

GX IEC Developer

(基本 教育)





目 录

1. 概 要	6 page
1.1 IEC61131-3的 概要.	
1.2 IEC61131-3的 特征.	
1.3 IEC61131-3的 分类.	
1.4 IEC61131-3的 语言.	
2. GX IEC Developer的 介绍	7 page
2.1 什么叫 GX IEC Developer	
2.2 GX IEC Developer & GX Developer 比较.	
2.3 GX IEC Developer的 系统事项.	
3. GX IEC Developer 程序的构成要素	9 page
3.1 关于 结构化程序	
3.2 什么叫 POU (Program Organisation Units)	
3.3 什么叫 TASK.	
3.4 在GX IEC Developer里的画面构成	
4. POU的Class	12 page
4.1 Program (PRG)	
4.2 Function (FUN)	
4.3 Function Block (FB)	
5. POU的 构成	17 page
5.1 POU的 Header	
5.2 POU的 Body	
6. 变数 (Variable)	19 page
6.1 全域(Global)变数说明	
6.2 局部(Local)变数说明	
6.3 变数选定要素	
6.4 Data Type的 阶层	

**7. GX IEC Developer 使用 - (初期 设定 1) 25 page**

- 7.1 New Project 制作
- 7.2 Project 画面 构成

8. GX IEC Developer 使用 - (初期 设定 2) 31 page

- 8.1 PLC 传送设定
- 8.2 POU 作成方法
- 8.3 TASK 作成及登录方法
- 8.4 变数(Variable)设定
- 8.5 Check & Compiling

9. GX IEC Developer 使用 - (POU 程序-LD 制作) 40 page

- 9.1 利用三菱地址的 LD制作
- 9.2 根据全域变数的 LD制作
- 9.3 根据局部变数的 LD制作

10. GX IEC Developer 使用 - (POU 程序-FUN 制作) 47 page

- 10.1 FUN 作成
- 10.2 作成的FUN在PRG里的应用
- 10.3 在FUN内部软元件的使用

11. GX IEC Developer 使用 - (POU 程序 - FB 制作) 54 page

- 11.1 FB 制作
- 11.2 作成的FB在PRG里的应用
- 11.3 在FB的内部软元件使用
- 11.4 利用 MACRO的 FB制作
- 11.5 适用MACRO的FB内部软元件使用

12. GX IEC Developer 使用 - (程序传送) 64 page

- 12.1 Format
- 12.2 Download Project (PLC 写入)
- 12.3 Upload Project (PLC 读出)
- 12.4 Verify

**13. GX IEC Developer 使用 - 指令及其他使用方法 71 page**

- 13.1 A触点 & b触点的设定
- 13.2 输出线圈处理
- 13.3 脉冲处理
- 13.4 Timer
- 13.5 Counter
- 13.6 Data Register
- 13.7 Word 软元件的BIT指定
- 13.8 BIT 软元件模块化
- 13.9 Data Code 变换
- 13.10 Data 传送 (MOV)
- 13.11 比较运算指令
- 13.12 算术运算指令
- 13.13 FROM & TO 指令
- 13.14 变址寄存器

14. GX IEC Developer 使用 - (调试 / 监控 功能) 94 page

- 14.1 File Information
- 14.2 回路 监控 & Online Change Mode
- 14.3 登录 Data 监控
- 14.4 Buffer Memory 监控
- 14.5 Device 编辑
- 14.6 Cross Reference
- 14.7 变数 Import / Export
- 14.8 User Library 登录
- 14.9 Project File 管理

15. GX IEC Developer 使用 - (Sample Program) 110 page

- 15.1 课题 程序 1
- 15.2 课题 程序 2
- 15.3 课题 程序 3
- 15.4 Q64AD Sample Program



* 参考：FA 产品标准及注意事项（Mitsubishi FA 产品 基准）

参考1 SHOP别 CONFIGURATION

参考2 FA产品 标准 部品 LIST

参考3 设置及布线时 注意事项



1. 概 要

1.1 IEC61131-3的概要

最近在自动控制领域的特征是对Open System标准化的程序及相应的通信要求, 根据以上从 IEC(国际电气标准会议)的国际机关提出为了 标准化程序界面的 IEC61131-3 的标准案。

1.2 IEC61131-3的特征

- I. 程序的效率性增加
(程序的作成与问题掌握简单, 有高意识性)
- II. 国际标准规格
- III. 使用者最适合的语言选择
(在 IEC61131-3 委员会提出的 5种语言当中使用者根据自己的需求 而选择适当的语言及也可以混合使用)

1.3 IEC61131-3的分类

: IEC61131-3是 IEC61131构成的 第3部分. IEC61131的构成由以下所示

- ☞ Part 1 : General Overview
- ☞ Part 2 : Hardware
- ☞ Part 3 : **Programming Languages**
- ☞ Part 4 : User Guidelines
- ☞ Part 5 : Communication

1.4 IEC61131-3的语言

- Textual :
 - Instruction List (IL)
 - Structured Text (ST)
- Graphical :
 - Ladder Diagram (LD)
 - Function Block Diagram (FBD)
 - Sequential Function Chart (SFC)



2. GX IEC Developer 介绍

2.1 什么叫 GX IEC Developer?

: 根据 IEC61131-3 的标准而使用 MELSEC PLC 的软件工具

2.2 GX IEC Developer & GX Developer 比较

- Tab2-1: 比较表

GX Developer	GX IEC Developer
Easy to use	Programming according to IEC 1131
Available editors: <ul style="list-style-type: none"> • Instruction List • Ladder Diagram • Sequential Function Chart (SFC) 	Available editors : <ul style="list-style-type: none"> • Instruction List (IEC and MELSEC) • Ladder Diagram • Sequential Function Chart (SFC) • Structured Text • Function Block Diagram
Macros for System Q	Functions and Function Blocks
PLC Diagnostics	PLC Diagnostics
Network Diagnostics	Network Diagnostics

2.3 GX IEC Developer 的系统要求事项

① Hardware 要求事项

- Pentium II 350 Processor or above
- 32 MB RAM (Microsoft Windows 95/98/Me)
- 64 MB RAM (Microsoft Windows NT/2000 Professional)
- 128 MB RAM (Microsoft Windows XP)
- USB port
- Hard disk with at least 200 MB free
- CD-ROM drive
- 17" (43 cm) VGA monitor (1024 X 768 pixels)



⌚ Software 要求事项

: GX IEC Developer 是 32bit 的制品,支持以下的 OS.

- Microsoft Windows 98 / ME (ServicePack 1 或 以上)
- Microsoft Windows NT Workstation 4.0 (ServicePack 6 或 以上)
- Microsoft Windows 2000 Professional (ServicePack 2 或 以上)
- Microsoft Windows XP Professional (ServicePack 2 以上)
- Microsoft Windows XP Home Edition (ServicePack 2 以上)



3. GX IEC Developer 程序的构成要素

3.1 关于结构化程序

: 以前的程序是以各个指令为排列的集合形式的程序.

结构化程序是制作成各个程序以模块形式而区分的结构化程序.

这里的各个程序都是独立区分,构成单位是

“POUs (Program Organization Units)”

把以上的程序组成以所谓 **“Task”** 的群体而构成并施行

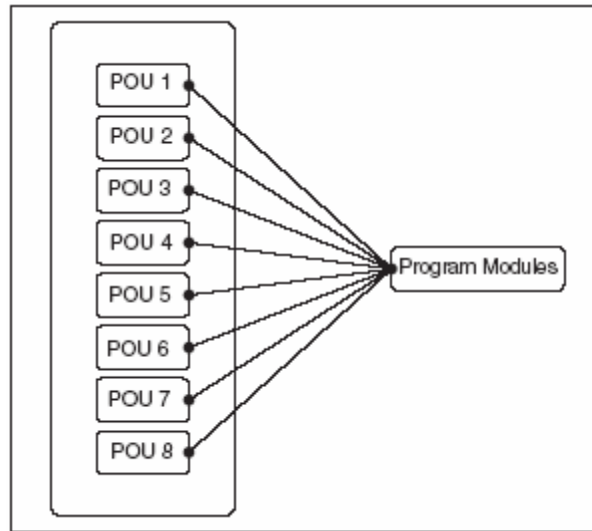


Fig3-1:

POU是所有程序里
组成Task的要素而使用

3.2 什么叫 POU(Program Organization Units)

: 在IEC61131-3的 PLC程序是以 Program Organization Units(POU)
的各个单一的程序模块而构成.

即, POU是顺控程序的最小独立的要素.

⇒ POU의 种类

- Programs
- Functions
- Function blocks

* 关于各个POU种类的说明在下一章里进行说明.



3.3 什么叫 Task

: 在 PLC里面所处理程序的最小单位.

即, POU是 构成程序的最小单位要素

Task是 由一个以上的POU组成而在处理PLC里使用的最小单位.

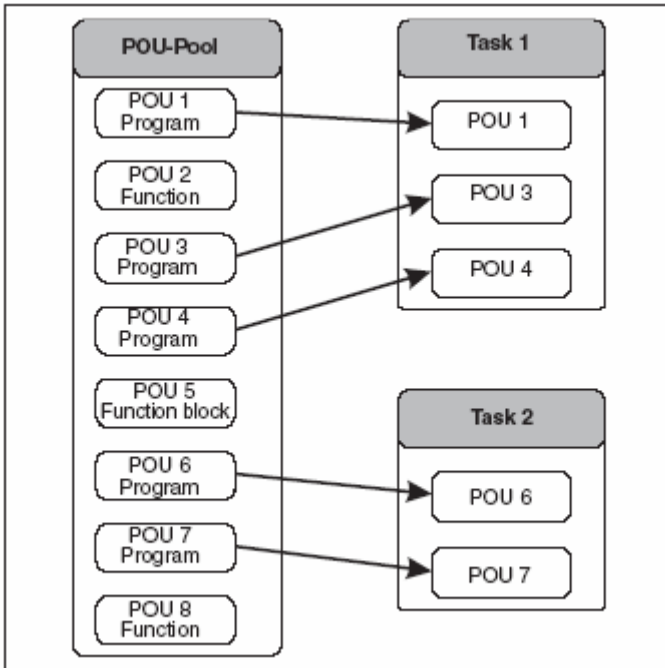


Fig3-2:

程序POU是在Tasks 组合并使用.

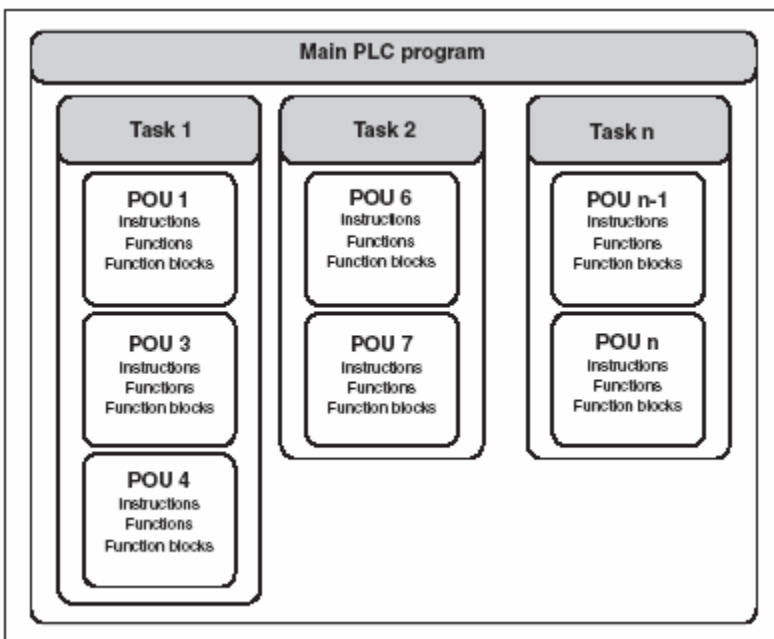


Fig3-3:

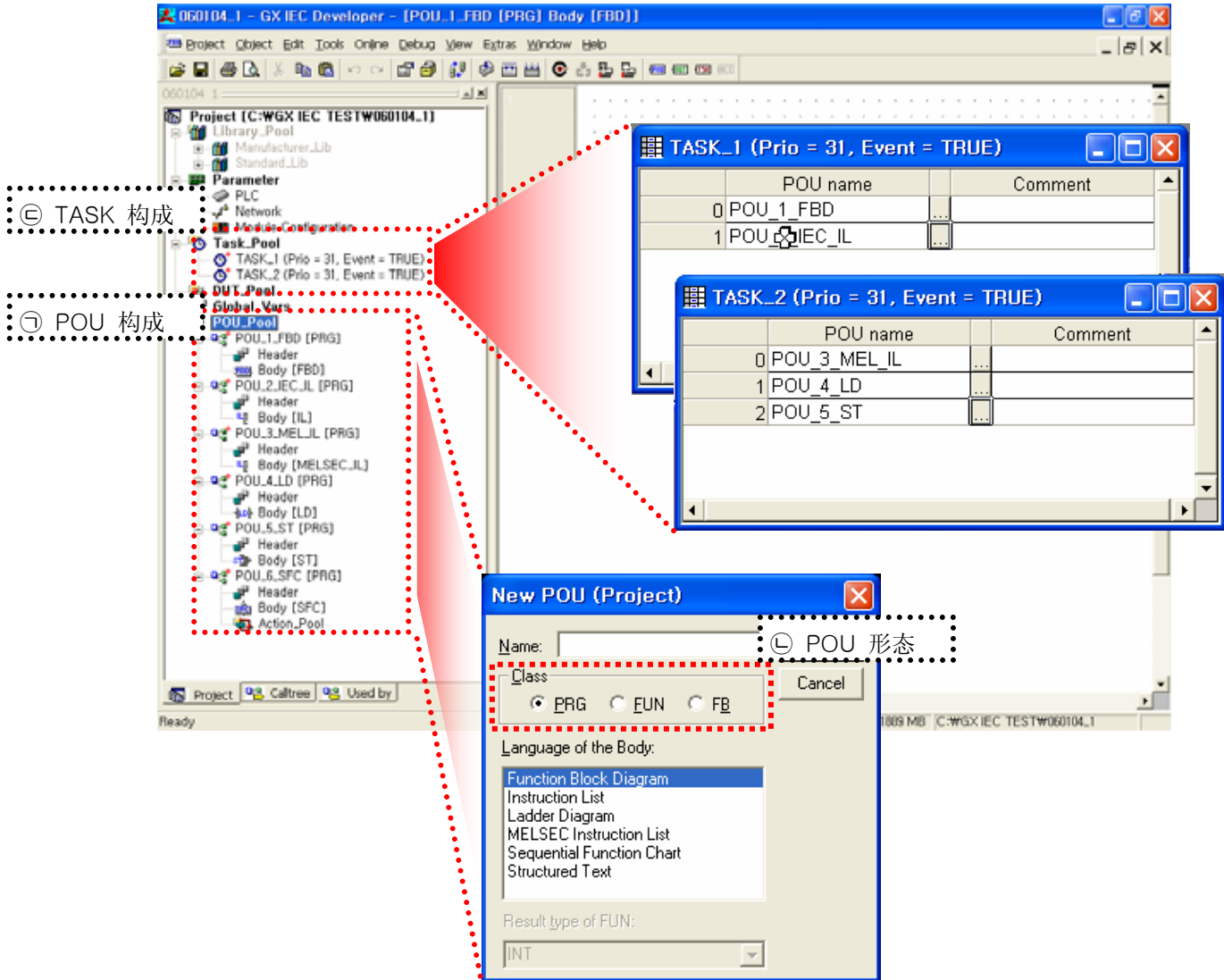
所有Task是为了 实现实际程序而 又组合在一起



3.4 GX IEC Developer的画面构成

: 关于GX IEC Developer画面构成的说明.

Fig3-4: POU & TASK 构成画面



说明

- ① 各个独立的 POU 作成. 由 Fig3-4所示 作成的 POU是以 POU Pool 而所综合管理.
- ② POU是可以作成 3种不同的形态, 根据各个形态也选择可以支持的语言种类.
- ③ TASK也可以分别区分化, 根据 各个 TASK而选择合适的POU 要素而登录. 再把这样的 TASK要素由 TASK Pool来综合管理.



4. POU의 Class

: POUs按功能分3Class来区分.

4.1 程序 (PRG)

: 由POU作成的3个Class中 PLC运转最主要的Class, 即由函数库, FUN, FB 都由程序指令来构成.

- * 参考 : 在TASK能登录的只有 POU作成的[PRG] Class. 即, 只有登录在TASK才能在 PLC运转时使用 实际运转程序就是由[PRG] Class来作成.

参考 能在TASK 登录的只有 POU来作成的 [PRG] Class. 即, 只有登录在TASK才能在 PLC运转时使用 实际运转程序就是由[PRG] Class来作成.

4.1.1 在[PRG]里 使用的语言有以下 6种.

① Instruction List (IL)

: IL是最底级别的语言.用于适合少数用途的特征部分的语言

GX IEC Developer IL 的语言有 2种区分.

- MELSEC IL
- IEC IL

Fig4-1: IEC Instruction List (IL)

IEC_AWL_POE [PRG] Body [IL]	
1	LD FALSE ST M101
2 Start:	LD Hauptschalter AND M100 ST M102
3	LD M100 ANDN M1 CJ_M Start

Fig4-2: MELSEC Instruction List

MELSEC_MAIN_PRG [PRG] Body [MELSEC_IL]	
1	LD M3 LD> K100 D3 OR M8 ANB OUT Y33



④ Structured Text (ST)

: 是高级别结构化的语言. 这种语言主要以图表形式表现, 所以主要用在难以充分表现内容的时候适用.

```

TEST [PRG] Body [ST]
CASE D0 + D2 + D3 OF
1, 5: D10 := 2;
-6, -4, -2: D10 := 4;
ELSE D10 := 6;
END_CASE;
    
```

Fig4-3: Structured Text (ST)

⑤ Ladder Diagram (LD)

: Boolean演算的图表表现方式.
LD与 电气 Relay回路一起使用

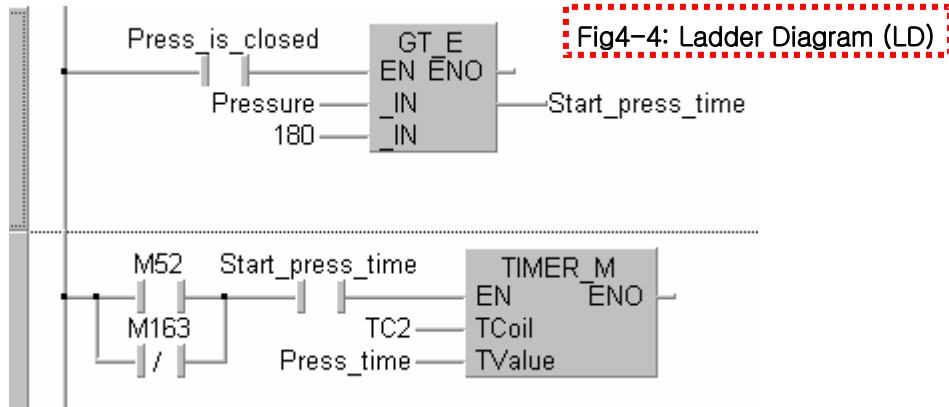


Fig4-4: Ladder Diagram (LD)

⑥ Function Block Diagram (FBD)

: 各种形式的长方形(箱子)来与功能块接触以图表来表现的一种方法

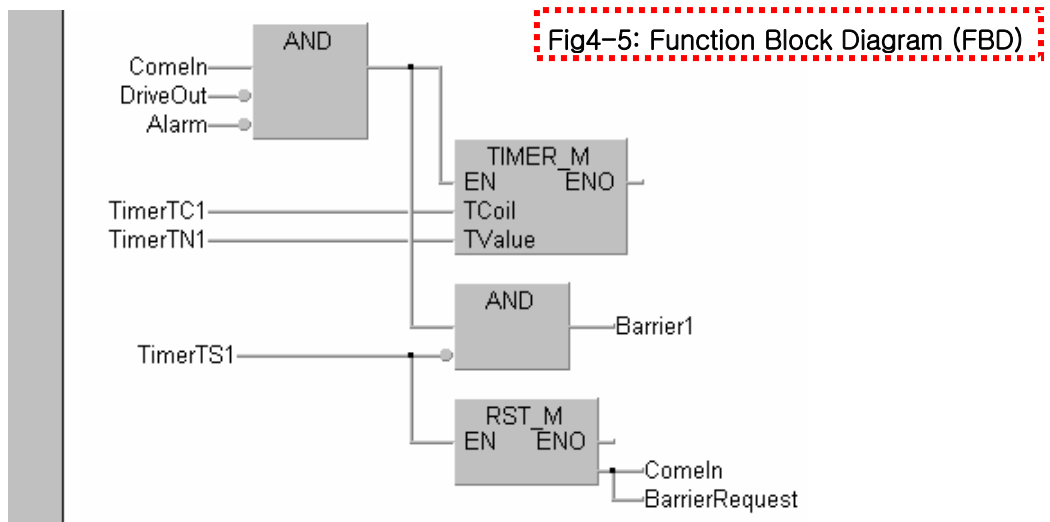


Fig4-5: Function Block Diagram (FBD)



④ Sequential Function Chart (SFC)

: 表现运作过程的步骤与步骤之间的移动条件部分（transition）的两种图表记号来组成，
步骤里面的运作是由别的语言（ST, IL, LD, FBD)来记述.

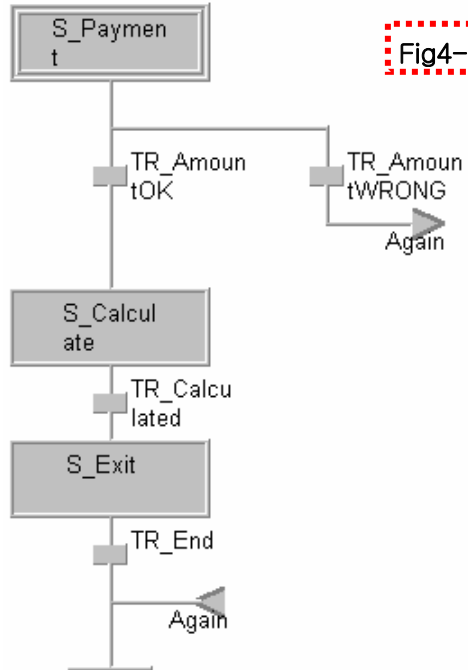


Fig4-6: Sequential Function Chart (SFC)



4.2 FUNCTION (FUN)

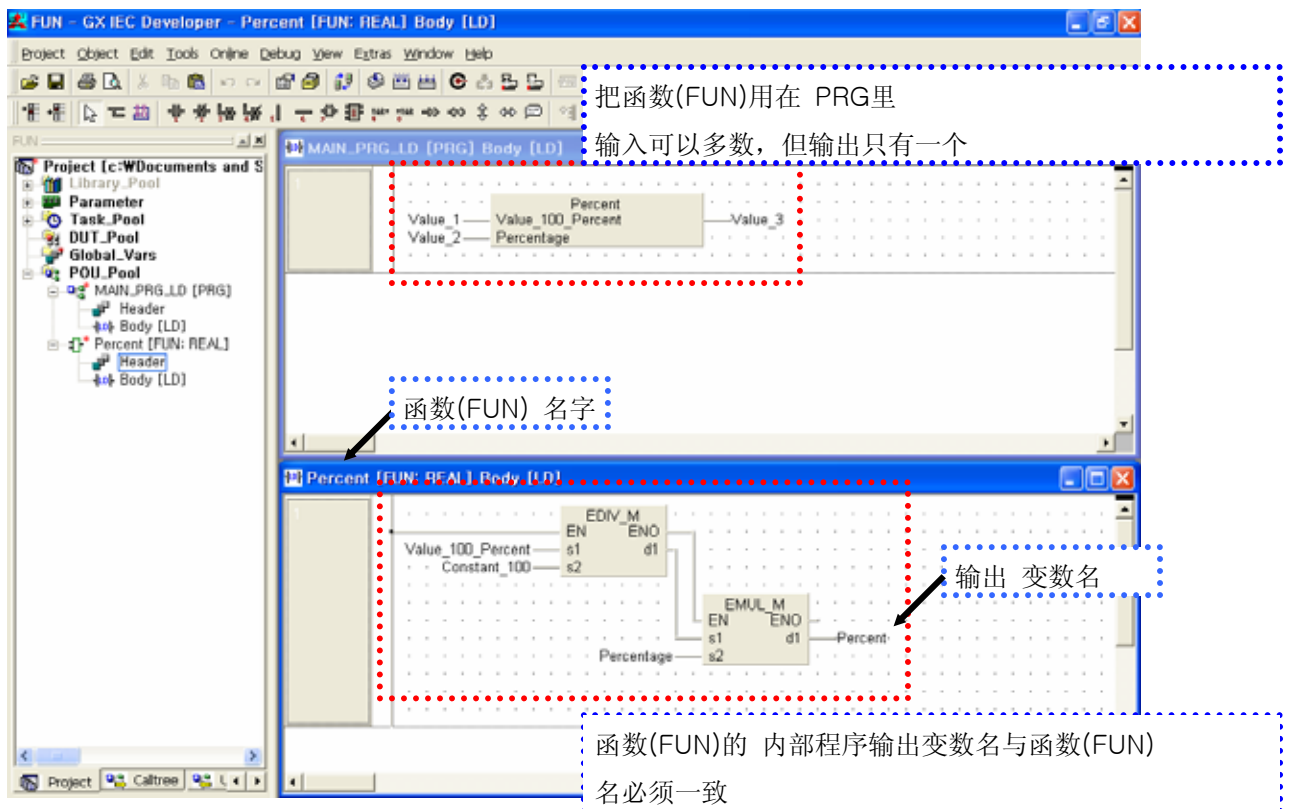
: 所谓FUNCTION(FUN)是 为了表现出特定功能的最基本的单位.

