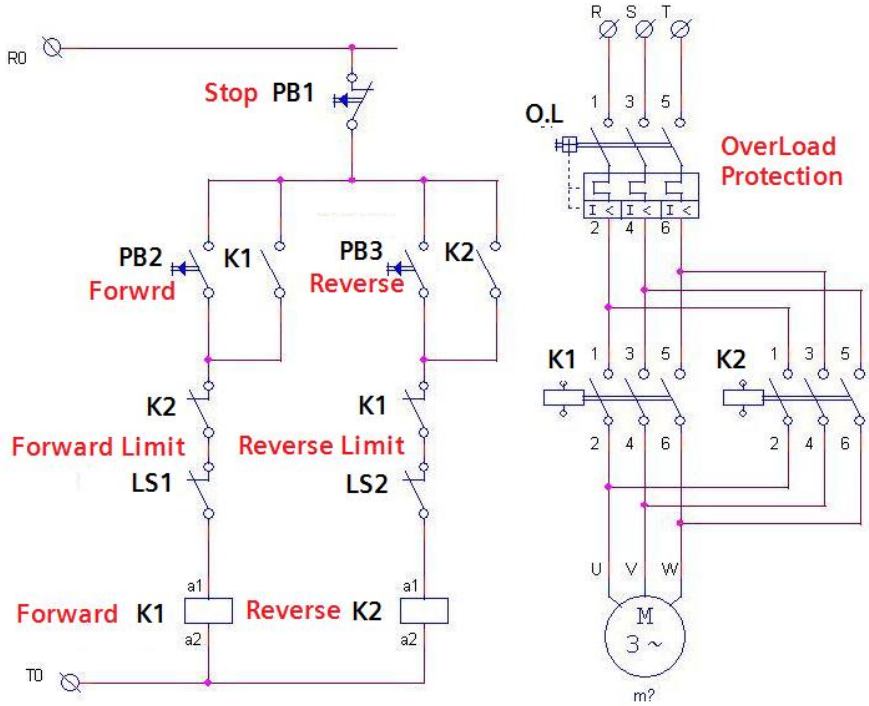
تشغيل وإيقاف محرك بنظام ستار/دلتا باستخدام عدد 2 مفتاح Push Button تشغيل/إيقاف وعدد 1 تايمر للانتقال من ستار إلى دلتا وعدد 3 كونتاكتور 1 عام وآخر للتشغيل ستار وآخر للتشغيل دلتا مع حماية أوفرلود



S0 = 'OFF' Push button
S1 = 'ON' Push button
K1 = Line contactor
K2 = Star contactor
K3 = Delta contactor
K4 = Star delta timer
F2 = Overload relay
F1 = Backup fuse
F3 = Control circuit fuse

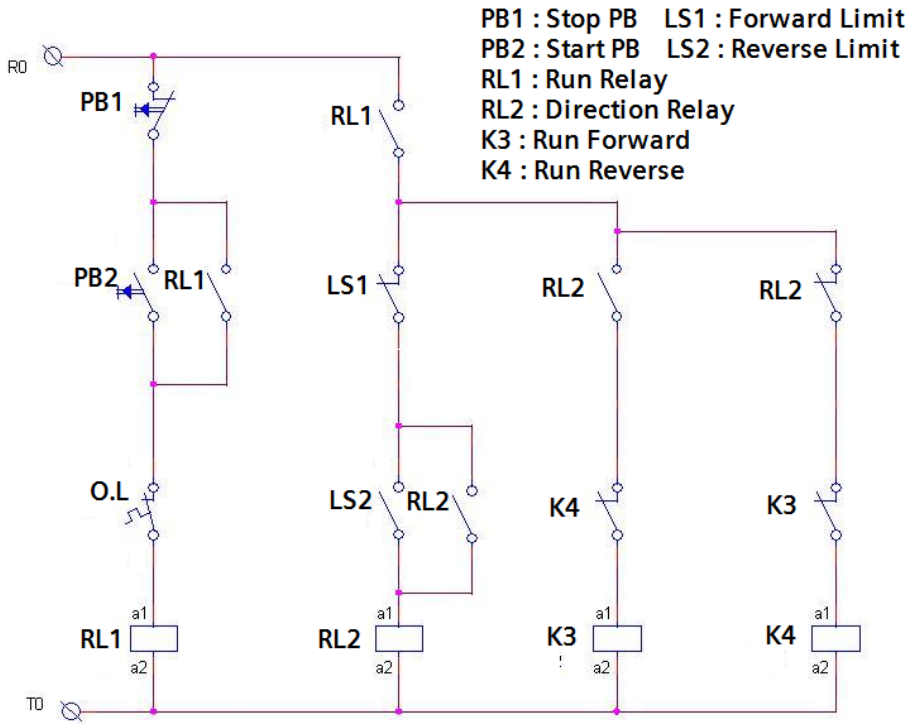
مثال 4 :

تشغيل وإيقاف محرك يعمل في اتجاهين بثلاثة مفاتيح Push Button أحدها تشغيل أمامي وآخر تشغيل خلفي وآخر للإيقاف مع مفتاحي حد أمامي وخلفي Limit Switch لفصل الحركة باستخدام عدد 2 كونتاكتور أحدهما للاتجاه الأمامي والآخر للاتجاه الخلفي مع حماية لعدم التداخل في التشغيل بينهما وحماية أوفرلود للمحرك



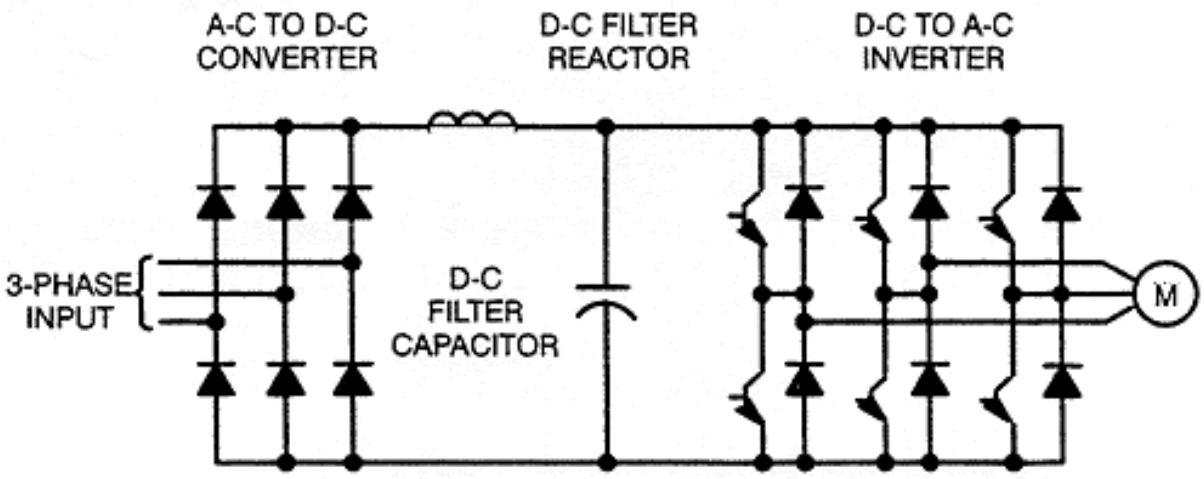
مثال 5 :

تشغيل وإيقاف محرك يعمل بنظام تشغيل/إيقاف باستخدام عدد 2 مفتاح Push Button ويعمل المحرك في اتجاهين ويعكس الاتجاه بشكل آلي ومستمر باستخدام عدد 2 مفتاح حد أحدهما للاتجاه الأمامي والآخر للاتجاه الخلفي مع اختيار أحد الاتجاهين بشكل افتراضي ودائم يبدأ فيه المحرك بمجرد التشغيل باستخدام عدد 2 ريلاي وعدد 2 كونتاكتور



أما عن التحكم في سرعة وعزم دوران محرك التيار المتردد فيتم باستخدام مغير سرعة Inverter حيث يمكن تغيير التردد والتحكم في تيار الحمل الأقصى للمحرك وبالتالي يمكن التحكم في السرعة والعزم

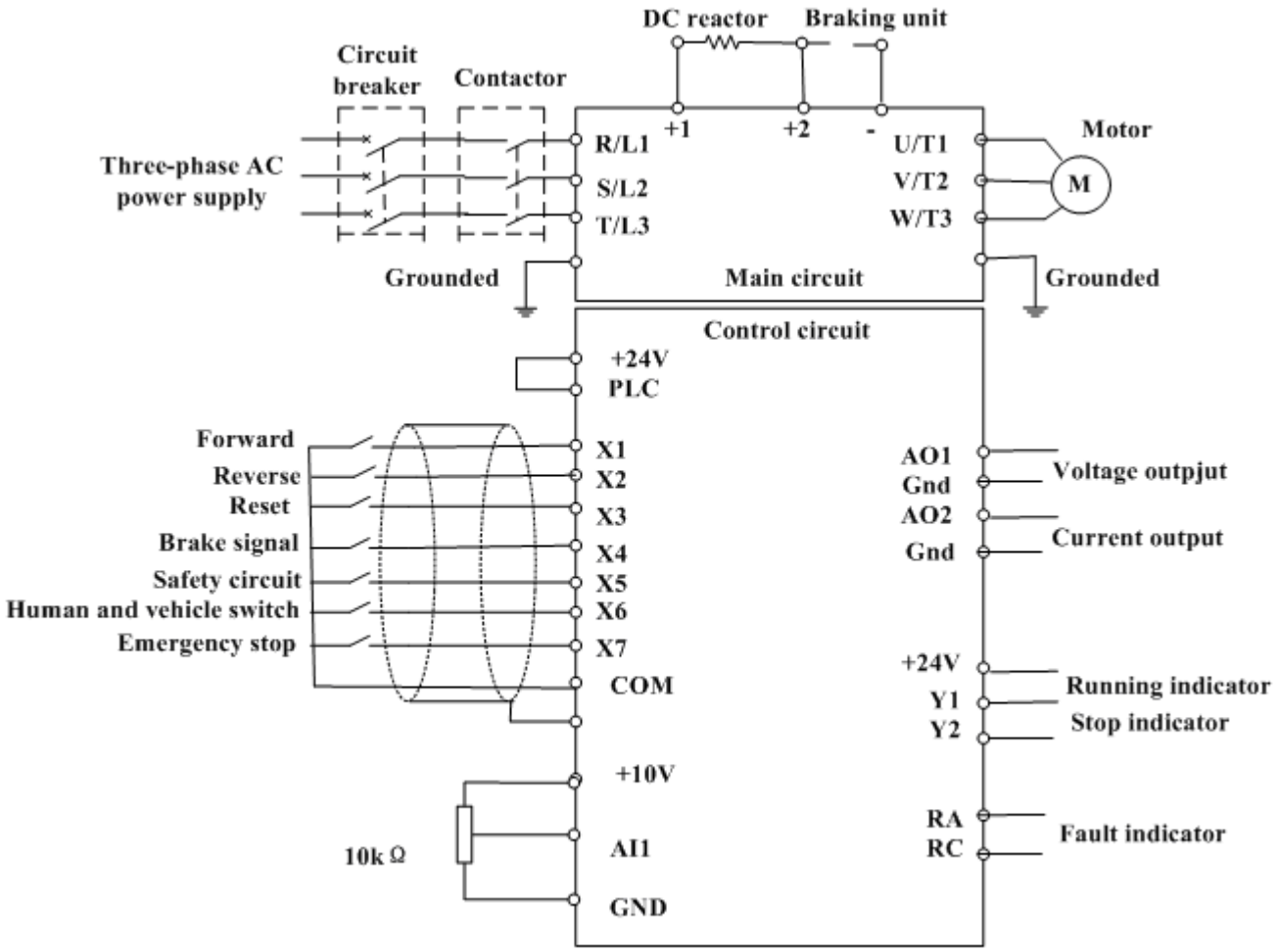
ومغير السرعة Frequency Inverter هو جهاز يستقبل القدرة الكهربية إما في شكل أحادي الفاز أو ثلاثي الفاز ويقوم بتوحيدها أولا عن طريق موحد داخلي Rectifier ثم يقوم بعد ذلك بتحويل القدرة المستمرة التي تم توحيدها إلى قدرة مترددة ثلاثية الفاز ولكن شكل الموجة يكون مربع Square wave وليس موجة جيبية Sin wave حيث يمكن التحكم في كل من قيمة الجهد RMS Value والتردد Hz وتيار الحمل الأقصى المسموح به Maximum current ولا يتجاوزه المحرك بأي حال



ولكي تتم هذه العملية فلابد من توافر متطلبات محددة كحد أدني ليكون مغير السرعة قادرا على تنفيذ المتطلبات بالإضافة إلى وحدة إنتاج القدرة الكهربية التي سبق توضيحها في الفقرة السابقة وهذه المتطلبات تتلخص في الآتي :

- مدخلات رقمية تسمح بالتحكم في تشغيل/إيقاف/عكس اتجاه قدرة تشغيل المحرك
- مخرجات رقمية تسمح بمراقبة تشغيل المحرك مثل الجاهزية Ready والتشغيل Run والسرعة صفر Zero Speed والخطأ إن وجد Fault/Error
- مدخلات تناظرية Analog inputs تسمح بتحديد السرعة المطلوبة وتيار الحمل المطوب وقراءة السرعة الفعلية Actual value
- مخرجات تناظرية Analog outputs تسمح بمراقبة وضع التشغيل الحالي للسرعة والتيار Actual current and speed
- إمكانية ضبط بيانات المحرك وبيانات التشغيل وغيرها عن طريق مجموعة من البيانات الحاكمة والمراقبة للتشغيل Parameters والتي من خلالها يمكن ضبط تشغيل مغير السرعة
- وحدة مراقبة للتشغيل والأعطال حيث يمكن التعرف على كل بيانات التشغيل من خلال شاشة رسائل أو لمبات بيان لتحديد تلك البيانات الضرورية اللازمة للتشغيل

مثال على توصيلات مغير سرعة



إن توفرت تلك المتطلبات فإننا وبكل بساطة يمكن أن نعتمد على الوحدة في التشغيل بالإضافة إلى عمليات تحكم إضافية يتطلبها الأمر حسب تطبيقات خاصة يمكن أن تتواجد في مغيرات السرعة والتي توفرها معظم الشركات حاليا حيث تعطي تلك الشركات عمليات تحكم أو تطبيقات جاهزة يمكن استخدامها بشكل مباشر في مغير السرعة مثل :

- التشغيل المعتاد مع التحكم في السرعة (تشغيل/إيقاف/عكس اتجاه حركة) Speed Control
- التحكم في العزم Torque/Current Control
- منظومة PID Control – PID
- السرعات الثابتة Fixed Speeds
- الزيادة/النقصان في السرعة Increment/Decrement
- التشغيل عبر وسائل الاتصال Communication

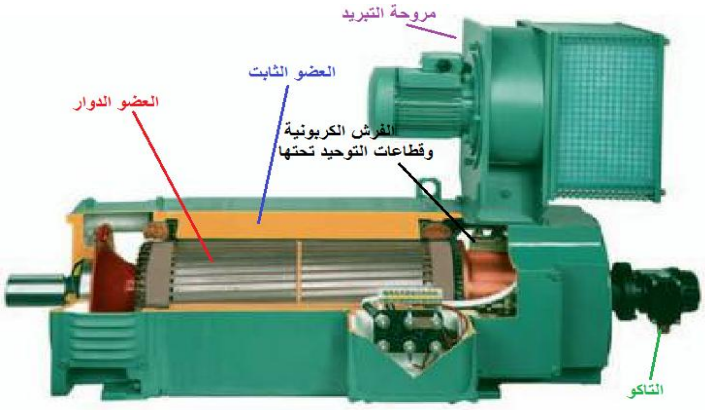
ومن المهم جدا أن نعتني بإدخال بيانات المحرك لمغير السرعة حتى نضمن تشغيلا أفضل للمحرك ، كذلك إدخال بيانات مصدر الجهد والتي تضمن حماية للجهاز ومراقبة جيدة للتشغيل.

وحسب التطبيق المطلوب لنا تنفيذه يتم البحث أولا في إمكانيات مغير السرعة وهل يوفي متطلباتنا أم لا وهل التطبيق ضمن التطبيقات الافتراضية به أم لا فإن كان ضمنها فمن الأفضل أن نبدأ بالموجود ثم نقوم بضبط البيانات لتعطي ما نريد وإلا قمنا بضبط الأقرب حسب متطلباتنا لنحصل في النهاية على التشغيل الأمثل للمحرك المطلوب تشغيله.

وقد تختلف بعض المسميات بين الشركات لكن في النهاية نبحث عن المعنى الذي نريده ومن خلال الكتالوجات وهناك العديد من الشركات التي لها وحدات مغيرات سرعة في السوق لهما سمعتها الطيبة مثل سيمنس وألن برادلي وباركر وإيه بي بي وشنايدر وميتسوبيشي وإل جي وياسكاوا وتوشيبا وهيتاشي ودلتا وغيرها ويفضل الاطلاع جيدا على الكتالوج الخاص بالاستخدام قبل الشروع في الاستخدام وفهم العملية الصناعية جيدا والتوفيق بينها وبين إمكانيات مغير السرعة المطلوب تشغيله

وبهذا نكون قد تحكمنا في الخصائص التي ذكرناها عن المحرك (التشغيل/الإيقاف/عكس الحركة) ثم السرعة وعزم الدوران

ثانيا : محركات التيار المستمر DC Motors



يتكون محرك التيار المستمر من عنصرين رئيسيين يتم التحكم في المحرك عن طريق التحكم فيهما وهما العضو الثابت والذي يشمل ملفات الفيلد وجزء من ملفات الأرميتشر والتي توجد غالبيتها في العضو الدوار كما توجد الفرش الكربونية والتي تقوم بعملية نقل القدرة الكهربية للعضو الدوار عبر قطاعات التوحيد كما توجد مروحة تبريد ولقياس السرعة يستخدم جهاز تاكو ميتر أو إنكودر حسب الاحتياج.

ومحرك التيار المستمر له جزءان من التغذية الكهربية الأول خاص بملفات الفيلد والثاني خاص بملفات الأرميتشر وتغذية الفيلد غالبا ما تكون قليلة القدرة في المحركات الصغيرة حيث لا يتجاوز استهلاكها أو قدرتها 0.5 كيلو فولت أمبير وربما أقل ويزيد هذا في المحركات الكبيرة ليصل تيار الحمل إلى 20 أمبير في بعضها.

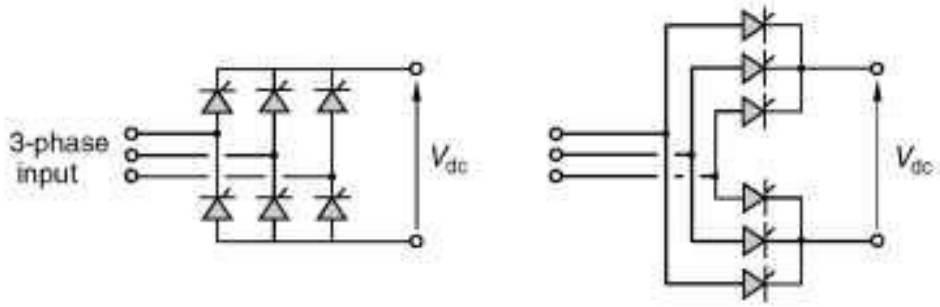
أما في ملفات الأرميتشر فهي التي يتم على أساسها احتساب قدرة المحرك ويختلف جهد التشغيل وتيار الحمل بناء على قدرة المحرك وتقاس قدرة المحرك بحاصل ضرب جهد التشغيل الأقصى في تيار الحمل الأقصى للمحرك.

وعن طريق التحكم في كل من جهد وتيار الحمل في ملفات الفيلد والأرميتشر يمكن التحكم في تشغيل المحرك كالآتي :

- يمكن التشغيل والإيقاف بتوصيل أو فصل القدرة الكهربية
- يمكن عكس اتجاه الحركة عن طريق عكس أي من الملفات الفيلد أو الأرميتشر
- يمكن التحكم في سرعة الحركة عن طريق قيمة فرق الجهد على ملفات الفيلد أو الأرميتشر
- يمكن التحكم في العزم عن طريق التحكم في فرق الجهد وتيار الحمل المسموح به على كل من ملفات الفيلد أو الأرميتشر

ويمكن أن يتم الشتغيل على سرعة ثابتة مثلما هو الحال مثلا في محرك مساحة زجاج السيارة حيث يتم التغذية بفولت 24 فولت ويتم التوصيل أو الفصل ويمكن عكس الاتجاه عن طريق عكس قطبية الملفات كما ذكرنا.

أما في حالة التغيير في جهد التغذية لكل من الفيلد أو الأرميتشر فيلزمنا هنا جهاز لتوحيد القدرة الكهربية ويمكن التحكم فيه حسب المطلوب وهو ما يطلق عليه DC Drive أو مغير سرعة تيار مستمر حيث يعتمد تماما على وحدات ثيرستور يتم توحيد القدرة الكهربية والتحكم فيها عن طريق التحكم في زاوية الإشعال للثيرستور



13

ومن المهم جدا أن نعلم المعلومات الآتية عن محركات التيار المستمر :

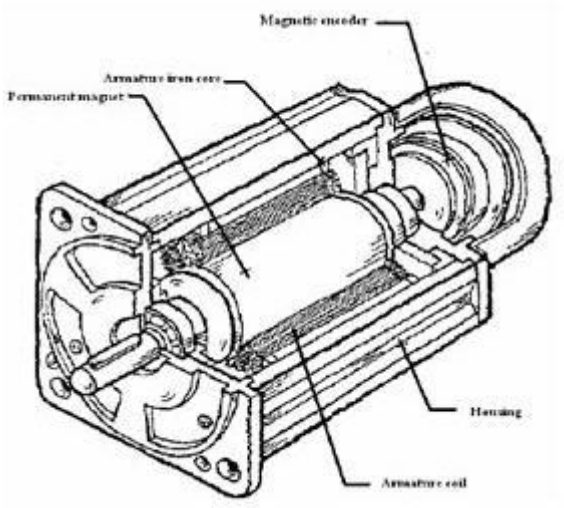
- تتناسب سرعة المحرك طرديا مع فرق الجهد على ملفات الأرميتشر
- نحصل على أقصى قدرة من المحرك وبالتالي أقصى عزم عند أقصى سرعة للمحرك
- زيادة المجال الكهربي الناتج عن ملفات الفيلد يؤدي إلى زيادة عزم الدوران ويكون الحد الأقصى للمحرك عند الحد الأقصى لتغذية ملفات الفيلد والأرميتشر

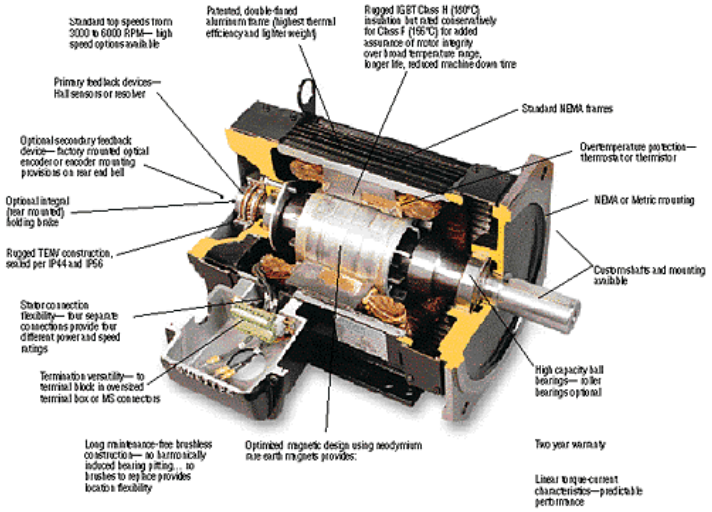
وبالتالي فإن أي درايف أو مغير سرعة لمحرك تيار مستمر سوف نجد به منظومة للتحكم في الفيلد وأخرى للتحكم في الأرميتشر وفي أحيان كثيرة يلجأ الكثيرون لتثبيت تغذية الفيلد إلا أن هذا لا ينفي أنه يمكن عن طريق التحكم في تغذية ملفات الفيلد زيادة عمليات التحكم في محرك التيار المستمر.

ولا يختلف الحال بالنسبة لمغير السرعة التيار المستمر عن التيار المتردد في الحد الأدنى للإمكانيات المتوقعة فيه والتي ذكرناها في معرض حديثنا عن مغيرات السرعة التيار المتردد.

ثالثا : المحركات ذات المغناطيس الدائم (بدون ملفات فيلد) – المحركات السيرفو :

النوع الثالث من المحركات وهي التي تستبدل ملفات الفيلد بمغناطيس دائم يعطي مجالا بديلا عن ذلك الذي تعطيه الملفات الكهربية وهو موجود أيضا بقسميه ذات التيار المتردد وذات التيار المستمر وله مميزات كبيرة عن المحركات العادية حيث التحكم العالي في السرعة ودقة الانتقال بين السرعات والوصول إلى سرعات عالية مقارنة بالمحركات العادية كما يمكن عكس الحركة ببساطة وبسرعة كبيرة في وقت قصير جدا كما أن الحجم مضغوط جدا مقارنة بالمحرك العادي إضافة إلى أن عزم الدوران لنفس القدرة الكهربية أكبر وأكثر ثباتا.





14 وتنتج الشركات أجهزة مغيرات سرعة تتعامل مع المحركات السيرفو الخاصة بها ومعها وسيلة قراءة السرعة الفعلية عبارة عن إنكودر Encoder أو تاكو Tacho-generator حيث يكون هناك توافق تام بين المحرك ومغير السرعة.

وتبعا لنوع المحرك إن كان تيار متردد أو تيار مستمر يكون نوع مغير السرعة كما سبق ويخضع لنفس المقاييس غير أن بيانات التشغيل قد تكون أقل في مغيرات السرعة السيرفو من مغيرات السرعة العادية.

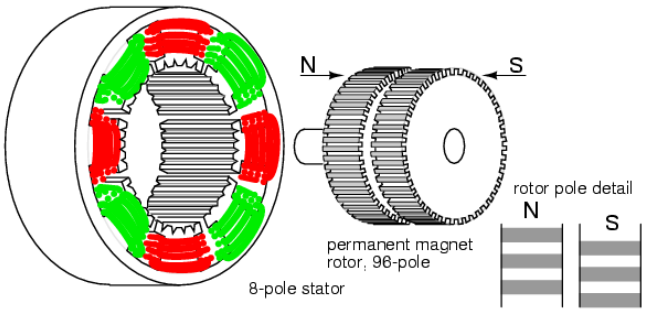
وما قلناه عن الحد الأدنى للمتطلبات في مغير السرعة العادي ينطبق تماما على مغير السرعة السيرفو.

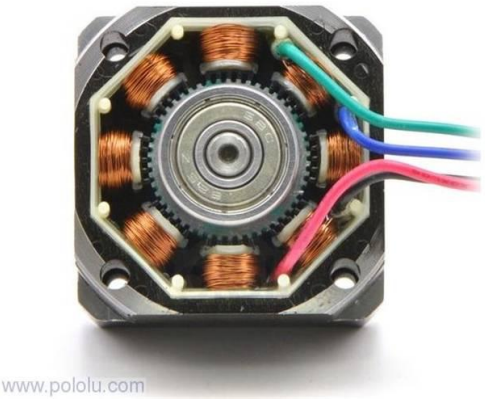
ونحتاج لمحركات السيرفو في التطبيقات التي تتطلب تغير سريع في السرعة أو عكس سريع للسرعة أو سرعات عالية ودقيقة في نفس الوقت وكذلك التطبيقات التي تطلب عزم أعلى وحجم أقل.

إلا أن من عيوبه ارتفاع السعر مقارنة بالمحركات العادية وكذلك صعوبة عمليات الصيانة مقارنة بالمحركات العادية بل إن بعض المحركات لا يمكن إصلاحها نهائيا عند حدوث أي مشكلة بها.

رابعا : محركات الخطوة Step Motors :

وهي من أنواع المحركات السيرفو غير أنها تتميز بميزة أخرى وهي التحرك بزواية محددة ودقيقة جدا وتصل إلى أجزاء من الثانية في الحركة الواحدة حسب دقة حركة المحرك والذي يعمل معه مغير سرعة من نفس النوع Step drive حيث يكون الخرج من درايف الخطوة في شكل نبضات قدرة كهربية كل نبضة يتحرك بها المحرك حركة ثابتة تعادل جزء من دقة المحرك والتي يمكن أن تصل في بعض المحركات أن يتحرك الدورة الواحدة (2ط) في 10000 خطوة وهي دقة عالية جدا يتطلب العمل بها في الماكينات التي تحتاج لمثل تلك الدقة مثل ماكينات CNC والروبوت وغيرها.





ويختلف التشغيل هنا في أن مغير السرعة يتطلب نبضات Pulse Train بالإضافة إلى إشارة التشغيل حيث مع وجود الاثنان يتحرك بحركة واحدة

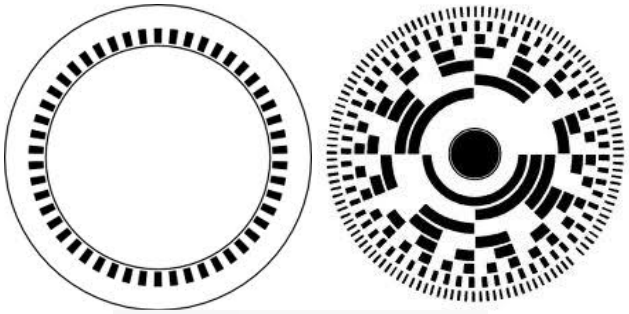
وهو يعطي من الدقة العالية في الحركة الموضعية ما لا يمكن لغيره أن يعطيه

أنواع أجهزة قراءة السرعة الفعلية للمحركات Speed feedback devices :

التاكو Tacho-generator

وهو جهاز يعطي الخرج منه في شكل جهد كهربي مستمر أو متردد يتناسب مع السرعة الدورانية والأشهر منه التيار المستمر ويكون الخرج منه منسوبا لكل 1000 لفة مثلا 40 فولت/1000 لفة أو 0.04 فولت/لفة بالإضافة إلى أقصى عدد ممكن من اللفات يمكن أن يعمل به الجهاز ومنه أشكال كثيرة سواء لها عمود محور يتحرك مع المحرك أو مجوف Hollow shaft

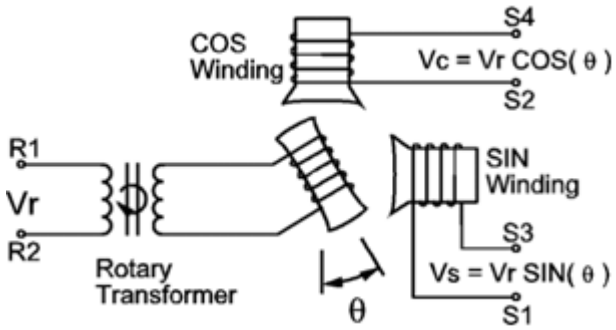
الانكودر Encoders

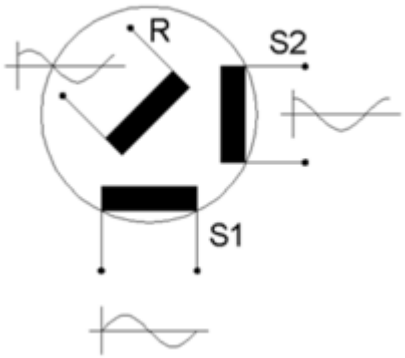


وهي أجهزة تعتمد على إنتاج نبضات Pulse عند دورانها اعتمادا على قرص دوار يمر أمام حساس ضوئي مثل المبين بالشكل ومنها نوعان الأول يسمى Absolute encoder حيث يكون هناك كود لكل زاوية من زوايا الدوران ويكون بها قرص مثل الذي إلى اليمين والذي يعطي كود مكون من 8 خانات أما النوع الآخر وهو Incremental ويعطي نبضات Pulse train على قناتين A,B بفارق زمني 90 درجة وعلى Z يعطي نبضة لموضع البداية في كل مرة ويمكن أن يعطي ثلاث قنوات وعكسها أو قناتين فقط أو قناة واحدة وعكسها.

Resolver

وهو جهاز يستخدم في تحديد زاوية الدوران ويعطي إشارة تناظرية وليس كود رقمي حيث يعطي ثلاث إشارات لتحديد الزاوية كما في الصورة المقابلة ويستخدم في حالات التحكم في الوضع Position control مثلما هو الحال مع Absolute encoder





وهذه الأجهزة والتي تعطي صورة عن السرعة الفعلية أو وضع زاوية الدوران للمحرك تساعد في الوصول إلى الشكل الأمثل أو الأفضل للأداء المطلوب من مغير السرعة والمحرك العامل معه.

بهذا نكون قد ألقينا الضوء على المحركات الكهربية بأشكالها المختلفة والعوامل التي يمكن أن نتحكم فيها في المحركات والأجهزة اللازمة لتنفيذ مهام التحكم خاصة مغيرات السرعة وكيفية قياس السرعة الفعلية للمحركات. 16

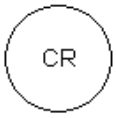
صياغة منظومة التحكم

بعد أن تعرفنا على بعضا من عناصر منظومة التحكم في العمليات الصناعية المختلفة فإن يتحتم علينا أن نربط بين تلك العمليات بشكل ما لتؤدي ما نريده منها وهناك أشكال كثيرة للتفكير في صياغة منظومة التحكم المناسبة وكلها حسب كلامنا السابق تأخذ صورتين إما صورة رقمية Digital أو صورة تناظرية Analog

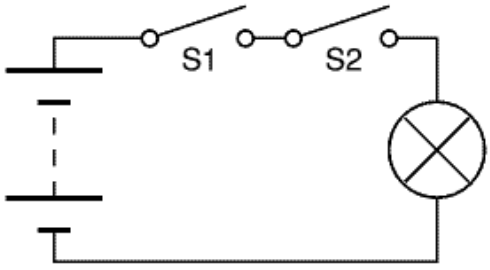
ففي الصورة الرقمية تخضع العملية تماما لعلاقات الجبر الثنائي Boolean algebra المعروفة حيث تتمثل الحالات 0,1 في شكل نقطة التوصيل التي يمثلها الريلاي أو الكونتاكتور بكل أشكالهما ويمثل ملف الريلاي أو الكونتاكتور أو اللمبة أو الصمام مكان تخزين النتيجة التي نريدها تماما

وبالتالي يمكننا أولا التعبير عن الحالة في صورة نقطة مفتوحة NO أو نقطة مغلقة NC

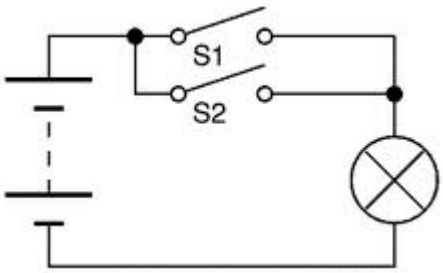
- تخزين نتيجة العمليات (في شكل ملف ريلاي)
- نقطة مفتوحة (تمثل الحالة 1 حالة التفعيل)
- نقطة مغلقة (تمثل الحالة 0 أو المعكوس حالة التفعيل)



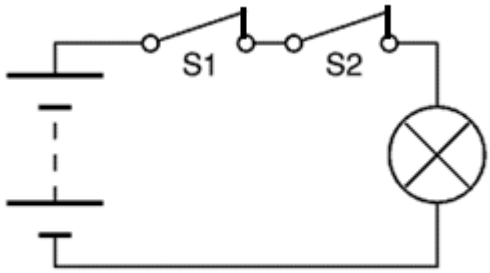
وتنطبق قواعد الجبر الثنائي تماما فالتوصيل على التوالي يعطي العلاقة AND والتوصيل على التوالي يعطي العلاقة OR والنقطة NC هي المقابل للنقطة NO والعكس صحيح



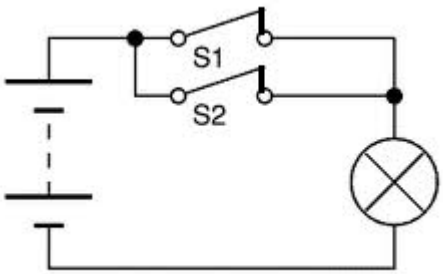
التوصيل على التوالي AND



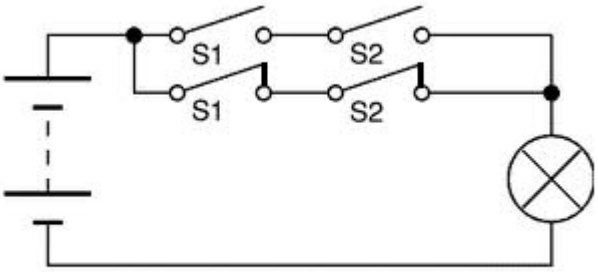
التوصيل على التوازي OR



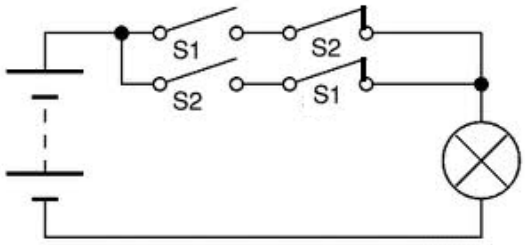
التوصيل على التوالي NAND



التوصيل على التوالي NOR



التوصيل بشرط أن يتحقق الشرطان لتحقيق الخرج

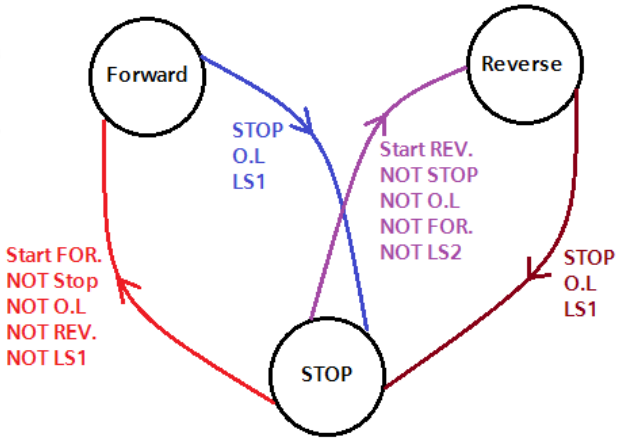


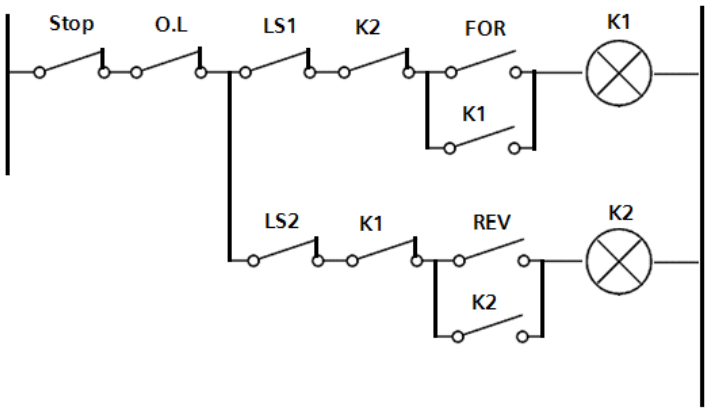
التوصيل بشرط أن يختلف الشرطان لتحقيق الخرج

ويمكن الخلط بين العمليات وبعضها بشرط أن تعطي العملية النتيجة المطلوبة ويمكن تكوين العمليات عن طريق صياغة جدول الحقائق Truth table حيث يعطي النتائج المحتملة بناء على حالة المدخلات ومنه يمكن صياغة العمليات.

هناك طريقة أخرى لصياغة عمليات التحكم الرقمي وهي طريقة الحالة أو التحكم بالحالة State Control حيث يتم تحديد حالات للعملية ويكون الانتقال بينها عبر شروط يجب تحقيقها في كل حالة ليتم الانتقال إلى حالة أخرى.

مثلا محرك يعمل في اتجاهين يكون له ثلاث حالات وهي (التوقف – العمل الأمامي – العمل الخلفي) وهناك شروط فمثلا يتم الانتقال من التوقف إلى العمل الأمامي عند الضغط على مفتاح العمل الأمامي بشرط عدم الضغط على التشغيل الخلفي أو الضغط على التوقف ويستمر العمل على هذا واستمرارا لحالة التشغيل الأمامي ، وحالة التشغيل الخلفي أيضا يتم الانتقال من التوقف إلى العمل الخلفي عند الضغط على مفتاح العمل الخلفي بشرط عدم الضغط على التشغيل الأمامي أو الضغط على التوقف ويستمر العمل على هذا واستمرارا لحالة التشغيل الخلفي ، أما حالة التوقف فيتم الوصول إليها من حالة التشغيل الأمامي أو الخلفي إن ضغطنا على التوقف أو حد خطأ أوفرلود أو وصل المحرك إلى نقطة مفتاح الحد Limit switch الخاص بنهاية منطقة التشغيل الأمامي أو وصل المحرك إلى نقطة مفتاح الحد Limit switch الخاص بنهاية منطقة التشغيل الخلفي ولا ينتقل التشغيل من التشغيل الأمامي إلى الخلفي أو العكس أبدا.

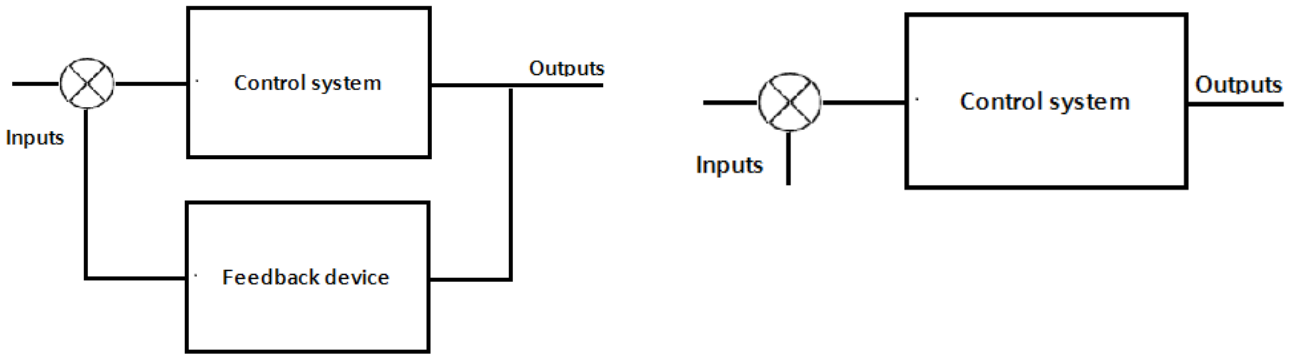




وتصميم عمليات التحكم باستخدام مخطط الحالة من الطرق المفيدة جدا في عمليات تصميم دوائر الماكينات خاصة تلك التي تعتمد على تتابع تنفيذ خطوات معينة حيث يكون الوصول إلى خطوة معينة مرتبطا بالحالة السابقة لها ويكون شرطا في الحالة التالية لها مع الشروط الأخرى الواجب توافرها للانتقال بين الحالات وبعضها بالصعود والهبوط.

وكل أنواع الريليهات سواء تايمر مؤقت أو عداد أو منظم عمل Process يمكن في النهاية ترجمة عمله إلى نقطة توصيل إما مغلقة NC أو مفتوحة NO وبالتالي يمكن تنفيذ الجزء الرقمي منها بعمليات الجبر الثنائي مثل ما سبق وفي الحالات المعقدة يمكن الاعتماد على التحكم باستخدام الحالة State control كما ذكرنا في الجزء السابق.

أما على مستوى الصورة التناظرية فهناك أشكال من التحكم منها التحكم المفتوح Open loop control حيث يتم تحديد قيمة للتشغيل دون الاعتماد على قياس نتيجة الأداء للوصول إلى نتيجة وبالتالي فوسائل القياس لا تكون موجودة في حين يجب توفر منظومة لتحويل الإشارة التناظرية إلى "فعل" عن طريق جهاز تحكم مناسب أو أداة تنفيذ مناسبة.



وأنظمة التحكم المغلق لها طرق كثيرة لتنفيذها منها :

- التحكم عن طريق الفرق في القيمة Differential حيث يتم تحديد قيمة صغرى وقيمة كبرى حيث يتم التشغيل عند القيمة الصغرى والإيقاف عن القيمة الكبرى
- التحكم عن طريق مكبر PID حيث يتم قياس الخطأ بين قيمة الضبط والقيمة الفعلية وحسب حساسية الخطأ يتم التصرف بثلاث طرق متوازية أو متتالية أحدها تكبير والأخرى تفاضلية والثالثة تكاملية ويوجد من منظومات PID أشكال كثيرة سواء في شكل كروت إلكترونية أو في شكل برامج حاسبات أو حاكمات.
- التحكم التقاربي حيث يتم التشغيل بقياس الفرق بين قيمة الضبط والقيمة الفعلية وكلما زاد الخطأ زادت سرعة استجابة النظام للعمل وكلما صغر الخطأ كانت نسبة الاستجابة أبطأ إلى أن يصل إلى أقل معدل مسموح به وعندها يتوقف النظام عن الاستجابة أويصل إلى مرحلة الثبات في التشغيل وهو ما يمكن أن يسمى Fuzzy control ويمكن استخدامه في العمليات التي لا يمكن صياغة علاقة رياضية لربط مكوناتها معا.

19

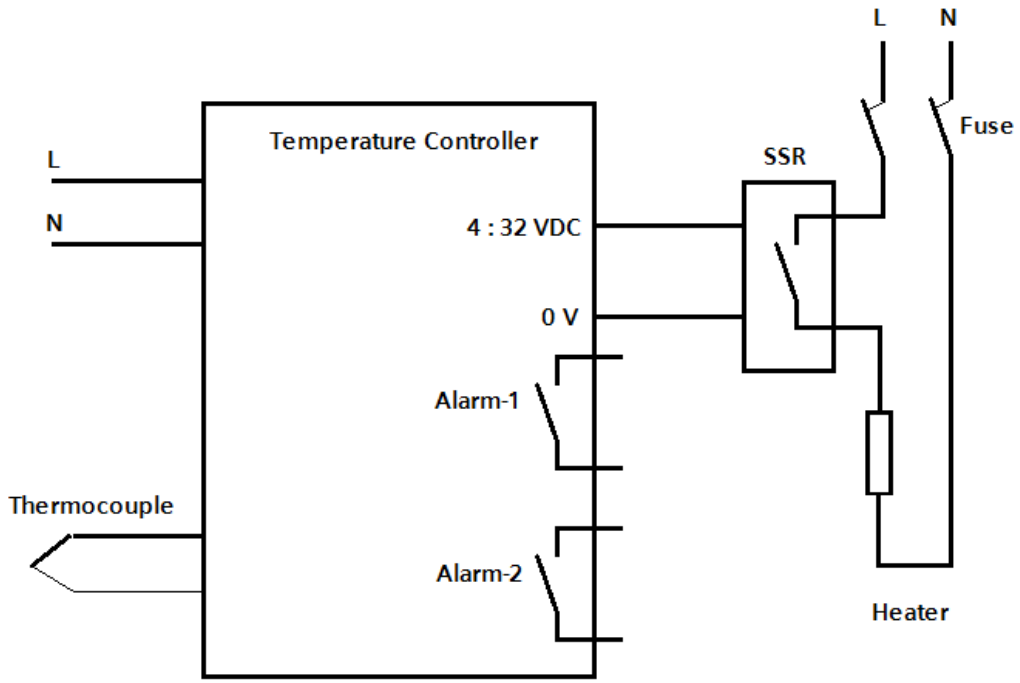
أمثلة على منظومات التحكم المغلق Closed Loop Control System

- منظومة تحكم في درجة الحرارة Temperature controller باستخدام PID
  - عناصر المنظومة (جهاز تحكم في الحرارة – حساس ثرموكابل – SSR – سخان – حماية)
  - الرسم الكهربي للدائرة
  - ماذا نضبط في جهاز التحكم
- منظومة التحكم في مستوى سائل باستخدام فارق المستوى Differential controller
  - عناصر المنظومة (جهاز تحكم في المستوى – عوامة حد أقصى – عوامة حد أدنى – كونتاكتور – مفتاح تشغيل/إيقاف – حماية)
  - الرسم الكهربي للدائرة
  - ماذا نضبط في جهاز تحكم المستوى
- منظومة التحكم في سرعة موتور مع دانسر (من 0 وحتى 10 فولت) باستخدام مغير سرعة Inverter
  - عناصر المنظومة (مغير سرعة – موتور – منظومة تشغيل/إيقاف – قيمة لضبط السرعة الخطية – دانسر – حماية)
  - الرسم الكهربي للدائرة
  - ماذا نضبط في جهاز مغير السرعة والدانسر

دائرة سخان مع ثرموكبل وريلاي SSR مع حماية فيوز أو قاطع

مكونات الدائرة

- مصدر التغذية L,N
- جهاز التحكم في الحرارة
- ريلاي SSR
- حساس ثرموكبل
- حماية فيوز أو قاطع
- سخان

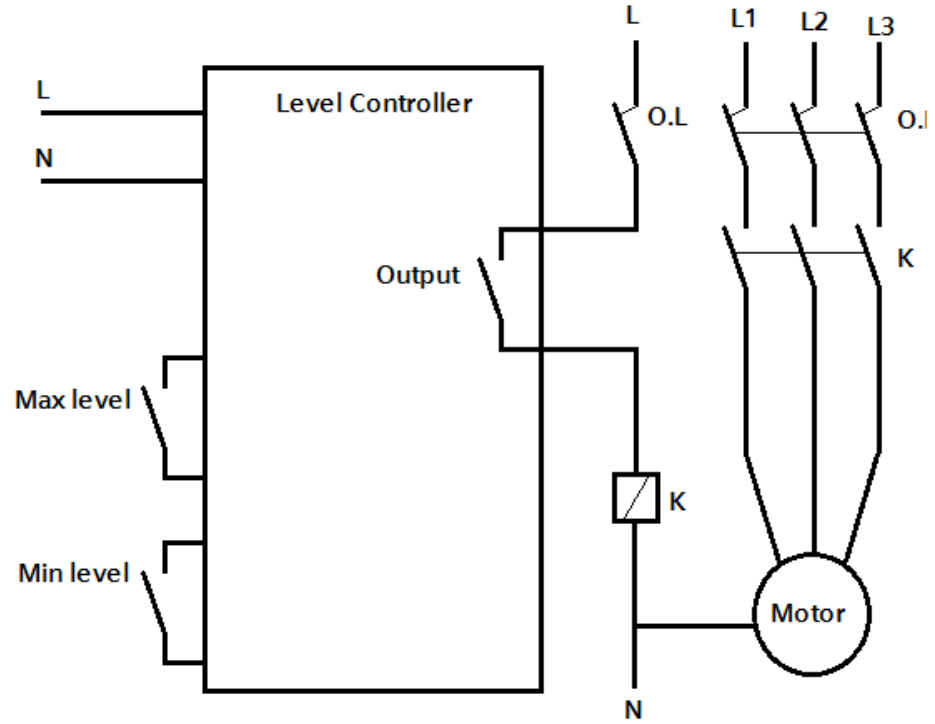


- **ما هي البيانات الواجب ضبطها في جهاز التحكم في الحرارة للحصول على أداء سليم ؟**

20 دائرة تحكم في مستوى سائل باستخدام جهاز تحكم في المستوى Level Controller مع عدد 2 عوامة وكونتاكتور لتشغيل طلمبة لملئ الخزان مع حماية أوفرلود أو فيوز ثلاثي

مكونات الدائرة

- جهاز تحكم في المستوى
- حساس حد أعلى للمستوى
- حساس حد أقصى للمستوى
- موتور طلمبة
- كونتاكتور للتشغيل
- حماية أوفرلود

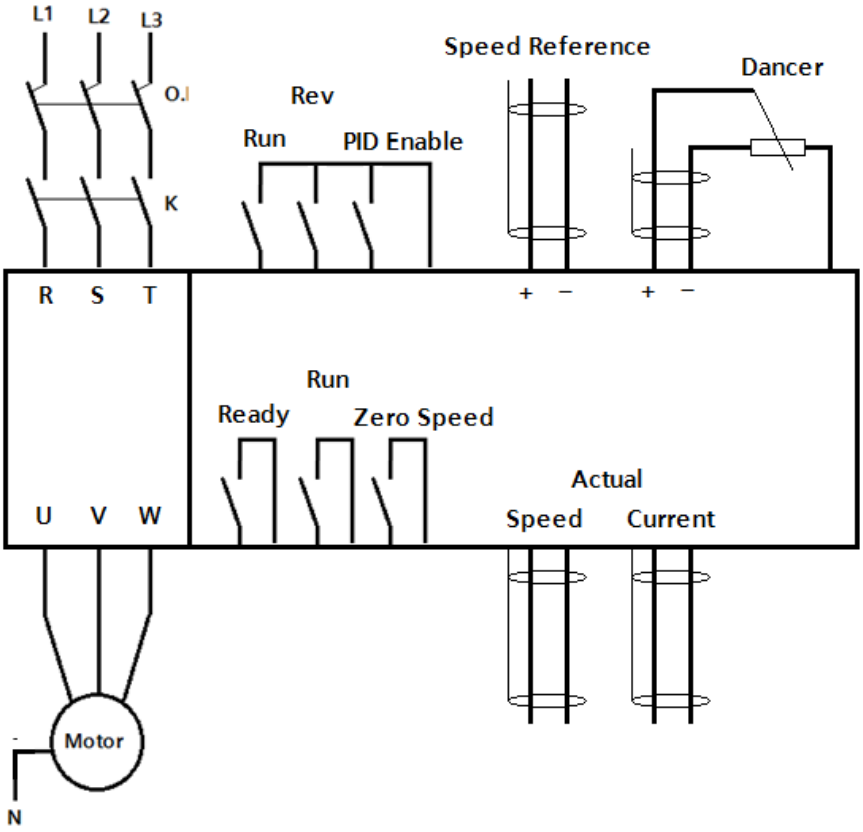


- **ما هي البيانات الواجب ضبطها في جهاز التحكم في المستوى للحصول على أداء سليم ؟**

دائرة تشغيل محرك مع مغير سرعة مع دانسر للحصول على Feedback لتصحيح السرعة

مكونات الدائرة

- مغير سرعة
- موتور
- دانسر
- كونتاكتور توصيل
- حماية أوفرلود
- مصدر للسرعة

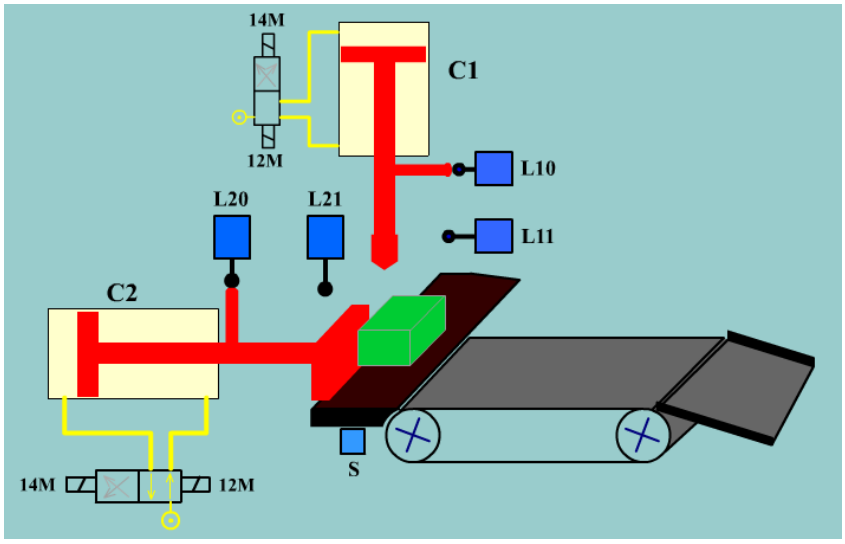


- **ما هي البيانات الواجب ضبطها في جهاز مغير السرعة للحصول على أداء سليم مع تشغيل الدانسر ؟**

21

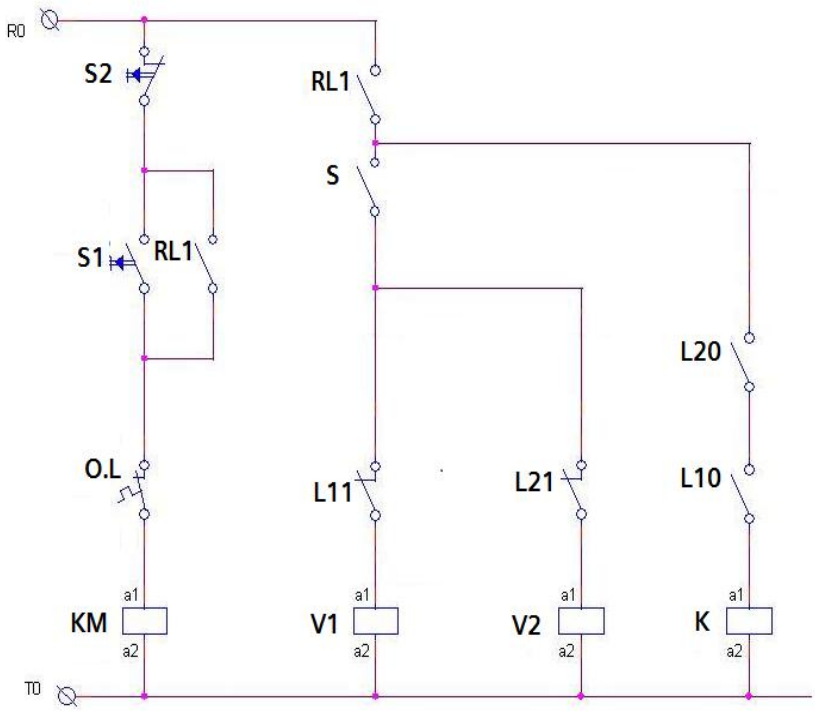
أمثلة على عمليات التحكم Control Examples

مثال-1 :



منظومة طباعة على جزء حيث تتكون المنظومة من موتور يتحرك باستمرار مع التشغيل وعند دفع القطعة يدويا حتى تصل إلى الحساس S يتم نزول البستم C1 عن طريق الصمام 14M ليضع العلامة فوق القطعة فيتحرك إلى موضع الحساس L11 ثم يتحرك إلى الموضع L10 فيقوم البستم C2 بدفع القطعة من الموضع L20 إلى الموضع L21 عن طريق الصمام 12M ثم يعود إلى الموضع L20 ثم تبدأ الدورة الجديدة

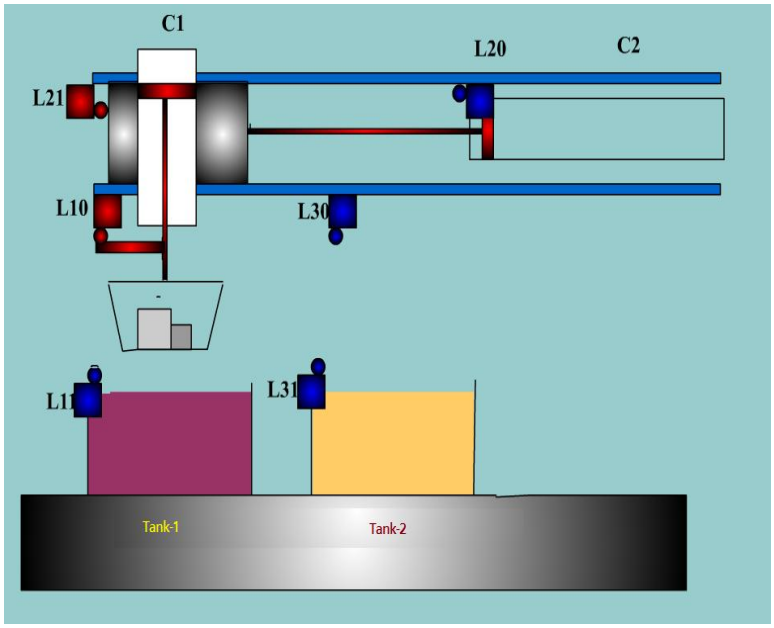
مكونات الدائرة :



- موتور M يعمل مع تشغيل الماكينة ويقف معها
- تشغيل وإيقاف بمفتاحين Start/Stop مع حماية أوفرلود للموتور
- حساس S حيث موضع الطباعة
- بستم C1 يعمل عن طريق الصمام V1 وعليه حساسان L11 للحد الأسف للحركة و L10 للحد الأعلى للحركة
- بستم C2 يعمل عن طريق الصمام V2 وعليه حساسان L21 للحد الأمامي وL20 للحد الخلفي للحركة
- الريلاي K يعطي إشارة لجاهزية الماكينة لاستقبال قطعة جديدة

22

مثال-2 :

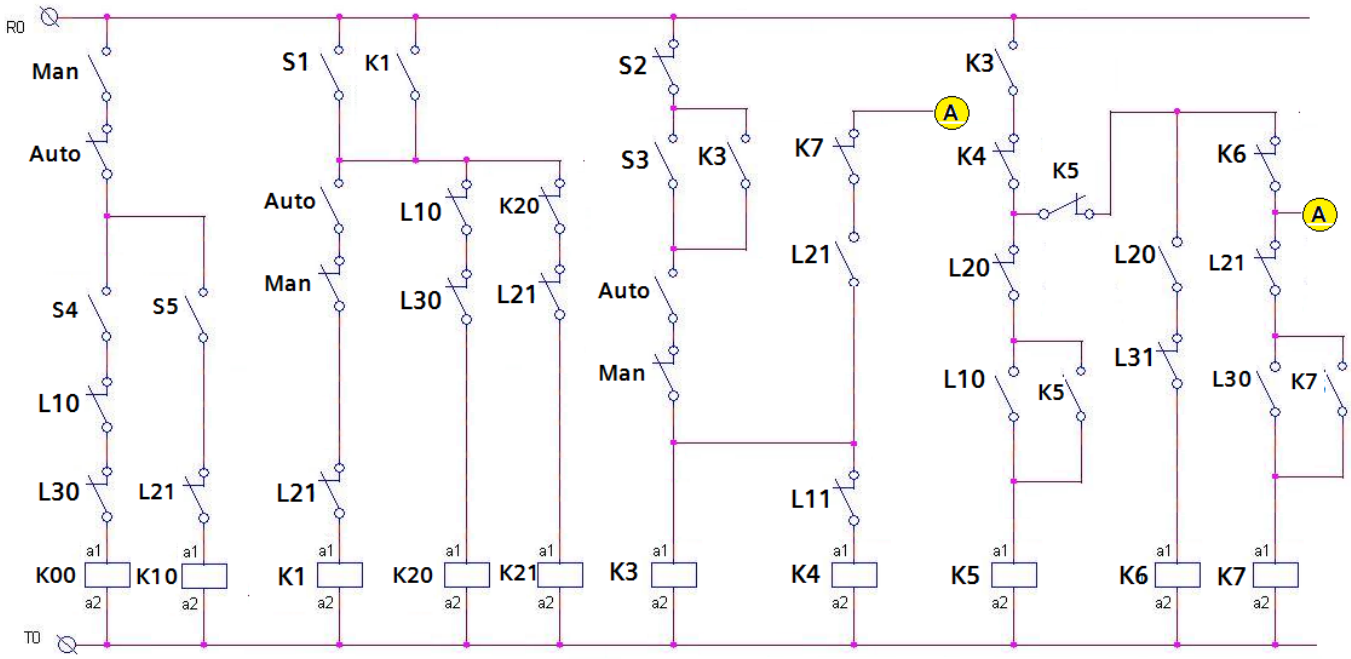


عربة تتحرك عن طريق بستم C2 بين خزانين حيث تبدأ الحركة من الخزان الأول حيث الوضع الأولي للحركة عند البدء أن تكون العربة أمام الحساس L21 وأن تكون العربة لأعلى أمام الحساس L10 وعند التشغيل يبدأ البستم C1 بالنزول حتى يغمس الجزء في الحوض الأول حتى الحساس L11 وبمجرد النزول يتم الرفع مرة أخرى حتى يصل للحساس L10 ثم تتحرك العربة عن طريق البستم C2 حتى تصل للحساس L30 ثم تنزل عن طريق البستم C1 حتى تصل إلى الحساس L31 ثم ترفع مرة أخرى حتى تصل للحساس L30 ثم تتحرك العربة مرة أخرى عن طريق البستم C2 مرة أخرى للوضع الأولي ويتم التكرار حتى الضغط على مفتاح التوقف

تم إضافة وضع تعديل Positioning لوضع العربة سواء آليا أو يدويا حيث يتم رفعها لأعلى مع حركة العربة لأقصى اليسار وبالتالي تكون جاهزة للتشغيل الآلي المتواصل حتى يتم الضغط على مفتاح التوقف

كل الشروط التي ترفع العربة لأعلى تقطع الهواء عن البستم C1 الخاص بالرفع فيعود للوضع لأعلى وكل الشروط التي تنزل العربة لأسفل تقوم بتوصيل الهواء إلى البستم C1 والذي يقود العربة لأسفل

كل الشروط التي تحرك العربة إلى اليسار توصل الهواء إلى البستم C2 وكل الشروط التي تحرك العربة إلى اليسار تقطع الهواء عن البستم C2



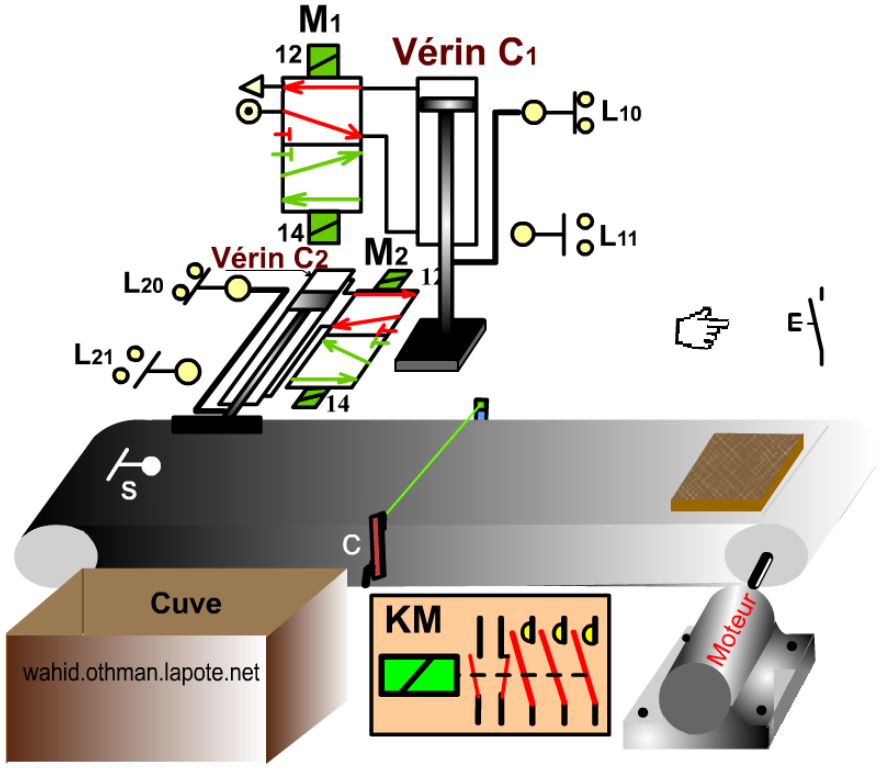
MAN : Switch Manual Initial Pos. Select
AUTO : Automatic Initial Pos. and Run Select
S1 : Auto Positioning
S2 : Stop Machine
S3 : Run Machine
S4 : Carriage Up Manual
S5 : Carriage Left Manual

K00 : Manual up C1
K10 : Manual Left C2
K1 : Automatic Positioning
K20 : C1 Automatic Up
K21 : C2 Automatic Left
K3 : Machine Run Automatic
K4 : C1 Down to L11

K5 : C2 Right to L20
K6 : C1 Down to L31
K7 : C2 Left to L21 and Restart sequence

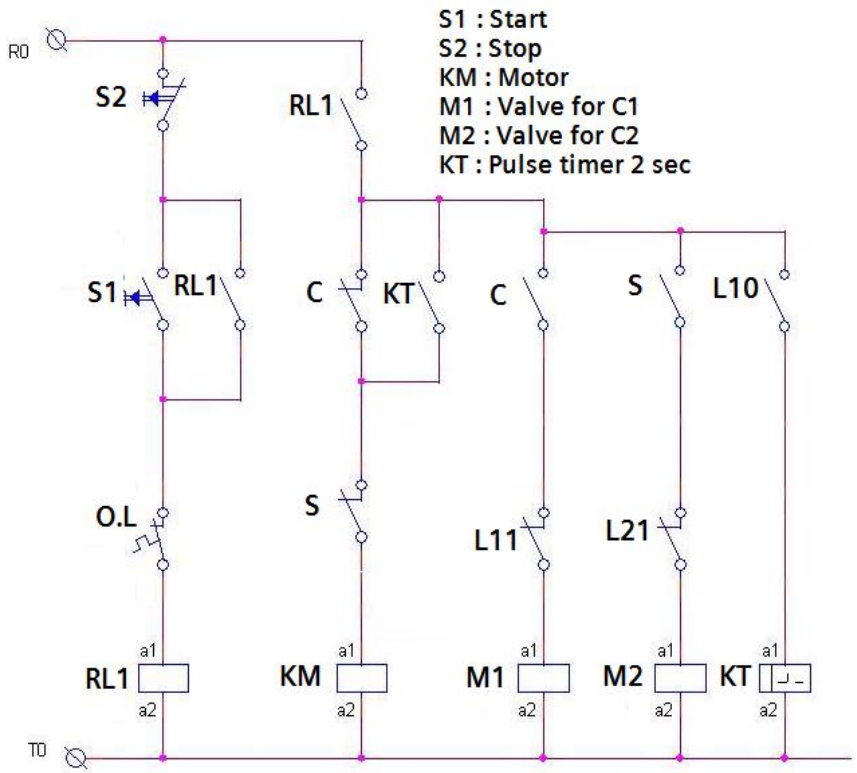
مثال-3 :

وحدة طباعة على حجر حيث يمر الحجر على سير متحرك عن طريق موتور يعمل عن طريق كونتاكتور KM ويتم إيقاف القطعة عند المرور أمام الحساس الضوئي C حيث ينزل البستم C1 عن طريق الصمام M1 حتى يصل للحساس L11 ثم يرفع البستم C1 حتى يصل إلى L10 فيتحرك المتور مرة أخرى حتى تصل القطعة إلى الحساس S فيتوقف الموتور فيدفعها البستم C2 حتى يصل إلى موضع الحساس L21 ثم يعود إلى الموضع L20 وتبدأ الدورة من جديد

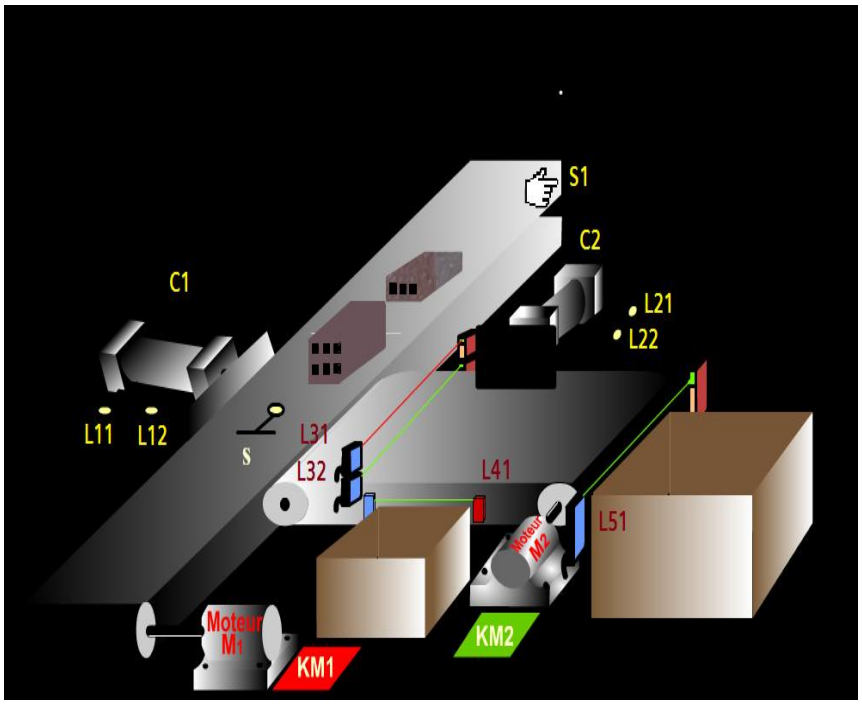


عناصر الدائرة :

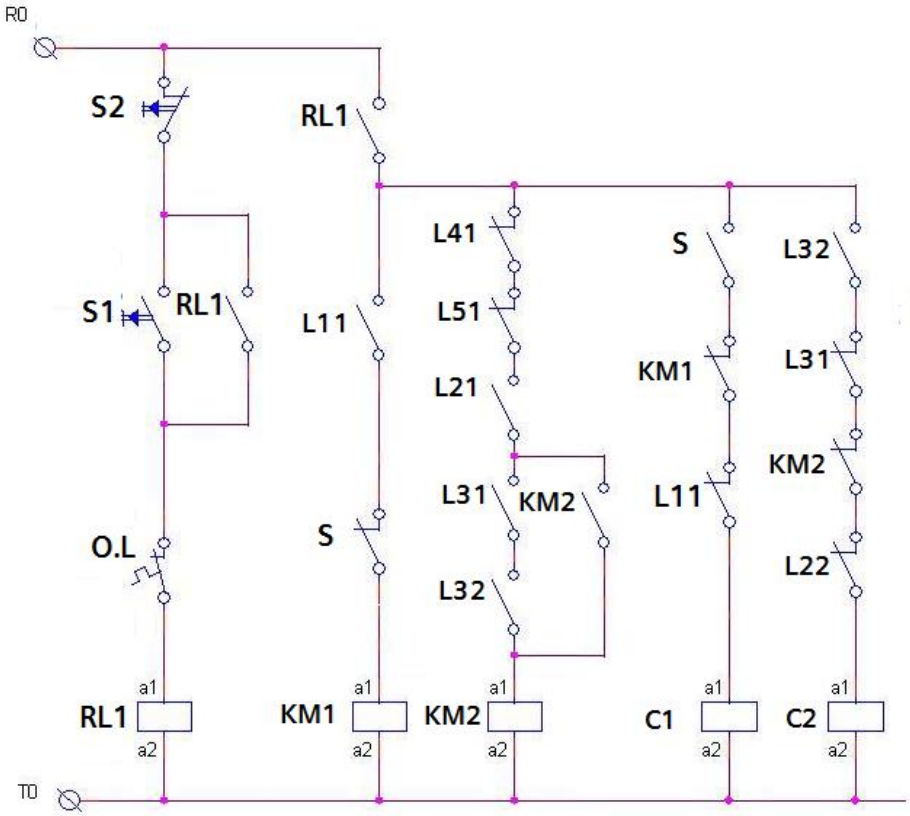
- موتور M يعمل عن طريق كونتاكتور KM
- بستم C1 يعمل عن طريق صمام M1
- بستم C2 يعمل عن طريق صمام M2
- مفتاح تشغيل S1 يبدأ التشغيل ومفتاح توقف S2
- حساس توقف القطعة للطباعة C
- حساس توقف القطعة للدفع للكرتونة S
- حساسات نهاية مشوار للبستم C1 فوق L10 وتحت L11
- حساسات نهاية مشوار للبستم C2 أمام L21 وخلف L20
- هنا يتم إضافة مؤقت Pulse لإعادة تشغيل موتور السير بعد الطباعة حتى يتجاوز الحساس C



مثال-4 :



وحدة تعبئة وفصل المنتج ذو الارتفاع الصحيح من ذو الارتفاع الخطأ حيث يعمل الموتور M1 لنقل المنتجات مع حتى يصل منتج إلى الحساس S فيتوقف الموتور M1 فيدفع البستم C1 القطعة حتى تمر من أمام الحساسين الضوئيين L31,L32 بفعل الدفعة فإن قطعت الاثنان كانت سليمة فيعمل السير الثاني عن طريق الموتور M2 ليضع المنتج السليم في الكرتونة فيمر من أمام الحساس L51 ليسمح للبستم C1 بدفع منتج جديد وإن مرت من أمام الحساس L32 فقط فلا يعمل الموتور M2 ويقوم البستم C2 بدفع المنتج غير المطابق إلى الكرتونة الأخرى ثم تبدأ دورة جديدة لدفع منتج من على السير ويلاحظ توقف السيور سواء M1 أو M2عند دفع المنتج من عليها



تتحرك القطعة على السير المتحرك بالموتور M1 حتى تصل إلى الموضع S حيث يتم الدفع لمنطقة الفرز عن طريق الذراع C1 فإن مرت عبر شعاعي الحساسين L31,L32 مرت إلى منطقة المطابق عبر الموتور M2 وإن مرت عبر الشعاع الخاص بالحساس L32 فقط تم دفعها بالذراع C2 إلى منطقة الغير مطابق وفي كل الحالات لا يتحرك أي من الموتورين طالما هناك أي جزء لم يدخل منطقة المطابق أو غير المطابق أو لم يكن الذراعان في المنطقة الخلفية

25

عمليات التحكم باستخدام مغيرات السرعة

تستخدم مغيرات السرعة بشكل عام للتحكم في المحركات بكل أشكالها سواء كانت AC أو DC أو سيرفو أو Step motor حيث بالتحكم في عوامل المحرك Motor parameters يمكن التحكم في المحرك.

فمثلا بالتحكم في جهد الأرميتشر في محركات التيار المستمر يمكن التحكم في السرعة وبالتحكم في التيار المسموح به للمرور عبر الملفات يمكن التحكم في العزم.

وبالتحكم في تردد التيار المتردد يمكن التحكم في سرعة المحرك وبالتحكم في فرق الجهد الكهربي وكذلك التيار يمكن التحكم في عزم الدوران للمحرك.

ولكي نقوم باستخدام مغير السرعة بشكل صحيح فيجب علينا المرور بخطوات محددة لذلك :

- توصيف مغير السرعة بشكل كامل بما يتناسب مع المحرك ومع التطبيق المزمع تنفيذه
  - تحديد القدرة وشكل الدخل الكهربي والخرج الكهربي من مغير السرعة
  - تحديد شكل التطبيق المطلوب
  - تحديد المدخلات والمخرجات الرقمية والتناظرية المطلوبة
  - تحديد نوع العزم (أحادي-رباعي) Single/Four Quadrant
- تثبيت وتوصيل مغير السرعة طبقا لتعليمات المورد حيث نبدأ التوصيل كالآتي:
  - توصيل مدخلات التغذية الرئيسية Main power
  - توصيل مدخلات التغذية الثانوية Auxiliary supply
  - توصيل أطراف المحرك Motor connections
  - توصيل وسيلة قياس السرعة Speed feedback
  - توصيل الشروط الضرورية للتشغيل مثل ثرموستات المحرك وشروط التوقف المفاجئ وغيرها
  - إدخال بيانات القدرة الكهربية والمحرك في مغير السرعة عن طريق شاشة التشغيل وقبل بدء التشغيل
  - إدخال بيانات المدخلات الرقمية والتناظرية قبل التشغيل
- تجربة تشغيل مغير السرعة مع الموتور بشكل يدوي محلي Local من لوحة التشغيل وذلك لضبط الاتجاه (قبل توصيل الحمل على المحرك) وتحديد قياسيات المحرك من حيث السرعة والتيار وغيرها.
- عمل توليف آلي (إن أمكن) Auto-tune لمغير السرعة مع الموتور لضبط قياسات الموتور الضرورية وضبط منظمات السرعة والتيار آليا حال استخدامها
- اختيار التطبيق المناسب مع عمل الضبط اللازم للتطبيق مع الالتزام بتعليمات الشركة المصنعة
- بدء التشغيل بشكل آلي مع المنظومة حسب التطبيق المستخدم
- مراقبة أداء مغير السرعة وقياسات الجهاز سواء الفولت والتيار

26

وسوف تجد في كل مغيرات السرعة من خلال كتالوجات الشركات المصنعة الأقسام الآتية :

1 - توصيف مغير السرعة
2 - توصيف المحرك الذي يعمل مع مغير السرعة
3 - الأبعاد الميكانيكية وكيفية التثبيت
4 - التوصيل الكهربي (القدرة الكهربية – التحكم الرقمي والتماثلي)
5 - كيفية برمجة الجهاز
6 - التطبيقات المستخدمة في مغير السرعة
7 - بيانات مغير السرعة التي يمكن ضبطها ومستوياتها المختلفة
8 - رسائل الخطأ بالجهاز وكيفية التعامل معها
9 - بيانات الاتصال وكيفية برمجة الاتصال (إن وجدت في مغير السرعة) وكيفية توصيل كابلات الاتصال
10 - كيفية التعامل عن طريق برامج الشركة المصنعة بدلا من لوحة المفاتيح (إن وجد)

التطبيقات الشهيرة في مغيرات السرعة:

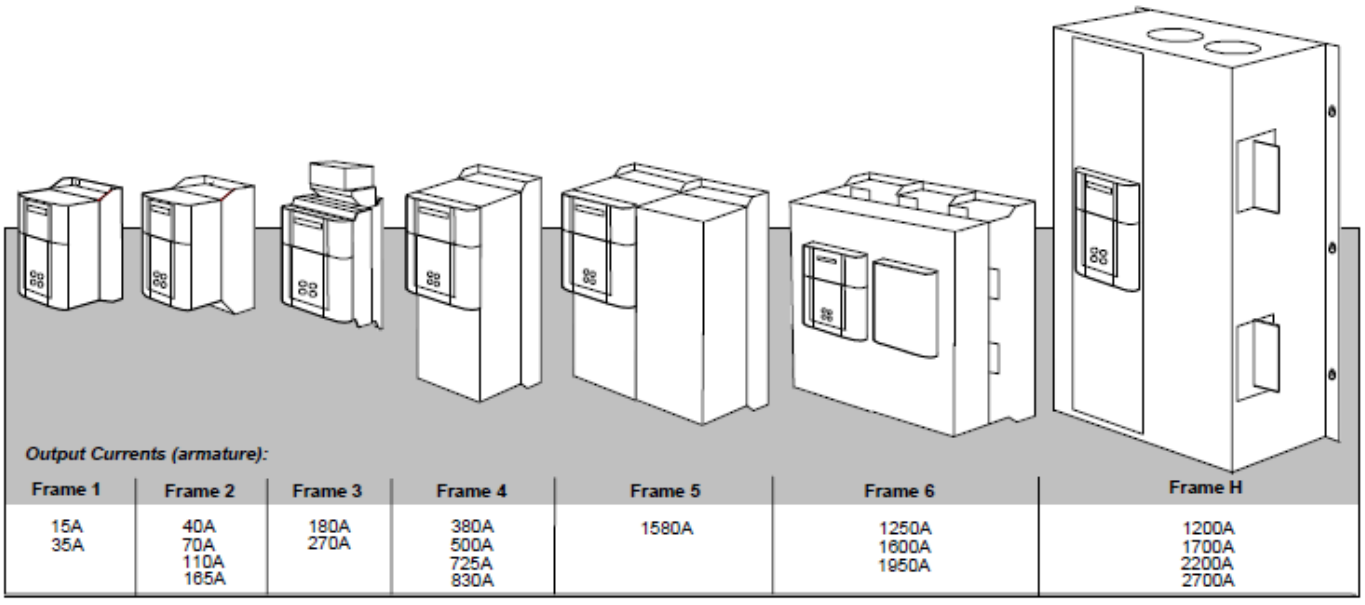
- التطبيق القياسي Speed Control حيث يعطي إمكانية التشغيل/الإيقاف وعكس الاتجاه عن طريق مدخلات رقمية وإدخال قيمة السرعة والتيار عن طريق مدخلات تناظرية
وهنا تكون علاقة السرعة بإشارة التحكم طردية وخطية أي كلما ازدادت إشارة الدخل كلما زادت السرعة عن طريق التردد في مغيرات السرعة للتيار المتردد وجهد الأرميتشر لمحركات التيار المستمر
- تطبيق التحكم في العزم Torque Control حيث يكون عزم الدوران هو محور التحكم ويتم ذلك عن طريق التحكم في تيار الحمل والجهد والتردد (في حالة محركات التيار المتردد) أيضا فالتغير في السرعة هنا يكون بناء على عزم الدوران المطلوب
- تطبيق السرعات الثابتة Fixed Speeds حيث هنا عن طريق مداخل رقمية وطبقا لعددها يكون عدد السرعات فإن كانا مدخلين كان عدد السرعات 4 وإن كان ثلاث مداخل كان عدد السرعات 8 وإن كان أربعة كان عدد السرعات 16 بمعنى أن عدد السرعات يساوي 2 مرفوعة إلى الأس مساويا لعدد مداخل التحكم ويتم إدخال جدول السرعات جميعها أو على الأقل السرعات المحتملة قبل التشغيل حيث يتم الانتقال من سرعة لأخرى بمجرد تغيير حالة المدخلات والتي يمكن أن تأخذ شكل الجدول التالي على سبيل المثال:

DI1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
DI2	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**
DI3	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
DI4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Speed	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**

- تطبيق زيادة/نقصان السرعة Increment/Decrement وهنا يكون العامل المؤثر في التشغيل هو اثنان من المداخل الرقمية أحدهما لزيادة السرعة والآخر لنقصان السرعة حيث عند انتقال حالة أي منهما من الحالة "0" إلى الحالة "1" أي على الحافة الموجبة Positive edge يتم زيادة السرعة أو نقصانها بمعدل ثابت يكون ضمن عوامل ضبط مغير السرعة فمثلا لو كان معدل زيادة سرعة جهاز 2 هرتز/لكل نبضة من المدخلات التي أشرنا إليها وكان يعمل مثلا على سرعة تتناسب مع التردد 12 هرتز وأردنا أن ينتقل إلى 20 هرتز فهذا يعني أننا نحتاج إلى 8 نبضات ليتيغير التردد من 8 إلى 20 بمعدل 2 هرتز لكل نبضة

مثال على تشغيل مغير سرعة تيار مستمر بشكل كامل (على سبيل المثال النوع SSD 590+ من شركة باركر)

فبناء على قدرة المحرك المطلوبة فسوف نحدد حجم الوحدة المطلوبة وشكل التثبيت لها وأبعادها مثل الشكل التالي:



Output Currents (armature):

Frame 1	Frame 2	Frame 3	Frame 4	Frame 5	Frame 6	Frame H
15A 35A	40A 70A 110A 165A	180A 270A	380A 500A 725A 830A	1580A	1250A 1600A 1950A	1200A 1700A 2200A 2700A

All units are available as a:

590+ : 4Q 3-phase, fully controlled, anti-parallel thyristor bridge configuration

591+ : 2Q 3-phase, fully controlled thyristor bridge configuration

وتأتي المرحلة الثانية وهي تحديد المواصفات الكهربية كاملة للوحدة ومطابقة ذلك مع مواصفات الشركة القياسية فمثلا في مغير السرعة الذي معنا يكون الوصف بتيار الحمل مثلما هو الحال تحت شكل كل حجم من أحجام الأجهزة وبالتالي فلن يكون هناك توصيف زائد إلا في حالة طلب مواصفات خاصة قد لا يبديها الوصف العام للجهاز فيجب توضيح ذلك تماما

وننتقل بعد ذلك لمرحلة التوصيل الكهربي وقد يكون للشركة متطلبات خاصة في عمليات التوصيل وخاصة خارج الجهاز كنوع الكابلات المستخدمة واستخدام فلاتر وملفات خنق في مناطق معينة للحد من التغير المفاجئ للتيار وهذا كما في الشكل التالي:

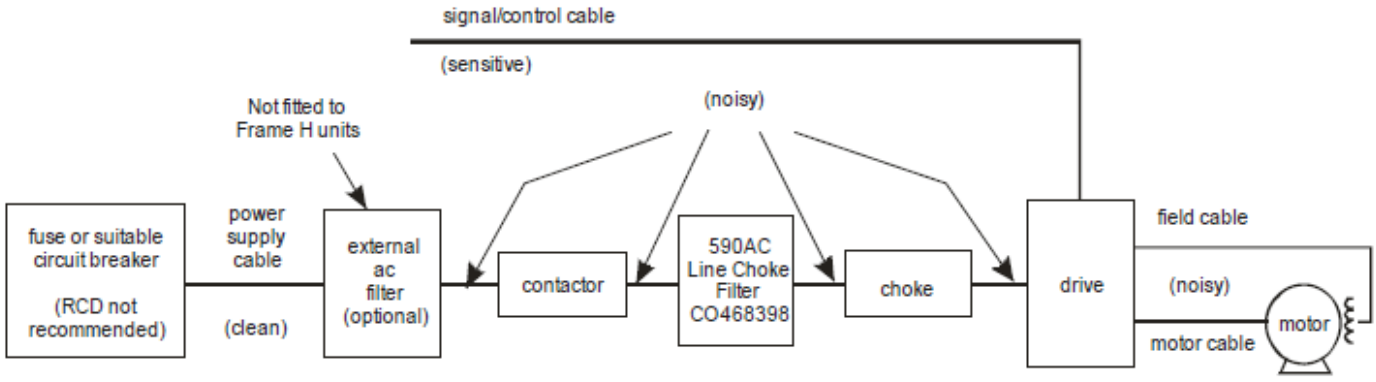


Figure 3- 5 Cabling Requirements

ثم تبدأ بعد ذلك مرحلة توصيل الأسلاك للجهاز حسب التطبيق المختار وحسب المطلوب لتشغيل الجهاز وفي الشكل التالي مثال على التوصيل في حالة التطبيق العام General purpose application وسوف نعلق على التوصيل بعد عرض الصورة أولا :

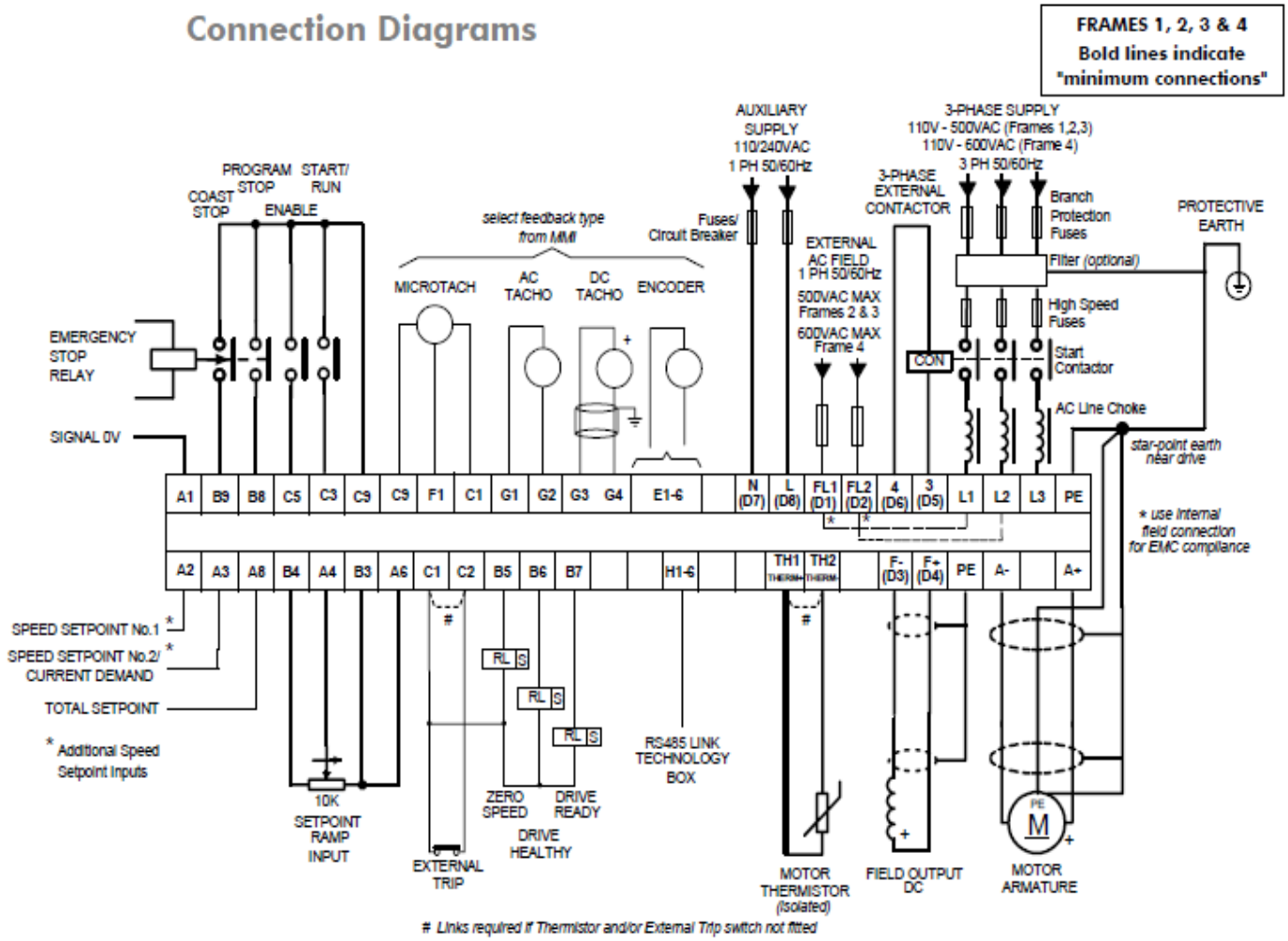


Figure 3- 8 Power Connections: Frames 1, 2, 3 & 4 ('general purpose' configuration)

ولنبدأ بالجزء على يمين الصورة من أعلى والذي يعرض لنا توصيل القدرة الكهربية الرئيسية للجهاز Input power وذلك بأن تكون مواصفات المصدر كما أوضحت الصورة حيث يتراوح فرق الجهد من 100 إلى 600 فولت أو 500 فولت حسب الجهاز ويكون التردد 50 أو 60 هرتز وهذا في شكل 3-Phase وتمر أطراف القدرة عبر 5 عناصر كما افترض المصنع حسب الوضع المثالي حيث وضع مجموعة فيوزات (مصهرات) في أول نقطة توصيل للمصدر للتوصيل وبعدها مباشرها وضع فلتر Filter ثم مجموعة فيوزات أخرى سريعة للحماية ثم وضع كونتاكتور توصيل القدرة الرئيسي والذي لا تمر القدرة إلا عند عمله ثم يلي الكونتاكتور بعد ذلك ملف خنق Chock Coil ثلاثي وهذا كله لحماية الجهاز والفصل والتوصيل كما يلي:

- مجموعة الفيوزات الأولى للفصل والتوصيل
- الفلتر للحماية من الشوشرة على مصدر التغذية
- مجموعة الفيوزات الثانية للحماية ضد قصر الدائرة Short circuit أو الحمل الزائد
- الكونتاكتور الرئيسي Main contactor لتوصيل وفصل القدرة الكهربية بناء على حالة جاهزية الجهاز للعمل
- ملفات الخنق Chock coils والتي تعارض الارتفاع المفاجئ للتيار الكهربي وتعمل على الزيادة التدريجية

ويتم توصيل مصدر التغذية بعد المرور على تلك المراحل (بأطرافة الخمسة R-S-T-N-PE)على الأطراف المقابلة في الجهاز L1,L2,L3 ويتم توصيل الأرضي من المعادل N على الطرف PE ما لم يكونا منفصلين في الجهاز

ويتم توصيل أطراف الأرميتشر للمحرك A+,A- وتوصيل طرف الأرضي للحماية ضد القصر في حالة قطع الكابل

وفي حالة تشغيل ملفات الفيلد فسوف نجد الحاجة لتوصيل مصدر آخر لتغذية الجزء الخاص بالفيلد على الأطراف FL1,FL2 عبر فيوز حماية يتناسب مع تيار حمل الفيلد

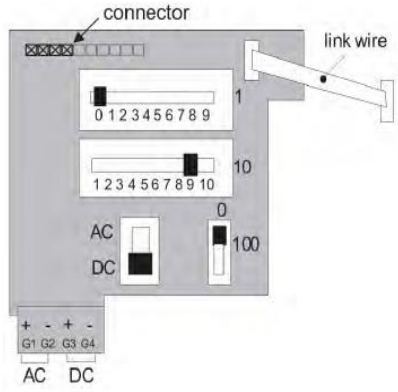
وفي المقابل يتم توصيل طرفي ملفات الفيلد على الأطراف F+,F- على الجهاز

وفي معظم الأجهزة المتوسطة والكبيرة وبعض الأجهزة الصغيرة يتم فصل تغذية الدوائر الإلكترونية تماما عن دوائر القدرة الكهربية وبالتالي فسوف يتطلب الأمر توصيل قدرة كهربية 110 أو 240 فولت كمصدر جهد مساعد Auxiliary supply متررد على الأطراف L,N والتي تقوم بتغذية الدوائر الإلكترونية لتشغيل الوحدة كما ذكرنا ويكون ذلك عبر فيوز حماية أيضا

يبقى لنا في الجزء الخاص بالقدرة شئ واحد وهو أطراف تشغيل الكونتاكتور الرئيسي والتي تكون عبر الأطراف المخصصة لذلك من داخل الدرايف (3,4) كما بالصورة حيث عندما يكون الجهاز جاهزا ويستقبل إشارة التجهيز Enable يقوم الجهاز بتشغيل الكونتاكتور الرئيسي لتمرير القدرة الكهربية إلى وحدة توحيد القدرة (الثيرستور)

وبعد توصيل القدرة نقوم بتوصيل الشروط الضرورية للتشغيل ومنها:

- ثرموستات المحرك والذي يعطي رؤية عن ارتفاع درجة حرارة المحرك ويتم التوصيل على الأطراف TH1,TH2
- وحدة قراءة السرعة الفعلية سواء كانت تاكو AC,DC أو كان انكودر مع ضبطه على الوضع الصحيح في حالة التاكو ليعطي الفولت الصحيح لكل لفة للمحرك عن طريق المفاتيح المخصصة لذلك على كارت التاكو أما الانكودر فيتم ضبط بياناته من داخل الدراف



- توصيل شروط التوقف المفاجئ Program stop, Coast stop وفي حالة عدم توافر شروط لمثل هذه الشروط يتم قصر الأطراف مع بعضها أي B8,B9 مع C9

وبهذه التوصيلات يكون الجهاز جاهزا للتشغيل اليدوي المحلي Local operation من لوحة التشغيل

ويتم هذا بعد توصيل القدرة الكهربية والتأكد من وصولها للجهاز والتأكد من عمل الجهاز حيث من لوحة التشغيل

- يتم تحويل التشغيل إلى Local عن طريق المفتاح الذي عليه الرمز L/R عندها ينتقل التشغيل تماما إلى اللوحة
- بالضغط على مفتاح I الأخضر تعمل الوحدة وعلى مفتاح 0 الأحمر تتوقف الوحدة ويمكن زيادة السرعة وإنقاصها عن طريق الأسهم لأعلى وأسفل ويمكن عكس الاتجاه عن طريق المفتاح الذي عليه سهمان متعاكسان <>

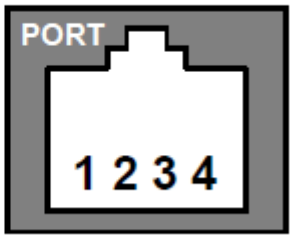
وبالتالي نستطيع التأكد من عمل الجهاز والمحرك وعمل القياسات اللازمة وتحديد الاتجاه الصحيح والدقة المطلوبة للتشغيل كل ذلك قبل أن نشرع في التشغيل الفعلي للوحدة مع التطبيق لدينا

ونأتي بعد ذلك لمرحلة هامة وهي بداية التعرف على كيفية ضبط عوامل تشغيل الجهاز Parameters والقوائم الخاصة بها لتنفيذ ما يؤدي إلى ضبط صحيح وكامل للعملية المطلوبة

تتم عملية الضبط بوسيلتين الأولى عن طريق لوحة التشغيل Keypad مثل تلك التي قمنا بالتشغيل اليدوي من خلالها أو من خلال برنامج مع PC مع كابل اتصال فالحاسب المطلوب يلزم أن يتوافربه منفذ RS232 أو وسيلة لتحقيق هذا والبرنامج هو CE-Lite من شركة Parker SSD وهو خاص ببرمجة مغيرات السرعة الخاصة بها وكابل التوصيل هو كابل سيريال على RJ مثل المستخدمة في سماعة التليفون المنزلي كما بالشكل التالي:

Use a standard P3 lead to connect to the Drive.

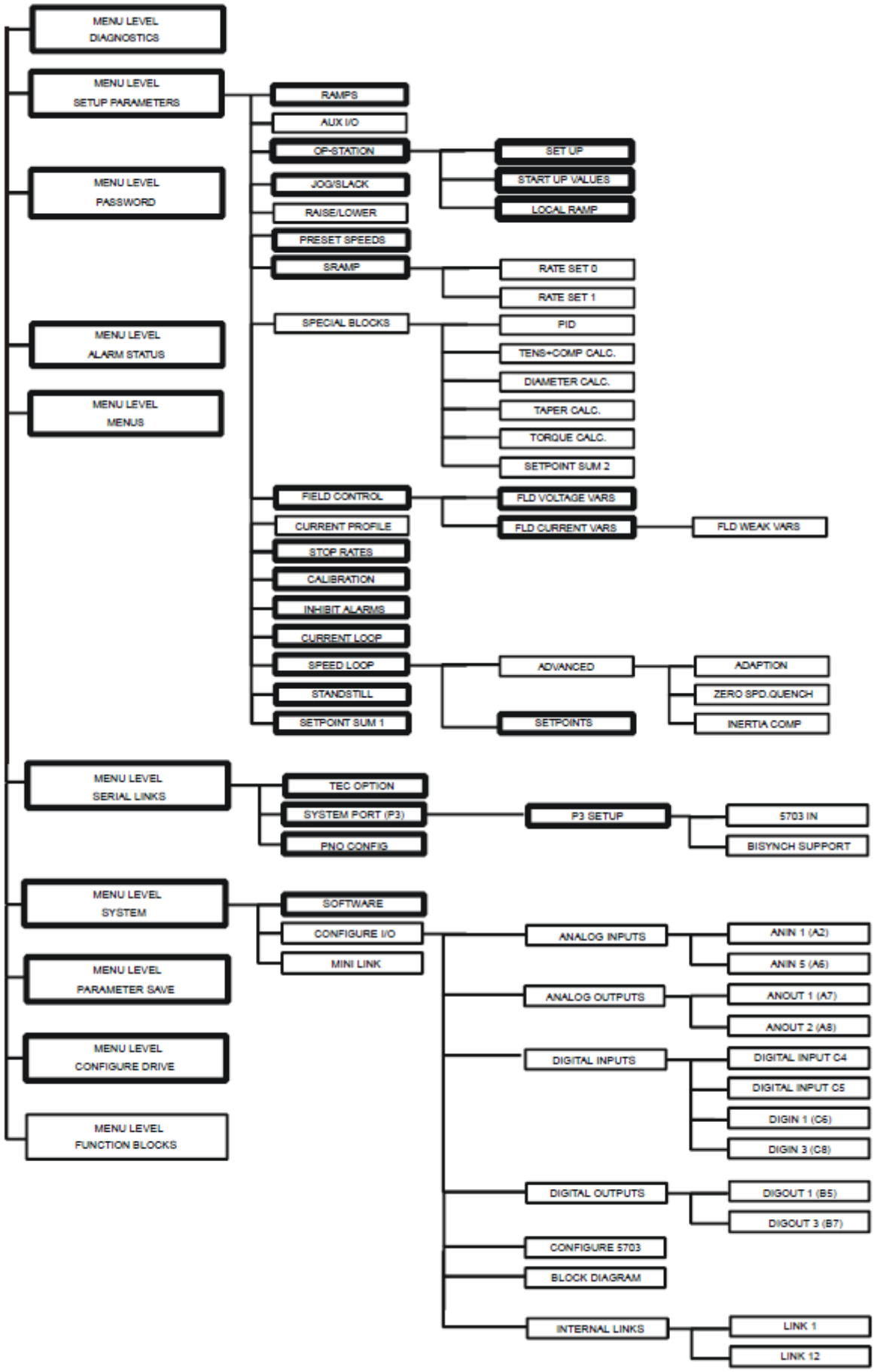
P3 Port Pin	Lead	Signal
1	Black	0V
2	Red	24V
3	Green	TX
4	Yellow	RX



هذا على جانب مغير السرعة أما على الجانب الآخر فسوف يكون الكابل RS232 بصورته القياسية وعلى هذا فالطرف رقم-2 والذي عليه 24 فولت لا يتم توصيله في حالة الاتصال مع الحاسب بل هو لتشغيل لوحة التشغيل السابقة فقط أما الحاسب فيتم توصيل 3 أطراف فقط وهي 1،3،4 ويناظرها في الجانب الآخر الأطراف 2،3،5 كما بالجدول التالي للسوكت 9 أطراف والسوكت 25 طرف:

P3 Port Pin	Lead	Female DB9 Pin	Female DB25 Pin
1	Black	5	7
2	Red	not connected	not connected
3	Green	2	3
4	Yellow	3	2

ويتم إدخال بيانات التشغيل عبر قوائم الضبط Menus والتي تتكون من مجموعة من القوائم الرئيسية تحتها قوائم فرعية حتى نصل إلى كل البيانات الخاصة بضبط الوحدة



32

وعبر هذه القوائم من البيانات يمكن الوصول إلى كل عنصر من العناصر التي نريد ضبطها وعمل الضبط اللازم

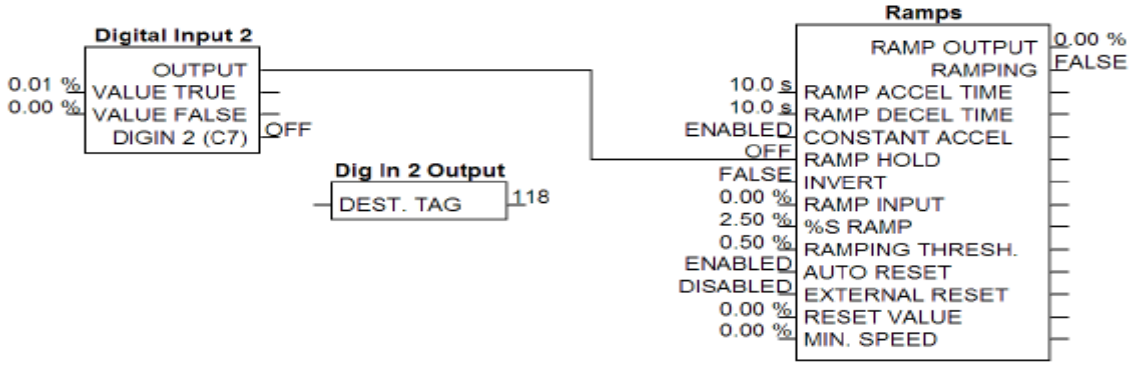
وبعد الانتهاء من عمل الضبط اللازم لكل البيانات المطلوبة تماما يتم البدء في التشغيل عبر أطراف التشغيل الخاص بذلك وهي Enable والذي يقوم بإعداد الدوائر الإلكترونية للتشغيل ثم Run والذي يقوم عمليا ببدء تشغيل وحدات الثيرستور وFor/Rev والذي يقوم بعكس اتجاه الحركة بالإضافة إلى المدخلات التناظرية Analog inputs الخاصة بالسرعة وتيار الحمل

ويمكن مراقبة عمل الوحدة عن طريق مخرجات رقمية يمكن استخدامها في دوائر التحكم مثل جاهزية مغير السرعة أو التشغيل أو الوصول إلى السرعة صفر أو الاتجاه

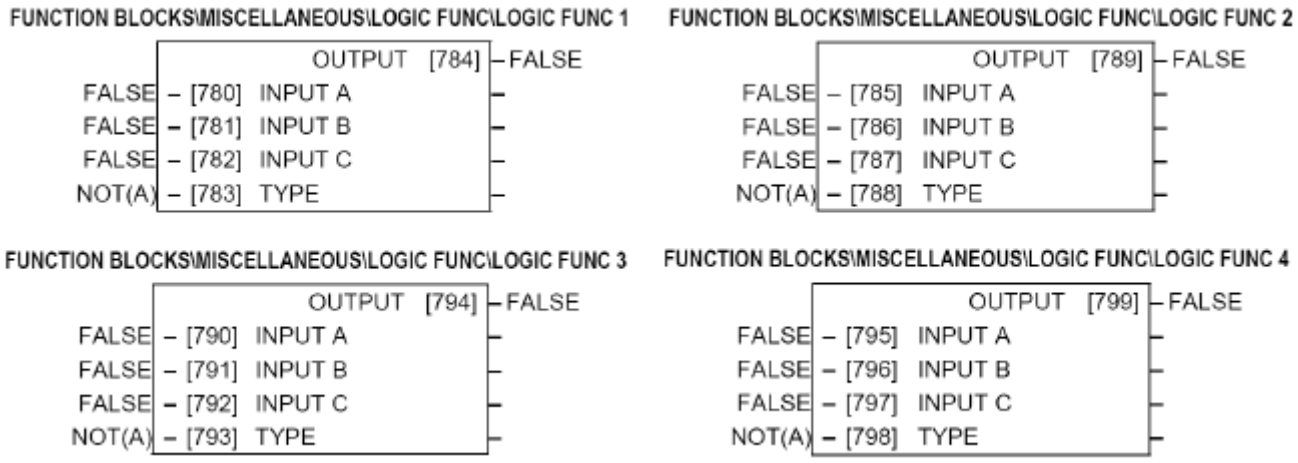
ويمكن عن طريق لوحة التشغيل أيضا معرفة حالة بيانات التشغيل كلها بالإضافة إلى مرقبة تامة للسرعة وتيار الحمل المطلوب والفعلي وقياسات المدخلات الرقمية والتناظرية كلها وحالات المخرجات أيضا بنوعيها

كذلك يمكن عن طريق وحدات الاتصال التشغيل عن طريق منافذ الاتصال Communication ports حيث يتم تمرير البيانات إلى Tags الخاصة بها ويجد جدول كامل بعناوين Tags الخاصة بالوحدة والتي يتم عن طريقها تشغيل الوحدة عن بعد وليس عن طريق أطراف التوصيل

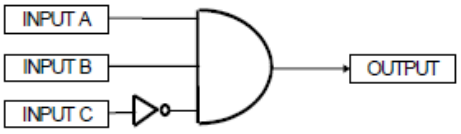
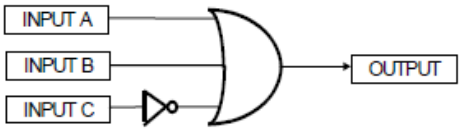
وعند التعامل في ضبط الوحدة عن طريق البرامج CE-Lite فإننا سوف نجد ضبط البيانات في شكل قوالب Blocks لكل منها مدخلات ومخرجات وبينها وبين بعضها وصلات Links حيث يمكن ضبط البيانات وتغيير الوصلات أيضا بين الوحدات لتكوين شكل التحكم المطلوب كما تريد أنت وكما ترى في الشكل التالي:



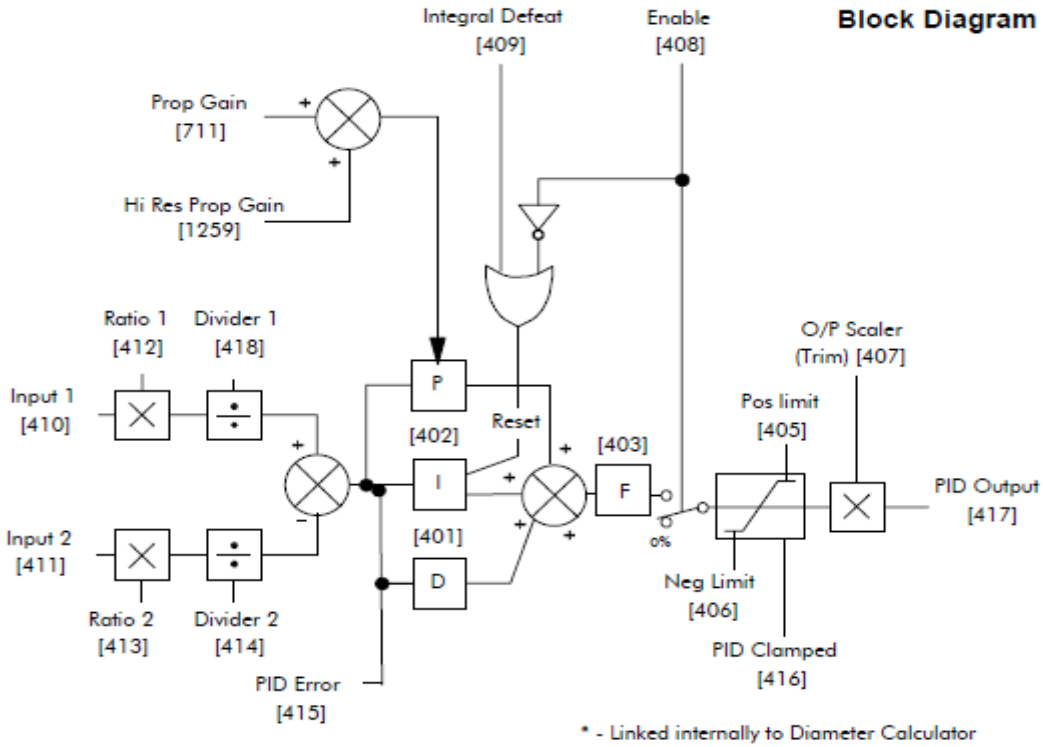
كما أنه يوجد العديد من العمليات المنطقية Logic function جاهزة وحرة تستخدمها وقتما تشاء داخل البرنامج بمجرد ضبطها كما في الشكل التالي حيث يتم فقط اختيار الدالة واختيار نوعها ومدخلاتها Input tags ومخرجاتها Output tags ليتم استخدامها بشكل ما في عمليات التحكم



وفي الشكل التالي مثال على شكل مثل هذه الدوال Logic function

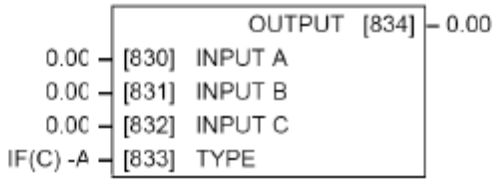
Operation	Description					
AND(A,B,!C)	AND(A,B,!C)			Input State		
				A	B	C	Output State
				0	0	0	0
				0	0	1	0
				0	1	0	0
				0	1	1	0
	Refer to the Truth Table.			1	0	0	0
	FALSE = 0, TRUE = 1.			1	0	1	0
				1	1	0	1
				1	1	1	0
OR(A,B,!C)	OR(A,B,!C)			Input State		
				A	B	C	Output State
				0	0	0	1
				0	0	1	0
				0	1	0	1
	Refer to the Truth Table.			0	1	1	1
	FALSE = 0, TRUE = 1.			1	0	0	1
				1	0	1	1
				1	1	0	1
				1	1	1	1

كما ستجد وصفا لوظيفيا للعمليات الخاصة مثل PID لا يجب إغفاله بل يجب التدقيق في كل شئ منه خاصة عند استعمال مثل تلك المنظومات الخاصة في التطبيق لمعرفة كيف يتم عمل هذه المنظومات

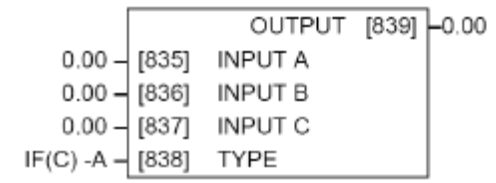


ومثلما كان الحال بالنسبة للعمليات المنطقية فسوف تجد عمليات حسابية ومقارنات تتعامل مع القيمة تسمى Value function حرة أيضا يمكن أن تستخدمها كما تشاء

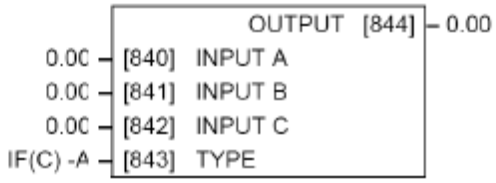
FUNCTION BLOCKS\MISCELLANEOUS\VALUE FUNC\VALUE FUNC 1



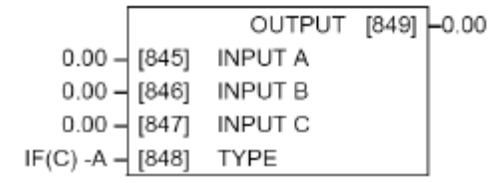
FUNCTION BLOCKS\MISCELLANEOUS\VALUE FUNC\VALUE FUNC 2



FUNCTION BLOCKS\MISCELLANEOUS\VALUE FUNC\VALUE FUNC 3



FUNCTION BLOCKS\MISCELLANEOUS\VALUE FUNC\VALUE FUNC 4



ويصل عدد العمليات إلى 46 عملية مختلفة كما في الجدول التالي حيث نختار نوع العملية من بينها

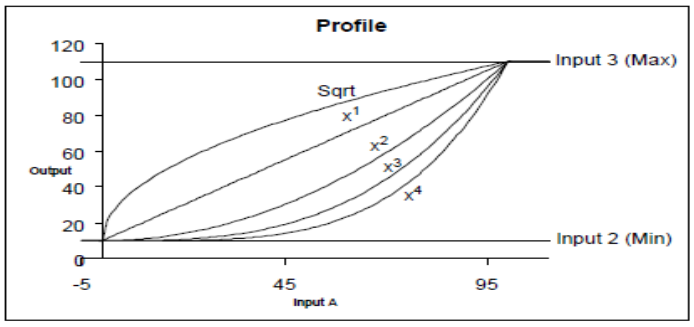
The operation to be performed on the three inputs to produce the output value.

0: IF(C) -A
1: ABS(A+B+C)
2: SWITCH(A,B)
3: (A*B)/C
4: A+B+C
5: A-B-C
6: B<=A<=C
7: A>B+/-C
8: A>=B
9: ABS(A)>B+/-C
10: ABS(A)>=B
11: A(1+B)
12: IF(C) HOLD(A)
13: BINARY DECODE
14: ON DELAY
15: OFF DELAY
16: TIMER
17: MINIMUM PULSE
18: PULSE TRAIN
19: WINDOW
20: UP/DWN COUNTER
21: (A*B)/C ROUND
22: WINDOW NO HYST
23: WIND A>=B,A<=C
24: A<=B
25: ((A*B)/100)+C
26: MIN(A,B,C)
27: MAX(A,B,C)
28: PROFILE SQRT
29: PROFILE LINEAR
30: PROFILE x^2
31: PROFILE x^3
32: PROFILE x^4
33: ON A>B, OFF A<C
34: (A+B) CLAMPED C
35: (A-B) CLAMPED C
36: (A*B) CLAMPED C
37: (A/B) CLAMPED C
38: A>=B:A, A<=C:0
39: (A * B) + C
40: A * (B + C)
41: A * (B - C)
42: A * (1+B/C)
43: A * (1+(B * C))
44: MONOSTABLE HIGH
45: MONOSTABLE LOW
46: FILTER

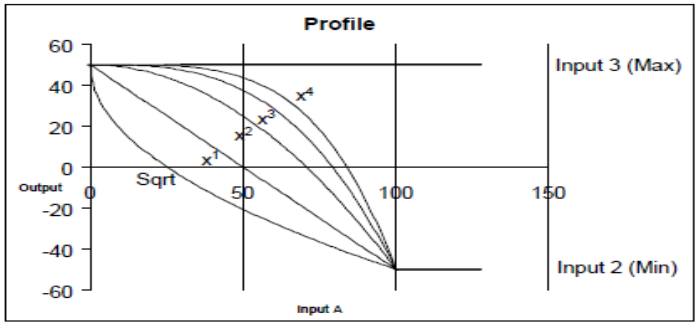
35

وسوف تجد توضيحا وشرحا لكيفية تنفيذ مثل تلك العمليات أوماذا تعني مع كل منها مثل الشكل التالي :

PROFILE SQRT
PROFILE X^1
PROFILE X^2
PROFILE X^3
PROFILE X^4

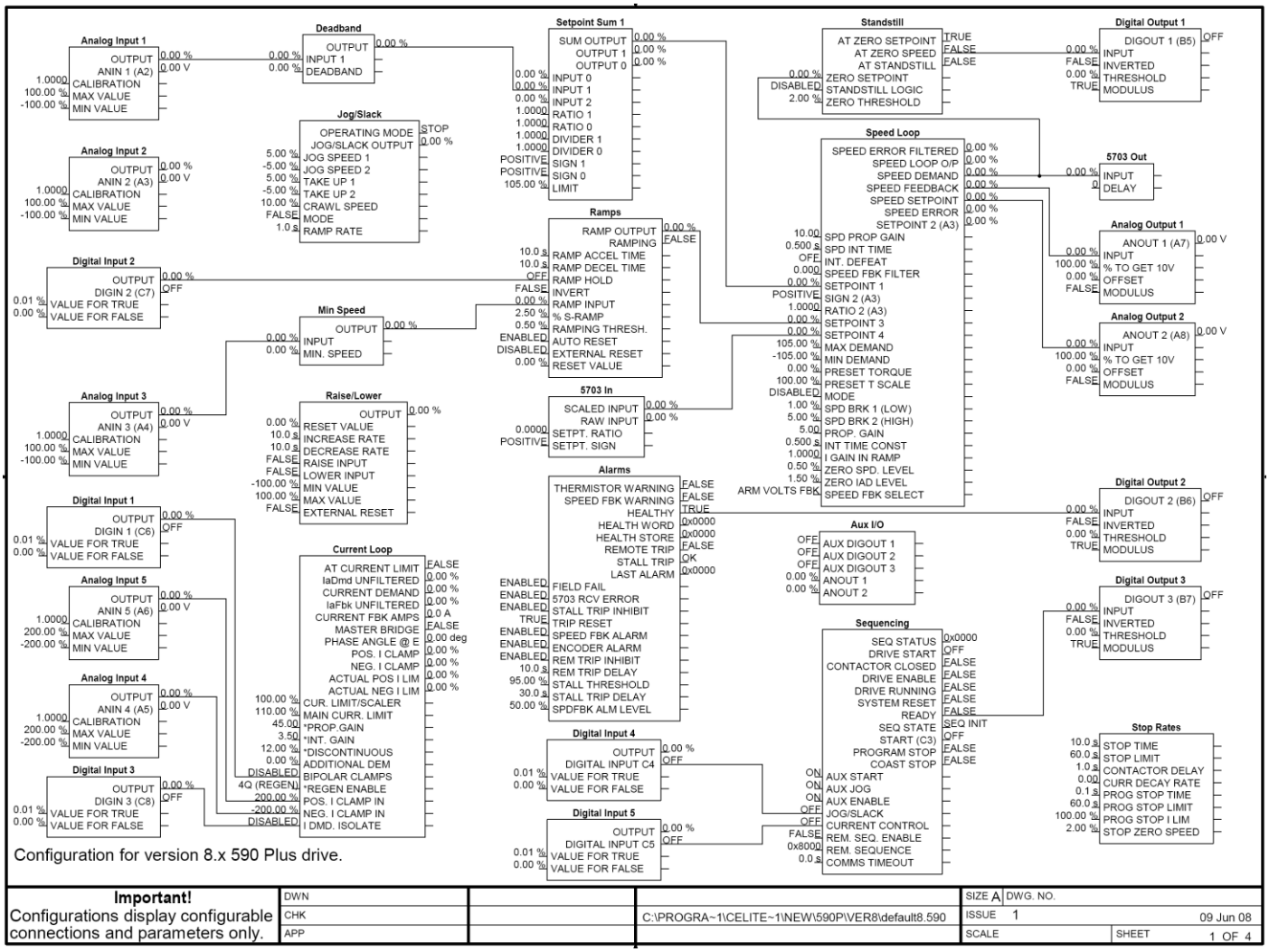


Example : Profile Min = 10, Max = 110



Example : Profile Min = 50, Max = -50

وفي النهاية سوف تجد شكلا مجمعا للتطبيقات المختلفة يشمل كل شئ على صفحة أو أكثر



أرجو أن تفيد مثل هذه المعلومات في تشغيل مغيرات السرعة وإن كان هذا نوع من أنواعها فكل الأنواع تقوم بنفس الوظيفة تقريبا وسوف تجد نفس العمليات مع اختلاف بسيط في التوصيل أو الضبط مثلما هو الحالي مع منتجي التكنولوجيا في كل مجالات الحياة كل ما عليك هو أن تدرك الإطار العام للعمل وتتبع تعليمات المصنع في التثبيت والتوصيل والبرمجة أو الضبط والتشغيل والمتابعة أثناء التشغيل وسوف تحصل على أعلى كفاءة في التشغيل مهما كان نوع أو منتج مغير السرعة لديك.

طرق أخرى للتحكم

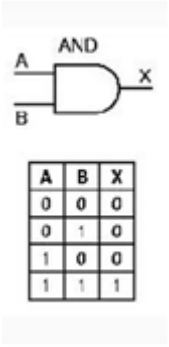
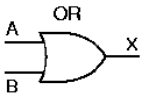
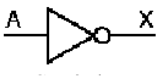
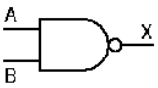
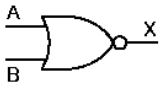
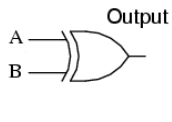
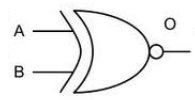
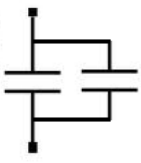
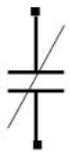
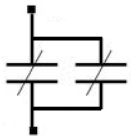
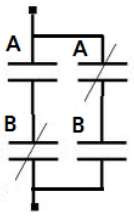
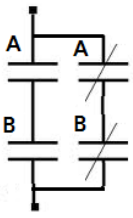
للتحكم طرق معروفة كما ذكرنا من قبل منها التحكم التقليدي والتحكم عن طريق أجهزة تحكم العمليات Process controllers ومنها التحكم عن طريق الكروت الإلكترونية Electronic boards ومنها التحكم عن طريق المتحكمات الصغيرة Micro-controllers ومنها التحكم عن طريق الحاكمات المنطقية PLC ونظم الاسكادا

وفي هذا نلقي الضوء قليلا عن ما يمكن أن نفعله باستخدام العناصر الإلكترونية أو الكروت الإلكترونية للتحكم في العمليات المختلفة بعناصر المعروفة لنا من قبل حيث أنه من معرض حديثنا السابق فإن هناك قسمان رئيسيان لعمليات التحكم حسب نوع الإشارة أو العملية التي نتحكم فيها وهما قسم رقمي Digital control وقسم تناظري Analog control وبالتالي سوف نتعرض من وجهة النظر تلك لما يمكن أن نقوم به في عمليات التحكم المختلفة

التحكم المنطقي Logic/Digital Control

يشمل التحكم المنطقي كل أشكال التحكم التي تعتمد على الحالة Status والتي تدخل في إطار الجبر الثنائي Boolean algebra كما يطلق عليه وهي العمليات أو عناصر التحكم التي يكون مادخلها أو نتيجتها إما "نعم" أو "لا" والتي سبق وتعرضنا لها في الحديث عن التحكم التقليدي وهنا في الحديث عن الإلكترونيات سيكون عن طريق البوابات المنطقية Logic Gates والتي تشكل نفس العناصر السابقة ولكن في شكل عناصر إلكترونية بسيطة قليلة القدرة الكهربية وصغيرة الإشارة الكهربية

وبالتالي فعناصر هذا النوع من التحكم هي البوابات المنطقية وعمليات الجبر الثنائي القياسية الخمسة عشر الشهيرة

AND	OR	NOT	NAND	NOR	EOR(XOR)	ENOR
 A B X: 0 0 0; 0 1 0; 1 0 0; 1 1 1	 A B X: 0 0 0; 0 1 1; 1 0 1; 1 1 1	 A X: 0 1; 1 0	 A B X: 0 0 1; 0 1 1; 1 0 1; 1 1 0	 A B X: 0 0 1; 0 1 0; 1 0 0; 1 1 0	 A B Output: 0 0 0; 0 1 1; 1 0 1; 1 1 0	 A B O: 0 0 1; 0 1 0; 1 0 0; 1 1 1
 Serial	 Parallel	 NOT	 NAND	 NOR	 XOR	 XNOR
هنا لابد من وجود الاثنين حى يوجد الخرج	هنا يكفي وجود واحد فقط ليوجد الخرج	هنا الخرج عكس حالة الدخل	هنا لابد من عدم وجود أحدهما على الأقل	هنا لابد من عدم وجود الاثنين معا	هنا لابد من اختلاف الاثنين ليوجد الخرج	هنا لابد من تشابه الاثنين ليوجد الخرج

ومن هذه البوابات السبعة يمكن تكوين كل شئ في عالم الجبر الثنائي أو التحكم عن طريق الدوائر الإلكترونية حيث نستخدم علاقات وطرق الجبر الثنائي لصياغة عملية التحكم المطلوبة

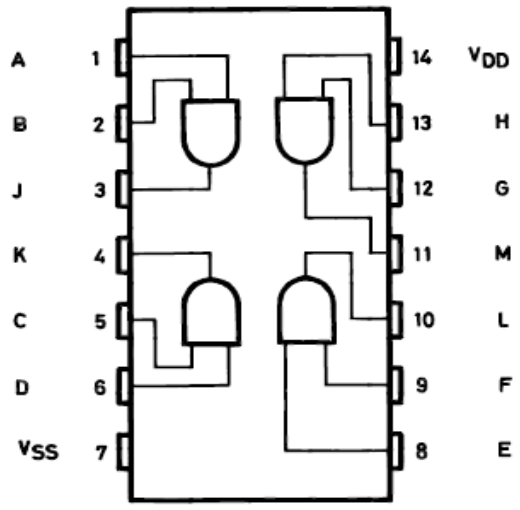
The 16 Possible Boolean Functions of Two Variables

Function #		Description
0	0	Zero or Clear. Always returns zero regardless of A and B input values.
1	NOR	Logical NOR (NOT (A OR B)) = (A+B)'
2	AND NOT	Inhibition = BA' (B, not A). Also equivalent to B>A or A < B.
3	NOT	NOT A. Ignores B and returns A'.
4	AND NOT	Inhibition = AB' (A, not B). Also equivalent to A>B or B<A.
5	NOT	NOT B. Returns B' and ignores A
6	XOR	Exclusive-or (XOR) = A ⊕ B. Also equivalent to A≠B.
7	NAND	Logical NAND (NOT (A AND B)) = (A•B)'
8	AND	Logical AND = A•B. Returns A AND B.
9	XNOR	Equivalence = (A = B). Also known as exclusive-NOR (not exclusive-or).
10	=	Copy B. Returns the value of B and ignores A's value.
11	OR NOT	Implication, B implies A = A + B'. (if B then A). Also equivalent to B >= A.
12	=	Copy A. Returns the value of A and ignores B's value.
13	OR NOT	Implication, A implies B = B + A' (if A then B). Also equivalent to A >= B.
14	OR	Logical OR = A+B. Returns A OR B.
15	1	One or Set. Always returns one regardless of A and B input values.

هذا الجدول يحتوي على كل العمليات الأولية المحتملة في عالم الجبر الثنائي والتي يتكون منها بعد ذلك كل العمليات المعقدة الأخرى

وهذه البوابات المنطقية موجودة في شكل دوائر متكاملة يمكن استخدامها بشكل مباشر في التطبيقات المختلفة حسب التصميم الذي نريده وهناك منها مثلا :

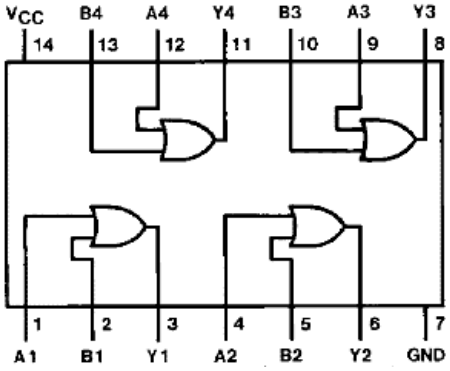
الدائرة المتكاملة من النوع CMOS والتي تحتوي على عدد 4 بوابات منطقية من النوع AND ثنائية المدخل أي لها مدخلان فقط لكل بوابة ورقمها هو 4081B والذي يمكن شراؤها به في الأسواق ويوجد منها أيضا أنواع أخرى TTL بنفس الشكل تماما كما توجد منها أنواع أخرى حيث يختلف عدد البوابات ويختلف عدد المدخلات للبوابات كما يمكن أن يضاف لها نوع آخر من البوابات مثل NOT للاستخدام عند الحاجة سواء من عائلة CMOS والتي تبدأ بالرقم 4000 أو من عائلة TTL والتي تبدأ بالرقم 74000



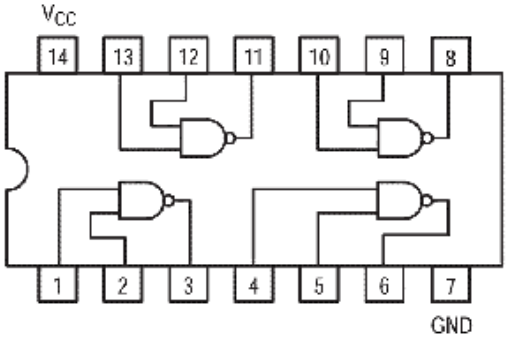
IC 4081B (4-AND Gates)

بنفس الطريقة تماما يوجد دوائر متكاملة IC من النوعين CMOS والتي تبدأ بالرقم 4000 والنوع TTL والتي تبدأ بالرقم 74000 تمثل البوابة المنطقية OR كما بالشكل المقابل

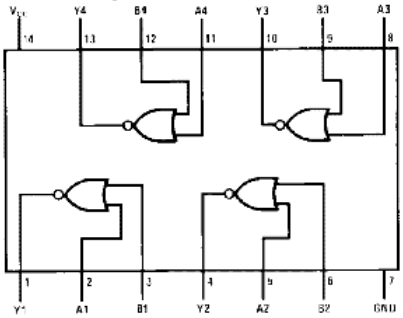
بوابات منطقية من النوع OR



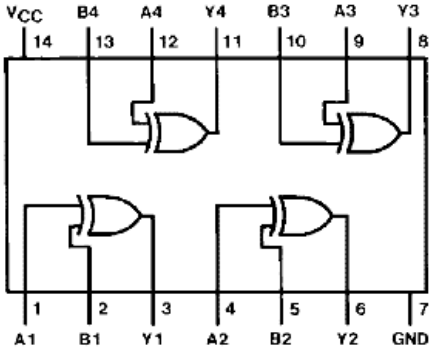
بوابات منطقية من النوع NAND



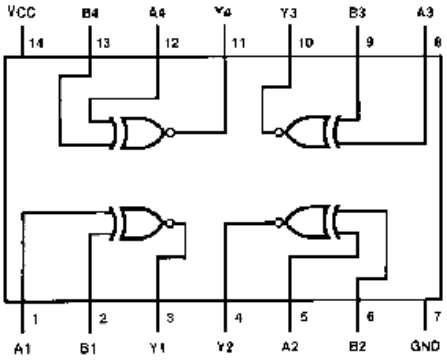
بوابات منطقية من النوع NOR



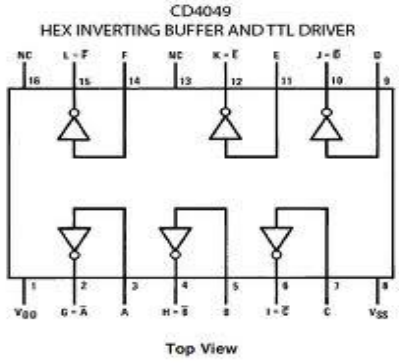
بوابات منطقية من النوع XOR



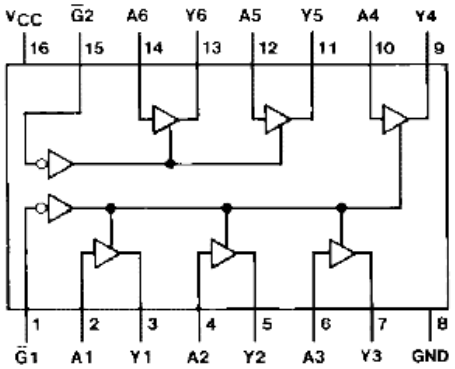
بوابات منطقية من النوع XNOR



بوابات منطقية من النوع NOT



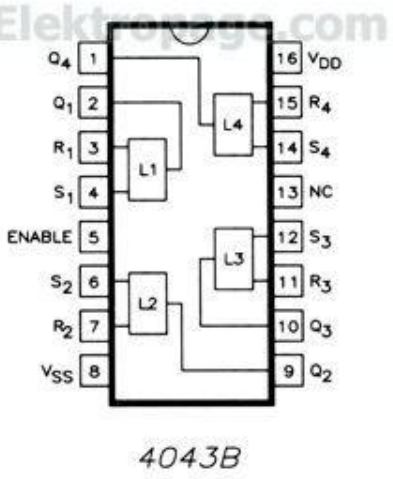
بوابات منطقية من النوع Buffer لنقل الحالة فقط



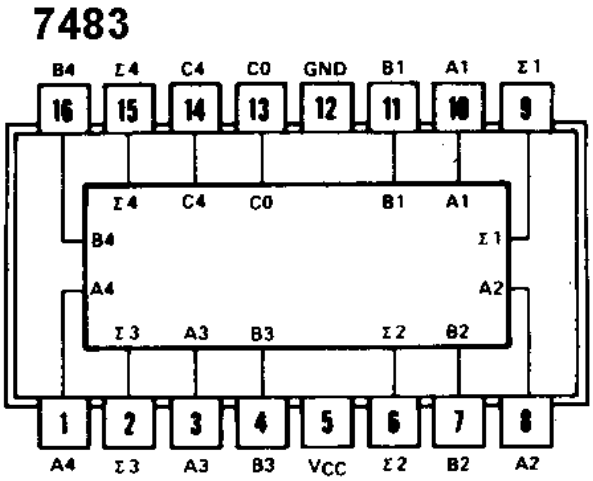
كذلك فإنه يوجد عمليات مركبة من تلك العمليات مثل عمليات التشفير Coding وفك التشفير Decoding والمفاتيح المتعددة المدخلات Multiplexing والعكس De-multiplexing والعدادات وصولا إلى المعالجات كلها تعتمد على البوابات المنطقية

من أشهر الكتب باللغة الانجليزية التي تتحدث في هذا الموضوع كتاب Digital design الشهير للمؤلف الشهير موريس مانو وأوصي جميع القراء بقراءته حيث يتعرض بشكل مفصل لنظم الأرقام ويتحدث عن النظام الثنائي والجبر الثنائي ثم يفصل كثيرا في طرق اختصار العمليات والتركيبات بعد ذلك التي تستخدم بتوسع في العمليات المنطقية حتى يصل إلى أعقد العمليات بأسلوب في منتهى الروعة وندعو الله أن يوفق لترجمته إلى اللغة العربية تيسيرا على القراء إن شاء الله تعالى.

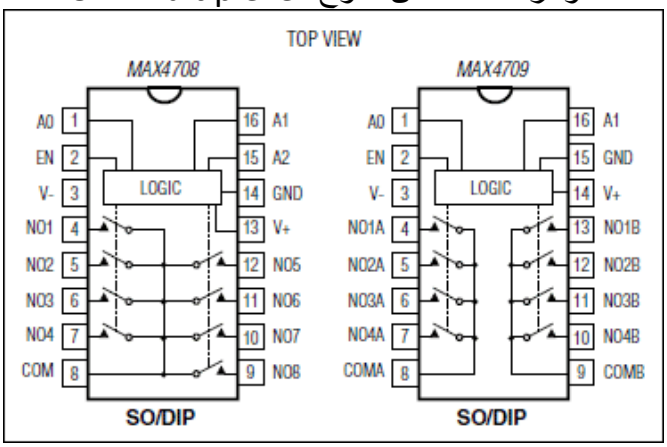
دائرة متكاملة لعناصر RS Flip Flop



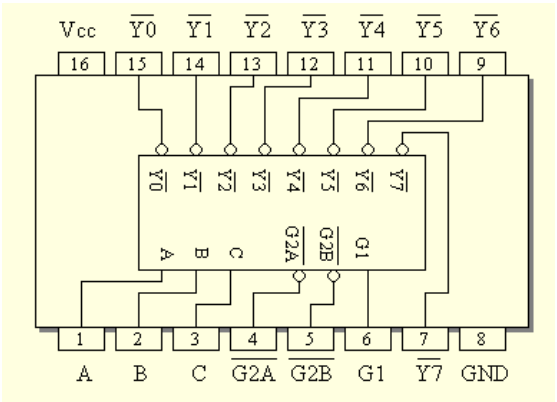
دائرة متكاملة 4-bits Full adder



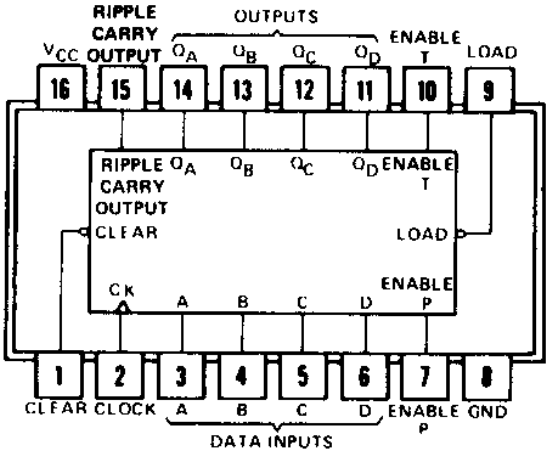
دوائر متكاملة من النوع 8x1 Multiplexer



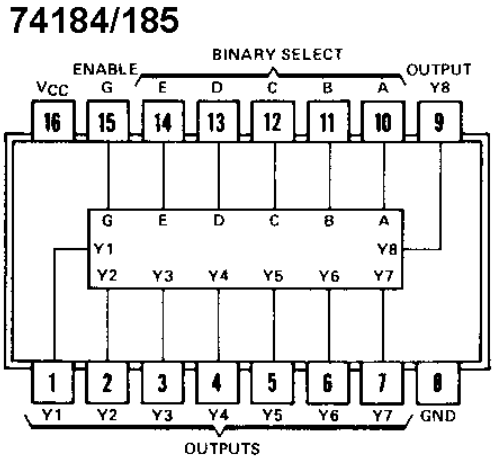
دائرة متكاملة 74138 من النوع Decoder لاختيار 1 من 8 بناء على 3 مدخلات



دائرة متكاملة من النوع Counter

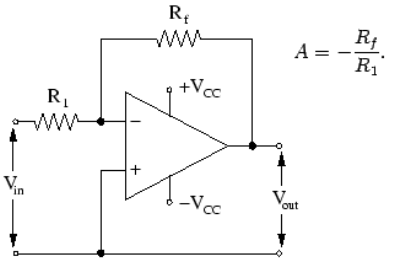


دائرة متكاملة للتحويل من النظام الثنائي إلى BCD



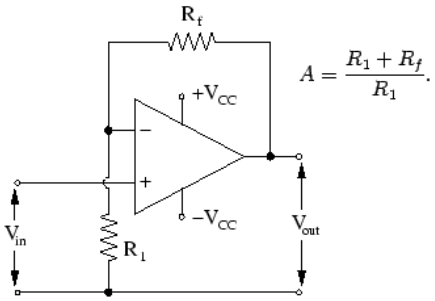
أما على مستوى العمليات التناظرية فهناك أيضا الكثير الذي يمكن تنفيذه عن طريق استخدام الدوائر الإلكترونية خاصة مع وجود مكبر العمليات بأشكاله المختلفة Operational amplifier فباستخدامه يمكن عمل مثل العمليات الآتية :

مكبر عاكس للإشارة



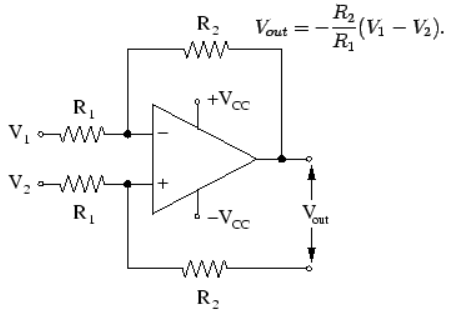
Inverting amplifier circuit.

مكبر غير عاكس للإشارة



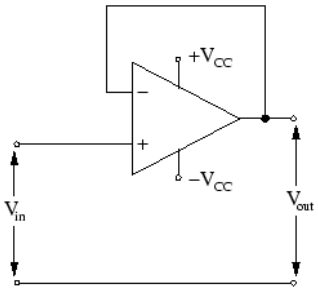
Noninverting amplifier circuit.

مكبر فرق بين إشارتين



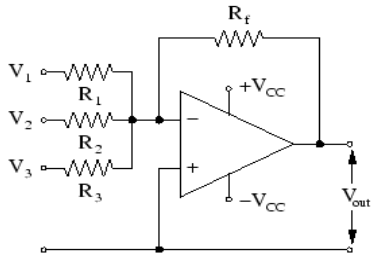
Differential amplifier circuit.

ناقل للإشارة لتحسين مقاومة الدخل



Voltage follower circuit.

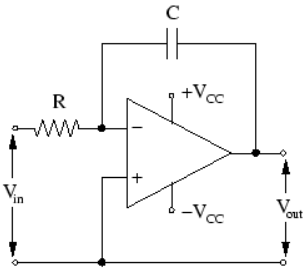
مكبر جمع لأكثر من إشارة



$$V_{out} = -R_f\left(\frac{V_1}{R_1} + \frac{V_2}{R_2} + \frac{V_3}{R_3}\right)$$

Summing amplifier circuit.

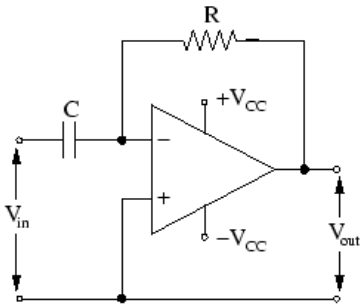
مكبر تكامل



$$V_{out} = -\frac{1}{RC}\int V_{in}dt.$$

Integrator circuit.

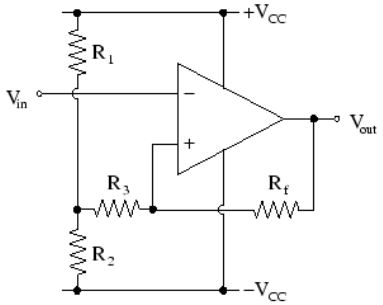
مكبر تفاضلي



$$V_{out} = -RC\frac{dV_{in}}{dt}.$$

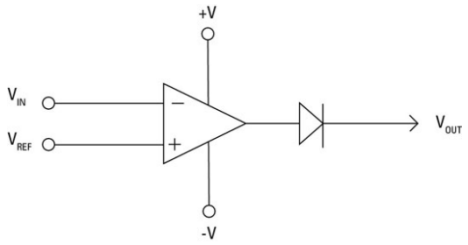
Differentiator circuit.

Schmitt Trigger

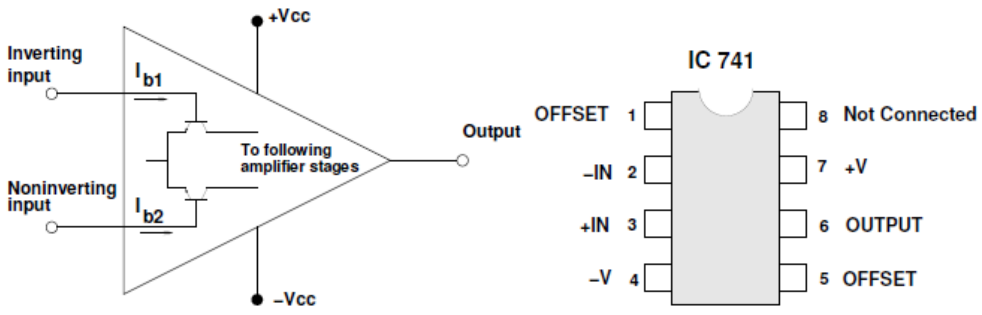


Schmitt trigger circuit. V_{out} and V_{in} are relative to ground, or some reference between $-Vcc$ and $+Vcc$.

استخدام مكبر العمليات للمقارنة

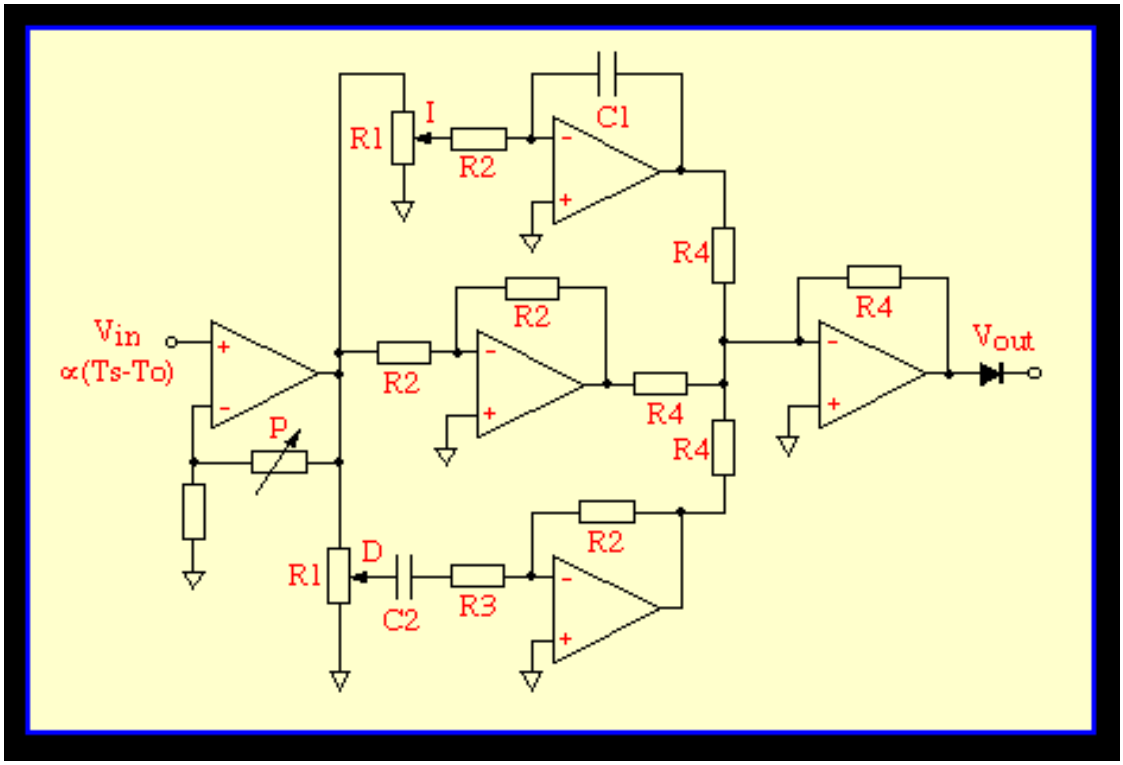


وبمثل هذه الدوائر البسيطة يمكن تنفيذ الكثير من العمليات في عالم التحكم مع الإشارات التناظرية خاصة مع توافر الدوائر المتكاملة التي توظف Op-Amp بشكل كبير مثل الدائرة المتكاملة IC 741 كما في الشكل التالي



Operational ampifier IC 741.

ومن أشهر العمليات التي يستخدم فيها مكبر العمليات هي منظومة PID والتي يلعب فيها مكبر العمليات الدور الرئيسي في الأداء



في الشكل مثال على PID باستخدام مكبر عمليات ومجموعة من المقاومات والمكثفات حيث Kp يتم التحكم فيه عن طريق المقاومة P بينما العامل Ki يتم التحكم فيه عن طريق المقاومة I ويتم التحكم في قيمة Kd عن طريق المقاومة D

ومن أجل أن نضمن أداء سليم لكل الدوائر السابقة فإنها تحتاج منا إلى دوائر تغذية كهربية أو بالتحديد إلى مصدر جهد كهربي Power Supply يتناسب مع متطلبات الدائرة وهناك العديد من أنواع مصادر الجهد التي تعمل مع الدوائر الإلكترونية حسب أنواعها سواء كانت TTL أو CMOS حيث تعمل على الجهود التالية :

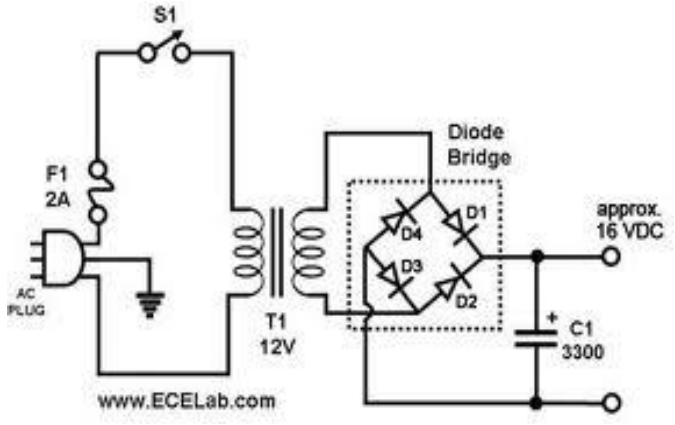
- +5 فولت مستمر (+5 VDC)
- -5 فولت مستمر (-5 VDC)
- +12 فولت مستمر (+12 VDC)
- -12 فولت مستمر (-12 VDC)
- +15 فولت مستمر (+15 VDC)
- -15 فولت مستمر (-15 VDC)
- +24 فولت مستمر (+24 VDC)
- -24 فولت مستمر (-24 VDC)

ولعمل مصادر جهد فإنه يتطلب الأمر منا مجموعة من الأجزاء لابد من توافرها على النحو التالي :

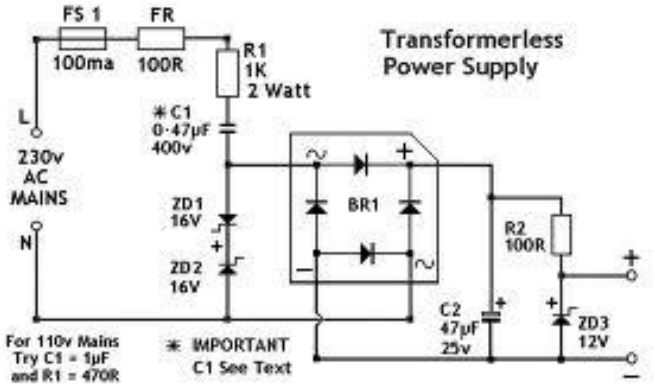
- محول لتحويل الجهد الكهربي للجهد المطلوب مثلا (230 إلى 24 أو 15 إو 5 فولت)
- موحد جهد سواء وصلة ثنائية أو قنطرة توحيد كاملة
- مكثف تنعيم على خروج قنطرة التوحيد
- وحدات منظم جهد لتثبيت الجهد والحصول على القطبية المطلوبة

وسوف نعرض في السطور التالية نماذج لدوائر مصادر جهد مختلفة سواء لها أكثر خرج أو خرج واحد

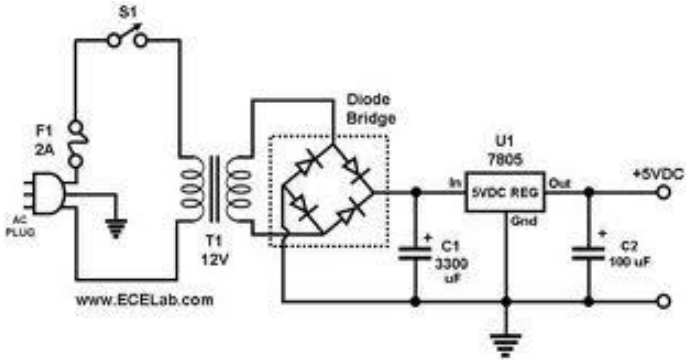
- مثال بسيط لدائرة مصدر جهد عبارة عن محول يقوم بخفض الجهد الكهربي ثم قنطرة توحيد ثم مكثف تنعيم تعطي إشارة تيار مستمر موجة كامة التوحيد



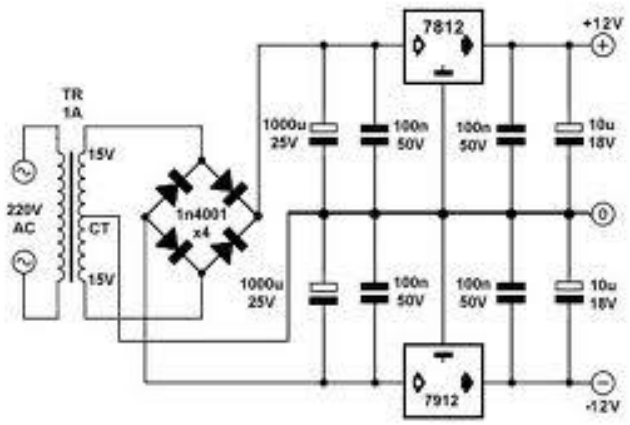
- مثال آخر لمصدر جهد تيار مستمر فولت منخفض بدون استخدام محول وباستخدام قنطرة توحيد مع مقاومة حرارية على الدخل وزينر على الخرج للحماية ضد ارتفاع الفولت



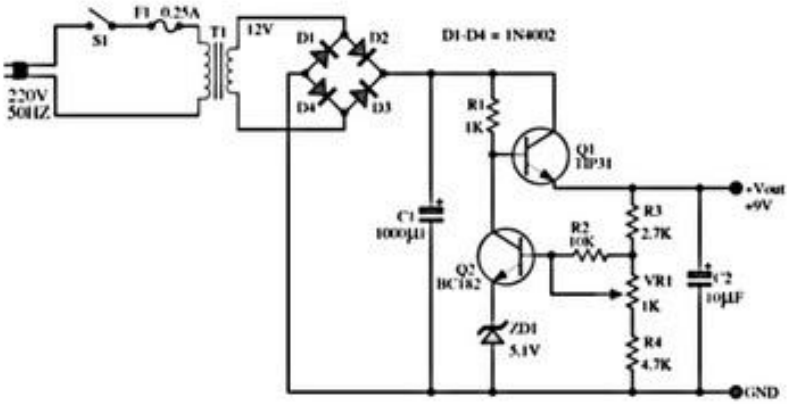
- مثال لمصدر جهد تيار مستمر باستخدام منظم جهد 7805 مع استخدام محول خفض وقنطرة توحيد



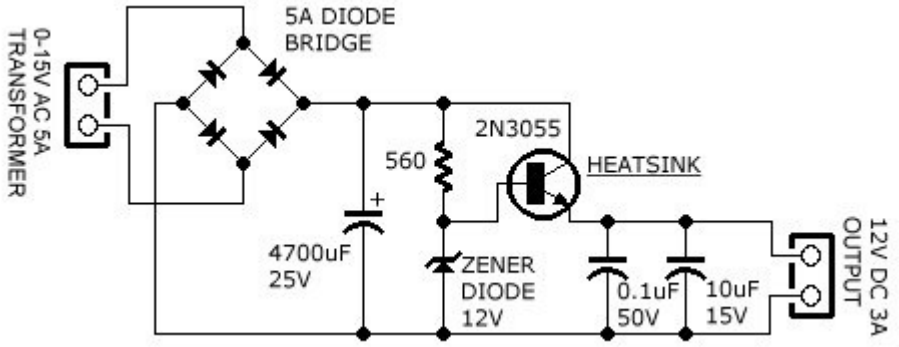
- مثال لمصدر جهد 12+ , 0 , 12- مستمر باستخدام منظمات فولت 7812 و 7912 مع محول خفض وقنطرة توحيد بالإضافة إلى مكثفات توحيد
المحول المستخدم هنا ملفات الثانوي (15,0,15)



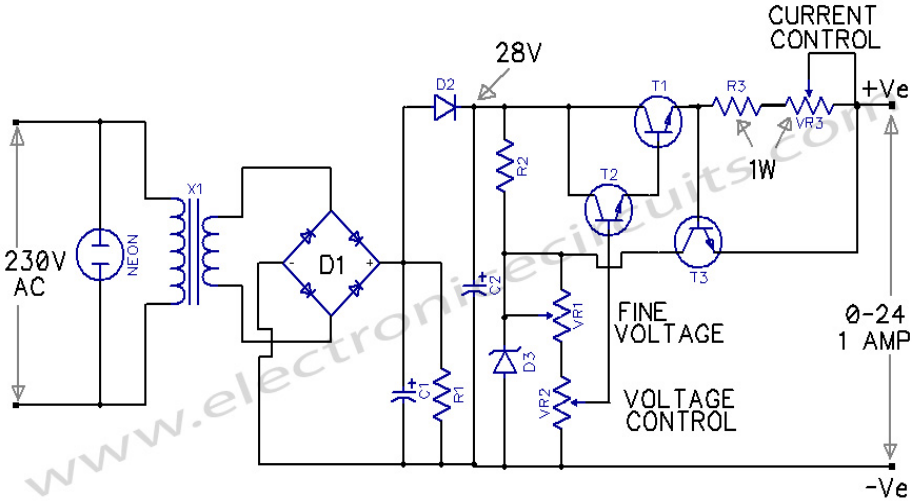
- مصدر جهد قابل للضبط من 6 إلى 12 فولت مستمر



- مصدر جهد 12 فولت 3 أمبير باستخدام ترانزستور باور

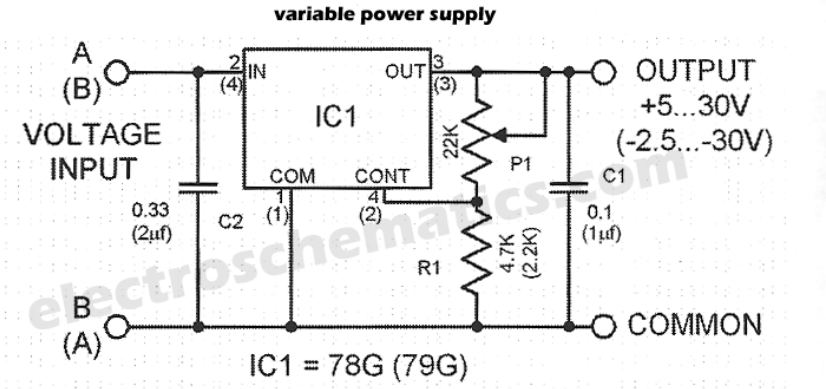


مصدر جهد متغير من 0 إلى 24 فولت 1 أمبير



R1	1kΩ 0.5W
R2	820Ω 0.5W
R3	0.6Ω 1W
VR1	470Ω LIN
VR2	10K LIN
VR3	10Ω 1W
C1	2200µF 50V
C2	200µF 50V
D1	5A Diode Bridge
D2	2A Diode
D3	24V 500mw zener
T1	2N 3055
T2	SL 100
T3	BC 148B
X1	SEC 24V, 1AMP

- مصدر جهد متغير باستخدام دائرة متكاملة



التحكم باستخدام الحاكمات المنطقية Control using PLC

مقدمة:

أصبح التحكم في العمليات الصناعية وغيرها باستخدام الحاكمات المنطقية PLC-Programmable Logic Controllers في الآونة الأخيرة هو الأكثر انتشارا وقدرة وعلى تلبية متطلبات العاملين في المجال حيث أعطت سهولة كبيرة في صياغة العديد من عمليات التحكم في شكل برامج توفر الكثير من الجهد والمكونات بالإضافة إلى سرعة الاستجابة وسهولة عمليات الصيانة نسبيا والمرونة الكبيرة التي تمنحها نظم الحاكمات المنطقية PLC عن غيرها.

والمتحكم المنطقي يجمع بين مواصفات الحاسب الآلي فالجزء الرئيسي فيه عبارة عن وحدة معالجة مركزية CPU تقوم بجميع العمليات الحسابية والمنطقية مستخدمة في ذلك نفس وسائل الحاسب تماما حيث يوجد مسار لنقل البيانات بينها وبين وحدات المنظومة كلها BUS إضافة إلى وحدات ذاكرة لتخزين البرامج عليها بنوعيها النظام وبرنامج المستخدم فمنها الذاكرة غير القابلة للكتابة عليها من النوع ROM والتي تخص النظام والنوع RAM والتي تختص ببرنامج وبيانات المستخدم وكل ما هو عارض الاستخدام أثناء التشغيل ، وإلى جانب ذلك فهناك وحدات الإدخال والإخراج ووحدات تخزين للبرنامج ومنافذ اتصال.

كما يميز المتحكم المنطقي إضافة إلى مواصفات الحاسب التي ذكرناها كونه متحكم فهو يتميز بوجود عناصر إدخال وإخراج قابلة للتعامل مع الإشارات الكهربية القياسية سواء الرقمية منها Digital In/Out أو التناظرية Analog In/Out والتي تسمح بالتعامل مع العمليات الصناعية بشكل مباشر إضافة إلى إتاحة طرق سهلة لكتابة البرنامج تمكن المستخدم من التعامل ببساطة مع العملية الصناعية هذا بالإضافة إلى التوسع مؤخرا في إمكانيات الاتصال بالمتحكمات والعمل مع الشبكات وجميع منظومات الاتصال والتي تتيح تكوين أنظمة تحكم هائلة باستخدام أكثر من متحكم وربطها معا عن طريق إمكانيات الشبكات.