4.2.1 特征

- ① 有独立的组成程序的要素.
- ② 没有Memory的 Subroutine.
结果就是同一个输入参数引出函数(FUN)(Subroutine)的话
输出同样的结果.
即, 不会保存内部情报信息.
- ③ 但 只有输出一个
- ④ 函数(FUN)单元里可以使用别的函数(FUN)单元, 但不能使用功能块(FB)

注意 制作函数(FUN)的时候 函数(FUN)的输出函数(FUN)名与输出
变数名一致

Fig4-7: 函数(FUN)的 构成





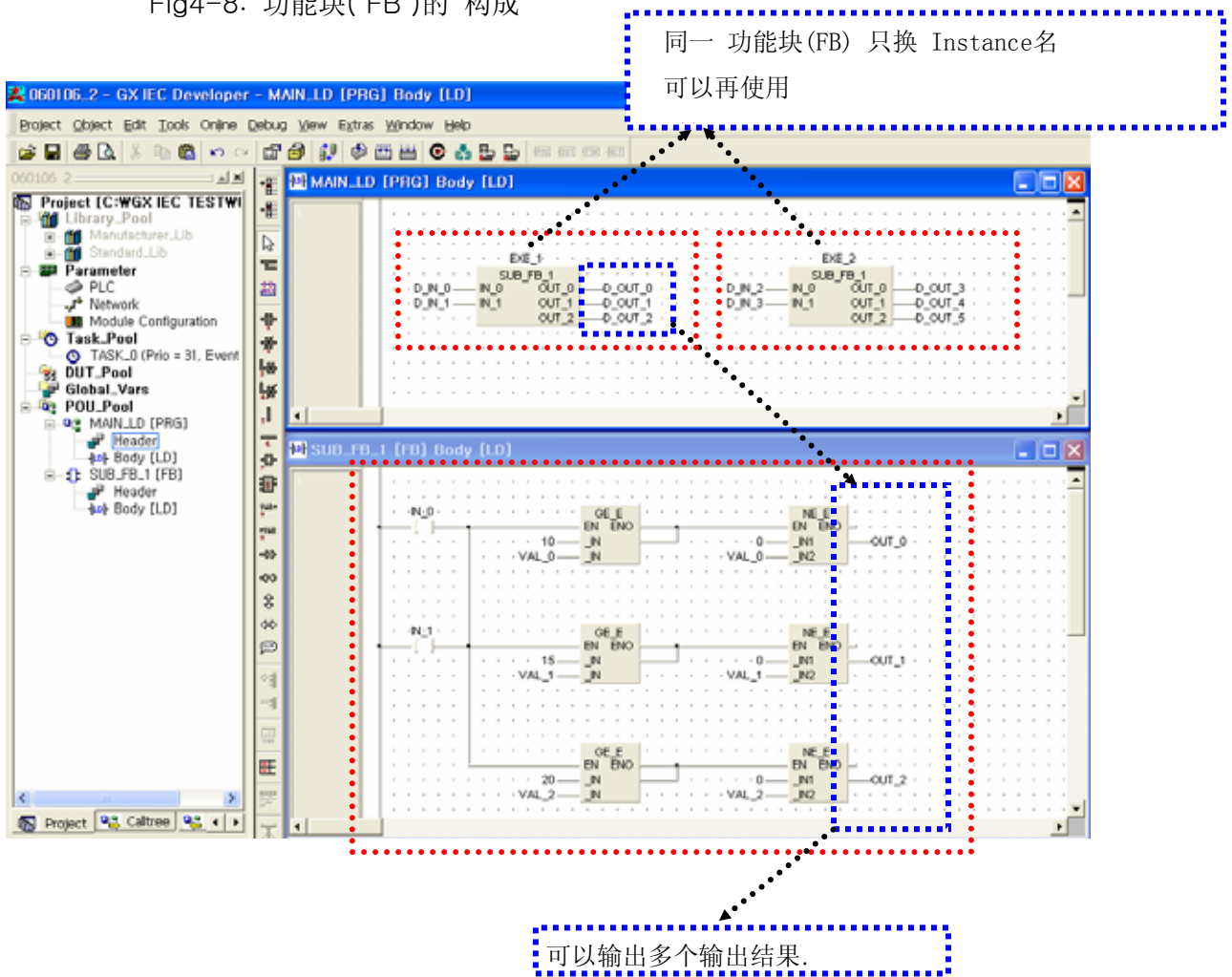
4.3 功能块 (FB)

: 功能块(FB)是由一个单位来使用的连续的功能的集合.

4.3.1 特征

- ① 有独立的程序要素.
- ② 有Memory的 Subroutine.
结果就是被保存的Data可以在下一个功能块(FB)里引出使用
- ③ 即, 可以保存内部状态的情报.
- ④ 可以有多个输出
- ⑤ 功能块(FB)可以使用别的功能块(FB),与别的FUNCTION

Fig4-8: 功能块(FB)的 构成





5. POU的 构成

: 作成POU的话基本上由 Header & Body来构成.

Fig5-1: Project构成

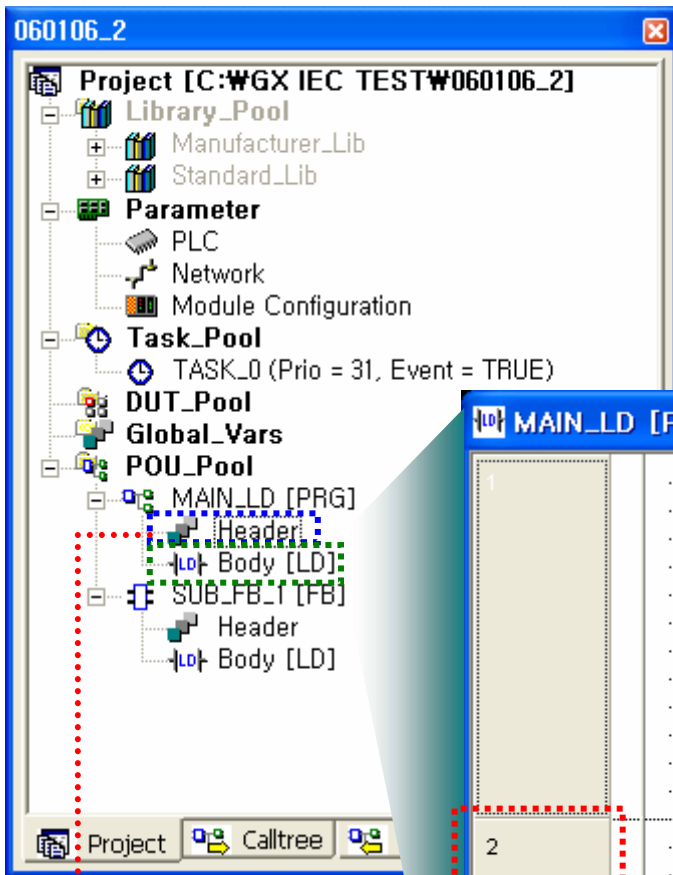


Fig5-3: Body 构成

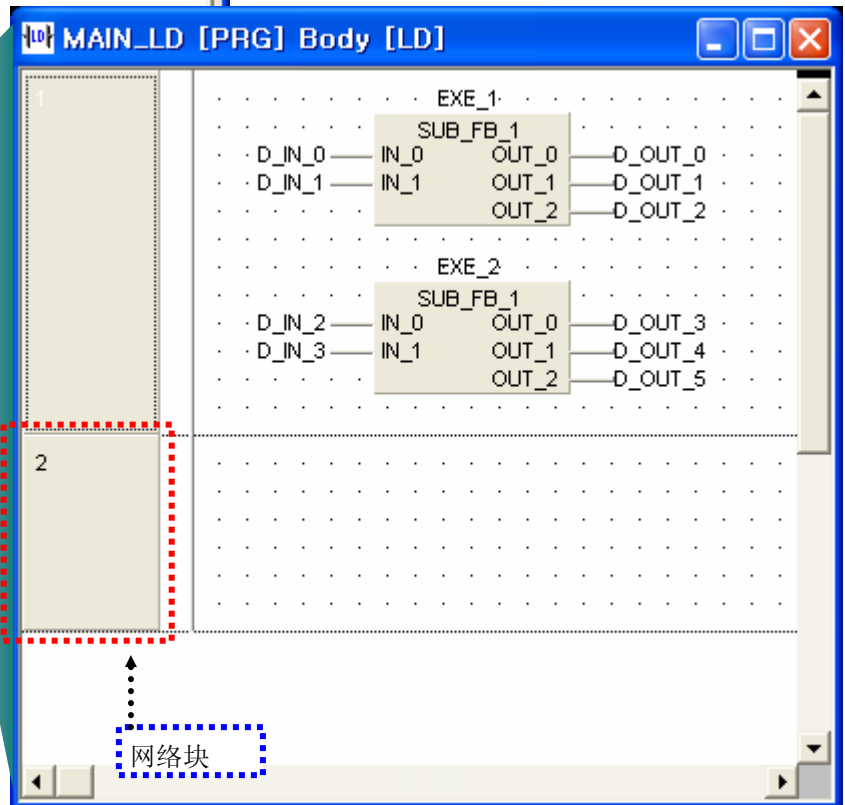


Fig5-2: Header 构成

	Class	Identifier	Type	Initial	Comment
0	VAR	EXE_1	SUB_FB_1	...	
1	VAR	EXE_2	SUB_FB_1	...	



5.1 POU的 Header

: POU Header是保存在POU程序里使用的变数
与 选定的用途来使用.

5.1.1 Header 构成

- ① [Fig5-1]里看 POU是由 Header与 Body来构成,
选择Header的话 激活[Fig5-2]一样的画面.
- ② 从[Fig5-2]里可以看出 Header是由 5种要素组成.
 - ☞ Class : 定义变数的性质.
 - ☞ Identifier : 定义变数的名字.
 - ☞ Type : 定义变数所表现的data形态.
 - ☞ Initial : 根据变数data形态的初始值.
 - ☞ Comment : 关于定义变数的说明

* 关于上述的说明在下一章里详细说明.

5.2 POU的 Body

: POU的 Body是 拥有PLC 程序的实行情报.

5.2.1 Body 构成

- ① Body是 由[Fig5-3]里看出是由可以作成 PLC程序的
“网络”来构成.
 - ** 所谓网络
 - : PLC 程序是由适用于运转程序单元来构成.
 - 这样的程序单元叫做网络.

参 考

各个网络里只有包含一个连续的回路
即,只有一个输出.
但是,一个网络里面可以作成多个输出的程序
的时候在驱动方面不会出现错误
但会出现警告提示。



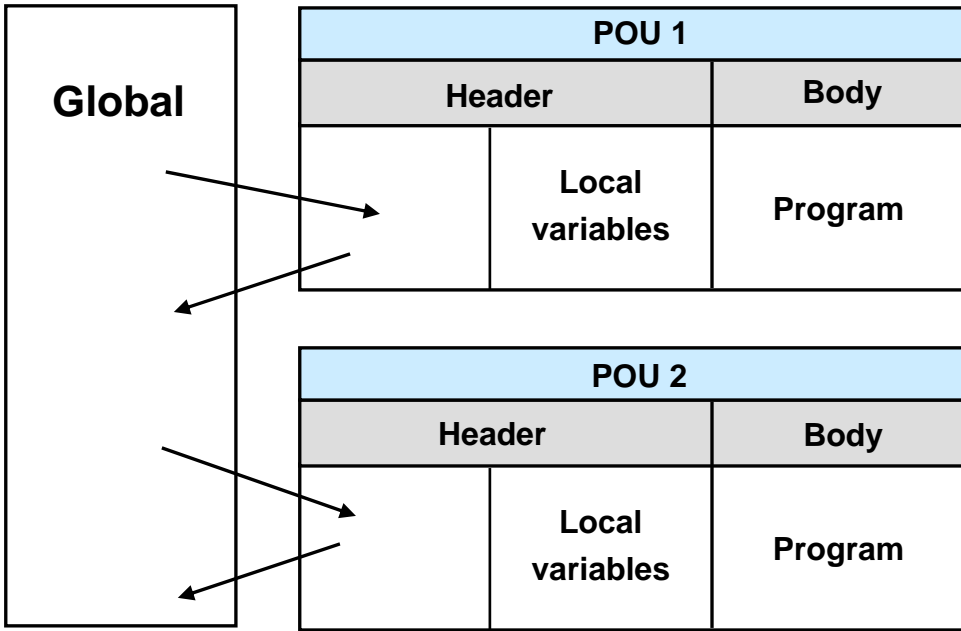
6. 变数 (Variable)

: 变数包含 PLC的输入, 输出或者内部Device值.

变数有两种不同形态来区分.

- Global Variables (全域 变数)
- Local Variables (局部 变数)

Fig6-1: 变数的 构成



6.1 Global Variables (全域变数)

: Global变数是 登录在 Project 目录的, 在所有 POU里面可以自由的使用. 跟着也可以进行在各个POU里data的交换

Fig6-2 : Global Variables 设定 画面

Global Variable List								
	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial	Comment	Remark
0	VAR_GLOBAL	D_IN_0	X0	%IX0	BOOL	FALSE		
1	VAR_GLOBAL	D_IN_1	X1	%IX1	BOOL	FALSE		
2	VAR_GLOBAL	D_IN_2	X2	%IX2	BOOL	FALSE		
3	VAR_GLOBAL	D_IN_3	X3	%IX3	BOOL	FALSE		
4	VAR_GLOBAL	D_OUT_0	Y20	%QX32	BOOL	FALSE		
5	VAR_GLOBAL	D_OUT_1	Y21	%QX33	BOOL	FALSE		
6	VAR_GLOBAL	D_OUT_2	Y22	%QX34	BOOL	FALSE		
7	VAR_GLOBAL	D_OUT_3	Y23	%QX35	BOOL	FALSE		
8	VAR_GLOBAL	D_OUT_4	Y24	%QX36	BOOL	FALSE		
9	VAR_GLOBAL	D_OUT_5	Y25	%QX37	BOOL	FALSE		

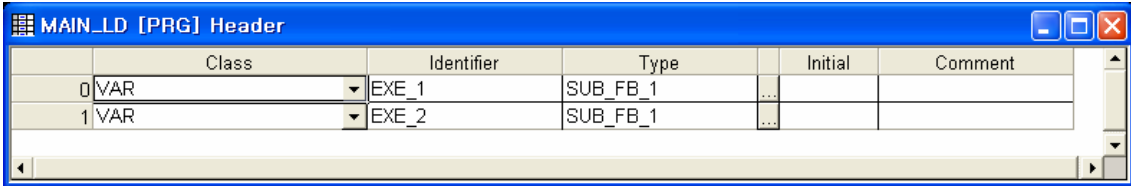


6.2 Local Variables (局部变数)

: Local 变数只有在一个 POU里才能使用.

还有 Local变数只有在自己被定义的POU里才能使用.

Fig6-3 : Local Variables 设定 画面



6.3 变数选定

: 首先进行实际 Programming之前选定全部 Project 即 各个 POU里所使用的变数.

6.3.1 变数选定必要要素

: 参考 [Fig6-2] & [Fig6-3].

⇒ **Class**

: 能够在Project里使用而指定变数的属性.

Tab.6-1: Class的适用

Class	POUs 种类			说明
	PRG	FUN	FB	
VAR	○	○	○	使用在 POU的 Local 变数来使用
VAR_CONSTANT	○	○	○	使用在 POU的 Local 变数来使用, 还有变数的 Data是固定常数的时候使用
VAR_INPUT	-	○	○	使用在 POU的 外部输入
VAR_OUTPUT	-	-	○	使用在 POU的 外部输出
VAR_IN_OUT	-	-	○	使用在 POU的 输入既输出要素的变数使用可能
VAR_GLOBAL	○	-	○	Global 变数来使用
VAR_GLOBAL_CONSTANT	○	-	○	Global 变数来使用, Data是常数的时候使用



⇒ Identifier (标示符)

: 各个变数都有象征性的地址(名字)。这样的地址(名字)可以自由的选择,但是地址开头必须是由英文字母及下划线来组成。

identifiers的 例 : S02.3
 Drive_2_ready
 Open_valve_1
 Motor_M1_ON
 _DEFEB_2

- | | |
|------------|---|
| 参 考 | <ul style="list-style-type: none"> - Identifier不能用韩语作成. - 不能使用运算符号的文字 |
|------------|---|

⇒ Absolute address (只有在Global Variables里使用)

: Absolute address是 显示输入,输出,内部继电器的内存记忆位置的地址。只有在Global 变数指定里使用,没有指定的时候自动设定成任意的地址。Absolute address是IEC体系及MITSUBISHI体系来区分两种都可以使用。

☞ IEC 体系 % : 为了 IEC 体系的标示符
 I, Q, M : 输入, 输出, 内部继电器
 X, W, D : Boolean, word, double word

Tab.6-2: Absolute address例

IEC	MITSUBISHI	意 义
%IX15	X0F	输入 X0F
%QX3	Y3	输出 Y3
%MW0.100	D100	Data register D100



⇒ Data Type

: Data Type是变数拥有的范围值或定义与bit值一样的变数特性

Tab.6-3: Data Type的 定义

	Data type	Value range	Size
BOOL	Boolean	0 (FALSE), 1 (TRUE)	1 bit
INT	Integer	-32768 to +32767	16 bits
DINT	Double Integer	-2,147,483,648 to 2,147,483,647	32 bits
WORD	Bit string 16	0 to 65,535	16 bits
DWORD	Bit string 32	0 to 4,294,967,295	32 bits
REAL	Floating point value	3.4E +/-38 (7 digits)	32 bits
TIME	Time value	-T#24d0h31m23s648.00ms to T#24d20h31m23s647.00 ms	32 bits
STRING (Q 系列)	Character string	最大 50 characters	

⇒ Initial value

: 初始值由系统自动设定,所以不能随意变更.

⇒ Comment

: 可以作成关于变数的说明文. (最大 64K characters)

⇒ Remark (只有在Global Variables里适用)

: Comment以外可以追加使用者情报.



6.5 Data 形态的阶层

: 下图是在 GX IEC里的Data 阶层表.

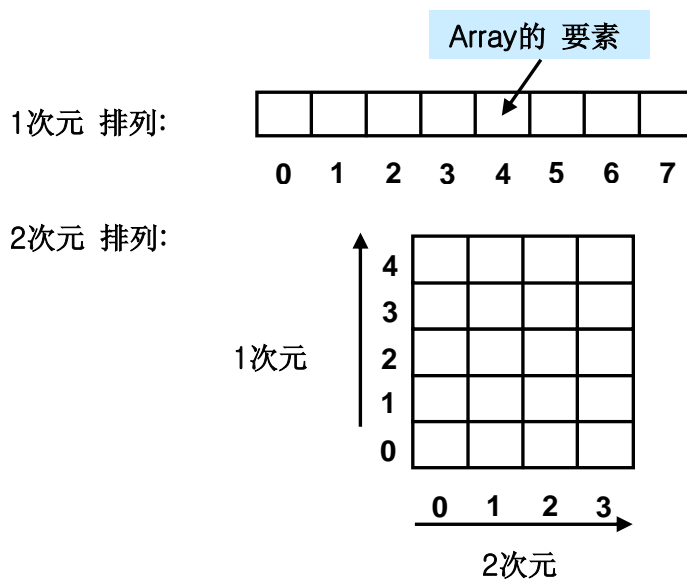
ANY					
ANY_SIMPLE				ARRAY	Data Unit Types (DUT)
ANY_BIT	ANY_NUM		TIME		
BOOL	ANY_INT	ANY_REAL			
WORD	INT	REAL			
DWORD	DINT				

6.5.1 Array

: 同一 Data形态变数的集合

- 适用 -

1. 在局部变数及全域变数里都可以使用
2. 可以使用3次元排列.





- 说明 -

	Class	Identifier	Type	Initial
0	VAR	Array_1_Dim	ARRAY [0..7] OF INT	[8(0)]
1	VAR	Array_2_Dim	ARRAY [0..3,0..4] OF DINT	[20(0)]

* 说明

- 有8个 INT 形态的 Data的 1次元排列

* 说明

- 有20个DINT形态的 Data的 2次元排列
- 1次元是 4个要素, 2次元是 5个要素构成



7. GX IEC Developer 使用 - (初期 设定 1)

7.1 New Project 制作

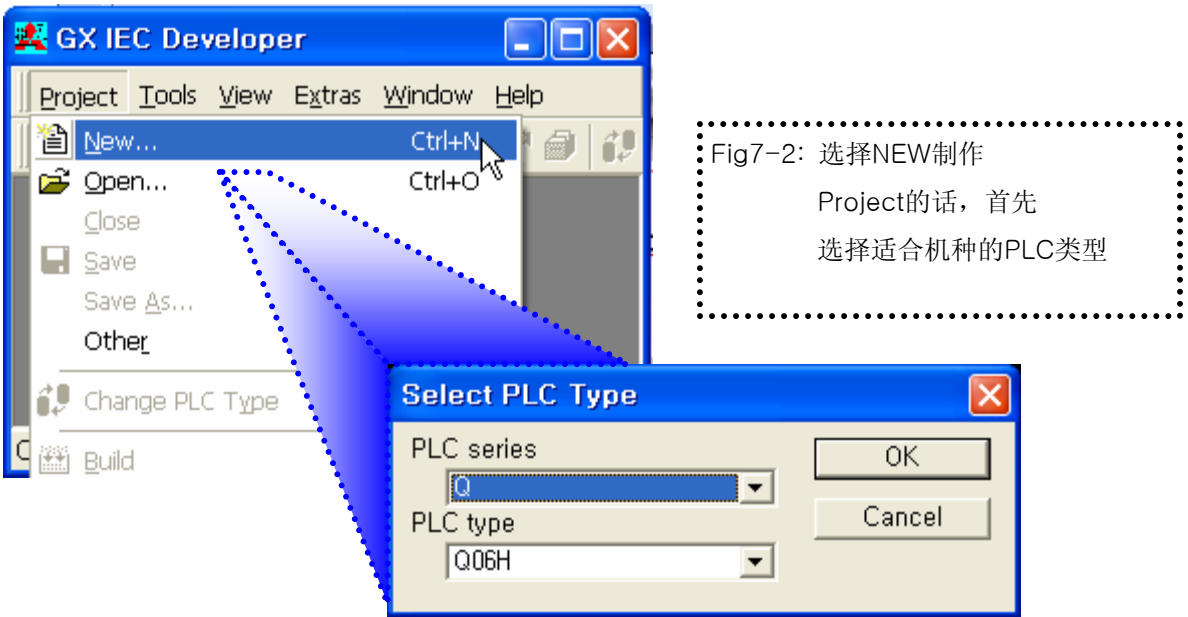
㊶ GX IEC Developer 开始

: 开始 > 程序 > MELSOFT Application > GX IEC Developer



㊷ 为了制作新工程而设定 PLC Type

: Project > New

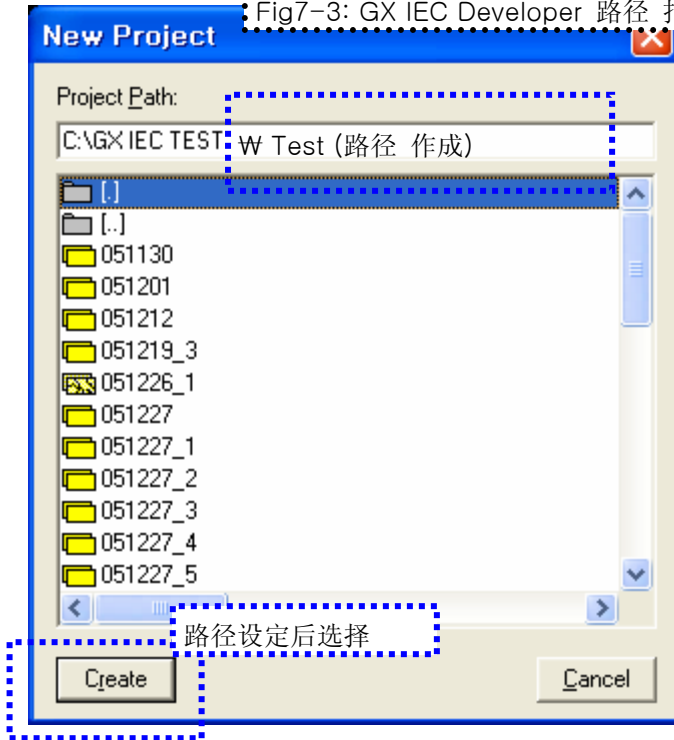




㊦ 指定 新工程的路径

: PLC Type에서 [OK] 选择 自动生成

Fig7-3: GX IEC Developer 路径 指定 画面

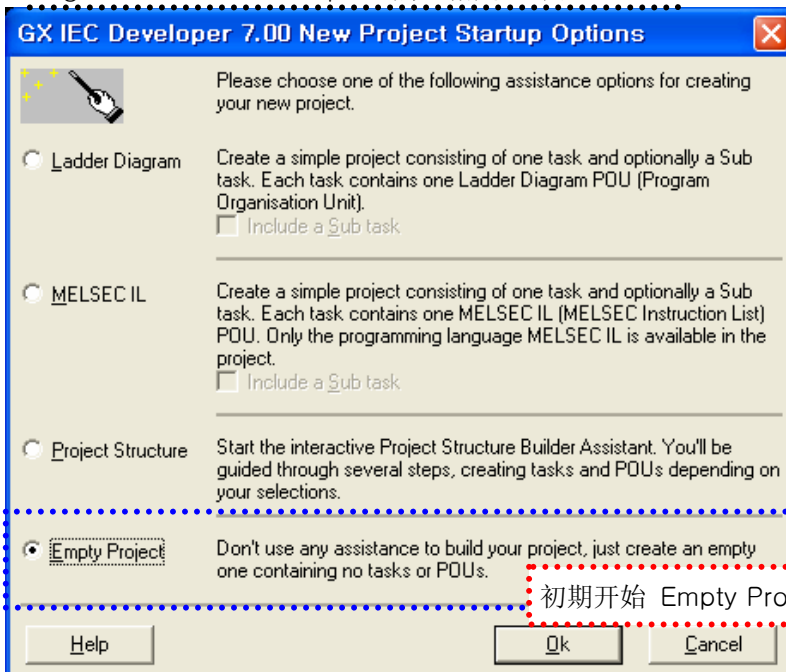


㊦ 工程选项

: 为了制作 Project而设定的初期选项.

根据选项的选择形成工程的初始化状态也不一样.

Fig7-4: GX IEC Developer 路径 指定 画面





⇒Ladder Diagram (LD)

：制作工程时 把基本工程用LD形态来制作.

EX.) 在GX Developer里作成工程的话 与“MAIN”的程序自动形成的一样.

⇒MELSEC IL

：作成工程时 把基本工程用 MELSEC IL 形态来制作

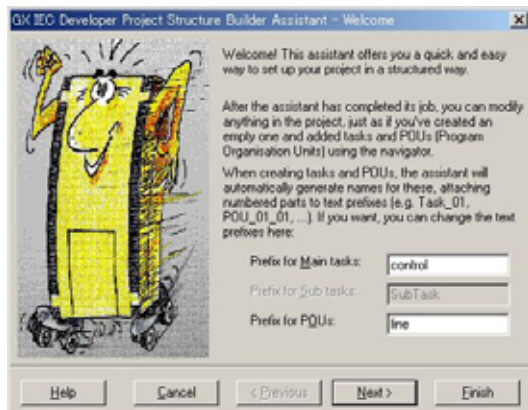
⇒Project Structure

：把 在工程里作成的Task 及 POU等初期设定里根据使用者设定而制作并开始.

Ex: 作成 例

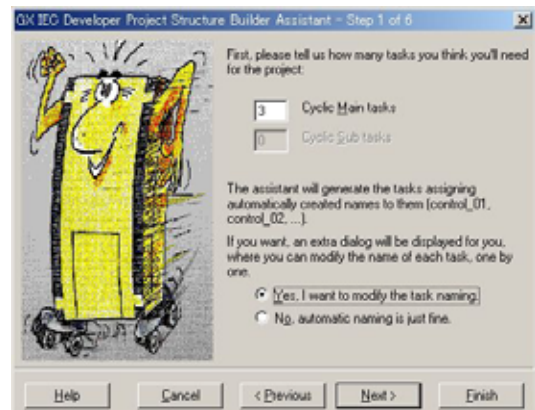
Welcome画面

POU与Task的 名字输入



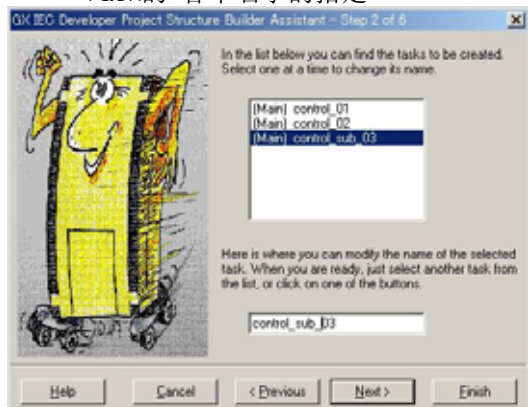
STEP1

Task的 个数与名字指定方法



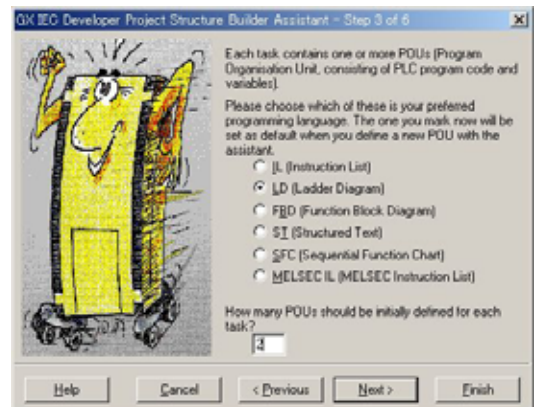
STEP2

Task的 各个名字的指定



STEP3

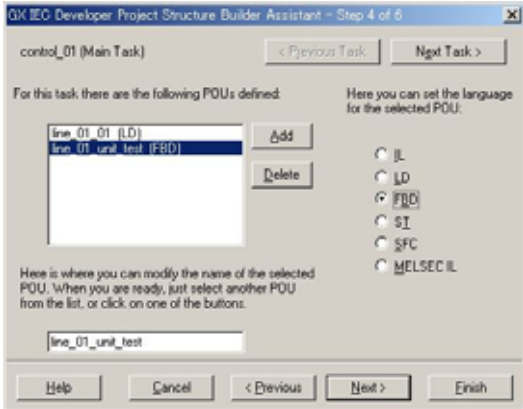
1 Task当POU的 个数与语言





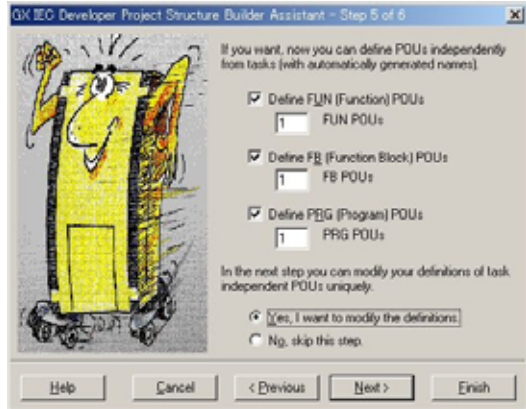
STEP4

对全部 POU指定名字及语言



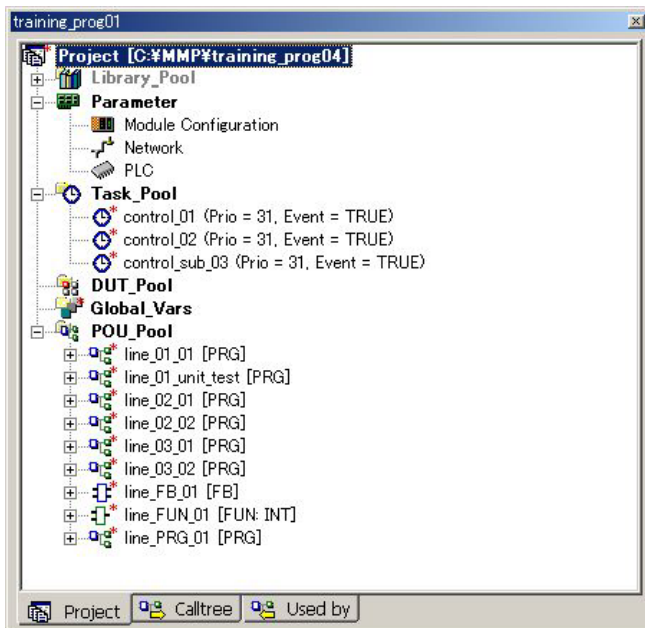
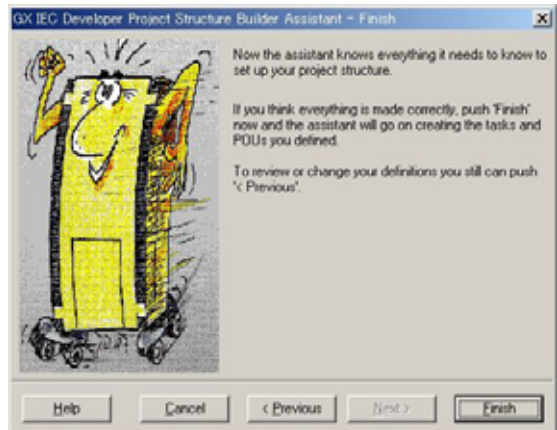
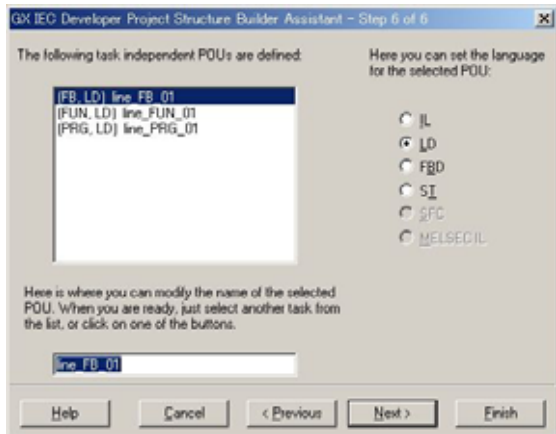
STEP5

不能分配在Task的 POU作成



STEP6

不能在Task分配的POU名字,语言的指定





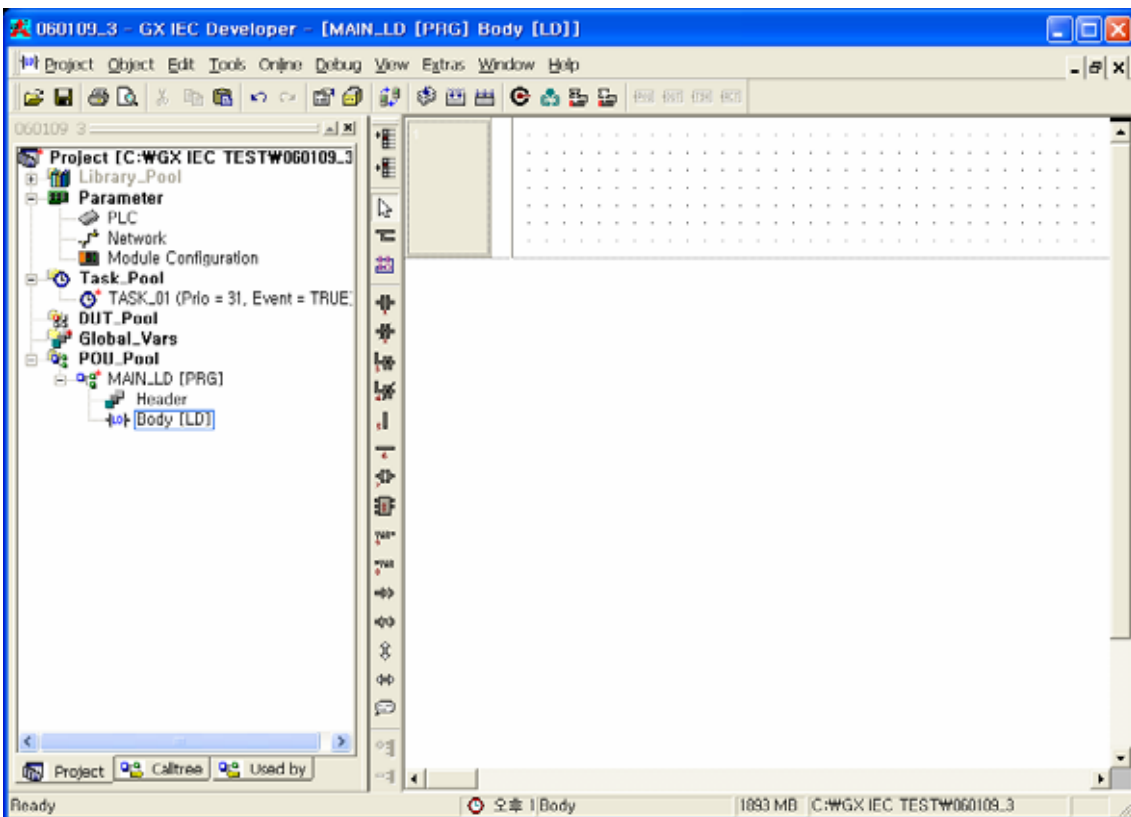
⇒Empty Project

: 在工程里基本能够作成的Pool只存在而没有内容
使用者形成工程以后一个一个制作并使用

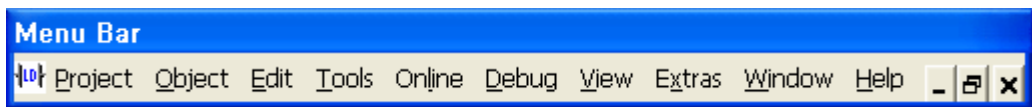
7.2 Project 画面 构成

: 简略说明工程制作后的画面.

Fig7-5 : GX IEC Developer的 工程画面构成



① 工程的 Main Menu Bar



② 工程的基本 Menu

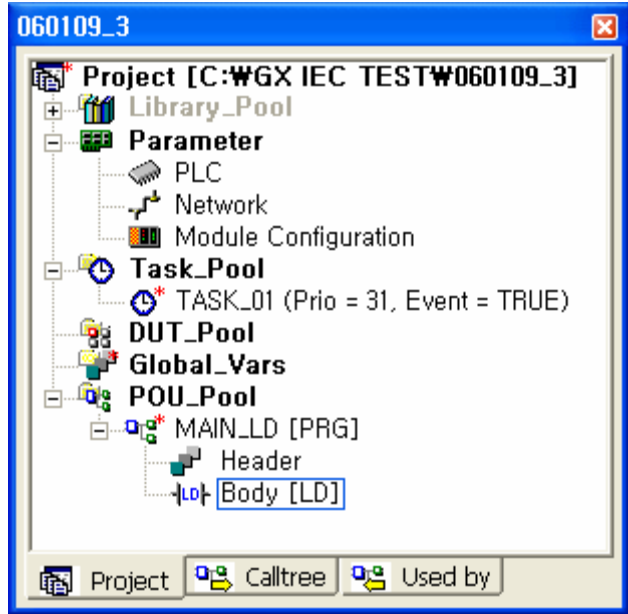




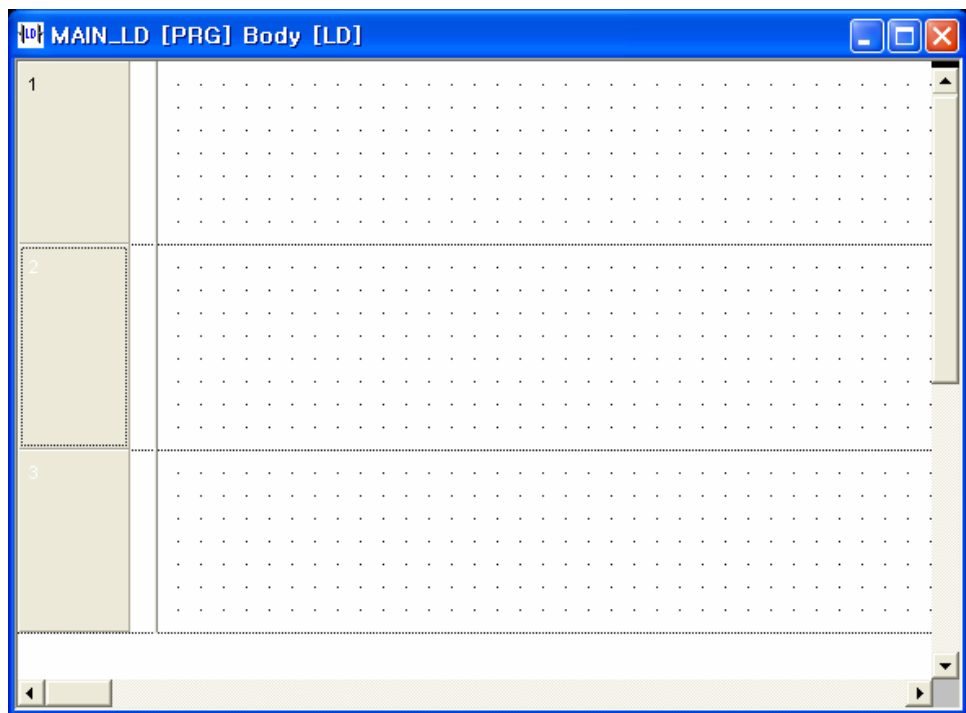
③ LD 制作 Menu



④ 工程的目录画面



⑤ 程序制作画面





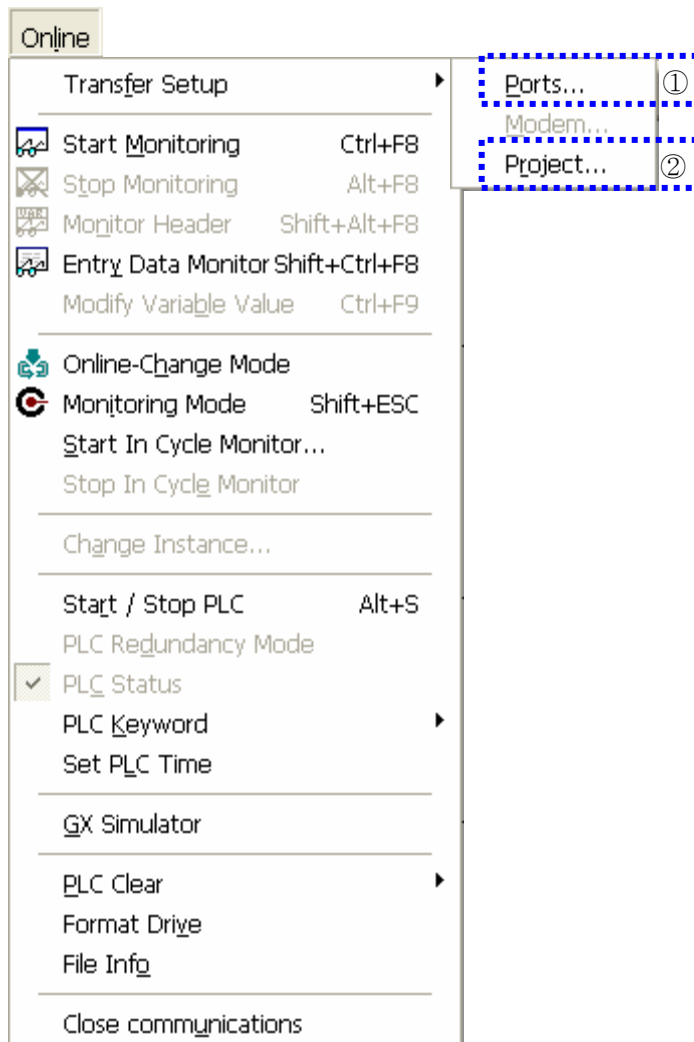
8. GX IEC Developer 使用 - (初期 设定 2)

8.1 PLC 传送设定

: GX IEC Developer与被激活的PC, PLC程序, 其他Data的进行通信的通信Port的设定及 用PLC进行Data写入存储器时的选择及其他附属内容的选择.

(Port 设定与GX Developer的连接设定内容同一)

Fig8-1: Transfer Setup (连接 对象 指定)





① Port 设定

: 选择 PC - PLC 连接 设定

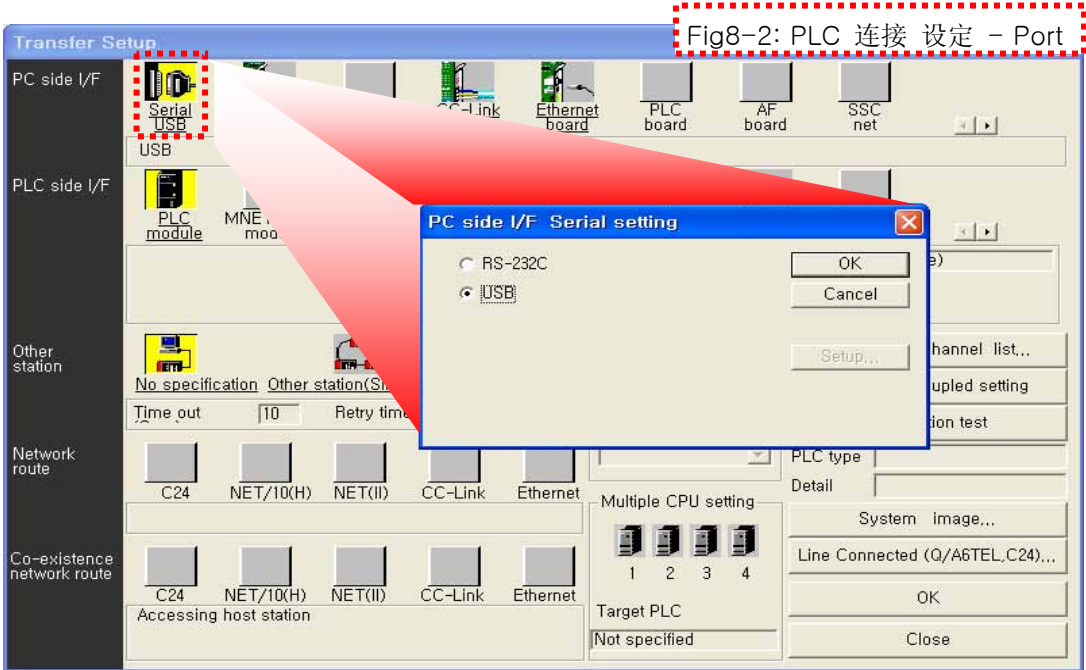


Fig8-2: PLC 连接 设定 - Port

② Project 设定

: 指定用PLC下载项目及位置.