والحاكمات المنطقية ليست نوع واحد أو شكل واحد أو حجم واحد ونقصد بالحجم هنا حجم الإمكانيات التي تمنحها فمنها الصغير جدا Micro PLC حيث تكفي لعمليات بسيطة للغاية تتضمن بعض العمليات المنطقية البسيطة بالإضافة إلى عمليات المؤقتات وربما العد أو المقارنة حتى أنها البعض يطلق عليها ريلاي أو مرحل ذكي Smart Relay ومنها أمثلة كثيرة مثل LOGO من شركة سيمنس و ZELIO من شركة شنايدر أو Easy من شركة مولر وغيرها

وتتميز هذه الوحدات الصغيرة بصغر الحجم وسهولة عملية البرمجة والتي تتم غالبا عن طريق مفاتيح بسيطة على واجهة الوحدة وشاشة صغيرة لبيان النصوص والحالة ويكون حجم البرنامج بسيط يسهل معه استعراضه والتعرف على محتوياته والتعديل فيه أو مراقبة الأداء عبر شاشة بسيطة أثناء التشغيل وقد يكون التشغيل عن طريق برنامج حاسب ويتم التعامل معها مثل كل الحاكمات ومثال لذلك برنامج LOGO Soft من شركة سيمنس للتعامل مع وحدات LOGO

ومع زيادة حجم العمليات نسبيا تتصاعد الحاجة إلى حاكمات ذات مرونة أكبر في عملية التحكم والمواصفات الفنية فنجد ما يسمى بالحاكمات ذات الوحدات Modular PLC حيث تتكون منظومة المتحكم من مجموعة من الوحدات يمكن أن يتغير تركيبها معا حسب الحاجة لتعطي المطلوب لتنفيذ عمليات التحكم وهناك من أنواع الحاكمات ذات الوحدات حاكمات صغيرة ومتوسطة وكبيرة فالحاكمات الصغيرة محدودة بعدد المدخلات والمخرجات التي تتعامل معها وحجم البرنامج والعمليات التي يمكن أن تقوم بها بينما الحجم في الحاكمات المتوسطة أكبر من حيث عدد المدخلات والمخرجات وحجم البرنامج وسرعة التنفيذ والإمكانيات الوظيفية بينما في الحاكمات الكبيرة فعدد المدخلات والمخرجات هائل وحجم البرنامج كبير جدا وسرعة تنفيذ العمليات عالية والعمليات الوظيفية المعقدة متوفرة أيضا ولو ضربنا مثلا بمنتجات شركة سيمنس على سبيل المثال قديما وحديثا فسوف نجد من العائلة القديمة لسيمنس مثلا عائلة Step-5 فمنها:

- حاكمات صغيرة مثل S5-90U, S5-95U, S5-100U
- حاكمات متوسطة مثل S5-115U
- حاكمات كبيرة مثل S5-135U, S5-155U



S7-200

ويمكن اعتبار المتحكم S7-200 من الحاكمات الصغيرة أيضا

بينما في الأنظمة الحديثة فقد تم استبدال S7-200 بالمتحكم S7-1200 وحلت الحاكمات S7-300 محل الحاكمات الصغيرة والمتوسطة حيث يوجد نطاق عريض من المتحكمات S7-300 بينما المتحكم S7-400 فقد حل محل المتحكمات الكبيرة بل والهائلة.

S7-400 S7-300 S7-1200



وأنظمة الحاكمات كي نستخدمها بشكل سليم وبكفاءة فلابد من توافر مجموعة من المتطلبات وهي :

1- مكونات أو عتاد يوفي بمطالب النظام Hardware أو وحدات النظام Modules فالمكونات هي التي نقوم باستخدامها لفعل كل شئ وكل منظومة لمتحكم لها مكونات أساسية لابد منها وهي :
   - حامل للمكونات وناقل للبيانات Rack and Bus system حيث يتم عليه تثبيت المكونات واستخدامه في نقل التغذية الكهربية للوحدات وتبادل البيانات أيضا بين الوحدات
   - وحدات تغذية كهربية Power supply تعطي التغذية الكهربية المطلوبة لدوائر التشغيل والتحكم الكهربية الخارجية والداخلية
   - وحدة معالجة مركزية CPU حيث يتم فيها تنفيذ جميع العمليات الحسابية والمنطقية ومراقبة المكونات وإدارة الذاكرة وإدارة الاتصال إلى غير ذلك من وظائف وحدات المعالجة
   - وحدات إدخال رقمية Digital input modules حيث يتم عن طريقها إدخال الإشارات الرقمية إلى النظام في صورة كهربية قياسية
   - وحدات إخراج رقمية Digital output modules حيث يتم عن طريقها إخراج الإشارات الرقمية الناتجة عن النظام في صورة كهربية قياسية
   - وحدات إدخال تناظرية Analog input modules حيث يتم عن طريقها إدخال الإشارات التماثلية إلى البرنامج في صورها الكهربية القياسية المختلفة
   - وحدات إخراج تناظرية Analog output modules حيث يتم عن طريقها إخراج الإشارات التناظرية الناتجة عن البرنامج في شكل إشارات كهربية تناظرية
   - وحدات وظائف خاصة مثل العدادات وقياس التردد والتحكم في الوضع وغيرها
   - وحدات إدارة اتصال خاصة بتنظيم عمليات الاتصال بأشكالها المختلفة
2- نظام برمجة أو برنامج لإدارة عملية البرمجة والمراقبة وتحميل البيانات من وإلى المتحكم ولكل أنواع المتحكمات يوجد برنامج لتنفيذ تلك العمليات فمثلا في منظومة S5 يستخدم برنامج STEP5 وفي منظومة S7-200 يستخدم برنامج MicroWin وفي منظومة S7-300/400 يستخدم برنامج Simatic manager وكل من هذه البرامج لابد أن تتوفر فيه مجموعة من الشروط أو العمليات الضرورية يقوم بها: 47
   - عمليات إدارة الملفات (إنشاء جديد – فتح – حذف – حفظ .... إلخ)
   - عمليات تحرير للبرنامج الخاص بالمتحكم
   - عمليات تصحيح أثناء كتابة البرنامج
   - عمليات اتصال بالمتحكم (نقل إلى – نقل من – مراقبة – تشغيل/إيقاف .... إلخ) للمتحكم
   - مكتبة للعمليات الخاصة يستعين بها المبرمج أثناء البرمجة
   - مساعدة في كتابة البرنامج وأدوات البرمجة يستعين بها المبرمج أثناء البرمجة
   - إمكانية تشخيص الأعطال في الوحدات أثناء التشغيل
3- أدوات اتصال بين جهاز البرمجة والمتحكم وفي الغالب يكون كابل خاص مثل كابل USB-TTY Converter في منظومة S5 وكابل PC/PPI Cable USB بالنسبة لمنظومة S7-200 وكابل PC-Adapter USB بالنسبة لمنظومة S7-300/400

USB-TTY (S5 cable)

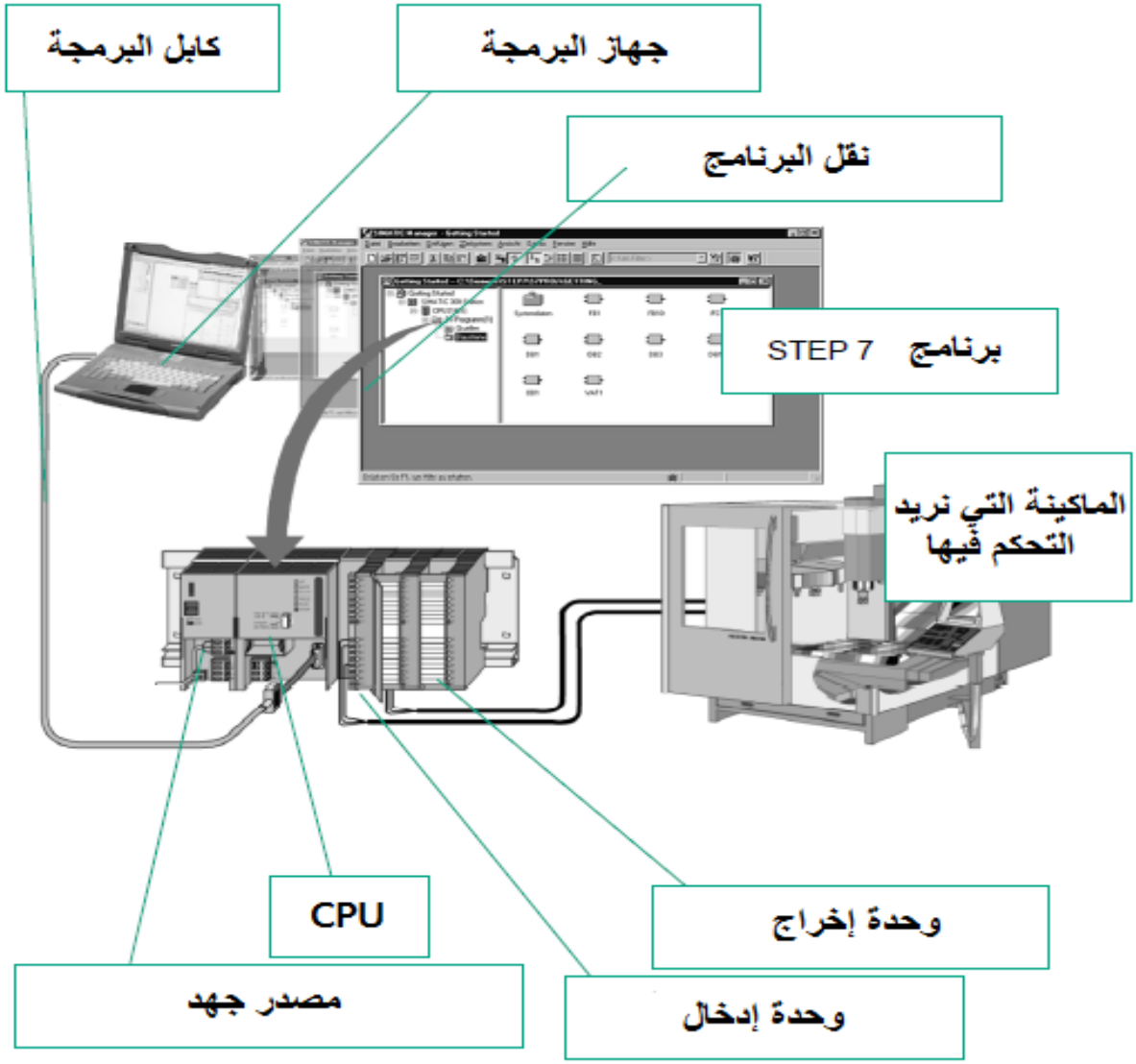
PC/PPI Cable USB (S7-200)

PC-Adapter USB (S7-300/400)

4- طرق البرمجة وهي كثيرة والأشهر منها والأكثر انتشارا هي ثلاثة وهي التعليمات النصية Statement List أو STL حيث يتم كتابة البرنامج في شكل تعليمات نصية أو أكواد من أعلى لأسفل والشكل الثاني منها هو البرمجة باستخدام المخطط السلمي Ladder diagram أو LAD حيث يتم كتابة البرنامج في شكل رسوم تأخذ شكل السلم من الشمال إلى اليمين ومن أعلى لأسفل والنوع الثالث وهو البرمجة باستخدام الرسم الوظيفي بالقوالب Function block diagram أو FBD حيث يتم التعبير عن العمليات في شكل قوالب لها مدخلات ومخرجات ويتم كتابة اسم العملية عليها والطرق الثلاث متوفرة في معظم أنواع المتحكمات المعروفة ويمكن باستخدام برنامج البرمجة الانتقال بينها.

5- برنامج التطبيق الخاص بعملية التحكم وهو البرنامج الذي يكتبه المبرمج User program حيث يحدد فيه المطلوب تنفيذه من المتحكم وهو ما سينتج إن شاء الله في نهاية هذه المجموعة من المحاضرات والتي هدفها برمجة المتحكم المنطقي

هذه المتطلبات الخمسة هي القواعد التي سنبني عليها عملنا في باقي الحلقات القادمة إن شاء الله من هذه السلسلة.



48

والآن ننتقل مباشرة إلى نوعية المتحكم الذي سيكون محور اهتمامنا كمثال عملي على باقي موضوعات هذه السلسلة من أول التعرف على الإمكانيات العامة وحتى البرمجة إن شاء الله تعالى وهو منظومة Step-7 والتي تشمل المتحكمات S7-300 وS7-400 حيث سنفصل في حديثنا عنها إن شاء الله تعالى وبنفس التسلسل الذي بدأنا به

منظومة الحاكمات S7-300/400 مثلها مثل غيرها من الحاكمات لها مكونات وهي العتاد Hardware وبرنامج البرمجة والتعامل مع المتحكم وهي Simatic manager ووسيلة الاتصال وهي متعددة مثل PC-Adapter USB أو PC-Adapter RS232 ومنها أنواع أخرى سنتعرف عليها في حينها إن شاء الله تعالى وتسمح أيضا بالبرمجة بطرق البرمجة الثلاث الشهيرة LAD/STL/FBD وبها مجموعة متميزة من الدوال والعمليات القياسية تحت مكتبة متميزة Library يتم استخدامها بنتهى البساطة داخل برنامج المستخدم

وسوف نبدأ كلامنا عن برنامج مدير السيماتيك Simatic manager حيث بيئة العمل الرئيسية والذي من خلاله يتم تنفيذ كل شئ من برمجة ومراقبة لمتحكم ومن أدق العمليات إلى تهيئة المكونات فكل شئ يتم عن طريق Simatic manager والذي يتوافر منه الإصدار V5.5 حتى ديسمبر 2012 تاريخ تحرير هذه السطور وهو يعمل مع برنامج التشغيل ويندوز 7 أو فيستا أو Xp-SP3 ويتوفر البرنامج من شركة سيمنس على اسطوانات مدمجة يتم تثبيته ببساطة على جهاز الحاسب الشخصي حيث يتم التجاوب مع البرنامج والذي يعمل بشكل آلي بمجرد تشغيل الاسطوانة المدمجة لاختيار مواصفات التشغيل المناسبة.

وفي النهاية سوف يطلب البرنامج إعادة التشغيل قم بإعادة التشغيل ولا تفتح البرنامج قبل عمل الترخيص حيث يتم توريد قرص عليه رخصة البرنامج وسوف تجد بعد التثبيت على سطح المكتب أيقونة البرنامج  وأيقونة برنامج إدارة الترخيص  حيث يمكن عن طريقها تشغيل برنامج إدارة الترخيص لنقل الرخصة من على القرص المرن إلى الجزء الذي تم تثبيت البرنامج عليه في جهازك

وبعد انتهاء الترخيص قم بفتح البرنامج للتشغيل أول مرة وسوف نتعرض في السطور القادمة إن شاء الله للمصطلحات الأساسية للتعامل مع برنامج Simatic manager

المشروع Project:

يتعامل برنامج Simatic manager مع المشاريع مثلما يتعامل برنامج Windows explorer مع الملفات والمشروع Project هو المحتوي لكل ما يتم تنفيذه في منظومة التحكم باستخدام المتحكم S7-300/400 وكل العناصر الأخرى هي جزء من المشروع ولا يتم أي شئ إلا تحت مشروع

يتكون المشروع من عناصر فرعية مباشرة تحت المشروع تسمى المحطات Stations والتي تعبر عن مستوى المكونات والتي يمكن أن تكون منظومة S7-300 أو S7-400 أو S7-200 أو S5 أو HMI أو PC Station أو غيرها من المتحكمات الأخرى فالمشروع يتكون من مجموعة محطات

المحطة Station:

المحطة Station هي الوحدة الفرعية لبناء المشروع وتتكون من المكونات Hardware والبرامج User Program وطريقة الاتصال مع المشروع أو مع عناصر المشروع الأخرى MPI/PROFIBUS وكما ذكرنا يمكن أن تكون بأشكال كثيرة

العتاد أو المكونات Hardware:

المكونات هي مجموعة الوحدات التي تتكون منها منظومة المتحكم والبيانات الخاصة بضبطها وكيفية عملها وتصنيفها وترتيبها وتثبيتها ويتم كل هذا باستخدام برنامج ليعطي صورة برمجية عن المكونات والذي يجب أن يماثل ويحاكي الواقع تماما

المكتبة Library:

وهي مجموعة من الدوال القياسية التي تم دمجها في برنامج Siamtic manager بحيث تقوم بعمليات قياسية محددة حيث يمكن استدعاؤها من داخل البرنامج وتوظيفها وتحميلها مع التوظيف مع برنامج المستخدم كجزء منه

البلوك Block:

هو وحدة بناء برنامج المستخدم وتعليماته وبياناته حيث تضم كل مجموعة من البيانات قالب أو بلوك واحد وتضم كل حزمة من التعليمات أيضا قالب أو بلوك واحد ومنها نوعان قالب بيانات Data Block حيث يحتوي على بيانات فقط وقالب منطقي Logical Block حيث يحتوي على تعليمات وبيانات وبالتالي فأي تعليمات أو بيانات لابد من كتابتها داخل أحد القوالب ويتكون قالب التعليمات المنطقي من مجموعة من الحلقات تسمى Networks حيث يتم كتابة جزء من التعليمات في كل واحدة من هذه الحلقات Networks سواء كانت مرتبطة معا أو غير مرتبطة وذلك لتسهيل عملية كتابة البرنامج.

وسيلة الاتصال MPI/PROFIBUS:

طريقة الاتصال الافتراضية في منظومة S7-300/400 هي نظام الواجهة متعددة النقاط Multi-point interface والذي يطلق عليه اختصارا MPI وهو على المنفذ الرئيسي 9-Pin المتوفر في كل الوحدات حيث يتم البرمجة والمراقبة والاتصال عن طريقه ويمكن أيضا أن يتم ضبطه ليعمل بطريقة PROFIBUS للبرمجة أو المراقبة أو التشغيل أو الاتصال بعناصر أخرى أيضا وهناك أجهزة محددة تخدم عمليات الاتصال بالمنفذ ومنها PC-Adapter USB وكذلك PC-Adapter RS232 ومنها كذلك وحدات يتم تثبيتها داخل الحاسب العادي تسمي CP5611 إلى غير ذلك.

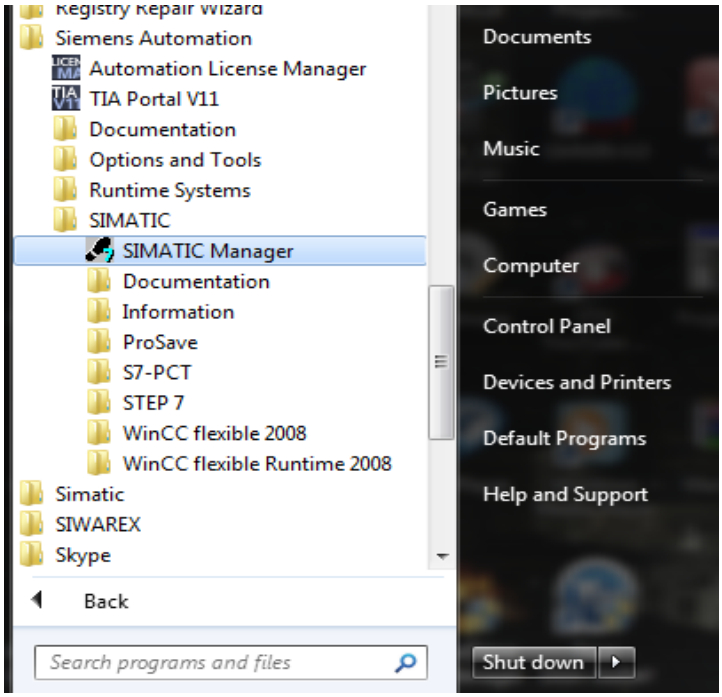
50

جهاز البرمجة PG:

هو الجهاز الذي نستخدمة لكتابة البرامج وضبط المكونات ونقل البرنامج من وإلى المتحكم ومراقبة التشغيل وتتبع الأعطال وتحليل بيانات الأعطال بتثبيت برنامج Simatic manager عليه وتحقيق وسيلة اتصال تتوافق مع المتحكم.

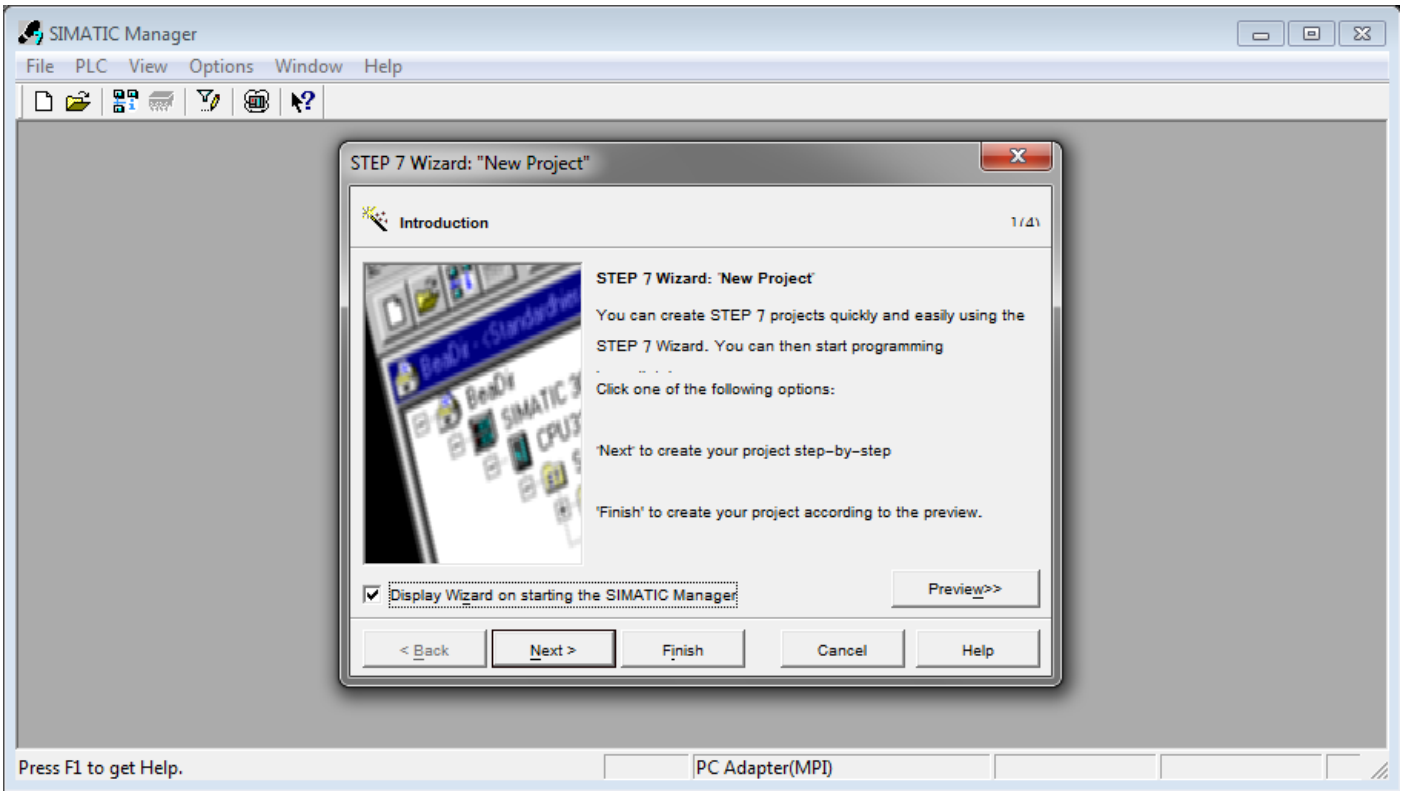
والآن نستعرض معا مختلف أنواع النوافذ أو الشاشات المحتمل التعامل معها في برنامج Simatic manager والتي سنحتاجها قطعا أثناء أعمال كتابة البرنامج أو نقله من وإلى المتحكم Download/Upload ومراقبة التشغيل Monitoring أو تتبع الأعطال Diagnostic

الشاشة الرئيسية (الافتتاحية) Simatic manager main screen:



عند تشغيل برنامج Simatic manager سواء باستخدام أيقونة سطح المكتب أو من خلال القائمة الرئيسية لويندوز تحت الدليل Siemens Automation ← SIMATIC ← SIMATIC Manager

سوف يفتح البرنامج بالشكل التالي مفترضا إجراء إنشاء مشروع بشكل منهجي New Project Wizard

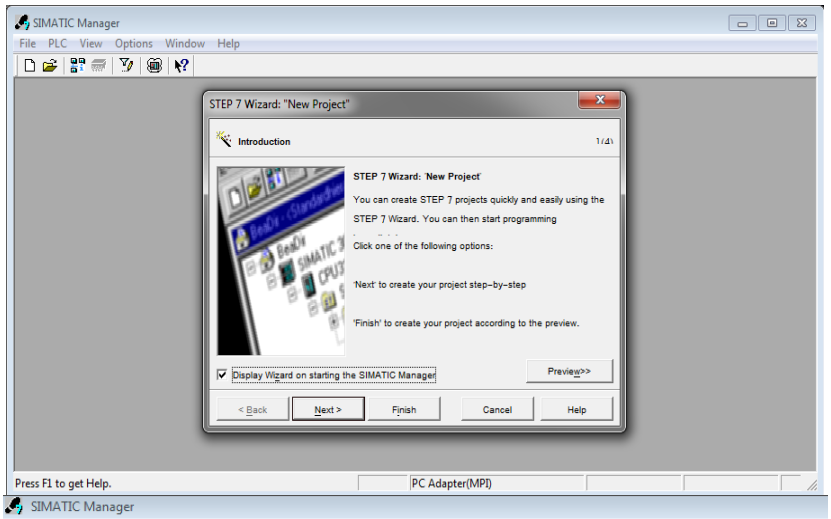


حيث يمكن إنشاء مشروع بخطوات استرشادية عن طريق البرنامج ويمكن إلغاء ظهور مثل تلك النافذة مستقبلا بإلغاء الاختيار ويمكن الاستمرار في نفس المشروع بالخطوات الاسترشادية بالضغط على Next أو الإلغاء بالضغط على Cancel لفتح البرنامج بدون مشروع أو على آخر مشروع مفتوح لم يتم غلقه

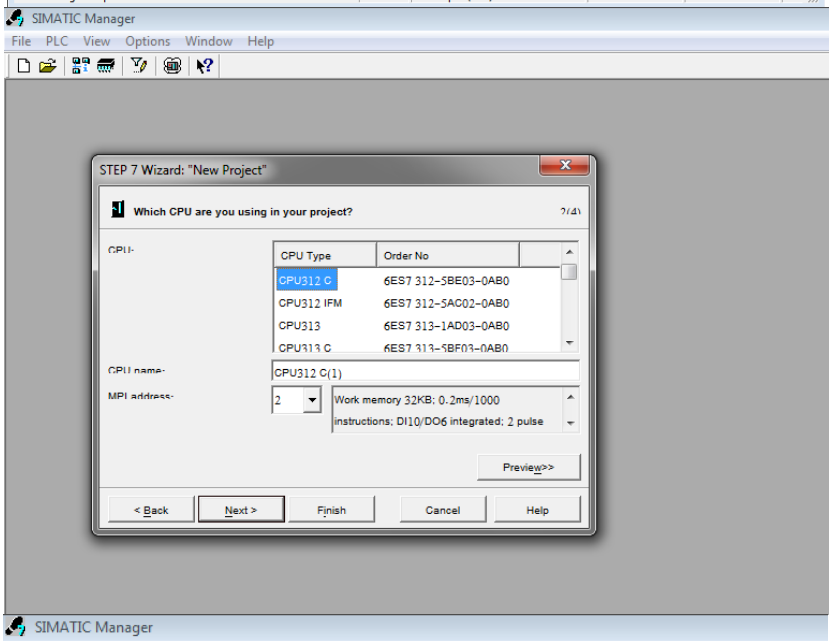
العمل مع برنامج SIMATIC Manager

إنشاء المشروع Creating Project:

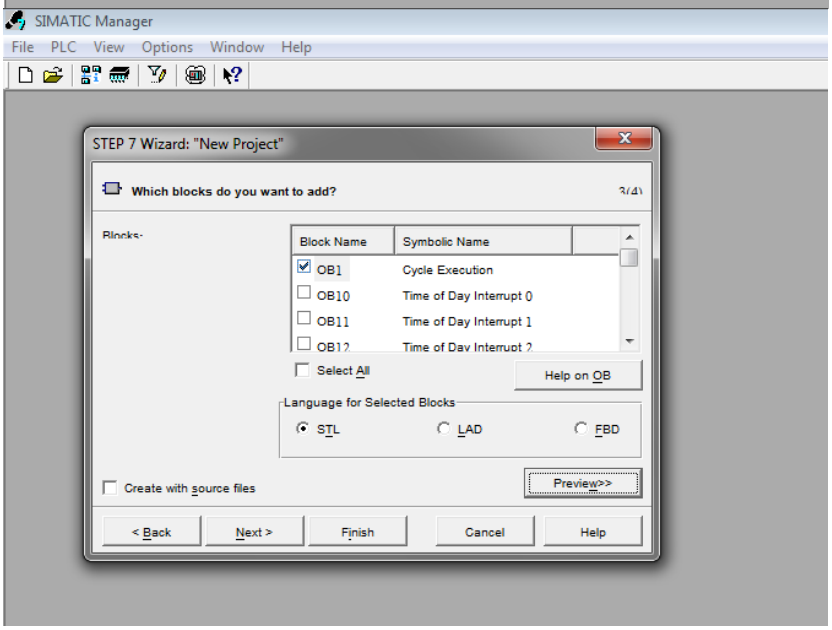
يتم إنشاء المشروع بإحدى طريقتين الأولى بمساعدة البرنامج في اختيار عناصر التشغيل عن طريق New Project Wizard حيث يساعد في تهيئة بيئة العمل بشكل آلي بأقل الاختيارات مثل الآتي :



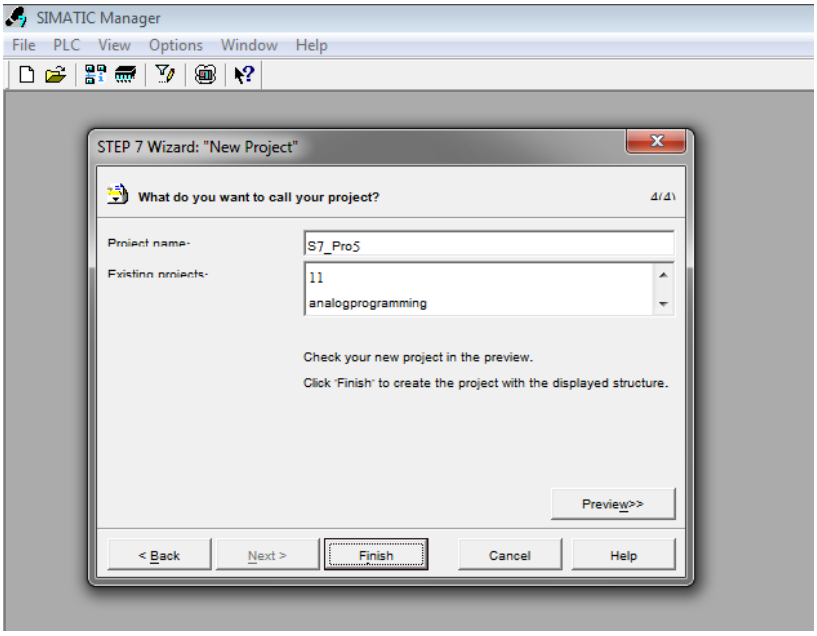
1- تظهر هذه الشاشة الافتتاحية وبالضغط على Next ننتقل إلى المرحلة الثانية وهي مرحلة اختيار CPU



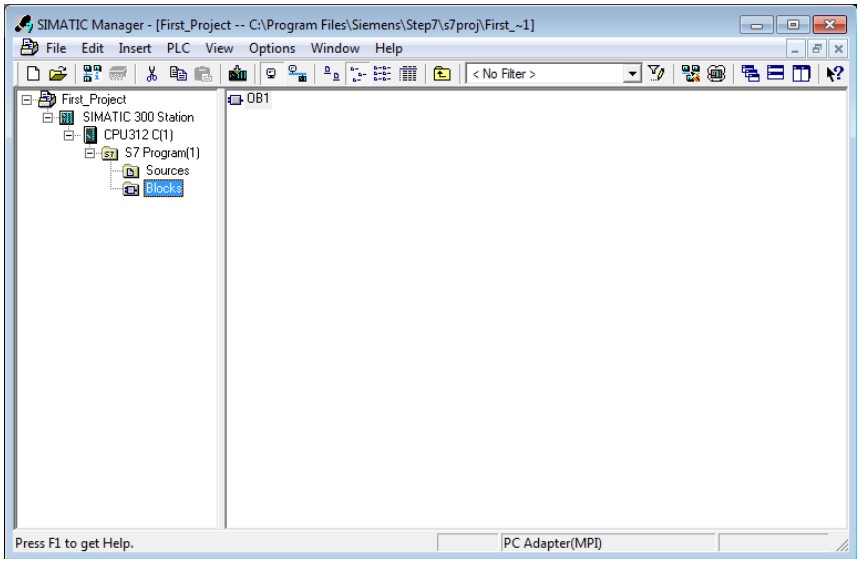
2- نقوم باختيار وحدة المعالجة CPU حيث لا يتم كتابة برنامج إلا بعد تحديد وحدة المعالجة ونقوم بتحديد عنوان MPI للوحدة أو نتركه بالقيمة الافتراضية (2) ثم نضغط Next للانتقال للمرحلة الثالثة وهي مرحلة اختيار لغة البرمجة والبلوكات OB المطلوب إنشاؤها مع الوحدة



3- نقوم باختيار قوالب أو بلوكات OB وعلى الأقل OB1 حيث يحتوي على الدورة الرئيسية للبرنامج واختيار طريقة البرمجة STL/LAD/FBD ثم اضغط Next للانتقال للمرحلة التالية وهي مرحلة تسمية المشروع حيث نقوم باختيار اسم للمشروع ، اضغط Next للانتقال إليها



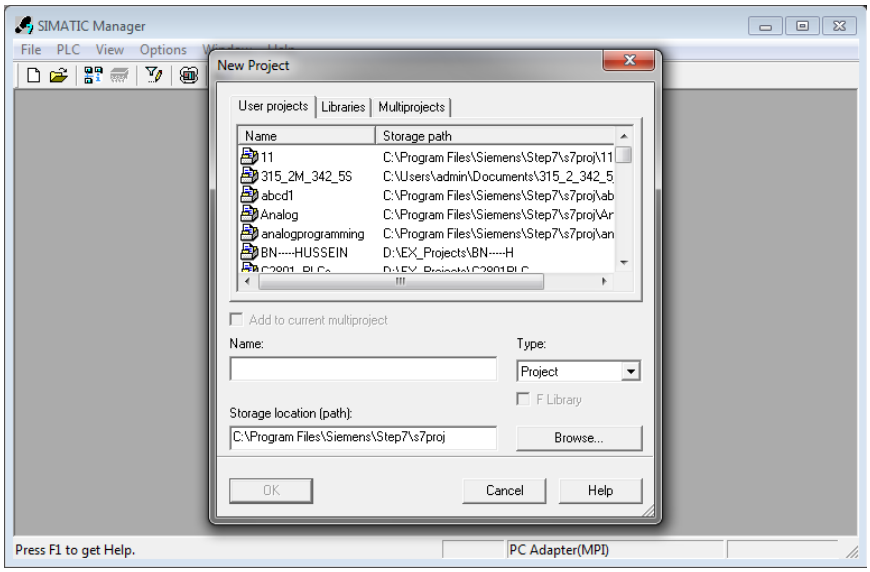
4- قم باختيار اسم للمشروع في السطر Project name ثم اضغط Finish ليتم إنشاء المشروع بالاختيارات التي قمت بها



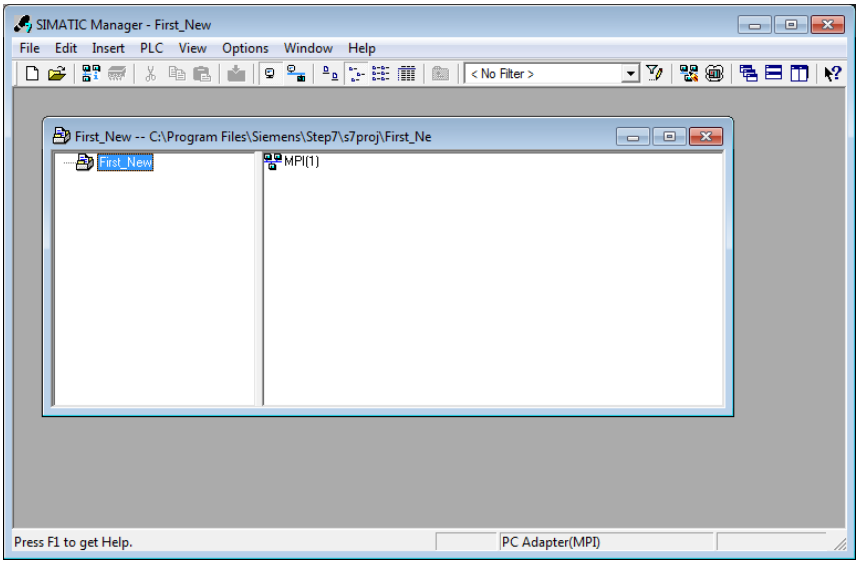
وبعد الانتهاء من Wizard يفتح البرنامج المشروع على الشكل الذي أمامنا والذي يشتمل على الآتي بعد فتحه:

- ضبط المكونات ويحتوي على وحدة المعالجة CPU
- برنامج S7 والذي يحتوي على البلوكات التي اخترتها على الأقل OB1 إن لم تختر بلوكات أخرى
- منفذ اتصال MPI

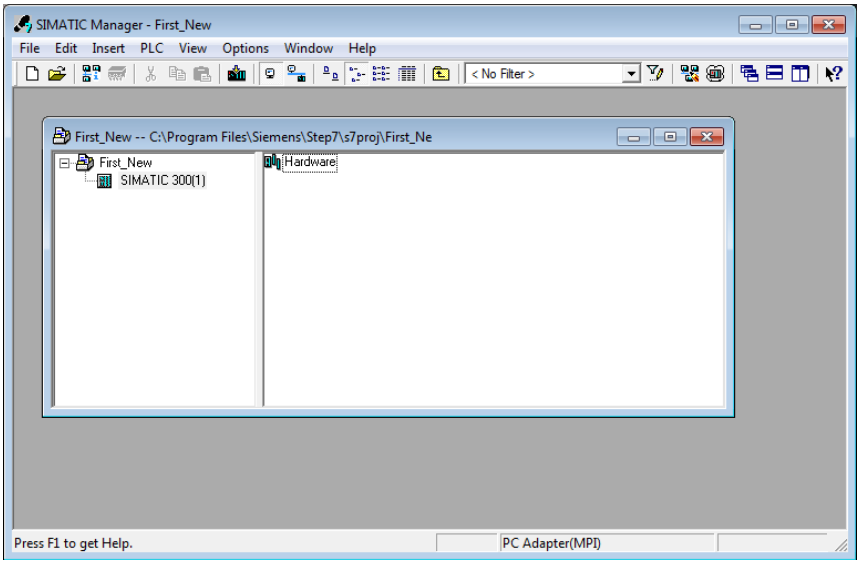
أما الطريقة الثانية فهي عن طريق New Project سواء عن طريق القائمة Project ثم New أو عن طريق الضغط على أيقونة جديد من شريط الأدوات مباشرة حيث نقوم بالخطوات كالآتي:



1- بالضغط على Project ثم New أو على أيقونة جديد يفتح لنا صندوق مثل الذي في الصورة لتحديد اسم المشروع الجديد ومكان تخزينه مع عرض قائمة بالمشاريع الموجودة من قبل قم باختيار الاسم والمكان ثم اضغط OK

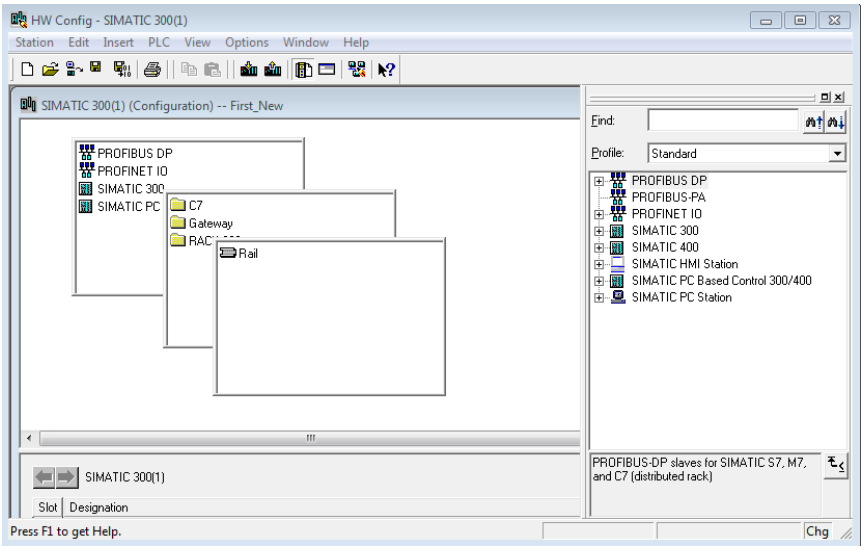


2- يظهر المشروع خالي تماما ليس فيه شئ إلا عنصر الاتصال الافتراضي MPI حيث يشترط وجوده في العناصر التي سيتم إنشاؤها وهنا يمكن إدراج المكونات بشكل تدريجي يدويا حيث سيتم إدراج محطة S7-300 إما من قائمة Insert أو بالوقوف في الجزء الأيمن ثم الضغط ماوس يمين ثم اختيار إدراج محطة S7-300 وبعد إدراجها سنجد داخلها عنصر Hardware والذي سنقوم بفتحة لإدراج وضبط المكونات



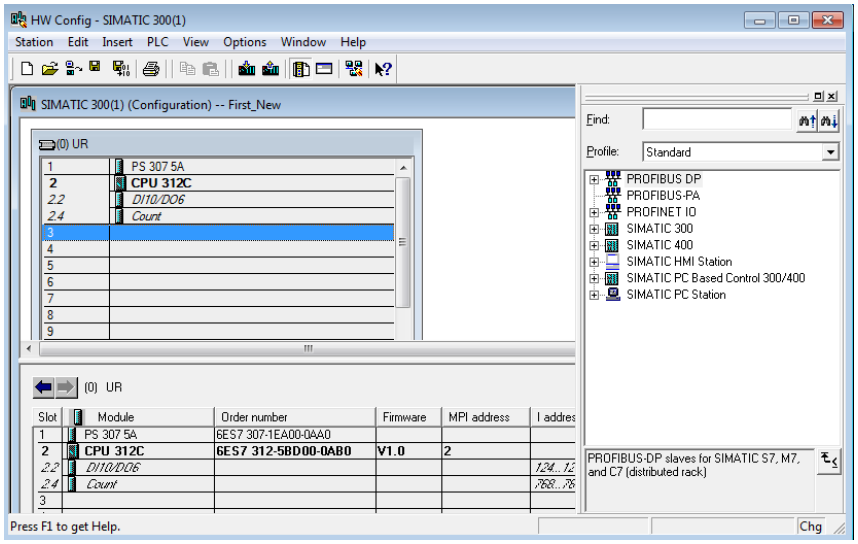
3- بالضغط على العنصر Hardware تظهر لنا شاشة جديدة هي شاشة تهيئة المكونات Hardware Configuration والتي عن طريقها نقوم بتكوين منظومة المتحكم وعمل الضبط اللازم لكل عناصرها وسنكتفي هنا بمصدر تغذية Power supply ووحدة معالجة CPU على راك

54

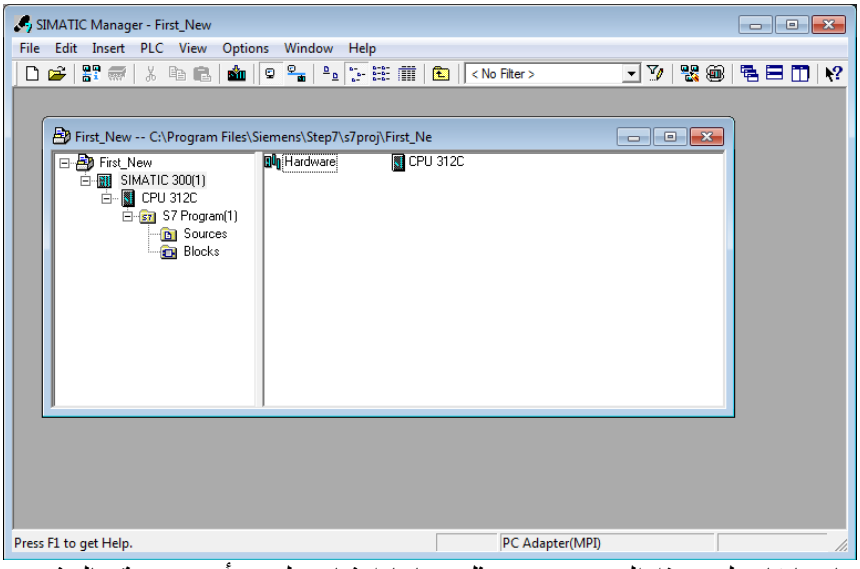


4- نقوم أولا بإدراج Rail الذي نثبت عليه المكونات سواء من قائمة Insert في شاشة HW Config أو بالضغط ماوس يمين ثم اختيار Insert new object ثم نختر SIMATIC 300 ثم RACK 300 ثم Rail كما في الشكل وسوف يظهر لنا جدول به 11 صف تمثل عدد العناصر الممكن إدراجها عليه وندخل عنصر Power supply في الخانة-1 ثم وحدة المعالجة CPU في الخانة-2 وسوف نختر الوحدة CPU 312-c كمثال نعمل به وهذا سواء من قائمة Insert أو بالوقوف في المكان على Rail والضغط ماوس يمين ثم اختيار إدراج عنصر جديد

5- لاحظ في الشكل أمامنا تم إدراج وحدة Power supply ووحدة معالجة CPU 312-C اضغط على التخزين ليتم تكوين صورة جاهزة من المكونات ليتم عليها تنفيذ البرنامج وسوف تلحظ اختلافا عند العودة إلى الشاشة الرئيسية لبرنامج SIMATIC Manager حيث ستظهر عناصر جديدة عن التي تركناها آخر مرة قبل تهيئة المكونات



لاحظ التغير على المشروع في الشكل أمامنا فقد ظهرت إلى جوار أيقونة Hardware عنصر آخر وهو CPU 312C كذلك في الجزء الأيسر فقد ظهر أيضا عنصر S7-Program وتحته عنصران هما Sources و Blocks



55

وبالتالي فإنشاء المشروع يمكن إنفاذه كما ذكرنا سابقا على هذا النحو وهو يمثل تماما إنشاء ملف بأي صيغة فالمشروع هو الملف الكبير الذي يحتوي كل العمل ، وكما أنشأنا الملف الجديد أو المشروع الجديد فإنه يمكن أيضا تنفيذ كل عمليات الملفات مثل فتح مشروع موجود Open Project سواء من قائمة File أو باستخدام أيقونة فتح مشروع مباشرة من شريط الأدوات  حيث سيفتح لنا صندوق لاختيار الدليل واختيار المشروع الذي نريد فتحه كما يمكن استخدام إمكانية البحث الآلي عن أي مشاريع Projects على الجهاز أو على جزء ما من الجهاز حيث بعد البحث يعطي قائمة بكل المشاريع الموجودة ويلاحظ أيضا أن جميع المشاريع التي تم فتحها من قبل تظل مخزنة بمسارها عند الرغبة في فتح مشروع موجود يمكن البحث في تلك القائمة والتي يتم ترتيبها أبجديا.

كذلك يمكن حفظ المشروع باسم آخر غير الذي تم إنشاؤه به من خلال قائمة File ثم Save As كما يمكن غلق مشروع مفتوح أو حذفه أو ضغطه وتخزينه Archive أو فك ملف مضغوط Retrieve كذلك يمكن طباعة المشروع حيث لطباعة المشروع مدير طباعة يقوم بتنسيق عملية الطباعة

كذلك سوف تجد جزء خاص للتعامل مع Memory Card ويتم ذلك حال الاتصال مع المتحكم حيث يمكن نقل بيانات إلى كارت الذاكرة أو منه ، كل هذه الإمكانيات سوف تجدها تحت قائمة File مثلها مثل كل التطبيقات التي تتعامل مع الملفات في تطبيقات الويندوز.

أما القائمة الثانية على مستوى الشاشة الرئيسية فهي قائمة Edit وعند الوجود على مستوى المشروع فهي تختص بعمليات التحرير على مستوى المحطات حيث يمكن الحذف Delete أو النسخ Copy أو القص Cut أو اللصق Paste وتغيير الاسم Rename وعرض الخصائص Properties وإنشاء عناصر مراقبة Monitoring لمراقبة عناصر البرنامج.

أما القائمة الثالثة على مستوى الشاشة الرئيسية وهي قائمة Insert وهي تختص على مستوى Project بإدراج محطات بأنواعها المختلفة أو عناصر شبكات للاتصال بين المحطات أو برامج S7-Programs ويلاحظ أنه تضاف عمليات أخرى بالانتقال للمستويات الأقل سواء على مستوى HW Config أو غيرها من الشاشات الأخرى التي سنتعرف عليها لاحقا.

والقائمة الرابعة وهي قائمة PLC والتي تشمل على هذا المستوى إما تحميل محتويات محطة إلى PLC أو رفع محتويات PLC إلى المشروع أو التعامل مع نقاط الاتصال الموجودة بالفعل Accessible nodes أو التعامل مع حالة التشغيل لوحدات PLC أو تتبع الأعطال Diagnostic

أما القائمة التالية فهي قائمة View أو العرض وتختص باختيار وتنسيق طريقة العرض بالنسبة لعناصر الصفحة وشرائط الأدوات وشكل الأيقونات إلى غير ذلك من طرق العرض

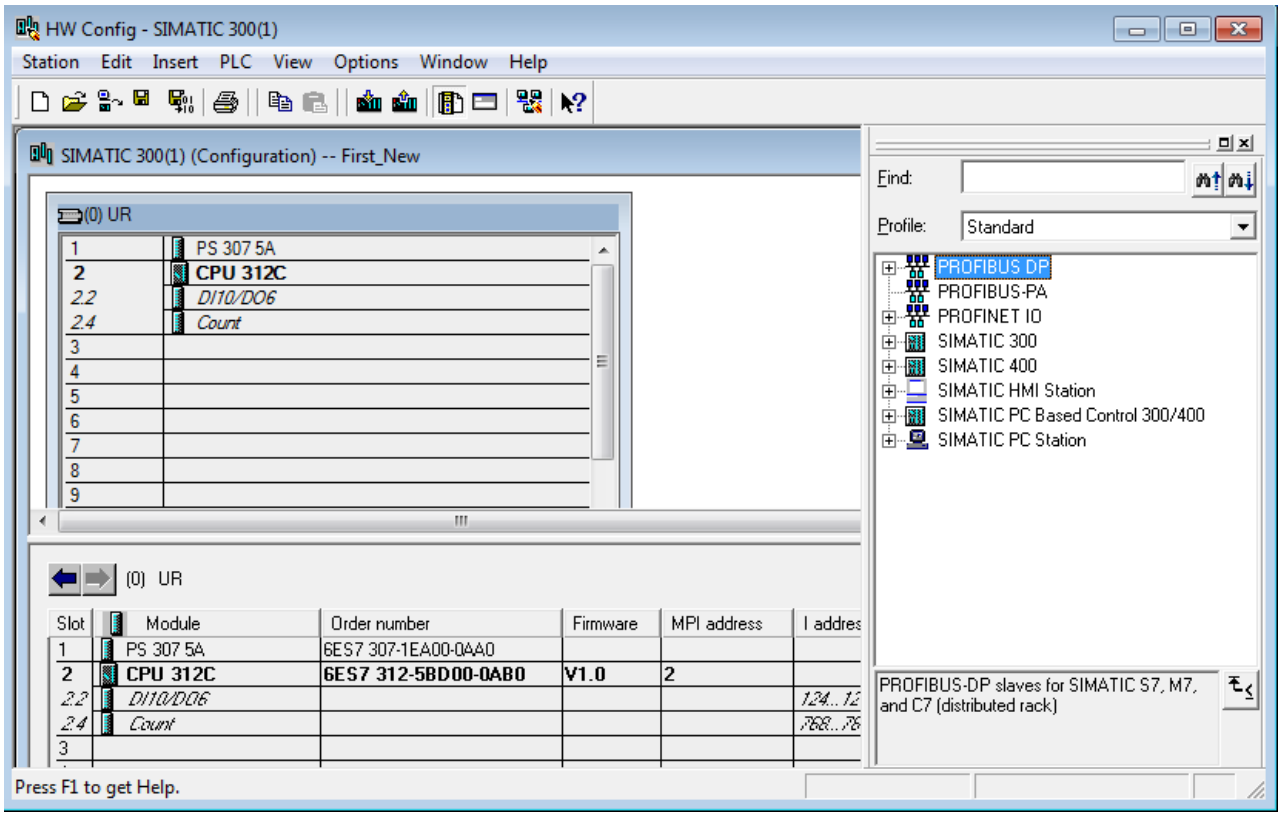
أما القائمة Options فتختص بضبط خصائص البرنامج والتعامل مع البيئة المحيطة وضبط الاتصال مع المتحكم

أما قائمة Window فتختص بطريقة عرض النوافذ وترتيبها وتسجل النوافذ المفتوحة للانتقال بينها بشكل مباشر

أما قائمة Help فتختص بمعلومات عن البرنامج وإصداره الحالي بالإضافة إلى قائمة بالمساعدة في استخدام البرنامج وتعليمات البرمجة.

56

وسوف نستعرض معا في شكل سريع الشاشات الأخرى التي يمكن فتحها وملخص سريع لما يمكن أن نجده فيها ووظيفتها مع العلم بأن نفس القوائم التي استعرضناها تقوم بنفس العمل ولكن على مستوى العنصر الفرعي الذي تعمل عليه الصفحة.



شاشة ضبط المكونات .Hardware Configuration (HW Config)

وفيها يتم عرض وتحرير وضبط خصائص مكونات منظومة المتحكم وتتكون الشاشة من القائمة الرئيسية في أعلاها وشريط الأدوات لتعرض الأدوات الأكثر استخداما ثم الجزء الأيسر والذي ينقسم إلى جزئين الأعلى يتم فيه اختيار المكونات والأسفل يظهر خصاص الوحدة الأساسية ووصفها ورقم الأمر رقم Order number وعناوين الاستخدام داخل البرنامج أما الجزء الأيمن فيظهر فيه قائمة بجميع منتجات شركة سيمنس وغيرها والتي يمكن استخدامها في البرنامج حيث يتم الاختيار من بينها ويتم سحب المكون إلى المكان المخصص له وأسفل هذه القائمة توجد نافذة عرض لتعرض تفاصيل العنصر المختار حيث تقدم وصفا مختصرا له وخصائصه

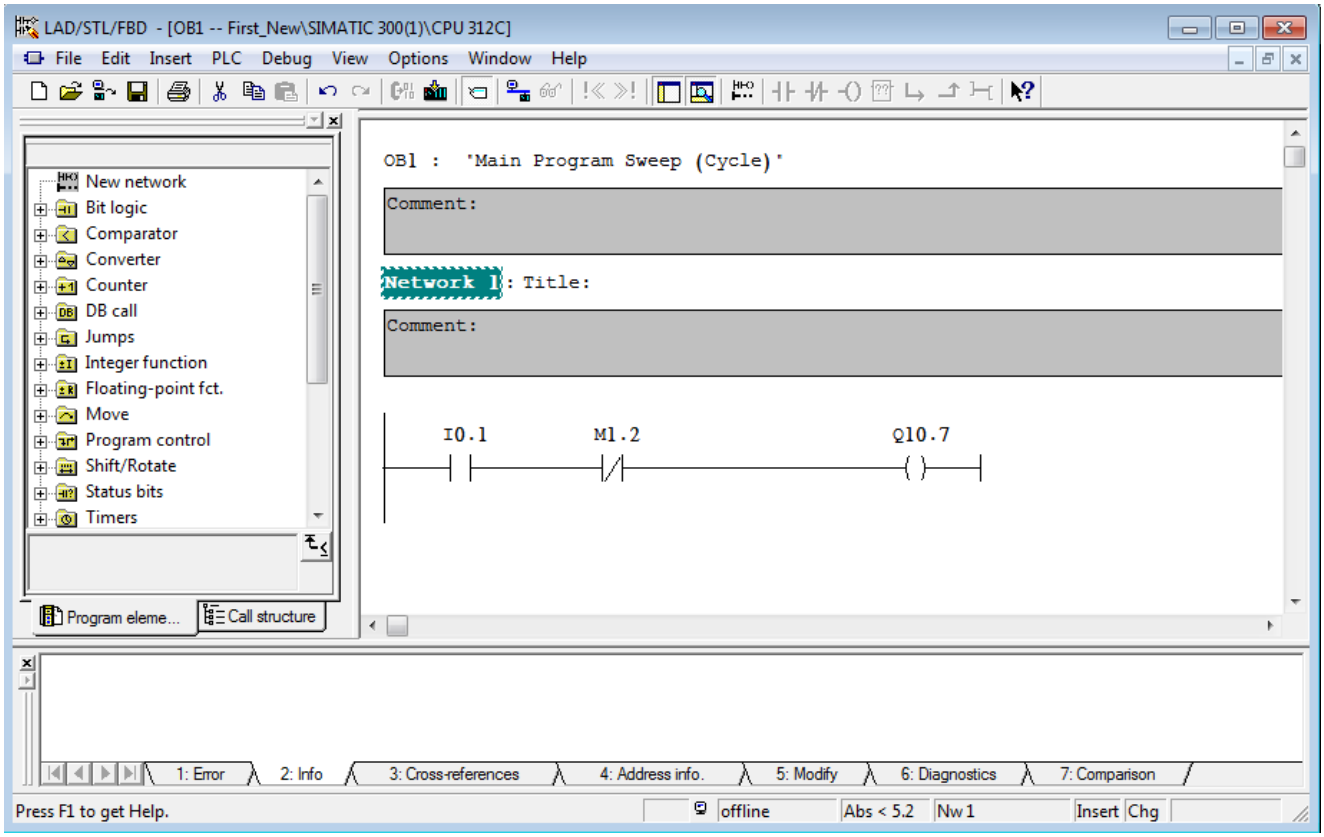
وتحتوي القائمة على عمليات الملفات تحت القائمة Station حيث أن هذه الشاشة تتعامل مع Stations فيمكن إنشاء جديد أو فتح محطة موجودة أو غلق محطة مفتوحة أو حفظها أو حفظها بإسم أو حذفها أو استيراد نسق أو تصديره إلى غير ذلك من عمليات الملفات ولكن على مستوى المحطة Station

والقائمة Edit للتحرير بوظائفه المختلفة وكذلك إدراج Insert والقائمة PLC والقائمة View والقائمة Options والقائمة Window والمساعدة فكلها تقوم بنفس المهام التي سبق أن تحدثنا عنها ولكن على مستوى المحطة Station وعناصرها الفرعية.

كما يمكن تحديث محتويات الكتالوج بآخر إصدارات سيمنس عن طريق قائمة Options ثم Update Catalog أو إضافة أي مكون غير منتجات سيمنس إلى الشبكة أو إلى المكونات باستخدام خيار Options أيضا بإضافة ما يسمى GSD File حيث أن هذا النوع من الملفات هو ما يلزم ليتم إضافة أي عنصر (غير سيمنس) ليعمل مع سيمنس ويتم توريده مع الجزء أو يتم البحث عنه على شبكة الانترنت للحصول عليه ويمثل تماما Driver أو مفهومه بالنسبة للحاسب.

57

محرر البرنامج LAD/STL/FBD Editor



يستخدم محرر البرنامج LAD/STL/FBD Editor في كتابة التعليمات في القوالب المنطقية Logical Blocks ومحتويات قوالب البيانات Data Blocks وكما بالشكل السابق فإن الصفحة تحتوي على سطر القوائم والذي يحتوي أيضا على جزء خاص بوظائف الملفات File لكنه في هذه الحالة ينطبق على البلوك ويشمل أيضا إنشاء جديد وفتح بلوك موجود وحفظ وحفظ بإسم وحذف إلى غير ذلك من عمليات الملفات

ثم قائمة Edit والتي تشمل عمليات تحرير البلوك سواء نسخ أو لصق أو قطع إلى غير ذلك من وظائف التحرير

ثم قائمة Insert والتي تمثل هنا إدراج عمليات وتعليمات البرمجة داخل البلوك

ثم قائمة PLC والتي تتضمن عمليات حال الاتصال بالمتحكم مثل النقل من وإلى والتحكم في وضع تشغيل المتحكم ومراقبة أداء الوحدة

ثم قائمة Debug والتي تشمل عمليات لتصحيح البرنامج وتشغيل البرنامج على خطوات

ثم قائمة View والتي ترتبط بعرض مكونات الشاشة وأشرطة الأدوات

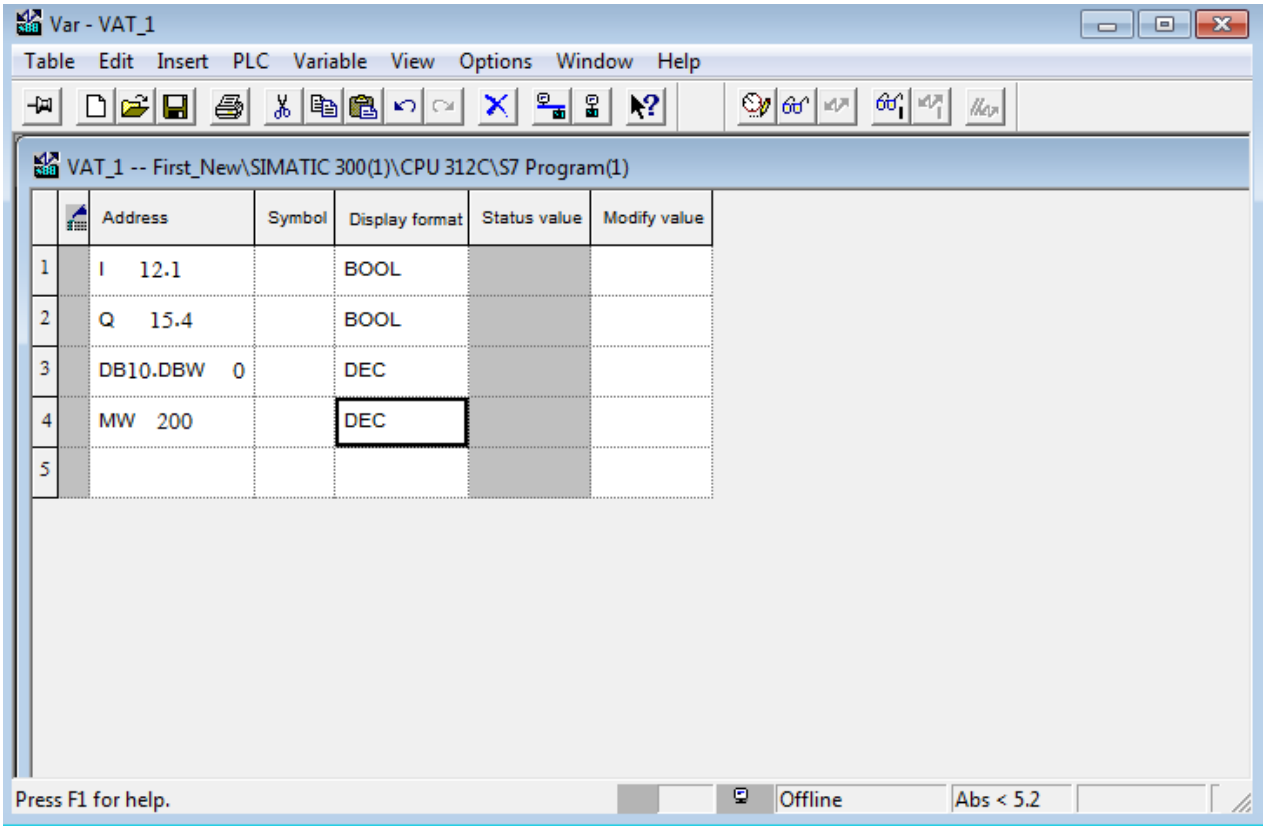
ثم قائمة Options والتي تختص بضبط المحرر Editor وعمليات الاتصال

ثم قائمة Windows لعرض النوافذ والانتقال بينها وأخيرا قائمة Help للمساعدة

وإذا نظرنا إلى الشاشة فسنجد فيها الجزء الرئيسي حيث يتم كتابة البرامج والتعليمات فينقسم إلى شبكات Networks كل واحدة منها يمكن أن نكتب فيها جزء من البرنامج في حين أن الجانب الأيسر يمثل قائمة بالتعليمات والمكتبة والقوالب المستخدمة والتي يمكن استخدامها في عمليات البرمجة أما الجزء الأسفل فيحمل رسائل النظام للمبرمج

58

جدول المتغيرات Variable Table (VAT):



يتم إنشاء جدول المتغيرات في الشاشة الرئيسية لبرنامج SIMATIC Manager من خلال قائمة Insert ثم اختيار Variable Table حيث يمكن اختيار مجموعة من العناصر لمراقبتها أثناء التشغيل والتعرف على حالتها الفعلية بل وتمرير قيم إلى متغيرات داخل البرنامج حال التشغيل ويستخدمه الكثيرون لاختبار البرامج وتتبع الأعطال والتعرف على قيم المتغيرات OnLine مع المتحكم ويتكون كما هو واضح من جدول به صفوف تمثل المتغيرات المطلوب قراءة حالتها أو تمرير قيمها إلى المتحكم وأعمدة من اليسار تمثل رقم مسلسل ثم عنوان المتغير ثم الرمز داخل البرنامج ثم نوع البيانات أو طريقة العرض للبيانات ثم القيمة الحالية ثم القيمة المطلوب تمريرها للمتحكم عند الرغبة في ذلك

وكما هو الحال في كل الشاشات فإنه توجد عمليات ملفات لكن في هذه الحالة ينطبق كلامنا على الجدول وسوف تجد قائمة الملفات بإسم Table وتشمل عمليات الملفات File التي تحدثنا عنها من قبل ولكن تتحدث هنا عن Table الجدول

ثم قائمة التحرير Edit والتي تشمل عمليات النسخ والقص واللصق والحذف إلى غير ذلك من عمليات التحرير وهنا تكون على مستوى عناصر الجدول في شكل صفوف

ثم قائمة Insert والتي تشمل هنا إدراج صف أو صفوف أو عناصر محددة من عناصر الجدول

ثم قائمة PLC والتي تشمل عمليات الاتصال مع المتحكم

أما قائمة Variable فهي لبدء القراءة من المتحكم أو لنقل البيانات إليه في شكل المتغيرات

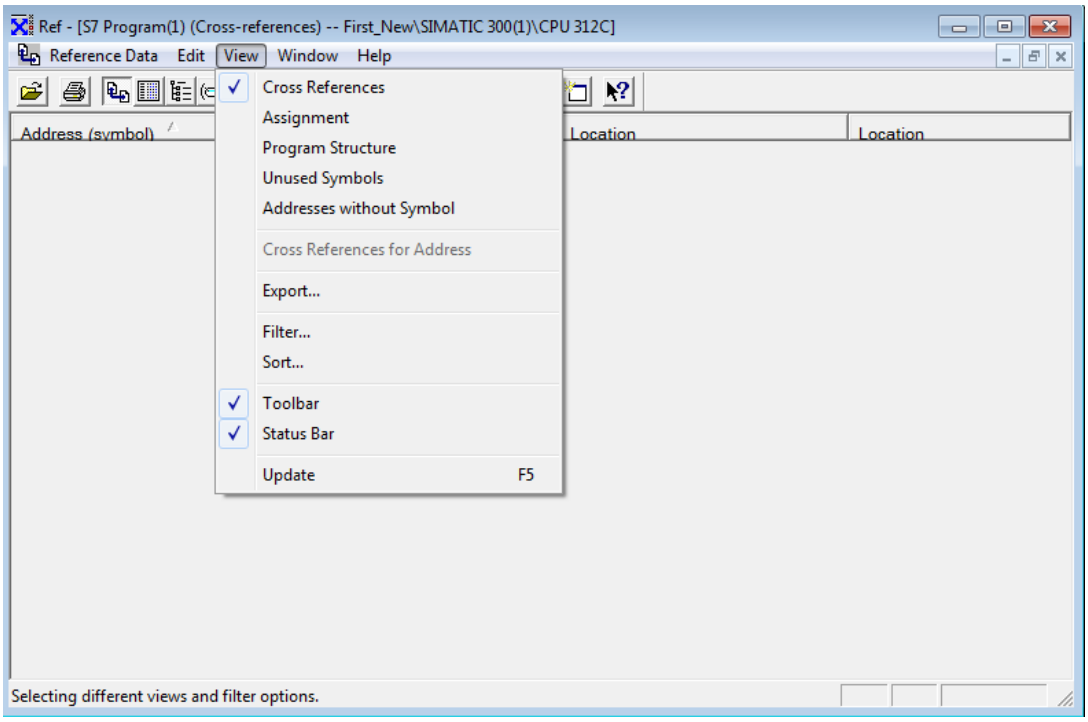
وتختص قائمة View باختيار طرق العرض للبرنامج وشريط الأدوات

وقائمة Options فهي لخصائص الجدول وقائمة Windows لتنظيم عرض النوافذ والانتقال بينها وقائمة Help للمساعدة

59

وتمثل الأدوات في شريط الأدوات وصول سريع للوظائف بديلا عن القوائم حيث يظهر في شريط الأدوات كثير الاستخدام من الأدوات.

فهرس البرنامج Cross Reference:

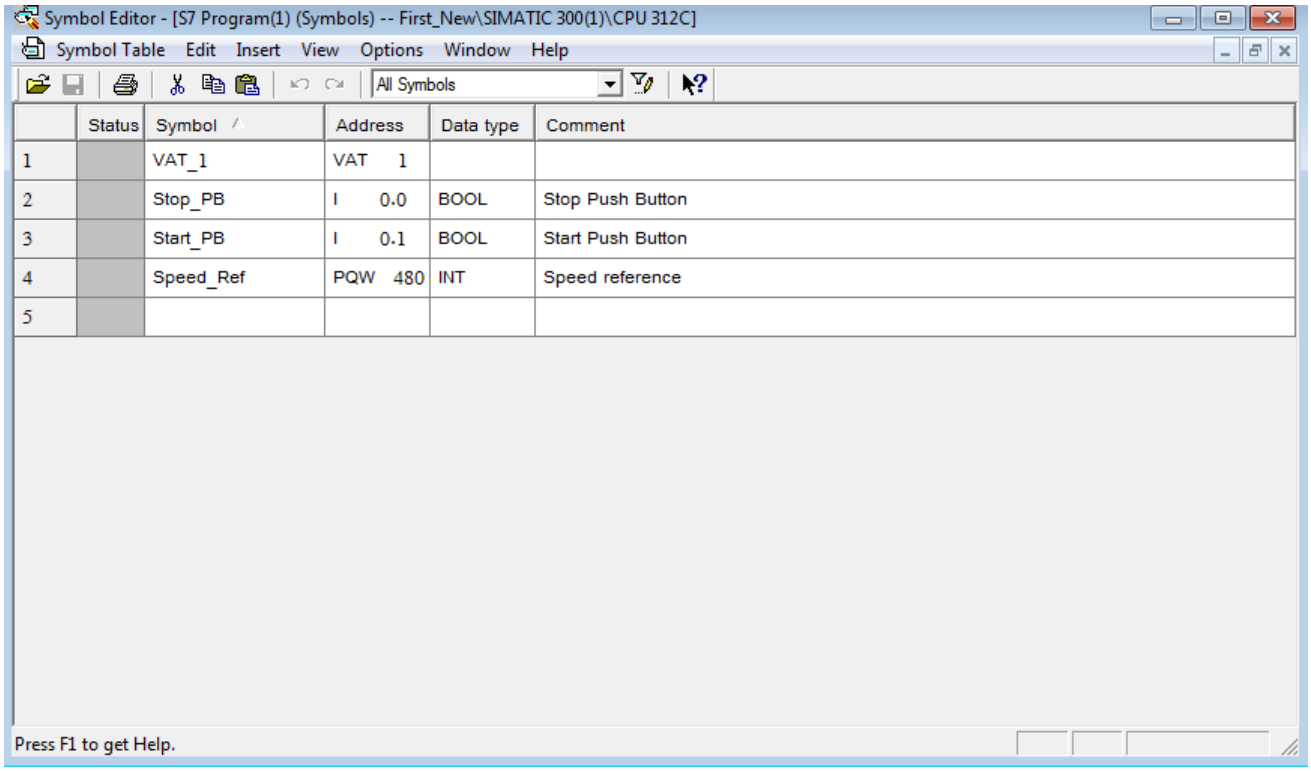


هذا الفهرس أو تلك الإمكانية Cross Reference في منتهى الأهمية للمبرمج حيث يقوم البرنامج بفهرسة البيانات التي تستخدمها خاصة القوالب والذواكر المختلفة في شكل قاعدة بيانات الصفوف تمثل العناصر والأعمدة تمثل أماكن استخدامها وكيفية استخدامها في البرنامج وهذه ليست الإمكانية الوحيدة بل هناك أيضا Assignment أو التخصيص بما تم تخصيصه في البرنامج وكيفية التخصيص

كذلك تشتمل على عرض تركيب البرنامج Program Structure وتسلسل التنفيذ فيه ويشمل أيضا الرموز غير المستخدمة Unused Symbols حيث يمكن أن يكتب المبرمج رموزا لذواكر معينة ولم يستخدمها فيساعده البرنامج في الوصول لها كذلك يعرض الذواكر المستخدمة والتي لم يكتب لها رموز Addresses without symbols

وأهم شئ في هذه الشاشة هو اختيار طريقة العرض من خلال View كذلك الفلتر Filter الذي من خلاله ستختار عناصر العرض

جدول الرموز Symbol Table:



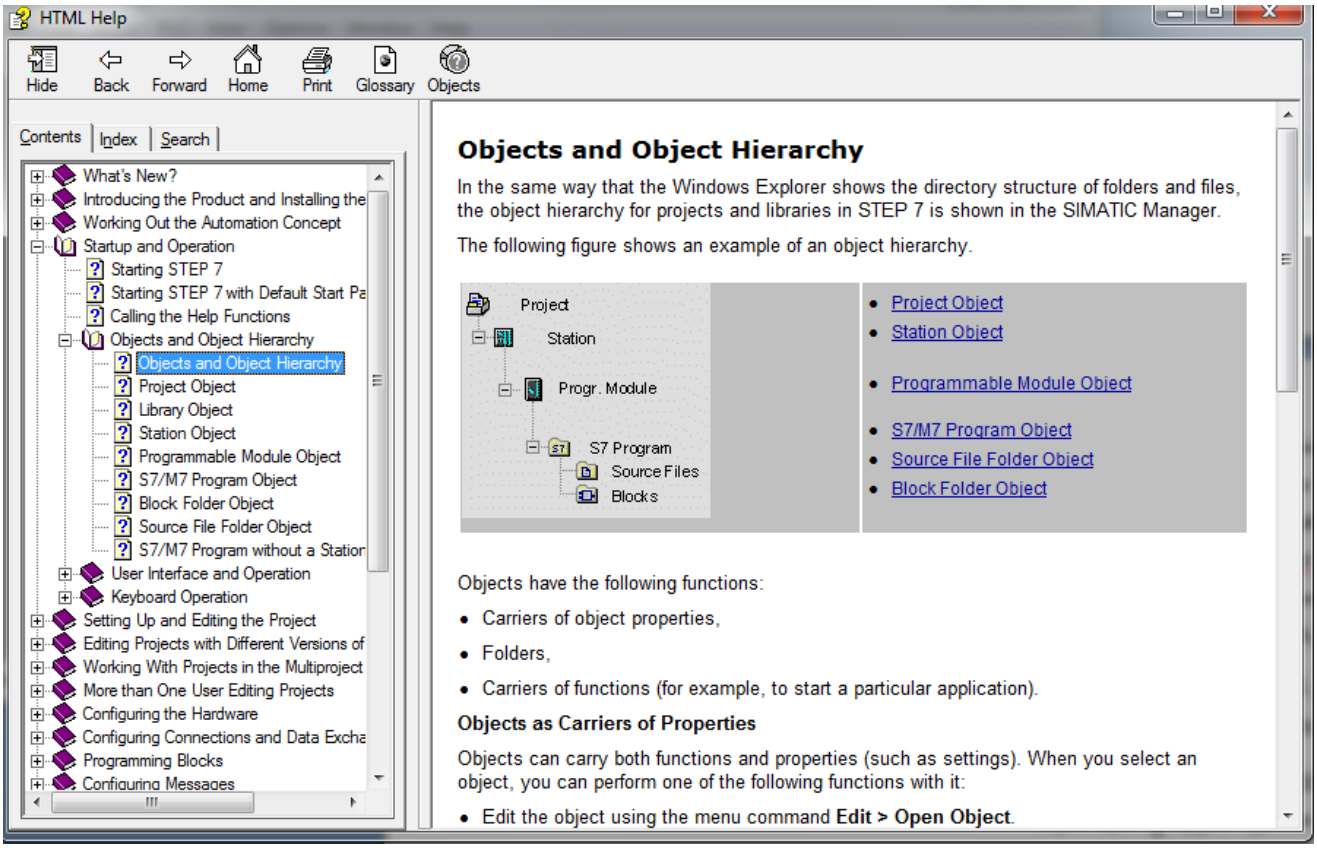
60

يستخدم جدول الرموز لكتابة رموز للعناوين والذواكر المستخدمة داخل البرنامج حيث يمكن استخدام تلك العناصر واستدعاؤها بعناوينها المطلقة أو باستخدام الرموز Symbols وكما بالشكل السابق فإنها تكون في شكل جدول نصل إليه من داخل محرر LAD/STL/FBD Editor من قائمة Options ثم اختيار Symbol Table وكل سطر من سطور الجدول يمثل عنصرا والأعمدة تمثل الرمز Symbol ثم العنوان Address ثم نوع البيانات Type وأخيرا تعليق Comment ويمكن اختيار أن تظهر تلك الرموز والتعليقات مع العنوان داخل البرنامج أو تظهر العناوين فقط

وتتيح هذه الشاشة أيضا عمليات الملفات في شكل جدول رموز Symbol Table ثم عمليات التحرير Edit والتي تسهل عمليات الكتابة كثيرا ثم عمليات الإدراج Insert لإدراج رموز أو خلايا بيانات جديدة

ثم قائمة العرض View والخيارات Options والنوافذ Windows والمساعدة Help بنفس المفاهيم السابقة ولكن على مستوى جدول الرموز.

المساعدة Help:



يمنح برنامج SIMATIC Manager ملف مساعدة في منتهى القوة عن كيفية استخدام البرنامج وكذلك تعليمات البرمجة ويمكن أن تصل إليه من خلال قائمة Help وسوف يساعدك كثيرا في فهم كل شئ في البرنامج والبرمجة والتعامل مع منظومة Step-7 بشكل عام إلى جانب مجموعة الوثائق المرفقة بالبرنامج والتي يتم تثبيتها على الجهاز مع البرنامج وسوف تجدها في القائمة الرئيسية لنظام ويندوز تحت حافظة SIMATIC تحت اسم Documents بخمس لغات مختلفة للأسف ليس بينها اللغة العربية.

61

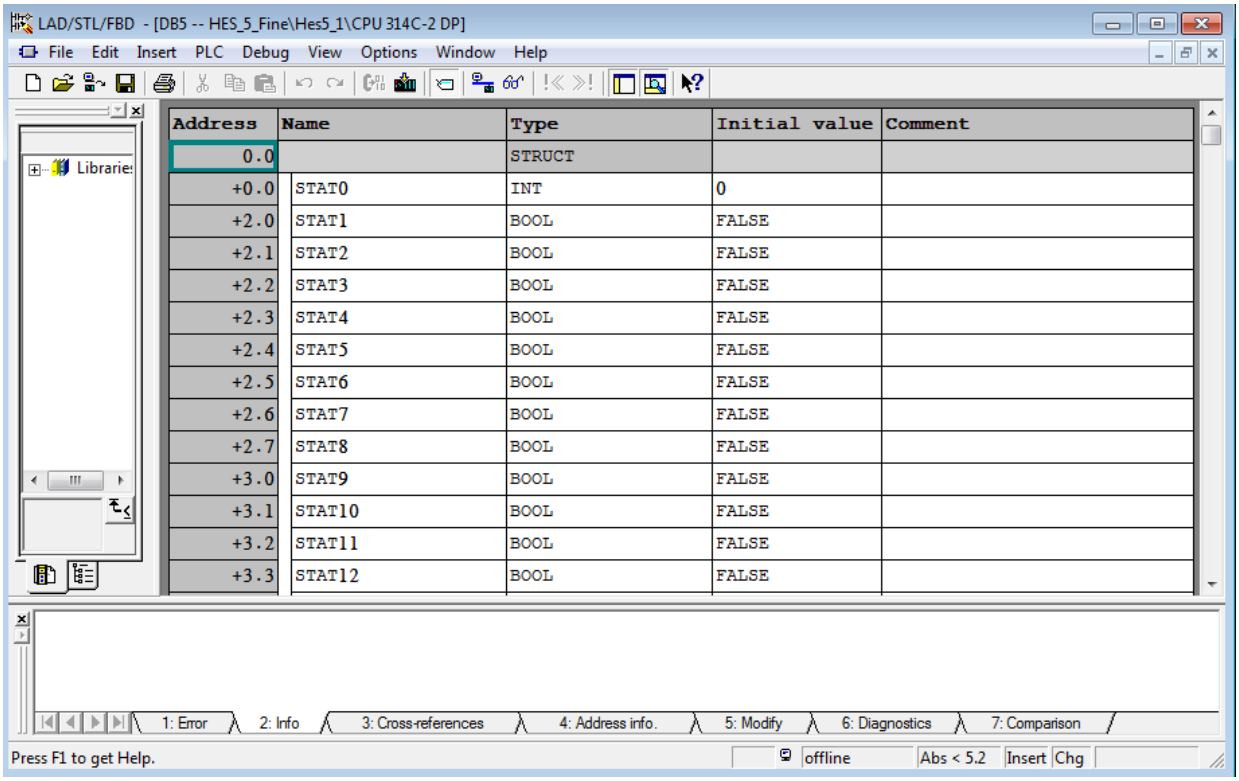
S7 Blocks, Data Types and Addressing أنواع البلوكات والبيانات والعناوين

أنواع البلوكات Block Types:

هناك نوعان رئيسيان للقوالب Blocks في منظومة المتحكمات S7-300/400 وهما :

- قوالب بيانات Data Blocks وتحتوي على بيانات فقط ومنها نوعان فرعيان هما:
  - قوالب بيانات Data Blocks (DB)
  - أنواع بيانات مستخدم User Data Types (UDT)

والاثنان لهما نفس الشكل ونفس طريقة التصنيف حيث يتكون البلوك من قاعدة بيانات صفوفها البيان وأعمدتها عبارة عن العنوان Address والاسم Name ونوع البيانات Type والقيمة المبدئية Initial value وتعليق Comment



- قوالب منطقية Logical Blocks وتحتوي على تعليمات وبيانات حيث يتم كتابة البرنامج فيها أو باستخدامها وتتكون من حلقات أو شبكات تسمى Networks وهي وحدة بناء البلوك الأصلي ويوجد بها جزء خاص بتنظيم الذواكر المحلية يسمى Declaration في أعلى البلوك عند الكتابة ومنها أنواع على النحو التالي:
  - قوالب تنظيمية Organization Blocks (OB) وهي الطريق الرئيسي لتنفيذ كل تعليمات البرنامج ولا يتم تنفيذ أي شئ إلا من خلالها بشكل مباشر أو غير مباشر ومنها OB1 وهو المختص بالدورة الرئيسية للبرنامج حيث يتم تنفيذ ما فيه ثم يتم التكرار مرة أخرى بمجرد الانتهاء منه ويتم كتابة البرنامج الرئيسي من خلاله ومنها أيضا OB100 والذي يتم تنفيذه مرة واحدة في بداية التشغيل ومنها OB35 والذي يتم تنفيذه على فترات زمنية ثابتة بعد قطع التسلسل العادي للتشغيل ومنا ما يتم بناء على حدث معين مثل OB86 و OB87 ويمكن أن يتغير عدد بلوكات التنظيم تبعا لنوع وحدة المعالجة ويتم التعرف على الموجود منها من جدول الخصائص الخاص بوحدة المعالجة.
  - قوالب وظيفية Functions (FC) وهي التي يتم فيها كتابة تعليمات المستخدم وبيانات التشغيل بشكل مباشر ويتم الترقيم بداية من FC1 وحتى الحد الأقصى المسموح به لوحدة المعالجة.