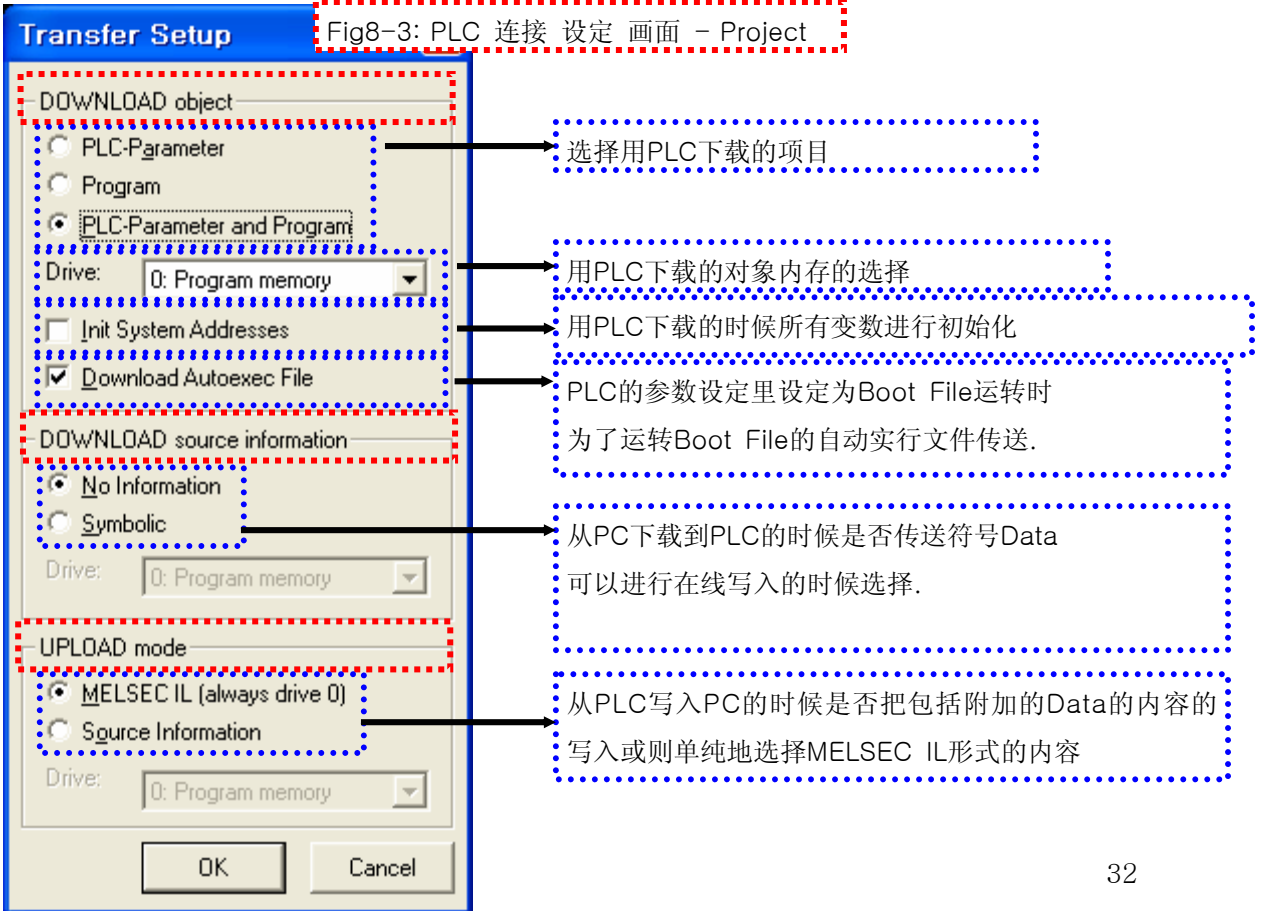


Fig8-3: PLC 连接 设定 画面 - Project

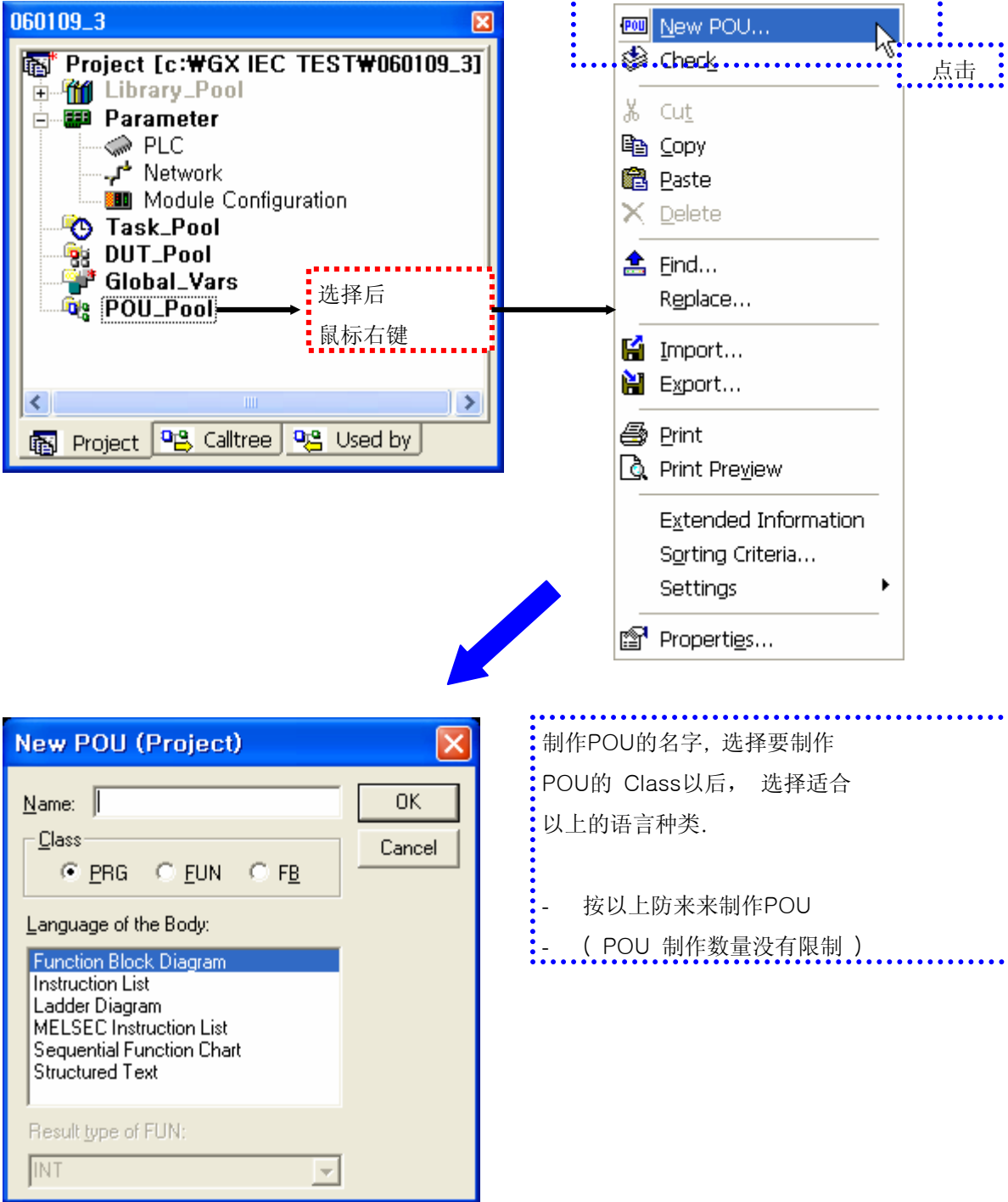
- 选择用PLC下载的项目
- 用PLC下载的对象内存的选择
- 用PLC下载的时候所有变数进行初始化
- PLC的参数设定里设定为Boot File运转时为了运转Boot File的自动实行文件传送.
- 从PC下载到PLC的时候是否传送符号Data可以进行在线写入的时候选择.
- 从PLC写入PC的时候是否把包括附加的Data的内容的写入或则单纯地选择MELSEC IL形式的内容



8.2 POU 制作 方法

: 关于POU制作的说明

Fig8-4: POU制作 顺序



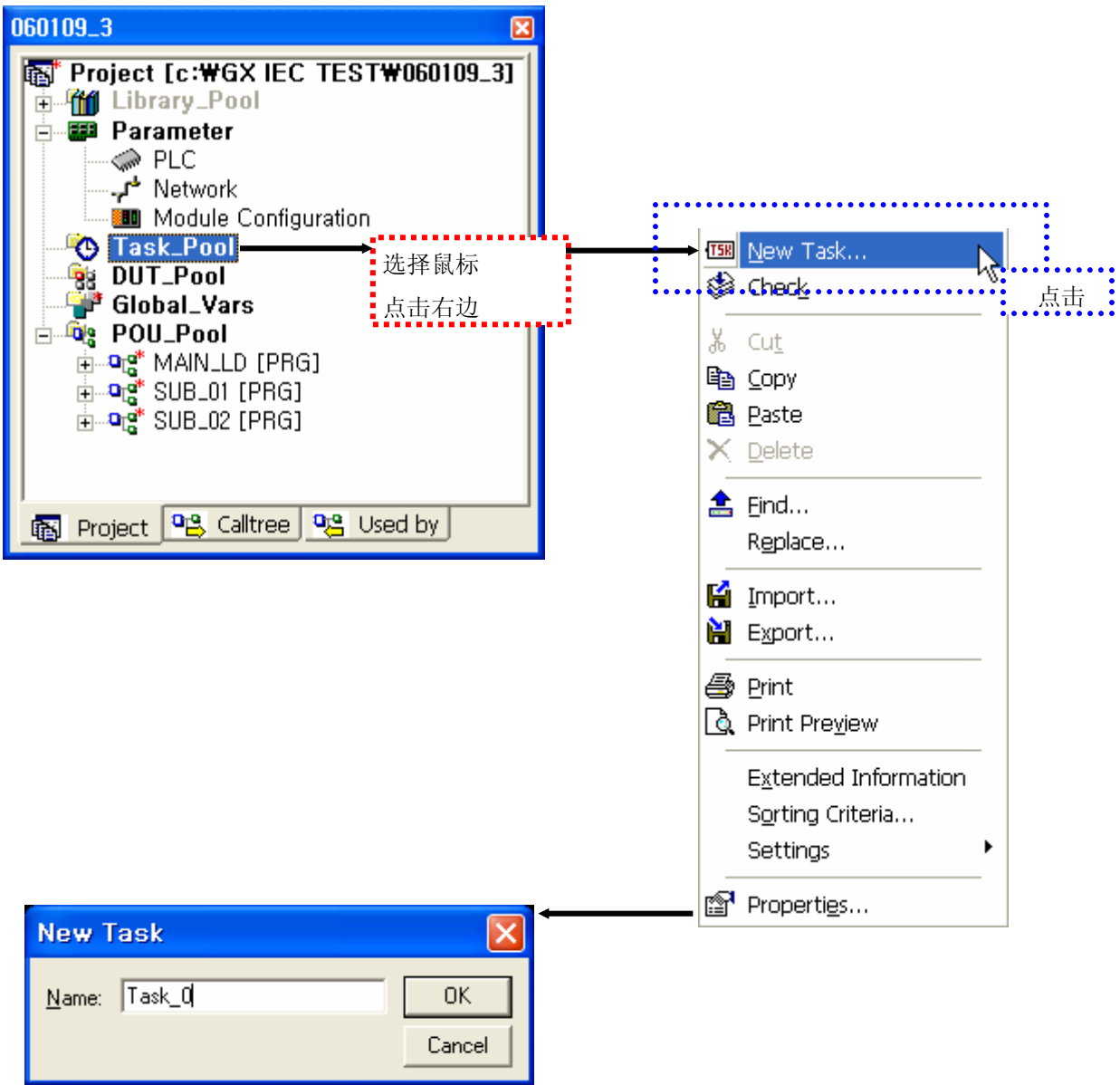


8.3 Task 制作方法

: 关于Task的制作及登录的说明.

8.3.1 Task 制作

Fig8-5: Task 制作 顺序





8.3.2 在Task登录POU的方法

Fig8-5: Task 登录 顺序

Task_0 (Prio = 31, Event = TRUE)

POU name	Comment
0	

点击

Program Selection

Libraries:

- <ALL>
- <Project>
- Manufacturer_Lib
- Standard_Lib

Programs:

- MAIN_LD
- SUB_01
- SUB_02

在形成的POU中找选择.



8.4 变数(Variable)设定

: 说明关于变数设定的方法.

8.2.1 Global Variables (全域变数) 设定

Fig8-5: Global的变数登录顺序

060109_3

Project [c:\WGX IEC TEST\W060109_3]

- Library_Pool
- Parameter
 - PLC
 - Network
 - Module Configuration
- Task_Pool
- DUT_Pool
- Global_Vars**
- POU_Pool

Project Calltree Used by

双击

制作变数的时候利用以下快捷键
作成变数LIST并使用

Declaratio...

Class	Identifier	MIT-Addr	IEC-Addr	Type	Initial	Comment	Remark
VAR_GLOBAL							
VAR_GLOBAL							
VAR_GLOBAL CONSTANT							

单击

选择输入表示符, 地址是从三菱地址
或IEC地址其中的一个

选择变数的Data 形态.

Type Selection

Libraries: <ALL>

Types:

- ARRAY
- BOOL
- DINT
- DWORD
- INT
- REAL
- STRING[32]
- TIME
- WORD

Type Class

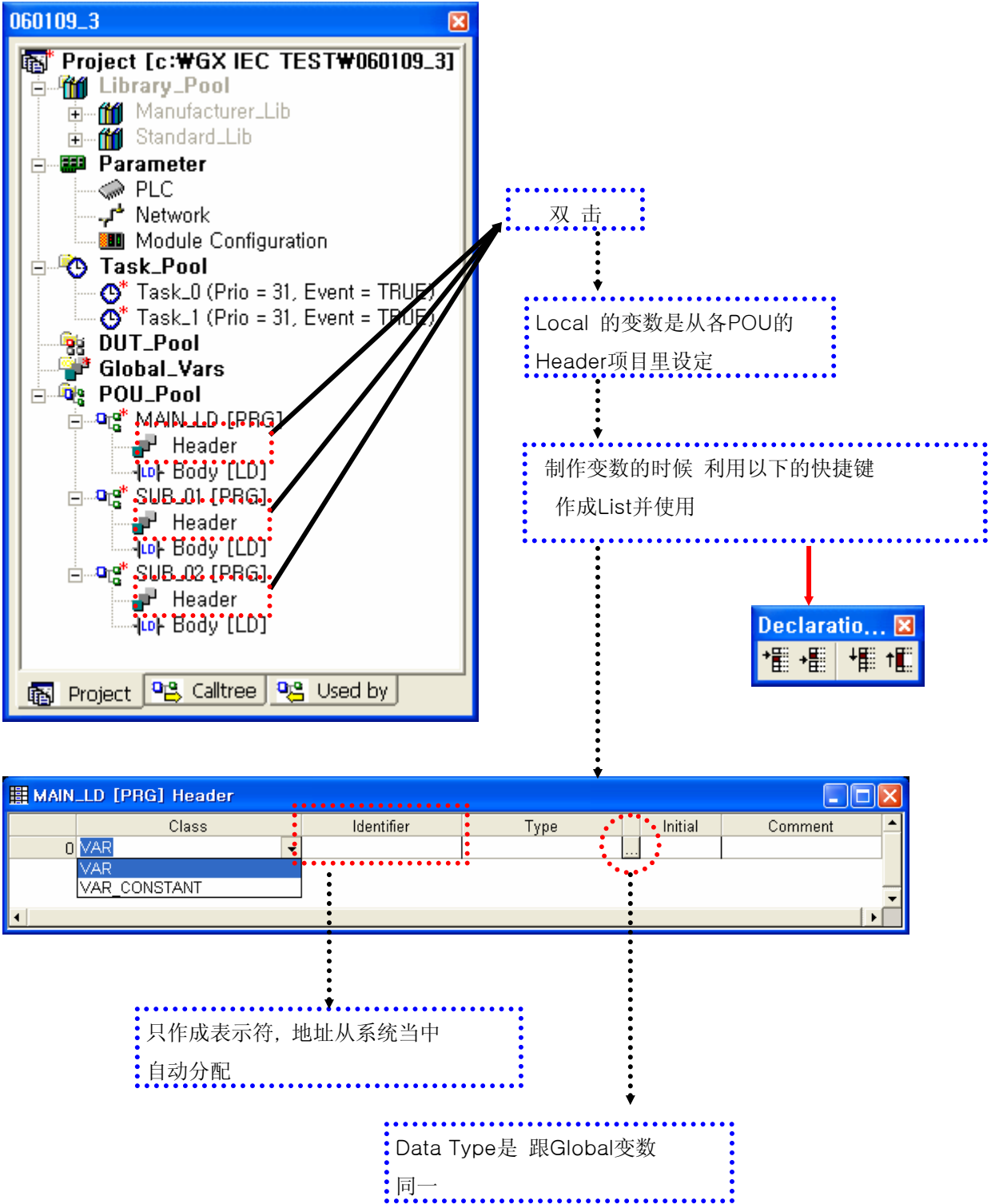
- Simple Types
- Data Unit Types
- Function Blocks

OK Cancel Help



8.2.2 Local Variables (局部变数) 设定

Fig8-5: Local 变数 登录 顺序





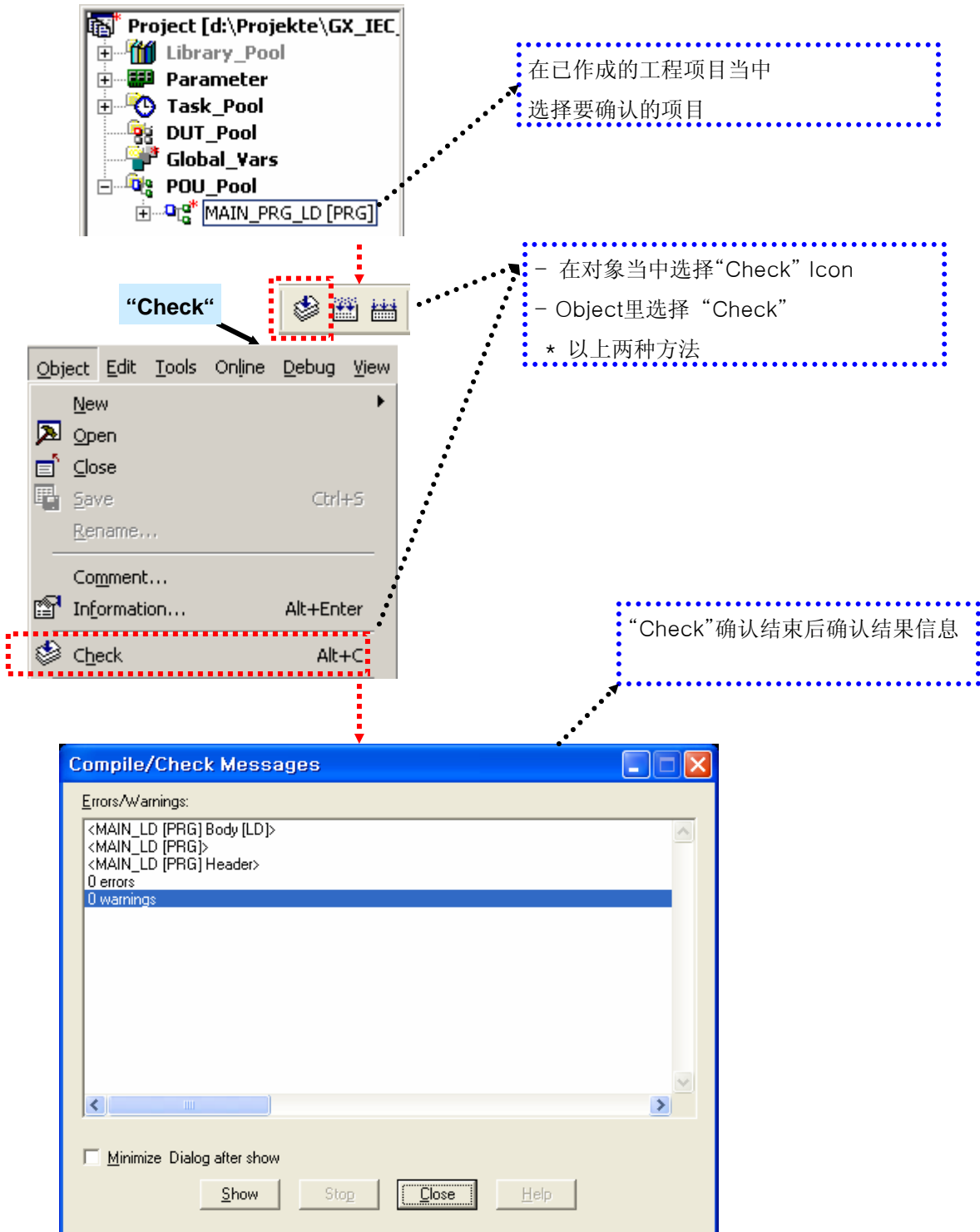
8.5 Check & Compiling

8.5.1 PLC 程序的 Check

: 检查程序构成的错误.

Check任意几次都可以.

Fig8-6: Check 顺序

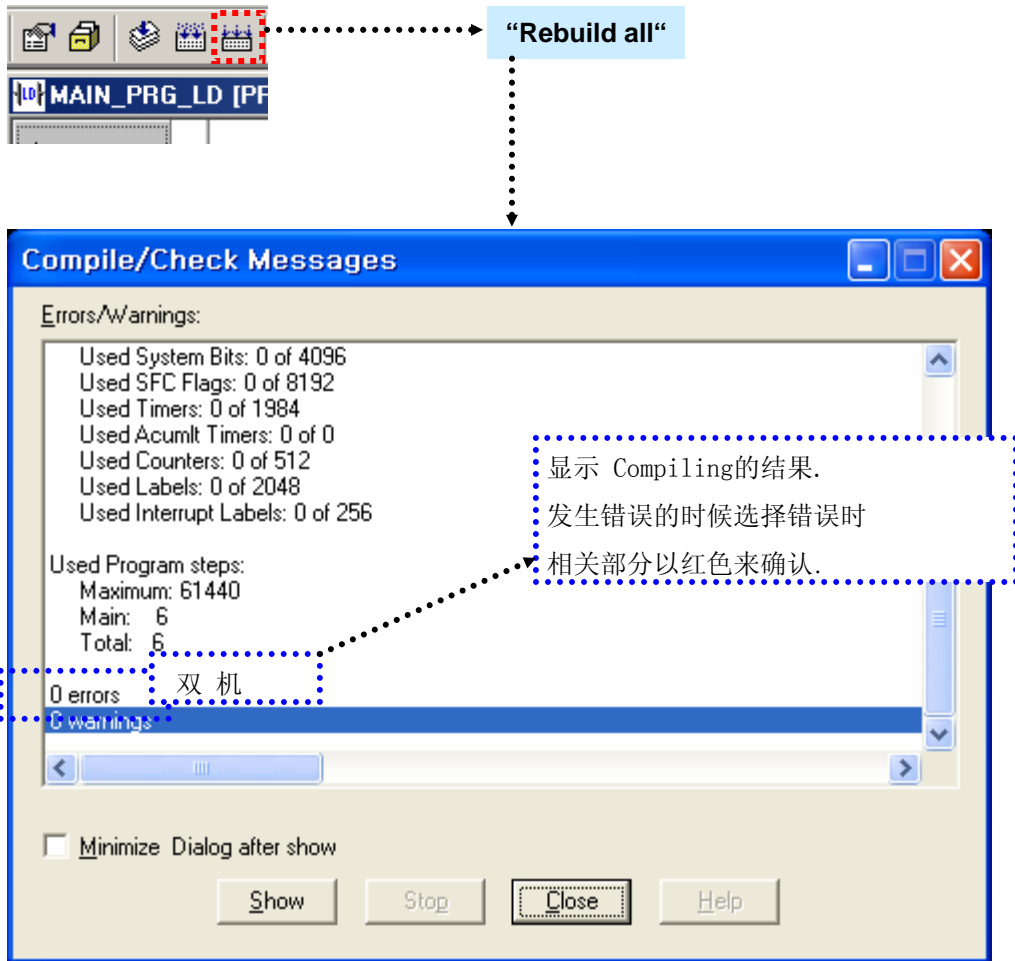




8.5.2 工程 Compiling

：把作成的工程在 PLC CPU里变换成能够实行的Code 的作业。

Fig8-7: Compiling 顺序



参考

- Compiling是只对Task & POU 进行Compiling .
- POU是只对登录在Task的POU进行Compiling.



9. GX IEC Developer的使用 - (POU 程序 - LD 作成)

** 基本作成工具的说明 **

Fig9-1: 程序作成画面

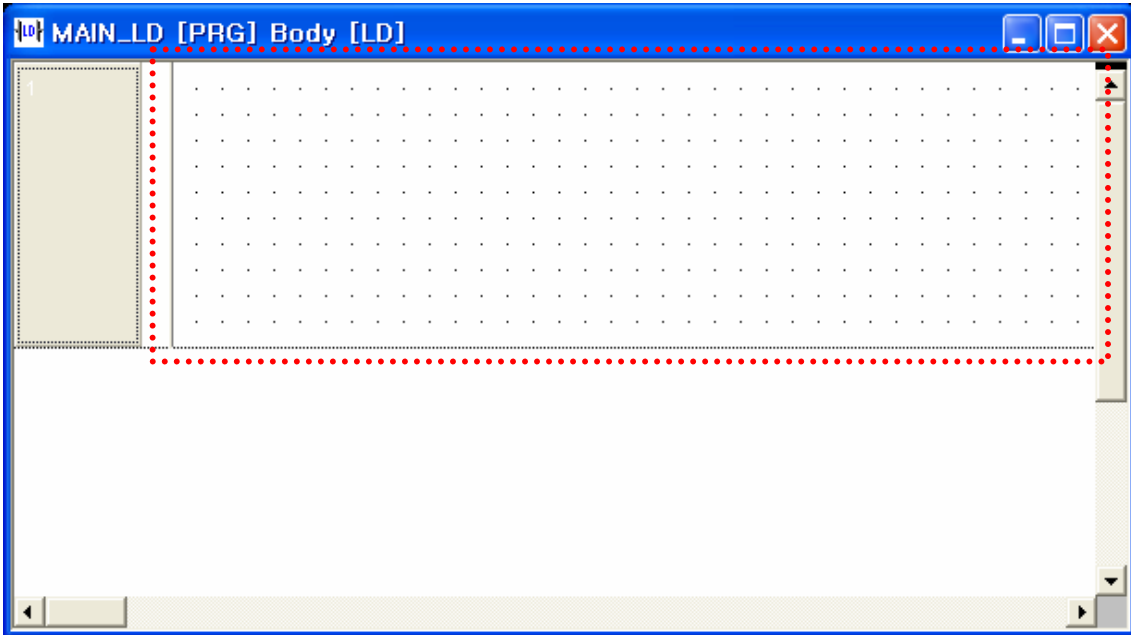





Fig9-2: LD 制作工具 (点击程序制作画面就被激活)



 : 追加新网络时使用

 : 选择, 画线等工具

 : 输入触点, 输出线圈, 功能等使用



9.1 利用MITSUBISHI地址的LD制作

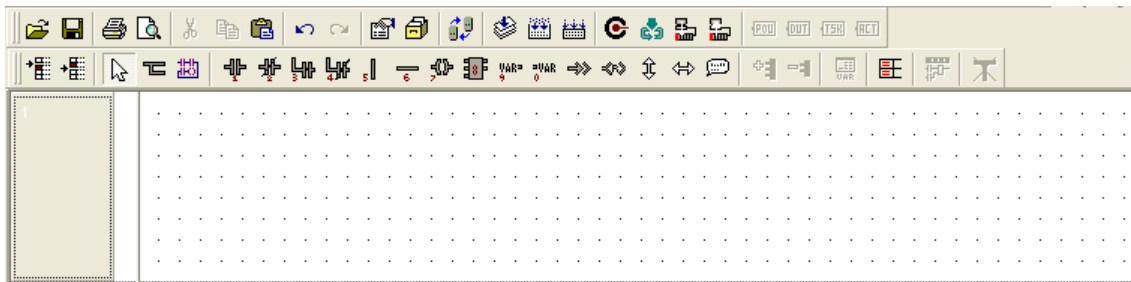
☞ 动作

- Switch 0 变为 ON时 LED 0 就变为 ON.
- Switch 0 变为 OFF时 LED 0 保持 ON状态
- Switch 5 变为 ON时 LED 0 就变为 OFF.

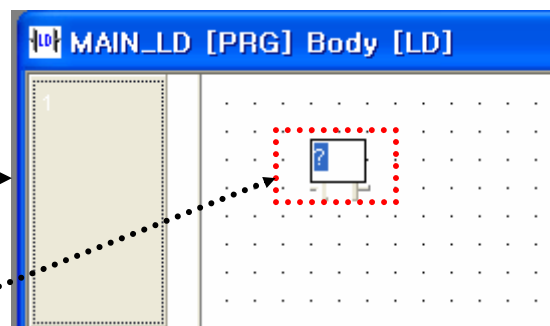
☞ 使用的软元件：

Switch 0:	输入 X0
Switch 5:	输入 X5
LED 0:	输出 Y20

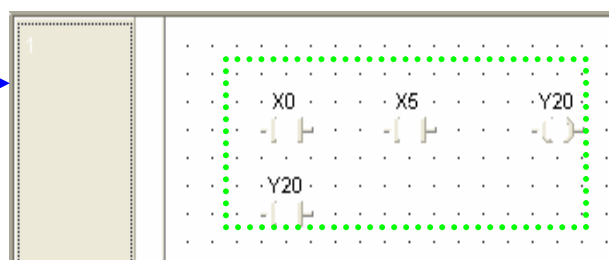
☞ 触点符号的使用及软元件的输入



1. 选择输入触点符号后
(点击鼠标左键)
- 点击后点的形态
2. 在程序编辑画面里选择要
放置的位置
(点击鼠标左键)



直接输入 三菱地址





☞ b触点选择

B触点是把符号双击
(选择的符号变颜色)

Signal Configuration

X5

Normal Negation

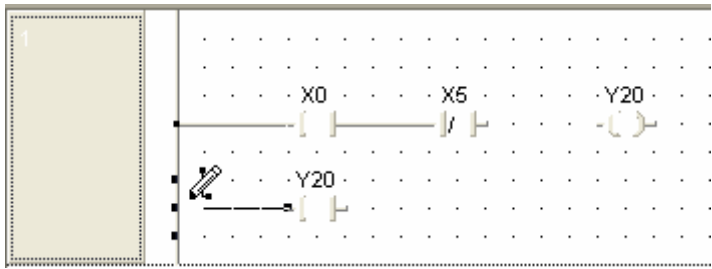
Set Reset

OK Cancel

B触点 选择

☞ 画线方法

选择画线符号工具后把每个触点连接就可以



*** 点击鼠标右键

Interconnect Mode Ctrl+T

Guided Editing Ctrl+G

Auto Connect

Contact

Coil

VAR Input Variable

VAR Output Variable

OP/FUN/FB

基本的符号工具都显示

点击画线功能时
选择自动连接模式

☞ Check & Compiling

: 所有程序制作结束后用本文 [8.5]的方法来 Check 及 Compiling内容



9.2 根据 Global Variable的程序作成

** 基本动作内容及使用软元件与 [9.1]的内容同一

☞ 变数设定

: 首先定义所使用软元件的标示符.

使用 输入 X0 = START_0 的标示符

使用 输入 X5 = STOP_0 的标示符

使用 输出 Y20 = LAMP_0 的标示符

☞ 变数作成

Class	
0	VAR GLOBAL
	VAR GLOBAL
	VAR GLOBAL CONSTANT

Identifier
START_

MIT-Addr.	IEC-Addr.
X0	%IX0

Type
BOOL

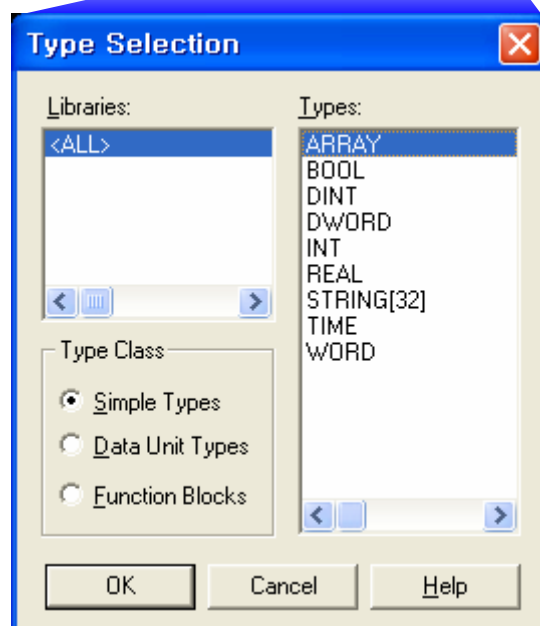
选择变数是Data类型

还是常数类型

变数的标示符 即,
写入变数名.

变数所拥有的地址输入

选择变数的Data类型



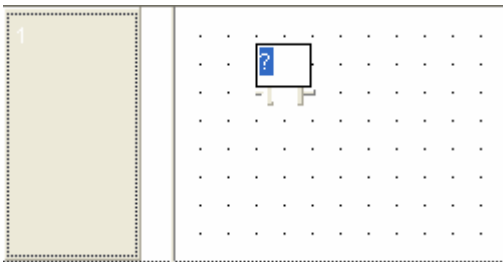


☞ 变数作成完了

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	START_0	X0	%IX0	BOOL	FALSE
1	VAR_GLOBAL	STOP_0	X5	%IX5	BOOL	FALSE
2	VAR_GLOBAL	LAMP_0	Y20	%QX32	BOOL	FALSE

** 变数作成完成后先Check后再关闭.

☞ 程序当中的适用



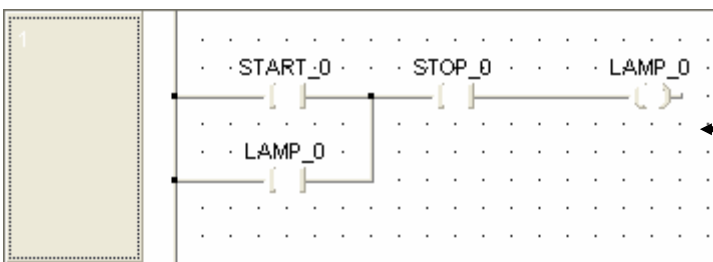
把触点选择在适当的位置后在“?”里点击鼠标右键激活变数选择窗。
根据所使用变数的分类来选择
指定要使用的变数名.

The dialog box 'Variable Selection' has the following settings:

- Scope: <ALL>
- Variables: LAMP_0 (selected)
- Type: BOOL
- Type Class: Simple Types
- IEC 61131-3: VAR_GLOBAL LAMP_0 AT %QX32 : BOOL := FALSE;

Annotations in the image point to the 'Scope' dropdown, the 'Variables' list, and the IEC 61131-3 declaration line.

根据用途选择变数



根据变数指定的
程序完成



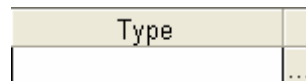
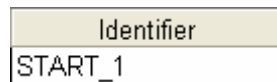
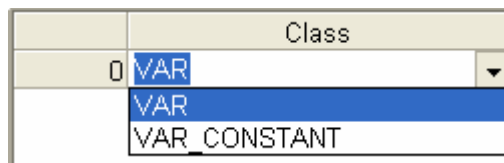
9.3 根据 Local Variable 的程序作成

- ** 基本的动作内容及使用软元件跟 [9.1]的内容同一
 在变数设定方法上跟 [9.1]的内容相似但还是有点差异,
 以下是关于差异点的说明

☞ 变数设定

- : 首先定义要使用的软元件的标示符
 使用 输入 START_1 的标示符
 使用 输入 STOP_1 的标示符
 使用 输出 LAMP_1 的标示符

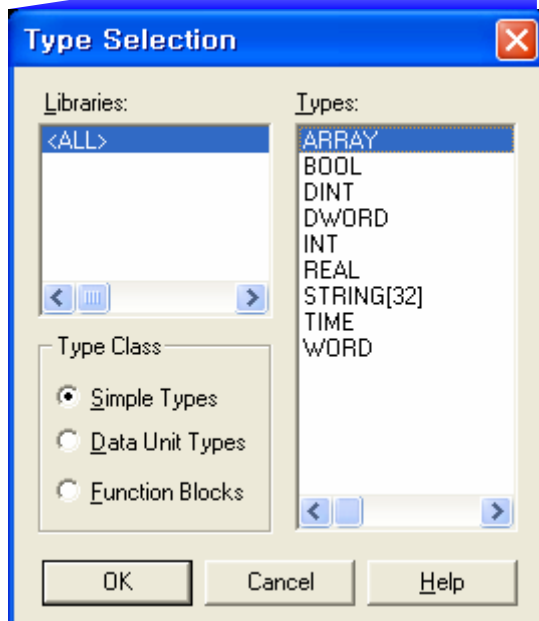
☞ 变数作成



选择变数是 Data类型
 还是常数类型

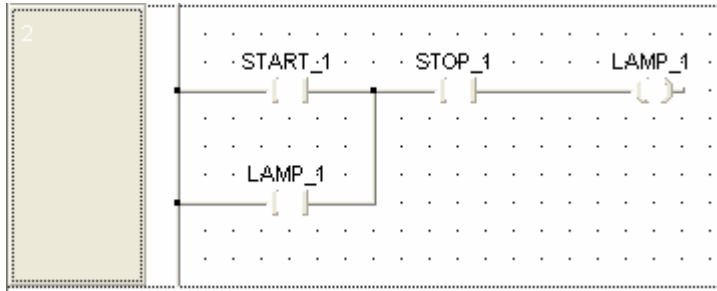
输入变数的标示符即,
 变数名

选择变数的Data 类型





在Local Variable 里的程序



根据变数指定的
程序完成

注意

根据Local Variable而程序作成时变数地址在系统里自动被分配。

即，根据所有触点及输出线圈内部软元件而分配
根据以上的话实际输入/输出都以全域变数而使用
内部继电器的话在系统里自动分配, 既而提高LD程序的效率.

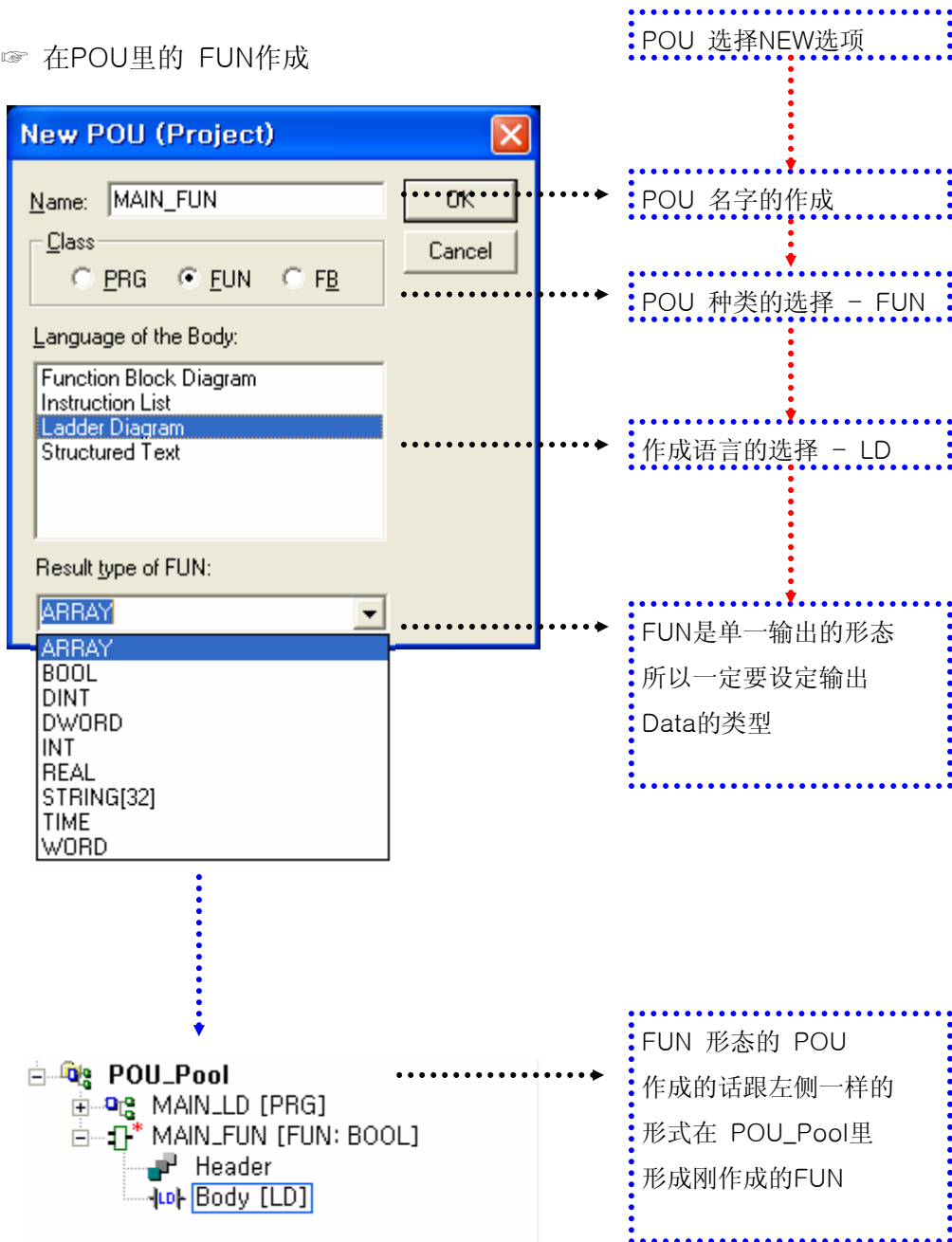


10. GX IEC Developer 使用 - (POU 程序 - FUN 作成)

: 关于根据FUN的 POU作成法及简单适用的说明.

10.1 FUN 作成

☞ 在POU里的 FUN作成



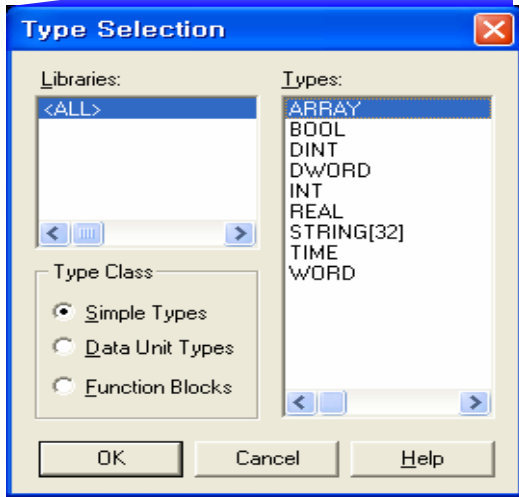


- ☞ 在 FUN里的变数作成
 - ** FUN的动作内容跟 [9.1]同一

Class	
0	VAR
	VAR
	VAR_INPUT
	VAR_CONSTANT

Identifier
START_2

Type
...



FUN的变数设定跟 Local Variable的设定相似

VAR：只有在FUN的内部才有可能动作跟FUN外部输入没有关系
 VAR_INPUT：把FUN的外部输入值在FUN的内部直接适用的要素
 VAR_CONSTANT：FUN的内部常数

变数的变数名(标示符)

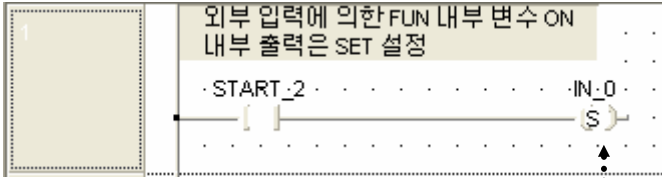
选择变数的Data类型

FUN 变数作成完了 - [Check]

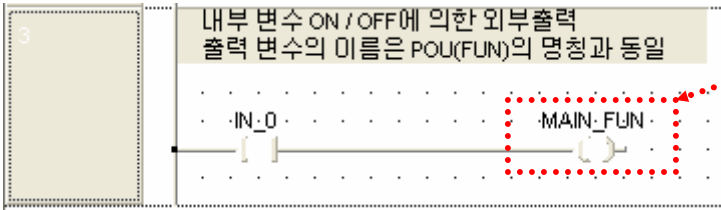
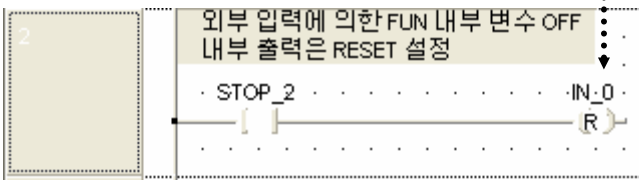
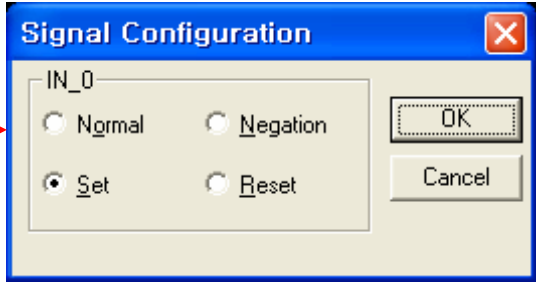
	Class	Identifier	Type	Initial
0	VAR_INPUT	START_2	BOOL	FALSE
1	VAR_INPUT	STOP_2	BOOL	FALSE
2	VAR	IN_0	BOOL	FALSE



☞ FUN的内部程序作成
 : 根据程序特性而区分网络

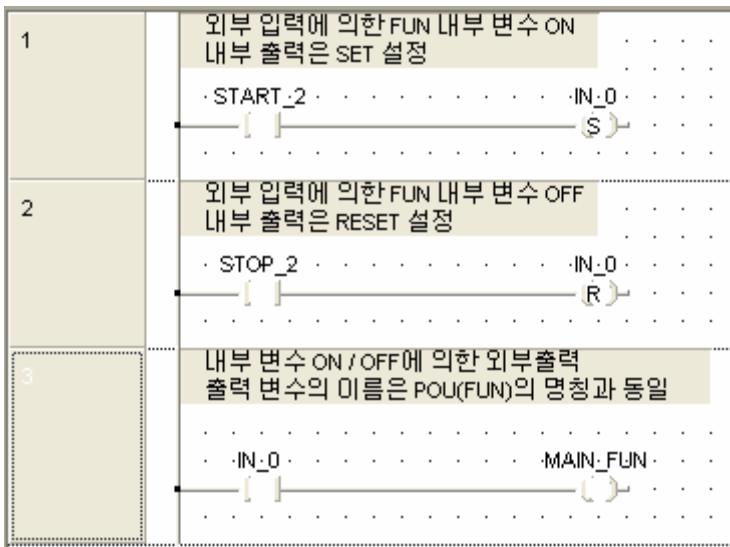


· 输出线圈的SET / RESET 设定
 · 线圈符号点击鼠标右键
 · 激活右侧窗口并选择



FUN的输出变数名 必须跟FUN(POU)的名一致.

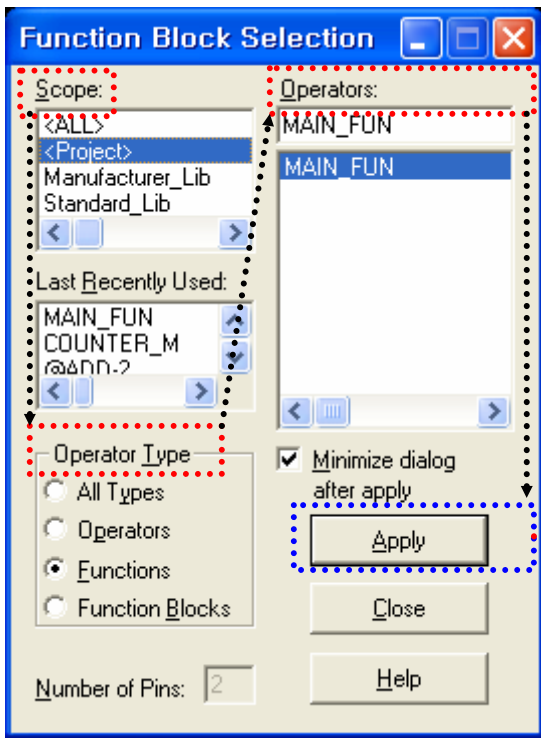
FUN (POU)의 内部程序





10.2 作成的FUN在PRG里的使用

☞ 使用方法



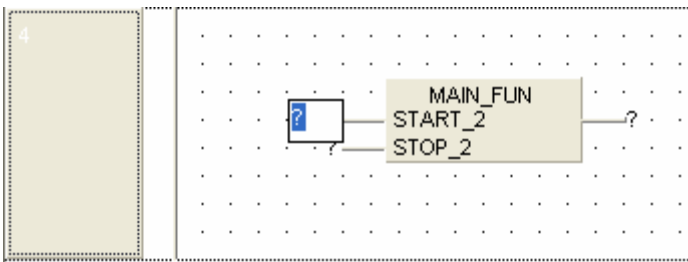
在LD编辑工具栏里

选择右侧图标

选择图标后激活右侧画面

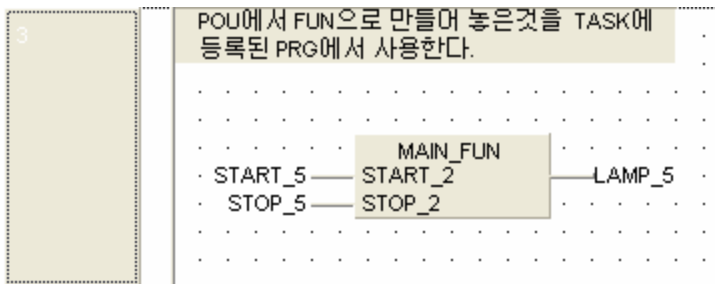
1. Scope : 选择搜索领域
2. Operator Type : 选择机能
3. Operators : 选择演算者
4. Apply : 适用选择的演算者

图标状态



选择 使用的FUN的输入及输出变数.