- قوالب وظيفية Function Blocks (FB) ويتم فيها أيضا كتابة تعليمات المستخدم بشكل مباشر ويتم الترقيم بداية من FB1 وحتى الحد الأقصى المسموح به لوحدة المعالجة ويجب أن يصاحبها بلوك بيانات Data Block في عملها يطلق عليه Instant Data Block يحتوي على بيانات تخص البلوك وتشغيله ويتم إنشاء بلوك البيانات بمعرفة البلوك FB في الجزء الخاص ب Declaration في أعلى البلوك
- قوالب وظيفية نظامية System Functions (SFC) وهي قوالب أو بلوكات تم إعدادها من شركة سيمنس لتقوم بعمليات محددة مسبقا ويتم استخدامها فقط بعد استدعائها من المكتبة Library ولا يمكن التعديل فيها
- قوالب وظيفية نظامية System Function Blocks (SFB) وهي قوالب أو بلوكات تم إعدادها من شركة سيمنس لتقوم بعمليات محددة مسبقا ويتم استخدامها فقط بعد استدعائها من المكتبة Library ولا يمكن التعديل فيها وتختلف عن SFC في أنها لابد أن يكون معا Instant Data Block أي بلوك بيانات يحتوي على بيانات التشغيل الخاصة بها

هذه القوالب بشقيها المنطقية والبيانات هي التي يتكون من برنامج البرمجة الذي نقوم بتحميله على وحدة المعالجة لتنفيذ مهام التحكم التي نريدها من المنظومة.

أنواع البيانات الأولية Elementary Data Types:

معظم العمليات التي يقوم بها المتحكم تتم على معاملات أو بيانات فالعمليات عموما تتكون من شقين عملية Operation ومعامل Operand والعمليات تتم وفق تقسيم معين يعتمد على نفس نوع المعاملات التي تتم عليها وبالتالي وجب علينا التعرف على أنواع البيانات التي تتم عليها البيانات ومنها أنواع أولية أو أساسية وأنواع مركبة ونبدأ بالأنواع الأولية

63

التقسيم حسب حجم البيانات

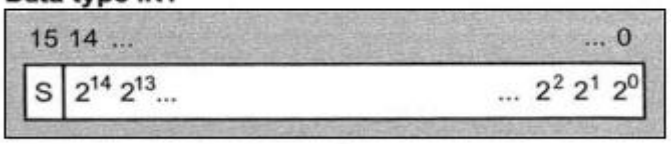
حجم البيانات الأساسية لها أنواع قياسية كالآتي:

- بيانات منطقية ثنائية Boolean وتأخذ الرمز BOOL وحجمها خانة واحدة Bit وتخضع لها جميع البيانات المنطقية والتي تحتمل الحالة "1" أو الحالة "0"
- بيانات في حجم 8 خانات Byte وتأخذ الرمز BYTE وتخضع لها البيانات التي حجمها 8 خانات 8-Bits أو Byte
- بيانات في حجم 16 خانة أو كلمة Word Or 16-Bit وتأخذ الرمز WORD وتخضع لها البيانات التي حجمها 16 خانة أو 2-Byte
- بيانات في حجم 32 خانة أو 2 كلمة Double Word Or 2 Word Or 32-Bits وتأخذ الرمز DWORD وتخضع لها البيانات التي حجمها 32 خانة أو 2 كلمة

أما من ناحية نوع البيانات المخزنة فهناك تقسيم آخر للبيانات الأولية على النحو التالي:

- بيانات منطقية ثنائية Boolean وتأخذ الرمز BOOL وحجمها خانة واحدة 1-Bit أو Byte أو Word أو DWord ويتم التعامل معها على مستوى Bit
- بيانات صحيحة بدقة 16 خانة 16-Bits Integers وتأخذ الرمز INT حيث يتم تمثيل البيانات في شكل رقم صحيح حيث تحتوي الخانات من 0 وحتى 14 على قيمة الرقم وتحتوي الخانة رقم 15 على الإشارة وبالتالي تتراوح قيمة الرقم بين 32768- و 32767+

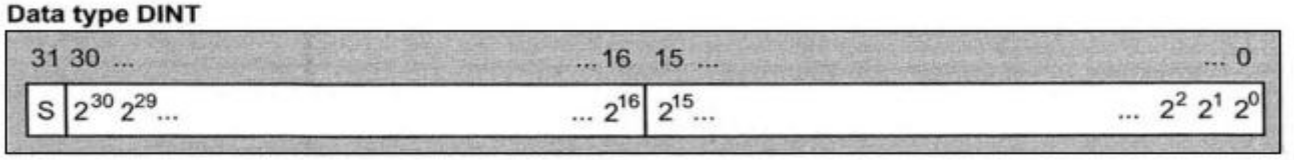
Data type INT



S Sign

- بيانات صحيحة دقة مضاعفة 32 خانة 32-Bit Integers أو Double-Integers وتأخذ الرمز DINT ويتم بها تمثيل البيانات في شكل رقم صحيح حيث تحتوي الخانات من 0 وحتى 30 على قيمة الرقم وتحتوي الخانة 31 على الإشارة وتتراوح قيمة الرقم بين 2147483648- وبين 2147483648+

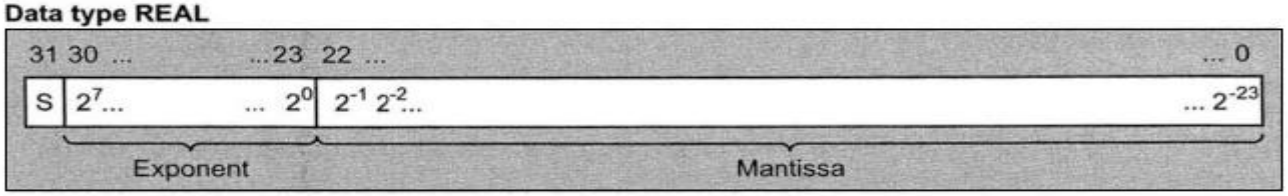
Data type DINT



S Sign

- بيانات حقيقية Real or Floating point بدقة 32 خانة حيث يتم تمثيل الأرقام العشرية في 32 خانة وتأخذ الرمز REAL وتخضع للعمليات على الأرقام العشرية Floating point mathematics وهنا تكون الدقة أعلى في الحسابات سواء على المستوى الصحيح أو الكسور.

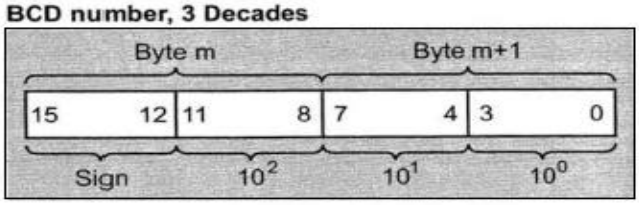
Data type REAL



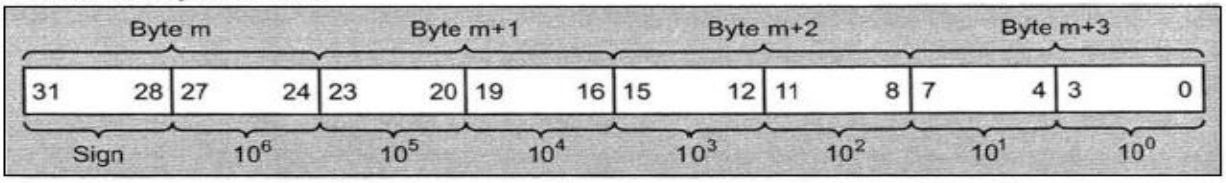
S Sign

- بيانات صحيحة بصيغة Binary Coded Decimal (BCD) حجم 16 خانة حيث تمثل قيمة الرقم بأربعة رموز كل واحد منها من 0 إلى 9 وتبدأ من 0000 وتنتهي ب 9999 وفي حالة 32 خانة تبدأ ب 0000_000± وتنتهي ب 9999_999±

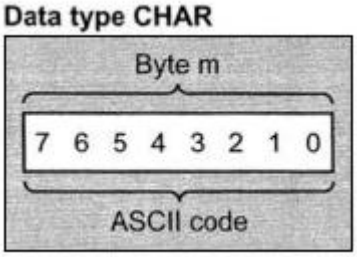
BCD number, 3 Decades



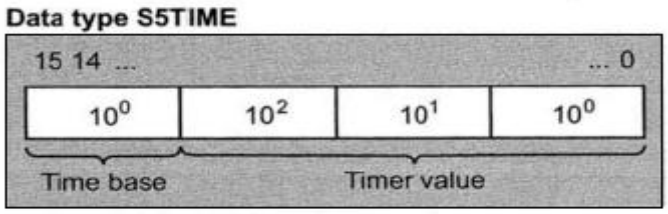
BCD numbers, 7 Decades



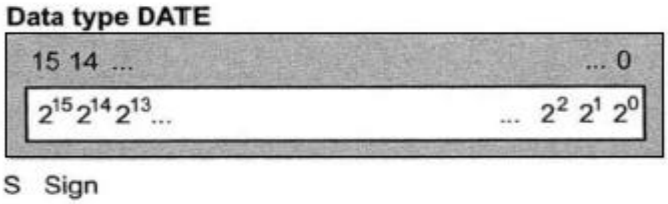
- بيانات رمز Character وتأخذ حجم 8 خانة وتأخذ الرمز CHAR حيث يتم تمثيل الرموز بها في شكل ASCII Code لتمثيل الرموز المعروفة كلها



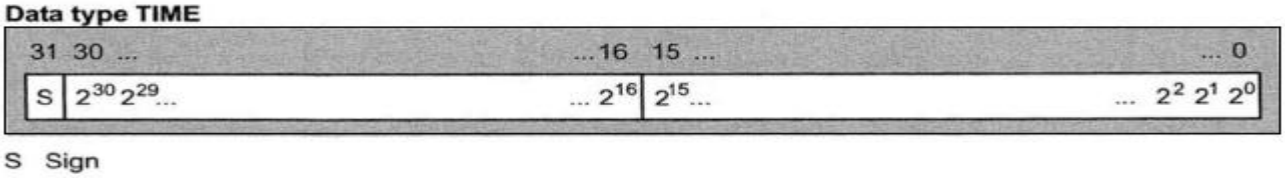
- بيانات وقت مؤقت S5Time ويمثل الوقت الذي تتعامل عليه المؤقتات ويأخذ الرمز S5T# أو S5TIME# حيث يتم كتابة القيمة الوقتية بعد الرمز وتأخذ الصيغة S5T#xhymzsums حيث h تشير إلى الساعات والقيمة x تشير إلى قيمة الساعات الرمز m تشير إلى الدقائق والقيمة y تشير إلى قيمة الدقائق والرمز s يشير إلى الثواني والقيمة z تشير إلى قيمة الثواني والرمز ms يشير إلى مللي ثانية والقيمة u تشير إلى قيمة أجزاء المللي ثانية ويتم تمثيل قيمة تشغيل المؤقت في حجم 16 خانة حيث تمثل الخانات من 0 وحتى 11 قيمة الوقت بينما تمثل الخانات 12,13 قاعدة الوقت والخانات 14,15 غير مستخدمة وبالتالي فأقصى قيمة للوقت هي 9990 ثانية أو 2:46:30.0 ساعتان و46 دقيقة و30 ثانية.



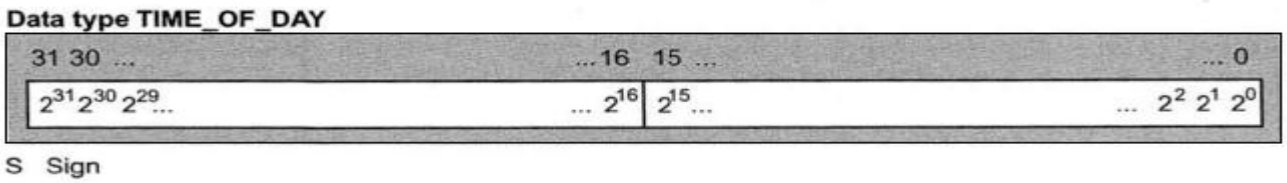
- التاريخ Date ويتم تمثيل التاريخ بحجم 16 خانة في شكل صيغة أربعة أرقام للسنة ورقمان للشهر ورقمان لليوم yyyy-mm-dd حيث yyyy تمثل السنة و mm تمثل الشهر و dd تمثل اليوم وتأخذ الرمز D# أو DATE# متبوعا بقيمة التاريخ. 65



- قيمة الوقت Time ويمثل قيمة الوقت متضمنا الأيام والساعات والدقائق والثواني وأجزاء الثانية ويتم تمثيل ذلك في حجم بيانات 32 خانة ويستخدم الرمز TIME# أو T# متبوعا بقيمة الوقت مثلا T#10d2h55m30s100ms



- قيمة الوقت الآني Time of Day حيث تعبر عن قيمة الساعة الآن وتأخذ الصيغة TOD# أو TIME_OF_DAY# بحجم بيانات 32 خانة مثلا TOD#23:44:30.330



وهناك أنواع من البيانات المركبة والتي تتكون من أكثر من نوع من البيانات الأولية ومن هذه الأنواع:

- المصفوفات Arrays حيث تمثل مجموعة البيانات المتشابهة تماما في النوع بمصفوفة من العناصر
- التركيبات Structures والتي تمثل تركيبة من عناصر مختلفة في النوع معا تحت نفس المتغير
- سلسلة رموز String حيث تمثل مجموعة من الرموز Characters من النوع CHAR
- الوقت والتاريخ DATE_AND_TIME حيث تمثل مزجا للوقت والتاريخ بحجميهما معا في متغير واحد

العنونة المباشرة Direct Addressing:

كل عنصر من عناصر الشبكة أو المكونات أو ذاكرة المتحكم له عنصر وحيد لا يتكرر في البرنامج ولكل نوع من أنواع الذواكر حد أقصى للعناوين التي يمكن استخدامها وسوف نستعرض معا أنواع العناوين على مستوى المكونات والذواكر:

- **المدخلات الرقمية Digital Inputs**

تستخدم المدخلات الرقمية لتوصيل إشارات الإدخال الرقمية إلى منظومة المتحكم وتأخذ المدخلات الرقمية الرمز "I" ويمكن أن يكون حجمها خانة واحدة 1-bit أو 8 خانات Byte أو 16 خانة Word أو 32 خانة Double-Word ويكون التعبير كالآتي:

I x.y حيث x تمثل عنوان Byte و y تمثل عنوان bit والذي يترواح بين 0 ، 7 مثلا I 10.7 ، I 2.5
IB x حيث x تمثل عنوان Byte ويعبر عن عدد 8 خانات ، مثلا IB 23 ، IB 0 ، IB 12
IW x حيث x تمثل عنوان أول Byte فمثلا IW 10 تعبر عن IB 10 + IB 11 بحجم 16 خانة
ID x حيث x تمثل عنوان أول Byte فمثلا ID 20 تعبر عن IB 20 + IB 21 + ID 22 + ID 23 بحجم 32 خانة

- **المخرجات الرقمية Digital Outputs**

تستخدم المخرجات الرقمية لإخراج نتيجة العمليات داخل المتحكم في شكل إشارة كهربية وتأخذ المدخلات الرقمية الرمز "Q" ويمكن أن يكون حجمها خانة واحدة 1-bit أو 8 خانات Byte أو 16 خانة Word أو 32 خانة Double-Word ويكون التعبير كالآتي:

Q x.y حيث x تمثل عنوان Byte و y تمثل عنوان bit والذي يترواح بين 0 ، 7 مثلا Q 13.6 ، Q 6.7
QB x حيث x تمثل عنوان Byte ويعبر عن عدد 8 خانات ، مثلا QB 23 ، QB 2 ، QB 20
QW x حيث x تمثل عنوان أول Byte فمثلا QW 12 تعبر عن QB 12 + QB 13 بحجم 16 خانة
QD x حيث x تمثل عنوان أول Byte فمثلا QD 10 تعبر عن QB 10 + QB 11 + QD 12 + QD 13 بحجم 32 خانة

- **الذاكرة العامة Global Memory**

تستخدم الذاكرة العامة في تخزين نتيجة عمليات وسيطة على مدار البرنامج حيث تحتفظ بحالتها في كل بلوكات البرنامج وتأخذ الذاكرة العامة الرمز "M" ويمكن أن يكون حجمها خانة واحدة 1-bit أو 8 خانات Byte أو 16 خانة Word أو 32 خانة Double-Word ويكون التعبير كالآتي:

M x.y حيث x تمثل عنوان Byte و y تمثل عنوان bit والذي يترواح بين 0 ، 7 مثلا M 13.6 ، M 6.7
MB x حيث x تمثل عنوان Byte ويعبر عن عدد 8 خانات ، مثلا MB 23 ، MB 2 ، MB 20
MW x حيث x تمثل عنوان أول Byte فمثلا MW 12 تعبر عن MB 12 + MB 13 بحجم 16 خانة
MD x حيث x تمثل عنوان أول Byte فمثلا MD 10 تعبر عن MB 10 + MB 11 + MD 12 + MD 13 بحجم 32 خانة

- **الذاكرة المحلية Local Memory**

تستخدم الذاكرة العامة في تخزين نتيجة عمليات وسيطة في البلوك المستخدمة داخله فقط ولا تحتفظ بحالتها من بلوك لآخر وتأخذ الذاكرة العامة الرمز "L" ويمكن أن يكون حجمها خانة واحدة 1-bit أو 8 خانات Byte أو 16 خانة Word أو 32 خانة Double-Word ويكون التعبير كالآتي:

L x.y حيث x تمثل عنوان Byte و y تمثل عنوان bit والذي يترواح بين 0 ، 7 مثلا L 13.6 ، L 6.7

LB x حيث x تمثل عنوان Byte ويعبر عن عدد 8 خانات ، مثلا LB 23 ، LB 2 ، LB 20

LW x حيث x تمثل عنوان أول Byte فمثلا LW 12 تعبر عن LB 12 + LB 13 بحجم 16 خانة

LD x حيث x تمثل عنوان أول Byte فمثلا LD 10 تعبر عن LB 10 + LB 11 + LD 12 + LD 13 بحجم 32 خانة

- **قوالب البيانات Data Blocks (DB) and User Data Types (UDT)**

تستخدم قوالب البيانات في تخزين نتيجة عمليات وسيطة في البلوك المستخدم وتأخذ قوالب الرمز "DB" متبوعة برقم البلوك ثم تأخذ البيانات الرمز "DB" ويمكن أن يكون حجمها خانة واحدة 1-bit أو 8 خانات Byte أو 16 خانة Word أو 32 خانة Double-Word ويكون التعبير كالآتي:

DBw.DBXx.y حيث w تمثل رقم البلوك و x تمثل عنوان Byte و y تمثل عنوان bit والذي يترواح بين 0 ، 7 مثلا DB10.DBX20.3 و UDT1.DBX10.7

DBw.DBBx حيث x تمثل عنوان Byte ويعبر عن عدد 8 خانات ، مثلا DB2.DBB10

DBw.DBWx حيث x تمثل عنوان أول Byte فمثلا DB40.DBW10 تعبر عن DB40.DBB10 + DB40.DBB11 بحجم 16 خانة

DBw.DBDx حيث x تمثل عنوان أول Byte فمثلا DB60.DBD10 تعبر عن DB40.DBB10 + DB40.DBB11 + DB40.DBB12 + DB40.DBB13 بحجم 32 خانة

- **المدخلات الطرفية Peripheral Inputs**

تستخدم المدخلات الطرفية لتوصيل الإشارات الكهربية التناظرية Analog inputs على مستوى 16 خانة وإشارات التردد على مستوى 32 خانة إلى المتحكم وتأخذ الرمز "PI" متبوعا بحجم البيان ثم عنوان كلمة البداية على النحو التالي:

PIW x حيث x تمثل عنوان أول Byte فمثلا PIW 240 تمثل مدخل تناظري بحجم 16 خانة

PID x حيث x تمثل عنوان أول Byte فمثلا PID 720 تمثل مدخل تردد لعداد مثلا بحجم 32 خانة

- **المخرجات الطرفية Peripheral Outputs**

تستخدم المخرجات الطرفية لتوصيل الإشارات الكهربية التناظرية Analog inputs على مستوى 16 خانة وإشارات التردد على مستوى 32 خانة من المتحكم للعملية وتأخذ الرمز "PQ" متبوعا بحجم البيان ثم عنوان كلمة البداية على النحو التالي:

PQW x حيث x تمثل عنوان أول Byte فمثلا PQW 440 تمثل مخرج تناظري بحجم 16 خانة

PQD x حيث x تمثل عنوان أول Byte فمثلا PQD 760 تمثل مخرج تردد لعداد مثلا بحجم 32 خانة

- **المؤقتات Timers**

تستخدم المؤقتات لتنفيذ العمليات المرتبطة بالوقت وتأخذ المؤقتات الرمز "T" متبوعا برقم المؤقت مثلا T10 أو T25 وهكذا

ويمكن استخدام المؤقت بنفس العنوان كحالة BOOL أو كقيمة للوقت الحالي

- **العدادات Counters**

تستخدم العدادات لتنفيذ عمليات العد بأشكالها وتأخذ الرمز "C" متبوعا برقم العداد مثلا C2 و C36 وهكذا ويمكن استخدام العداد بنفس العنوان كحالة BOOL أو كقيمة للعد الحالي

العنونة غير المباشرة Indirect addressing:

العنونة المباشرة تعطي المبرمج معرفة تامة بالعناوين قبل بدء العمل أو التشغيل للبرنامج بينما العنونة الغير مباشرة لا يعرف فيها العنوان إلا أثناء التنفيذ

حيث يتم إعطاء عنوان متغير أو يخضع للتغير والحسابات من البرنامج فمثلا لو قلنا [IW [MW 200 فهذا يعني أن العنوان المخصص للإدخال هو محتويات الذاكرة MW 200 وبالتالي بتغيير محتويات MW 200 يمكن تعيين عنوان متغير لا يتم معرفته إلا عند التنفيذ للبرنامج

كما يمكن استخدام Address register (AR) وهو مسجل العناوين لتنفيذ عمليات عنونة غير مباشرة

وللعنونة غير المباشرة تفاصيل أكثر تدخل ضمن نطاق البرمجة المتقدمة سوف يأتي تفصيل لها إن شاء الله

اختيار وضبط خصائص المكونات Hardware Configuration

اختيار وضبط خصائص المكونات جزء من البرنامج بل إن هناك بعض الأشياء لا يتم عملها إلا من خلال ضبط خصائص المكونات ويتم تحميل ضبط خصائص المكونات إلى وحدة المعالجة كجزء من البرنامج ، ولا يتم كتابة البرنامج في صورة القوالب التي تحدثنا عنها إلا بعد اختيار وحدة المعالجة على الأقل.

وتتيح عملية اختيار وضبط خصائص المكونة مرونة كبيرة لتكوين منظومات متعددة للمتحكم من بين عائلة كبيرة من المكونات تتناسب مع العمليات المختلفة في عالم الصناعة وسوف نستعرض في الأسطر القادمة أشكال الوحدات المختلفة والتي من بينها نختار المنظومة للمتحكم S7-300:

1- وحدات مصدر الجهد Power Supply:

ويتم توصيفها بتيار الحمل سواء كان 2 ، 5 ، 10 أمبير
وتأخذ الرمز PS307
ويتم تمييزها برقم أمر يحتوي على توصيفها مثل
6ES7 307-1KA00-0AA0 و 6ES7 307-1BA00-0AA0

2- وحدات المعالجة المركزية Central Processing Units CPU:

ويتم توصيفها برقمها في العائلة وعدد منافذ الاتصال ونقاط التوصيل إن وجدت والوظائف الخاصة من العدادات ونقاط التوصيل التناظرية
وتأخذ الرمز CPU31x حيث x يمثل رقمها في العائلة ويضاف لذلك المواصفات الخاصة إن وجدت مثل CPU314C-2DP أو CPU312
ويتم تمييزها برقم أمر يتم الطلب به من سيمنس مثل
6ES7 314-6CG03-0AB0 و 6ES7 312-1AE14-0AB0

3- وحدات الإدخال الرقمية Digital input modules:

ويتم توصيفها بعدد نقاط الإدخال والجهد الكهربي لإشارة الإدخال
فمنها 4 و 8 و 16 و 32 و 64 نقطة
ومنها 24 VDC و 48 VDC و 120/230 VDC
وتأخذ الرمز SM321 ، فمثلا SM321 DI32xDC24V ،
و SM321 DI64xDC24V و SM321 DI16xAC120/230V
ويتم تمييزها برقم أمر Order number يتم الطلب به من سيمنس مثل
6ES7 321-1BL00-0AA0 و 6ES7 321-1EL00-0AA0

4- وحدات الإخراج الرقمية Digital output modules:

ويتم توصيفها بعدد نقاط الإخراج وشكل الإخراج سواء كان جهد كهربي أو ريلاي فمنها
4 و 8 و16 و 32 و 64 نقطة
ومنها 24VDC ومنها 48 VDC ومنها 120/230 VAC ومنها ريلاي
وتأخذ الرمز SM322 ، فمثلا SM322 DO16xDC24V/0.5A و
SM322 DO32Xac120V/1A و SM322 DO8xDC24V/2A
ويتم تمييزها برقم أمر Order number أيضا مثل
6ES7 322-5SD00-0AB0 و 6ES7 322-1EH01-0AA0

5- وحدات الإدخال/الإخراج الرقمية (المختلطة) Digital input/output modules:

ويتم توصيفها بعدد نقاط الإدخال وعدد نقاط الإخراج وشكل إشارة كل منها
وهناك منها 8+8 ، 16+16 نقطة إدخال+إخراج
وتأخذ الرمز SM323 أو SM327 مثلا
SM323 DI8/DO8xDC24V/0.5 وSM323 DI16/DO16xDC24V/0.5A
ويتم تمييز برقم أمر أيضا مثل
6ES7 323-1BL00-0AA0 و 6ES7 327-1BH00-0AB0

6- وحدات الواجهة Interface Modules:

وتستخدم في شكل أزواج يربط بينها كابل اتصال خاص في حالة استخدام أكثر من Rack للمكونات ويرمز لها بالرمز IM36x
ويكون العنصر على الراك المركزي مرسل Send والآخرين مستقبل ومرسل لما يليه Send/Receive
ومنها مثلا IM360 S و IM361 R ويتم تمييزها برقم أمر أيضا
6ES7 360-3AA00-0AA0 و 6ES7 361-3CA00-0AA0

7- وحدات الإدخال التناظرية Analog input modules:

ويتم توصيفها بعدد قنوات الإدخال ودقة الإشارة والتي يرمز لها بعدد الخانات
وهناك منها 2 ، 4 ، 6 ، 8 قنوات إدخال
ومنها فولت وملي أمبير وثرموكبل
وتأخذ الرمز SM331 مثلا SM331 AI2x12Bit و SM331 AI8xRTD
ويتم تمييزها برقم أمر Order number يتم الطلب به من سيمنس مثل:
6ES7 331-7KF01-0AB0 و 6ES7 331-7PF10-0AB0

70

8- وحدات الإخراج التناظرية Analog output modules:

ويتم توصيفها بعدد قنوات الإخراج ودقة الإشارة والتي يرمز لها بعدد الخانات
وهناك منها 2 ، 4 ، 8 قنوات إدخال ومنها فولت وملي أمبير
وتأخذ الرمز SM332 مثلا SM332 AO4x16Bit و SM332 AO8x12Bit
ويتم تمييزها برقم أمر Order number يتم الطلب به من سيمنس مثل:
6ES7 332-5HD01-0AB0 و 6ES7 332-5HF00-0AB0

9- وحدات الإدخال/الإخراج التناظرية Analog input/output modules:

ويتم توصيفها بعدد قنوات الإدخال والإخراج ودقة الإشارة لكل منهما ويرمز لها بعدد الخانات ومنها 4 إدخال/2 إخراج أو 4 إدخال/4 إخراج ومنها فولت وملي أمبير
وتأخذ الرمز SM334 أو SM335 مثلا SM334 AI4/AO2x12Bit و
SM334 AI4/AO4x14/12Bit
ويتم تمييزها برقم أمر Order number يتم الطلب به من سيمنس مثل:
6ES7 334-0CE00-0AA0 و 6ES7 335-7HG01-0AB0

10- وحدات معالج الاتصال Communication processors:

ويتم تمييز هذه الوحدات حسب نوع الاتصال الذي تقوم به فمنها ما يتصل RS232 ومنها ما يتصل PROFIBUS ومنها ETHERNET إلى غير ذلك من وظائف الاتصال
وتأخذ الرمز CP34x مثلا CP342-5 و CP342-2 ASi
ويتم تمييزها برقم أمر Order number يتم الطلب به من سيمنس مثل:
6GK7 342-5DA00-0XE0 و 6GK7 343-1EX00-0XE0

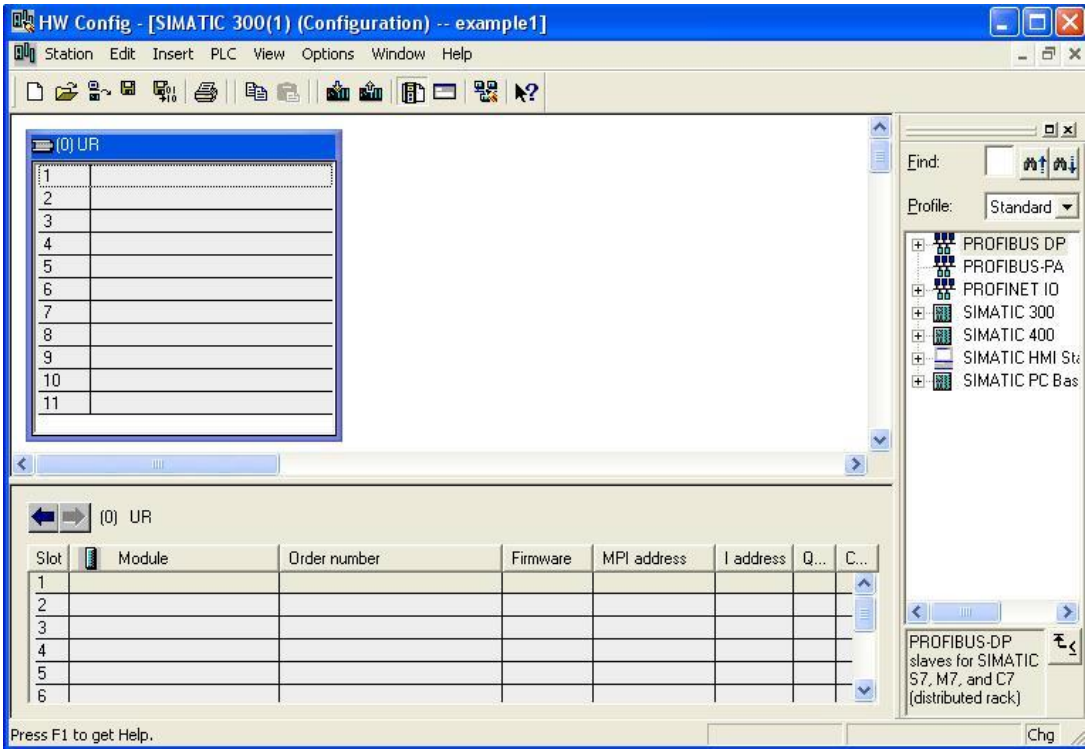
11- الوحدات الوظيفية Function modules:

ويتم تمييز هذه الوحدات حسب الوظيفة التي تقوم بها حيث لكل وظيفة نوع معين منها مثل العدادات ومنظومات التحكم في الوضع وأجهزة التحكم في الحرارة وتأخذ الرمز FM35x مثلا FM350-1 و FM355C PID و FM354 F
ويتم تمييزها برقم أمر Order number يتم الطلب به من سيمنس مثل:
6ES7 355-0VH00-0AE0 و 6ES7 351-1AH01-0AE0

اختيار المكونات

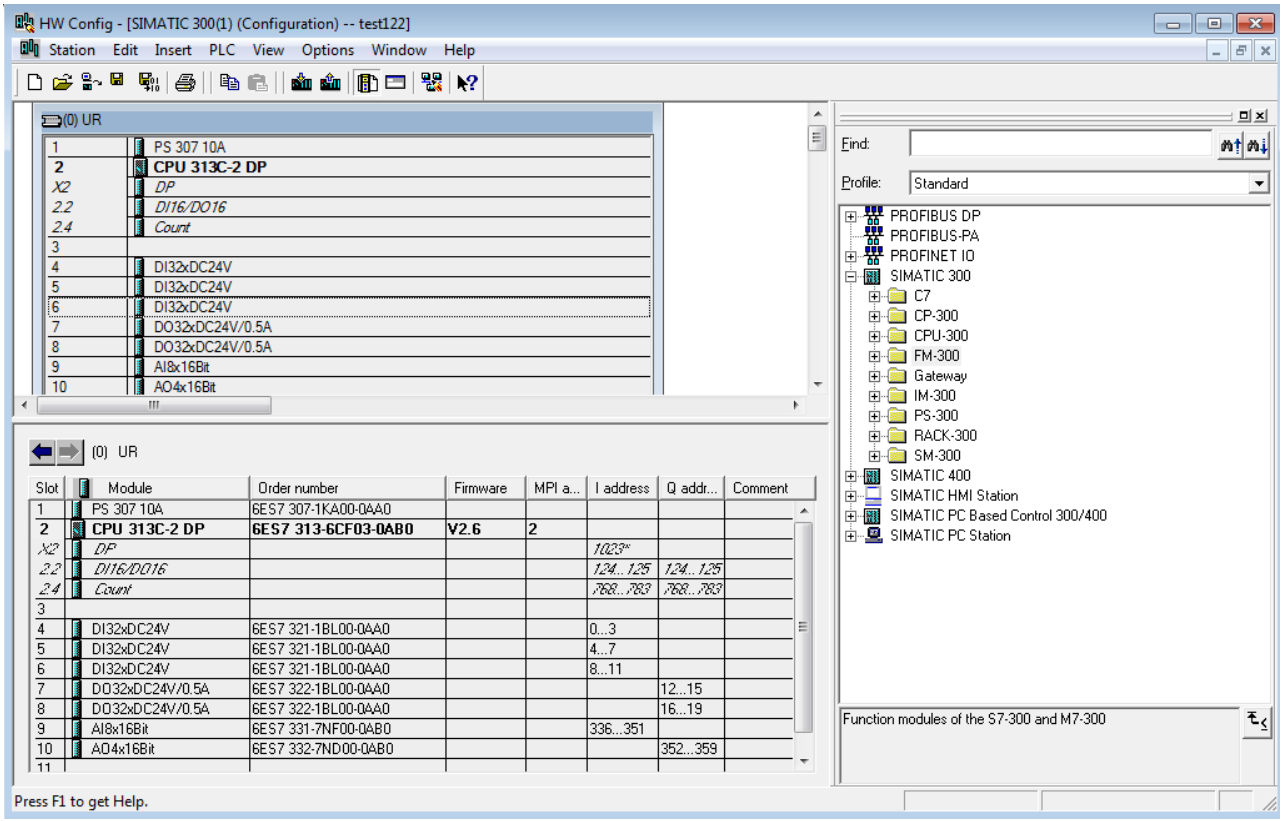
وحتى نقوم باختيار وضبط المكونات يجب علينا أولا أن ننشئ مشروع Project لنختار له المكونات حيث نقوم بإدراج محطة S7-300 Station بعد إنشاء المشروع وبمجرد تنفيذ ذلك سوف نجد عنصر Hardware تحت المحطة المنشأة حديثا نضغط عليه لنفتح أداة ضبط المكونات أو شاشة ضبط المكونات HW Config والتي سنستخدمها لاختيار العناصر وضبط الخصائص الخاصة بكل عنصر

71



لاحظ أن الشاشة أمامنا تتكون من القائمة وأشرطة الأدوات للوصول السريع للعمليات الأكثر استخداما والجزء الأيمن يمثل الكتالوج الذي يحتوي على كل العناصر المعتمدة لدى سيمنس وأسفل منها يوجد نافذة عرض لتفاصيل العنصر المختار والجزء في صدر الصفحة لأعلى هو الذي نقوم بإدراج العناصر المختارة إليه أما الجزء الأسفل فيحتوي على تفاصيل العناصر المختارة سواء رقم الأمر أو العناوين أو وصف العنصر

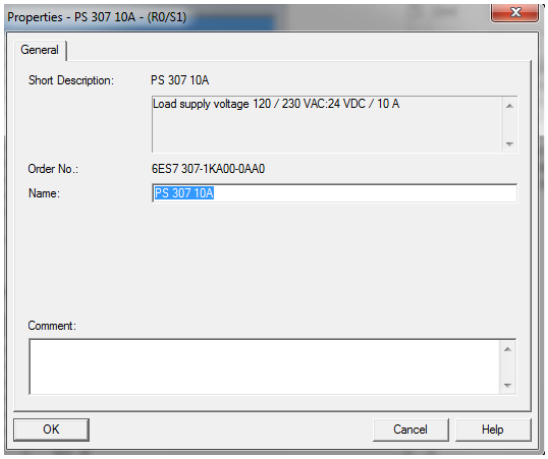
- الخطوة الأولى لتكوين المنظومة هي إدراج Rail لتثبيت المكونات عليه ويحتوي كل Rail على عدد (11) مكان لتثبيت المكونات الأول منها محجوز لمصدر الجهد PS والثاني لوحدة المعالجة CPU والثالث لوحدات الواجهة IM ومن الرابع وحتى الحادي عشر بعدد (8) وحدات لباقي أنواع الوحدات
- نقوم بعد ذلك باختيار العناصر إما من قائمة Insert ثم Insert Object وسوف يظهر فهرس بالعناصر الممكنة حسب الموضع نختار من بينها أو بالوقوف على المكان المطلوب إدراج العنصر إليه ثم نضغط ماوس يمين ثم Insert Object أو أن نختار من الكتالوج في يمين الصفحة ونسحب العنصر للموضع المطلوب
- وبعد الانتهاء من اختيار العناصر نخزن العمل ونخرج أو نكمل خطوات الضبط إن كان هناك ضبط مطلوب وهو ما سنتعرض له في الجزء التالي (لاحظ الشكل بعد إدراج المكونات بالطرق المذكورة)



72

ضبط خصائص المكونات

من شاشة ضبط المكونات ومن الجزء الأعلى الذي يتم فيه إدراج المكونات لعرض وضبط المكونات نضغط على العناصر التي اخترناها لكي نقوم بعملية ضبط الخصائص لها وسوف نستعرض جزءا من ذلك

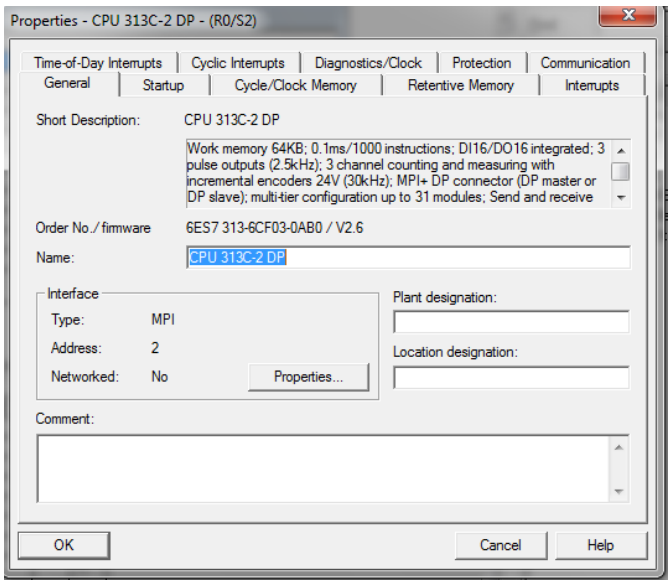


شاشة خصائص مصدر الجهد تحتوي على بيانات عن مواصفات الوحدة من جهد التغذية مثلا 230/120 فولت وجهد الخروج وهو 24 فولت مستمر وتيار الحمل
هذا بالإضافة إلى رقم الأمر Order number
والاسم الرمزي للعنصر Name وهو في حالتنا PS 207 10A

خصائص وحدة المعالجة متعددة ويتم ضبطها في خمس شاشات مختلفة ومقسمة تقسيم وظيفي

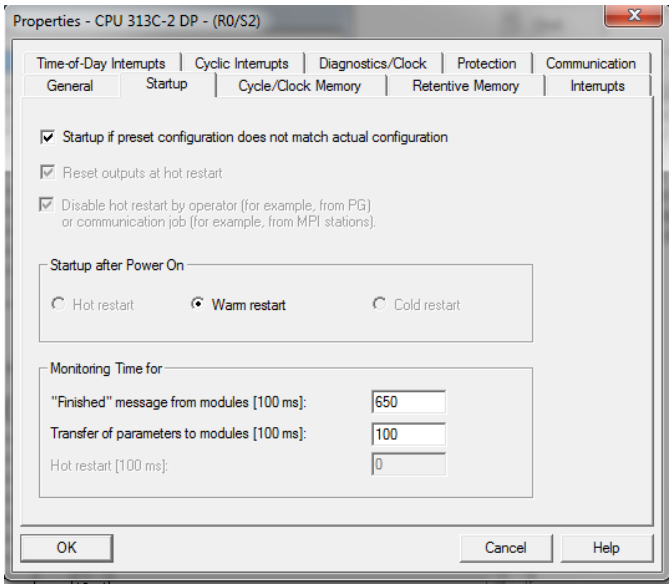
1- الخصائص العامة General

وتشمل وصف مختصر للوحدة مثلا CPU 313C-2DP ووصف كامل لإمكانيات الوحدة الذاكرة وسرعة تنفيذ العمليات وعدد المدخلات والمخرجات على الوحدة ... إلى آخر الوصف الفني للوحدة ، إضافة إلى الجزء Interface والخاص بضبط منافذ الاتصال كما يمكن إضافة تعليق بسيط يظهر مع الوحدة في المشروع



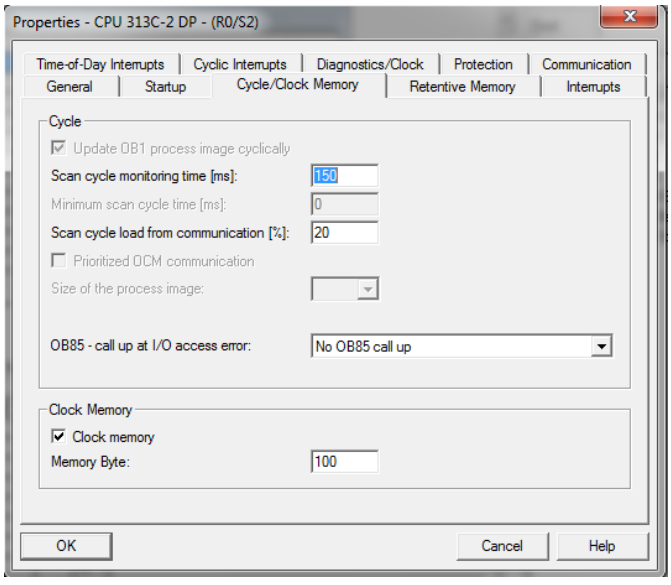
2- خصائص بدء التشغيل Start up

وتشمل اختيار مراجعة موجودات المكونات الفعلية مع التي تم ضبطها والبدء في حالة الاختلاف أم لا وتشمل كذلك طريقة اختيار نوع البدء بعد إعادة توصيل التغذية كما تشمل ضبط لنوعين من الوقت الأول هو وقت انتظار جاهزية الوحدات الفرعية والثاني هو وقت نقل بيانات الضبط للوحدات الفرعية



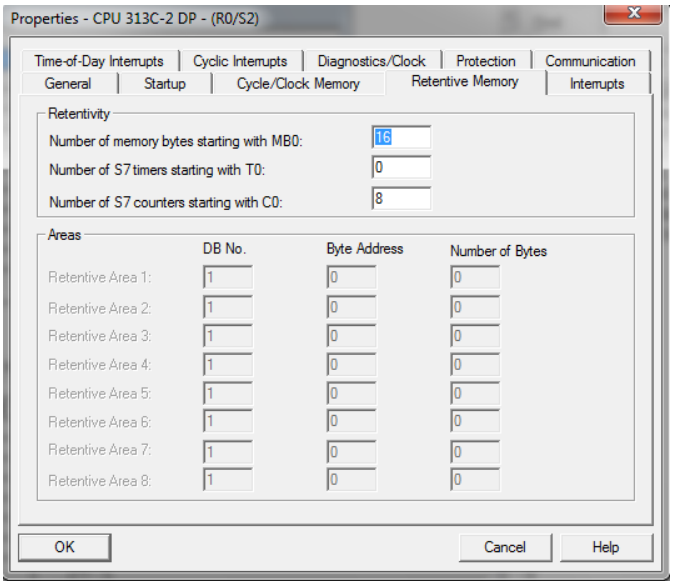
3- ضبط دورة التنفيذ والذاكرة النبضية Clock/Cycle Memory

وتشمل ضبط الحد الأقصى لوقت تنفيذ دورة البرنامج والتحميل من منافذ الاتصال كما تتضمن برمجة Memory Byte لتعطي نبضات بوقت مختلف Pulses على خاناتها المختلفة حيث تكون الخانة رقم "0" هي الأسرع والخانة رقم "7" هي الأبطأ



4- ضبط الذاكرة المستديمة Retentive Memory:

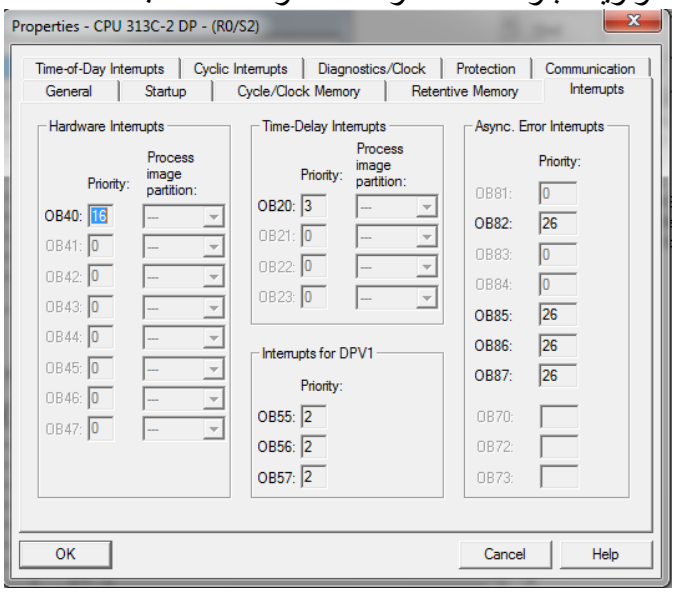
وتشمل اختيار عناصر الذاكرة بأشكالها المختلفة سواء من النوع M أو المؤقتات أو العدادات أو قوالب البيانات Data Blocks والتي تحتفظ بحالتها عند فصل التغذية الكهربية



73

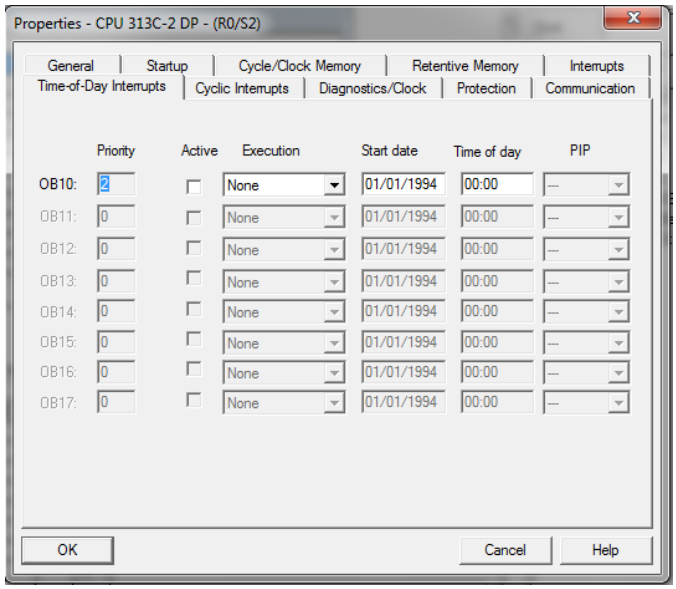
5- ضبط المقاطعة Interrupts:

ويتم فيها ضبط تشغيل بلوكات المقاطعة Interrupts والتي تشمل مقاطعة المكونات Hardware Interrupts ومقاطعة تأخير الوقت Time-Delay Interrupts ومقاطعة تزامن الأخطاء Async. Error Interrupts ومقاطعة نظم الاتصال Interrupts for DPV1 حيث الاولوية للبلوكات المدعومة فقط لوحدة المعالجة



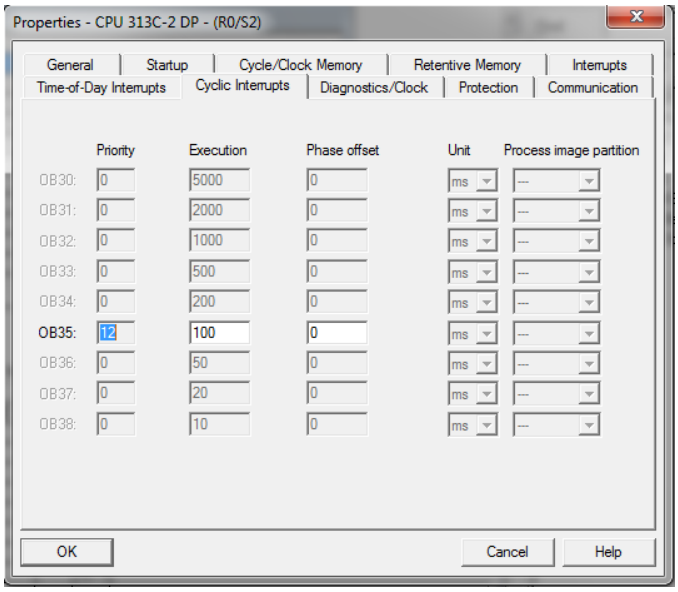
6- مقاطعة وقت اليوم Time of Day Interrupts:

ويتم فيها ضبط المقاطعات المرتبطة بالوقت واليوم حيث يتم اختيار تاريخ ووقت التشغيل للبلوك المدعوم في وحدة المعالجة ويتم كتابة التعليمات المطلوب تنفيذها في البلوك المطلوب



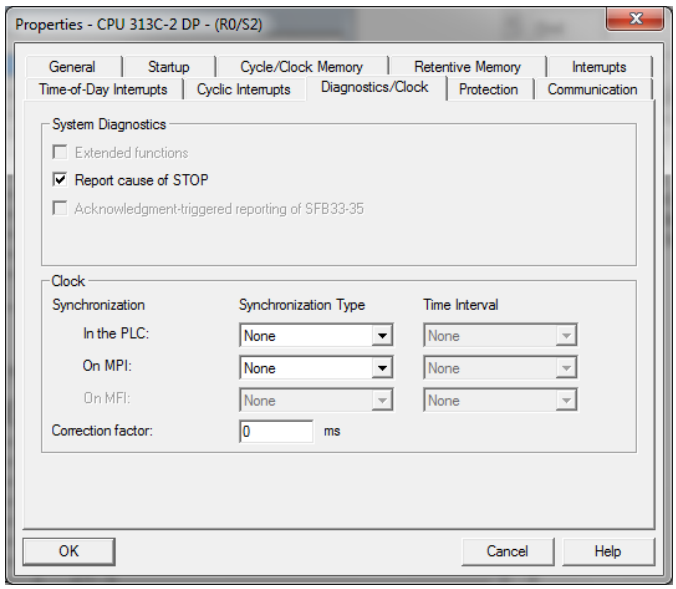
7- ضبط المقاطعة الدورية Cyclic Interrupts:

وتشمل ضبط وقت الحساب والأولوية لبلوكات المقاطعة الدورية أو المدعوم منها والتي تحتوي عمليات الوقت الحقيقي Real-time operations ويتم كتابة تلك التعليمات داخل البلوك



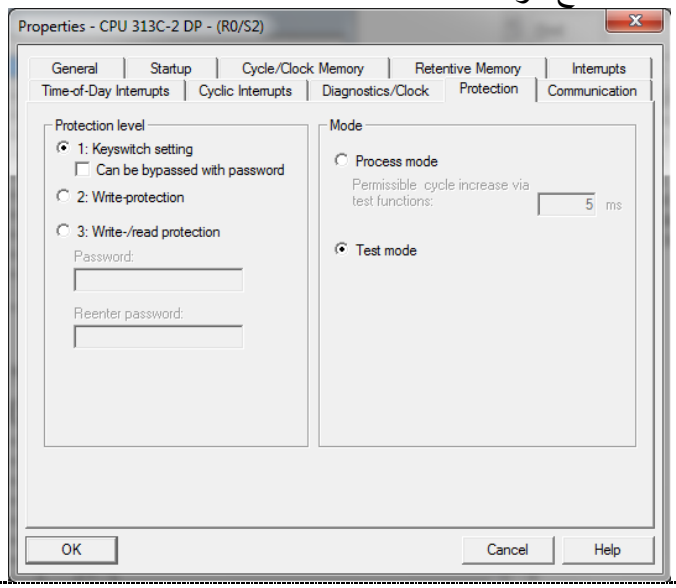
8- ضبط تتبع الأعطال والساعة Diagnostic/Clock:

وتشمل اختيار الإفادة بسبب التوقف أو لا وضبط تزامن الساعة الخاصة بوحدة المعالجة إما مع المتحكم أو على قناة الاتصال MPI



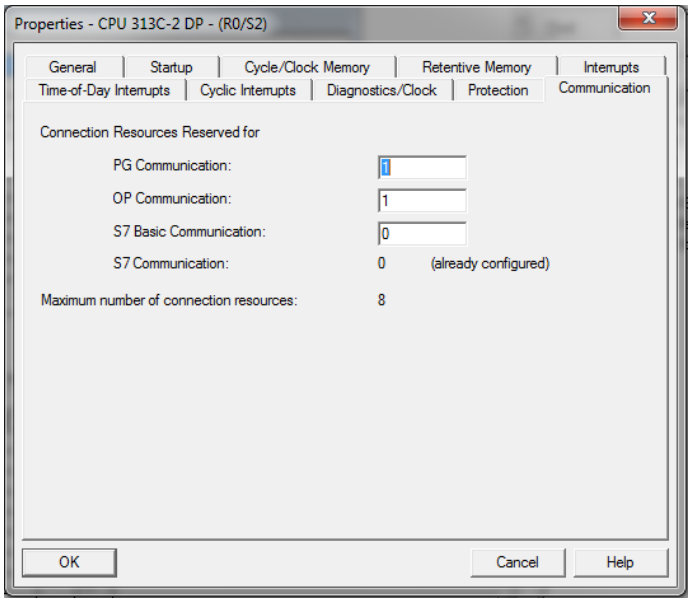
9- ضبط الحماية Protection:

وتشمل ضبط مستويات الحماية سواء وصول كامل للبيانات بالقراءة Read والكتابة Write حيث يمكن القراءة من الوحدة والكتابة عليها أو مستوى القراءة فقط حيث يتم فقط قراءة البيانات أو منع القراءة والكتابة مطلقا وهو النوع الثالث وأعلى درجات الحماية وفي الحالتين الثانية والثالثة يتطلب الأمر إدخال كلمة سر للتمكن من التعامل مع الوحدة



10- ضبط الاتصال Communication:

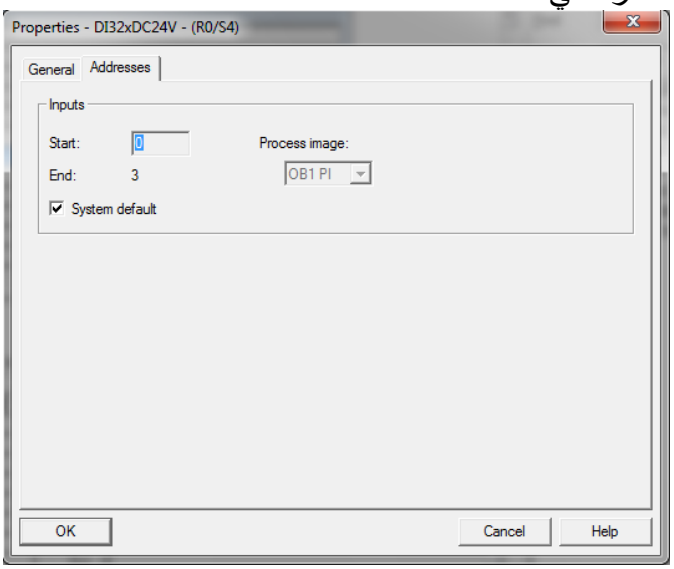
وتشمل ضبط بيانات الاتصال مع جهاز البرمجة ووحدات التشغيل حيث عناوين تلك الوحدات للتعامل



ويتم أيضا ضبط خصائص وحدات الإدخال والإخراج الرقمية والتناظرية من النوع SM بكل أشكالها على النحو التالي ومثلما يتم ضبط الوحدات المنفردة يتم ضبط الوحدات المختلطة تماما للمدخلات وحدها والمخرجات وحدها

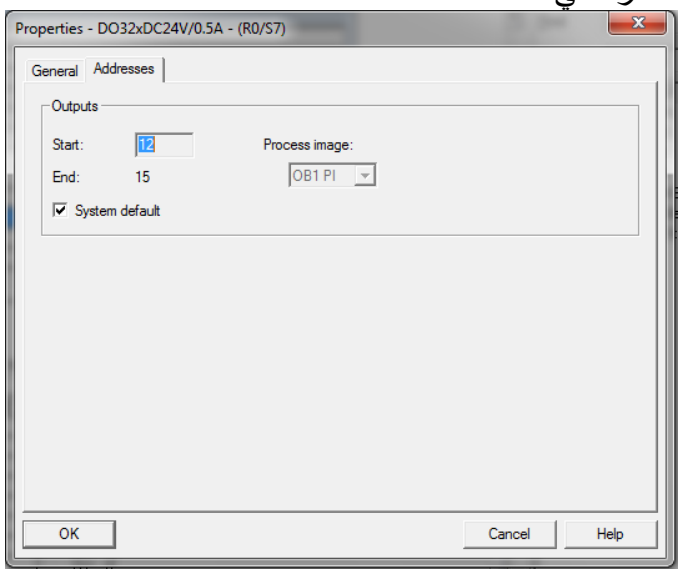
1- وحدات الإدخال الرقمية Digital input modules:

في الشاشة الأولى يتم عرض مواصفات الوحدة والاسم الرمزي المختار لها والتعليق الخاص وفي الشاشة الثانية يتم ضبط العنوان Address أو الإبقاء على العنوان الافتراضي



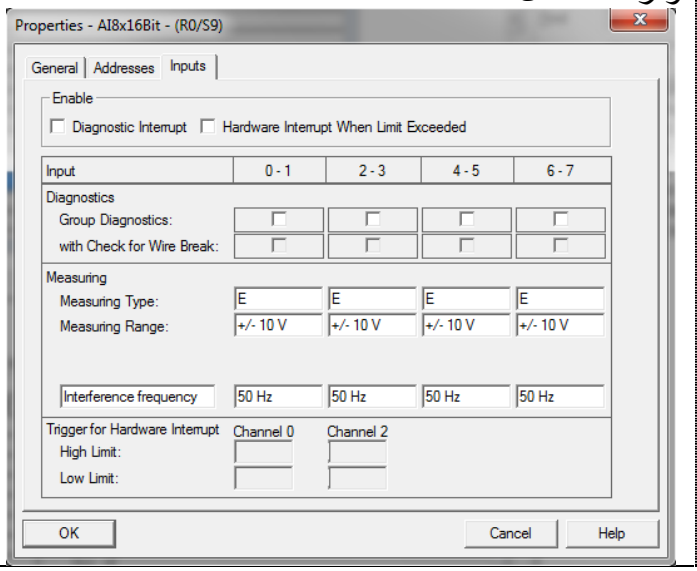
2- وحدات الإخراج الرقمية Digital output modules:

في الشاشة الأولى يتم عرض مواصفات الوحدة والاسم الرمزي المختار لها والتعليق الخاص وفي الشاشة الثانية يتم ضبط العنوان Address أو الإبقاء على العنوان الافتراضي



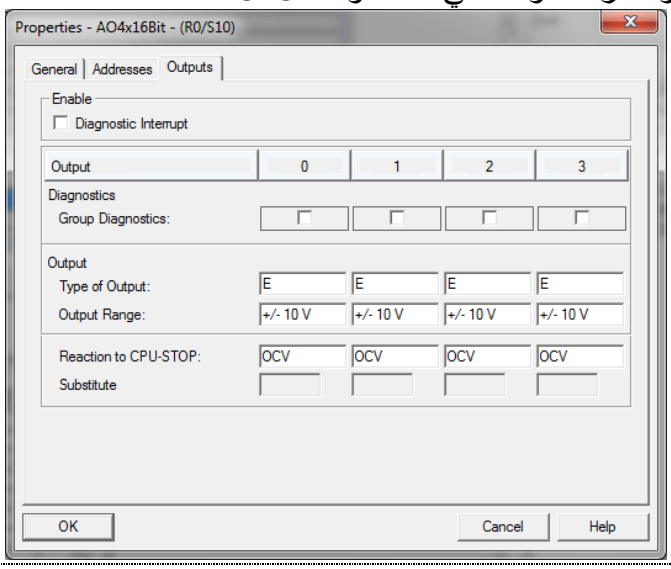
3- وحدات الإدخال التناظرية Analog input modules:

في الشاشة الأولى يتم عرض مواصفات الوحدة والاسم الرمزي المختار لها والتعليق الخاص وفي الشاشة الثانية يتم ضبط العنوان Address أو الإبقاء على العنوان الافتراضي وفي الشاشة الثالثة يتم اختيار تفعيل المقاطعة في حالة حدوث خطأ وتتبع الأعطال ونوع القياس ونطاقه وتردد التداخل



4- وحدات الإخراج التناظرية Analog output modules:

في الشاشة الأولى يتم عرض مواصفات الوحدة والاسم الرمزي المختار لها والتعليق الخاص وفي الشاشة الثانية يتم ضبط العنوان Address أو الإبقاء على العنوان الافتراضي وفي الشاشة الثالثة يتم اختيار تفعيل المقاطعة في حالة حدوث خطأ وتتبع الأعطال ونوع القياس ونطاقه وتصرف الوحدة في حالة توقف CPU



ويتم ضبط الوحدات الوظيفية الخاصة طبقا للتطبيق المستخدمة به والبلوكات القياسية المستخدمة معها والتي سوف تجدها في كتاب S7-300 CPU Technological Functions حيث يتعرض بالشرح التفصيلي لاستخدام تلك البلوكات مع وحدات العمليات الخاصة والضبط التفصيلي لها.

76

كتابة محتويات البلوكات Editing Blocks

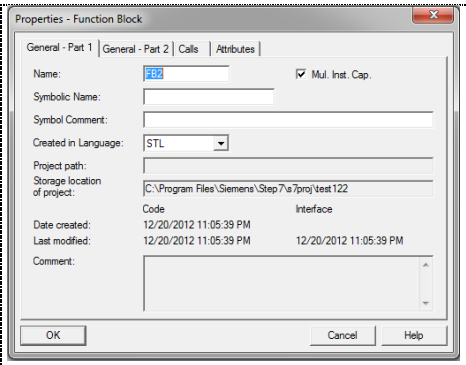
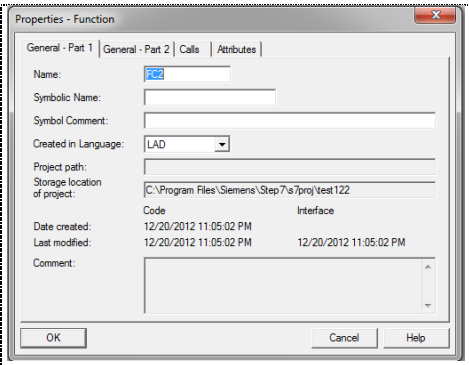
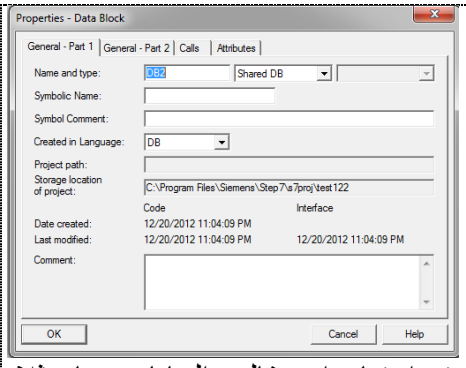
ذكرنا من قبل أن البلوكات STEP7 Block تنقسم إلى نوعين رئيسيين بحسب محتوياتها إلى :

- بلوكات بيانات Data Blocks بنوعيها DB, UDT حيث يتم كتابة بيانات فقط فيها
- بلوكات منطقية Logical Blocks بأنواعها الخمسة OB,FC,FB,SFB,SFC حيث يتم كتابة تعليمات وبيانات فيها

ولكتابة محتويات البلوكات يتم ذلك عن طريق محرر البلوكات Block Editor والذي يطلق عليه LAD/STL/FBD Editor حيث يستخدم لكتابة كل من بلوكات البيانات وبلوكات التعليمات

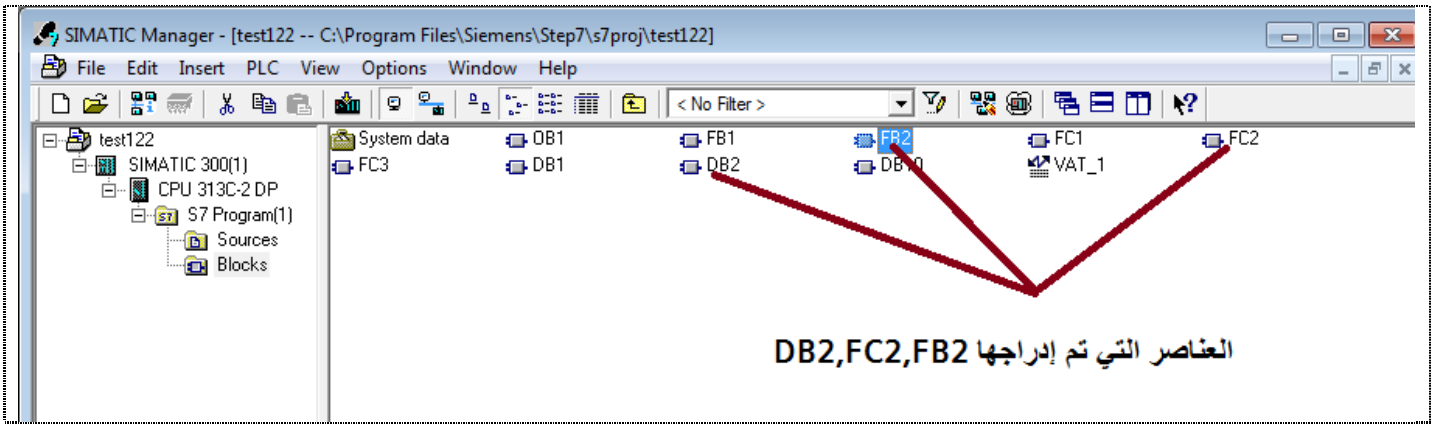
في البداية نقوم بإنشاء المشروع Project ثم ندرج تحته محطة S7-300 Station ثم ننشئ المكونات من خلال أداة ضبط المكونات HW Config وعلى الأقل نقوم بإدراج وحدة معالجة ننشئ لها البرنامج وتحت S7-Program سوف تجد دليل فرعي بإسم Blocks بشكل مبدئي يحتوي على OB1 وهو المكان الذي سننشئ فيه البلوكات كلها ونقوم بتحرير محتوياتها باستخدام الشاشة LAD/STL/FBD Editor

نقوم بإدراج 3 بلوكات الأول Data Block DB والثاني FC والثالث FB إما من خلال قائمة Insert ثم S7 Block ثم نختار أولا Data Block ثم بعد ذلك وبنفس الطريقة Function ثم وبنفس الطريقة Function Block

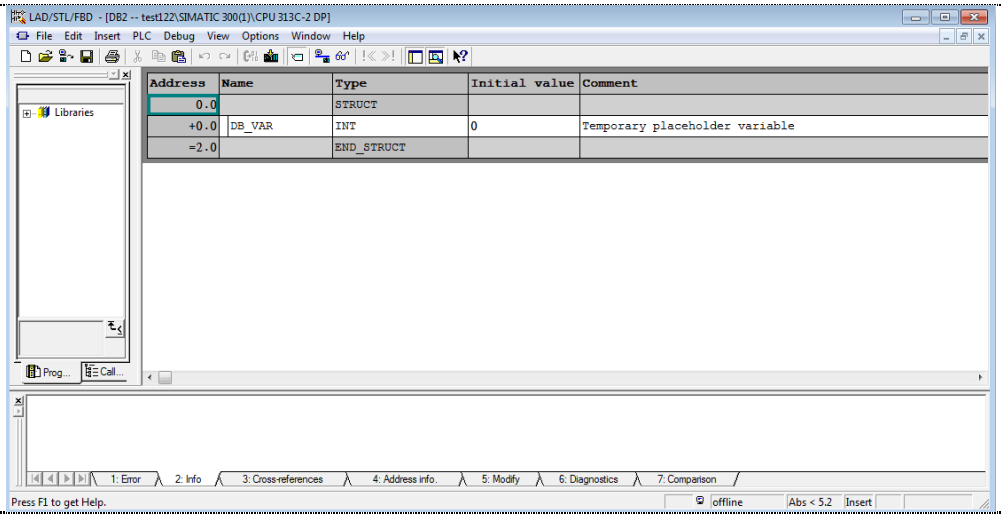
		
يتم اختيار اسم البلوك مثلا FB2 وإدخال اسم رمزي وتعليق ثم OK	يتم اختيار اسم البلوك مثلا FC2 وإدخال اسم رمزي وتعليق ثم OK	يتم اختيار اسم قالب البيانات هنا مثلا DB2 وندخل اسم رمزي وتعليق ثم نضغط OK

77

لاحظ وجود البلوكات الثلاثة بعد إدراجها



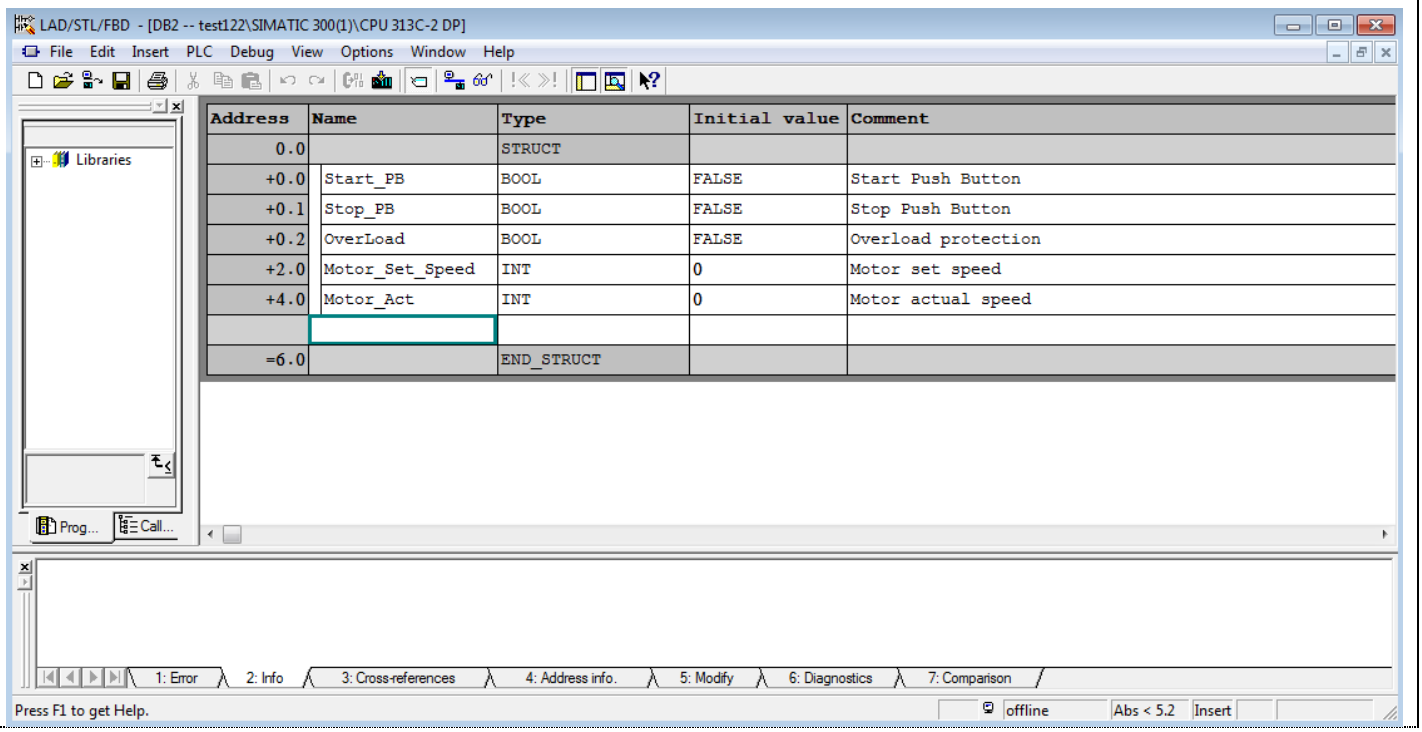
والآن نقوم بتحرير محتويات بلوك البيانات الذي أنشأناه بالضغط عليه مرتين بالماوس وسوف يفتح نافذه مثل التالية



كما يظهر بالصورة فإن البيانات تأخذ شكل الجدول الذي يمثل قاعدة بيانات كل سجل فيها يمثل أحد البيانات في حين أن حقول البيانات هي الأعمدة ولكل بيان

- عنوان Address
- إسم رمزي Name
- نوع البيانات Type
- قيمة أولية Initial value
- تعليق Comment

وتخضع العناوين للحجم والعنوان الذي يظهر على الشمال هو عنوان أول Byte في البيان فإن كان Bit يأخذ خانة واحدة وإن كان Byte يأخذ 8 خانات ، لكن العناوين يتم تكوينها آليا بفارق Word أي 2-Byte وبشكل زوجي دائما وفي الشكل التالي مجموعة من البيانات بعد كتابتها



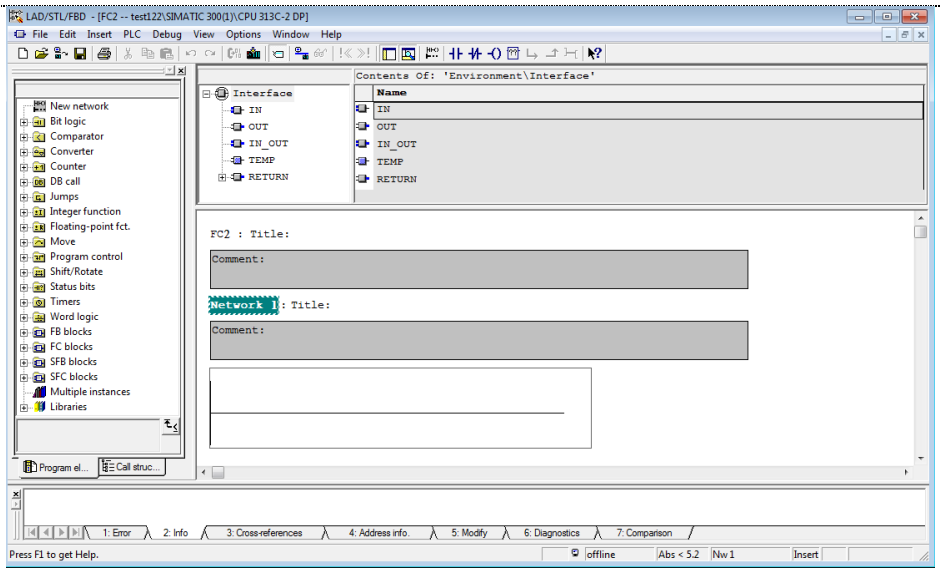
78

ويجب تحرير جميع بلوكات البيانات قبل استخدامها في البرنامج ولا يجب أن تستدعى بيانات بلوكات البيانات داخل البرنامج قبل إنشائها وتخزينها وتحميلها مع البرنامج وعند استعداء بيانات لم يتم إنشاؤها سوف يؤدي ذلك إلى توقف المتحكم

ويمكن منادة البيانات بالحد الأقصى لها مثلا DB2.DBX0.1 و DB2.DBW2 و DB2.DBW4 وهكذا

وعند الرغبة في مراقبة التشغيل للمتحكم Online من خلال Data Block فسفوف تظهر خانة جديدة في الجدول تمثل القيمة الفعلية عند المراقبة Actual value يكون فيها القيمة الفعلية للبيانات وتكون على يمين Initial value

ننتقل بعد ذلك إلى تحرير Function FC2 كمثال على تحرير أي بلوك من هذا النوع

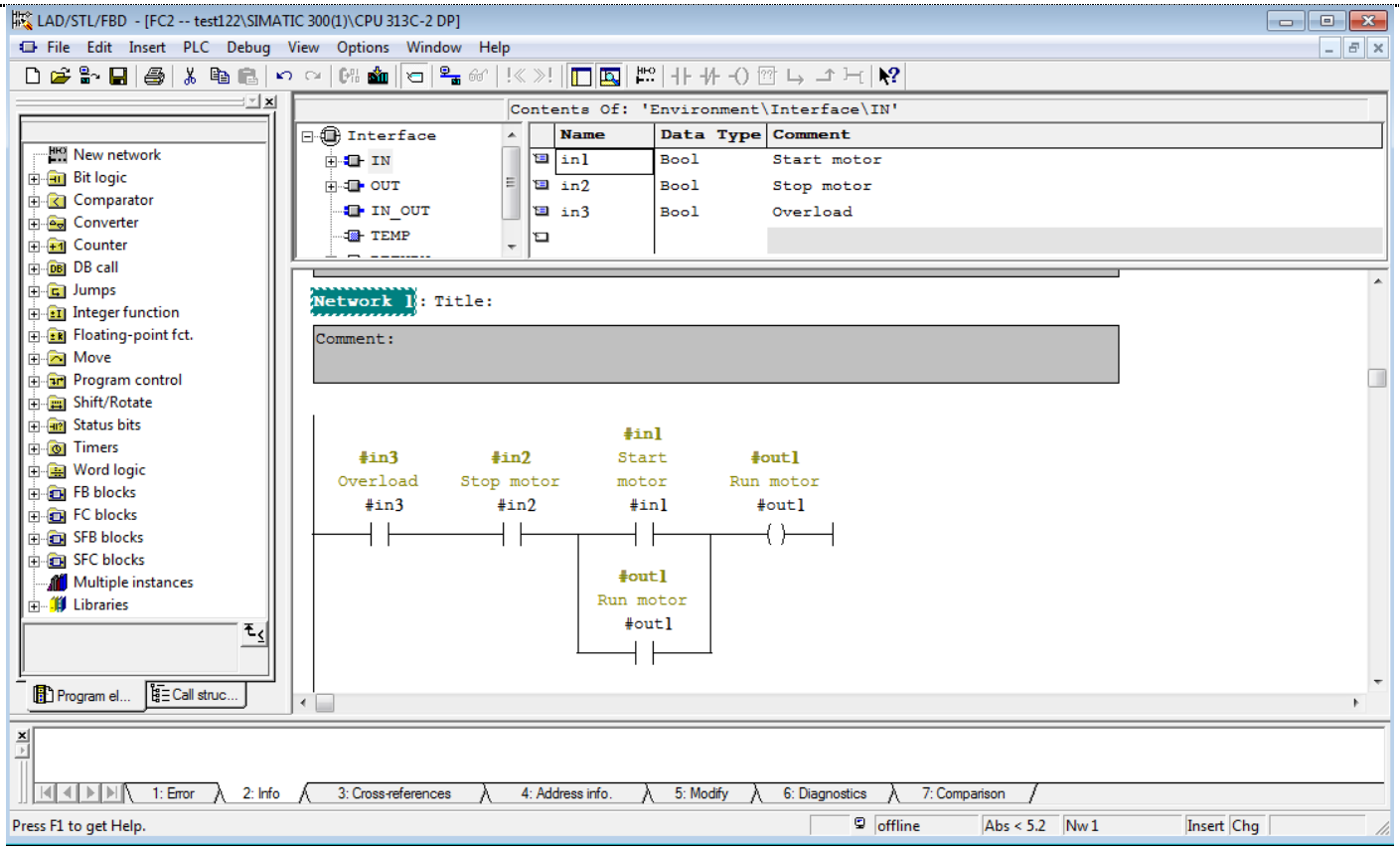


لاحظ في الشكل أمامنا أنه يتكون من منطقة القائمة وشريط الأدوات الذي يحتوي على الأدوات الأكثر استخداما ثم إلى اليسار مجموعة التعليمات التي تستخدم في الكتابة وإلى اليمين في أعلى الصفحة يوجد التوصيف الخاص بمدخلات ومخرجات البلوك Interface والذي يحتوي على 5 أنواع هي المدخلات IN والمخرجات OUT والعناصر التي تشترك في الإدخال والإخراج IN_OUT والذاكرة المؤقتة TEMP والقيمة التي تعود من البلوك RETURN

القسم الأعلى والخاص بالتوصيف يتم فيه اختيار المدخلات والمخرجات حيث يتم كتابة أسماء تلك المتغيرات ونوعها والذي سيظهر أثناء استدعاء البلوك للتنفيذ