这时候的变数是由Global或Local
来设定

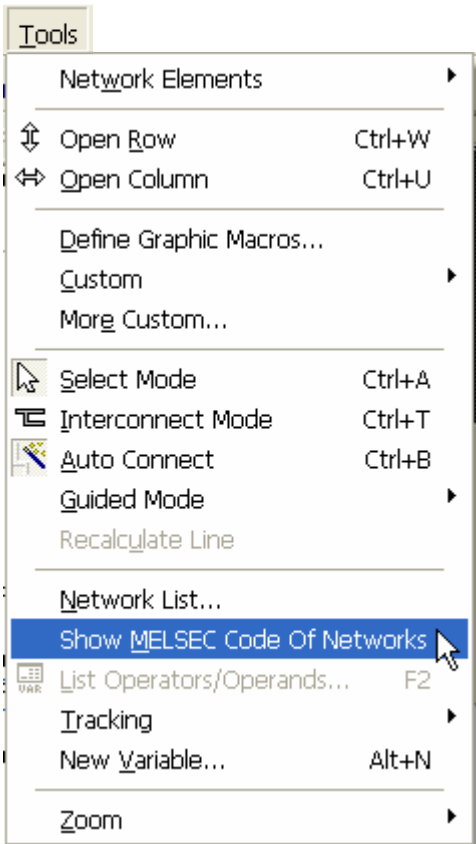
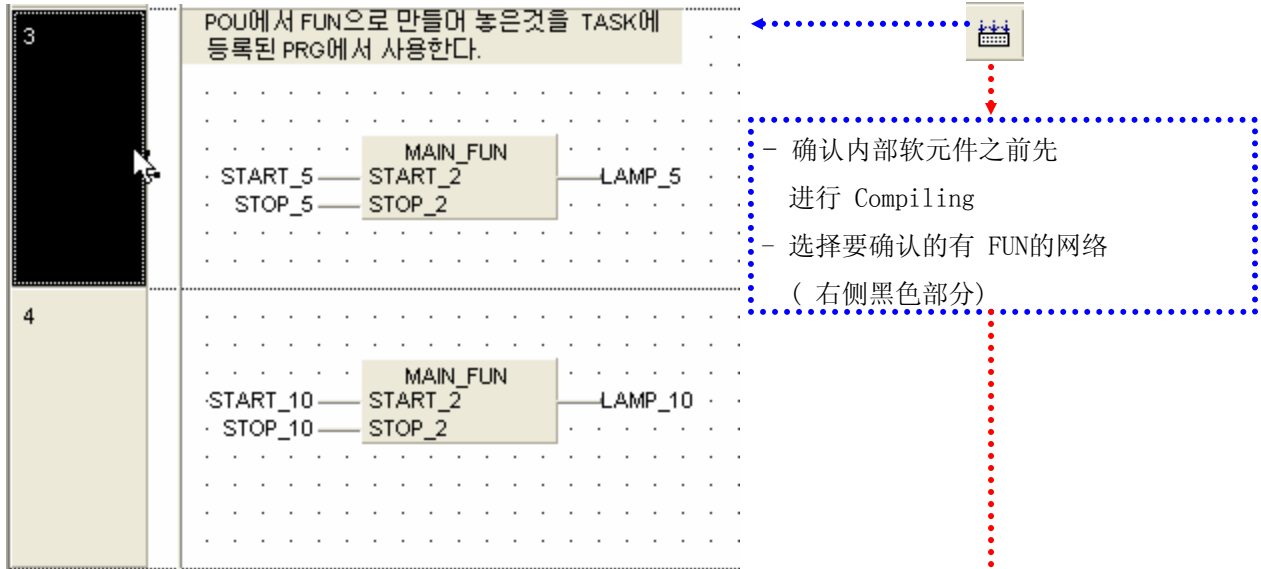


使用FUN的 PRG 作成



10.3 在FUN里的内部软元件使用

☞ 确认在FUN里自动分配的内部软元件的方法



- MELSEC 地址查看
[Tool] - [Show MELSEC CODE OF Networks 选择]



☞ 相同的 FUN在 POU里使用多个的时候内部变数的分配

** FUN自身CODE的确认

	Class	Identifier	Type	Initial
0	VAR_INPUT	START_2	BOOL	FALSE
1	VAR_INPUT	STOP_2	BOOL	FALSE
2	VAR	IN_0	BOOL	FALSE

所谓“MAIN_FUN”
FUN的 内部 CODE分配

```

(* MAIN_FUN [FUN: BOOL] *)
(* MELSEC code of NW #1 in MAIN_FUN *)
(* Code generated at 2006-01-11 오후 1:30:14, 8 steps, 8 steps for the whole object *)

P_2048:
LD      M4099      (* 27 *)
SET     M4102      (* 28 *)
LD      M4100      (* 29 *)
RST     M4102      (* 30 *)
LD      M4102      (* 31 *)
OUT     M4101      (* 32 *)
RET     M4101      (* 33 *)
RET     M4101      (* 34 *)
    
```

对于各个变数的CODE是
由系统自动分配.



** FUN适用在PRG的情况

MELSEC code of NW #3 in MAIN_LD

```
(* MAIN_LD [PRG] *)
(* MELSEC code of NW #3 in MAIN_LD *)
(* Code generated at 2006-01-11 오전 10:48:22, 9 steps, 26 steps for the whole object *)

LD      X8          (* 8 *)
OUT     M4099       (* 9 *)
LD      X9          (* 10 *)
OUT     M4100       (* 11 *)
LD      SM400       (* 12 *)
CALL    P_2048      (* 13 *)
LD      M4101       (* 15 *)
OUT     Y25         (* 16 *)
```

POU에서 FUN으로 만들어 놓은것을 TASK에 등록된 PRG에서 사용한다.

```
MAIN_FUN
· START_5 — START_2 — LAMP_5
· STOP_5 — STOP_2
```

- 使用多个相同的FUN的时候 FUN的 VAL_INPUT, VAL, FUB输出都有相同的 MELSEC CODE .

MELSEC code of NW #4 in MAIN_LD

```
(* MAIN_LD [PRG] *)
(* MELSEC code of NW #4 in MAIN_LD *)
(* Code generated at 2006-01-11 오전 10:48:22, 9 steps, 26 steps for the whole object *)

LD      X1          (* 17 *)
OUT     M4099       (* 18 *)
LD      X6          (* 19 *)
OUT     M4100       (* 20 *)
LD      SM400       (* 21 *)
CALL    P_2048      (* 22 *)
LD      M4101       (* 24 *)
OUT     Y21         (* 25 *)
```

POU에서 FUN으로 만들어 놓은것을 TASK에 등록된 PRG에서 사용한다.

```
MAIN_FUN
· START_10 — START_2 — LAMP_10
· STOP_10 — STOP_2
```

参考

- FUN是以各不同的特性而作成并使用..
- 使用多个相同的FUN的时候即使把FUN的输出用别的变数来使用，都有相同MELSEC CODE的输出。

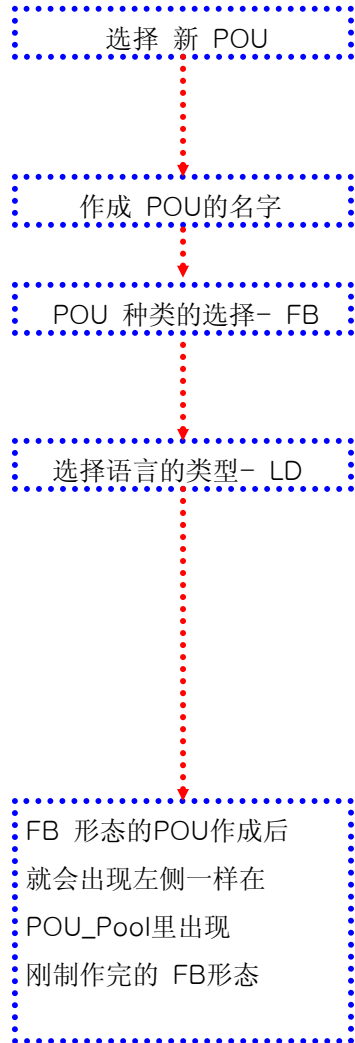
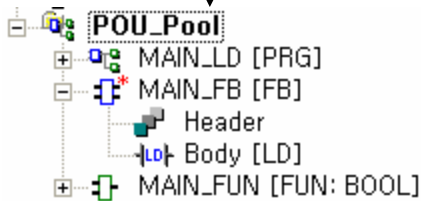
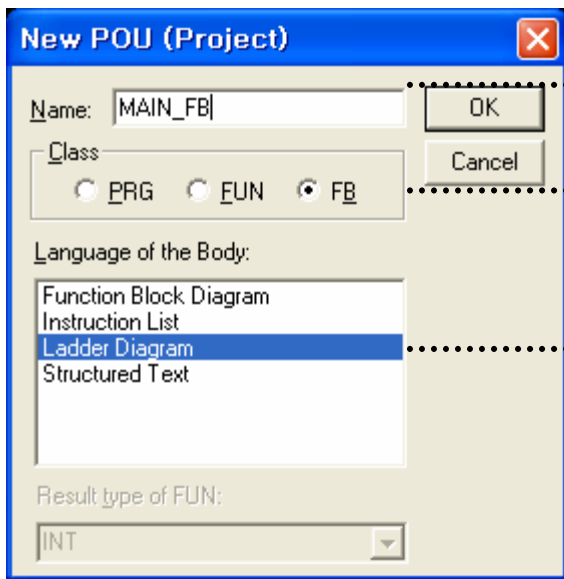


11. GX IEC Developer 使用 - (POU 程序 - FB 制作)

: 关于FB的POU制作方法及简单使用方法的说明.

11.1 FB 作成

☞ 在POU里制作 FB



参考

- 在第4章里已经说明过FUN与FB的不同点就是输出变数是否是一个或则是多个而区分，根据这种特点作成POU时会出现是否选择输出形态。这一点在变数指定里也会出现不同点



- ☞ 在FB里的变数作成
 - ** 在FB里的动作内容与 [9.1]的同一.

Class	
0	VAR
	VAR
	VAR_INPUT
	VAR_OUTPUT
	VAR_IN_OUT
	VAR_CONSTANT

Identifier	Type
START_2	
	...

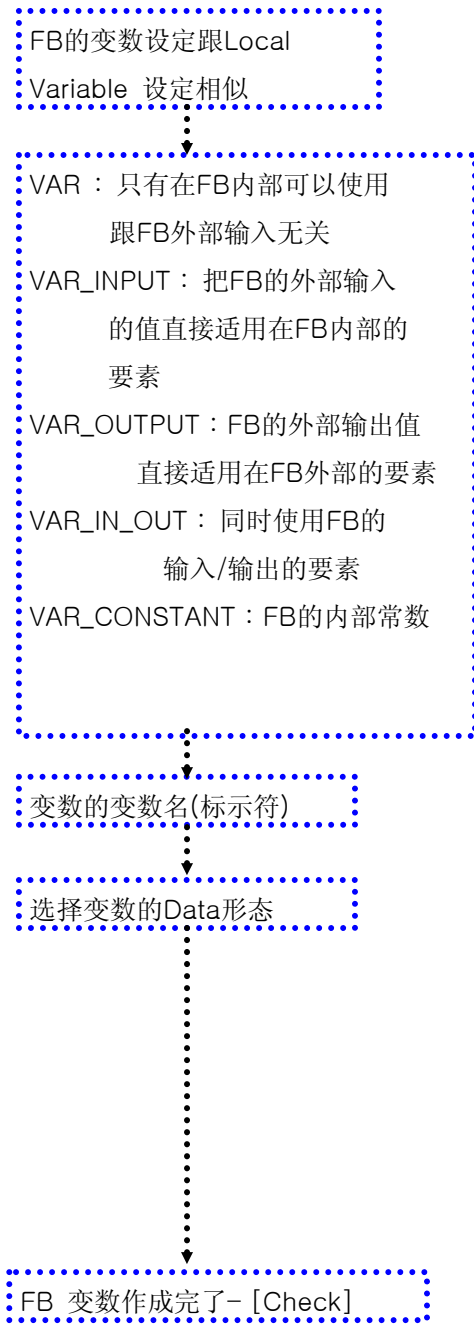
Type Selection

Libraries: <ALL>

Types: ARRAY, BOOL, DINT, DWORD, INT, REAL, STRING[32], TIME, WORD

Type Class: Simple Types, Data Unit Types, Function Blocks

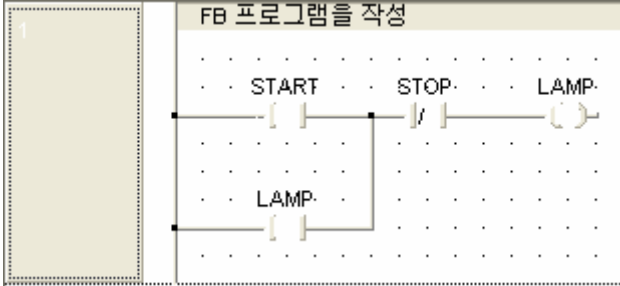
OK Cancel Help



	Class	Identifier	Type	Initial
0	VAR_INPUT	START	BOOL	FALSE
1	VAR_INPUT	STOP	BOOL	FALSE
2	VAR_OUTPUT	LAMP	BOOL	FALSE

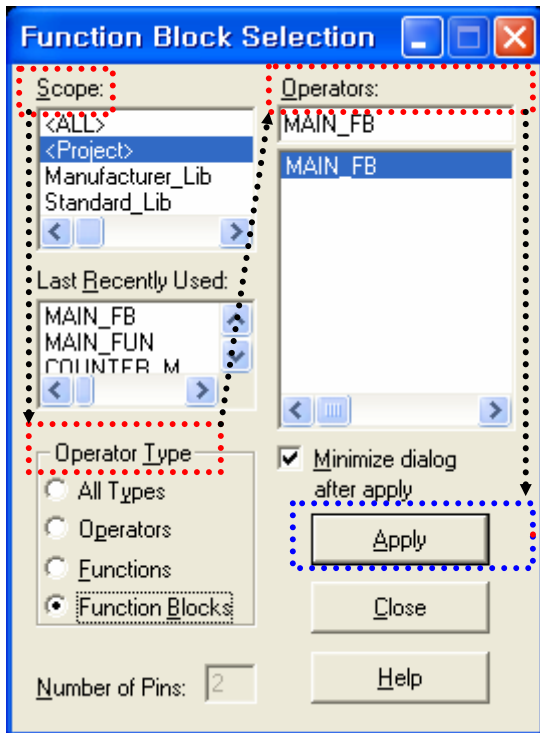


- ☞ FB的内部程序作成
 - ：把第9章里制作的程序同一作成后FB化



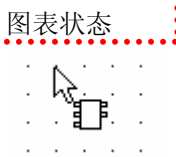
11.2 作成的FB在PRG当中的使用

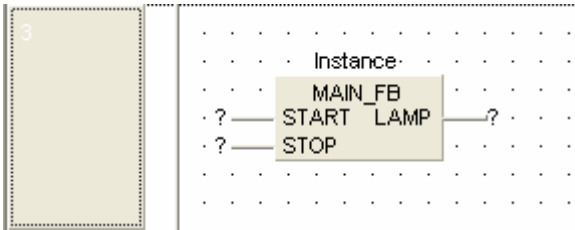
- ☞ 使用方法



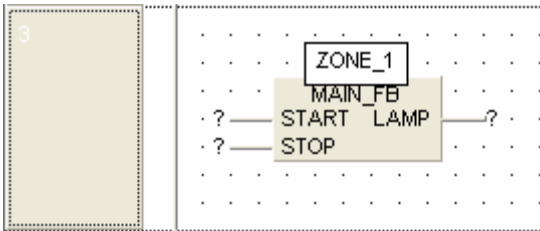
在梯形图编辑栏里
选择右边的图表

- 选择图表以后激活右侧画面
1. Scope : 选择搜索领域
 2. Operator Type : 选择技能
 3. Operators : 选择操作者
 4. Apply : 选择操作者的适用

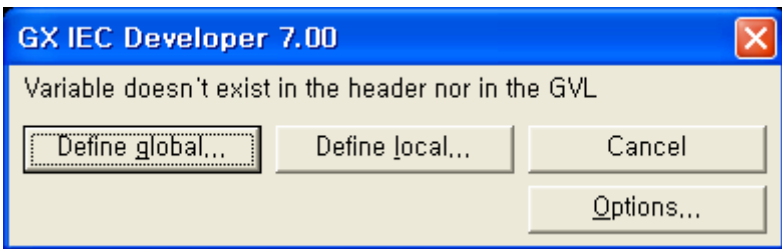




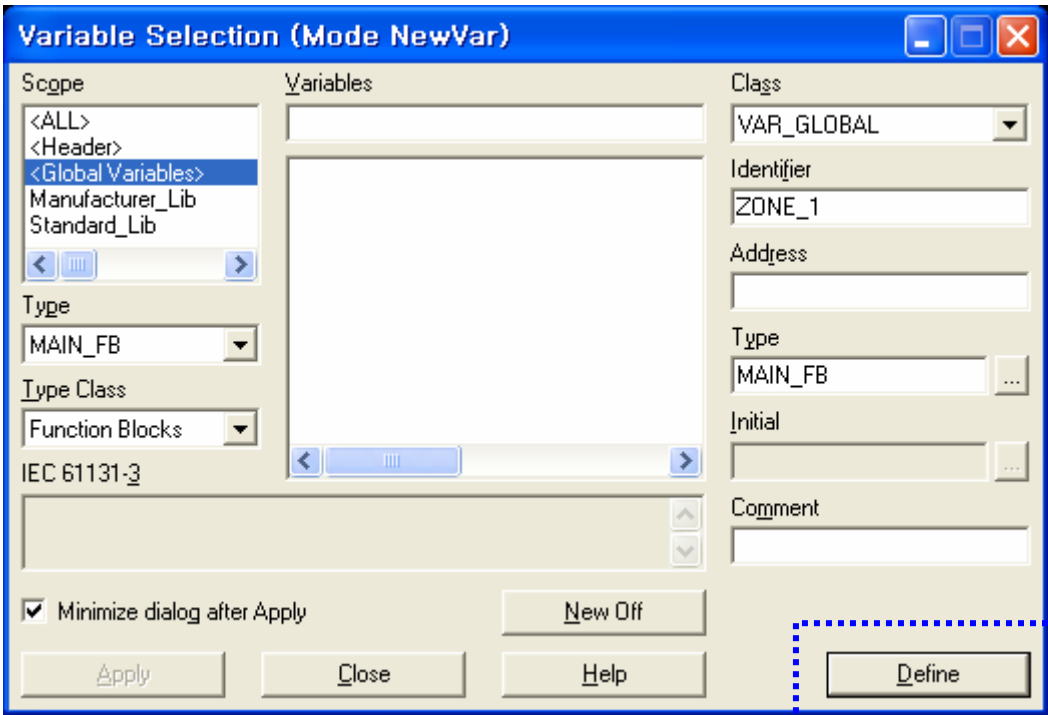
- 引出作成的FB
- 点击Instance的标题后要在PRG里作成使用FB的变数名



如果变数没有被登陆的话出现“变数名不存在”提示，在这里可以选择要把刚刚作成的变数用在哪儿



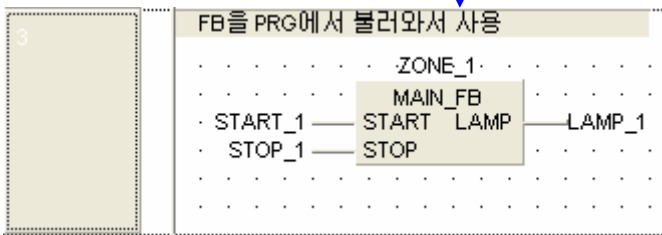
选择要使用变数形态的话激活登录窗口



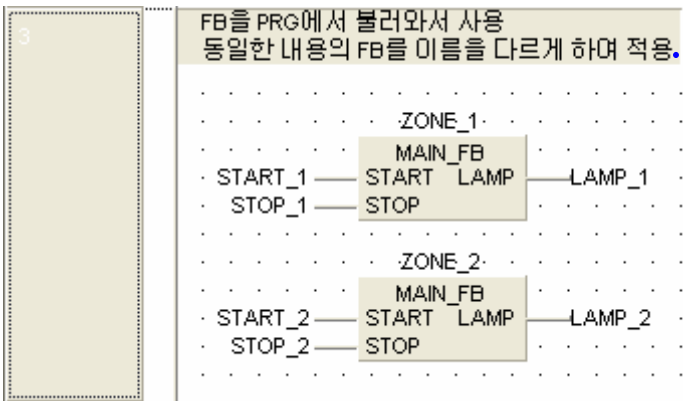


Global Variable 列表里作成以下形式登录
 - 作成的FB以 “ MAIN_FB ” 的 Data形态来被登录.

Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
VAR_GLOBAL	ZONE_1			MAIN_FB	...



被“ZONE_1”定义的FB的输入输出的变数指定并完成.



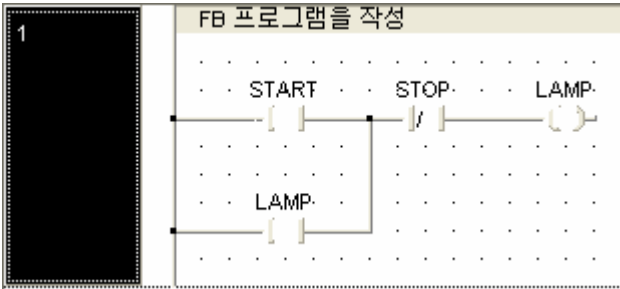
把相同的FB作成Data Type的变数. 来作成并使用

参考 - 使用作成的FB的话在对一定的动作Patten里可以以单位化的模块来利用



11.3 在FB里的内部软元件使用

- ☞ 确认在FB里使用的内部软元件（FB的另一个变数使用）
 - ：内部软元件的确认(MELSEC CODE)跟 [10.3]的内容一样说明与FUN的不同点.