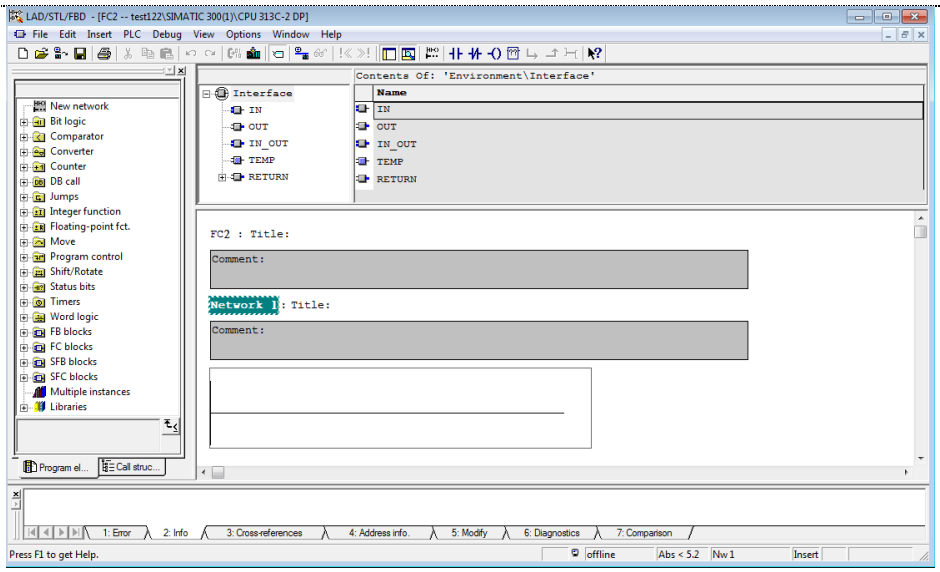
أما الجزء الرئيسي والذي سيتم كتابة التعليمات فيه فهو يتكون من أجزاء فرعية تسمى Networks يتكون منها البلوك حيث يتم كتابة محتويات البلوك على أجزاء كل جزء منها في أحد تلك الحلقات أو الشبكات Networks

في الشكل التالي مثال على شكل Interface وكذلك جزء من التعليمات



يجب ملاحظة أن هذا البلوك أو غيره من نفس النوع FC لا يتم تنفيذها فعليا في البرنامج إلا إن تم استدعاؤها بشكل مباشر أو ضمنيا في أحد البلوكات من النوع Organization blocks OB

ننتقل بعد ذلك إلى تحرير Function FC2 كمثال على تحرير أي بلوك من هذا النوع

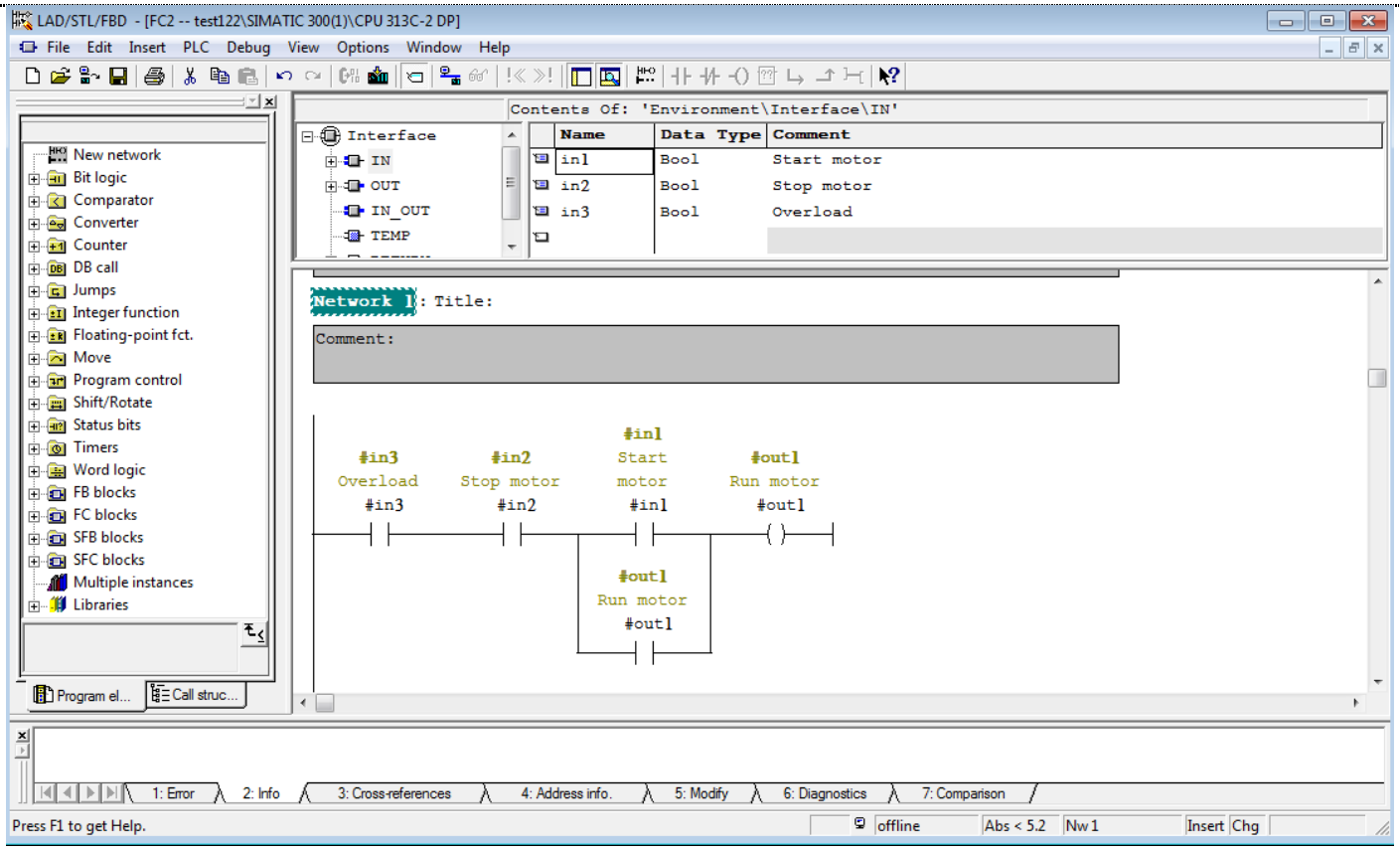


لاحظ في الشكل أمامنا أنه يتكون من منطقة القائمة وشريط الأدوات الذي يحتوي على الأدوات الأكثر استخداما ثم إلى اليسار مجموعة التعليمات التي تستخدم في الكتابة وإلى اليمين في أعلى الصفحة يوجد التوصيف الخاص بمدخلات ومخرجات البلوك Interface والذي يحتوي على 5 أنواع هي المدخلات IN والمخرجات OUT والعناصر التي تشترك في الإدخال والإخراج IN_OUT والذاكرة المؤقتة TEMP والقيمة التي تعود من البلوك RETURN

القسم الأعلى والخاص بالتوصيف يتم فيه اختيار المدخلات والمخرجات حيث يتم كتابة أسماء تلك المتغيرات ونوعها والذي سيظهر أثناء استدعاء البلوك للتنفيذ

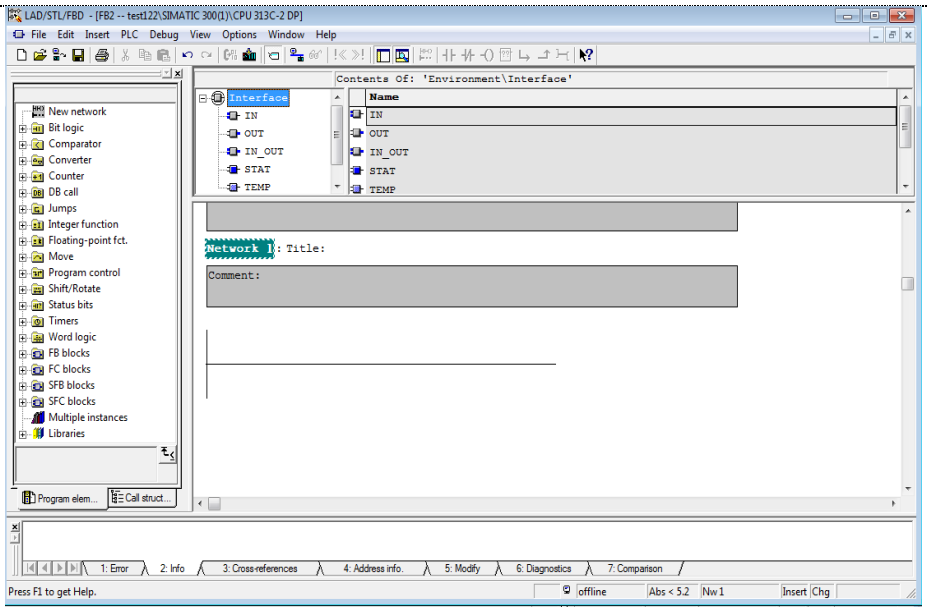
أما الجزء الرئيسي والذي سيتم كتابة التعليمات فيه فهو يتكون من أجزاء فرعية تسمى Networks يتكون منها البلوك حيث يتم كتابة محتويات البلوك على أجزاء كل جزء منها في أحد تلك الحلقات أو الشبكات Networks

في الشكل التالي مثال على شكل Interface وكذلك جزء من التعليمات



يجب ملاحظة أن هذا البلوك أو غيره من نفس النوع FC لا يتم تنفيذها فعليا في البرنامج إلا إن تم استدعاؤها بشكل مباشر أو ضمنيا في أحد البلوكات من النوع Organization blocks OB

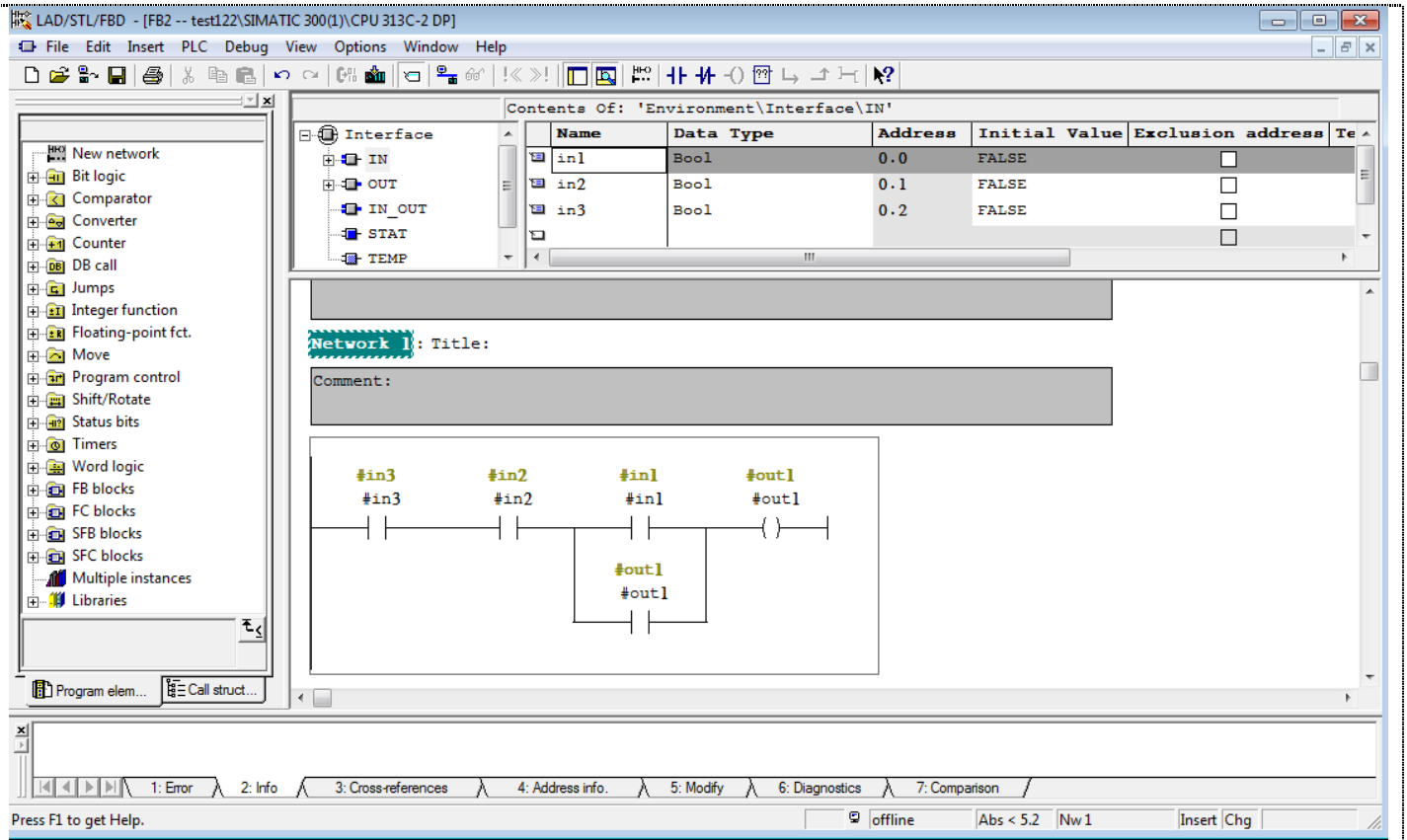
ننتقل بعد ذلك إلى تحرير Function Block FB2 كمثال على تحرير أي بلوك من هذا النوع



لاحظ في الشكل أمامنا أنه يتكون من منطقة القائمة وشريط الأدوات الذي يحتوي على الأدوات الأكثر استخداما ثم إلى اليسار مجموعة التعليمات التي تستخدم في الكتابة وإلى اليمين في أعلى الصفحة يوجد التوصيف الخاص بمدخلات ومخرجات البلوك Interface والذي يحتوي على 6 أنواع خلافا للنوع FC هي المدخلات IN والمخرجات OUT والعناصر التي تشترك في الإدخال والإخراج IN_OUT والحالة STAT والذاكرة المؤقتة TEMP والقيمة التي تعود من البلوك RETURN

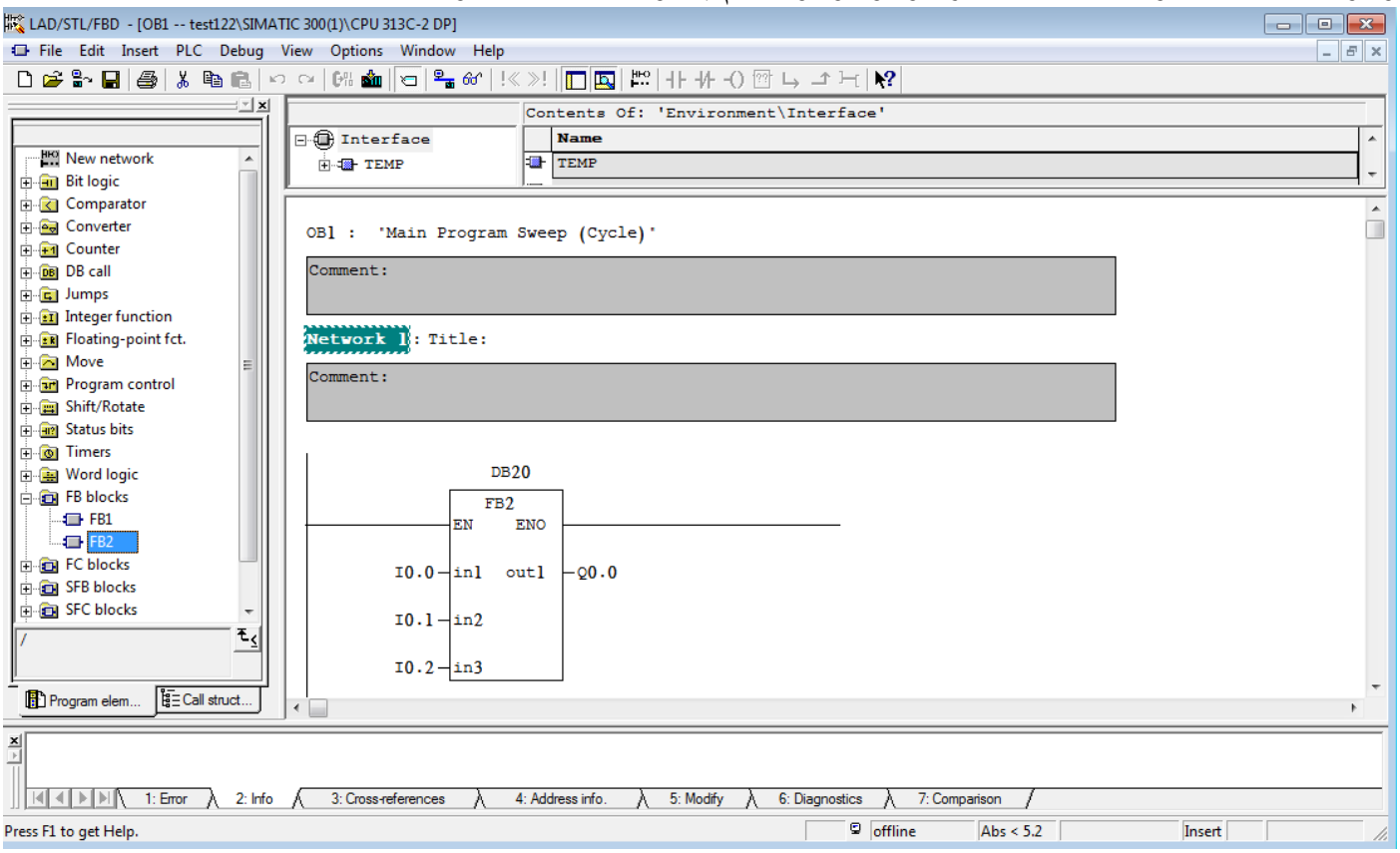
القسم الأعلى والخاص بالتوصيف يتم فيه اختيار المدخلات والمخرجات حيث يتم كتابة أسماء تلك المتغيرات ونوعها والذي سيظهر أثناء استدعاء البلوك للتنفيذ والأنواع الأربعة IN,IN_OUT,OUT,STAT يمثل تكوينها وعناوينها وتوصيفها ما سيكون عليه شكل Data Block المصاحب للبلوك والذي سيتم إنشاؤه عن طريق FB الذي قمنا بإنشائه أما الجزء الرئيسي والذي سيتم كتابة التعليمات فيه فهو يتكون من أجزاء فرعية تسمى Networks يتكون منها البلوك حيث يتم كتابة محتويات البلوك على أجزاء كل جزء منها في أحد تلك الحلقات أو الشبكات Networks

81 في الشكل التالي مثال على شكل Interface وكذلك جزء من التعليمات



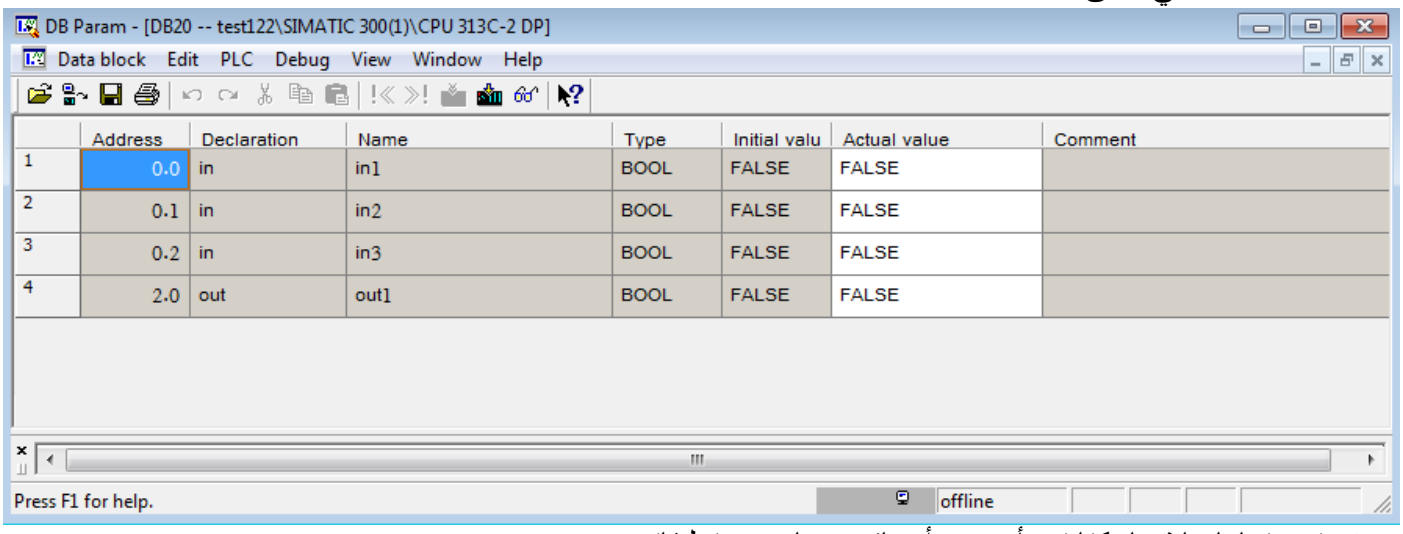
يجب ملاحظة أن هذا البلوك أو غيره من نفس النوع FB لا يتم تنفيذها فعليا في البرنامج إلا إن تم استدعاؤها بشكل مباشر أو ضمنيا في أحد البلوكات من النوع Organization blocks OB

يتم استدعاء FB2 في OB1 مثلا كي يتم تنفيذه وسوف تجد البلوكات التي تم إنشاؤها جميعا في القائمة على شمال الشاشة تحت FB blocks و FC blocks ونقوم بسحب البلوك إلى Network 1 ونقوم بكتابة اسم بلوك البيانات مثلا DB20 وسوف يسأل أن البلوك المصاحب غير موجود وسوف يتم إنشاؤه ، اضغط OK وأكمل



82

ولاحظ شكل بلوك البيانات في الصورة التالية وقارن بينه وبين شكل العناصر التي قمنا بتوصيفها في الجزء الخاص بعمل Interface في أعلى شاشة الكتابة



سوف تجد تماما ما قمنا بكتابته بأنوعه وأسمائه وعناوينه وتعليقاته

الأنواع الأخرى SFC,SFB يتم استدعاؤها فقط داخل البلوكات الأخرى لتنفيذ مهام خاصة حسب التصميم الخاص بها من شركة سيمنس

النوع OB يخضع تماما لنفس القواعد في الكتابة مثل FC غير أن الجزء Interface مخصص من قبل حسب المقاطعة الخاصة به Interrupt وحسب الظروف التي يستدعى فيها البلوك ويمكن استخدام تلك البيانات لكن لا يتم التغيير في شكلها.

قبل أن نبدأ في عرض أوامر البرمجة نستعرض معا الذواكر الرئيسية التي تتم عليها عمليات البرمجة داخل وحدة المعالجة فكما ذكرنا أنه حسب حجم البيانات والتي تنقسم إلى (bit و byte و word و double word) فحسب هذا الحجم تتم العمليات ولابد من وجود ذواكر وسيطة Buffer Memory or Registers لتتم عليها تلك العمليات:

- فالعمليات التي تتطلب حجما أكبر من bit واحدة تتم على ما يسمى Accumulator حيث أنه يوجد لكل وحدة معالجة وحدتي Accumulator على الأقل ويرمز بهما بالرموز ACCU1 و ACCU2 وفي بعض الأنواع من المعالجات يوجد عدد 4 وحدات Accumulator وتأخذ الرموز ACCU1 و ACCU2 و ACCU3 و ACCU4 تتم عليها العمليات التي تتطلب حجم بيانات أكبر من bit ، وحجم وحدة Accumulator هو 32-bits

Accumulator(32-bit)																															
High Word (W-H)																Low Word (W-L)															
High Byte (H-H)								Low Byte (H-L)								High Byte (L-H)								Low Byte (L-L)							
31															16	15															0

ويتكون Accumulator من عدد 2-WORD وعدد 4-BYTE فعند التعامل مع BYTE يتم تحميل البيانات في البايت L-L وفي حالة التعامل مع WORD يتم التعامل مع W-L أما في حالة التعامل فيتم التعامل مع DWORD ويخضع شكل البيانات في Accumulator لنفس صيغ البيانات التي تحدثنا عليها من قبل سواء كانت صحيحة INT أو صحيحة مضاعفة DINT أو حقيقية REAL أو غير ذلك من البيانات مثل S5T أو DATE أو TIME

- أما على مستوى البيانات التي تحتل خانة واحدة فهناك مجموعة كبيرة من الخانات يصل عددها إلى 9 خانات يجمعها معا ما يسمى STATUS Register ويطلق عليها STATUS Bits حيث كل واحدة منها تتغير طبقا للعمليات المختلفة لكنها تعطي انطباعا منطقيا عن حالة تنفيذ العمليات وهذه الخانات STATUS Bits هي:
  - **RLO** نتيجة العمليات المنطقية Result of Logic Operations حيث يتم فيها تخزين نتيجة آخر عملية منطقية وتتم عليها العمليات المنطقية أيضا.
  - **OV** زيادة في قيمة البيانات Overflow وتقوم بمراقبة حدوث تجاوز في القيمة الرقمية لحدود التخزين عند تنفيذ العمليات الحسابية ويتم تصفيرها بمجرد البدء في عملية جديدة
  - **OS** زيادة في قيمة البيانات مع تخزين الحالة Stored Overflow وتقوم بنفس عمل OV تماما غير أنها تحتفظ بالحالة "1" ويتم تصفيرها عن طريق أوامر برمجة
  - **CC0,CC1** وتستخدم في عمليات كثيرة منها المقارنة وحسب حالة الاثنين معا يتم تحديد نتيجة المقارنة ، كذلك في الحسابات الحقيقية والصحيحة تعطي انطباعا عن النتيجة وتستخدم في عمليات Shift/Rotate حيث تدخل في بعض العمليات كجزء من مكان التخزين
  - **STA** ووتستخدم لإظهار حالة المتغير الثنائي المستخدم حاليا في العملية وهذا غير RLO والتي تعكس النتيجة للعمليات المنطقية
  - **/FC** يستخدم لتحديد بداية الحسابات المنطقية في كل مرة تبدأ فيها مجموعة من العمليات وتقوم العمليات بتغيير قيمة /FC بما يتناسب مع طبيعة العملية فمثلا And تقوم بتغيير /FC إلى "1" قبل التنفيذ بينما العملية Or تقوم بتغيير /FC إلى الحالة "0"
  - **OR** تستخدم لتخزين النتائج الوسيطة عند عمل عمليات AND قبل أن يتم عليها تنفيذ عملية OR حيث يتم فيها تخزين نتيجة العمليات AND الأخيرة
  - **BR** أو نتيجة ثنائية Binary Result حيث تستخدم عند الانتقال من بلوك لآخر

8	7	6	5	4	3	2	1	0
BR	CC0	CC1	OV	OS	OR	STA	RLO	/FC

وسوف نتحدث عن الطرق الثلاث للبرمجة معا إن شاء الله ونبدأ معا بأول نوع من العمليات وهو العمليات المنطقية

العمليات المنطقية على خانة واحدة Bit Logic Operations:

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
And Operation		
الصيغة العامة A <Address> <Address> →I,Q,M,L,T,C,DB **Example**: A I10.3 A M3.7 A T11 A DB30.DBX2.7 A C2 A L0.3 A Q12.4	**الصيغة العامة** <Address1> <Address2> <Address> →I,Q,M,L,T,C,DB **Example**: M 10.3 I7.7	**الصيغة العامة** <Address1> <Address2> & <Address> →I,Q,M,L,T,C,DB **Example**: I 10.2 M40.7 C15 &
تقوم العملية AND بقراءة قيمة المتغير <Address> وعمل ANDing لها مع قيمة RLO وتخزين النتيجة في RLO		
OR Operation		
الصيغة العامة O <Address> <Address> →I,Q,M,L,T,C,DB **Example**: O I10.3 O M3.7 O T11 O DB30.DBX2.7 O C2 O L0.3 O Q12.4	**الصيغة العامة** <Address1> <Address2> <Address-i> →I,Q,M,L,T,C,DB **Example** DB40.DBX20.3 T 10	**الصيغة العامة** <Address1> <Address2> >=1 <Address-i> →I,Q,M,L,T,C,DB **Example**: I 10.2 M40.7 C15 >=1
تقوم العملية OR بقراءة قيمة المتغير <Address> وعمل ORing لها مع قيمة RLO وتخزين النتيجة في RLO		
NOT Operation		
الصيغة العامة NOT **Example**: O I10.3 O M3.7 O T11 NOT	**الصيغة العامة** NOT **Example** M 10.3 NOT	**الصيغة العامة** <Address> <Address> →I,Q,M,L,T,C,DB **Example**: I 10.2 M40.7 C15 &
تقوم العملية NOT بتغيير حالة RLO من "0" إلى "1" والعكس من "1" إلى "0"		

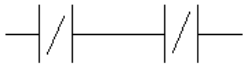
And-NOT Operation

الصيغة العامة

AN <Address>
<Address> →I,Q,M,L,T,C,DB
Example:
AN I10.3
AN M3.7
AN T11
AN DB30.DBX2.7
AN C2
AN L0.3
AN Q12.4

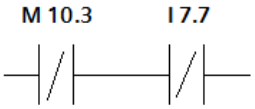
الصيغة العامة

<Address1> <Address2>

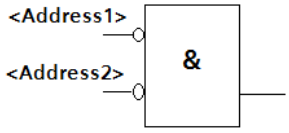


<Address-i> →I,Q,M,L,T,C,DB

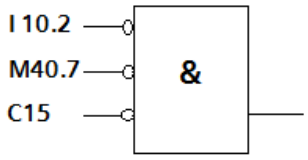
Example:



الصيغة العامة



<Address> →I,Q,M,L,T,C,DB
Example:



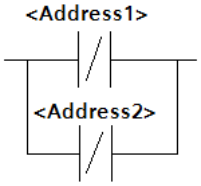
تقوم العملية AND-NOT بقراءة قيمة المتغير <Address> وعمل NOT له ثم ANDing لها مع قيمة RLO وتخزين النتيجة في RLO

OR-NOT Operation

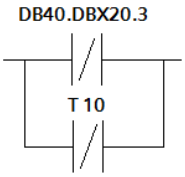
الصيغة العامة

ON <Address>
<Address> →I,Q,M,L,T,C,DB
Example:
ON I10.3
ON M3.7
ON T11
ON DB30.DBX2.7
ON C2
ON L0.3
ON Q12.4

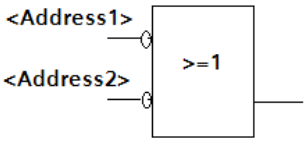
الصيغة العامة



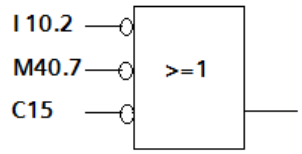
Example



الصيغة العامة



<Address> →I,Q,M,L,T,C,DB
Example:



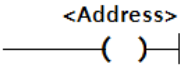
تقوم العملية OR-NOT بقراءة قيمة المتغير <Address> وعمل NOT له ثم ORing لها مع قيمة RLO وتخزين النتيجة في RLO

ASSIGN Operation

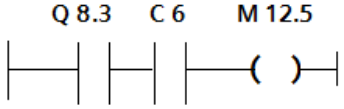
الصيغة العامة

= <Address>
<Address> →I,Q,M,L,DB
Example:
A I10.3
A M3.7
A DB30.DBX2.7
= L0.3
= Q12.4

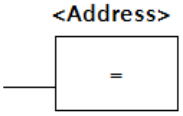
الصيغة العامة



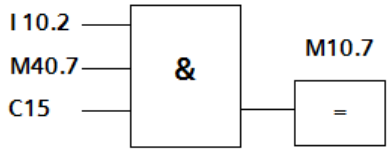
<Address> →I,Q,M,L,DB
Example



الصيغة العامة



<Address> →I,Q,M,L,DB
Example:



تقوم العملية "=" بكتابة قيمة RLO في العنوان <Address>

And, And-NOT before OR Operation

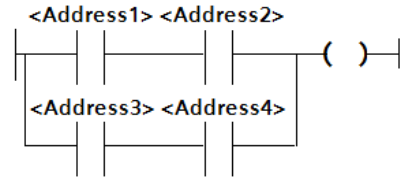
الصيغة العامة

```
O(
A <Address1>
A <Address2>
)
O(
A <Address3>
AN <Address4>
)
<Address-i> →I,Q,M,L,T,C,DB
```

Example:

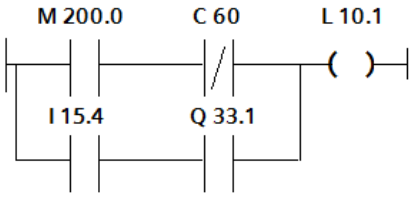
```
O(
A I10.3
AN M3.7
)
O(
AN T11
A C2
)
= Q10.7
```

الصيغة العامة

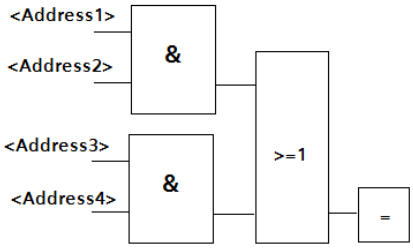


<Address-i> →I,Q,M,L,T,C,DB

Example:

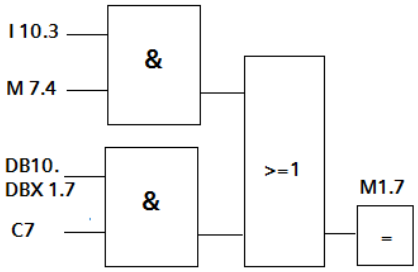


الصيغة العامة



<Address-i> →I,Q,M,L,T,C,DB

Example:



تقوم العملية AND-Before OR بعمل ANDing لمجموعة من المتغيرات قبل عمل OR لها مع مجموعة أخرى تم عمل ANDing لها بنفس الطريقة وتخزين النتيجة في RLO

86

OR,OR-NOT before AND Operation

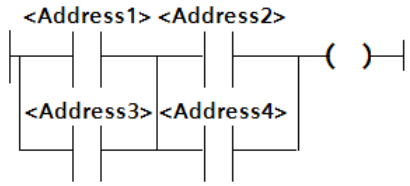
الصيغة العامة

```
A(
O <Address1>
O <Address2>
)
A(
O <Address3>
ON <Address4>
)
<Address-i> →I,Q,M,L,T,C,DB
```

Example:

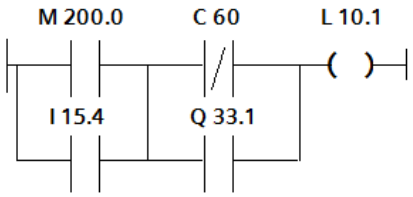
```
A(
O I10.3
ON M3.7
)
A(
ON T11
O C2
)
= Q10.7
```

الصيغة العامة

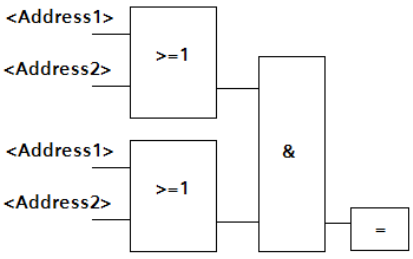


<Address-i> →I,Q,M,L,T,C,DB

Example:

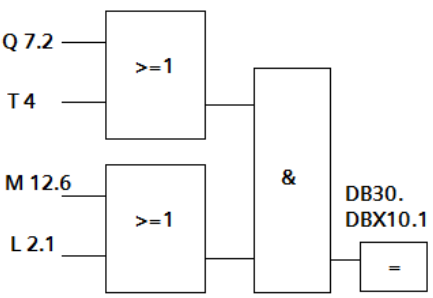


الصيغة العامة



<Address-i> →I,Q,M,L,T,C,DB

Example:



تقوم العملية OR-Before AND بعمل ANDing لمجموعات من المتغيرات قبل عمل OR لها وتخزين النتيجة في RLO

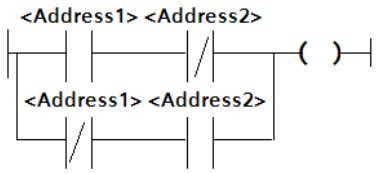
XOR Operation

الصيغة العامة

X <Address1>
X <Address2>
<Address-i> →I,Q,M,L,DB,C,T

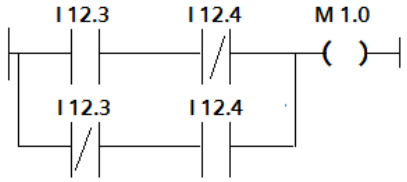
Example:
X I10.3
X M3.7
= Q12.4

الصيغة العامة

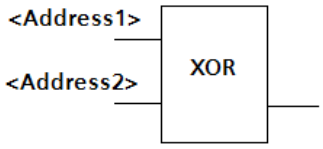


<Address-i> →I,Q,M,L,DB,C,T

Example

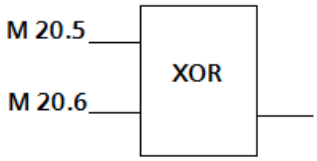


الصيغة العامة



<Address-i> →I,Q,M,L,DB,C,T

Example:



تقوم العملية "XOR" بحساب نتيجة XORing بين المدخلات <Address> وكتابة النتيجة في RLO

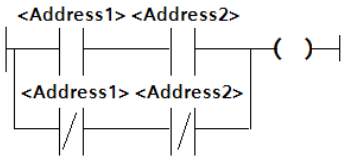
XOR-NOT Operation

الصيغة العامة

XN <Address1>
XN <Address2>
<Address-i> →I,Q,M,L,DB,C,T

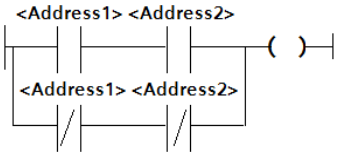
Example:
XN I10.3
XN M3.7
= Q12.4

الصيغة العامة

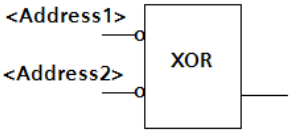


<Address-i> →I,Q,M,L,DB,C,T

Example

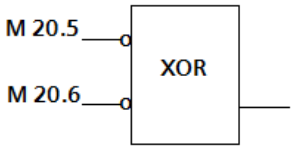


الصيغة العامة



<Address-i> →I,Q,M,L,DB,C,T

Example:



تقوم العملية "NOT-XOR" بعمل NOT للمتغيرات <Address1>,<Address2> ثم حساب نتيجة XORing بينها وكتابة النتيجة في RLO

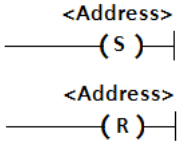
Set/Reset Operations

الصيغة العامة

S <Address>
<Address> →I,Q,M,L,DB
R <Address>
<Address> →I,Q,M,L,DB,C,T

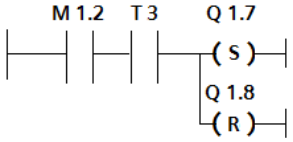
Example:
A I10.3
A M3.7
S Q12.4
R M20.4

الصيغة العامة

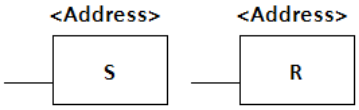


<Address> →I,Q,M,L,DB

Example

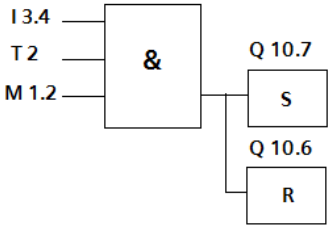


الصيغة العامة



<Address> →I,Q,M,L,DB

Example:



تقوم العملية "SET" بتحويل حالة RLO إلى "1" ونقلها إلى <Address> بينما تقوم العملية "RESET" بتحويل حالة RLO إلى "0" ونقلها إلى <Address>

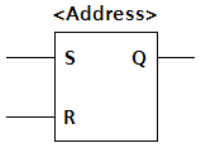
Set Before Reset Operations

الصيغة العامة

S <Address>
R <Address>
<Address> →I,Q,M,L,DB

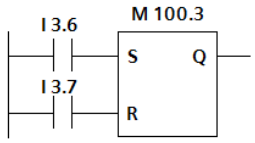
Example:
A I10.3
S Q12.4
A I10.4
R Q12.4

الصيغة العامة

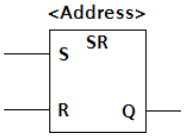


<Address> →I,Q,M,L,DB

Example

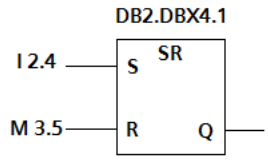


الصيغة العامة



<Address> →I,Q,M,L,DB

Example:



تقوم العملية SR إما بتحويل الحالة إلى "0" أو إلى "1" مع الاستمرار وفي حالة تواجد الشرطين R,S معا تكون النتيجة "0" ولا تكون الحالة "1" إلا إذا كان S حالته "1" و R حالته "0"

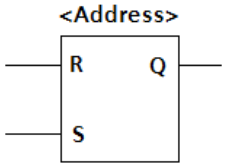
Reset Before Set Operations

الصيغة العامة

R <Address>
S <Address>
<Address> →I,Q,M,L,DB

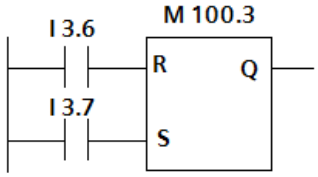
Example:
A I10.3
R Q12.4
A I10.4
S Q12.4

الصيغة العامة

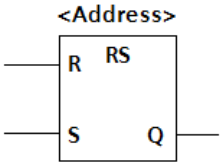


<Address> →I,Q,M,L,DB

Example

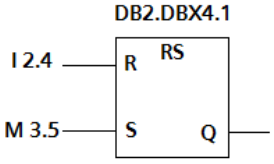


الصيغة العامة



<Address> →I,Q,M,L,DB

Example:



تقوم العملية RS إما بتحويل الحالة إلى "0" أو إلى "1" مع الاستمرار وفي حالة تواجد الشرطين R,S معا تكون النتيجة "1" و تكون الحالة "1" إذا كان S حالته "1" مهما كانت حالة R

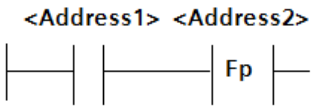
Positive edge Operation

الصيغة العامة

A <Address1>
Fp <Address2>
<Address1> →I,Q,M,L,DB,C,T
<Address2> →I,Q,M,L,DB

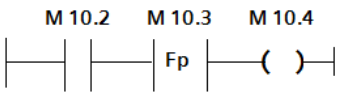
Example:
A M10.2
Fp M10.3
= M12.4

الصيغة العامة

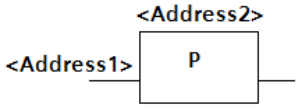


<Address1> →I,Q,M,L,DB,C,T
<Address2>→I,Q,M,L,DB

Example

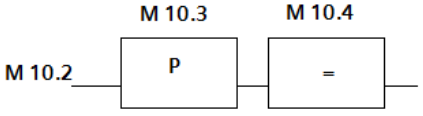


الصيغة العامة



<Address1> →I,Q,M,L,DB,C,T
<Address2>→I,Q,M,L,DB

Example:



تقوم العملية Fp بقراءة التحول من "0" إلى "1" على المتغير <Address1> وإخراج نبضة في حالة التغير إلى RLO مع الاحتفاظ بالقيمة القديمة قبل التغيير في <Address2>

88

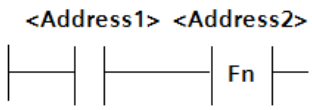
Negative edge Operation

الصيغة العامة

A <Address1>
Fn <Address2>
<Address1> →I,Q,M,L,DB,C,T
<Address2> →I,Q,M,L,DB

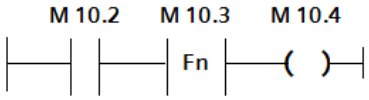
Example:
A M10.2
Fn M10.3
= M12.4

الصيغة العامة

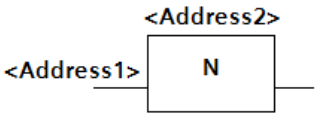


<Address1> →I,Q,M,L,DB,C,T
<Address2>→I,Q,M,L,DB

Example

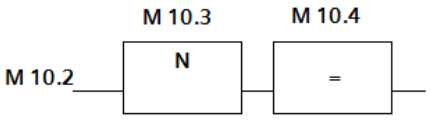


الصيغة العامة



<Address1> →I,Q,M,L,DB,C,T
<Address2>→I,Q,M,L,DB

Example:



تقوم العملية Fn بقراءة التحول من "1" إلى "0" على المتغير <Address1> وإخراج نبضة في حالة التغير إلى RLO مع الاحتفاظ بالقيمة القديمة قبل التغيير في <Address2>

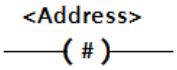
Mid-ASSIGN Operation

الصيغة العامة

= <Address>
A <Address>
<Address> →I,Q,M,L,DB

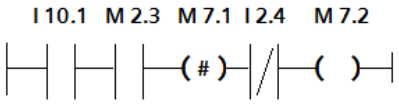
Example:
A I10.3
A M3.7
A DB30.DBX2.7
= L0.3
= Q12.4

الصيغة العامة

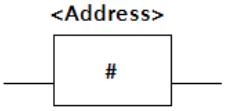


<Address> →I,Q,M,L,DB

Example

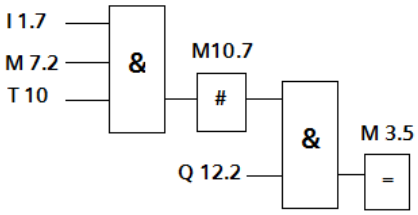


الصيغة العامة



<Address> →I,Q,M,L,DB

Example:



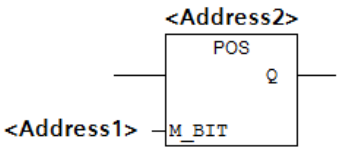
تقوم العملية "(#)" بتخزين ناتج العمليات قبلها في <Address> مع الاحتفاظ بها واستخدامها في العمليات التالية وتستخدم في العمليات المرحلية

POS Operation

الصيغة العامة

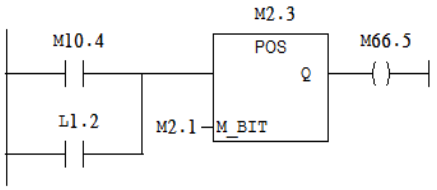
Same as Fp Operation

الصيغة العامة

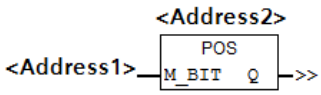


<Address1> →I,Q,M,L,DB,C,T
<Address2> →I,Q,M,L,DB

Example

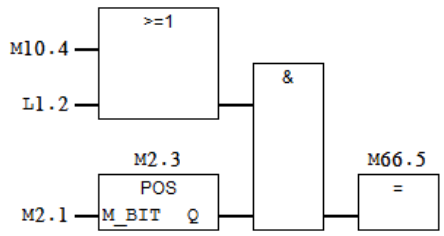


الصيغة العامة



<Address1> →I,Q,M,L,DB,C,T
<Address2> →I,Q,M,L,DB

Example:



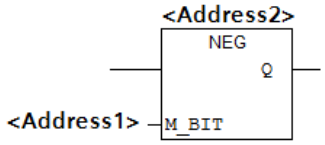
تقوم العملية " POS" بمراقبة التغير في حالة <Address1> من "0" إلى "1" وإخراج نبضة Pulse في RLO والاحتفاظ بالحالة في <Address2>

NEG Operation

الصيغة العامة

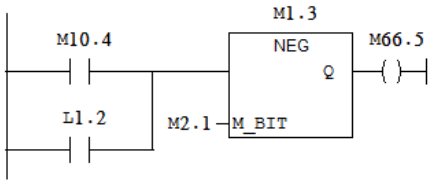
Same as Fp Operation

الصيغة العامة

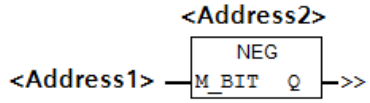


<Address1> →I,Q,M,L,DB,C,T
<Address2> →I,Q,M,L,DB

Example

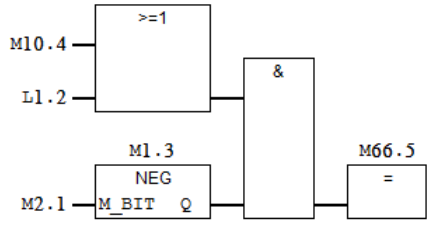


الصيغة العامة



<Address1> →I,Q,M,L,DB,C,T
<Address2> →I,Q,M,L,DB

Example:



تقوم العملية "NEG" بمراقبة التغير في حالة <Address1> من "1" إلى "0" وإخراج نبضة Pulse في RLO والاحتفاظ بالحالة في <Address2>

Nesting XOR Operation

الصيغة العامة

X(
X <Address1>
X <Address2>
)
X(
X <Address3>
X <Address4>
)

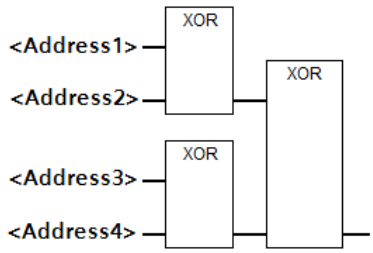
Example

X(
X I 1.2
X I 1.3
)
X(
X M 10.2
X M 10.5
)
= M 20.1

الصيغة العامة

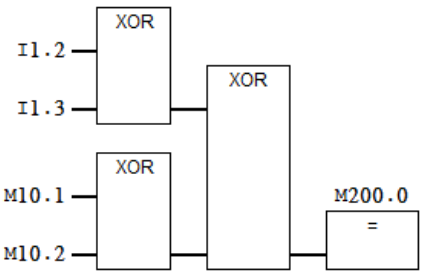
العملية غير موجودة ويتم تمثيلها بالعمليات الأساسية

الصيغة العامة



<Address-i> →I,Q,M,L,DB,C,T

Example:

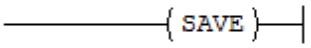


تقوم العملية "X(" بتنفيذ عمليات XOR المتداخلة من الداخل إلى الخارج حيث يتم تنفيذ داخل الأقواس أولا ويجب مراعاة توزيع الأقواس

وينطبق نفس الكلام تماما على العمليات NOT-XOR وبنفس الطريقة

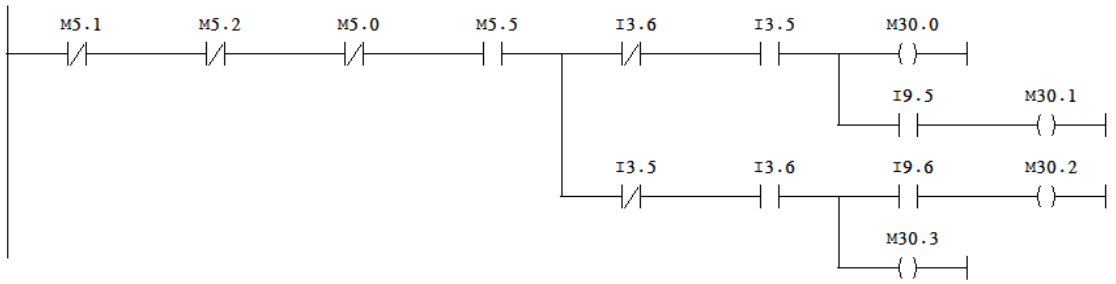
90

SET Operation		
الصيغة العامة	**الصيغة العامة**	**الصيغة العامة**
العملية غير موجودة	العملية غير موجودة	SET **Example** SET = Q 1.2
تقوم العملية "SET" بتحويل حالة RLO إلى "1"		

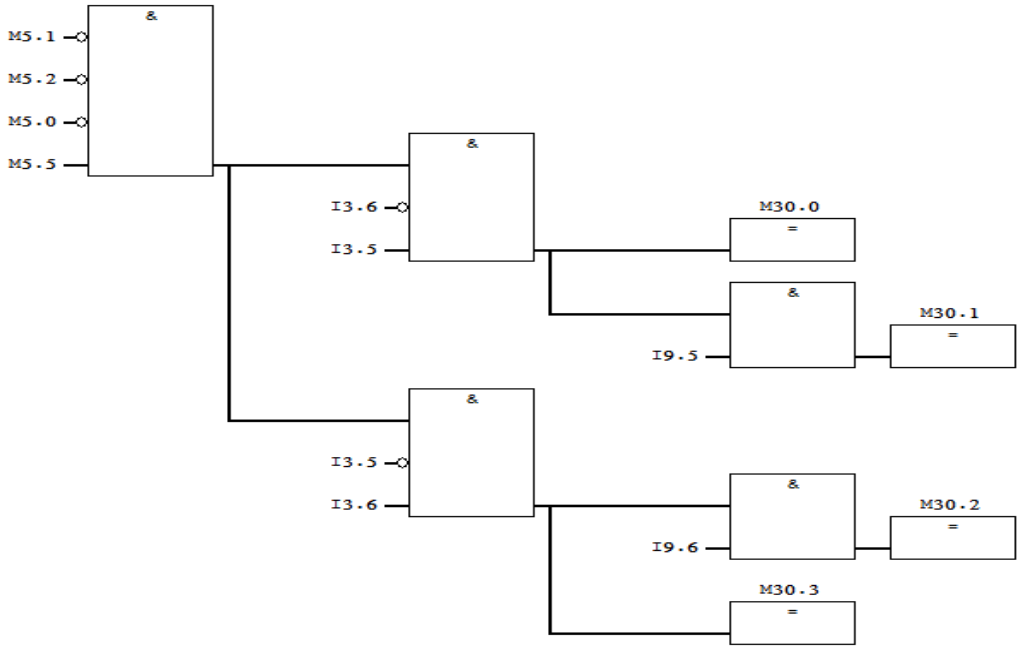
SAVE Operation		
الصيغة العامة	**الصيغة العامة**	**الصيغة العامة**
العملية غير موجودة		SAVE **Example** SAVE
تقوم العملية "SAVE" بتخزين حالة RLO في خانة BR		

لاحظ أنه يمكن التحويل بين الثلاث طرق للبرمجة لو التزمنا بالطريقة المثالية للكتابة ولكن هناك بعض العمليات الغير متاحة في كل من LAD أو FBD ويتم تكوينها باستخدام STL مع العلم بأنه يمكن كتابة كل شئ باستخدام STL

والعمليات الفعلية التي نستخدمها في دوائر التحكم مزيج من العمليات السابقة وليست بسيطة بالشكل الذي عرضناه فهي تحتوي على خليط من كل الأشكال مثل الشكل التالي والذي يمثل صورة فعلية من أحد البرامج




```
AN    M     5.1
AN    M     5.2
AN    M     5.0
A     M     5.5
=     L    12.0
A     L    12.0
AN    I     3.6
A     I     3.5
=     L    12.1
A     L    12.1
BLD   102
=     M    30.0
A     L    12.1
A     I     9.5
=     M    30.1
A     L    12.0
AN    I     3.5
A     I     3.6
=     L    12.1
A     L    12.1
A     I     9.6
=     M    30.2
A     L    12.1
BLD   102
=     M    30.3
```



بعد أن تعرضنا للعمليات المنطقية والتي تتم على مستوى Bit فإن هناك أشكال أخرى للبيانات تخضع للأنواع التي تحدثنا عنا من قبل سواء من حيث الحجم والذي يمكن أن يكون Byte أو Word أو Double Word أو من حيث النوع سواء كان CHAR أو INT أو DINT أو REAL أو S5TIME أو TIME أو DATE أو TIME_OF_DAY

وهذه الأنواع من البيانات تتم معالجتها في ACCUMULATORS سواء رقم-1،2 في المتحكمات التي تحتوى على اثنين فقط أو على 1,2,3,4 في المتحكمات التي تحتوي على 4 وحدات ACCUMULATOR وبالتالي فيجب تحميل البيانات في ACCUMULATORS قبل إجراء العمليات عليها من عناوين الذاكرة التي تحتوي على تلك البيانات كما يجب نقل تلك البيانات إلى عناوين الذاكرة بعد انتهاء المعالجة عليها وعملية النقل هذه تتم باستخدام أمرين هما LOAD/TRANSFER

عمليات النقل LOAD/TRANSFER:

عمليات النقل LOAD/TRANSFER غير مدعومة في كل من LAD و FBD حيث تتم بشكل ضمني في العمليات المختلفة

Statement List (STL)

LOAD Instruction (L)

الصيغة العامة

L <Address>
<Address> →I,Q,M,L,T,C,DB,P (Byte/Word/DWord)

تفاصيل التحميل	**Example**:
- تحميل بايت الدخول رقم-20 إلى ACCU1 في البايت الأدنى (أول 8 خانات)	L IB 20
- تحميل وورد الخروج رقم-14 إلى ACCU1 في الوورد الأدنى (أول 16 خانة)	L QW 14
- تحميل وورد مضاعف الذاكرة العامة رقم-200 في ACCU1 (كله 32 خانة)	L MD 200
- تحميل بايت الذاكرة المحلية رقم-0 في ACCU1 في البايت الأدنى (أول 8 خانات)	L LB 0
- تحميل وورد رقم-20 في بلوك بيانات رقم-10 إلى ACCU1 الوورد الأدنى (أول 16 خانة)	L DB10.DBW20
- تحميل القيمة الحالية للعداد رقح-10 إلى ACCU1 في الوورد الأدنى (أول 16 خانة)	L C10
- تحميل القيمة الحالية للمؤقت رقم-1 إلى ACCU1 في الوورد الأدنى (أول 16 خانة)	L T1
- تحميل الدخل التناظري رقم-340 إلى ACCU1 في الوورد الأدني (أول 16 خانة)	L PIW 340
- تحميل دخل التردد رقم-700 إلى ACCU1 (كله 32 خانة)	L PID 700

تقوم العملية LOAD (L) بنقل محتويات ACCU3 إلى ACCU4 ونقل محتويات ACCU2 إلى ACCU3 ونقل محتويات ACCU1 إلى ACCU2 ونقل محتويات <Address> إلى ACCU1 مع فقد محتويات ACCU4 القديمة ولا تؤثر هذا العملية أو تتأثر بنتيجة RLO مطلقا فإذا مر عليها التنفيذ فسوف يتم تنفيذها بغض النظر عن قيمة RLO

هناك عمليات تحميل خاصة أخرى تستعمل الأمر Load(L) لتحميل محتويات خاصة في البرنامج مثل مسجل الحالة STATUS-Register والذي يتكون من خانات الحالة STATUS Bits والتي تحدثنا عنها من قبل وهي على الترتيب :
BR – CC0 – CC1 –OV – OS – OR – STA – RLO – /FC
وتستخدم لذلك الصيغة : L STW

كما يمكن استخدام الأمر Load لتحميل محتويات مسجل العناوين Address register-1,2 حيث يستخدمان في التحكم في عنوان الأمر الحالي للتنفيذ وعنوان الأمر القادم للتنفيذ فيمكن تحميل تلك العناوين إلى ACCU1 أو إلى موقع من

92

مواقع الذاكرة المختلفة بحجم 32 خانة أول تحميل مسجل العناوين رقم-1 من 2 وتستخدم لذلك الصيغ التالية:

- تحميل محتويات مسجل العناوين رقم-1 إلى ACCU1 ونستخدم لذلك : L AR1
- تحميل محتويات مسجل العناوين رقم-2 إلى ACCU1 ونستخدم لذلك : L AR2
- تحميل محتويات مسجل العناوين رقم-1 إلى موقع ذاكرة ونستخدم لذلك : L AR1 <Address> حيث <Address> هو مكان الذاكرة حيث يتم تحميل محتويات AR1
- تحميل محتويات مسجل العناوين رقم-2 إلى موقع ذاكرة ونستخدم لذلك : L AR2 <Address> حيث <Address> هو مكان الذاكرة حيث يتم تحميل محتويات AR2
- تحميل محتويات مسجل الذاكرة رقم-1 من مسجل الذاكرة رقم-2 ونستخدم لذلك الأمر L AR1 AR2

كما يمكن استخدام الأمر Load لتحميل قيمة ثابتة Constant في ACCU1 كما في الأمثلة الآتية:

L 120 // تحميل قيمة 120 صحيحة في ACCU1

L L#200 // تحميل قيمة 200 صحيحة مضاعفة فيACCU1

L 100.0 // تحميل قيمة 100.0 عشرية فيACCU1

L S5T#200MS // تحميل قيمة 200 مللي ثانية بصيغة الوقت فيACCU1

L W#16#FFFF // تحميل قيمة FFFF بصيغة سداسي عشر في ACCU1

L 2#1110_0001_0010_0001 // تحميل قيمة ثنائية 1110_0001_0010_0001 في ACCU1

L 'A' // تحميل الرمز 'A' في شكل كود في ACCU1

Transfer Instruction (T)

الصيغة العامة

T <Address>

<Address> →I,Q,M,L,DB,PQ (Byte/Word/DWord)

93

تفاصيل التحميل

- تحميل بايت الدخول رقم-20 من ACCU1 في البايت الأدنى (أول 8 خانات)
- تحميل وورد الخروج رقم-14 من ACCU1 في الوورد الأدنى (أول 16 خانة)
- تحميل وورد مضاعف الذاكرة العامة رقم-200 من ACCU1 (كله 32 خانة)
- تحميل بايت الذاكرة المحلية رقم-0 في ACCU1 من البايت الأدنى (أول 8 خانات)
- تحميل وورد رقم-20 في بلوك بيانات رقم-10 من ACCU1 الوورد الأدنى (أول 16 خانة)
- تحميل الخرج التناظري رقم-340 إلى ACCU1 من الوورد الأدني (أول 16 خانة)
- تحميل خرج التردد رقم-700 من ACCU1 (كله 32 خانة)

Example:

T IB 20

T QW 14

T MD 200

T LB 0

T DB10.DBW20

T PQW 340

T PQD 700

يقوم الأمر <Address> Transfer T بنقل محتويات ACCU1 إلى عنوان الذاكرة <Address> حسب حجمها (بايت أو وورد أو وورد مضاعفة) مع عدم تغيير محتويات أي من ACCU1,ACCU2,ACCU3,ACCU4

هناك عمليات نقل خاصة أخرى تستعمل الأمر Transfer (T) لتحميل محتويات خاصة في البرنامج مثل مسجل الحالة STATUS-Register والذي يتكون من خانات الحالة STATUS Bits والتي تحدثنا عنها من قبل وهي على الترتيب :

BR – CC0 – CC1 –OV – OS – OR – STA – RLO – /FC

وتستخدم لذلك الصيغة : T STW

كما يمكن استخدام الأمر Transfer لنقل محتويات ACCU1 إلى مسجل العناوين Address register-1,2 حيث يستخدمان في التحكم في عنوان الأمر الحالي للتنفيذ وعنوان الأمر القادم للتنفيذ فيمكن تحميل تلك العناوين من ACCU1 أو من موقع من مواقع الذاكرة المختلفة بحجم 32 خانة أول تحميل مسجل العناوين رقم-1 من 2 وتستخدم لذلك الصيغ التالية:

- تحميل محتويات مسجل العناوين رقم-1 من ACCU1 ونستخدم لذلك : T AR1
- تحميل محتويات مسجل العناوين رقم-2 من ACCU1 ونستخدم لذلك : T AR2
- تحميل محتويات مسجل العناوين رقم-1 من موقع ذاكرة ونستخدم لذلك : T AR1 <Address> حيث <Address> هو مكان الذاكرة حيث يتم تحميل محتويات AR1
- تحميل محتويات مسجل العناوين رقم-2 من موقع ذاكرة ونستخدم لذلك : T AR2 <Address> حيث <Address> هو مكان الذاكرة حيث يتم تحميل محتويات AR2
- تحميل محتويات مسجل الذاكرة رقم-1 إلى مسجل الذاكرة رقم-2 ونستخدم لذلك الأمر T AR1 AR2

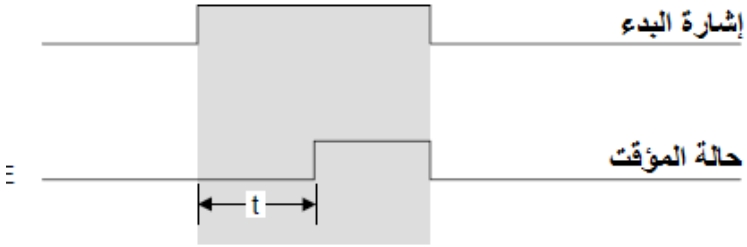
ويستخدم الأمر CAR لتبادل محتويات مسجلي الذاكرة AR1,AR2 فيضع محتويات AR1 في AR2 والعكس

تعليمات عمل المؤقتات Timer Instructions:

المؤقتات بشكل عام لها خمسة أشكال من التشغيل هي الأشهر والتي ترتبط بالعوامل الأساسية لتشغيل المؤقت Timer وهي إشارة البدء وقيمة وقت المؤقت الأولية وقيمة الوقت الحالية وإشارة التصفير (الإعادة لنقطة البداية) وحالة المؤقت وهذه الأشكال الخمسة على النحو التالي :

1- مؤقت تأخير بدء ON-Delay timer

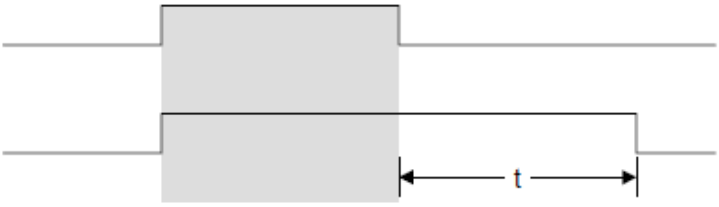
وهنا يتم تغيير حالة المؤقت State من "0" إلى "1" بعد وقت محدد "t" من تحول إشارة البدء من الحالة "0" إلى الحالة "1" وتظل موجودة على الحالة "1" حتى تتحول إشارة البدء إلى الحالة "0" أو يتم تصفير أو عمل Reset للمؤقت فيتحول إلى الحالة "0"



مؤقت تأخير بدء ON-Delay timer

2- مؤقت تأخير فصل OFF-Delay timer

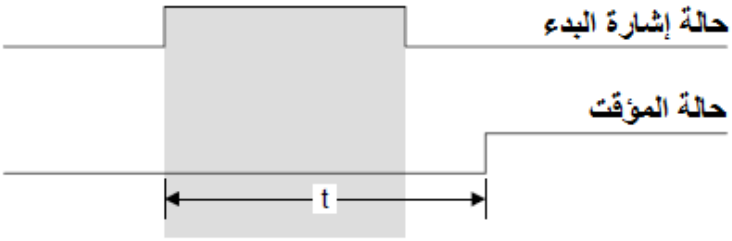
وهنا يتم تغيير حالة المؤقت من "0" إلى "1" مع إشارة البدء وتبقى على الحالة "1" طالما إشارة البدء موجودة أو لم يتم تصفير المؤقت وعند تحول إشارة البدء من الحالة "1" إلى الحالة "0" وبعد وقت "t" من تحول إشارة البدء تتحول حالة المؤقت من "0" إلى "1"



مؤقت تأخير فصل OFF-Delay timer

3- مؤقت تأخير بدء مخزن Retentive (Stored) ON-Delay timer

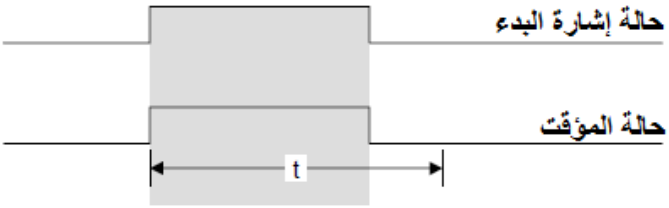
وهنا يتم تغيير حالة المؤقت من "0" إلى "1" بعد وقت "t" من تحول حالة إشارة البدء من "0" إلى "1" وتبقى على حالتها حتى لو تحولت إشارة البدء إلى "0" ولا تتحول إلى الحالة "0" إلا بتصفير المؤقت Reset



Retentive (Stored) ON-Delay timer
مؤقت تأخير بدء مخزن

4- مؤقت نبضي Pulse timer

وهنا يتم تغيير حالة المؤقت من "0" إلى "1" مع تغير حالة إشارة البدء من "0" إلى "1" وتنتهي النبضة إما بتحول حالة إشارة البدء من "1" إلى "0" قبل انتهاء الوقت "t" أو بانتهاء الوقت في حالة استمرار إشارة البدء على "1" منذ البداية أو بعمل تصفير Reset للمؤقت

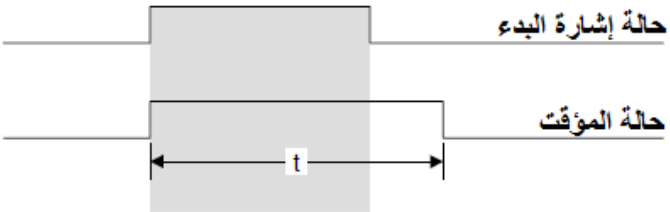


Pulse timer مؤقت نبضي

95

5- مؤقت نبضي ممتد Extended pulse time

وهنا يتم تغيير حالة المؤقت من "0" إلى "1" مع تغير حالة إشارة البدء من "0" إلى "1" وتنتهي النبضة بتحول حالة المؤقت إلى "0" بعد وقت محدد "t" أو بعمل تصفير للمؤقت Reset حتى في حالة تحول إشارة البدء إلى "0" والذي لا يؤثر على حالة المؤقت

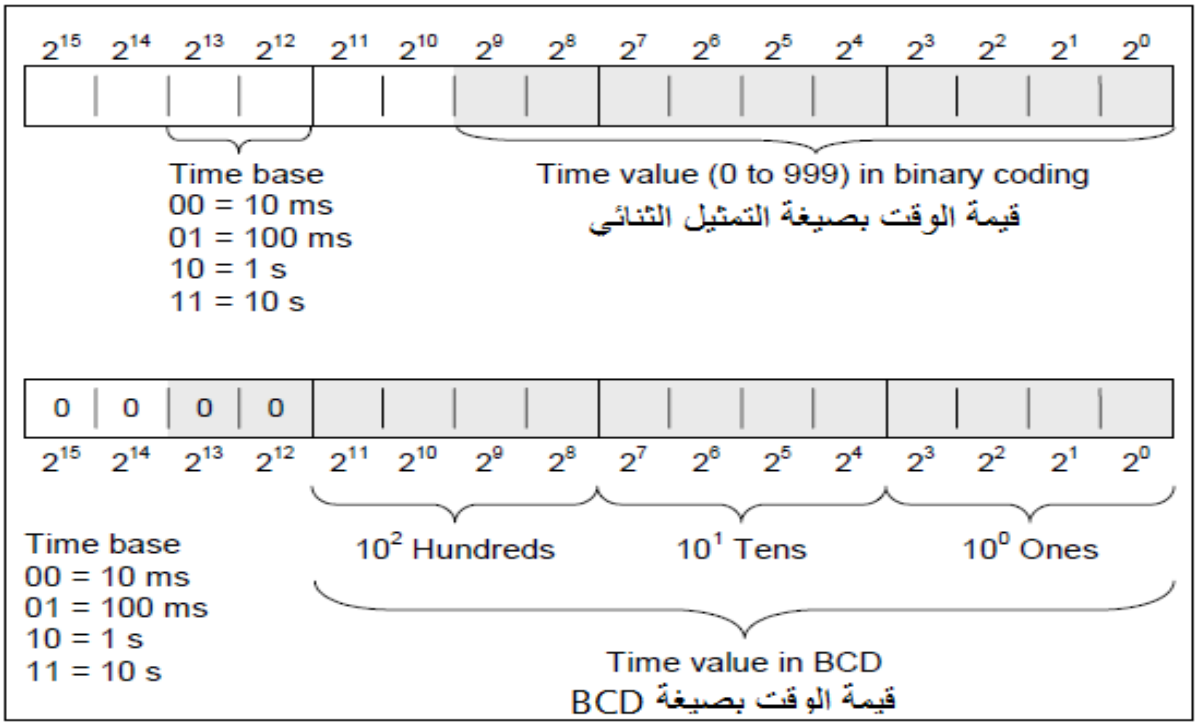


Extended pulse timer مؤقت نبضي ممتد

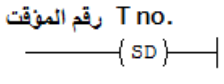
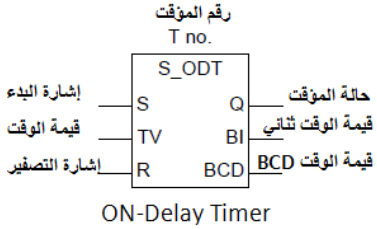
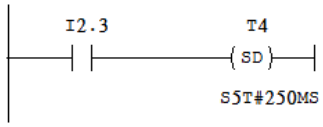
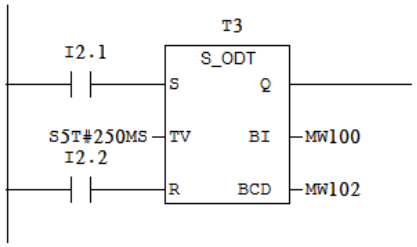
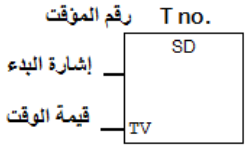
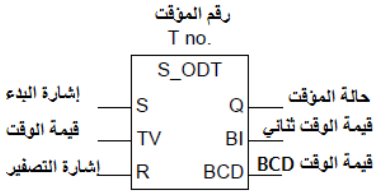
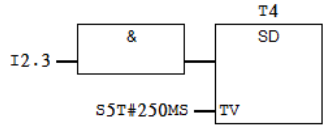
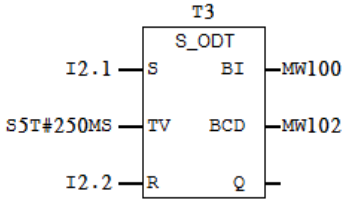
وسوف نستعرض معا في الأسطر التالية تعليمات Step-7 لتمثيل تلك الأنواع من المؤقتات في البرامج

وأول مرحلة لتجهيز المؤقت للعمل هي تحميل قيمة الوقت "t" حيث يتم تحميل قيمة الوقت باستخدام الأمر Load في ACCU1 عند البرمجة باستخدام STL أما في البرمجة باستخدام LAD أو FBD فيتم إدخال قيمة الوقت مباشرة كجزء من أمر البرمجة

ويتم تحميل قيمة الوقت بالأمر Load إما في شكل عنوان ذاكرة يحتوي على قيمة الوقت أو كثابت في صيغة S5TIME أو قيمة عند تحويلها إلى صيغة الوقت تأخذ القيمة المطلوبة للوقت وهي حسب الشكل التالي :



وبالتالي فمن المهم جدا التأكد بتحميل قيمة الوقت في ACCU1 قبل تشغيل المؤقت مثل: L MW200 ، L S5T#200MS ، L S5T#1M20S

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
ON-Delay Timer (SD)		
الصيغة العامة SD <Tn> n → Timer number 1,2,.. etc **Example** L S5T#250MS A I 2.3 SD T4 L S5T#250MS A I 2.1 SD T3 A I 2.2 R T3	**الصيغة العامة**   ON-Delay Timer **Example**  	**الصيغة العامة**   ON-Delay Timer **Example**  

OFF-Delay Timer (SF)

الصيغة العامة

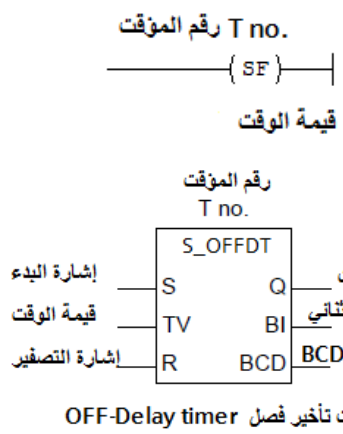
SF <Tn>
n → Timer number 1,2,.. etc

Example

L S5T#250MS
A I 2.3
SF T4

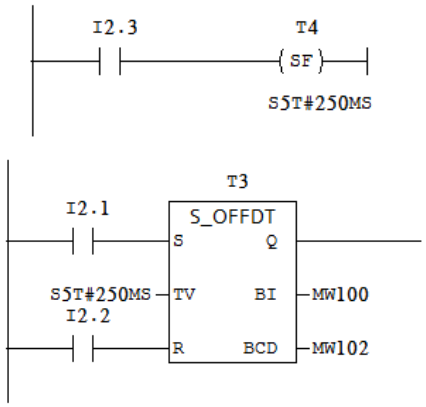
L S5T#250MS
A I 2.1
SF T3
A I 2.2
R T3

الصيغة العامة

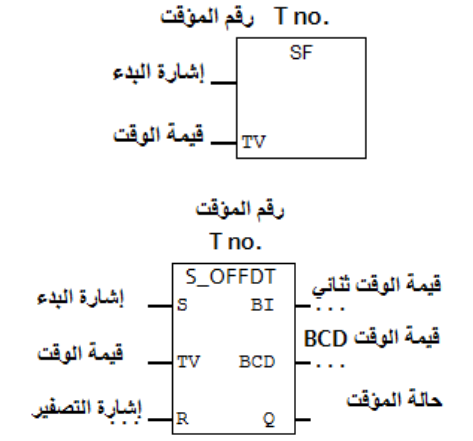


مؤقت تأخير فصل OFF-Delay timer

Example

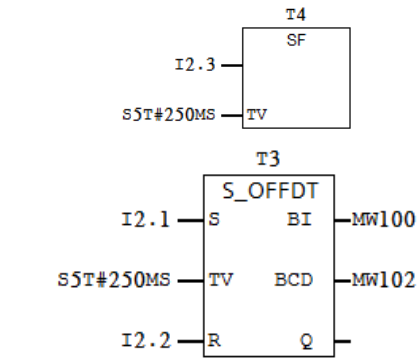


الصيغة العامة



مؤقت تأخير فصل OFF-Delay timer

Example



97

Retentive ON-Delay Timer (SS)

الصيغة العامة

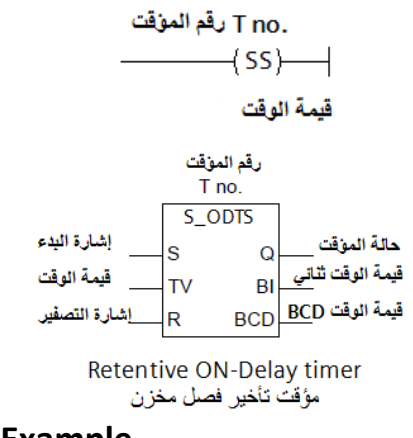
SS <Tn>
n → Timer number 1,2,.. etc

Example

L S5T#250MS
A I 2.3
SS T4

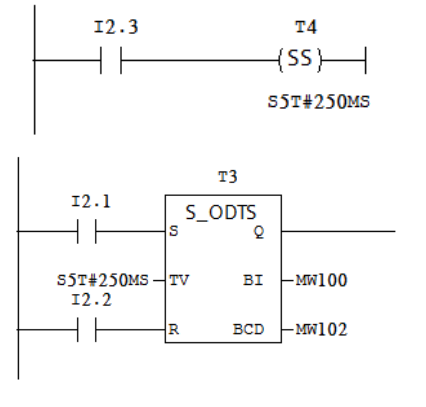
L S5T#250MS
A I 2.1
SS T3
A I 2.2
R T3

الصيغة العامة

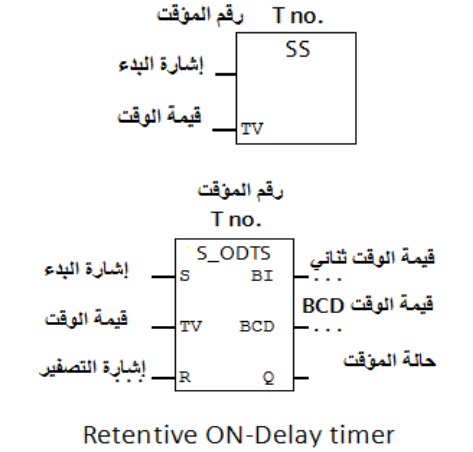


Retentive ON-Delay timer
مؤقت تأخير فصل مخزن

Example

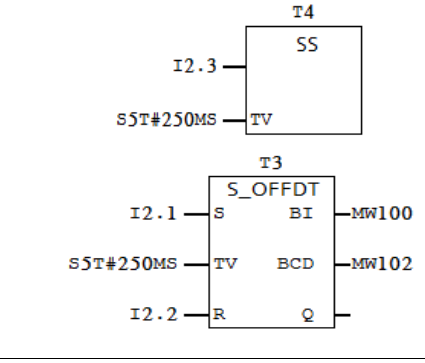


الصيغة العامة



Retentive ON-Delay timer
مؤقت تأخير فصل مخزن

Example



Pulse Timer (SP)

الصيغة العامة

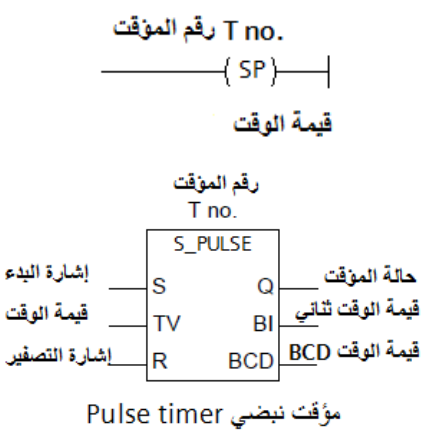
SP <Tn>
n → Timer number 1,2,.. etc

Example

L S5T#250MS
A I 2.3
SP T4

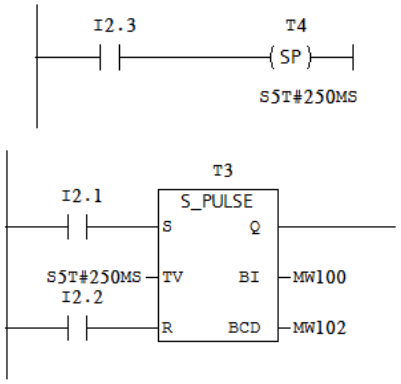
L S5T#250MS
A I 2.1
SP T3
A I 2.2
R T3

الصيغة العامة

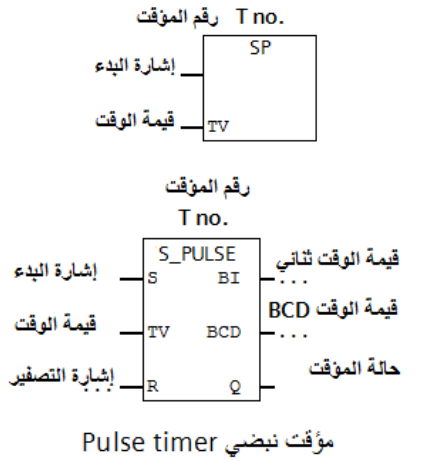


مؤقت نبضي Pulse timer

Example

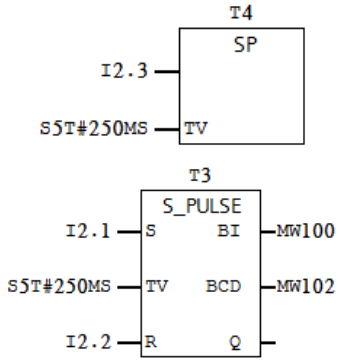


الصيغة العامة



مؤقت نبضي Pulse timer

Example



98

Extended Pulse Timer (SE)

الصيغة العامة

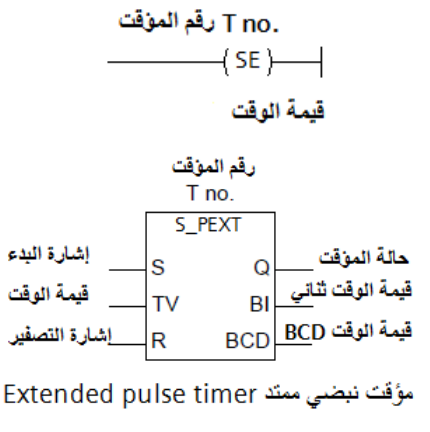
SE <Tn>
n → Timer number 1,2,.. etc

Example

L S5T#250MS
A I 2.3
SE T4

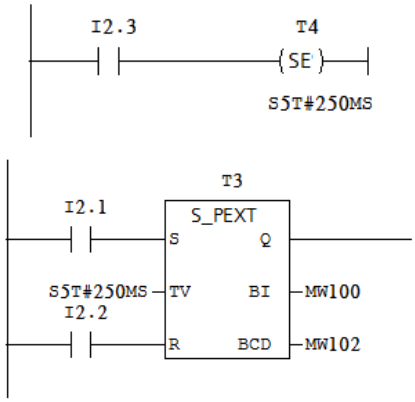
L S5T#250MS
A I 2.1
SE T3
A I 2.2
R T3

الصيغة العامة

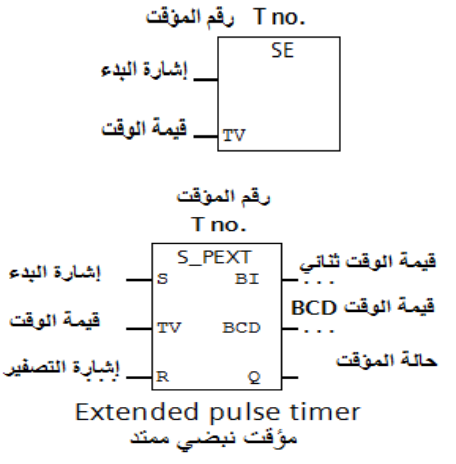


مؤقت نبضي ممتد Extended pulse timer

Example

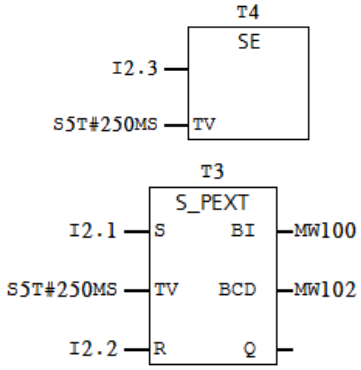


الصيغة العامة

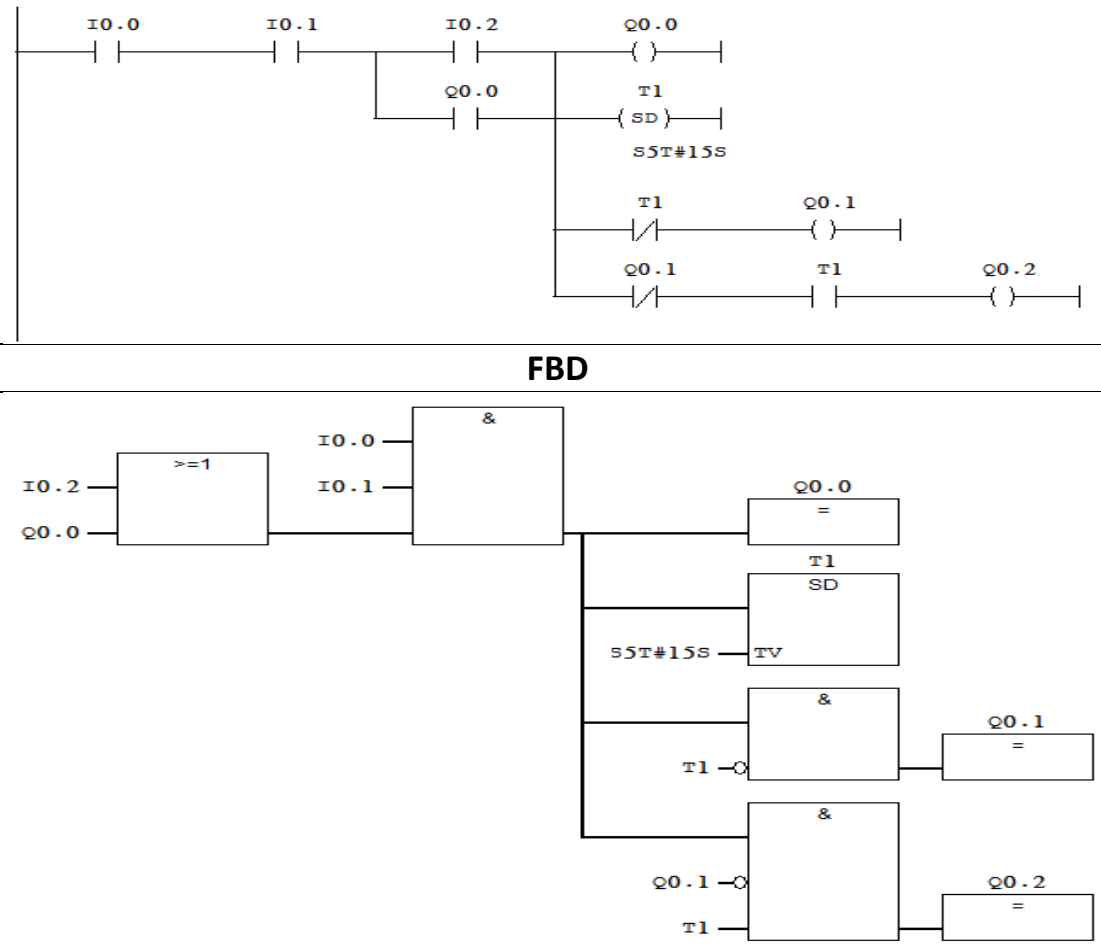


Extended pulse timer
مؤقت نبضي ممتد

Example



مثال: استعمال مؤقت تأخير بدء في دائرة تشغيل محرك ستار دلتا
تشغيل I0.2 ، إيقاف I0.0 ، أوفرلود I0.1 ، كونتاكتور رئيسي Q 0.0 ، كونتاكتور ستار Q 0.1
كونتاكتور دلتا Q 0.2 ، المؤقت المستخدم T1 بوقت 15 ثانية

STL	LAD / FBD
A I 0.0 A I 0.1 A(O I 0.2 O Q 0.0) = Q 0.0 A Q 0.0 L S5T#15S SD T1 A Q 0.0 AN T1 = Q 0.1 A Q 0.0 AN Q 0.1 A T1 = Q 0.2	

Enable timer (FR)

الصيغة العامة	الصيغة العامة	الصيغة العامة
الأمر FR غير مدعوم في FBD	الأمر FR غير مدعوم في STL	FR <Tn> n → Timer number 1,2,.. etc **Example** A I 2.0 FR T3 L S5T#250MS A I 2.1 SE T3 A I 2.2 R T3

الأمر FR يستخدم لفعيل تشغيل المؤقت وفي المعتاد لا يتم استخدامه إلا في حالة واحدة وهي إعادة تشغيل المؤقت في حالة عدم تغير حالة إشارة بدء التشغيل حيث يتم تجديد وقت المؤقت مع الاحتفاظ بتشغيله عكس Reset والذي يقوم بتصفير المؤقت وعدم تشغيله

99

Loading actual time value (L,LC)		
الصيغة العامة الأمران L,LC غير مدعومان ويتم تنفيذهما ذاتيا مع استدعاء المؤقت حيث يوجد مخرج للقيمة الصحيحة ومخرج للقيمة BCD بشكل مباشر على المؤقت	**الصيغة العامة** الأمران L,LC غير مدعومان ويتم تنفيذهما ذاتيا مع استدعاء المؤقت حيث يوجد مخرج للقيمة الصحيحة ومخرج للقيمة BCD بشكل مباشر على المؤقت	**الصيغة العامة** L <Tn> LC <Tn> n → Timer number 1,2,.. etc **Example** L T3 LC T3
- لتحميل قيمة الوقت الحالية في ACCU1 على صورة رقم صحيح integer نستخدم الأمر L العادي - لتحميل قيمة الوقت الحالية في ACCU1 على صورة BCD نستخدم الأمر LC		

أمثلة مطلوب تنفيذها :

- محركان يعملان بالتبادل حيث يعمل الأول لمدة 10 دقائق والثاني بالتبادل 5 دقائق ويبدأ التشغيل بمفتاح تشغيل ويتوقف التشغيل بمفتاح إيقاف مع وجود حماية أوفرلود للمحركين
- مطلوب ترجمة الضغط على مفتاح بأزمنة مختلفة لمدة أقل من ثانية ولمدة أقل من 5 ثواني ولمدة أكبر من 10 ثواني بحيث نحصل على ثلاث ذواكر M x.0, M x.1, M x.2 واحدة في كل حالة
- محرك تيار مستمر يتم فصل ملفات الفيلد بعد التوقف بخمس دقائق ويتم فصل مروحة التبريد بعد التوقف بعشر دقائق ما لم يتم إعادة التشغيل مرة أخرى

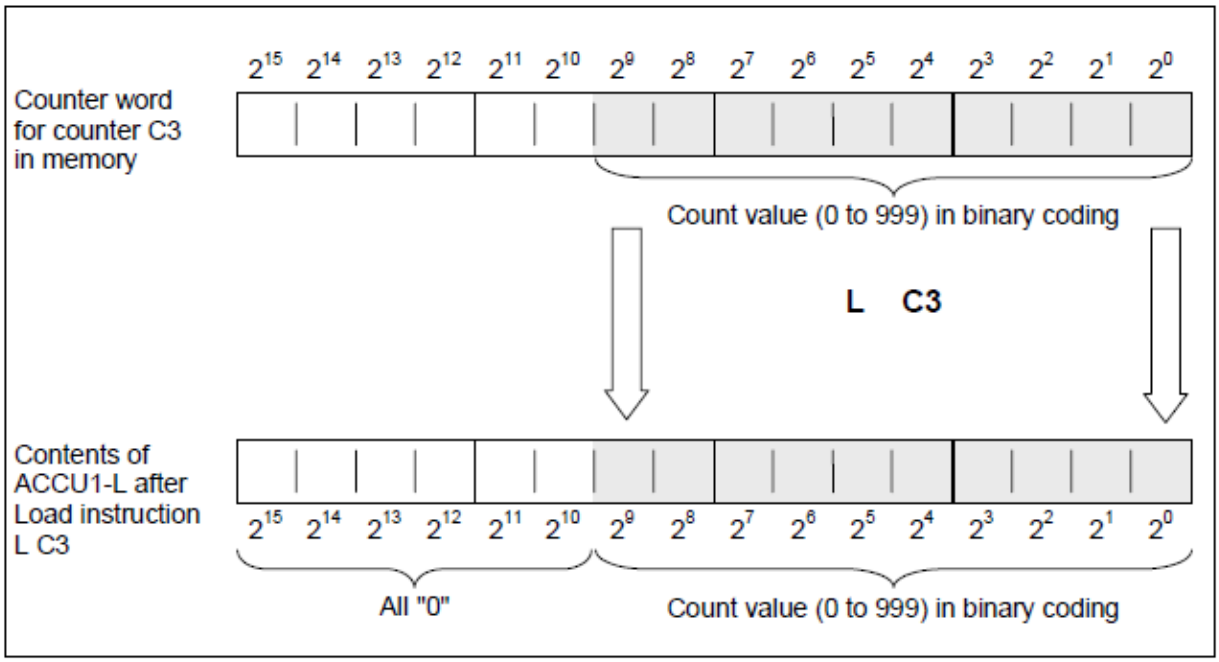
100

لاشك أن عمليات العد والعدادات تمثل جزء هام من وظائف التحكم سواء كانت عملية مستقلة أو تقنية من تقنيات التحكم التي يمكن استخدامها في البرنامج وهناك نوعان من العدادات في منظومة S7-300/400 وهما العدادات المعتادة Counters والعدادات فائقة السرعة High Speed Counters (HSC) والتي تختص وتدخل ضمن العمليات الخاصة ولها وحدات خاصة من النوع FM Modules حيث تستقبل الإشارات الترددية فائقة السرعة High frequency وتستخدم بلوكات خاصة من بلوكات النظام الموجودة في المكتبة Library وسوف نركز في هذا الجزء على العدادات المعتادة Counters ونرجئ الحديث عن العدادات فائقة السرعة للجزء الخاص بالبرمجة المتقدمة.

تعليمات العدادات Counter Instructions:

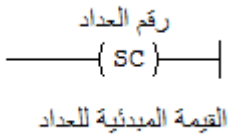
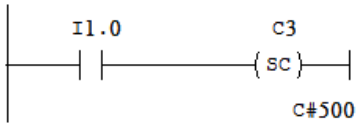
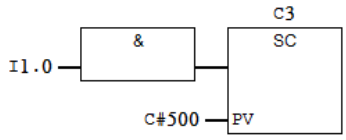
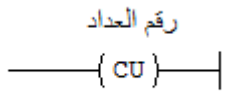
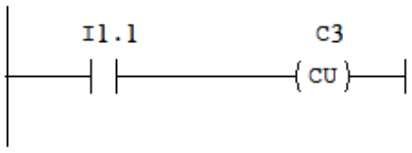
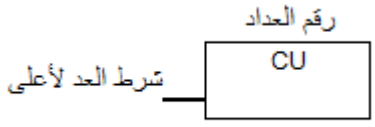
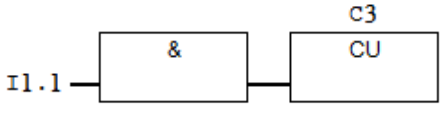
ونبدأ بالحديث أولا بشكل عام عن العدادات حيث:

- يتراوح العد بين القيمة "0" والقيمة "999" ولا يكون العد أقل من "0" ولا أكبر من "999"
- طالما قيمة العداد أكبر من "0" أي من "1" وحتى "999" فإن حالة العداد تكون "1" وتكون حالة العداد "0" فقط عندما تكون قيمة العد "0"
- لحظة العد أو انتقال قيمة العد للقيمة الأكبر أو القيمة الأقل مرتبط بلحظة تحول حالة شرط العد من "0" إلى "1"
- شكل تمثيل قيمة العد يتم تمثيلها بصيغة BCD وعند تحميل قيمة عداد للعداد لابد أن تكون بتلك الصيغة مثلا C#520 حتى لا يحدث خطأ يؤدي إلى توقف وحدة PLC



في الشكل يبين كيفية تمثيل القيمة الحالية للعداد في شكل Binary وليس BCD وكيف أنه في هذه الحالة لا يحتوي سوى على 10 خانات فقط والتي تكفي لتمثيل تلك القيمة في حين أن تلك القيمة بصيغة BCD والتي تتطلب 12 خانة وليس 10 خانات مثل التمثيل الثنائي

والعمليات التي يمكن تنفيذها في العدادات هي العد التصاعدي Count-up والعد التنازلي Count-down وتحميل قيمة مبدئية للعداد يتم العد من عندها Preset value وتصفير قيمة وحالة العداد Reset counter وتفعيل العداد Enable counter كما يمكن قراءة حالة العداد باستخدام العمليات المنطقية وتحميل القيمة الحالية للعداد في شكل تمثيل ثنائي و في شكل تمثيل BCD

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
Enable Counter (FR)		
الصيغة العامة FR <Cn> n → Counter number 0,1,2,.. etc **Example** A I 2.3 FR C4	**الصيغة العامة** الأمر FR غير مدعوم في LAD	**الصيغة العامة** الأمر FR غير مدعوم في FBD
يقوم الأمر FR بإعادة تفعيل العداد وليس ذا قيمة كبيرة ولا يؤثر على حالة العداد إلا في حالة كون شرط العد لأعلى أو لأسفل موجود فيقوم بتجديد Pulse حيث يمثل نقطة قراءة جديدة لحالة دخل العد		
Loading Preset Value (SC)		
الصيغة العامة S <Cn> n → Counter number 0,1,2,.. etc **Example** A I 1.0 L C#500 S C3	**الصيغة العامة**  **Example**  	**الصيغة العامة**  **Example** 
يقوم الأمر SC بتحميل القيمة الموجودة في ACCU1-LOW Word والتي من المفروض أن تكون في صورة C# في العداد كقيمة يبدأ من عندها العد ويمكن أن تكون ثابت أو تكون عنوان متغير يحتوي على قيمة بصيغة C#		
Count up (CU)		
الصيغة العامة CU <Cn> n → Counter number 0,1,2,.. etc **Example** A I 1.1 CU C3	**الصيغة العامة**  **Example**  	**الصيغة العامة**  **Example** 
يقوم الأمر CU عند تحول حالة RLO قبله مباشرة من الحالة "0" إلى الحالة "1" بزيادة قيمة العداد بمقدار "1" في كل مرة وعندما تكون قيمة العداد الحالية أكبر من "0" فإن حالة العداد تصبح "1" وعندما تصل القيمة إلى "0" تصبح حالة العداد "0" ، وعند وصول العداد إلى القيمة "999" لا ينتقل آليا إلى القيمة "0" ولا يزيد أكثر من ذلك حتى لو تغير شرط العد من "0" إلى "1"		

102

Count down (CD)

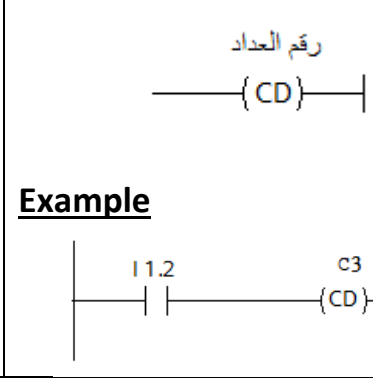
الصيغة العامة

CD <Cn>
n → Counter number 0,1,2,..
etc

Example

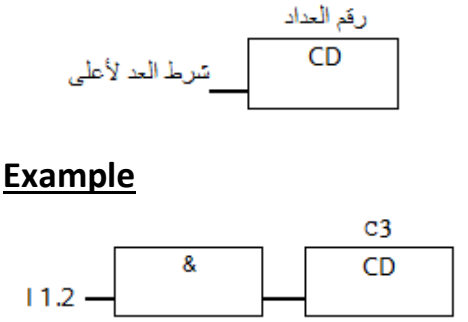
A I 1.2
CD C3

الصيغة العامة



Example

الصيغة العامة



Example

يقوم الأمر CD عند تحول حالة RLO قبله مباشرة من الحالة "0" إلى الحالة "1" بخفض قيمة العداد بمقدار "1" في كل مرة وعندما تكون قيمة العداد الحالية أكبر من "0" فإن حالة العداد تصبح "1" وعندما تصل القيمة إلى "0" تصبح حالة العداد "0" ، وعند وصول العداد إلى القيمة "0" لا يعد العداد بقيمة سالبة حتى لو تغير شرط العد من "0" إلى "1"

Reset Counter (R)

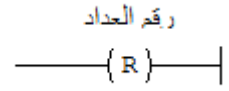
الصيغة العامة

R <Cn>
n → Counter number 0,1,2,..
etc

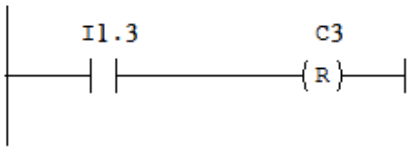
Example

A I .3
R C3

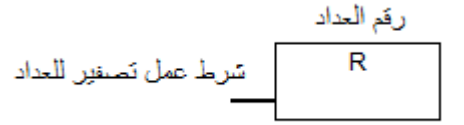
الصيغة العامة



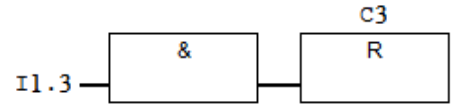
Example



الصيغة العامة



Example



يقوم الأمر Reset (R) عند تحول حالة RLO قبله مباشرة من الحالة "0" إلى الحالة "1" بتحويل قيمة العداد الحالية إلى القيمة "0" وتحويل حالة العداد إلى الحالة "0"

Up-Counter Block

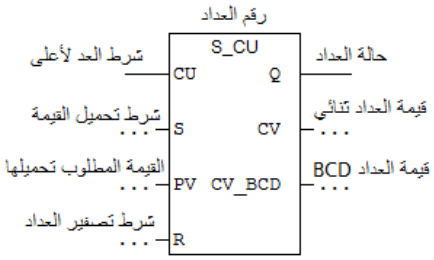
الصيغة العامة

CD <Cn>
.
S <Cn>
.
R <Cn>
n → Counter number 0,1,2,..
etc

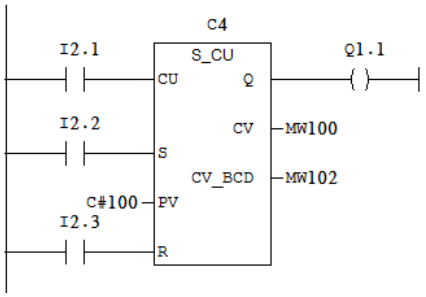
Example

L C#100
A I 2.1
CU C4
A I 2.2
S C4
A I 2.3
R C4

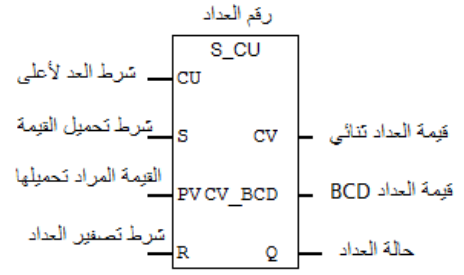
الصيغة العامة



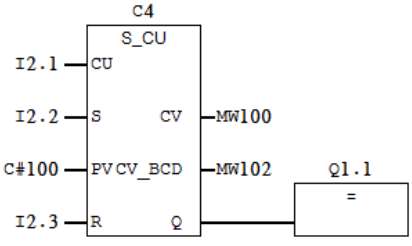
Example



الصيغة العامة



Example



هنا تتم ثلاث عمليات معا وهي العد لأعلى وتحميل القيمة وتصفير العداد إضافة إلى إخراج القيمة والحالة في بلوك واحد

Down-Counter Block

الصيغة العامة

CD <Cn>

.

S <Cn>

.

R <Cn>

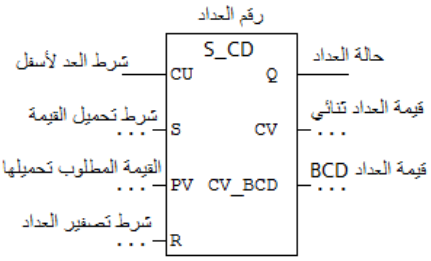
n → Counter number 0,1,2,..

etc

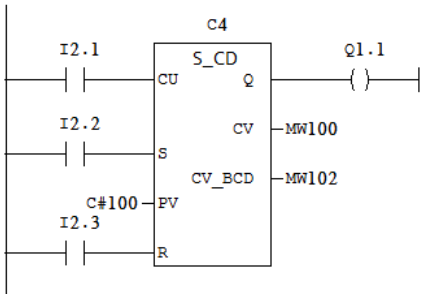
Example

```
L C#100
A I 2.1
CD C4
A I 2.2
S C4
A I 2.3
R C4
```

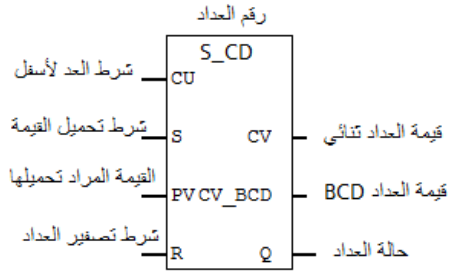
الصيغة العامة



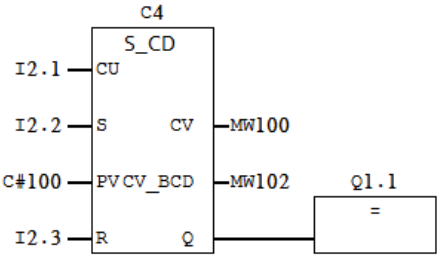
Example



الصيغة العامة



Example



هنا تتم ثلاث عمليات معا وهي العد لأسفل وتحميل القيمة وتصفير العداد إضافة إلى إخراج القيمة والحالة في بلوك واحد

Up-Down Counter

الصيغة العامة

CU <Cn>

.

CD <Cn>

.

S <Cn>

.

R <Cn>

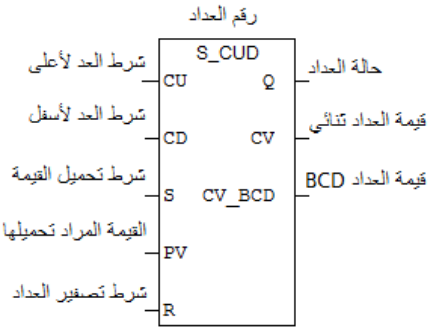
n → Counter number 0,1,2,..

etc

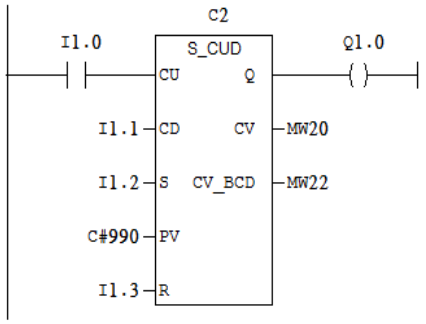
Example

```
L C#990
A I 1.0
CU C2
A I 1.1
CD C2
A I 1.2
R C2
A I 1.3
R C2
```

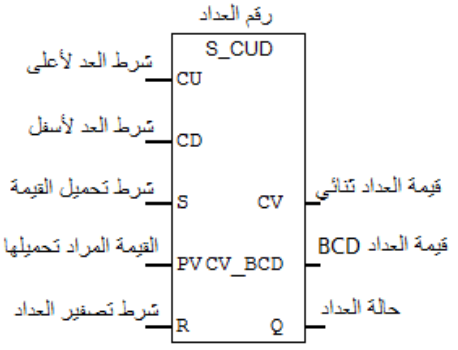
الصيغة العامة



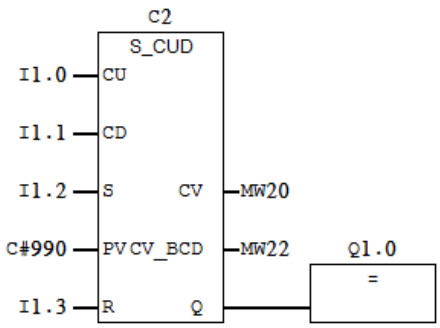
Example



الصيغة العامة



Example



هنا تتم كل العمليات الخاصة بالعدادات معا في نفس الوقت ، العد لأعلى ، العد لأسفل ، تحميل قيمة ، تصفير للعداد ، قراءة حالة العداد ، وقراءة قيمة العداد في الصيغة الثنائية أو في صيغة BCD

104

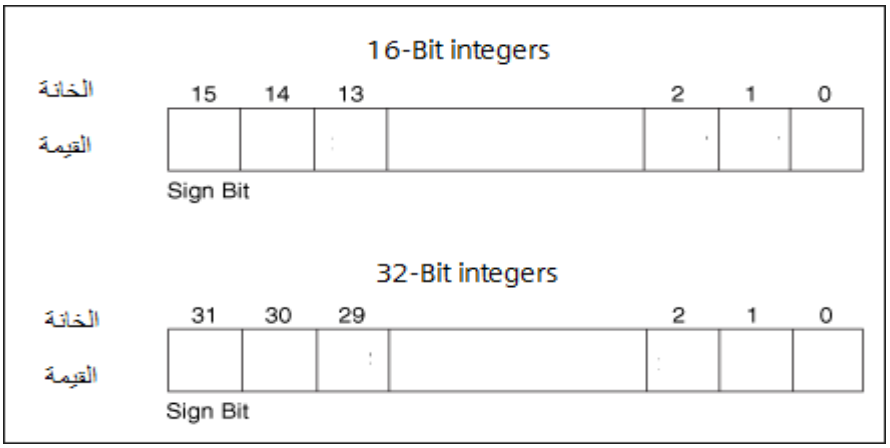
ننتقل إلى قسم آخر من التعليمات وهو تحويل البيانات من صيغة رقمية إلى أخرى للاستخدام في الحسابات حسب الدقة المطلوبة لتلك الحسابات أو للاستخدام خلال خطوات تنفيذ البرنامج حيث تتطلب بلوكات النظام إلى تهيئة البيانات بصيغ محددة وبالتالي كان لابد من وجود طريقة للتحويل من صيغة إلى أخرى

والأنواع التي نحول فيها من صيغة لأخرى هي الأرقام الصحيحة بشكليها سواء Integers و BCD بالدقة العادية 16 خانة والدقة المضاعفة 32 خانة والأرقام العشرية Real numbers وبالتالي فهذه هي الصيغ التي ستشملها تعليمات التحويل أو ما يطلق عليها Conversion instructions

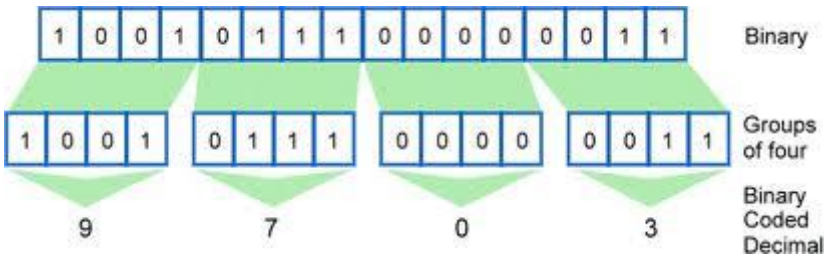
تعليمات التحويل Conversion Instructions:

كما سبق وعرضنا في كلامنا السابق عن شكل تمثيل البيانات الرقمية وذكرنا منها

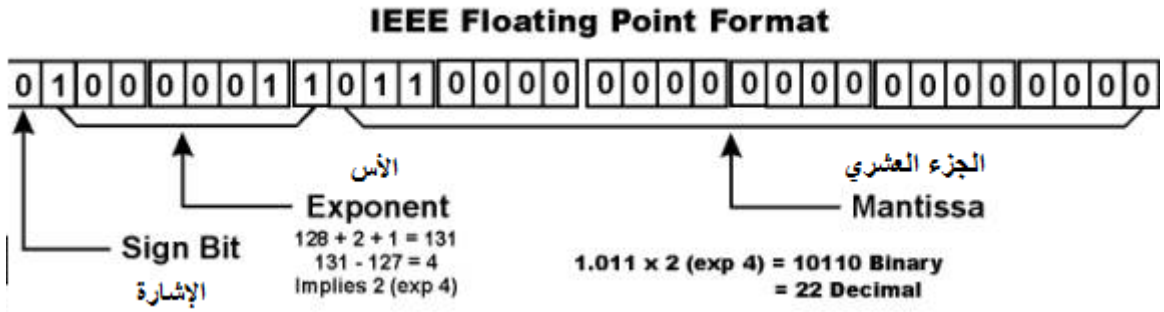
- الأرقام الصحيحة 16 خانة Integers والتي تأخذ الرمز INT عند اختيار توصيف البيانات ويتم تمثيل الرقم في Word مكونة من 16 خانة حيث تمثل الخانات من 0 وحتى 14 قيمة الرقم بتمثيل ثنائي تماما بينما الخانة الأخيرة رقم 15 تمثل الإشارة فتكون قيمة الخانة رقم-0 هي 1 والخانة رقم-1 تكون قيمتها 2 والخانة رقم-2 تكون قيمتها 4 وهكذا
- الأرقام الصحيحة 32 خانة Double integers والتي تأخذ الرمز DINT عند اختيار التوصيف ويتم تمثيلها في Double word مكونة من 16 خانة وتكون القيمة في الخانات من 0 وحتى 30 بينما الإشارة في الخانة رقم 31 وبالطبع هنا تكون القيمة أعلى كثيرا من الصيغة الصحيحة فقط INT

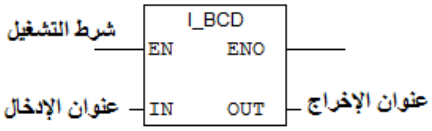
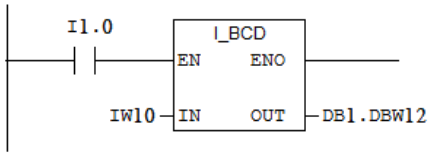
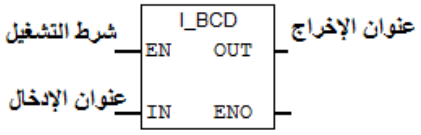
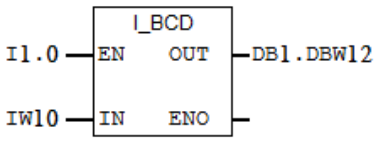
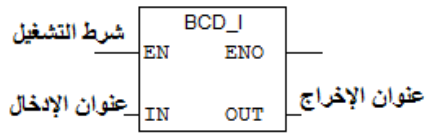
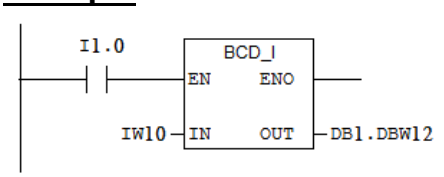
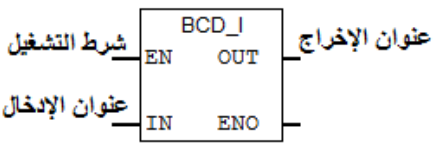
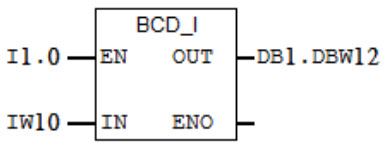


- كما يمكن تمثيل الأرقام الصحيحة في شكل BCD أو Binary coded decimal حيث يتم تمثيل الرقم الصحيح 32 خانة في شكل 4 أرقام فقط وبالتالي تتراوح قيمته من "0000" إلى "9999" وعن تمثيل الرقم الصحيح الدقة المضاعفة 32 خانة فتكون قيمته بين "0000_0000" و "9999_9999"



- أما الأرقام العشرية ذات الكسر أو ذات العلامة العشرية فيتم تمثيلها بشكل مختلف نسبيا حيث يتم تقسيم الرقم إلى ثلاث مناطق الأولى وتشمل الخانات 23 من اليمين أي من الخانة 0 وحتى الخانة 22 وتمثل قيمة الجزء العشري من الرقم ويطلق عليه Mantissa في شكل x.y حيث x تمثل العدد الصحيح و y تمثل الكسر والرقم x.y سيتم ضربه بعد ذلك في الأس والذي يتم تمثيله في الجزء الثاني ويطلق عليه Exponent ويحتل عدد 8 خانات بدءا من الخانة 23 وحتى الخانة 30 بينما الخانة رقم 31 تمثل إشارة الرقم



Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
16-bit Integer to 16-bit BCD(ITB)		
الصيغة العامة ITB **Example** L IW 10 ITB T DB1.DBW12	**الصيغة العامة**  **Example** 	**الصيغة العامة**  **Example** 
يقوم الأمر ITB بتحويل صيغة عنوان الدخول أو محتويات ACCU1 من قيمة صحيحة في صيغة INT إلى صيغة BCD أو Binary coded decimal بنفس الحجم 16 خانة حيث تحتوي قيمة الرقم المحول على ثلاث أرقام فقط في أول 12 خانة من رقم-0 وحتى رقم 11 ويتم وضع قيمة خانة الإشارة في الخانات من 12 وحتى 15 فإن كان الرقم موجبا تصبح حالتها "0000" وإن كان سالبا تصبح حالتها "1111" وبالتالي فالقيمة المتوقعة تتراوح بين "999-" و "999+" والتي تستخدم لأغراض المؤقتات والعدادات وإن تجاوزت القيم هذا الرقم فسوف تتأثر بذلك خانات OV,OS في Status		
16-bit BCD to 16-bit Integer (BTI)		
الصيغة العامة BTI **Example** L IW 10 BTI T DB1.DBW12	**الصيغة العامة**  **Example** 	**الصيغة العامة**  **Example** 
يقوم الأمر BTI بتحويل صيغة عنوان الدخول أو محتويات ACCU1 من قيمة صحيحة في صيغة BCD أو Binary coded decimal إلى صيغة INT بنفس الحجم 16 خانة وإن تجاوزت القيم هذا الرقم فسوف تتأثر بذلك خانات OV,OS في Status		

16-bit BCD to 16-bit Integer (BTI)

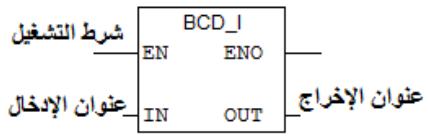
الصيغة العامة

BTI

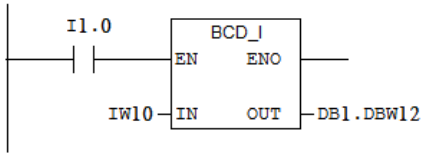
Example

L IW 10
BTI
T DB1.DBW12

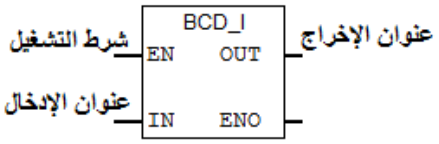
الصيغة العامة



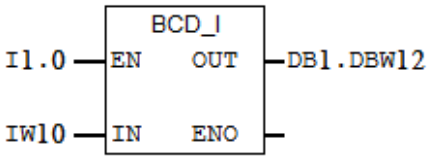
Example



الصيغة العامة



Example



يقوم الأمر BTI بتحويل صيغة عنوان الدخول أو محتويات ACCU1 من قيمة صحيحة في صيغة BCD أو Binary coded decimal إلى صيغة INT بنفس الحجم 16 خانة وإن تجاوزت القيم هذا الرقم فسوف تتأثر بذلك خانات OV,OS في Status

32-bit Integer to 32-bit BCD(DTB)

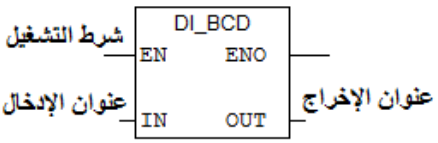
الصيغة العامة

DTB

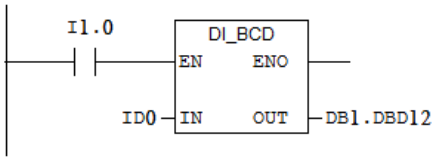
Example

L ID 10
DTB
T DB1.DBD12

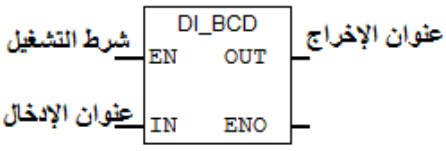
الصيغة العامة



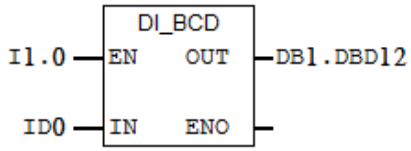
Example



الصيغة العامة



Example



يقوم الأمر DTB بتحويل صيغة عنوان الدخول أو محتويات ACCU1 من قيمة صحيحة في صيغة DINT إلى صيغة BCD أو Binary coded decimal بنفس الحجم 32 خانة حيث تحتوي قيمة الرقم المحول على 7 أرقام فقط في أول 28 خانة من رقم-0 وحتى رقم 27 ويتم وضع قيمة خانة الإشارة في الخانات من 28 وحتى 31 فإن كان الرقم موجبا تصبح حالتها "0000" وإن كان سالبا تصبح حالتها "1111" وبالتالي فالقيمة المتوقعة تتراوح بين "9999999-" و "9999999+" وإن تجاوزت القيم هذا الرقم فسوف تتأثر بذلك خانات OV,OS في Status

32-bit BCD to 32-bit Integer (BTD)

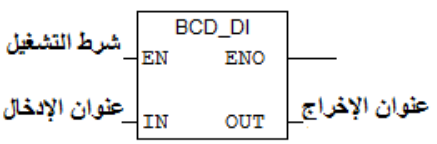
الصيغة العامة

BTD

Example

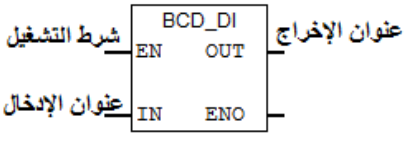
L ID 10
BTD
T DB1.DBD12

الصيغة العامة

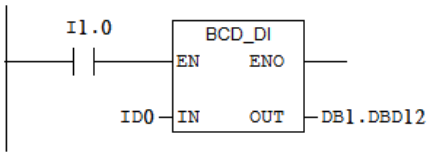


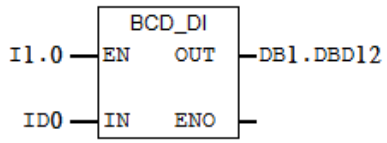
Example

الصيغة العامة



Example





يقوم الأمر BTD بتحويل صيغة عنوان الدخول أو محتويات ACCU1 من قيمة صحيحة في صيغة BCD أو Binary coded decimal إلى صيغة DINT بنفس الحجم 32 خانة وإن تجاوزت القيم هذا الرقم فسوف تتأثر بذلك خانات OV,OS في Status

16-bit Integer to 32-bit Integer (ITD)

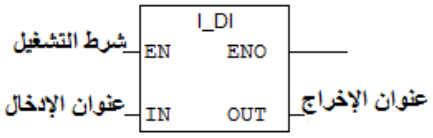
الصيغة العامة

ITD

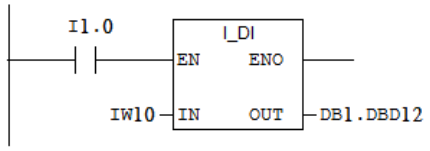
Example

L IW 10
ITD
T DB1.DBD12

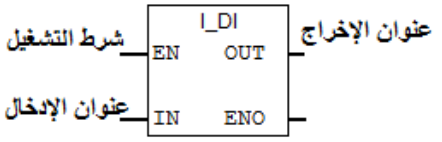
الصيغة العامة



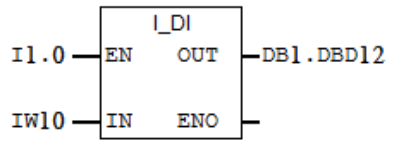
Example



الصيغة العامة



Example



يقوم الأمر ITD بتحويل صيغة عنوان الدخول أو محتويات ACCU1 من قيمة صحيحة بحجم 16 خانة في صيغة INT إلى صيغة DINT بحجم 32 خانة وإن تجاوزت القيم هذا الرقم فسوف تتأثر بذلك خانات OV,OS في Status

32-bit Integer to Real (Floating point) (DTR)

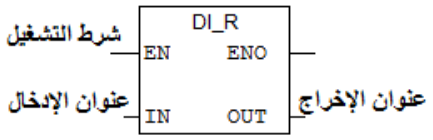
الصيغة العامة

DTR

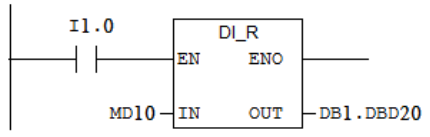
Example

L MD 10
DTR
T DB1.DBD20

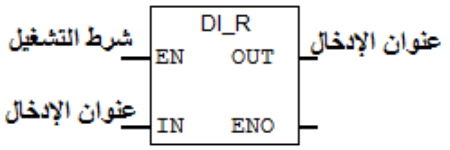
الصيغة العامة



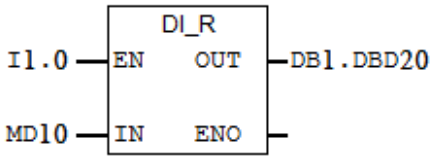
Example



الصيغة العامة



Example



يقوم الأمر DTR بتحويل صيغة عنوان الدخول أو محتويات ACCU1 من قيمة صحيحة بصيغة DINT بحجم 32 خانة إلى رقم عشري بصيغة REAL بحجم 32 خانة طبقا للتوصيف IEEE وإن تجاوزت القيم هذا الرقم فسوف تتأثر بذلك خانات OV,OS في Status

وإذا كان هناك أمر وحيد للتحويل من DINT إلى REAL فإن التحويل من REAL إلى DINT يتم بمجموعة من الأوامر وكلها تحت مسمى واحد وهو التقريب سواء لأقرب رقم صحيح أو لأكبر رقم صحيح أو لأصغر رقم صحيح أو بحذف الكسر كله مهما كانت قيمته وبالتالي فلدينا أربعة عمليات تقريب وهي RND للتقريب لأقرب رقم صحيح و +RND للتقريب لأكبر رقم صحيح و -RND للتقريب لأصغر رقم صحيح و TRUNC لحذف الكسر مهما كانت قيمته كما أن الأمرين +RND و -RND اسمين آخرين عند استخدام المخطط السلمي LAD أو القوالب الوظيفية FBD وهو FLOOR و CIEL حيث يؤديان نفس الوظيفة كما سنعرض في السطور التالية

Round to the nearest double integer (RND)

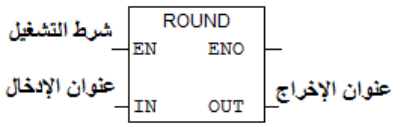
الصيغة العامة

RND

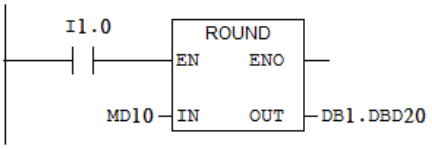
Example

L MD 10
RND
T DB1.DBD20

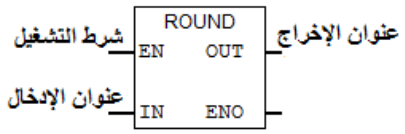
الصيغة العامة



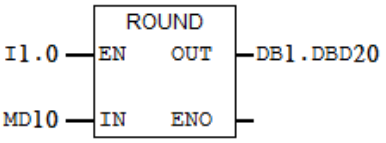
Example



الصيغة العامة



Example



يقوم الأمر RND بتقريب محتويات ACCU1 لأقرب رقم صحيح بصيغة DINT بحجم 32 خانة فإن كانت الكسر 0.5 فإن عملية التقريب تتم لأقرب رقم صحيح زوجي مثلا 11.5 يتم تقريبها إلى 12 في حين أن 10.5 يتم تقريبها إلى 10 وإن تجاوزت القيم هذا الحدود فسوف تتأثر بذلك خانات OV,OS في Status

Round to the largest double integer (RND+)

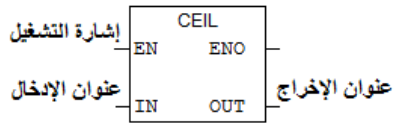
الصيغة العامة

RND+

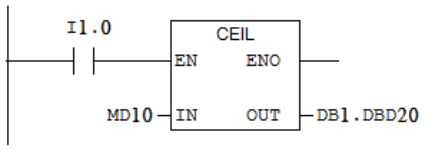
Example

L MD 10
RND+
T DB1.DBD20

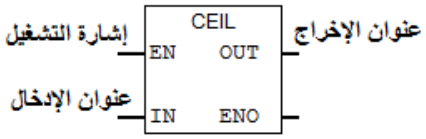
الصيغة العامة



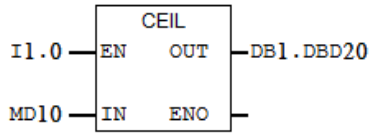
Example



الصيغة العامة



Example



يقوم الأمر RND+ أو CIEL بتقريب محتويات ACCU1 لأكبر رقم صحيح أكبر من الجزء الصحيح للرقم العشري بصيغة DINT بحجم 32 خانة وإن تجاوزت القيم هذا الحدود فسوف تتأثر بذلك خانات OV,OS في Status

Round to the Smallest double integer (RND-)

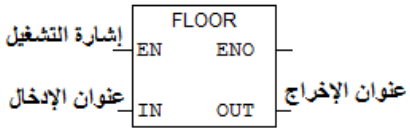
الصيغة العامة

RND-

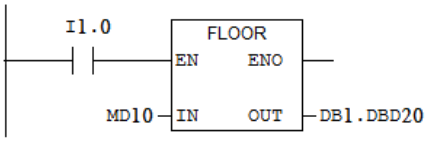
Example

L MD 10
RND-
T DB1.DBD20

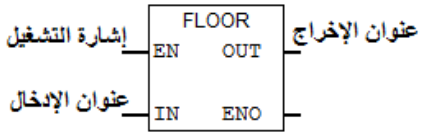
الصيغة العامة



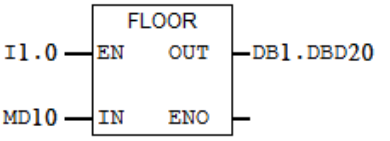
Example



الصيغة العامة



Example



يقوم الأمر RND- أو FLOOR بتقريب محتويات ACCU1 لأصغر رقم صحيح أكبر من أو يساوي الجزء الصحيح للرقم العشري بصيغة DINT بحجم 32 خانة وإن تجاوزت القيم هذا الحدود فسوف تتأثر بذلك خانات OV,OS في Status

هناك نوع آخر من عمليات التحويل ولكن على مستوى خانات وحدة الذاكرة وتشمل المقلوب الأحادي one's complement والمقلوب الثنائي two's complement وإجراءات قلب ترتيب خانات وحدة الذاكرة

One's Complement 16-bit (INVI)

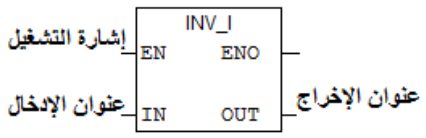
الصيغة العامة

INVI

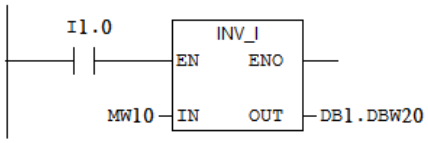
Example

L MW 10
INVI
T DB1.DBW20

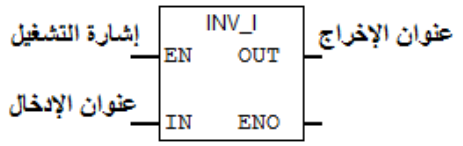
الصيغة العامة



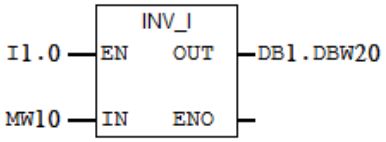
Example



الصيغة العامة



Example



يقوم الأمر INVI بتحويل محتويات ACCU1 إلى حالة One's complement بتحويل جميع الخانات من "0" إلى "1" وتحويل "1" إلى "0" في جميع خانات ACCU1

One's Complement 32-bit (INVD)

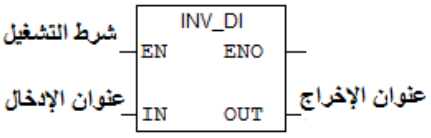
الصيغة العامة

INVD

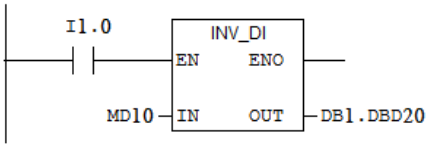
Example

L MD 10
INVI
T DB1.DBD20

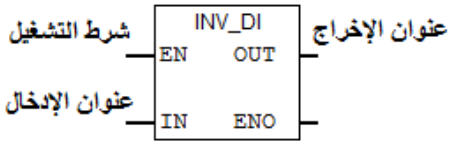
الصيغة العامة



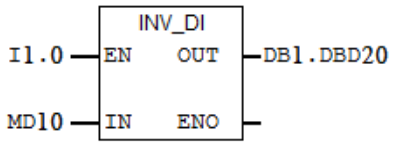
Example



الصيغة العامة



Example



يقوم الأمر INVD بتحويل محتويات ACCU1 إلى حالة One's complement بتحويل جميع الخانات من "0" إلى "1" وتحويل "1" إلى "0" في جميع خانات ACCU1

110

Two's Complement 16-bit (NEGI)

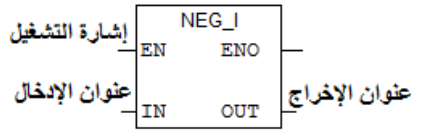
الصيغة العامة

NEGI

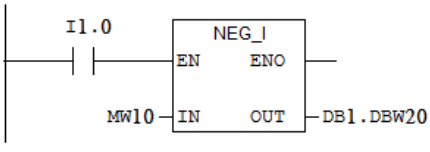
Example

L MW 10
NEGI
T DB1.DBW20

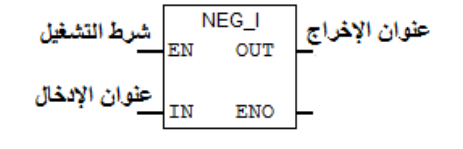
الصيغة العامة



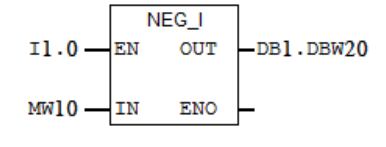
Example



الصيغة العامة

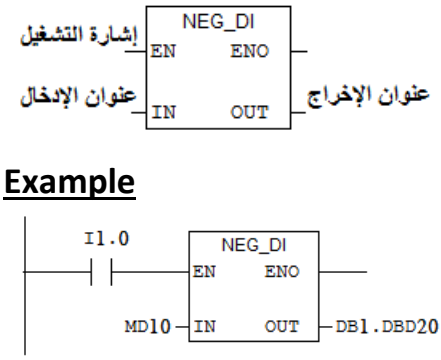
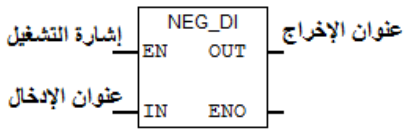
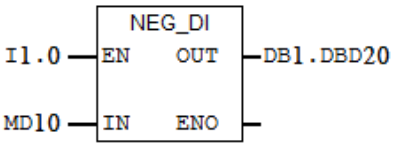


Example



يقوم الأمر NEGI بتحويل محتويات ACCU1 إلى حالة Two's complement جميع الخانات (من الخانة 0 وحتى 15) من الحالة "0" إلى "1" وتحويل "1" إلى "0" في جميع خانات ACCU1 ثم إضافة "1" إلى كل خانة ما عدا خانة الإشارة أو بالأحرى ضرب الرقم في "-1" فهو يوازي تماما نفس الرقم بإشارة سالبة أو المعكوس الجمعي للرقم

Two's Complement 32-bit (NEGD)

الصيغة العامة	الصيغة العامة	الصيغة العامة
NEGD **Example** L MD 10 NEGD T DB1.DBD20	 **Example** 	 **Example** 

يقوم الأمر NEGD بتحويل محتويات ACCU1 إلى حالة Two's complement بتحويل جميع الخانات (من الخانة 0 وحتى 31) من الحالة "0" إلى "1" وتحويل "1" إلى "0" في جميع خانات ACCU1 ثم إضافة "1" إلى كل خانة ما عدا خانة الإشارة أو بالأحرى ضرب الرقم في "1-" فهو يوازي تماما نفس الرقم بإشارة سالبة أو المعكوس الجمعي للرقم

Exchange of Bytes of Low Word of ACCU1 (CAW)

الصيغة العامة	الصيغة العامة	الصيغة العامة
CAW **Example** L IW 10 CAW T QW 20	الأمر FR غير مدعوم في LAD	الأمر FR غير مدعوم في FBD

111

الخانات من "24" إلى "31"	الخانات من "16" إلى "23"	الخانات من "8" إلى "15"	الخانات من "0" إلى "7"
ACCU1-H-H	**ACCU1-H-L**	**ACCU1-L-H**	**ACCU1-L-L**
value A	value B	value C	value D
value A	value B	value D	value C

لاحظ تغير ترتيب البايت رقم "C" مع البايت "D" بعد الأمر

يقوم الأمر CAW بتبديل ترتيب محتويات الوورد السفلى LOW Word في ACCU1 على مستوى البايت حيث يتم تبديل البايت السفلى LOW Byte مع البايت العليا High Byte في الخانات 16 الأولى من "0" وحتى "15"
لاحظ هنا أن محتويات High word تبقى بلا تغيير

Exchange of Bytes of ACCU1 (CAD)

الصيغة العامة	الصيغة العامة	الصيغة العامة
الأمر FR غير مدعوم في FBD	الأمر FR غير مدعوم في LAD	CAD **Example** L ID 10 CAD T QD 20

الخانات من "0" إلى "7"	الخانات من "8" إلى "15"	الخانات من "16" إلى "23"	الخانات من "24" إلى "31"
ACCU1-L-L	**ACCU1-L-H**	**ACCU1-H-L**	**ACCU1-H-H**
value D	value C	value B	value A
value A	value B	value C	value D

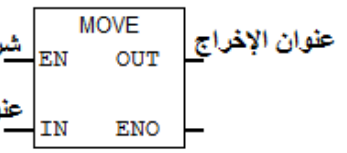
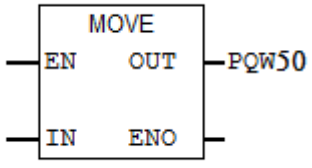
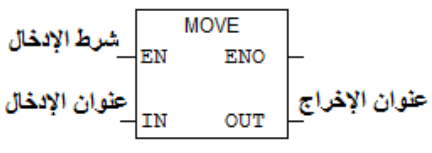
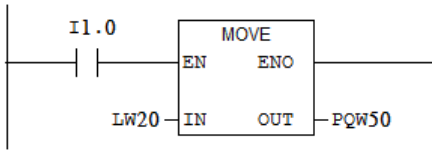
لاحظ تغير ترتيب كل البايتات حيث حلت "A" محل "D" وحلت "B" محل "C" والعكس بالعكس
فتم عكس ترتيب الوورد Word Exchange
كما تم أيضا عكس ترتيب البايت داخل كل وورد Byte Exchange

يقوم الأمر CAD بتبديل ترتيب محتويات الوورد السفلى LOW Word والعليا High Word في ACCU1 على مستوى البايت حيث يتم تبديل البايت السفلى LOW Byte مع البايت العليا High Byte كما يتم أيضا تبديل الوورد العليا High Word مع الوورد السفلى Low Word

كنا قد ذكرنا أن الأمرين Load/Transfer غير مدعومين في كل من البرمجة باستخدام LAD ولا باستخدام FBD إلا أن هناك أمرا يوازيهما تماما ويحل محلها معا حيث يقوم بالعمليتين في وقت واحد فبدلا من التحميل ثم النقل فهو يقوم بالتحميل من عنوان أو من ثابت والنقل لعنوان آخر في نفس الوظيفة ويطلق عليه اسم MOVE

أمر النقل MOVE Instruction:

MOVE Instruction

الصيغة العامة	الصيغة العامة	الصيغة العامة
 **Example** 	 **Example** 	L <Address1> T <Address2> <Address1,2> : Byte/Word/Double word **Example** L IB 10 T MB 200 L LW 20 T PQW 500 L DB1.DBD20 T QD 14

يقوم الأمر MOVE بنقل محتويات عنوان الإدخال بحجمها (Byte/Word/Double word) إلى عنوان الإخراج

العمليات المنطقية على الكلمات Word Logic Instructions:

كنا قد تعرضنا للعمليات المنطقية على الخانة الواحدة Bit Logic Instructions حيث تعرضنا للعمليات الأولية مثل AND و OR و NOT و XOR و EOR إلى غير ذلك من العمليات وفي هذا الجزء سنقوم بعمل عمليات منطقية على حزمة كاملة من البيانات في حجم Word 16 خانة أو Double Word بحجم 32 خانة مرة واحدة حيث يتم تنفيذ العملية المنطقية بين كل Bit في ACCU1 وما يوازيها من ACCU2 مثلا Bit-5 مع Bit-5 وهكذا ووضع النتيجة في ACCU1 وهنا لابد من تحميل القيم التي ستتم عليها العمليات أولا إلى ACCU1, ACCU2 ثم يتم نقل النتيجة من ACCU1 إلى عنوان الذاكرة الذي سنستخدمه بعد ذلك في البرنامج ويمكن عند البرمجة باستخدام LAD/FBD إدخال العناوين مباشرة في البلوك الخاص بالعملية المنطقية وكذلك وضع عنوان تسجيل النتيجة وفي السطور القادمة سوف نستعرض معا هذه العمليات والتي تتلخص في 6 عمليات ؛ 3 على مستوى Word و 3 على مستوى DWord وهي عمليات AND, OR, XOR فقط

16-Bit AND (AW)

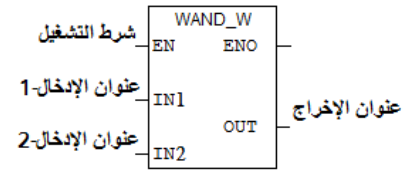
الصيغة العامة

AW

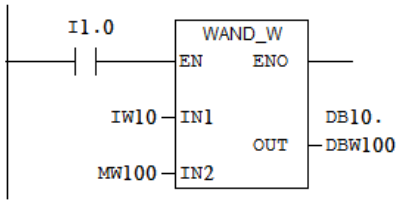
Example

```
L IW 10
L MW 100
AW
T DB10.DBW100
```

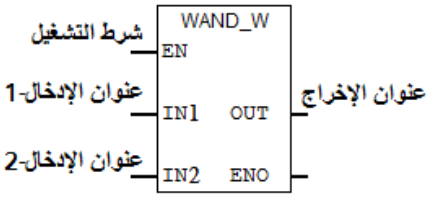
الصيغة العامة



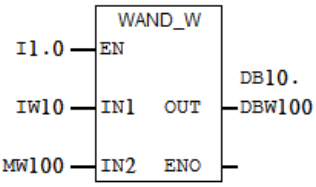
Example



الصيغة العامة



Example



يقوم الأمر AW بعمل AND بين كل خانة Bit في ACCU1 وما يوازيها في ACCU2 ووضع النتيجة في الخانة نفس الخانة في ACCU1 على مستوى الوورد السفلى Low word

16-Bit OR (OW)

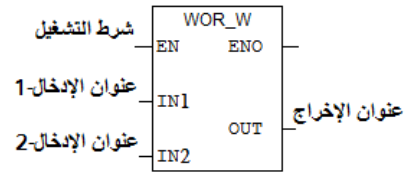
الصيغة العامة

OW

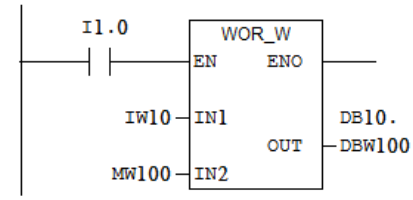
Example

```
L IW 10
L MW 100
OW
T DB10.DBW100
```

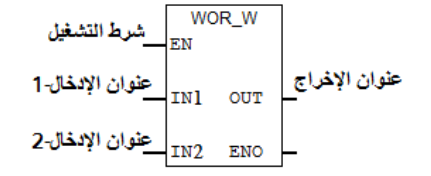
الصيغة العامة



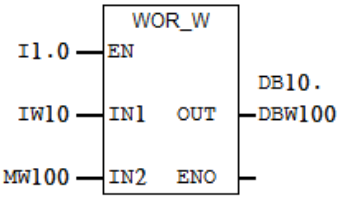
Example



الصيغة العامة



Example



يقوم الأمر OW بعمل OR بين كل خانة Bit في ACCU1 وما يوازيها في ACCU2 ووضع النتيجة في الخانة نفس الخانة في ACCU1 على مستوى الوورد السفلى Low word

16-Bit XOR (XOW)

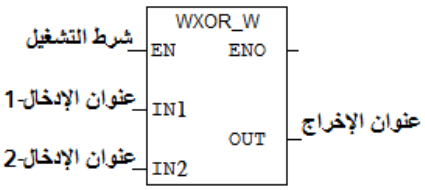
الصيغة العامة

XOW

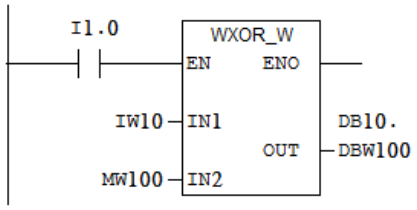
Example

```
L IW 10
L MW 100
XOW
T DB10.DBW100
```

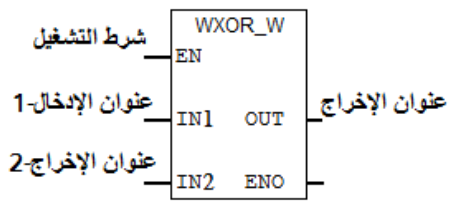
الصيغة العامة



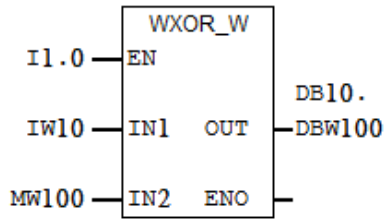
Example



الصيغة العامة



Example



يقوم الأمر XOW بعمل XOR بين كل خانة Bit في ACCU1 وما يوازيها في ACCU2 ووضع النتيجة في الخانة نفس الخانة في ACCU1 على مستوى الوورد السفلى Low word

32-Bit AND (AD)

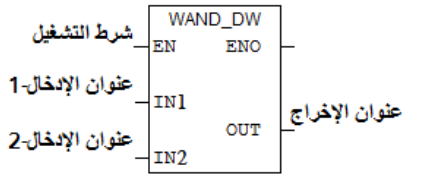
الصيغة العامة

AD

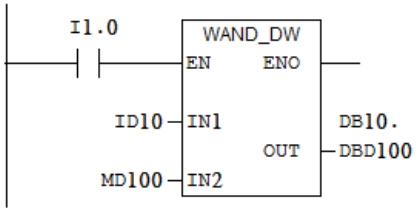
Example

```
L ID 10
L MD 100
AD
T DB10.DBD100
```

الصيغة العامة



Example



الصيغة العامة

Example

يقوم الأمر AD بعمل AND بين كل خانة Bit في ACCU1 وما يوازيها في ACCU2 ووضع النتيجة في الخانة نفس الخانة في ACCU1

32-Bit OR (OD)

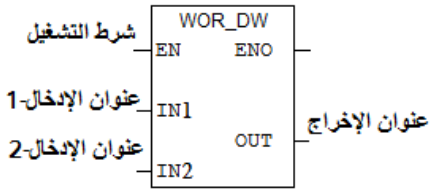
الصيغة العامة

OD

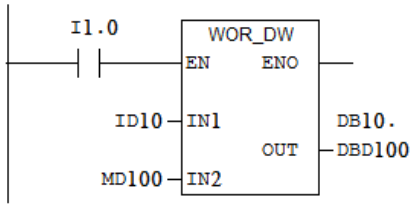
Example

```
L ID 10
L MD 100
OD
T DB10.DBD100
```

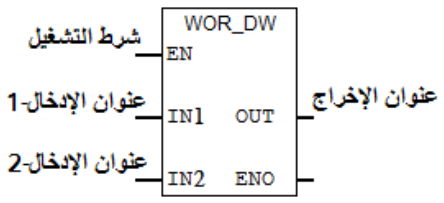
الصيغة العامة



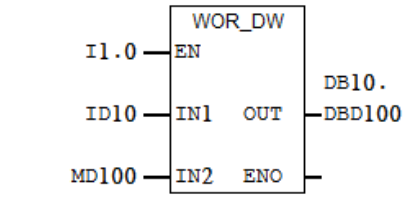
Example



الصيغة العامة



Example



يقوم الأمر OD بعمل OR بين كل خانة Bit في ACCU1 وما يوازيها في ACCU2 ووضع النتيجة في الخانة نفس الخانة في ACCU1

32-Bit XOR (XOD)

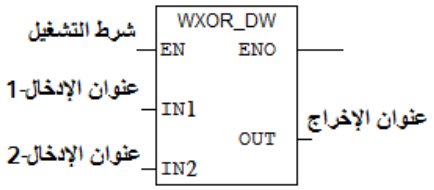
الصيغة العامة

XOD

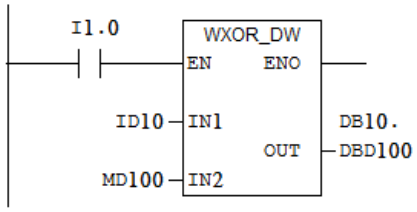
Example

```
L ID 10
L MD 100
XOD
T DB10.DBD100
```

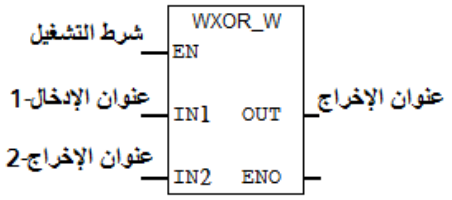
الصيغة العامة



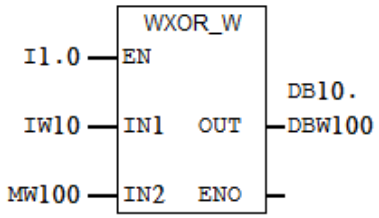
Example



الصيغة العامة



Example



يقوم الأمر XOD بعمل XOR بين كل خانة Bit في ACCU1 وما يوازيها في ACCU2 ووضع النتيجة في الخانة نفس الخانة في ACCU1

115

تعليمات المقارنة Comparison Instructions:

تمثل عمليات المقارنة ضرورة كبيرة في عمليات البرمجة والتي على أساسها يتم اتخاذ العديد من القرارات في البرامج خاصة عندما يعتمد الأمر على قيم رقمية والمقارنة تتم بين أنواع متشابهة في الصيغة وفي الحجم ومنها ثلاثة أنواع طبقا للبيانات التي تتم مقارنتها وهي على مستوى الأرقام الصحيحة INT والأرقام الصحيحة المضاعفة الدقة DINT والأرقام العشرية REAL ونتيجة عمليات المقارنة تكون حالة إما نعم "1" أو لا "0" طبقا لشرط المقارنة

وعمليات المقارنة تتم تماما وفقا للمبادئ الرياضية فهي إما:

- أكبر من : "<"
- أقل من : ">"
- تساوي : "=="
- أكبر من أو يساوي : "=<"
- أقل من أو يساوي : "=>"
- لا يساوي : "<>"

فإذا أضفنا لكل نوع من هذه الأنواع الستة من المقارنة رمزا يمثل صيغة البيانات التي يتم مقارنتها فسوف نحصل على 18 عملية مقارنة حيث يتم مقارنة :

- الأرقام الصحيحة : <I , >I , ==I , <=I , >=I , <>I
- الأرقام الصحيحة المضاعفة : <D , >D , ==D , <=D , >=D , <>D
- الأرقام العشرية : <R , >R , ==R , <=R , >=R , <>R

وتتم المقارنة بين محتويات ACCU1 و ACCU2 فتتم مقارنة محتويات ACCU1 نسبة إلى ACCU2 حيث يجب تحميل محتوياتهما قبل القيام بعمليات المقارنة ووفقا لحالة المقارنة يتم تسجيل النتيجة في RLO فإن تحققت المقارنة يتم تحويل حالة RLO إلى "1" وإلا فتكون "0"

كما تتم أيضا مراقبة عمليات المقارنة أيضا عن طريق الخانتين CC0,CC1 في مسجل الحالة Status register على النحو التالي :

CC1	CC0	نتيجة المقارنة
0	1	ACCU1 > ACCU2
1	0	ACCU1 < ACCU2
0	0	ACCU1 = ACCU2

وعند عمليات المقارنة تكون القيمة داخل ACCU1 هي المقارن به والقيمة داخل ACCU2 هي المقارن فمثلا لو أردنا مقارنة محتويات MW100 مع محتويات DB10.DBW12 لاختبار هل MW100 أكبر من أو يساوي DB10.DBW12 فإننا نقوم أولا بتحميل MW100 ثم بعد ذلك بتحميل DB10.DBW12 وبالتالي يصبح المقارن MW100 داخل ACCU2 ويصبح المقارن به DB10.DBW12 داخل ACCU1

وفي حالة البرمجة باستخدام LAD/FBD فإننا نستخدم بلوك كامل له عدد 2 مدخل يمثلان المقارن IN1 والمقارن به IN2 ونحصل على النتيجة على خرج البلوك

وسوف نستعرض معا تفاصيل عمليات المقارنة في الجدول التالي:

Integer Comparison (?I)

Statement List (STL)

Compare Integer numbers : <I , >I , ==I , <=I , >=I , <>I

الصيغة العامة

<I	أقل من
>I	أكبر من
==I	يساوي
<=I	أقل من أو يساوي
>=I	أكبر من أو يساوي
<>I	لا يساوي

Example

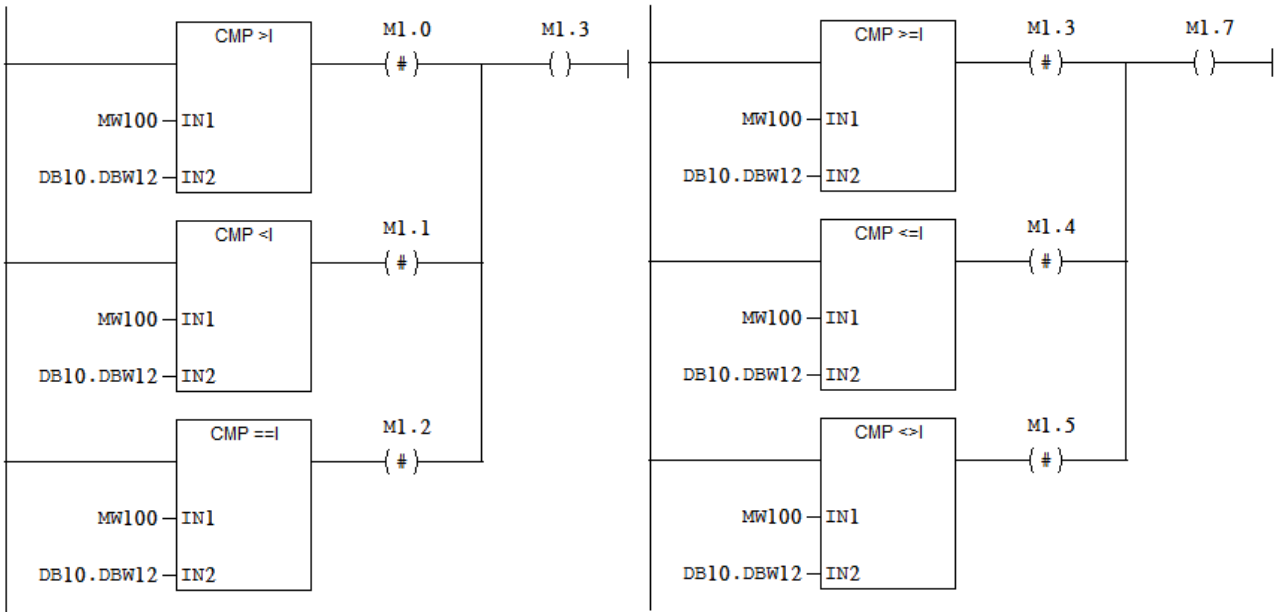
```
A(
L     MW   100
L     DB10.DBW   12
>I
)
=     M      1.0
A     M      1.0
O(
A(
L     MW   100
L     DB10.DBW   12
<I
)
=     M      1.1
A     M      1.1
)
O(
A(
L     MW   100
L     DB10.DBW   12
==I
)
=     M      1.2
A     M      1.2
)
=     M      1.6
```

```
A(
L     MW   100
L     DB10.DBW   12
>=I
)
=     M      1.3
A     M      1.3
O(
A(
L     MW   100
L     DB10.DBW   12
<=I
)
=     M      1.4
A     M      1.4
)
O(
A(
L     MW   100
L     DB10.DBW   12
<>I
)
=     M      1.5
A     M      1.5
)
=     M      1.7
```

Ladder Diagram (LAD)

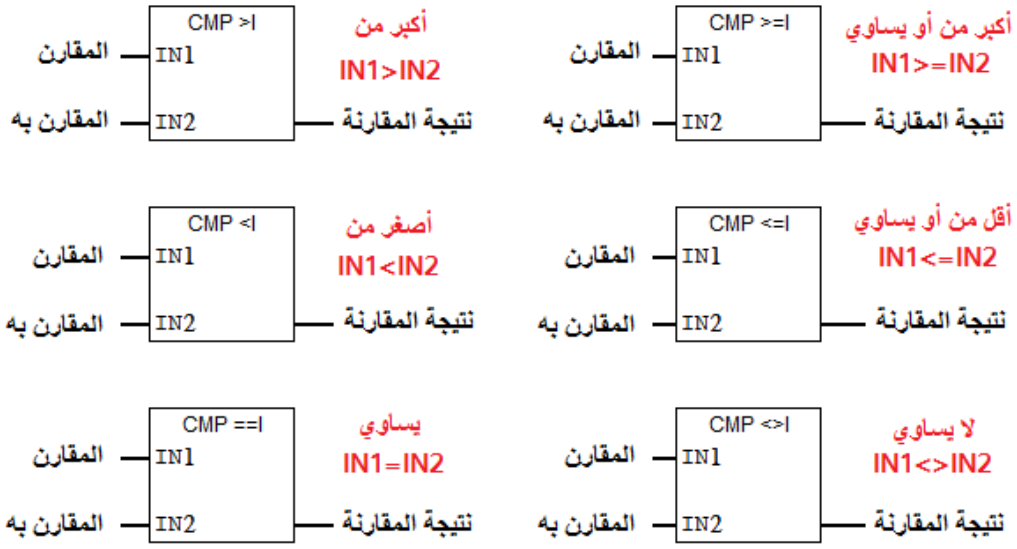
الصيغة العامة

Example

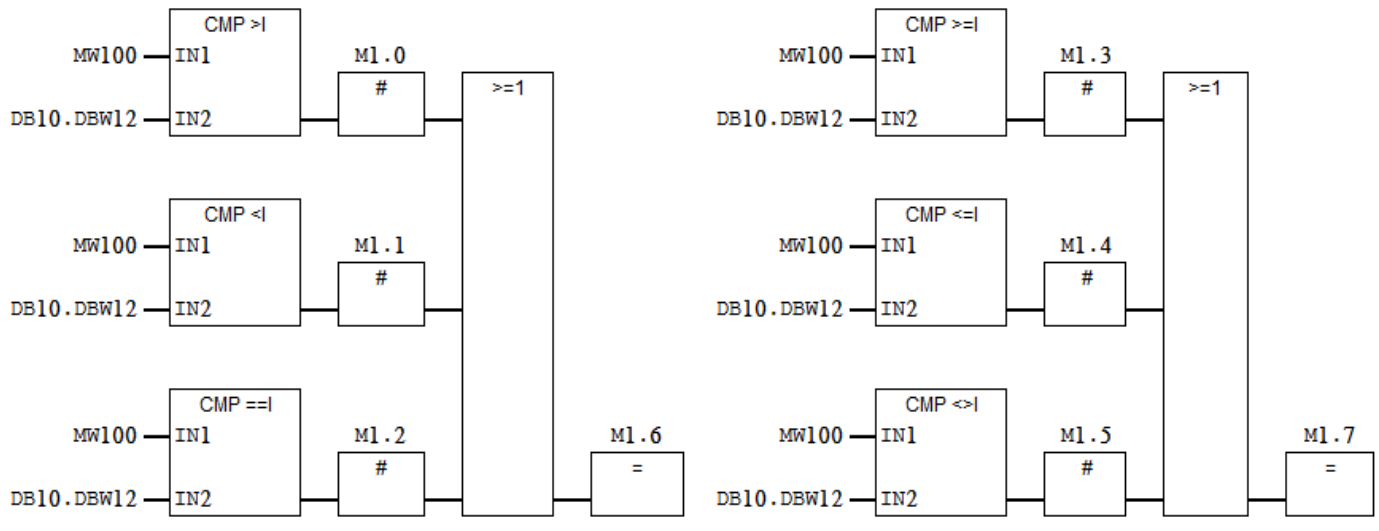


Function Block Diagram (FBD)

الصيغة العامة



Example



Statement List (STL)

Compare Double Integer numbers : <D , >D , ==D , <=D , >=D , <>D

الصيغة العامة

<D	***أقل من***
>D	***أكبر من***
==D	***يساوي***
<=D	***أقل من أو يساوي***
>=D	***أكبر من أو يساوي***
<>D	***لا يساوي***

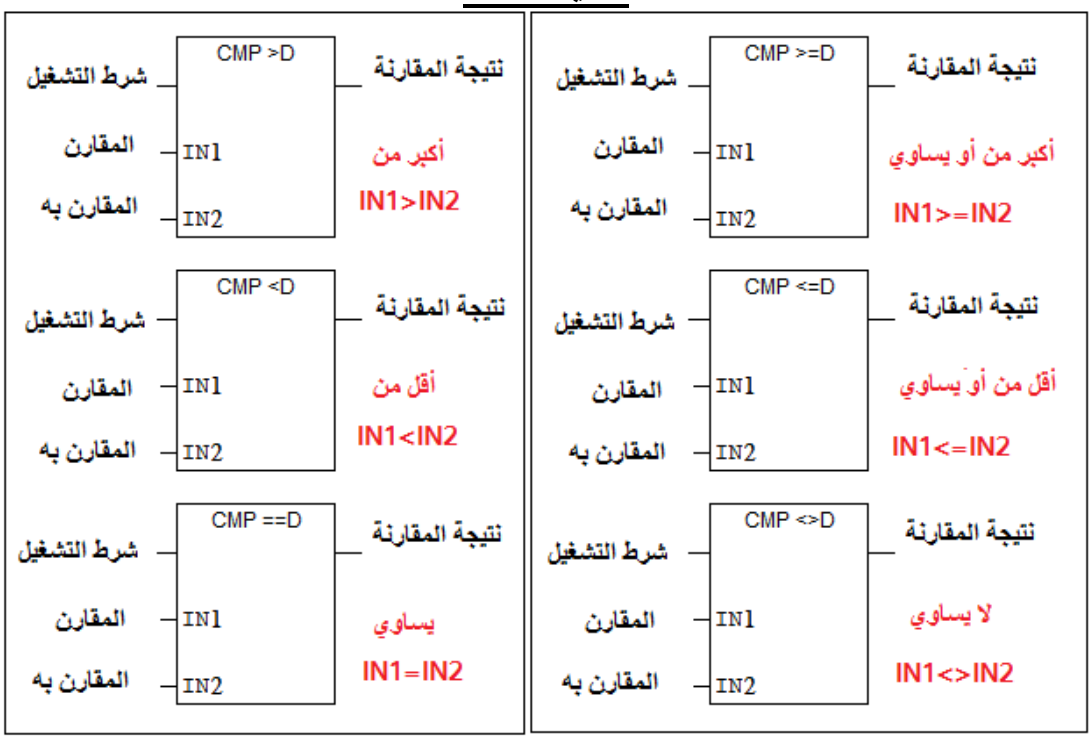
Example

```
A(
L     MD   100
L     DB10.DBD   12
>D
)
=     M     1.0
A     M     1.0
O(
A(
L     MD   100
L     DB10.DBD   12
<D
)
=     M     1.1
A     M     1.1
)
O(
A(
L     MD   100
L     DB10.DBD   12
==D
)
=     M     1.2
A     M     1.2
)
=     M     1.6
```