要确认FB本身程序自动分配的 MELSEC CODE

FB Instance Selection

Global_Vars: ZONE_1
Global_Vars: ZONE_2

OK

在PRG里把相同的FB以只改变变数名并使用的時候在各个变数当中使用的同一的FB的 MELSEC CODE就会不同

MELSEC code of NW #1. in Global_Vars: ZONE_1

(* MAIN_FB [FB] *)
(* MELSEC code of NW #1 in Global_Vars: ZONE_1 *)
(* Code generated at 2006-01-11 오후 5:16:06, 6 steps, 6 steps for the whole object *)

P_2048:		(* 27 *)
LD	M4096	(* 28 *)
OR	M4098	(* 29 *)
ANI	M4097	(* 30 *)
OUT	M4098	(* 31 *)
RET		(* 32 *)

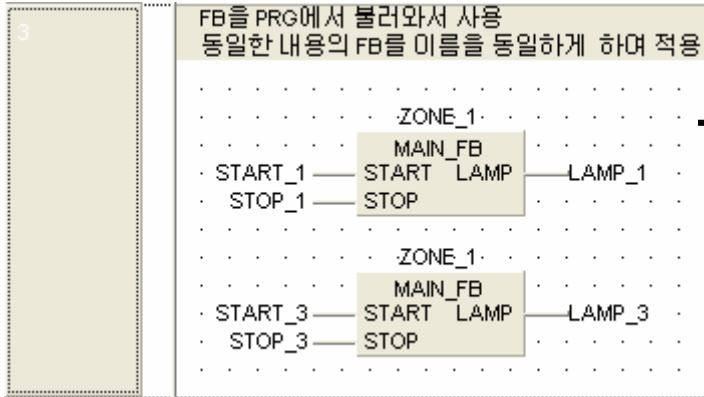
MELSEC code of NW #1. in Global_Vars: ZONE_2

(* MAIN_FB [FB] *)
(* MELSEC code of NW #1 in Global_Vars: ZONE_2 *)
(* Code generated at 2006-01-11 오후 5:16:06, 6 steps, 6 steps for the whole object *)

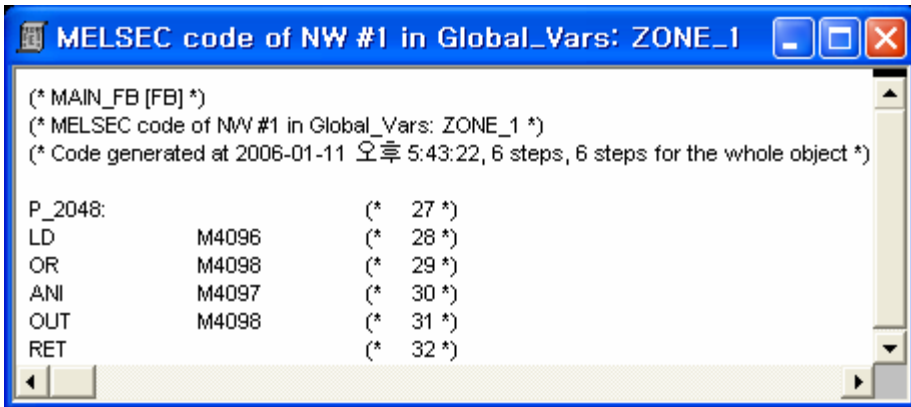
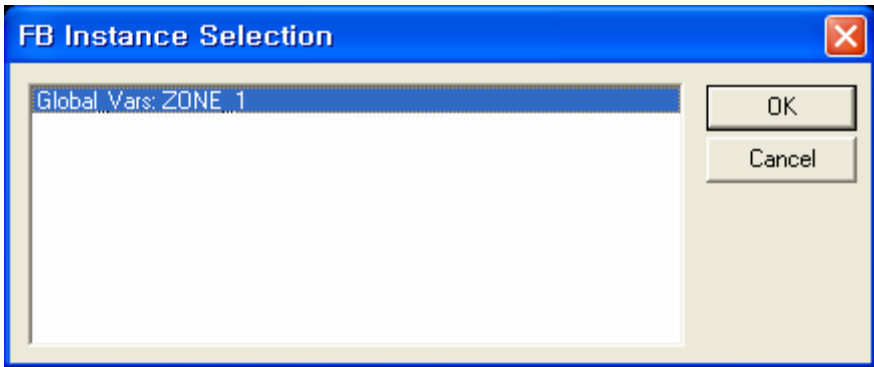
P_2049:		(* 33 *)
LD	M4099	(* 34 *)
OR	M4101	(* 35 *)
ANI	M4100	(* 36 *)
OUT	M4101	(* 37 *)
RET		(* 38 *)



☞ 确认在FB里使用的内部软元件（使用同一变数的FB）



在PRG里使用相同形态的FB变数时
FB里的MELSEC CODE会同一适用
即，各不一样的输入可在 FUN内部里
可以发生相同的输出并FUN的各不
一样的外部输出可以同时进行输出。





11.4 利用MACRO的FB制作

：利用MACRO功能可以减少在FB里的MELSEC CODE的使用。

☞ MACRO是追加FB功能特性的一种

：FB的作成发基本上跟[11.1]里的相同。

选择使用MACRO功能的FB
以后点击鼠标右键激活在
Menu里选择 Properties

选择MACRO功能使用

Function Information

Name: MAIN_FB_MACRO

Size: 44 Bytes

Use Macrocode

Use MCL/ MCH

Use with EN/END

Type: FB

Language: Ladder Diagram

Last Change: 2006-01-11 오후 6:04:46

Security Level: 0

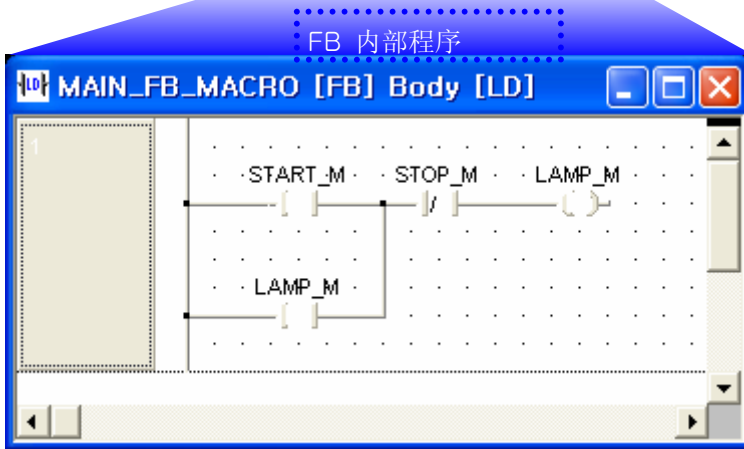
Allow Read Access for lower Levels

- 参 考**
- FB的作成方法基本上跟[11.2]的内容相同在这里就省略。
 - 注意[11.4]里的 MELSEC CODE分配。

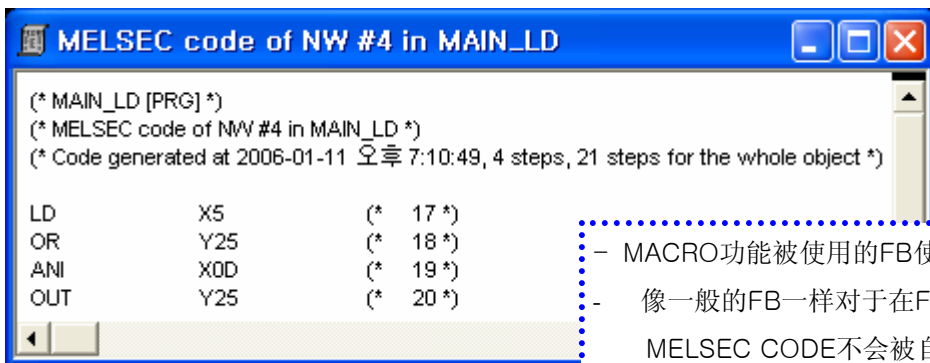


11.4 MACRO被使用的FB里的内部软元件的使用

☞ 确认根据MACRO的内部软元件的使用



基本上不管是有什么样的变数属性，被指定为MACRO的FB在FB 程序里面不能确认 MELSEC CODE

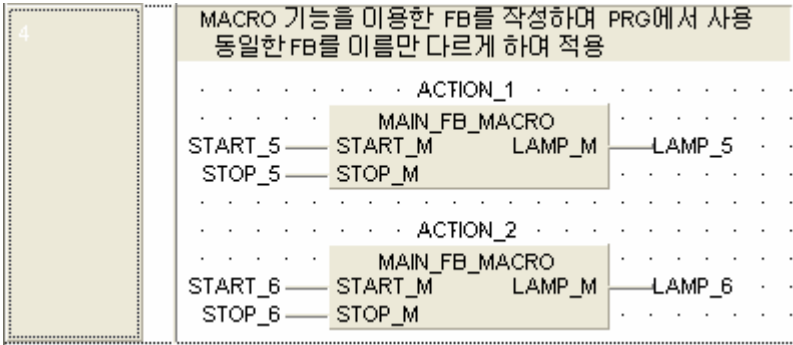


- MACRO功能被使用的FB使用的PRG里才能确认.
- 像一般的FB一样对于在FB内部使用的变数 MELSEC CODE不会被自动分配

即，跟一般的FB相比对于“VAR”除外的变数属性 MELSEC CODE不会被自动分配。
(VAR是 跟FB的输入/输出无关的只在FB内部使用的要素而可以自动分配.)



MACRO化的同一 FB的使用



不同于一般的FB即使使用
相同的FB可因 MACRO功能的
特性而MELSEC CODE
不会自动被分配. .

MELSEC code of NW #4 in MAIN_LD

```

(* MAIN_LD [PRG] *)
(* MELSEC code of NW #4 in MAIN_LD *)
(* Code generated at 2006-01-11 오후 7:22:14, 8 steps, 25 steps for the whole object *)

LD      X5      (* 17 *)
OR      Y25     (* 18 *)
ANI     X0D     (* 19 *)
OUT     Y25     (* 20 *)
LD      X6      (* 21 *)
OR      Y26     (* 22 *)
ANI     X0E     (* 23 *)
OUT     Y26     (* 24 *)
    
```



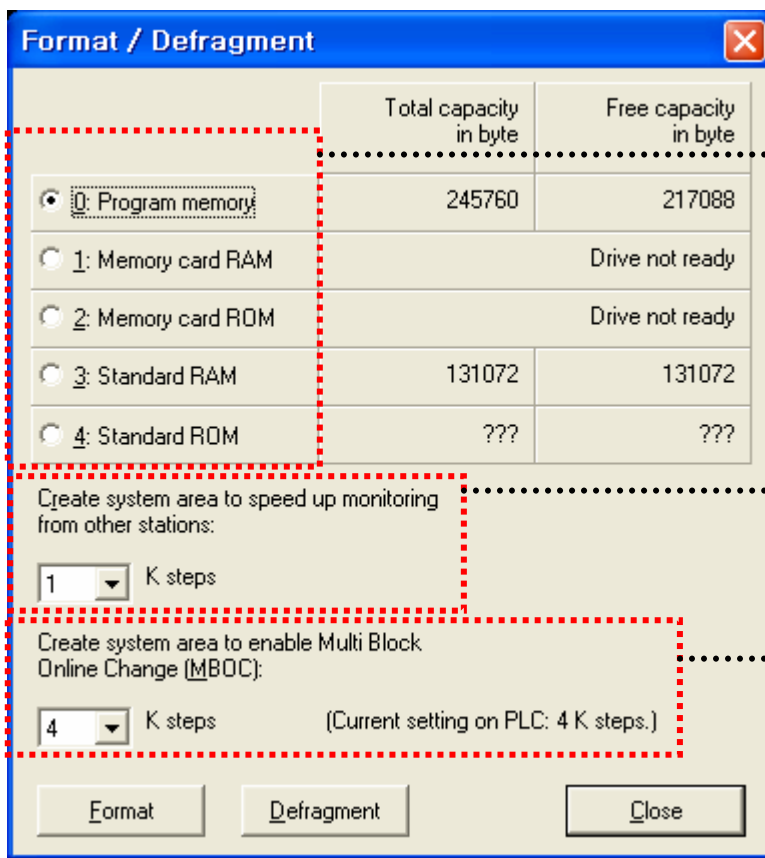
12. GX IEC Developer 使用 - (程序传送)

** 关于作成的Project 读取,写入,检查及其他Project传送的内容确认.

12.1 Format

: 对于 PLC Format 功能简单说明.

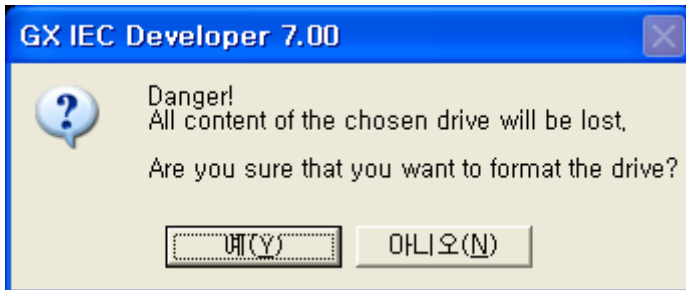
⇒ 选择路径 : [Online] - [Format Drive]



- 要选择Format的PLC存储器后再进行Format

- Online Change 即, 在线写入用途的系统领域设定的文件不存在而且会占据容量.

- MBOC Online Change 即, 选择MBOC在线写入用途的领域设定最多64block可能会生成单独文件



- 选择[Format]后再次确认



⇒ 关于 Project 传送的基本 Menu

Project 传送

- 在Menu里 [Project] 的选择
- [Transfer] 的选择
- 顺序说明各Tab的内容.

Transfer

- Upload from PLC...
- Download to PLC... **Ctrl+Alt+W**
- Verify
- Download Symbolic Informations
- Copy Program Memory to Flash ROM...

12.2 Download Project (PLC 写入)

: 把作成的程序传送到PLC.

⇒ 进行下载

Transfer to PLC

The current project will be downloaded to the PLC using the actual Ports & Project Transfer Setup.

Transfer Setup Ports:

Transfer Setup Project:

Download Project

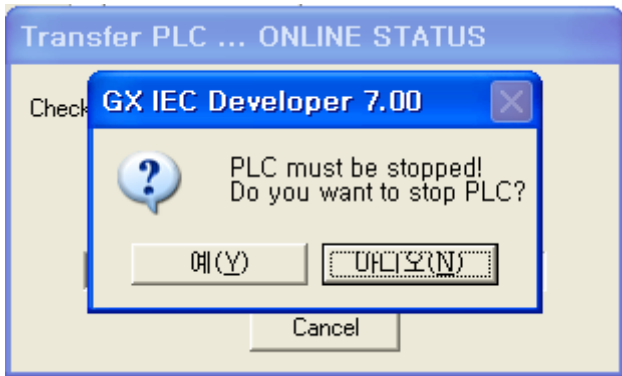
- 点击右侧图表
- PLC传送Setup画面被激活
- 这里的设定跟[8. 1]里说明的内容相同
- 跟初期设定内容不同的话在这个画面里需要再设定的话可以进行再设定.
- 设定结束后点击[OK]



Password (OK选择后)

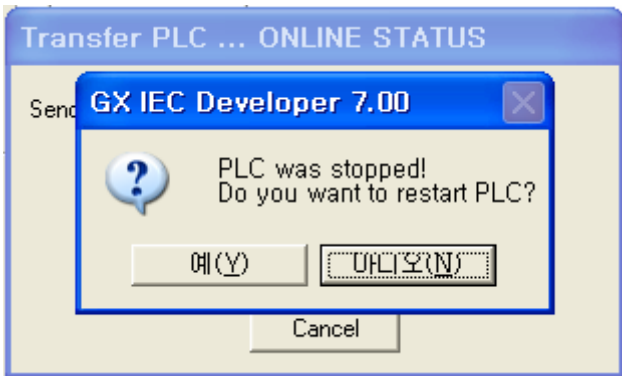
对于作成的程序Symbolic情报是分析程序的重要DATA

- 对于该Symbolic情报施行保安功能时可以设定Password
- 设定必要的话输入或不必要的话点击[OK]



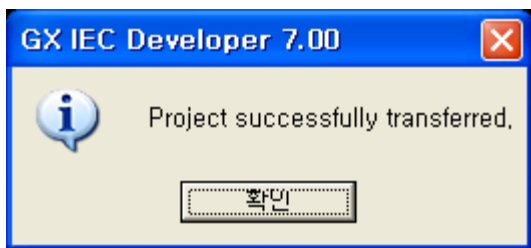
Project 传送中

- Project 传送中PLC要停止选择 [YES].



Project 传送结束

- Project 传送结束后PLC进行RUN. 选择[YES]



Project 传送完了信息

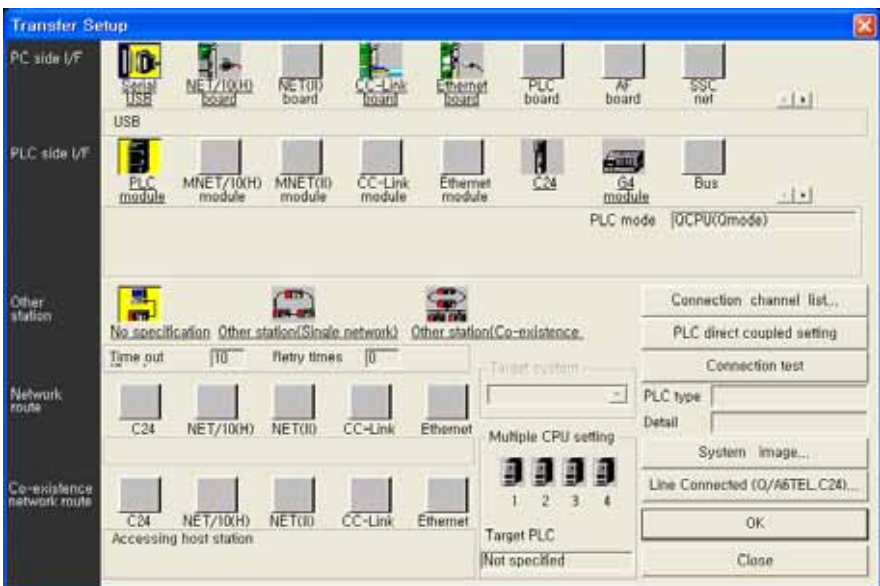
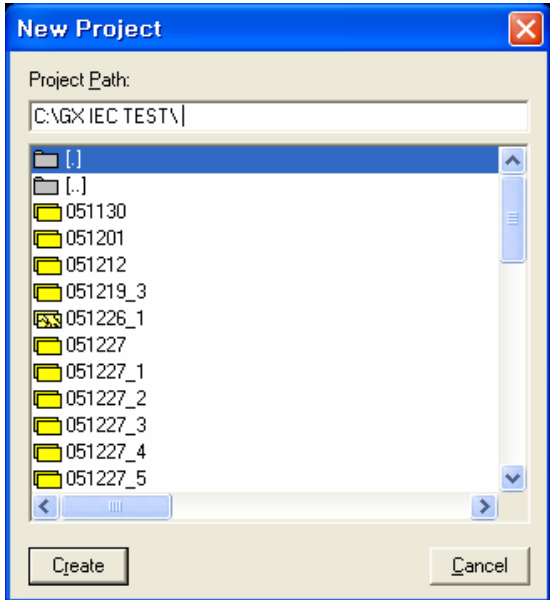
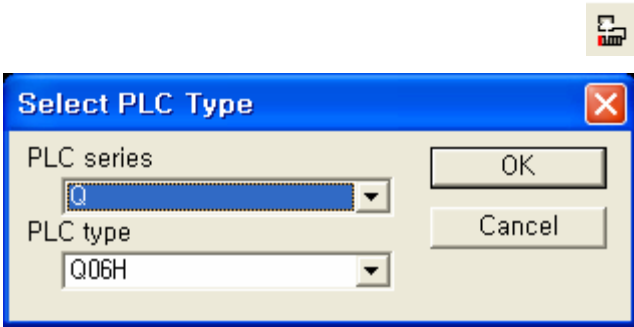
- Project 传送正常结束的话出现右侧的信息.

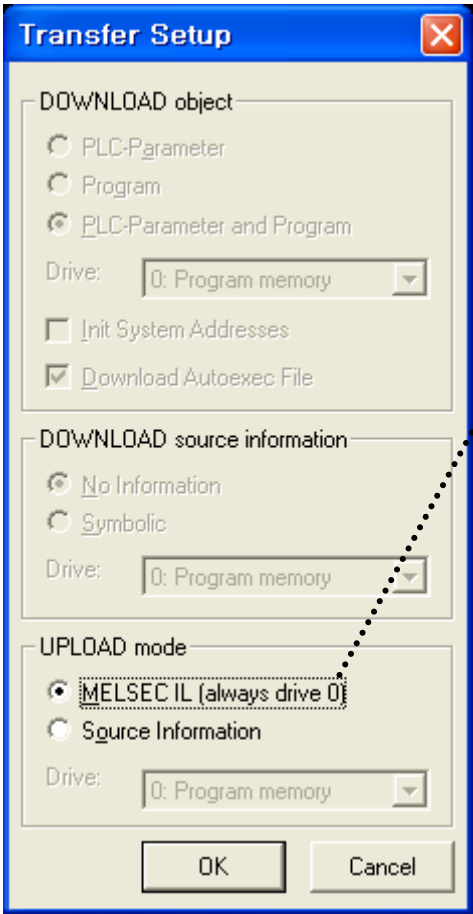


12.3 Upload Project (PLC 读取)

: 把PLC的程序传送到PC里.

12.3.1 软件初期状态下的进行读取

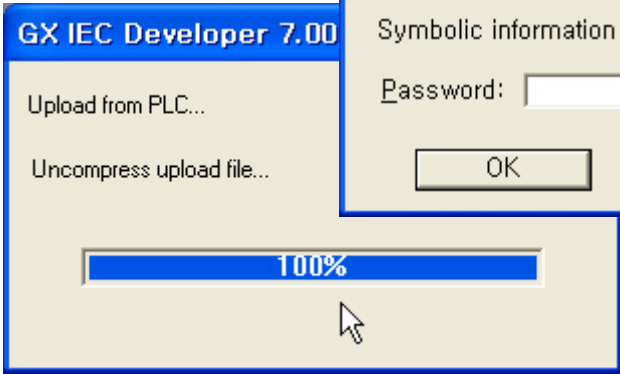
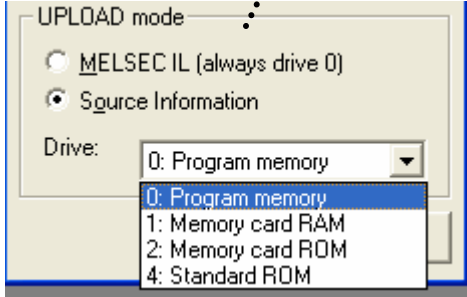




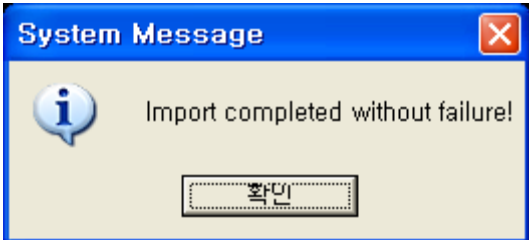
- 为了Upload的Setup画面激活
- 在UPLOAD Mode里有两种选项.选者后[OK]

1. MELSEC IL的时候Upload时
以MELSEC IL的形态来进行Upload.
(Symbolic Data 除外)

2. Source Information的时候
指定要驱动要读的路径
(Symbolic Data 包含)



- 对于基本参数Data的时候可以读取
对于Symbolic有Password的时候
要输入Password.



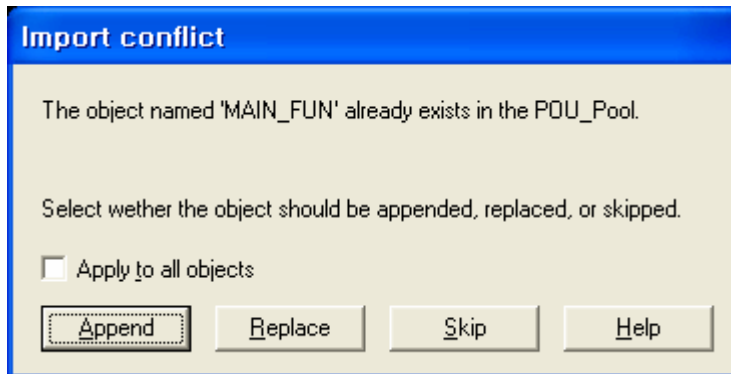
- Project Upload正常结束的话
出现右侧的信息.



12.3.2 程序被打开的状态下进行Upload

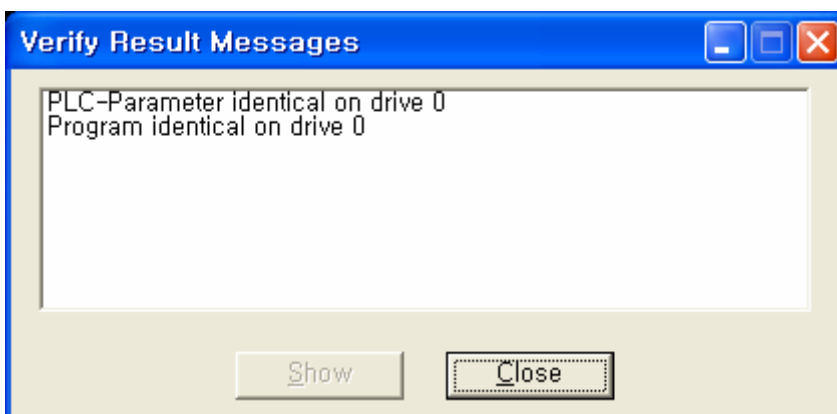
: 程序打开状态下的Upload的话从上面的[12.3.1]的
Port 设定画面开始

⇒ 不同的话就是程序内部已存在内容所以对于这些内容进行
追加, 代替及Skip的选择.