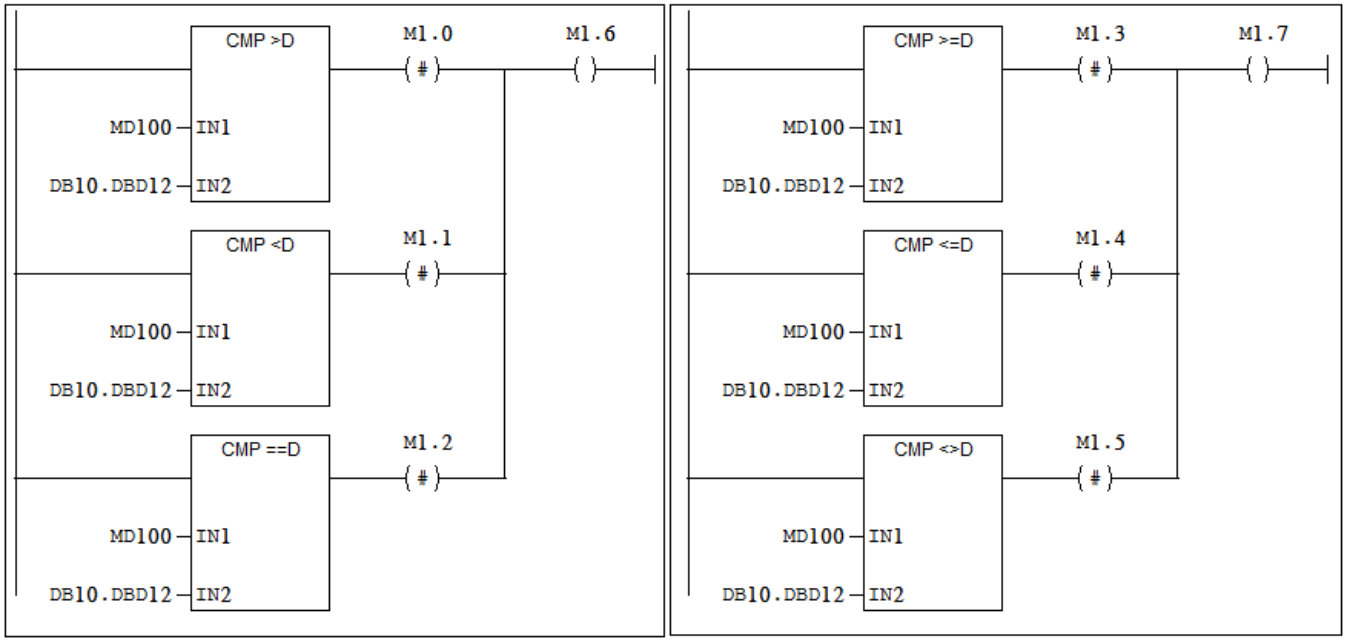
```
A(
L     MD   100
L     DB10.DBD   12
>=D
)
=     M     1.3
A     M     1.3
O(
A(
L     MD   100
L     DB10.DBD   12
<=D
)
=     M     1.4
A     M     1.4
)
O(
A(
L     MD   100
L     DB10.DBD   12
<>D
)
=     M     1.5
A     M     1.5
)
=     M     1.7
```

Ladder Diagram (LAD)

الصيغة العامة

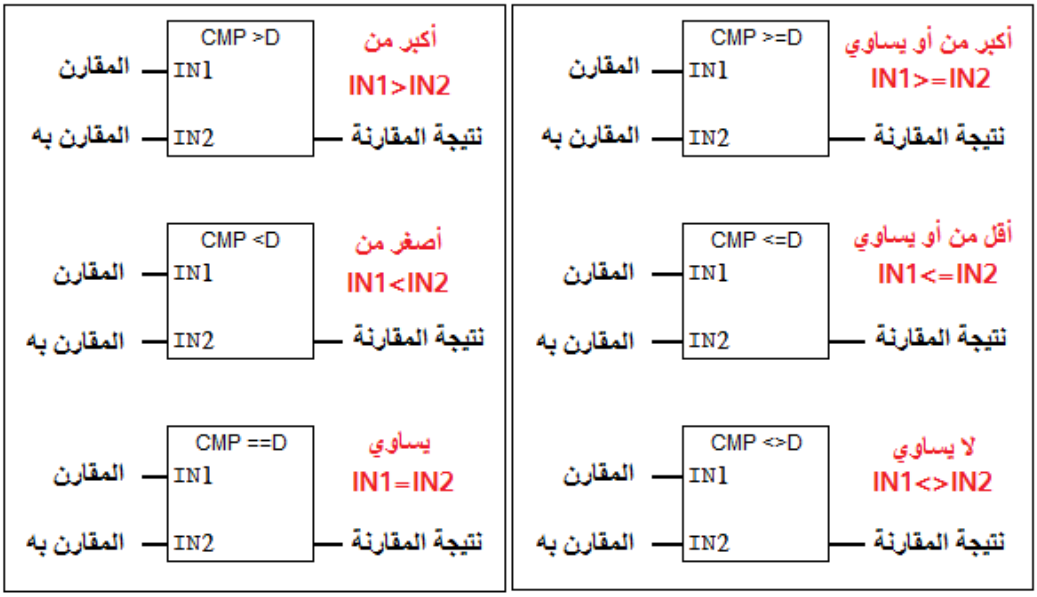


Example

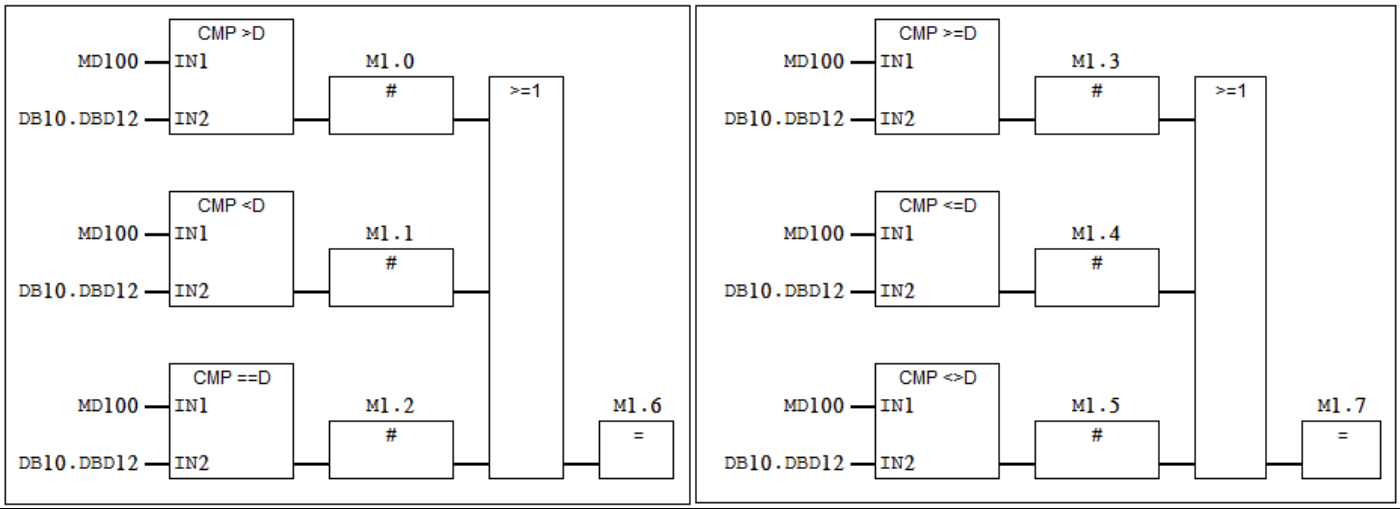


Function Block Diagram (FBD)

الصيغة العامة



Example



Statement List (STL)

Compare Real numbers : <R , >R , ==R , <=R , >=R , <>R

الصيغة العامة

<R	*أقل من*
>R	*أكبر من*
==R	*يساوي*
<=R	*أقل من أو يساوي*
>=R	*أكبر من أو يساوي*
<>R	*لا يساوي*

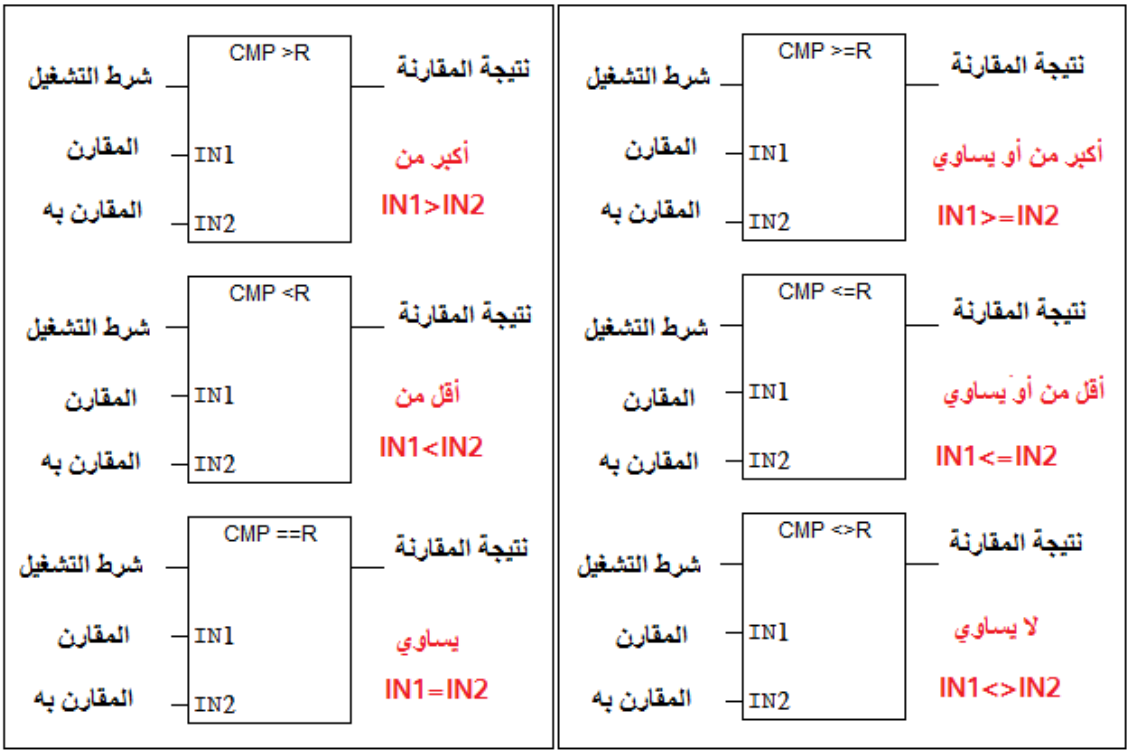
Example

```
A(
L     MD   100
L     DB10.DBD   12
>R
)
=     M     1.0
A     M     1.0
O(
A(
L     MD   100
L     DB10.DBD   12
<R
)
=     M     1.1
A     M     1.1
)
O(
A(
L     MD   100
L     DB10.DBD   12
==R
)
=     M     1.2
A     M     1.2
)
=     M     1.6
```

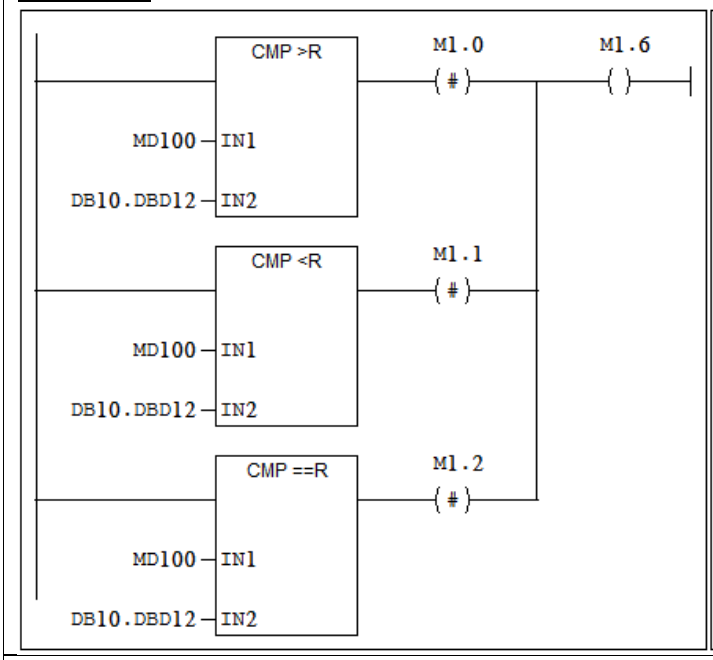
```
A(
L     MD   100
L     DB10.DBD   12
>=R
)
=     M     1.3
A     M     1.3
O(
A(
L     MD   100
L     DB10.DBD   12
<=R
)
=     M     1.4
A     M     1.4
)
O(
A(
L     MD   100
L     DB10.DBD   12
<>R
)
=     M     1.5
A     M     1.5
)
=     M     1.7
```

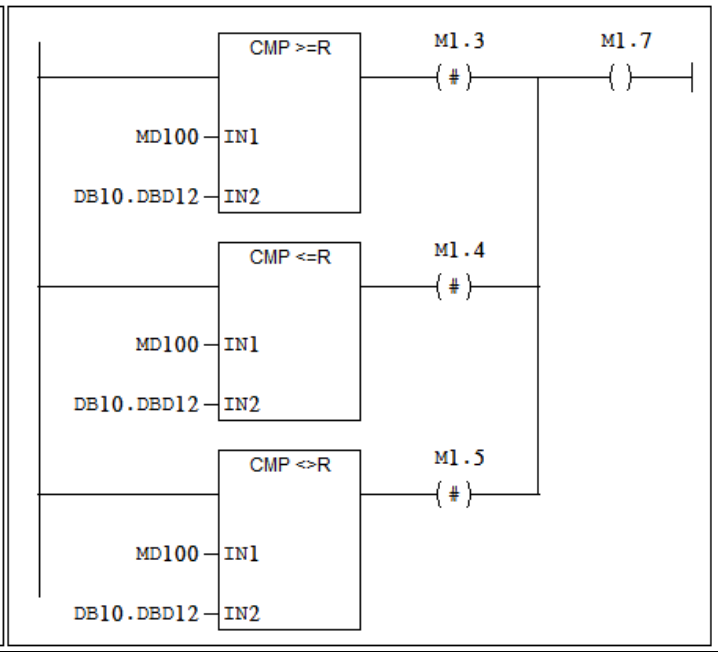
Ladder Diagram (LAD)

الصيغة العامة



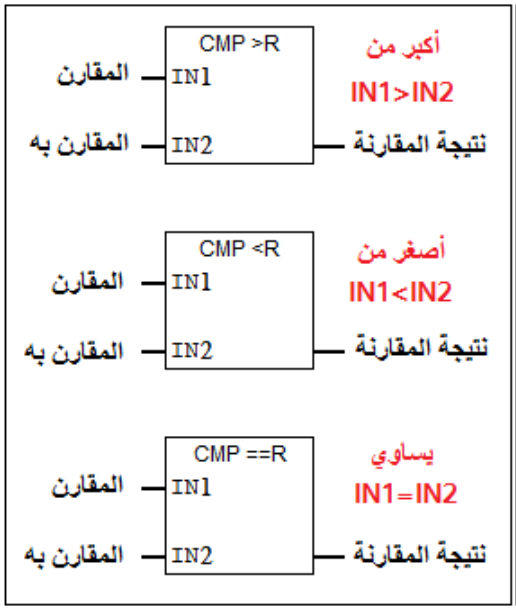
Example

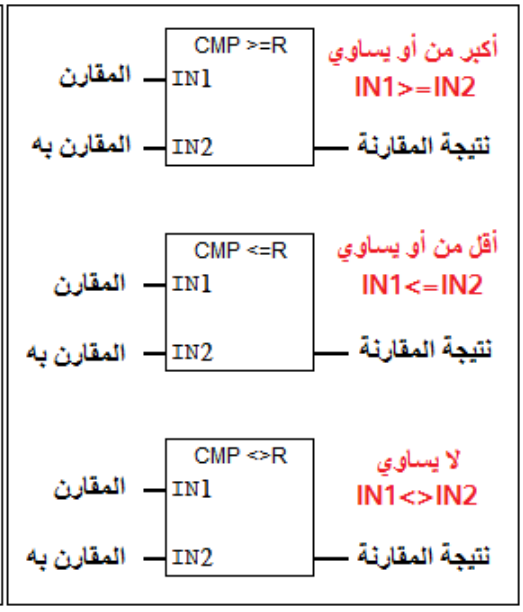




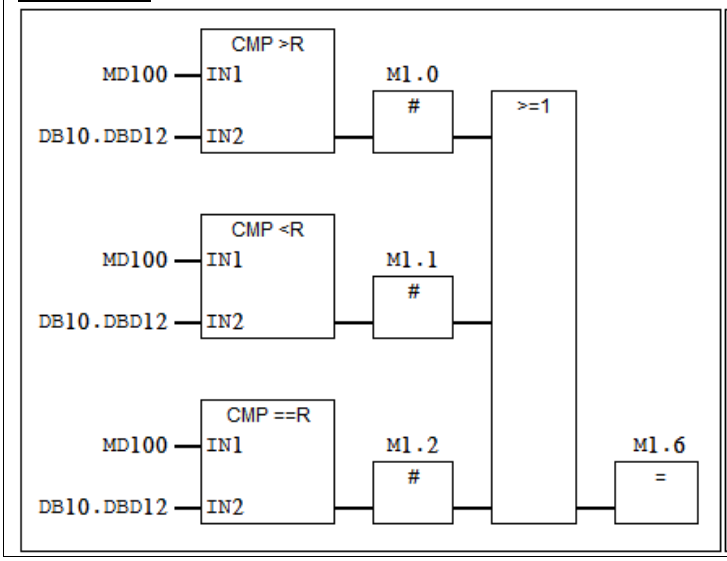
Function Block Diagram (FBD)

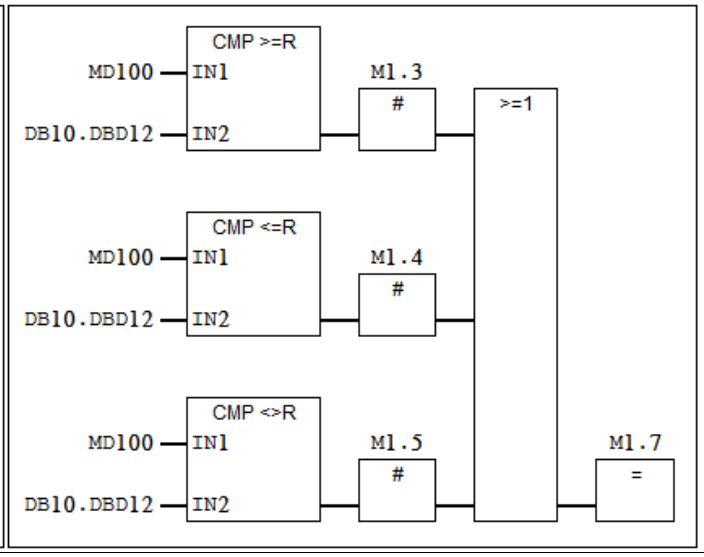
الصيغة العامة





Example





تعليمات الإزاحة والتدوير Shift and Rotate Instructions:

تتم عمليات الإزاحة والتدويرعلى محتويات ACCU1 إما إلى اليمين أو إلى اليسار بعدد خانات محددة ودون النظر إلى شكل البيانات نفسها (باستثناء أمرين فقط) ويعتمد فقط على الحجم إما في حجم WORD 16 خانة أو Double-Word بحجم 32 خانة

وتتم عمليات الإزاحة على محتويات ACCU1 بأشكال ستة منها ثلاثة على حجم بيانات WORD وثلاثة على حجم بيانات Double-Word وهي:

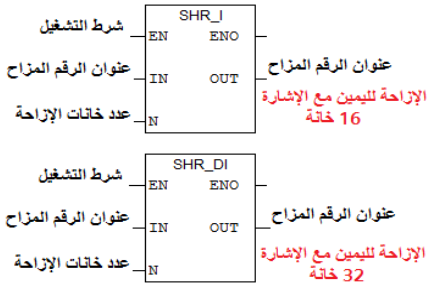
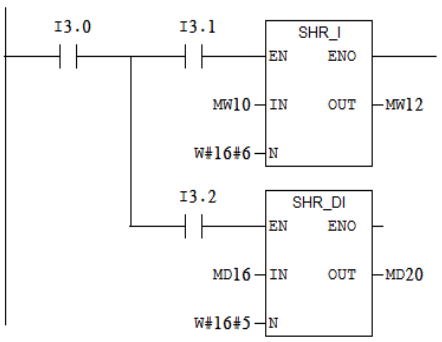
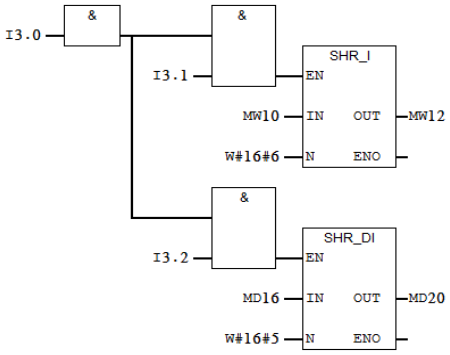
- الإزاحة إلى اليمين باعتبار الإشارة 16 خانة : SSI
- الإزاحة إلى اليمين باعتبار الإشارة 32 خانة : SSD
- الإزاحة المطلقة إلى اليمين (16 خانة) : SRW
- الإزاحة المطلقة إلى الشمال (16 خانة) : SLW
- الإزاحة المطلقة إلى اليمين (32 خانة) : SRD
- الإزاحة المطلقة إلى الشمال (32 خانة) : SLD

أما عمليات التدوير فتتم أيضا بطريقتين الأولى مع عدم وجود وسيط حيث يتم نقل آخر محتويات خانة إلى أول خانة وهكذا في اتجاه الإزاحة والثانية بوجود وسيط وهو CC0 حيث يتم استخدام محتوياتها في العملية وبالتالي يكون لدينا أربعة عمليات حيث تتم عملية التدوير على Double-Word فقط وهي :

- التدوير المطلق لليمين : RRD
- التدوير المطلق للشمال : RLD
- التدوير المطلق لليمين مع CC1 : RRDA
- التدوير المطلق للشمال مع CC1 : RLDA

123

Shift Signed Integers (SSI), Double Integers (SSD)

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة SSI SSD **Example:** L W#16#6 L MW 10 SSI T MW 12 L W#16#5 L MD 16 SSD T MD 20	**الصيغة العامة**  **Example:**  	**الصيغة العامة**  **Example:** 

يقوم الأمر SSI بإزاحة محتويات ACCU1 بعدد الخانات حسب قيمة ACCU2-Low Word-Low Byte بحد أقصى 15 خانة مع إحلال الخانات المزاحة بنفس قيمة خانة الإشارة فإن كانت إشارة الرقم سالبة أي أن الخانة رقم 15 فيها "1" يتم ملء كل الخانات المزاحة بالقيمة "1" وإن كان الرقم موجب أي أن الخانة 15 بها "0" يتم ملء الخانات المزاحة بالرقم "0" وبالتالي فإزاحة الرقم الموجب 16 خانة ينتج عنها قيمة "0" في جميع الخانات وإزاحة الرقم السالب 16 خانة إلى اليمين ينتج عنها القيمة "1" في جميع الخانات ولا تتأثر بذلك الوورد الأعلى من ACCU1

كذلك يقوم الأمر SSD بنفس الوظيفة ولكن على مستوى 32 خانة

Contents	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
before execution of **SSI 6**	0101	1111	0110	0100	1001	1101	0011	1011
after execution of **SSI 6**	0101	1111	0110	0100	1111	1110	0111	0100

Contents	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
before execution of **SSD 7**	1000	1111	0110	0100	0101	1101	0011	1011
after execution of **SSD 7**	1111	1111	0001	1110	1100	1000	1011	1010

Shift Right Word (SRW), Shift Left Word (SLW)

Statement List (STL)

الصيغة العامة

SRW

SLW

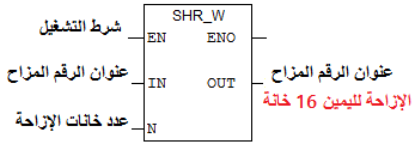
Example:

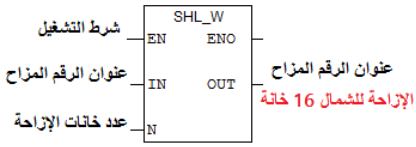
```
L     W#16#6
L     MW    100
SRW
T     MW    102

L     W#16#6
L     MW    100
SLW
T     MW    104
```

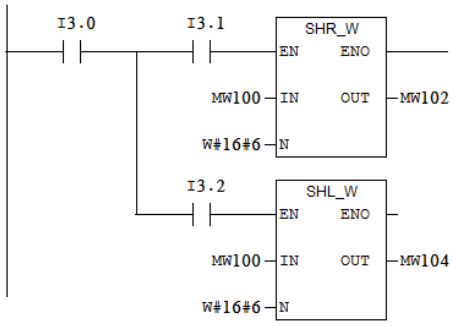
Ladder Diagram (LAD)

الصيغة العامة



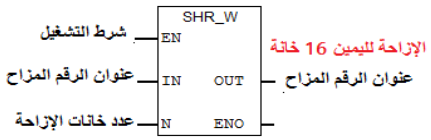


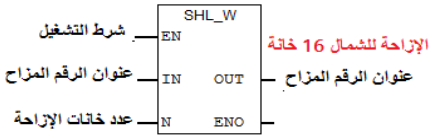
Example:



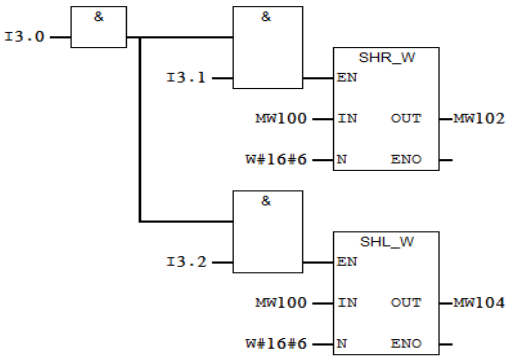
Function Block Diagram (FBD)

الصيغة العامة





Example:



يقوم الأمر SRW بإزاحة محتويات ACCU1 على مستوى Low-WORD بعدد خانات يساوي الرقم المسجل في Low-Byte من Low-Word من ACCU2 مع إحلال الخانات المزاحة بأصفار "0"

بينما يقوم الأمر SLW بنفس الشئ ولكن الإزاحة تكون إلى الشمال وفي الحالتين لا تتأثر High word بالإزاحة

Contents	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
before execution of **SRW 6**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **SRW 6**	0101	1111	0110	0100	0000	0001	0111	0100
before execution of **SLW 5**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **SLW 5**	0101	1111	0110	0100	1010	0111	0110	0000

Shift Right DWord (SRD), Shift Left DWord (SLD)

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)

الصيغة العامة

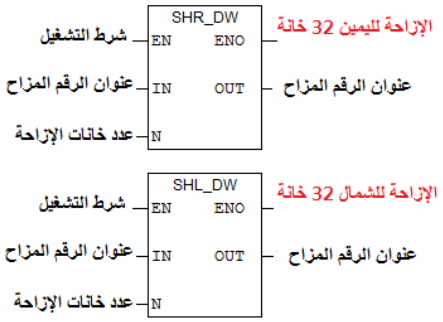
SRD
SLD

Example:

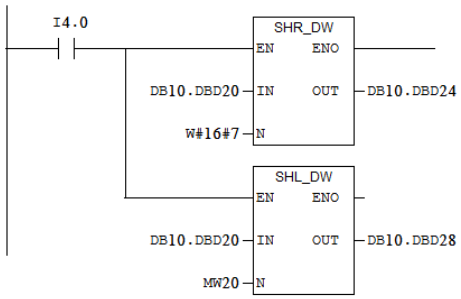
```
L     W#16#7
L     DB10.DBD   20
SRD
T     DB10.DBD   24

L     MW    20
L     DB10.DBD   20
SLD
T     DB10.DBD   28
```

الصيغة العامة

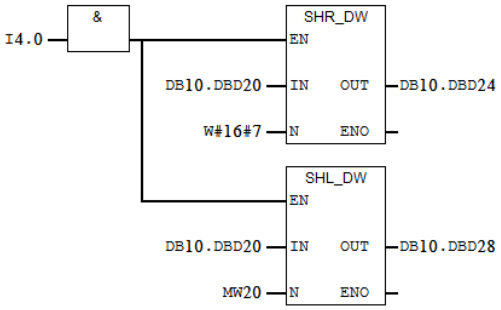


Example:



الصيغة العامة

Example:



يقوم الأمر SRD بإزاحة محتويات ACCU1 بعدد خانات يساوي الرقم المسجل في Low-Byte من Low-Word من ACCU2 مع إحلال الخانات المزاحة بأصفار "0"

بينما يقوم الأمر SLD بنفس الشئ ولكن الإزاحة تكون إلى الشمال

Contents	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
before execution of **SRD 7**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **SRD 7**	0000	0000	1011	1110	1100	1000	1011	1010
before execution of **SLD 5**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **SLD 5**	1110	1100	1000	1011	1010	0111	0110	0000

Rotate Right DWord (RRD), Rotate Left DWord (RLD)

Statement List (STL)	العملية غير موجودة في LAD/FBD

الصيغة العامة

RRDA
RLDA

Example:

```
L     W#16#8
L     ID     0
RRDA
T     QD     12

L     W#16#6
L     ID     4
RLDA
T     QD     16
```

يقوم الأمر RRDA بتدوير محتويات ACCU1 بعدد خانات الرقم الموجود في Low-Byte من Low-Word من ACCU2 مع نقل الخانة رقم "0" التي يتم تدويرها إلى CC1 ونقل محتويات CC1 إلى الخانة رقم "31"

وبنفس الطريقة يعمل الأمر RLDA حيث يتم نقل محتويات الخانة رقم "31" إلى CC1 ونقل محتويات CC1 إلى الخانة رقم "0" وبالتالي فهنا دوما نفقد المحتوى الأصلي للخانة القصوى عكس الدوران

Contents	CC 1	ACCU1-H				ACCU1-L			
Bit		31 . . .	. .	. .	. . . 16	15 . . .	. .	. .	. . . 0
before execution of **RRDA**	**X**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **RRDA**	**1**	X010	1111	1011	0010	0010	1110	1001	1101
before execution of **RLDA**	**X**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **RLDA**	**0**	1011	1110	1100	1000	1011	1010	0111	011X
	(X = 0 or 1, previous signal state of CC 1)								

Rotate Right DWord (RRD), Rotate Left DWord (RLD)

Statement List (STL)

الصيغة العامة

RRD
RLD

Example:

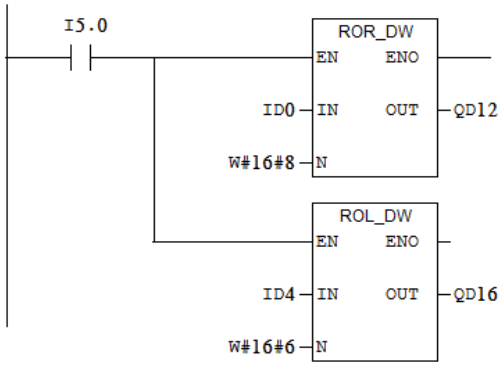
```
L     W#16#8
L     ID     0
RRD
T     QD     12

L     W#16#6
L     ID     4
RLD
T     QD     16
```

Ladder Diagram (LAD)

الصيغة العامة

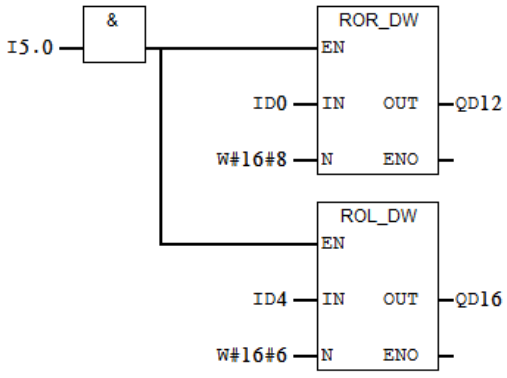
Example:



Function Block Diagram (FBD)

الصيغة العامة

Example:



126

يقوم الأمر RRD بتدوير محتويات ACCU1 بعدد خانات يساوي الرقم المسجل في Low-Byte من Low-Word من ACCU2 حيث يتم الإزاحة في اتجاه الدوران مع نقل الخانة رقم "0" إلى الخانة رقم "31" في كل مرة للإزاحة حتى تتم إزاحة كل الخانات المطلوبة مع تسجيل محتوى الخانة المزاحة كل مرة في CC1

بينما يقوم الأمر RLD بنفس الشئ ولكن التدوير يكون للشمال بحيث يتم نقل المحتويات إلى الشمال مع نقل محتوى الخانة رقم "31" في كل مرة إلى الخانة رقم "0" حتى تنتهي عمليات التدوير مع الاحتفاظ بمحتوى الخانة المزاحة في كل مرة في الخانة CC1 من مسجل الحالة

Contents	ACCU1-H				ACCU1-L			
Bit	31 . . .	. .	. .	. . . 16	15 . .	. .	. .	. . . 0
before execution of **RRD 4**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **RRD 4**	1011	0101	1111	0110	0100	0101	1101	0011
before execution of **RLD 4**	0101	1111	0110	0100	0101	1101	0011	1011
after execution of **RLD 4**	1111	0110	0100	0101	1101	0011	1011	0101

التحكم المنطقي في سير البرنامج Logic Control Instructions:

من الطبيعي أن يتم تنفيذ خطوات البرنامج بشكل متسلسل من أعلى لأسفل ومن الشمال لليمين وحسب خطوات كتابة البرنامج حيث أن السطر الأول يتم تنفيذه أولا ثم الذي يليه وهكذا ما يتم قطع هذا التسلسل بأحد أمرين ، الأول هو المقاطعة Interrupts بأشكالها المختلفة والتي ترتبط بأحداث محددة كما تحدثنا في ضبط وحدة المعالجة عن ضبط المكونات تختص كلها بالبلوكات التنظيمية OB ، أما الثاني فهو داخل الدورة الرئيسية للبرنامج أو داخل البلوك المنطقي نفسه حيث يتم كسر التنفيذ المتسلسل بالانتقال لمنطقة محددة وبالطبع يكون هذا اعتمادا على شرط محدد فطالما تحقق شرط الانتقال يتم قطع التسلسل الطبيعي للبرنامج والانتقال إلى المكان المحدد والذي يتم تحديده عن طريق رمز أو عنوان وحيد في البلوك ويسمى Label وتسمى عملية الانتقال تلك بالقفز jump وتكون داخل البلوك الواحد ويمكن الانتقال داخل نفس الحلقة Network أو خارجها بحدود كما لا يجوز تكرار نفس عنوان أو رمز الانتقال Label وعند الرغبة في الانتقال إلى بلوك آخر نستخدم أوامر أخرى تسمى أوامر الاستدعاء Call وليس التوجيه المنطقي الذي نتكلم عنه الآن ، ومن المنطقي أيضا أن يرتبط هذا الانتقال بخانات المنطق أو خانات الحالة Status Bits والتي تحدثنا عنها سابقا مثل RLO و OV و OS و BR ... إلخ ، فهذه الخانات هي التي تحتوي نتائج تترتب على العمليات المختلفة وبالتالي فإن عمليات التوجيه كلها ستقوم عليها وطبقا للإمكانيات المتاحة

<table>
<tr><th colspan="3">Statement List (STL)</th></tr>
<tr><th colspan="2">Jump Label (JL)</th><th>Unconditional Jump (JU)</th></tr>
<tr>
<td colspan="2">
الصيغة العامة
JL <Label1>
JU <Label2>
JU <Label3>
.
.
JU <Labelx>

<label1>: {instructions}
<label2>: {instructions}
<label3>: {instructions}
.
.
<labelx>: {instructions}
</td>
<td>
Example
<pre>
      L   MB   0
      JL   x001
      JU   x002
      JU   x003
      JU   x004
x001: R   Q   1.2
      R   Q   1.3
      R   Q   1.4
      JU   x005
x002: SET
      =   Q   1.2
      JU   x005
x003: SET
      =   Q   1.3
      JU   x005
x004: SET
      =   Q   1.4
      JU   x005
x005: NOP   0
</pre>
</td>
<td>
الصيغة العامة
JU <Label>
{ Instructions}
<Label> : {instructions}

Example
<pre>
      A   I   1.0
      JU   m001
      AN   I   1.0
      JU   m002
m001: SET
      =   Q   0.1
      JU   m003
m002: SET
      =   Q   0.2
      JU   m003
m003: NOP   0
</pre>
</td>
</tr>
<tr>
<td colspan="2">يقوم الأمر JL بتنفيذ عمليات Jump متعددة وبناء على شرط واحد رقمي بحد أقصى 255 حالة فيتم القفز إلى الرمز <Label2> في حالة القيمة 0 وإلى <Label3> في حالة القيمة 1 وهكذا حتى أقصى قيمة موجودة أكبر من عدد Labels فيتم الانتقال إلى <Label1> والموجود بعد JL
وبالطبع يتم تحميل رقم في حجم Byte داخل ACCU1-L-L قبل العملية يمثل دليل عملية Jump وحسب قيمته يتم الانتقال
ففي المثال حسب قيمة MB0 بدءا من 0 ثم 1 ثم 2 وهكذا يتم الانتقال إلى الرموز التالية بالترتيب</td>
<td>يقوم الأمر JU بالانتقال إلى المكان المحدد بالرمز <Label> دون الاعتماد على أي شروط فطالما وصل التنفيذ إليها يتم الانتقال مباشرة إلى المكان المحدد
في المثال السابق لدينا ثلاثة label وهي m001,m002,m003 وطبقا لطبيعة الأمر والذي لا يعتمد على شرط ينتقل دوما إلى m001 ثم إلى m003 ولا ينتقل أبدا إلى moo2</td>
</tr>
</table>

Statement List (STL)

Conditional Jump (JC)
Jump if RLO=1

الصيغة العامة

JC <Label>
{instructions}
<Label>: {Instructions}

Example

```
      A     I     2.0
      JC    a001
      SET
      =     Q     3.0
      R     Q     3.1
      JC    a002
a001: SET
      =     Q     3.1
      R     Q     3.0
a002: A     I     2.1
      R     Q     3.0
      R     Q     3.1
```

يقوم الأمر JC أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة RLO فإن كانت "1" ينتقل إلى الرمز وإن كانت "0" يستكمل العمل لما بعد الأمر JC مباشرة

Conditional Jump (JCN)
Jump if RLO=0

الصيغة العامة

JCN <Label>
{instructions}
<Label>: {Instructions}

Example

```
      A     I     2.0
      JCN   a001
      SET
      =     Q     3.0
      R     Q     3.1
      JC    a002
a001: SET
      =     Q     3.1
      R     Q     3.0
a002: A     I     2.1
      R     Q     3.0
      R     Q     3.1
```

يقوم الأمر JCN أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة RLO فإن كانت "0" ينتقل إلى الرمز وإن كانت "1" يستكمل العمل لما بعد الأمر JCN مباشرة

Conditional Jump (JCB)
Jump if RLO=1 with BR

الصيغة العامة

JCN <Label>
{instructions}
<Label>: {Instructions}

Example

```
      A     I     2.0
      JCB   a001
      SET
      =     Q     3.0
      R     Q     3.1
      JC    a002
a001: SET
      =     Q     3.1
      R     Q     3.0
a002: A     I     2.1
      R     Q     3.0
      R     Q     3.1
```

يقوم الأمر JC أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة RLO فإن كانت "1" ينتقل إلى الرمز وإلا يكمل ، مع نسخ قيمة RLO في الخانة BR بأي حال

Conditional Jump (JNB)
Jump if RLO=0 with BR

الصيغة العامة

JNB <Label>
{instructions}
<Label>: {Instructions}

Example

```
      A     I     2.0
      JNB   a001
      SET
      =     Q     3.0
      R     Q     3.1
      JC    a002
a001: SET
      =     Q     3.1
      R     Q     3.0
a002: A     I     2.1
      R     Q     3.0
      R     Q     3.1
```

يقوم الأمر JNB أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة RLO فإن كانت "0" ينتقل إلى الرمز وإلا يكمل مع نسخ RLO إلى BR على أي حال

Conditional Jump (JBI)
Jump if BR=1

الصيغة العامة

JBI <Label>
{instructions}
<Label>: {Instructions}

Example

```
      A     I     2.0
      JBI   a001
      SET
      =     Q     3.0
      R     Q     3.1
      JC    a002
a001: SET
      =     Q     3.1
      R     Q     3.0
a002: A     I     2.1
      R     Q     3.0
      R     Q     3.1
```

يقوم الأمر JBI أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة BR فإن كانت "0" ينتقل إلى الرمز وإن كانت "1" يستكمل العمل لما بعد الأمر JBI مباشرة

Conditional Jump (JNBI)
Jump if BR=0

الصيغة العامة

JNBI <Label>
{instructions}
<Label>: {Instructions}

Example

```
      A     I     2.0
      JNBI  a001
      SET
      =     Q     3.0
      R     Q     3.1
      JC    a002
a001: SET
      =     Q     3.1
      R     Q     3.0
a002: A     I     2.1
      R     Q     3.0
      R     Q     3.1
```

يقوم الأمر JNBI أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة BR فإن كانت "0" ينتقل إلى الرمز وإن كانت "1" يستكمل العمل لما بعد الأمر JNBI مباشرة

Statement List (STL)

Conditional Jump (JO)
Jump if OV=1

الصيغة العامة

JO <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     DB10.DBD   10
      JO    b001
      T     QD     12
      JU    b002
b001: L     L#0
      T     QD     12
b002: NOP   0
```

يقوم الأمر JO أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة OV فإن كانت "1" ينتقل إلى الرمز وإن كانت "0" يستكمل العمل لما بعد الأمر JO مباشرة

Conditional Jump (JOS)
Jump if OS=0

الصيغة العامة

JOS <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     DB10.DBD   10
      JOS   b001
      T     QD     12
      JU    b002
b001: L     L#0
      T     QD     12
b002: NOP   0
```

يقوم الأمر JOS أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة OS فإن كانت "0" ينتقل إلى الرمز وإن كانت "1" يستكمل العمل لما بعد الأمر JOS مباشرة

Conditional Jump (JZ)
Jump if Zero CC0=CC1=0

الصيغة العامة

JZ <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     IW     2
      SRW
      JZ    c001
      L     0
      T     QW     12
      JU    c002
c001: L     IW     2
      T     QW     12
c002: NOP   0
```

يقوم الأمر JZ أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة CC0,CC1 فإن كانتا معا "0" ينتقل إلى الرمز وإلا يكمل لما بعد الأمر JZ مباشرة

Conditional Jump (JN)
Jump if CC0 and CC1 not 0

الصيغة العامة

JN <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     MW     200
      L     DB10.DBD   20
      OW
      JN    d001
      L     0
      T     MW     220
      JU    d001
d001: L     MW     200
      T     MW     220
d002: NOP   0
```

يقوم الأمر JN أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة CC0,CC1 فإن كانتا مختلفتين ينتقل إلى الرمز وإن كانتا معا "0" يكمل لما بعد الأمر JN مباشرة

Conditional Jump (JP)
Jump if Plus

الصيغة العامة

JP <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     MW     200
      L     DB10.DBD   20
      OW
      JP    d001
      L     0
      T     MW     220
      JU    d001
d001: L     MW     200
      T     MW     220
d002: NOP   0
```

يقوم الأمر JP أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة CC0,CC1 فإن كانت CC1=1,CC0=0 ينتقل إلى الرمز وإلا يكمل لما بعد الأمر JP مباشرة

Conditional Jump (JM)
Jump if Minus

الصيغة العامة

JM <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     MW     200
      L     DB10.DBD   20
      OW
      JM    d001
      L     0
      T     MW     220
      JU    d001
d001: L     MW     200
      T     MW     220
d002: NOP   0
```

يقوم الأمر JM أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة CC0,CC1 فإن كانت CC0=1,CC1=0 ينتقل إلى الرمز وإلا يكمل لما بعد الأمر JM مباشرة

Statement List (STL)

Conditional Jump (JPZ)
Jump if Plus or Zero

الصيغة العامة

JPZ <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     MW    200
      L     DB10.DBD   20
      OW
      JPZ   d001
      L     0
      T     MW    220
      JU    d001
d001: L     MW    200
      T     MW    220
d002: NOP   0
```

يقوم الأمر JPZ أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة CC0,CC1 فإن كانت CC1=1,CC0=0 أو CC0=CC1=0 ينتقل إلى الرمز وإلا يكمل لما بعد الأمر JPZ مباشرة

Conditional Jump (JMZ)
Jump if Minus or Zero

الصيغة العامة

JMZ <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     MW    200
      L     DB10.DBD   20
      OW
      JMZ   d001
      L     0
      T     MW    220
      JU    d001
d001: L     MW    200
      T     MW    220
d002: NOP   0
```

يقوم الأمر JMZ أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة CC0,CC1 فإن كانت CC0=1,CC1=0 أو CC0=CC1=0 ينتقل إلى الرمز وإلا يكمل لما بعد الأمر JMZ مباشرة

Conditional Jump (JUO)
Jump if Un-Ordered

الصيغة العامة

JUO <Label>
{instructions}
<Label>: {Instructions}

Example

```
      L     MW    200
      L     DB10.DBD   20
      OW
      JUO   d001
      L     0
      T     MW    220
      JU    d001
d001: L     MW    200
      T     MW    220
d002: NOP   0
```

يقوم الأمر JUO أو الانتقال المشروط بالانتقال إلى الرمز المحدد بناء على حالة CC0,CC1 فإن كانتا معا "1" أي CC0=CC1=1 ينتقل إلى الرمز وإلا يكمل لما بعد الأمر JUO مباشرة

130

Conditional Jump (LOOP)
LOOP if ACCU1-L<>0

الصيغة العامة

<Label>: {Instructions}
LOOP <Label>
{instructions}

Example

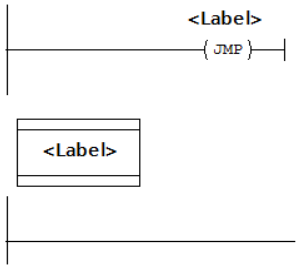
```
      L     L#1
      T     MD    20
      L     5
NEXT: T     MW    10
      L     MD    20
      AD
      T     MD    20
      L     MW    10
      LOOP  NEXT
      L     MW    24
      T     QW    14
```

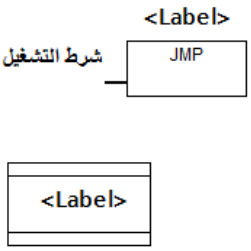
يقوم الأمر LOOP بإنقاص محتويات ACCU1-L بمقدار-1 وطالما أن القيمة لا تساوي "0" يقوم بالانتقال إلى العنوان الرمزي <Label> وعندما تصل القيمة إلى "0" يكمل لما بعد الأمر LOOP

Ladder Diagram (LAD)	Function Block Diagram (FBD)

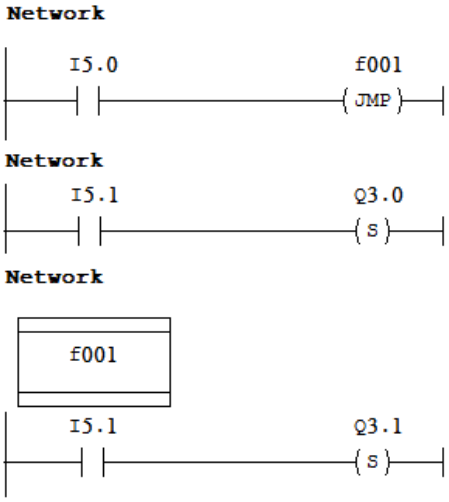
Conditional Jump (JMP)

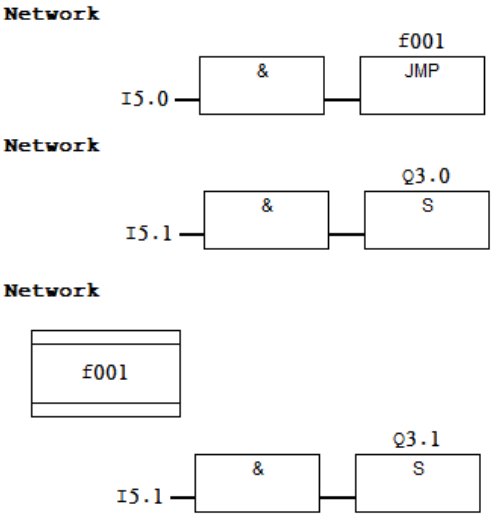
الصيغة العامة





Example

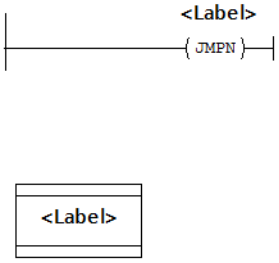


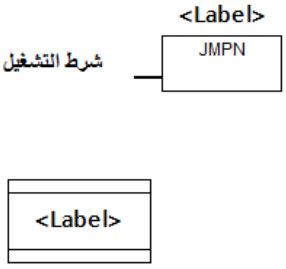


131 يقوم الأمر JMP بالانتقال إلى الرمز المحدد <Label> بناء على حالة RLO=1 وإلا يكمل لما بعده مباشرة

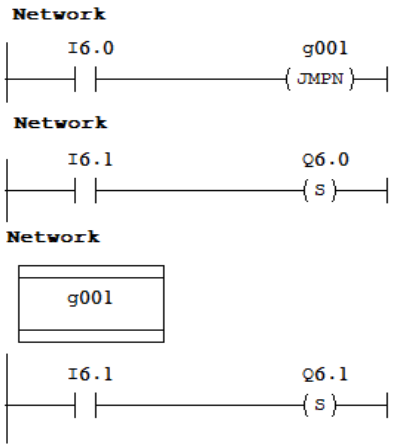
Conditional Jump (JMPN)

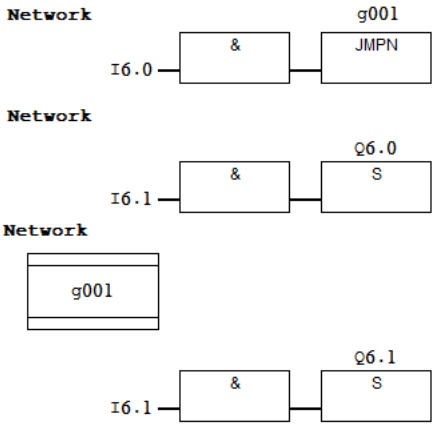
الصيغة العامة





Example





يقوم الأمر JMPN بالانتقال إلى الرمز المحدد <Label> بناء على حالة RLO=0 وإلا يكمل لما بعده مباشرة

تعليمات التحكم في البرنامج Program Control Instructions:

كما يمكن الانتقال من جزء إلى جزء داخل البلوك باستخدام أوامر Jump بمختلف أشكالها فإنه يمكن أيضا التحكم في الانتقال بين الأنواع المختلفة من البلوك وذلك عن طريق استدعاء بلوك آخر Call فيتم انتقال التنفيذ إلى البلوك المستدعى أو يمكن إنهاء بلوك وبالتالي ينتقل التنفيذ إلى ما يليه في OB1 وهكذا وهذه المجموعة من التعليمات يطلق عليها تعليمات التحكم في البرنامج Program Control Instructions

<table>
<tr><th colspan="3">Statement List (STL)</th></tr>
<tr><th>Block End (BE)</th><th>Block End Conditional (BEC)</th><th>Block End Un-Conditional (BEU)</th></tr>
<tr><td>الصيغة العامة
BE
Example
<pre>
      AN   I   7.0
      JC   m001
      BE
m001: =   Q   5.0
</pre></td><td>الصيغة العامة
BEC
Example
<pre>
      A   I   7.1
      BEC
      =   Q   5.1
</pre></td><td>الصيغة العامة
BEU
Example
<pre>
      A   I   7.2
      BEU
      AN   I   7.3
      BEC
</pre></td></tr>
<tr><td>يقوم الأمر BE بإنهاء عمل البلوك الحالي والانتقال بالتشغيل للبلوك التالي في العمل</td><td>يقوم الأمر BEC بإنهاء عمل البلوك الحالي بناء على حالة RLO=1 وينتقل للبلوك التالي وإلا يكمل العمل داخل البلوك</td><td>يقوم الأمر BEU بإنهاء عمل البلوك الحالي بغض النظر عن حالة RLO ومهما كانت حالتها طالما مر عليها التنفيذ</td></tr>
<tr><th>Call Block (CALL)</th><th>Call Block (CALL)</th><th>Call Block (CALL)</th></tr>
<tr><td>الصيغة العامة
CALL <Block Identifier>
CALL FCn
CALL FBn, DBn
CALL SFCn
CALL SFBn, DBn
Example1
<pre>
CALL FC   16
CALL FC   1
 a:=DB10.DBD0
 b:=DB10.DBD4
 c:=DB10.DBD8
 y:=DB10.DBD12
CALL FB   1 , DB30
 a:=MD10
 b:=MD14
 c:=MD18
 y:=MD22
</pre></td><td>Example2
<pre>
CALL 'SCALE'   FC105
 IN   :=PIW340
 HI_LIM :=2.000000e+002
 LO_LIM :=0.000000e+000
 BIPOLAR:=M10.0
 RET_VAL:=DB10.DBW20
 OUT   :=DB10.DBD22
</pre></td><td>Example3
<pre>
CALL 'TCONT_S' , DB55   FB59
 CYCLE   :=
 SP_INT :=
 PV_IN   :=
 PV_PER :=
 DISV   :=
 LMNR_HS:=
 LMNR_LS:=
 LMNS_ON:=
 LMNUP   :=
 LMNDN   :=
 QLMNUP :=
 QLMNDN :=
 PV   :=
 ER   :=
 COM_RST:=
NOP   0
</pre></td></tr>
<tr><td colspan="3">يقوم الأمر CALL باستدعاء بلوك معين سواء كان FC أو FB أو SFC أو SFB مع تمرير البيانات المطلوبة لكل بلوك إن كان له بيانات مطلوبة مثل FC1 في المثال-1 و FC105 من المكتبة في المثال-2 مع ذكر البلوك المصاحب Instance data block مع كل FB مثل DB30 مع FB1 في المثال-1 و DB55 مع FB59 في المثال-3 مع تمرير البيانات المطلوبة في المثال رقم-1 لأنها ليست في Data Block بينما لا يلزم هذا في المثال-3 لأن البيانات جزء من بلوك البيانات المصاحب DB55 ، كما يمكن استدعاء البلوك بالاسم الرمزي مثل SCALE في المثال-2 و TCONT_S في المثال رقم-3 كما يمكن استدعاء البلوك برقمه كالمعتاد كما في المثال رقم-1 كما أن الأمر CALL لا ينطبق إلا على البلوكات المنطقية فقط من الأنواع السابقة ولا ينطبق على بلوكات البيانات ولا بلوكات التنظيم OB والتي ترتبط بشكل خاص بجدول التنفيذ طبقا لنوع المقاطعة Interrupt الخاص بها</td></tr>
</table>

Condition Call Block (CC)	*Un-Conditional Call Block (UC)*	
الصيغة العامة CC <Block Identifier> CC FCn CC FBn **Example** A I 8.0 CC FC 1 CC FB 1 AN I 8.0 CC FC 2 CC FB 2	**الصيغة العامة** CC <Block Identifier> CC FCn CC FBn **Example** UC FC 1 UC FB 1	
يقوم الأمر CC باستدعاء بلوك معين سواء كان FC أو FB بناء على حالة RLO=1 بدون تمرير بيانات إلى البلوك أو استخدام بلوك بيانات مصاحب للبلوكات FB ولهذا يراعى استخدامها للبلوكات التي لا يتم فيها تمرير بيانات من النوعين أما الأمر UC فيقوم باستدعاء بلوك سواء كان FC أو FB بغض النظر عن حالة RLO ولا يقوم بتمرير بيانات أو إدخال رقم بلوك البيانات		

استخدام ريلاي التحكم القائد Master Control Relay:

في أنظمة التحكم المعتاد يمكن استخدام ريلاي بشكل معين ليكون شرطا في تشغيل أجزاء كثيرة بشرط تشغيله هو وهذا ما يطلق عليه الريلاي القائد Master Control Relay ويمكن استخدامه داخل البرنامج لتفعيل جزء معين أو عدم تفعيله بناء على عمل أو عدم عمل MCR

لاحظ أنه في المنطقة غير المفعلة Deactivating Rungs يتم الآتي:

- تصفير جميع الذواكر الغير دائمة Non-retentive
- الاحتفاظ بقيمة جميع الذواكر الدائمة Retentive

ولا يمكن إعادة تفعيل الذواكر الغير مفعلة إلا بعد تفعيل MCR مرة أخرى

وعلى هذا فإنه لابد من تفعيل دور MCR قبل البدء في أي عمل داخل نطاق MCR ثم إلغاء التفعيل بعد الانتهاء من عمله أما الأجزاء التي يتم تنفيذها في نطاق MCR فيجب تحديدها بفتح نطاق ثم غلقه بعد انتهاء مجموعة التعليمات التي تستخدم داخل هذا النطاق

- تفعيل الريلاي القائد MCR-Activate ويستخدم له الأمر MCRA
- إلغاء تفعيل الريلاي القائد MCR-Deactivate ويستخدم له الأمر MCRD
- فتح نطاق لأوامر على الريلاي القائد MCR-Open ويستخدم له الأمر)MCR
- غلق نطاق لأوامر على الريلاي القائد MCR-Close ويستخدم له الأمر (MCR

يمكن استخدام نطاقات متداخلة من نطاقات MCR وهناك تحذير شديد من استعمال أوامر MCR فيما يخص شروط الأمان وتوقف الطوارئ Emergency Stop في الماكينات كبديل عن الأنظمة الكهربية المعتادة والمعتمدة على مفاتيح وريليهات وتوصيلات سلكية حقيقية

Statement List (STL)		
MCR Activate (MCRA)	***MCR Deactivate (MCRD)***	***Open MCR Area (MCR( )*** ***Close MCR Area (MCR))***
الصيغة العامة MCRA **Example** A I 1.0 MCRA	**الصيغة العامة** MCRD **Example** A I 1.1 MCRD	**الصيغة العامة** MCR(**Example** A I 1.0 MCRA MCR(A I 2.0 = M1.2 L PIW420 T DB10.DBW20 MCR)
يقوم الأمر MCRA بتفعيل عمل الريلاي القائد MCR حيث يمكن بعد ذلك توظيف مناطق استخدامه	يقوم الأمر MCRD بإنهاء عمل الريلاي القائد وتصفير جميع الذواكر المؤقتة non-retentive مع الاحتفاظ بقيم الذواكر retentive	يقوم الأمر)MCR بفتح منطقة عمل الريلاي القائد MCR لتنفيذ مجموعة من التعليمات تكون مرتبطة بالريلاي MCR وينتهي عملها بالأمر (MCR وتعمل هذه المنطقة بشرط تفعيل MCR

العمليات الحسابية Mathematic Instructions:

العمليات الحسابية تشكل جزءا كبيرا ومهما وإضافة كبيرة لعمليات التحكم المنطقي خاصة بعد إضافة العمليات الحسابية على الأرقام الحقيقية لها والتي أضافت دقة كبيرة في الحسابات إلى كسور تصل إلى 38 رقم عشري وإلى أرقام كبيرة تصل لنفس العدد تقريبا بالإضافة إلى إمكانية التحويل بين الصيغ المختلفة للأرقام والتي أتاحت التحويل عند الحاجة إلى دقة عالية للانتقال إلى عمليات على أرقام حقيقية ، كذلك هناك مجموعة أساسية من العمليات الهندسية تسمح بتكوين باقي العمليات الفرعية بشكل ما وبالتالي فسوف نجد العمليات الحسابية الأساسية على طرق تمثيل الأرقام الثلاثة ، الصحيحة Integers والصحيحة المضاعفة الدقة Double Integers والحقيقية Real وهذه العمليات الأساسية هي الجمع والطرح والضرب والقسمة على النحو التالي:

- جمع الأرقام الصحيحة (I+) والطرح (I-) والضرب (I*) والقسمة (I/).
- جمع الأرقام الصحيحة مضاعفة الدقة (D+) والطرح (D-) والضرب (D*) والقسمة (D/) وباقي القسمة (MOD)
- جمع الأرقام الحقيقية (R+) والطرح (R-) والضرب (R*) والقسمة (R/)

أما عن العمليات الإضافية الهندسية والتي تم تمثيلها في Step7 فهي :

1- ABS وتمثل القيمة المطلقة
2- SQR وتمثل مربع الرقم
3- SQRT وتمثل الجذر التربيعي للرقم
4- LN وتمثل اللوغاريتم الطبيعي للرقم
5- EXP وتمثل القوة الأسية للرقم
6- SIN وتمثل جيب الزاوية بالتقدير الدائري
7- COS وتمثل جيب تمام الزاوية بالتقدير الدائري
8- TAN وتمثل ظل الزاوية بالتقدير الدائري
9- ASIN وتمثل الزاوية بالتقدير الدائري لرقم يمثل جيب الزاوية
10- ACOS وتمثل الزاوية بالتقدير الدائري لرقم يمثل جيب تمام الزاوية
11- ATAN وتمثل الزاوية بالتقدير الدائري لرقم يمثل ظل الزاوية

العمليات الأساسية والتي تحتاج إلى معاملين two operands تتم على محتويات ACCU1,ACCU2 مع تخزين نتيجة العملية في ACCU1 وتشمل عمليات الجمع والطرح والضرب والقسمة وباقي القسمة للأرقام الصحيحة مضاعفة الدقة فيجب تحميل الأرقام إلى ACCU1,ACCU2 قبل تنفيذ العملية على النحو التالي:

- الجمع (ACCU2+ACCU1 → ACCU1)
- الطرح (ACCU2-ACCU1 → ACCU1)
- الضرب (ACCU2*ACCU1 → ACCU1)
- القسمة (ACCU2/ACCU1 → ACCU1)
- باقي القسمة (ACCU2/ACCU1 → ACCU1)

مع ملاحظة أنه في حالة العمليات على الأرقام الصحيحة Integers يتم ذلك على Low-word من ACCU1,ACCU2 بينما في حالة REAL,DINT يتم ذلك على كل من ACCU1,ACCU2

أما العمليات الهندسية فكلها تتم على REAL والنتيجة لها تكون REAL وتتم على محتويات ACCU1 ويتم تخزين النتيجة في ACCU1

وجميع العمليات الحسابية تؤثر على خانات OV,OS,CC0,CC1 في خانات STATUS Bits

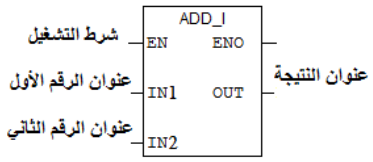
ففي حالة Overflow أو زيادة النتيجة عن السعة التخزينية ل ACCU1 تتحول حالة OV إلى "1" حتى العملية القادمة بينما تتحول قيمة OS إلى "1" إلى أن يتم تصفيرها وتتعلق حالة CC0,CC1 بالفارق بين محتوى ACCU1,ACCU2

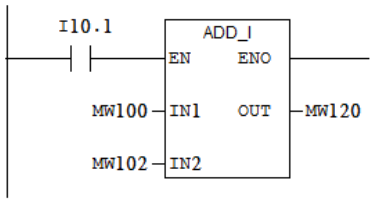
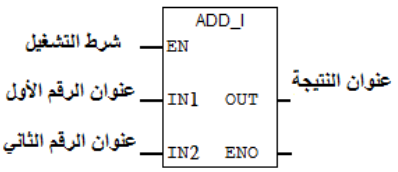
<u>أولا : العمليات على الأرقام الصحيحة:</u>

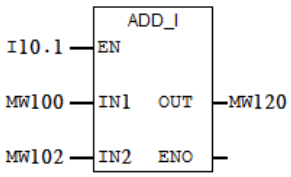
<table>
<tr><th colspan="3">Adding Integers : +I</th></tr>
<tr><th>Statement List (STL)</th><th>Ladder Diagram (LAD)</th><th>Function Block Diagram (FBD)</th></tr>
<tr>
<td>الصيغة العامة

+I

<u>Example</u>

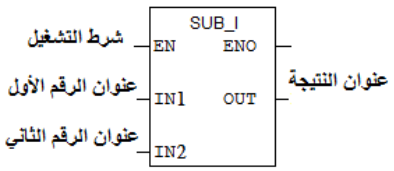
L MW 100
L MW 102
+I
T MW 120</td>
<td>الصيغة العامة
 

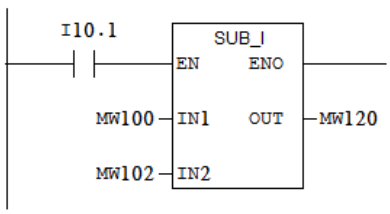
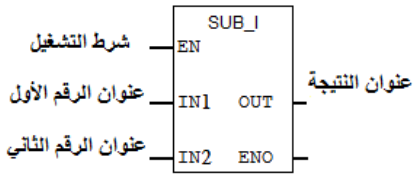
<u>Example</u>
 
</td>
<td>الصيغة العامة
 

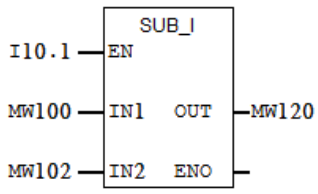
<u>Example</u>
 
</td>
</tr>
<tr><td colspan="3">يقوم الأمر I+ بإضافة محتويات ACCU1-Low أو عنوان الرقم الأول إلى محتويات ACCU2-Low أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1-Low أو عنوان النتيجة (مع اعتبار صيغة التخزين INT)</td></tr>
<tr><th colspan="3">Subtracting Integers : -I</th></tr>
<tr><th>Statement List (STL)</th><th>Ladder Diagram (LAD)</th><th>Function Block Diagram (FBD)</th></tr>
<tr>
<td>الصيغة العامة

-I

<u>Example</u>

L MW 100
L MW 102
-I
T MW 120</td>
<td>الصيغة العامة
 

<u>Example</u>
 
</td>
<td>الصيغة العامة
 

<u>Example</u>
 
</td>
</tr>
<tr><td colspan="3">يقوم الأمر I- بطرح محتويات ACCU1-Low أو عنوان الرقم الثاني من محتويات ACCU2-Low أو عنوان الرقم الأول وتخزين النتيجة في ACCU1-Low أو عنوان النتيجة (مع اعتبار صيغة التخزين INT)</td></tr>
</table>

Multiplying Integers : *I

Statement List (STL)

الصيغة العامة

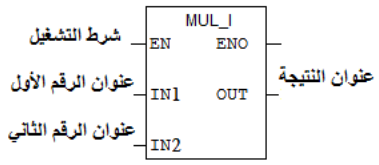
*I

Example

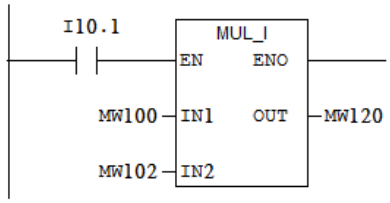
```
L     MW    100
L     MW    102
*I
T     MW    120
```

Ladder Diagram (LAD)

الصيغة العامة

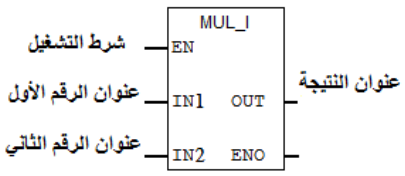


Example

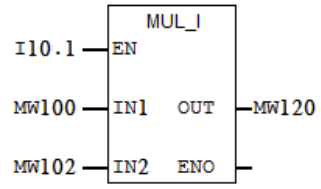


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر *I بضرب محتويات ACCU1-Low أو عنوان الرقم الأول في محتويات ACCU2-Low أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1-Low أو عنوان النتيجة (مع اعتبار صيغة التخزين INT)

Dividing Integers : /I

Statement List (STL)

الصيغة العامة

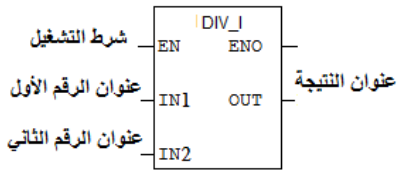
/I

Example

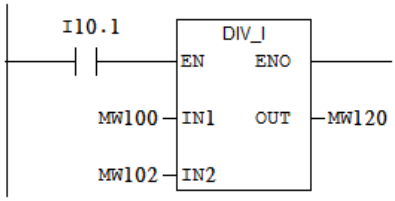
```
L     MW    100
L     MW    102
/I
T     MW    120
```

Ladder Diagram (LAD)

الصيغة العامة

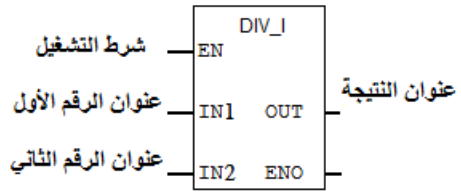


Example

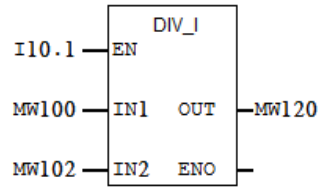


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر /I بقسمة محتويات ACCU2-Low أو عنوان الرقم الأول على محتويات ACCU1-Low أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1-Low أو عنوان النتيجة (مع اعتبار صيغة التخزين INT)

137

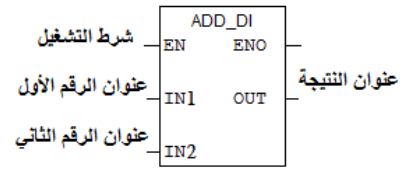
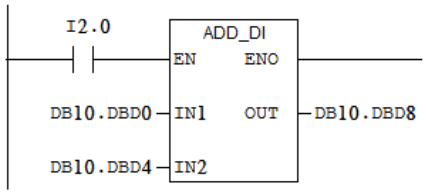
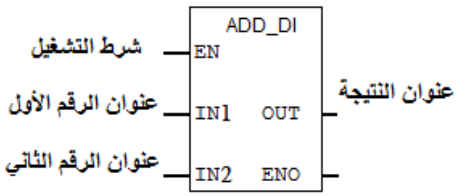
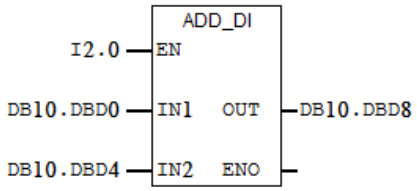
لاحظ أن عمليات الجمع والضرب والطرح قد ينتج عنها زيادة عن السعة التخزينية Overflow فمثلا لو جمعنا 25000 على 10000 سينتج لدينا 35000 وهو أكبر من السعة التخزينية ، كذلك لو طرحنا الرقم 10000 من الرقم 25000- فسوف نحصل على الرقم 35000- وهو أيضا أكبر من السعة التخزينية أما عمليات الضرب فهي أقرب كثيرا فمثلا لو ضربنا 500 في 700 فسوف نحصل على 350000 وهو أكبر كثيرا من السعة التخزينية

أما عمليات القسمة فإنها تمثل مشكلة من ناحيتين الأولى وهي الدقة حيث لو قسمنا 100 على 99 فالنتيجة تكون 1 ولو قسمنا 100 على 75 فالنتيجة أيضا 1 ولو قسمنا 100 على 51 فالنتيجة أيضا 1 ، أما الناحية الأخرى فهي عند القسمة على صفر فالنتيجة غير محددة

ثانيا : العمليات على الأرقام الصحيحة مضاعفة الدقة DINT:

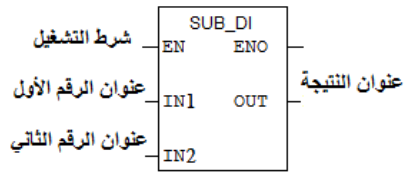
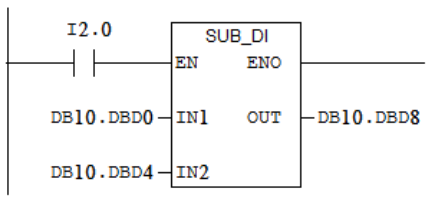
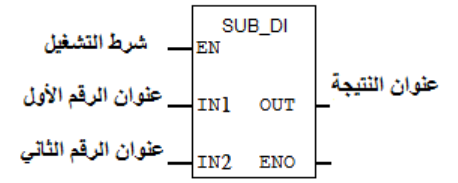
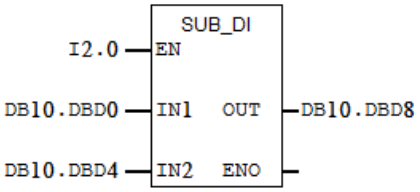
تتم العمليات الأساسية على الأرقام الصحيحة مضاعفة الدقة تماما مثل الأرقام الصحيحة الأربعة الجمع والطرح والضرب والقسمة ولكن على مستوى 32 خانة مع عملية خامسة إضافية وهي باقي القسمة حيث تقوم العملية بقسمة محتويات ACCU2 على ACCU1 وتخزين باقي القسمة في ACCU1 وفيما يلي العمليات الخمسة تفصيليا

Adding Double Integers : +D

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة +D **Example** L DB10.DBD 0 L DB10.DBD 4 +D T DB10.DBD 8	**الصيغة العامة**  **Example** 	**الصيغة العامة**  **Example** 

يقوم الأمر +D بإضافة محتويات ACCU1 أو عنوان الرقم الأول إلى محتويات ACCU2 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين DINT)

Subtracting Double Integers : -D

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة -D **Example** L DB10.DBD 0 L DB10.DBD 4 -D T DB10.DBD 8	**الصيغة العامة**  **Example** 	**الصيغة العامة**  **Example** 

يقوم الأمر -D بطرح محتويات ACCU1 أو عنوان الرقم الأول من محتويات ACCU2 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين DINT)

Multiplying Double Integers : *D

Statement List (STL)

الصيغة العامة

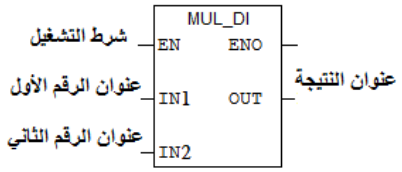
*D

Example

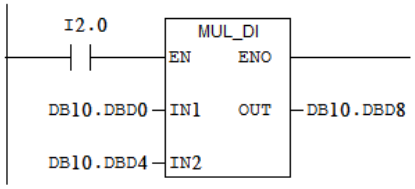
```
L     DB10.DBD   0
L     DB10.DBD   4
*D
T     DB10.DBD   8
```

Ladder Diagram (LAD)

الصيغة العامة

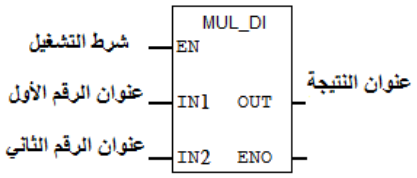


Example

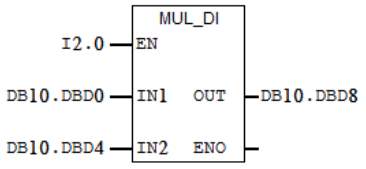


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر D* بضرب محتويات ACCU1 في عنوان الرقم الأول إلى محتويات ACCU2 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين DINT)

Dividing Double Integers : /D

Statement List (STL)

الصيغة العامة

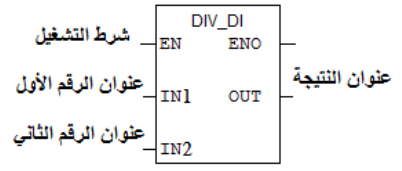
/D

Example

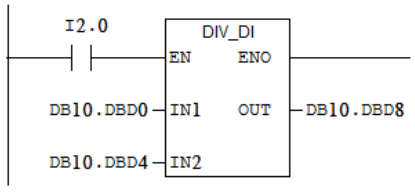
```
L     DB10.DBD   0
L     DB10.DBD   4
/D
T     DB10.DBD   8
```

Ladder Diagram (LAD)

الصيغة العامة

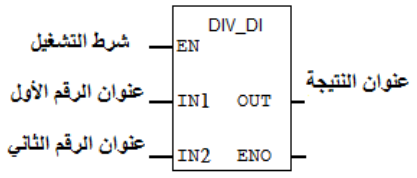


Example

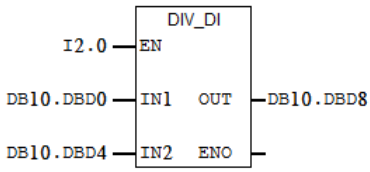


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر D/ بقسمة محتويات ACCU2 أو عنوان الرقم الأول على محتويات ACCU1 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين DINT)

Division Remainder Double Integers : /D

Statement List (STL)

الصيغة العامة

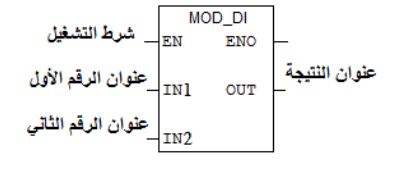
MOD

Example

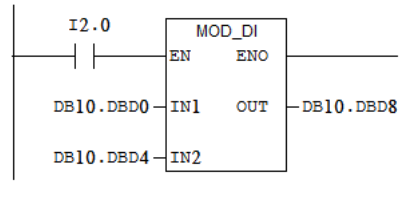
```
L     DB10.DBD   0
L     DB10.DBD   4
MOD
T     DB10.DBD   8
```

Ladder Diagram (LAD)

الصيغة العامة

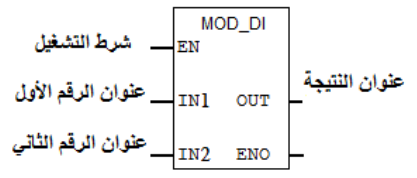


Example

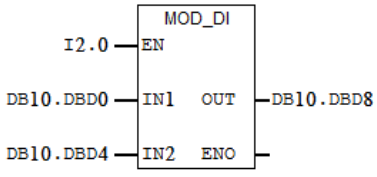


Function Block Diagram (FBD)

الصيغة العامة



Example



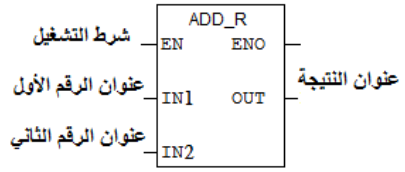
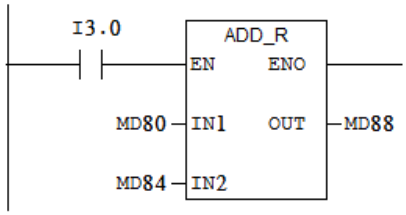
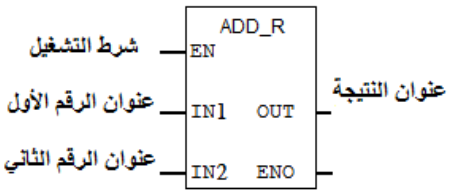
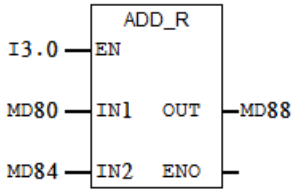
يقوم الأمر D/ بقسمة محتويات ACCU2 أو عنوان الرقم الأول على محتويات ACCU1 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين DINT)

139

ثالثا : العمليات على الأرقام الحقيقية (العشرية) Floating Point (Real) Numbers:

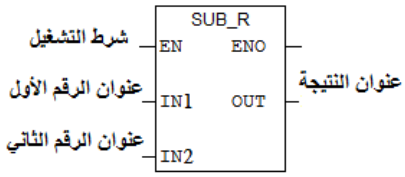
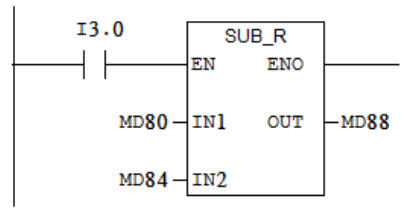
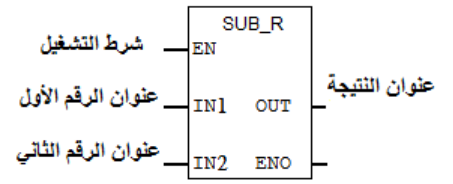
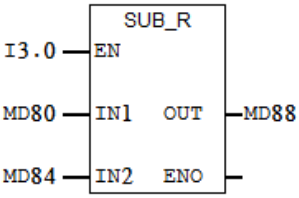
تستخدم الأرقام العشرية عند إجراء حسابات تستدعي دقة عالية أو تحتوي على حسابات هندسية مثل المربع واللوغاريتم والجزر التربيعي والدوال الجيبية المثلثية إلى جانب العمليات الأساسية وهو ما سنستعرضه معا في السطور التالية

Adding Real Numbers : +R

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة +R **Example** L MD 80 L MD 84 +R T MD 88	**الصيغة العامة**  **Example** 	**الصيغة العامة**  **Example** 

يقوم الأمر +R بإضافة محتويات ACCU1 أو عنوان الرقم الأول إلى محتويات ACCU2 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين REAL)

Subtracting Real Numbers : -R

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة -R **Example** L MD 80 L MD 84 -R T MD 88	**الصيغة العامة**  **Example** 	**الصيغة العامة**  **Example** 

يقوم الأمر -R بطرح محتويات ACCU1 أو عنوان الرقم الأول من محتويات ACCU2 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين REAL)

Multiplying Real Numbers : *R

Statement List (STL)

الصيغة العامة

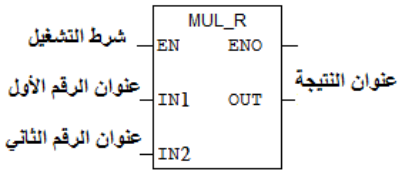
*R

Example

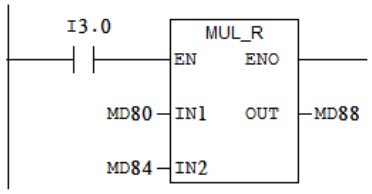
```
L     MD    80
L     MD    84
*R
T     MD    88
```

Ladder Diagram (LAD)

الصيغة العامة

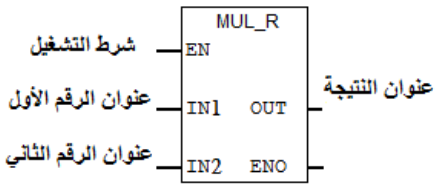


Example

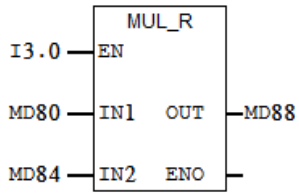


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر R* بضرب محتويات ACCU1 أو عنوان الرقم الأول في محتويات ACCU2 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين REAL)

Dividing Real Numbers : /R

Statement List (STL)

الصيغة العامة

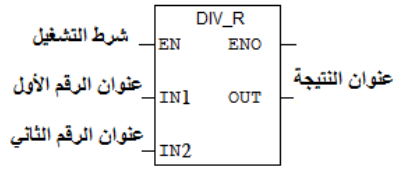
/R

Example

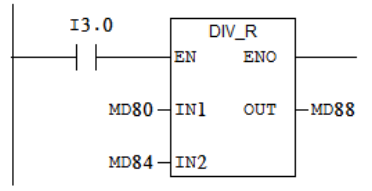
```
L     MD    80
L     MD    84
/R
T     MD    88
```

Ladder Diagram (LAD)

الصيغة العامة

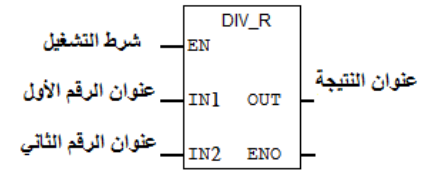


Example

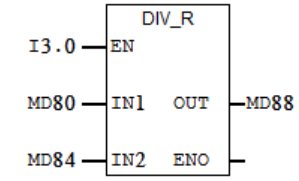


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر /R بقسمة محتويات ACCU2 أو عنوان الرقم الأول على محتويات ACCU1 أو عنوان الرقم الثاني وتخزين النتيجة في ACCU1 أو عنوان النتيجة (مع اعتبار صيغة التخزين REAL)

141

Absolute Value : ABS

Statement List (STL)

الصيغة العامة

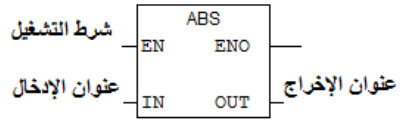
ABS

Example

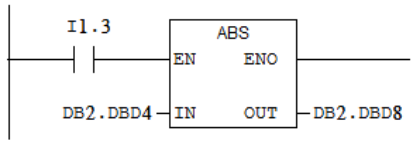
```
L     DB2.DBD    4
ABS
T     DB2.DBD    8
```

Ladder Diagram (LAD)

الصيغة العامة

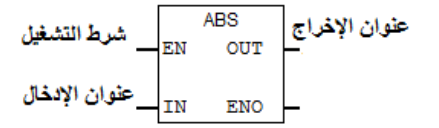


Example

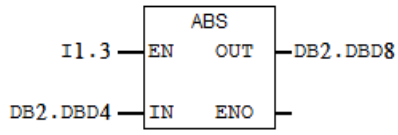


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر ABS بإيجاد القيمة الموجبة للرقم المخزن في ACCU1 مع إلغاء الإشارة وتخزين القيمة الرقمية الموجبة في ACCU1 (مع اعتبار صيغة التخزين REAL)

Square value : SQR

Statement List (STL)

الصيغة العامة

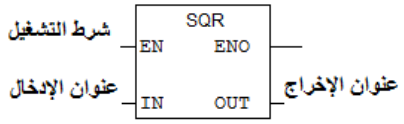
SQR

Example

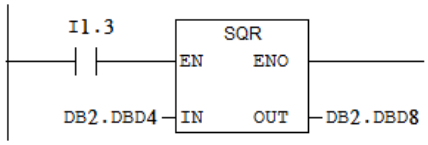
```
L     DB2.DBD    4
SQR
T     DB2.DBD    8
```

Ladder Diagram (LAD)

الصيغة العامة

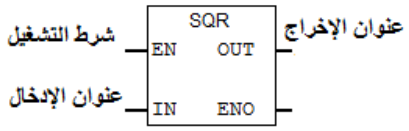


Example

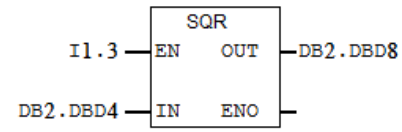


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر SQR بإيجاد مربع (حاصل ضرب الرقم × نفسه) الرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL)

Square Root value : SQRT

Statement List (STL)

الصيغة العامة

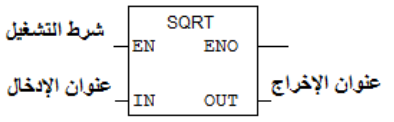
SQRT

Example

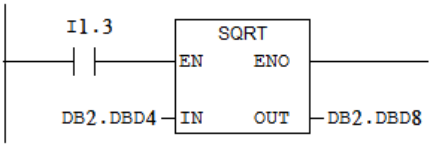
```
L     DB2.DBD    4
SQRT
T     DB2.DBD    8
```

Ladder Diagram (LAD)

الصيغة العامة

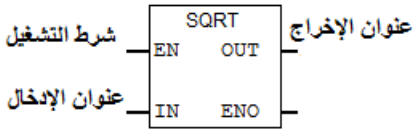


Example

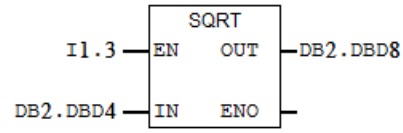


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر SQRT بإيجاد الجذر التربيعي للرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL)

142

Exponential Value : EXP

Statement List (STL)

الصيغة العامة

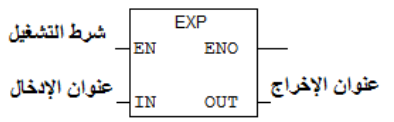
EXP

Example

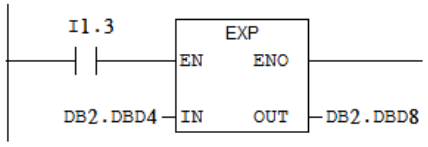
```
L     DB2.DBD    4
EXP
T     DB2.DBD    8
```

Ladder Diagram (LAD)

الصيغة العامة

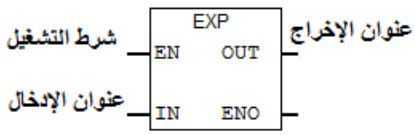


Example

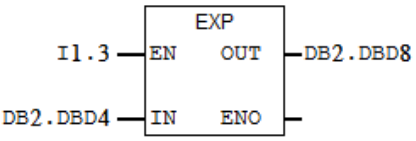


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمرEXP بإيجاد القيمة الأسية للأساس (e) الثابت الطبيعي (2.71828) للرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL)

Natural Logarithm Value : LN

Statement List (STL)

الصيغة العامة

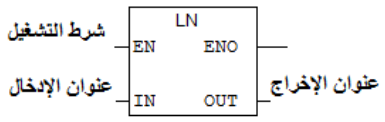
LN

Example

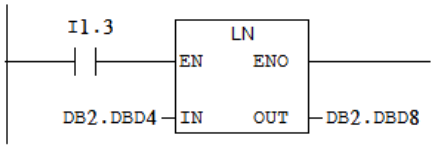
```
L     DB2.DBD   4
LN
T     DB2.DBD   8
```

Ladder Diagram (LAD)

الصيغة العامة

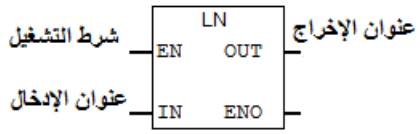


Example

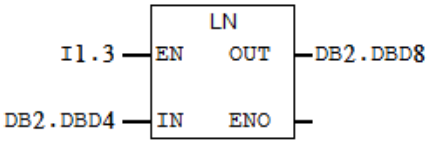


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمرLN بإيجاد اللوغاريتم للأساس (e) الثابت الطبيعي (2.71828) للرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL)

Sin(x) of Angle : SIN

Statement List (STL)

الصيغة العامة

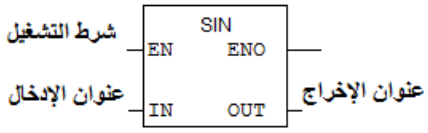
SIN

Example

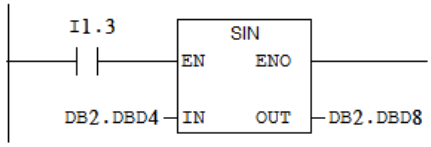
```
L     DB2.DBD   4
SIN
T     DB2.DBD   8
```

Ladder Diagram (LAD)

الصيغة العامة

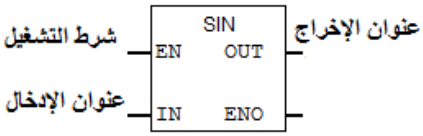


Example

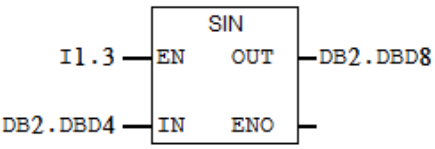


Function Block Diagram (FBD)

الصيغة العامة



Example



يقوم الأمر SIN بإيجاد جيب الزاوية للرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL) وحساب الزوايا يكون بالتقدير الدائري أي قيمة الزاوية × ط / 180

143

Cos(x) of Angle : COS

Statement List (STL)

الصيغة العامة

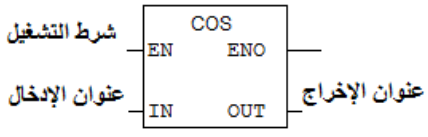
COS

Example

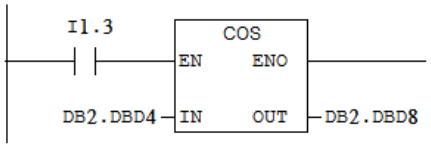
```
L     DB2.DBD   4
COS
T     DB2.DBD   8
```

Ladder Diagram (LAD)

الصيغة العامة

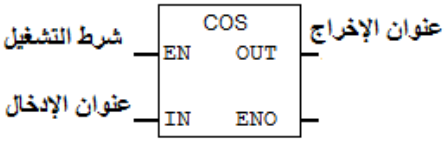


Example

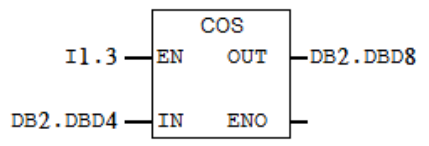


Function Block Diagram (FBD)

الصيغة العامة

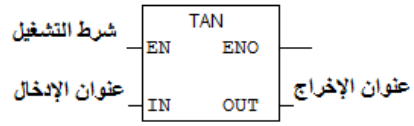
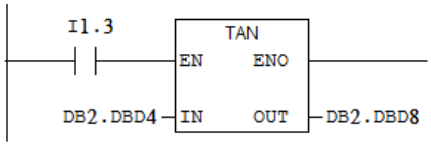
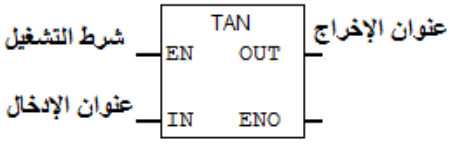
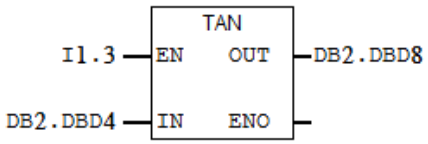


Example



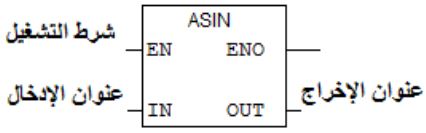
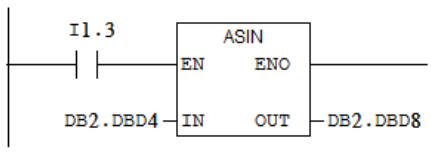
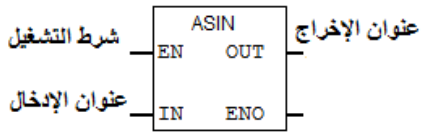
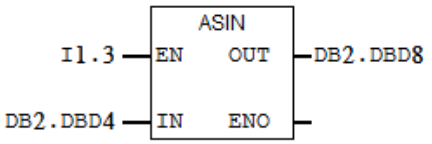
يقوم الأمر COS بإيجاد جيب تمام الزاوية للرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL) وحساب الزوايا يكون بالتقدير الدائري أي قيمة الزاوية × ط / 180

Tan(x) of Angle : TAN

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة TAN **Example** L DB2.DBD 4 TAN T DB2.DBD 8	**الصيغة العامة**  **Example**  	**الصيغة العامة**  **Example** 

يقوم الأمر TAN بإيجاد ظل الزاوية للرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL) وحساب الزوايا يكون بالتقدير الدائري أي قيمة الزاوية × ط / 180

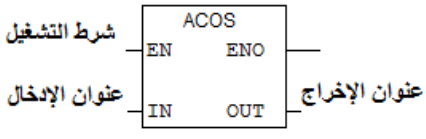
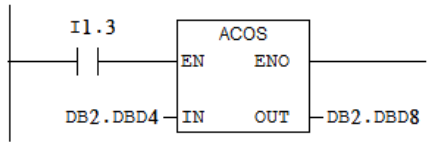
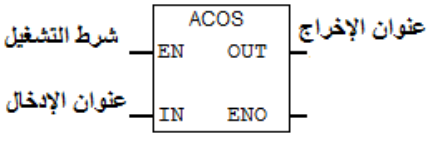
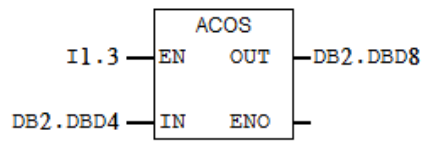
Arc-Sin(x) of Angle : ASIN

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة ASIN **Example** L DB2.DBD 4 ASIN T DB2.DBD 8	**الصيغة العامة**  **Example**  	**الصيغة العامة**  **Example** 

يقوم الأمر ASIN بإيجاد قيمة الزاوية لجيب الزاوية بالرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL) وحساب الزوايا يكون بالتقدير الدائري أي قيمة الزاوية × ط / 180

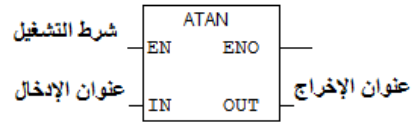
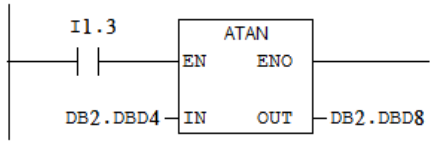
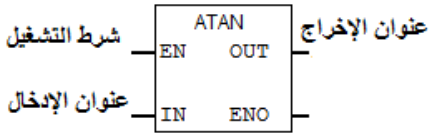
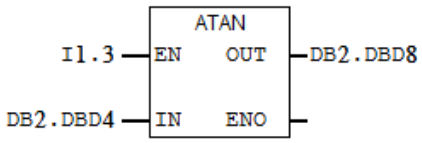
144

Arc-Cos(x) of Angle : ACOS

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة ACOS **Example** L DB2.DBD 4 ACOS T DB2.DBD 8	**الصيغة العامة**  **Example**  	**الصيغة العامة**  **Example** 

يقوم الأمر ACOS بإيجاد قيمة الزاوية لجيب تمام الزاوية بالرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL) وحساب الزوايا يكون بالتقدير الدائري أي قيمة الزاوية × ط / 180

Arc-Tan(x) of Angle : ATAN

Statement List (STL)	Ladder Diagram (LAD)	Function Block Diagram (FBD)
الصيغة العامة ATAN **Example** L DB2.DBD 4 ATAN T DB2.DBD 8	**الصيغة العامة**  **Example**  	**الصيغة العامة**  **Example** 

يقوم الأمر ATAN بإيجاد قيمة الزاوية لظل الزاوية بالرقم المخزن في ACCU1 وتخزين النتيجة في ACCU1 (مع اعتبار صيغة التخزين REAL) وحساب الزوايا يكون بالتقدير الدائري أي قيمة الزاوية × ط / 180

مثال تطبيقي -1

$$ax^2 + bx + c = 0$$

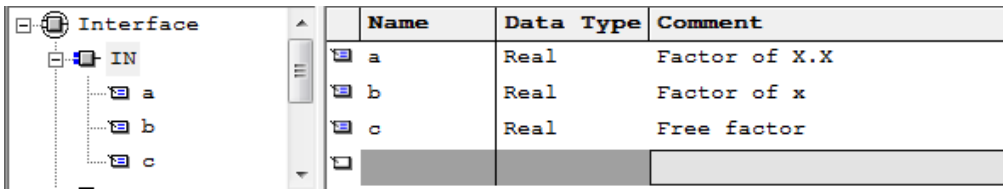
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

والآن لنعرض مثال للحسابات لتطبيق عملي لما استعرضناه من تعليمات وهو حل معادلة الدرجة الثانية في مجهول واحد والمطلوب عمل Function لحساب الحل في الحالة الأولى وهي قيمة المحدد صفر (جذرين حقيقيين متساويين) والحالة الثانية وهي قيمة المحدد أكبر من صفر وهي جذريين حقيقيين مختلفين) وإنهاء العمل مع وضع قيمة الجذرين صفر في الحالة الثالثة وهي قيمة المحدد أقل من صفر أي أن الجذرين تخيليين

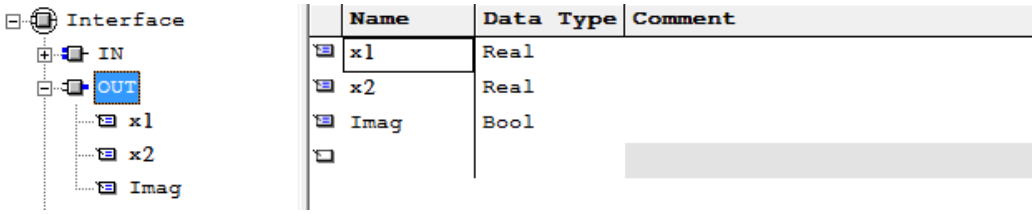
سنقوم أولا بتوصيف متغيرات الإدخال والإخراج إلى الدالة لتظهر على شكل قالب عند الاستدعاء مدخلاته الثوابت a,b,c ومخرجاته الجذران x1,x2 والمتغير الثنائي Imag والذي يكون "1" في حالة الجذرين تخيليين وسيكون ذلك على النحو التالي:

145

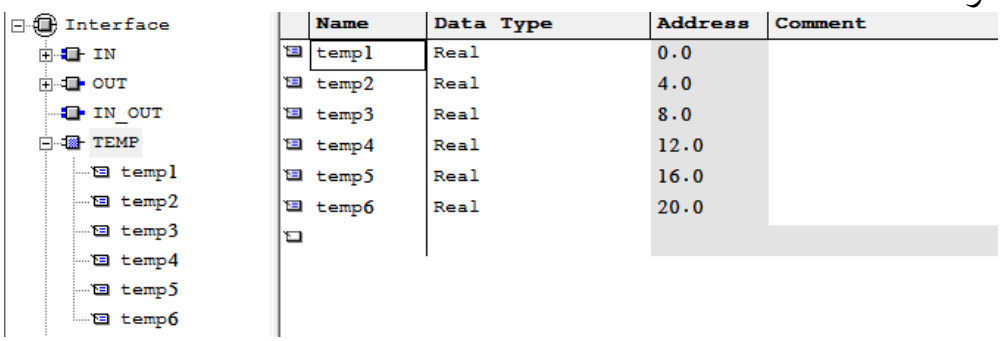
1- المدخلات



2- المخرجات



3- الذاكرة المؤقتة



```
      L     #b              #b        -- Factor of x
      SQR
      T     #temp1          #temp1
      L     #a              #a        -- Factor of X.X
      L     #c              #c        -- Free factor
      *R
      L     -4.000000e+000
      *R
      T     #temp2          #temp2
      L     #temp1          #temp1
      L     #temp2          #temp2
      +R
      T     #temp3          #temp3
      L     #temp3          #temp3
      L     0.000000e+000
      JZ    a001
      JP    a002
      JM    a003
      JU    a004
a001: L     #b              #b        -- Factor of x
      L     -2.000000e+000
      /R
      T     #x1             #x1
      T     #x2             #x2
      JU    a004
a002: L     #temp3          #temp3
      T     #temp4          #temp4
      SQRT
      L     #a              #a        -- Factor of X. X
      /R
      L     2.000000e+000
      /R

      T     #temp4          #temp4
      L     #b              #b        -- Factor of x
      L     -2.000000e+000
      /R
      T     #temp5          #temp5
      L     #temp5          #temp5
      L     #temp4          #temp4
      +R
      T     #x1             #x1
      L     #temp5          #temp5
      L     #temp4          #temp4
      -R
      T     #x2             #x2
      JU    a004
a003: L     0.000000e+000
      T     #x1             #x1
      T     #x2             #x2
      =     #Imag           #Imag
a004: NOP   0
```

- تحميل الثابت b
- تربيع b^2
- الحصول على -4ac
- الحصول على b^2-4ac (المحدد) وتخزين القيمة في #temp3
- مقارنة المحدد #temp3 مع 0.0
- إن كانت "0" انتقل إلى a001 وإن كانت موجة انتقل إلى a002 وإن كانت سالبة انتقل إلى a003 وإن كانت غير ذلك انتقل إلى a004
- في حالة المحدد "0" الجذرين متطابقين ويساوي b/2–
- في حالة المحدد موجب احسب الجذر التربيعي للمحدد واقسمه على 2a وخزن القيمة في #temp4
- الجذر الأول يساوي b/2– مجموعا على #temp4 والجذر الثاني يساوي b/2– مطروحا منه #temp4
- في حالة القيمة السالبة انقل القيمة 0.0 إلى كل من الجذرين وقم بتحويل قيمة #Imag إلى "1"

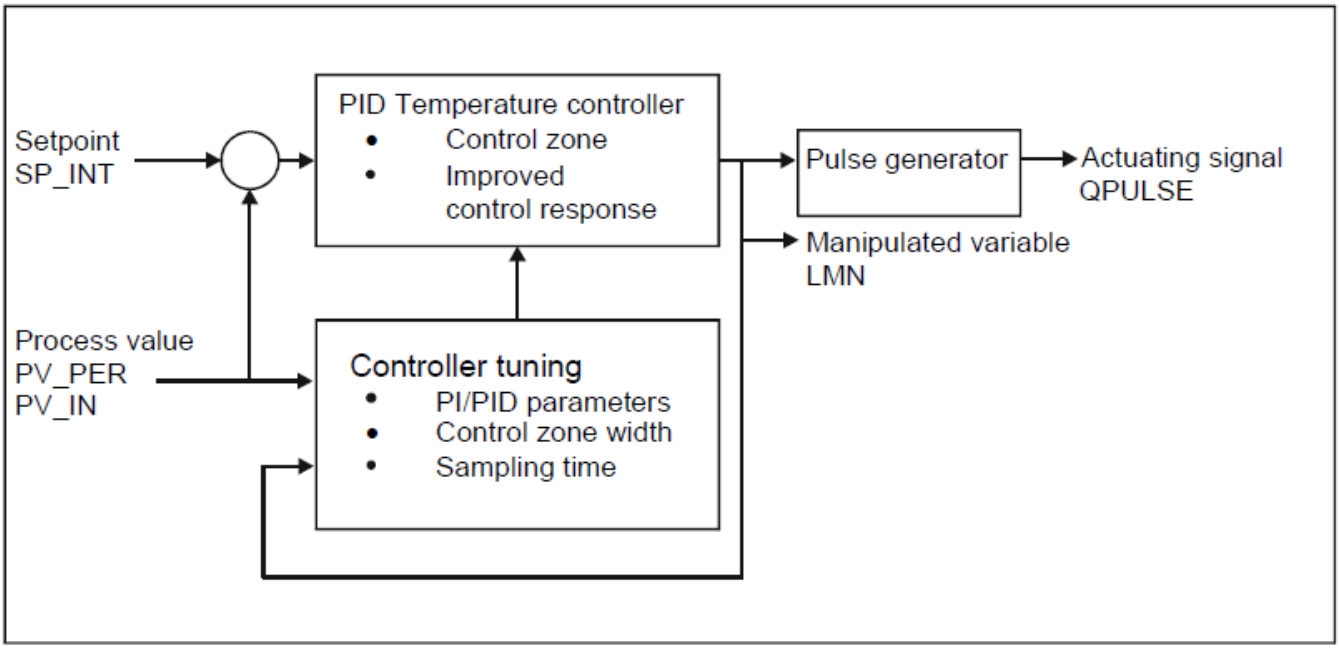
هذه العملية Function يتم استدعاؤها داخل أي بلوك آخر باسمها وتمرير القيم أو عناوين للقيم إليها لتقوم بالحسابات

دورة متقدمة في البرمجة Advanced Course in Step7

تتضمن الدورة المتقدمة التي سنتحدث عنها إن شاء الله في العناصر الآتية:

- منظومة التحكم PID
- العدادات فائقة السرعة High speed counters
- استخدام المدخلات والمخرجات الموزعة Distributed Input /Output units
- الاتصال بين وحدتي معالجة CPU 300 – CPU 300
- الاتصال بين CPU-300 ودرايف سيمنس Profibus
- الاتصال بين CPU-300 و S7-200
- تتبع الأعطال في منظومات S7-300/400
- عناصر المكتبة Library

منظومة التحكم في الحرارة PID-Temperature Controller (الجزء الأول):



تعتمد منظومة التحكم PID على إدخال قيمة للضبط SP_INT وقيمة فعلية إما PV_PER أو PV_IN ثم قياس الخطأ بينهما ثم إعطاء قيمة معالجة LMN إما في شكل إشارة تناظرية أو رقمية أو في شكل نبضات QPULSE

ويوجد هناك قسمان للمتحكم PID-Temperature Controller الأول هو منظومة التحكم نفسها والثاني هو منظومة الضبط أو التوليف Tuning لضبط بيانات تصرف الوحدة مع التغيرات

والعملية بشكل عام تتعامل مع مجموعة من المدخلات والمخرجات Inputs /Outputs منها منطقية BOOL ومنها رقمية سواء كانت صحيحة INT أو مضاعفة الدقة DINT أوعشرية REAL أو قيم وقت S5T

ويتم تمرير البيانات إما بشكل مباشر من وإلى البلوك عند استدعائه أو إلى بلوك البيانات المصاحب Instance Data Block

أما عن إدخال القيمة الفعلية للحرارة فيتم هذا عن طريق حساس للحرارة Temperature Sensor بالأشكال القياسية المتعارف عليها وقراءة ذلك عن طريق مدخل تناظري يتناسب مع الحساس المستخدم وسوف نستعرض فيما يلي كيفية إدخال قيمة الحرارة إلى البرنامج

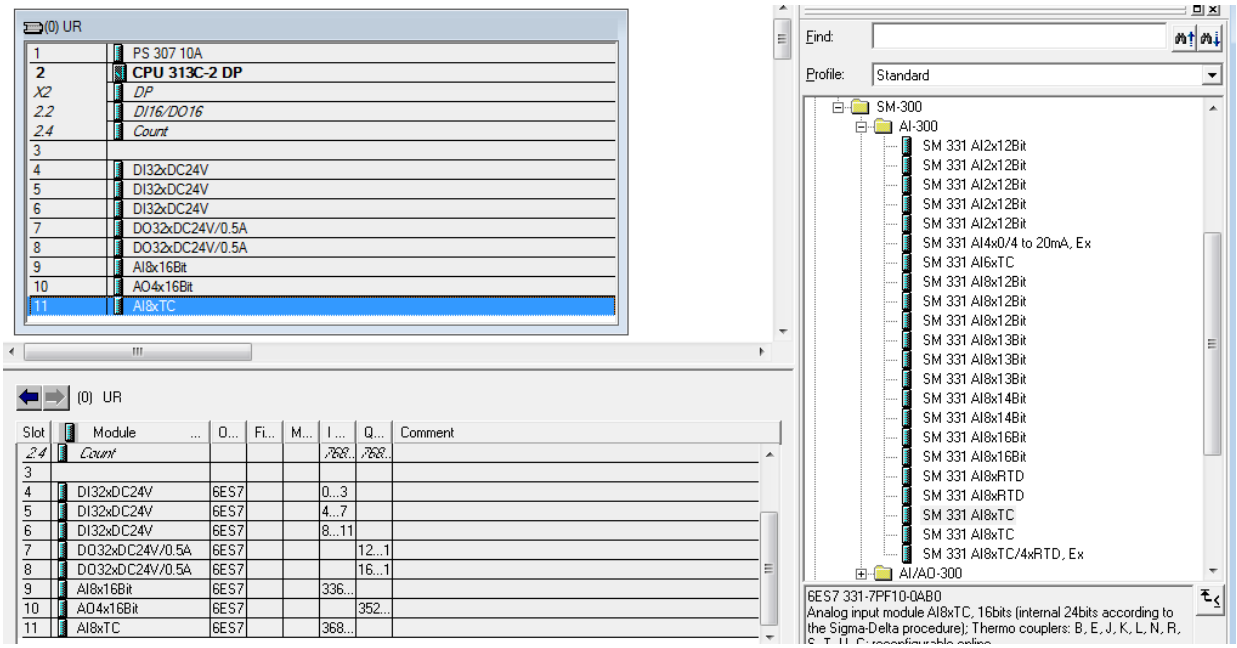
من أنواع حساسات الحرارة المستخدمة في قراءة درجة الحرارة:

- الثرموكبل Thermocouples بأنواعها المختلفة B, E, J, K, L, N, R, S, T, U, C
- المقاومات RTD بأنواعها Pt100, Pt200, Pt500, Pt1000, Ni100, Ni120, Ni1000, Cu10

وهذه الأنواع يتم توصيلها على وحدات مجهزة لذلك بعدد قنوات إدخال محددة تمثل عدد المدخلات التناظرية الممكن قراءتها مثل الوحدة 6ES7 331-7PF01-0AB0 ، Analog input module AI8xRTD, 16bits والتي يمكن أن يوصل عليها 8 حساسات من أنواع RTD المختلفة

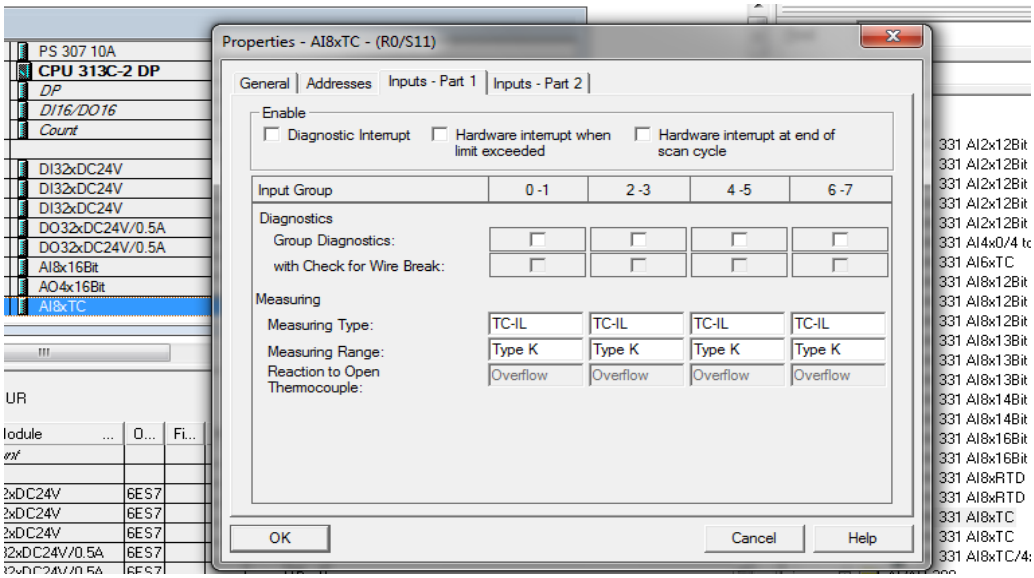
كذلك الوحدة 6ES7 331-7PF10-0AB0 ، Analog input module AI8xTC, 16bits والتي يمكن أن يتم توصيل 8 حساسات عليها من الأنواع ثيرموكبل

فيتم أولا تحديد أنواع الثرموكبل والوحدات التي سيتم استخدامها بناء على ذلك مثلا لو كنا سنستخدم حساس ثرموكبل النوع K عدد 8 وحدات وسوف نستخدم وحدة 6ES7 331-7PF10-0AB0 ، Analog input module AI8xTC, 16bits فسوف نقوم بفتح HW_Config ثم إدراج الوحدة ضمن مكونات النظام

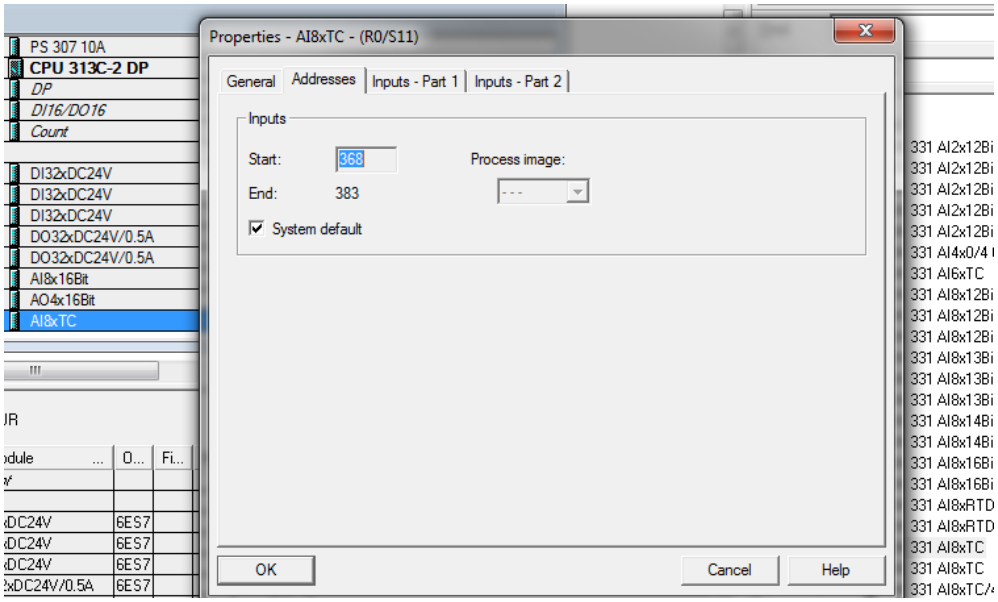


148

ثم نقوم بتحديد نوع الحساس من خصائص الوحدة



مع ملاحظة عنوان البداية والقنوات المختلفة والذي سوف نستخدمه داخل البرنامج



لاحظ أن العناوين المتضمنة هي

PIW368

PIW370

PIW372

PIW374

PIW376

PIW378

PIW380

PIW382

وهي 8 قنوات إدخال كما ذكرنا سابقا

وهنا إن أردنا إدخال قيمة الحرارة مباشرة من الحساس إلى منظومة PID فإن موضع الدخول سيكون PV_PER حيث يتم نقل القيمة PIW368 مثلا إلى DATA Word التي تمثل PV_PER

L PIW 368

T DBx.DBW4 x: Data Block no.

كما يمكن معالجة القيمة بشكل منفصل لتحويلها إلى رقم عشري يمثل درجة الحرارة أو نسبة مئوية أو كما يرغب المبرمج مع ملاحظة أنه في الحساسات العادية فإن درجة الحرارة يتم قراءتها برقم صحيح مضروب في 10 فمثلا 25 درجة مئوية تقرأ بالقيمة 250 والدرجة 135 تقرأ بالقيمة 1350 وذلك للحصول على كسر من درجة الحرارة فتصبح دقة القراءة حتى 0.1 درجة مئوية ولو أردنا مثلا الحصول على قيمة عشرية لتمريرها إلى PV_IN حيث صيغته REAL

149

L PIW 368

ITD

DTR

L 10.0

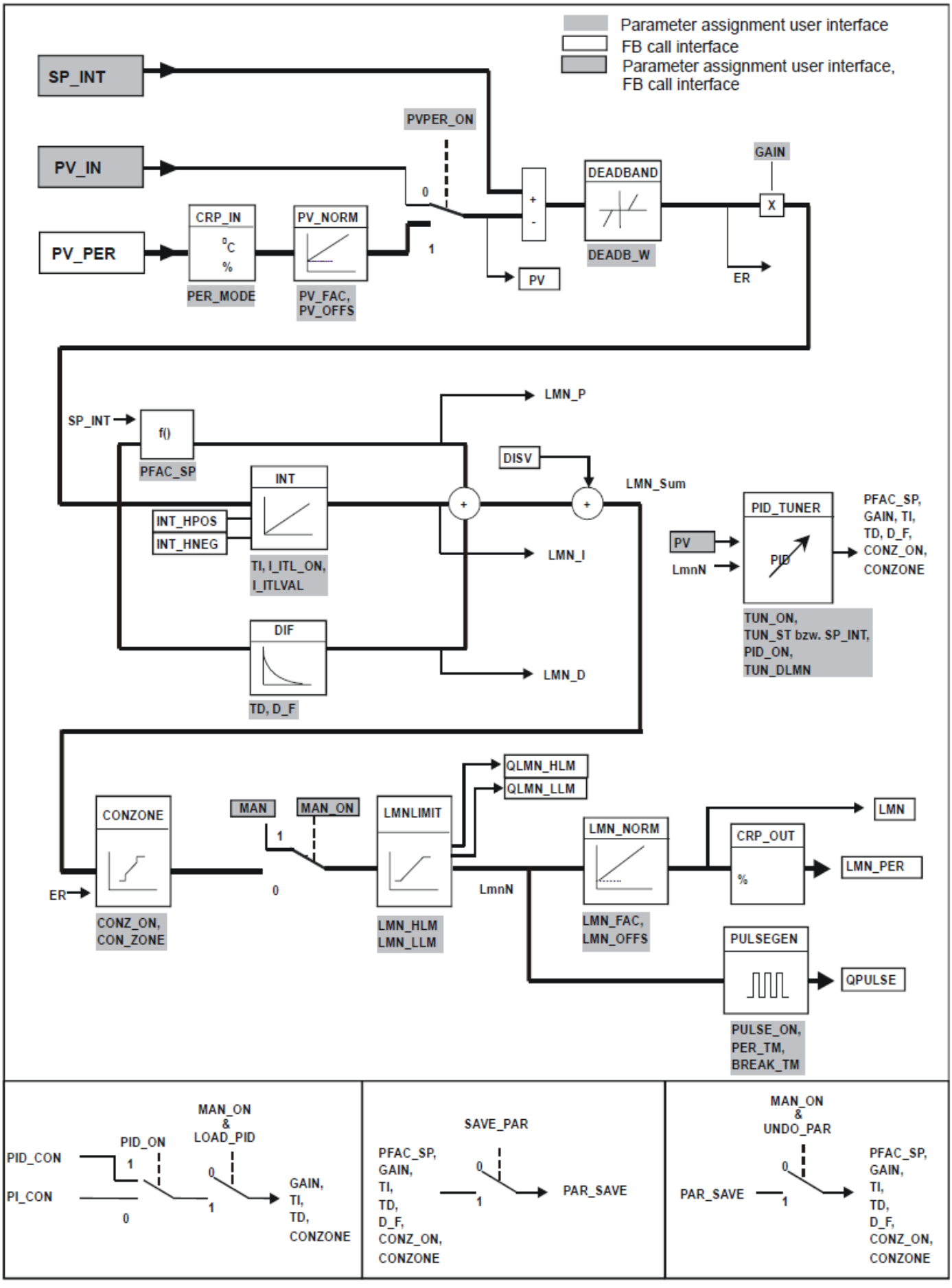
/R

T DBx.DBD0 x: Instance Data Block number

وهكذا نكون قد قمنا بإعداد طريقة إدخال القيمة الفعلية للمنظومة بشكليها المباشر من المدخل التناظري أو من مصدر بيانات معالج

أما عن القيمة التي نضبط عليها العمل أو المطلوب الوصول إليها Set value فإنه يتم تمريرها إلى Data Block إلى المتغير SP_INT حيث يتم تمريرها في شكل رقم عشري بصيغة REAL تمثل درجة الحرارة المطلوب الوصول إليها عن طريق عملية التسخين

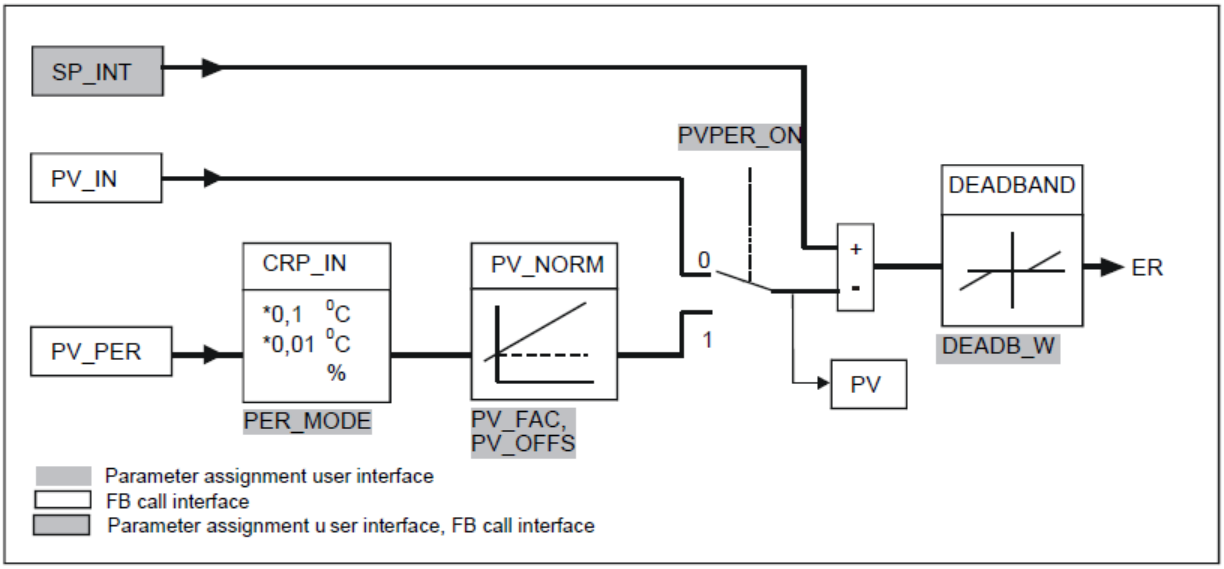
أما عن معالجة الإشارة وكيفية عمل منظومة PID فإن هذا يتضح لنا من الشكل التالي:



150

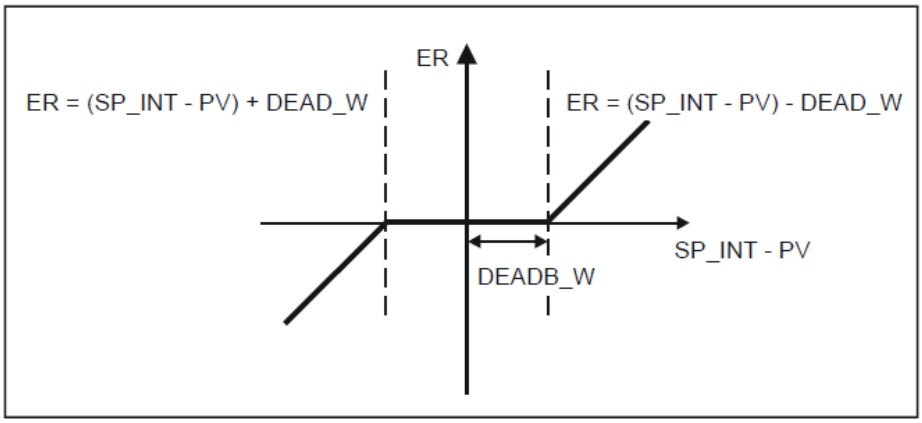
لو استعرضنا الشكل السابق فسوف نجد فيه البيانات الآتية:

- منطقة تحديد بيانات الضبط وقراءة القيمة الفعلية SP_INT و PV_IN أو PV_PER
- منطقة تهيئة القيمة الفعلية إن كانت PV_PER أي من مدخل تناظري مباشرة حيث يتم اختيار الدقة سواء كانت 0.1 أو 0.01 أو نسبة % عن طريق PER_MODE (0-1-2) ثم تحديد المقياس ونقطة الصفر Offset, Scale عن طريق PV_FAC و PV_OFFS ويتم الاختيار بين PV_IN وبين PV_PER عن طريق PVPER_ON وهو مفتاح تحويل وصيغته Binary يحدد أي من الطريقين ستمر من خلاله القيمة الفعلية
- بعد إدخال القيمتين SP(Set Point) و PV(Process Value) يتم تكوين الفرق بين الاثنين ERROR في المتغير ER والذي ستتم عليه المعالجة في كل المراحل التالية
- أو منطقة يدخل عليها الخطأ ER هي منطقة Dead Band عن طريق المتغير DEADB_W

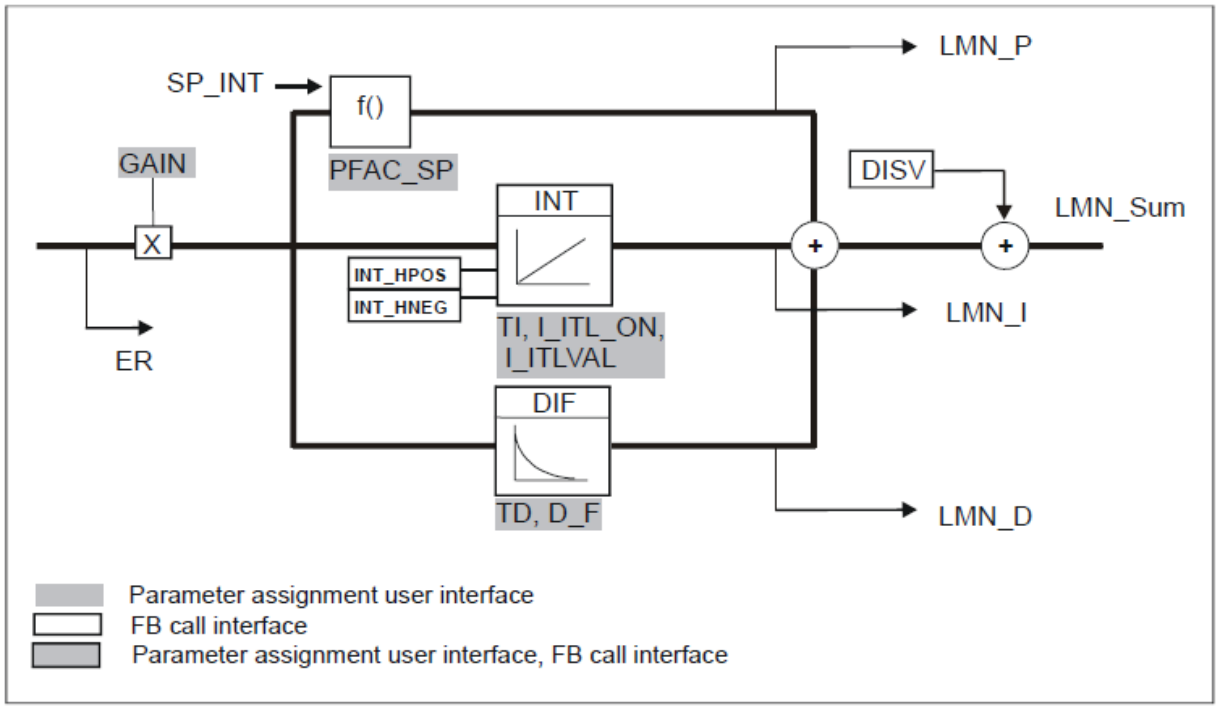


151

منطقة Dead Band هي منطقة لنطاق معين للخطأ لا يتم النظر إليه حتى لا تحدث عملية أرجحة حول قيمة معينة للمنظومة Oscillation فلو كان الخطأ أقل من حدود Dead Band أو المنطقة الميتة لا يعتبر تغيير في القيمة



والآن معنا الخطأ بين القيمة المطلوبة والقيمة الفعلية والمطلوب علاجه ليصل النظام إلى القيمة المطلوبة تماما بشكل ناعم Smooth وذلك من خلال مجموعة من العمليات المتتابعة والمتوازية وأول هذه العمليات هو تكبير الخطأ عن طريق GAIN إلى الحد المطلوب للدقة ويمكن أن يكون التكبير بالضرب في قيمة أكبر من أو أقل من 1.0 وكذلك بالإشارة فيمكن أن يتم تكبير الخطأ ER بإشارة موجبة أو إشارة سالبة لنحصل على خطأ مكبر بالضرب في قيمة GAIN وتتفرع بعد ذلك العملية إلى ثلاث طرق بالتوازي لتتعامل مع إشارة الخطأ المكبرة بعد المرور على GAIN

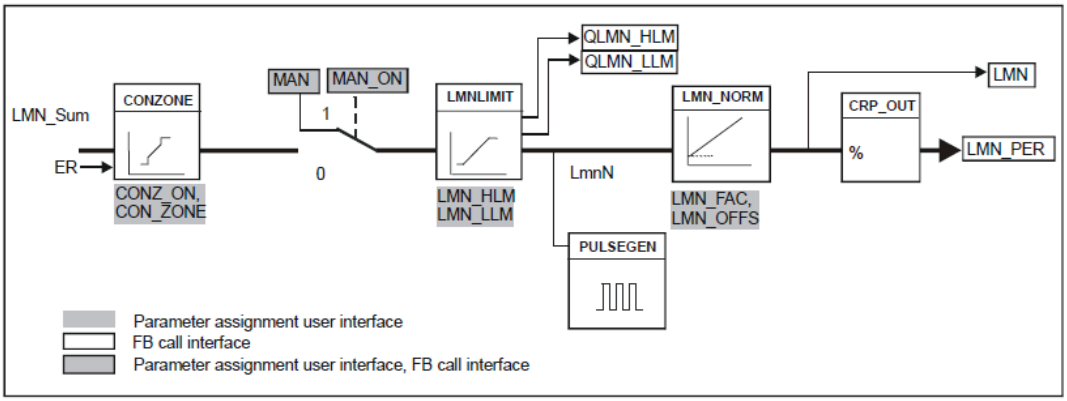


- الطريق الأول وهو Kp عن طريق PFAC_SP وهو معامل يمثل Kp حيث يتم ضرب القيمة في الخطأ
- الطريق الثاني وهو Ki (Integral) عن طريق المجموعة INT وهي:
  - معامل Ki ويمثله TI (قيمة وقت)
  - ويمثل المعامل I_ITL_ON حالة تفعيل العملية TI
  - ويمثل العامل I_ITLVAL القيمة المبدئية للمعامل TI
  - وهناك مخرجان BOOL وهما INT_HPOS و INT_HNEG وتتحول حالة كل منهما إلى "1" في حالة الوصول إلى الحد الأقصى الموجب أو السالب والذي يتم إدخاله كقيمة قصوى
- الطريق الثالث وهو Kd (Differential) عن طريق المجموعة DIF وهي:
  - معامل Td ويمثله TD (قيمة وقت)
  - معامل المقياس وهو D_F أي Derivative factor

وبالتالي نحصل على ثلاث قيم للمعالجة بالتوازي وهي LMN_P و LMN_I و LMN_D حيث يتم جمعها معا وإضافة قيمة هي DISV وتمثل قيمة مضافة إلى المجموع إن كان هذا مطلوبا وإلا نحصل على مجموع المعالجات الثلاثة لتعطي قيمة LMN_Sum والتي تمثل الخطأ بعد معالجته عن طريق المتحكم PID ويتبقى تمرير هذه القيمة إلى وسائل التحكم الخارجية والتي تتيح لنا التحكم الفعلي في الحرارة

أول منطقة تمر عليها الإشارة المعالجة هي منطقة Control zone ويتم فيها تحديد تفعيل عمل Control zone عن طريق CONZ_ON وهو متغير من النوع BOOL فإن كانت حالته "1" يتم تفعيل عمل Control zone حيث يتم احتساب الخطأ ER فإن تجاوزت قيمته قيمة المتغير CON_ZONE فإنه في هذه الحالة يتم تمرير القيمة القصوى الموجبة أو السالبة للإشارة المعالجة LMN_LLM أو LMN_HLM أما إن كان الخطأ ER داخل حدود Control Zone فيتم توصيل LMN_Sum من خلاله دون تغيير للمرحلة التالية

المرحلة التالية وهي إخراج القيمة المعالجة عن طريق المنظومة أو إخراج قيمة أخرى يدويا حيث يمكن عن طريق معامل مفتاح اختيار Selector switch الاختيار بين القيمة المعالجة والقيمة اليدوية والتي يمكن إدخالها في المتغير MAN عن طريق المفتاح MAN_ON والذي يجب أن يكون في الحالة "0" حتى تعمل المنظومة آليا



المرحلة التالية وهي مرحلة الحد الأقصى والأدنى Limiting حيث يتم تحديد حد أقصى للقيمة المعالجة وهو LMN_HLM وحد أدنى هو LMN_LLM وفي حالة تجاوز أي منهما فإن هناك مخرجان لقياس ذلك وهما من النوع BOOL تتحول حالتهما إلى "1" عن تجاوز القيمة وهما QLMN_HLM و QLMN_LLM وبالتالي نحصل بعدها على الإشارة بعد التحديد

وندخل بعد ذلك إلى مرحلة الإخراج النهائي للمنظومة والذي يتم توجيهه إما لإخراج نبضات Pulses عن طريق مولد نبضات PULSEGEN يتعامل بنظام PWM حيث يتغير عرض النبضة طبقا لقيمة LMN بنسبة من قيمة قصوى مثلا إن كان الزمن الكامل للنبضة هو 1 ثانية فيكون التغيير من 0.0 وحتى 1.0 ثانية حسب قيمة LMN وهذا له جزء خاص سنتحدث عنه تفصيليا فيما بعد إن شاء الله

أما الاختيار الثاني فهو إخراج القيمة في شكل رقم فبعد معالجته بمقياس LMN_FAC و LMN_OFFS يتفرع أيضا إلى فرعين الأول في شكل رقم عشري كما هو من المنظومة في المتغير LMN والثاني في شكل رقم صحيح يمكن تمريره مباشرة إلى مخرج تناظري عن طريق LMN_PER

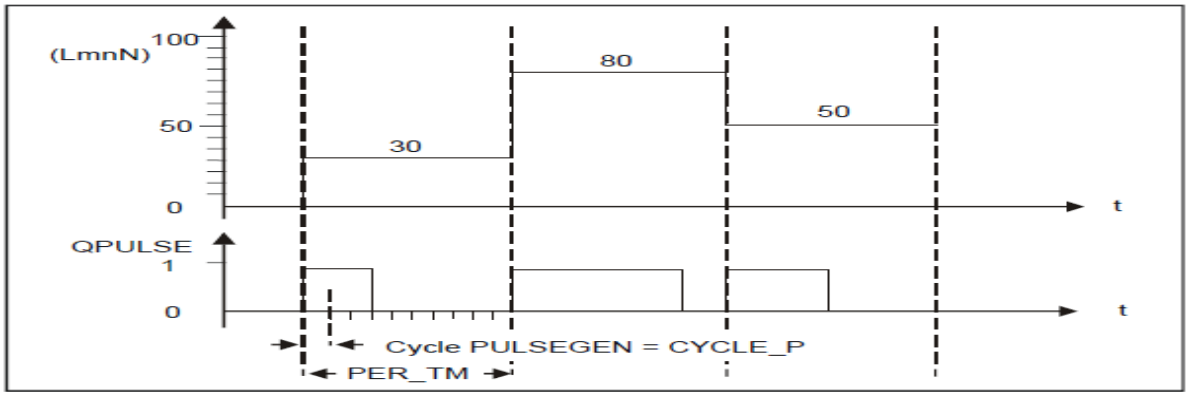
والآن بعد أن استعرضنا معا كيف تعمل منظومة PID نبدأ في التعرف على توظيف PID داخل برنامج STEP-7

أولا : لابد من تحديد نوع المنظومة هل ستكون PID أم PI فقط حيث يفضل الكثيرون استخدام منظومة PI فقط ويتم هذا عن طريق المتغير PID_ON فإن كانت قيمته "1" تكون المنظومة PID وإن كانت قيمته "0" تكون المنظومة PI فقط

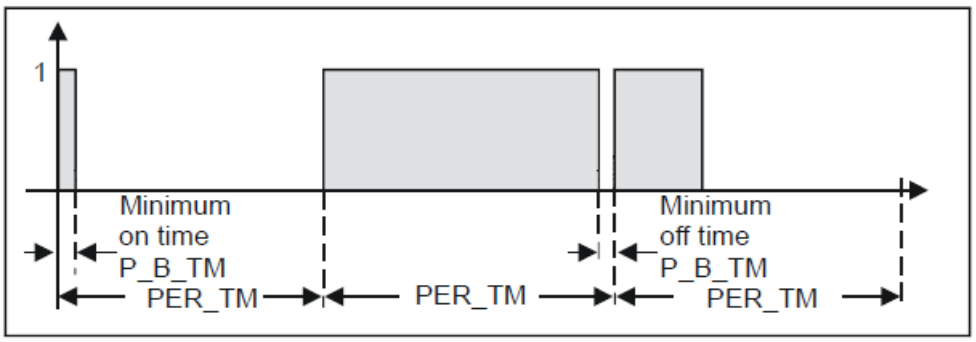
ثانيا : لابد من تحديد الخيارات المنطقية كلها أولا مثل PV_PER و MAN_ON و CONZ_ON والتي ستشكل شكل استخدام وعمل المنظومة

ثالثا : لابد من تحديد قيم DEADB_W و CON_ZONE و LMN_HLM و LMN_LLM

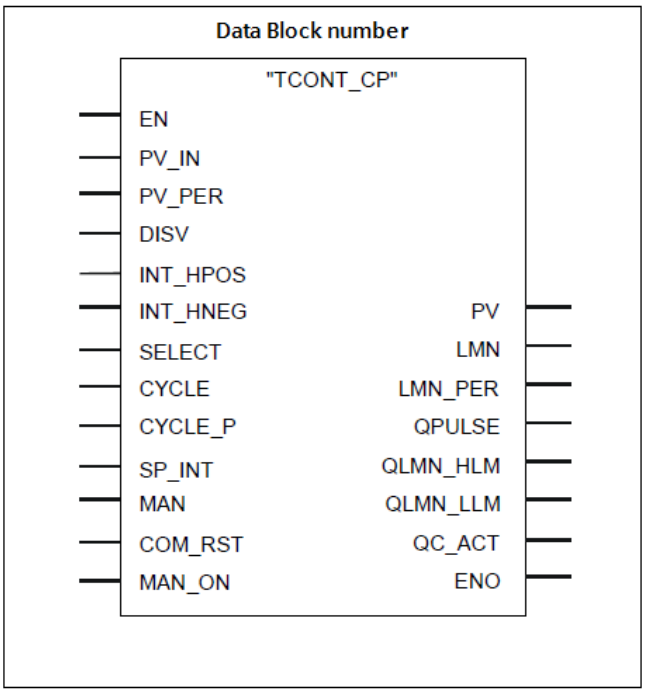
رابعا : لابد من تحديد هل سيتم استخدام Pulse generator أم لا حيث يتم توظيف الاستخدام لذلك PULSE_ON



ويتم كذلك تحديد زمن دورة التسخين PER_TM وأقل وقت للتسخين BREAK_TM



ويتم استدعاء البلوك FB58 والذي يحمل الاسم الرمزي "TCONT_CP" داخل أحد بلوكات التنظيم OB الدورية مثل OB35 والذي يتم استدعاؤه كل فترة زمنية محددة Cyclic Interrupt وبالتالي نضمن عملا مرتبطا ارتباطا فعليا بالوقت الحقيقي Real Time حيث تكون دورة تنفيذ OB35 من مضاعفات دورة تنفيذ عملية PID حيث يوجد عامل اسمه Cycle يحدد دورية عمل PID والذي يجب أن يرتبط مع دورية عمل OB35 كما ذكرنا

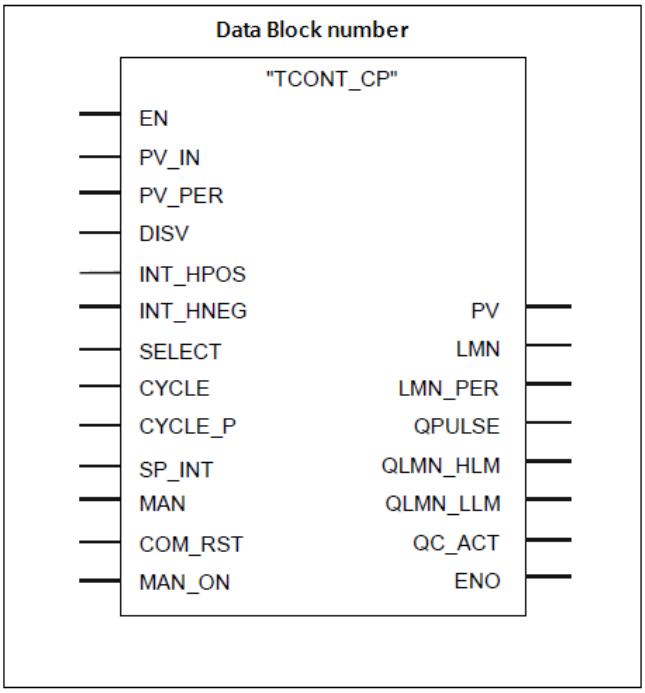


يتم استدعاء FB58 مع تحديد DB المصاحب والذي سيتم تمرير جميع البيانات الخاصة بالتشغيل إليه من خلال البرنامج

ولا يلزم هنا تمرير بيانات إلى FB58 حيث يمكن استدعاؤها كما بالشكل المقابل تماما ودون أي شئ باستثناء ذكر رقم بلوك البيانات والذي سيتم إنشاؤه وتوصيفه عن طريق البرنامج ولا دخل لنا بذلك فقط سنقوم بتمرير القيم إلى البلوك باستخدام الأمر Transfer مثلا وقراءة القيم التي نريدها إن كانت رقمية عن طريق الأمر Load مثلا

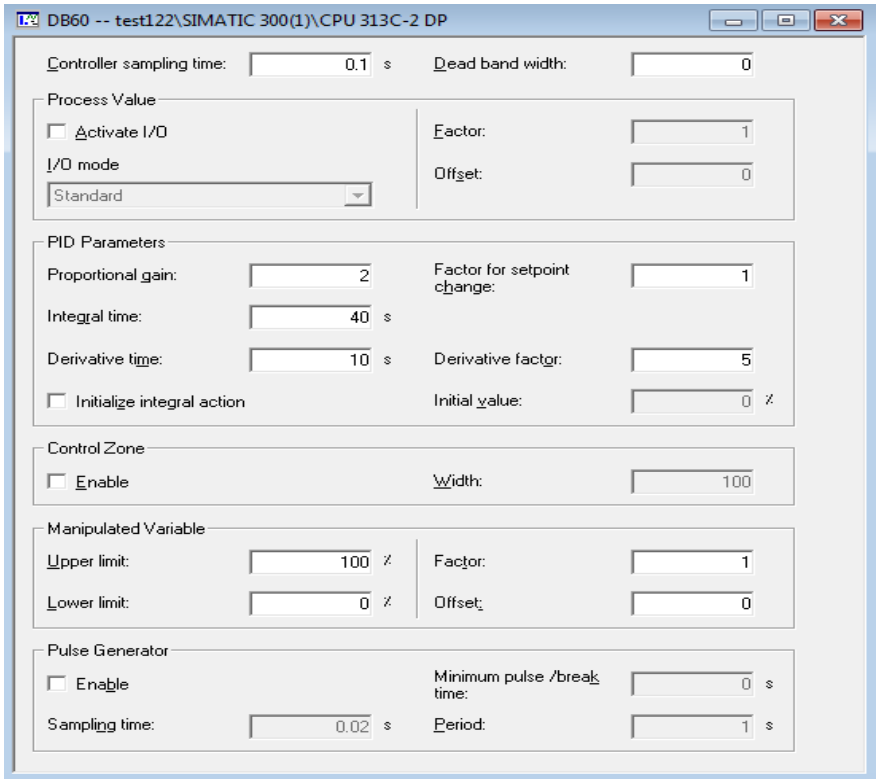
أما المخرجات BOOL فيتم استخدامها مباشرة في العمليات المنطقية مثلها مثل أي متغير منطقي

تابع استخدام منظومة PID

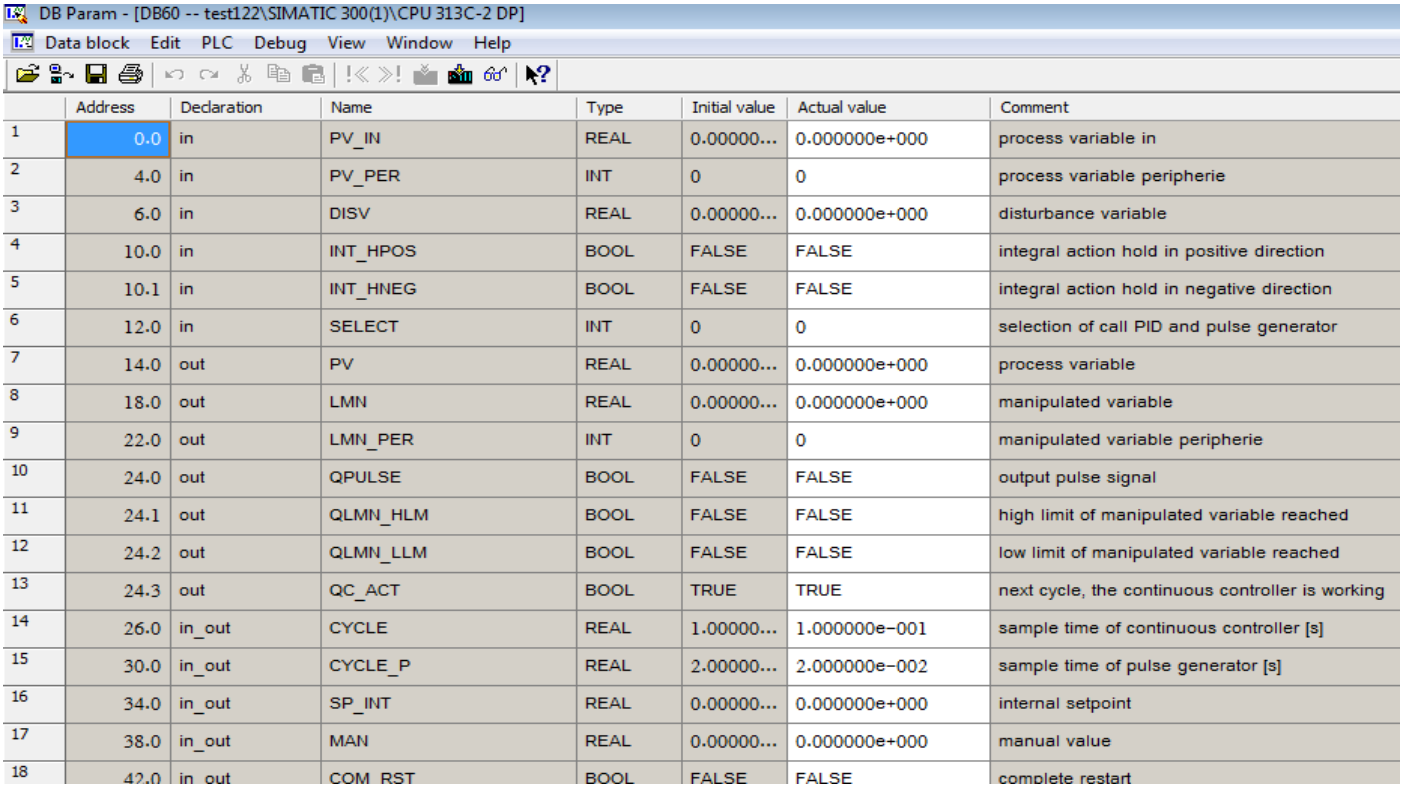


يتم استدعاء FB58 مع تحديد DB المصاحب والذي سيتم تمرير جميع البيانات الخاصة بالتشغيل إليه من خلال البرنامج
ولا يلزم هنا تمرير بيانات إلى FB58 حيث يمكن استدعاؤها كما بالشكل المقابل تماما ودون أي شئ باستثناء ذكر رقم بلوك البيانات والذي سيتم إنشاؤه وتوصيفه عن طريق البرنامج ولا دخل لنا بذلك فقط سنقوم بتمرير القيم إلى البلوك باستخدام الأمر Transfer مثلا وقراءة القيم التي نريدها إن كانت رقمية عن طريق الأمر Load مثلا
أما المخرجات BOOL فيتم استخدامها مباشرة في العمليات المنطقية مثلها مثل أي متغير منطقي

ويتم إنشاء DB بشكل آلي مع FB58 التي قمنا باستدعائها ويأخذ شكلين للعرض



الشكل الأول في شكل بيانات تخص PID مثل الشكل المقابل حيث تتعرض منطقة Process value لبيانات ضبط المدخل الخاص بالقيمة PV ويختص الجزء PID Parameters ببيانات ضبط وتوليف منظومة PID مثل Gain,Ti,Td أما الجزء الأعلى فيختص بدورية تنفيذ المنظومة ويختص المعامل Dead band width بتحديد المنطقة المهمل فيها تغييرات الخطأ كذلك المنطقة Control Zone حيث يتم التفعيل وإدخال القيمة أما المنطقة Manipulated Variable فتختص بمعالجة الخرج من المنظومة وأخيرا الجزء الخاص بتوليد النبضات والتي تشمل التفعيل وأقل وقت للنبضة ووقت الدورة


DB Param - [DB60 -- test122\SIMATIC 300(1)\CPU 313C-2 DP]

Data block Edit PLC Debug View Window Help

	Address	Declaration	Name	Type	Initial value	Actual value	Comment
1	0.0	in	PV_IN	REAL	0.00000...	0.000000e+000	process variable in
2	4.0	in	PV_PER	INT	0	0	process variable peripherie
3	6.0	in	DISV	REAL	0.00000...	0.000000e+000	disturbance variable
4	10.0	in	INT_HPOS	BOOL	FALSE	FALSE	integral action hold in positive direction
5	10.1	in	INT_HNEG	BOOL	FALSE	FALSE	integral action hold in negative direction
6	12.0	in	SELECT	INT	0	0	selection of call PID and pulse generator
7	14.0	out	PV	REAL	0.00000...	0.000000e+000	process variable
8	18.0	out	LMN	REAL	0.00000...	0.000000e+000	manipulated variable
9	22.0	out	LMN_PER	INT	0	0	manipulated variable peripherie
10	24.0	out	QPULSE	BOOL	FALSE	FALSE	output pulse signal
11	24.1	out	QLMN_HLM	BOOL	FALSE	FALSE	high limit of manipulated variable reached
12	24.2	out	QLMN_LLM	BOOL	FALSE	FALSE	low limit of manipulated variable reached
13	24.3	out	QC_ACT	BOOL	TRUE	TRUE	next cycle, the continuous controller is working
14	26.0	in_out	CYCLE	REAL	1.00000...	1.000000e-001	sample time of continuous controller [s]
15	30.0	in_out	CYCLE_P	REAL	2.00000...	2.000000e-002	sample time of pulse generator [s]
16	34.0	in_out	SP_INT	REAL	0.00000...	0.000000e+000	internal setpoint
17	38.0	in_out	MAN	REAL	0.00000...	0.000000e+000	manual value
18	42.0	in_out	COM_RST	BOOL	FALSE	FALSE	complete restart

طريقة العرض في شكل Data View أو Declaration View وتكون في شكل جدول يشمل العنوان والوصف الوظيفي in/out والاسم الرمزي ونوع البيانات والقيمة المبدئية والقيمة الحالية التي يمكن مراقبتها وملحوظات عن البيانات وهنا لا يكون التقسيم بشكل وظيفي مثل الطريقة السابقة في العرض

156 وعند برمجة المنظومة سوف نتبه الخطوات الآتية:

- تحديد طريقة إدخال القيمة الفعلية Process Value (PV) سواء عن طريق المدخل الطرفي أو من قيمة حقيقية
- تحديد مصدر القيمة المطلوبة Set Point (SP)
- تحديد شروط عمل Complete Reset
- تحديد شروط التشغيل العامة للمنظومة وتشمل :
  - PVPER_ON حسب اختيار طريقة إدخال القيمة الفعلية – DB60.DBX90.0
  - MAN_ON تكون على الوضع "0" - DB60.DBX42.1
  - PID_ON لاختيار نوع المنظومة =1 في حالة PID و =0 في حالة PI – DB60.DBX186.6
  - CONZ_ON لتحديد استخدام Control Zone – DB60.DBX186.0
  - PULSE_ON لتحديد استخدام Pulse generator مولد الذبذبات DB60.DBX90.2
  - I_ITL_ON لتحديد شروط تفعيل Initialization of Ti – DB60.DBX90.1
  - SELECT لتحديد مواصفات تشغيل PID – DB60.DBW12 سواء داخل OB1 أو OB35
  - CYCLE لتحديد زمن دورة التنفيذ – DB60.DBD26 لتتوافق مع زمن بلوك OB35
  - PV_IN لتحديد مصدر القيمة الفعلية في شكل رقم عشري Real من خلال DB60.DBD0 أو PV_PER لتحديد مصدر القيمة الفعلية في شكل رقم صحيح Int من خلال DB60.DBDW4
  - PV_FAC,PV_OFFS لتحديد نطاق تمثيل القيمة الفعلية – DB60.DBD60,DBD64
  - SP_INT لتحديد مصدر Set point لقيمة الحرارة المطلوبة DB60.DBD32
  - DEADB_W لتحديد منطقة إهمال الخطأ إن تم استخدامها أو تركها بالقيمة "0" – DB60.DBD44

- CON_ZONE لتحديد منطقة Control zone حيث خارج منطقة خطأ معين يتم التحكم بقيمة قصوى أو قيمة دنيا – DB60.DBD182

- تحديد المتغيرات الخاصة بضبط توليف المنظومة Manual Tuning
  - GAIN والذي من خلاله يتم تكبير أو تقليل الإحساس بالخطأ – DB60.DBD166
  - TI وقت Reset والذي من خلاله يتم تحديد وقت تصفير الخطأ – DB60.DBD170
  - TD وقت التفاضل في حالة استخدامه DB60.DBD174
  - PFAC_SP وهو معامل يضرب في الخطأ – DB60.DBD162
  - D_F وهو معامل للتفاضل يضرب بعد المعالجة في ناتج التفاضل – DB60.DBD178

- تحديد المتغيرات الخاصة بخرج المنظومة
  - PV وتمثل القيمة الفعلية بعد اختيارها داخل المنظومة – DB60.DBD14
  - ER وتمثل الخطأ بين القيمة المطلوبة والقيمة الفعلية عند الرغبة في استخدامها – DB60.DBD92
  - LMN وتمثل القيمة المعالجة في شكل Real ويمثلها – DB60.DBD18
  - LMN_PER وتمثل القيمة المعالجة في شكل Int رقم صحيح – DB60.DBW22
  - LMN_P وتمثل مركبة التكبير فقط في حالة الرغبة في استخدامها منفردة – DB60.DBD96
  - LMN_I وتمثل مركبة التكامل فقط في حالة الرغبة في استخدامها منفردة – DB60.DBD100
  - LMN_D وتمثل مركبة التفاضل فقط في حالة الرغبة في استخدامها منفردة – DB60.DBD104
  - QPULSE ويمثل خرج النبضة في حالة استخدام مولد النبضات – DB60.DBX24.0
  - QLMN_HLM وتمثل وصول القيمة المعالجة لأقصى قيمة – DB60.DBX24.1
  - QLMN_LLM وتمثل وصول القيمة المعالجة لأقل قيمة – DB60.DBX24.2

157

وبالتالي فهذه البيانات التي قمنا بتحديدها سابقا هي أهم ما يلزمنا فعليا لضبط المنظومة واستخدامها ومنها ما يتم ضبطه مرة واحدة ولا يتم استخدامه بعد ذلك خلال البرنامج مثل Switches الافتراضية في المنظومة وهذه أفضل أن يتم تفعيلها في OB100 حيث يتم عمل SET لهذه العناصر أو عمل RESET لها مرة واحدة ولا يتكرر ذلك في باقي البرنامج

مثلا:

```
R M0.0                  // Reset flag M0.0
AN M0.0                 // Check for M0.0=0
S DB60.DBX90.0          // Choose PVPER_ON
S DB60.DBX90.2          // Choose QPULSE_ON
R DB60.DBX42.1          // Choose Auto mode
R DB60.DBX186.6         // Choose PI mode
FP M0.1
= DB60.DBX42.0          // Make complete restart for PID
AN M0.0
L 2
T DB60.DBW12            // Choose to SELECT working with OB35
L 0.1
T DB60.DBD26            // Transfer value for CYCLE
```

أما العناصر الخاصة بالضبط أو توفيق القيم فنقوم بتنفيذها داخل أحد البلوكات FC العادية حيث نقوم بتمرير القيم إما عن طريق شاشة تشغيل مثلا أو عن طريق البرنامج :

```
L PIW300          // Load reading from temperature sensor
T DB60.DBW4       // Transfer temperature value to PV_PER
```

أما البيانات الأخرى التي تشمل عمليات الضبط فيمكن تمريرها من خلال شاشة تشغيل تشمل البيانات التالية :

```
DB60.DBD162       // PFAC_SP Factor multiplied by Error
DB60.DBD166       // GAIN
DB60.DBD170       // TI
DB60.DBD174       // TD
DB60.DBD178       // D_F
```

أما عن بيانات الإخراج فتشمل تمرير البيانات إلى نقاط الإخراج سواء كانت BOOL أو REAL :

```
DB60.DBD14        // PV Process value
DB60.DBD18        // LMN value
DB60.DBD92        // ER Error
DB60.DBD96        // LMN_P
DB60.DBD100       // LMN_I
DB60.DBD104       // LMN_D
```

158

```
A DB60.DBX24.0    // Pulse output
= Q 12.0          // Output to Heating Contactor
```

توليف منظومة PID

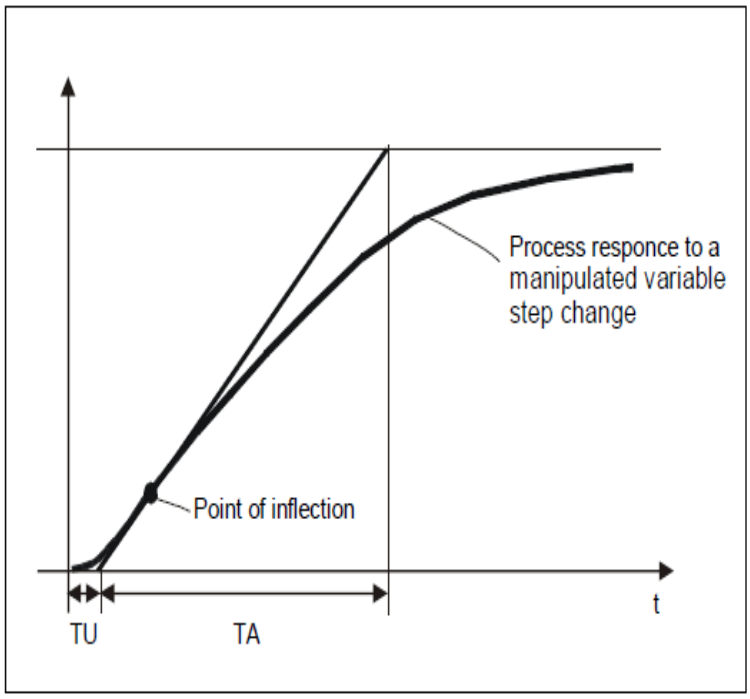
هناك طريقتان لتوليف منظومة PID

- التوليف بالاقتراب إلى نقطة التشغيل وذلك بتغيير قيمة Set-point مع تفعيل عملية التوليف
- التوليف على نقطة التشغيل فبعد الوصول لنقطة التشغيل يتم تفعيل التوليف وبدء عملية التوليف

ويمكن أيضا تنفيذ عملية التوليف من خلال نافذة بلوك البيانات الخاصة بالعرض الوظيفي Parameter assignment user interface

كما يمكن التوليف يدويا اعتمادا على خبرة المبرمج وفهمه لعمل المنظومة

وللوصول إلى أفضل أداء للمتحكم PID فإنه يجب معرفة أن المنظومة مصممة للعمل في بيئة خالية من التداخلات والشوشرة وبالتالي فإن استجابة المنظومة للتغيرات سوف تكون حادة نسبيا وقد ينتج عنها زيادة فجائية بين 10% ، 40% ولتفادي حدوث مثل هذا الأمر يمكن خفض التأثير المباشر للخطأ ER وذلك عن طريق المعامل PFAC_SP
كذلك يمكن الحد من هذه الزيادات الفجائية Overshoot عن طريق الحدود القصوى والدنيا للقيمة المعالجة والتي يمثلها العاملان LMN_HLM, LMN_LLM



ولفهم عملية التوليف Tuning وما يحدث فيها نلقي نظرة على آلية استجابة منظومة الحرارة لخرج PID فهناك خط منحنى يمثل عملية الاستجابة كما بالشكل المقابل كما أنه عند عمل خط يقطع معدل التغيير السريع مع البطئ والتي تمثل نقطة انعكاس في الاستجابة حيث غالبا ما تكون الاستجابة سريعة في البداية ثم تبدأ في البطء وهذه النقطة تسمى نقطة الانعكاس Point of inflection حيث تمثل منطقة الاستجابة من أول التشغيل وحتى الوصول لنقطة تقاطع الخط الذي يمثل الاستجابة الأولية مع نقطة التشغيل فهذا الوقت يمثل كما بالشكل TU+TA حيث TU قبل الانعكاس و TA بعد الانعكاس وبالتالي فهناك ثلاث احتمالات:

- أن يكون TA/TU أقل من 0.1
- أن يكون TA/TU تقريبا 0.1
- أن يكون TA/TU أكبر من 0.1

فالحالة الأولى هي التي تمثل ما تم تصميم المنظومة - التي تمثلها FB58 - من أجله
وعند عمل التوليف فكما قلنا من قبل فإنه في كل الحالات يجب أن يتم تشغيل الخانة TUN_ON ففي كل الحالات يتم الوصول إلى قيمة للحرارة ثم تفعيل TUN_ON والتي تمثلها الخانة DB60.DBD186.1 في البلوك DB60 والذي اخترناه كمثال

- ثم نقوم بإدخال قيمة جديدة للحرارة المطلوبة SP_INT وفي مرحلة الوصول بالحرارة من القيمة الحالية إلى القيمة الجديدة تتم عملية التوليف وعند نقطة الانعكاس Point of inflection يتم تسجيل القيم الجديدة لمتغيرات المنظومة
- الطريقة الثانية أن نقوم مع وصول الحرارة إلى النقطة المطلوبة بتفعيل كل من TUN_ON و TUN_ST وتمثلها الخانتان DB60.DBD186.1 و DB60.DBD186.2

159

وكنتيجة لعملية التوليف الآلية فإن المتغيرات الآتية يحدث عليها تغيير:

- معامل PFAC_SP
- GAIN
- TI
- TD
- D_F
- CONZ_ON
- CON_ZONE

- في حالة إيقاف عملية التوليف الآلي يتم العودة للقيم القديمة مرة أخرى
- وفي حالة الرغبة في تسجيل البيانات التي الحالية لقيم المتحكم PID التي عثرنا عليها نقوم بتفعيل SAVE_PAR وهو في العنوان في المثال لدينا DB60.DBX186.4
- وفي حالة الرغبة في استعادة الضبط الأخير قبل الوضع الحالي يتم ذلك عن طريق تفعيل UNDO_PAR وهو في العنوان DB60.DBX186.3 في المثال لدينا

هذه القيم السابقة التي تم ضبطها هي أهم ما يلزم معرفته لاستخدام منظومة التحكم في الحرارة باستخدام FB58

وللمزيد يمكن الرجوع للإصدار التفصيلي من شركة سيمنس **PID Temperature Control** والخاص بالتحكم في منظومات الحرارة عن طريق PID