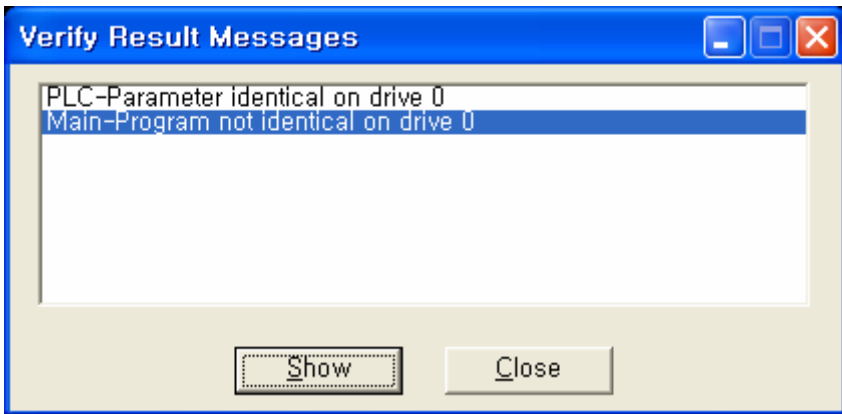


12.4 Verify (校验)

: [Project] 的 [Transfer] Menu里选择[Verify]

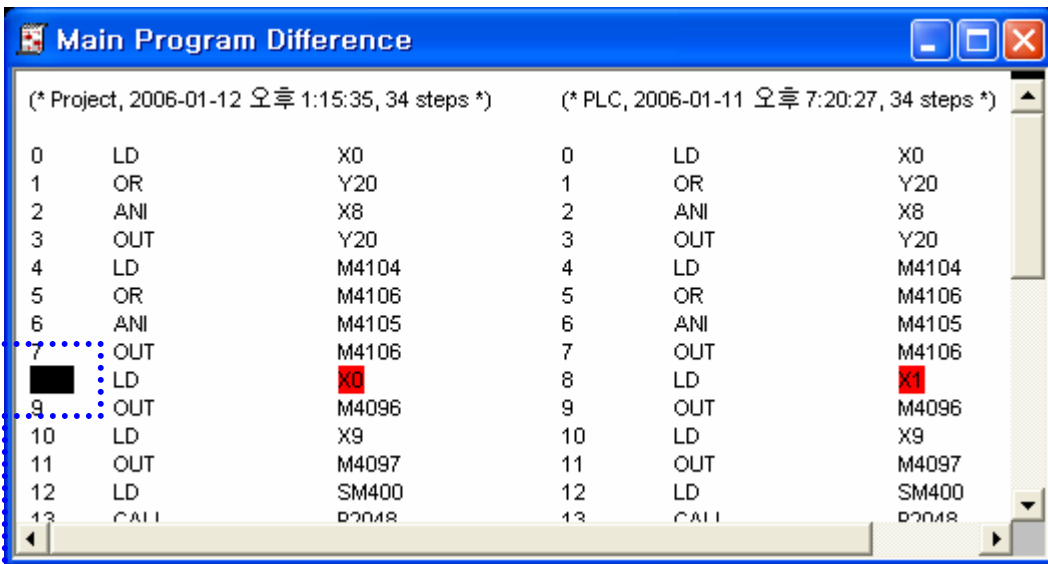


- PLC程序与 PC的程序之间
没有差异的话出现一致的
信息确认

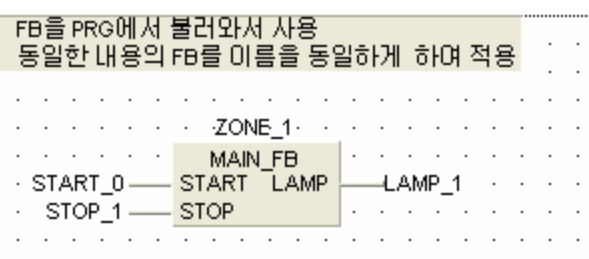
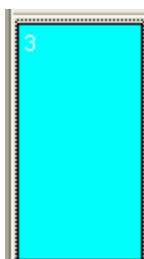


- PLC 程序与 PC程序之间
有不同地方的话出现
不一致的信息确认

- 选择错误部分后点击
Show的话可以确认
详细情报.



双击错误地方程序里显示有错误部分的网络
(网络显示颜色会变换)



修整错误部分以后再次校验程序并确认是否还有错误
(网络显示颜色在再次检验以后复原)

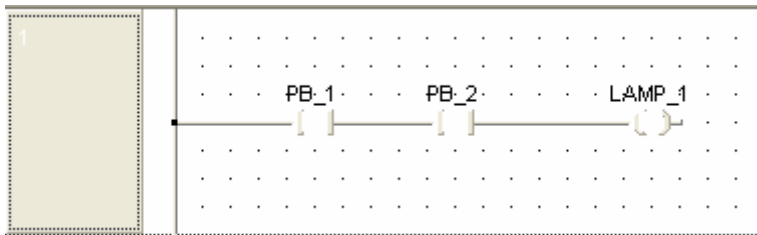


13. GX IEC Developer 使用 - (顺序 & 指令)

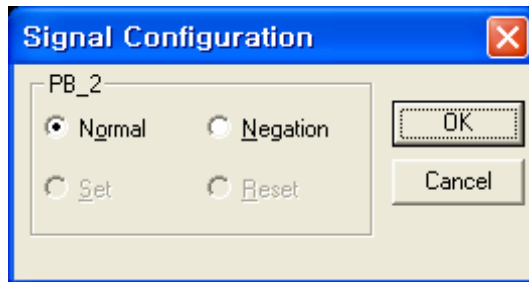
13.1 a触点 & b触点的设定

: 基本上是在LD方式里a触点 & b触点是LD符号来使用

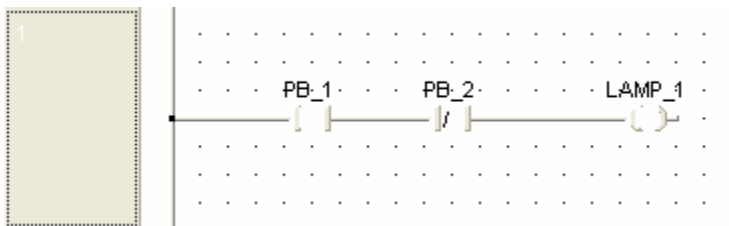
Ex.) 在下面的LD里把 PB_2改成b触点



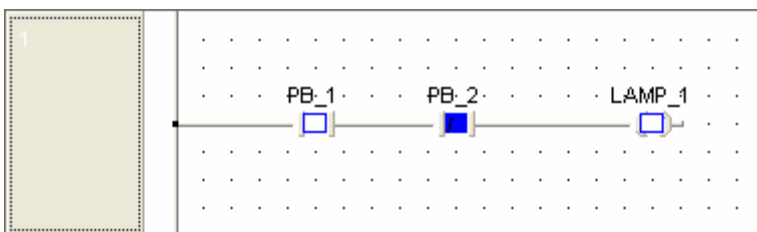
- ① 双击PB_2的触点符号.
 - 出现下面的信号构成窗口信息.
 - 在这里选择“Negation”处理b触点



- ② 显示为以下形式.



- ③ 以下是监控状态画面.



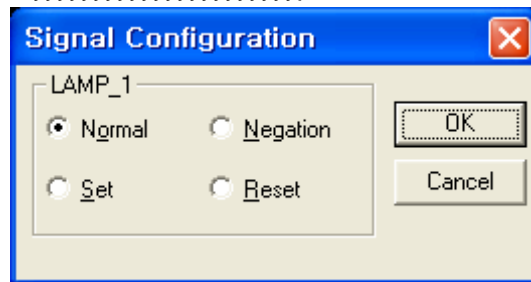


13.2 关于输出线圈的处理

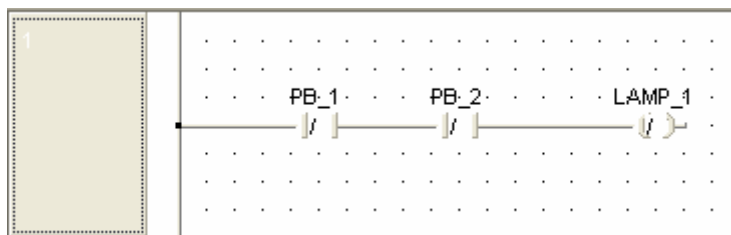
：输出线圈信号自身就可以处理以下的信号.

- ① 跟上一节一样双击线圈符号.
- 出现以下形式的关于输出线圈的信号构成窗口.

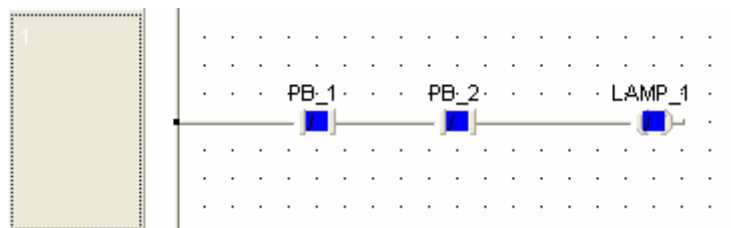
Fig13-1: 信号构成



- 输出线圈的“Negation”处理时与GX Developer的反转指令的功能同一使用
- 作成



- 监控



*** 参 考 ***

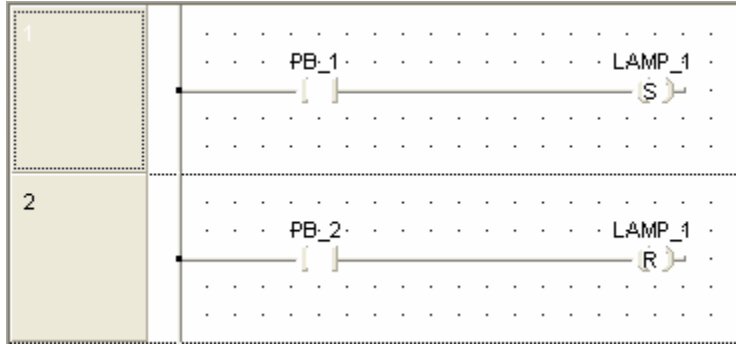
LD 输出的监控是否定输出虽然以ON的形式被监控
但实际输出是 OFF状态.



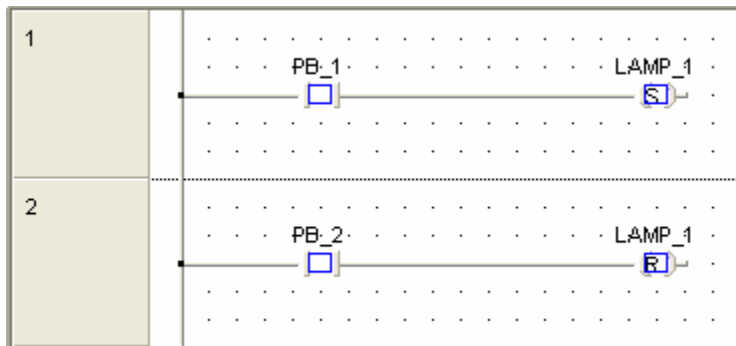
- SET & RESET 处理 (方法1)

: 与上一节[图13-1]一样输出线圈自身就可以设第SET与 RST信号

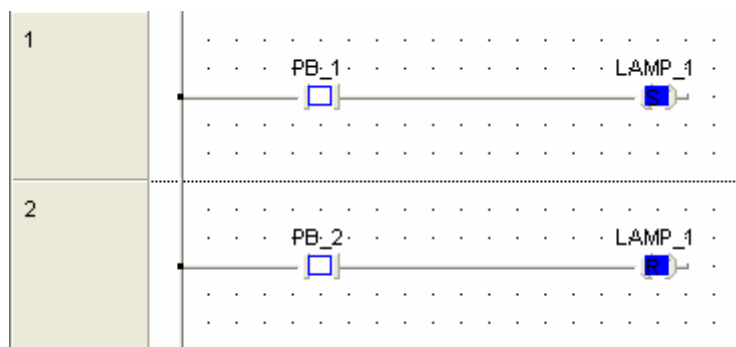
- 作成



- 监控 (没有被SET时)



- 监控 (被设定SET时)

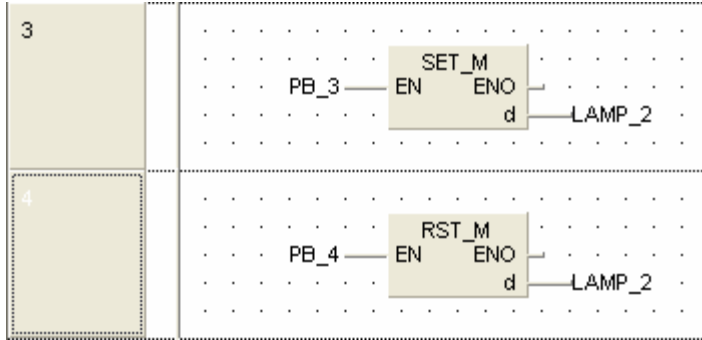




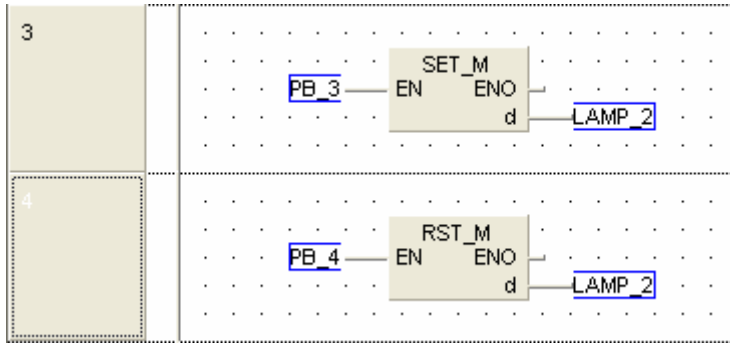
- SET & RESET 处理 (方法2)

: 一般 SET, RST 指令根据FUNCTION的处理

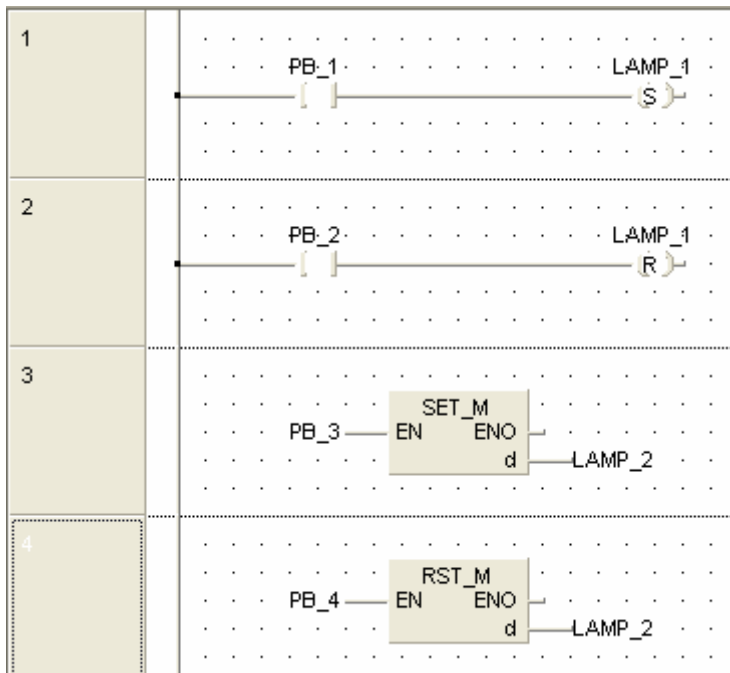
- 作成



- 监控



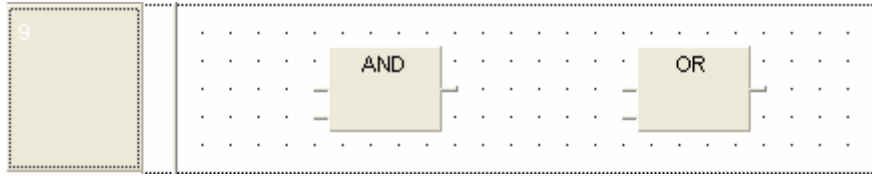
- 根据 方法1 & 方法2 的作成



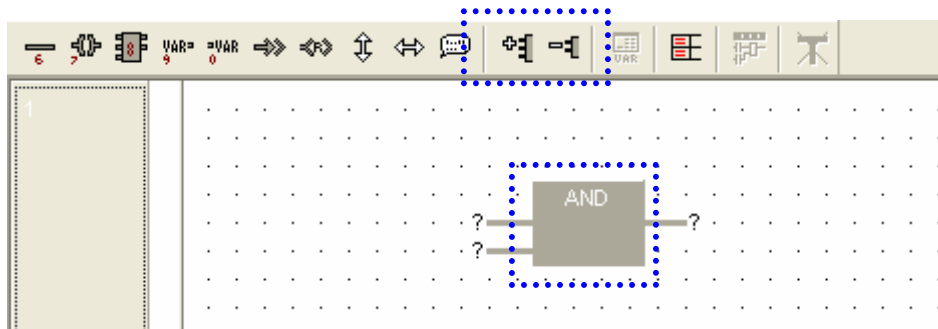


- 触点的 AND & OR 连接.

以下使用 AND & OR 的Function



** AND & OR的 Pin数的调整

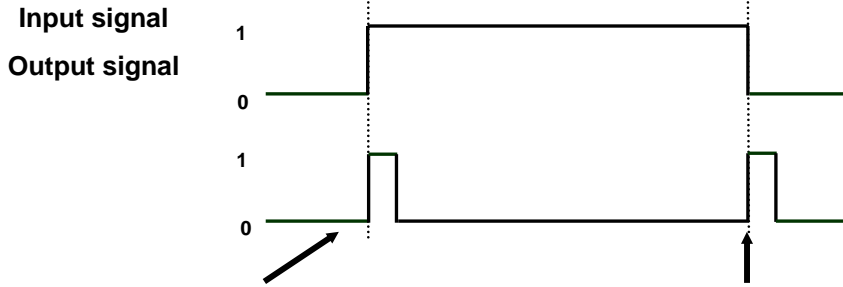


- ① 选择要调整 pin数的Function
- ② 根据工具图表上的增加/减少可以选择调整



13.3 脉冲处理

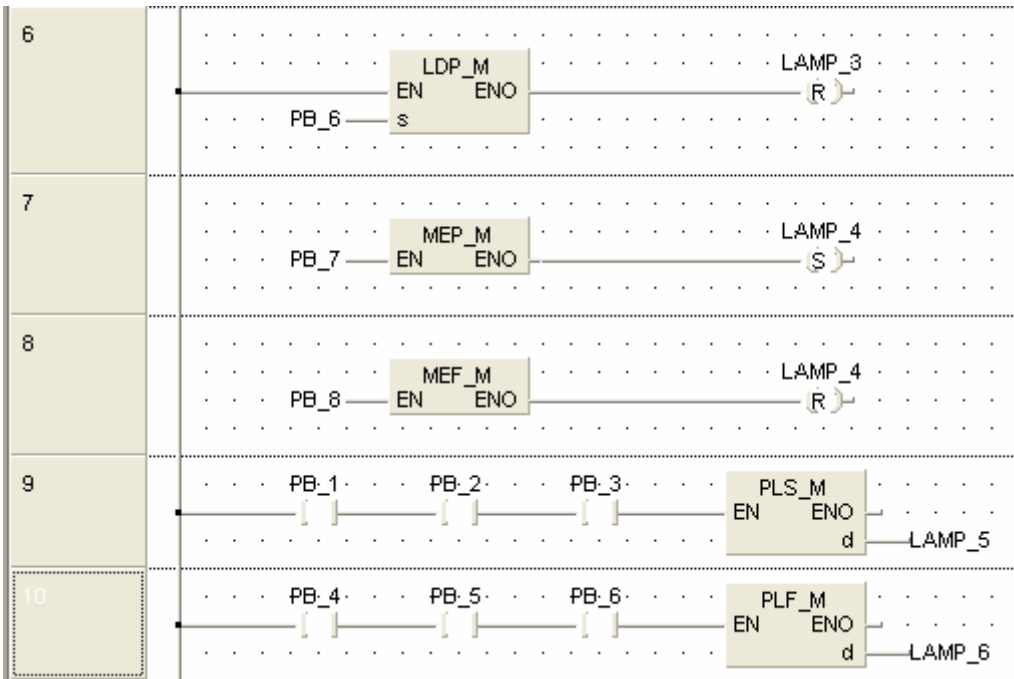
: 输入/输出信号的脉冲处理 Function



Input : LDP, ANDP, ORP, MEP 指令 处理
Output : PLS 指令 处理

Input : LDF, ANDF, ORF, MEF 指令 处理
Output : PLF 指令 处理

- 作成



*** 脉冲处理指令跟在 GX Developer里的指令一样根据以下规则来适用

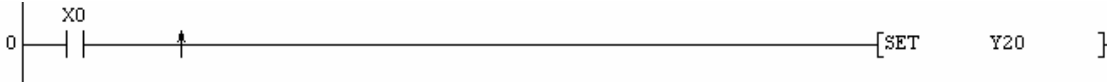
- 指令+P : 脉冲 上升时
- 指令+F : 脉冲 下降时



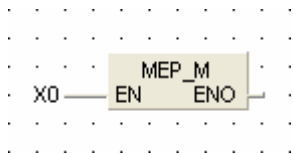
** GX Developer与 GX IEC 的比较

1) Edge处理

① GX Developer

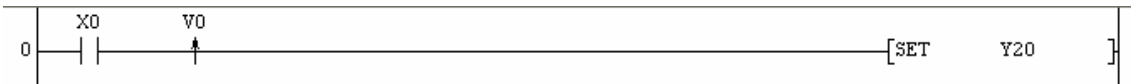


② GX IEC Developer



2) Edge 继电器使用时

① GX Developer

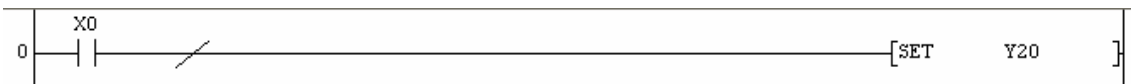


② GX IEC Developer

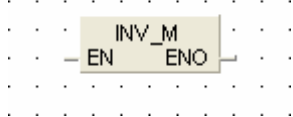


3) 反转处理

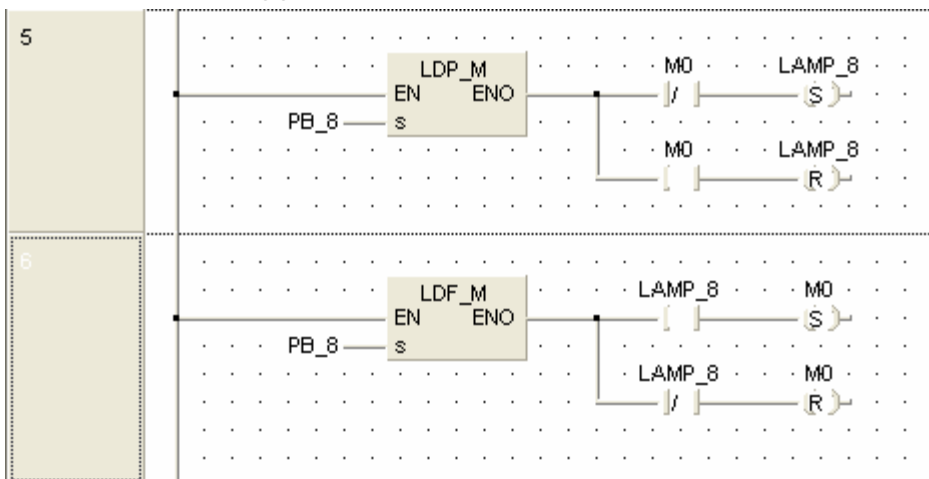
① GX Developer



② GX IEC Developer



Ex.) 简单的 Toggle 开关程序作成





13.4 Timer

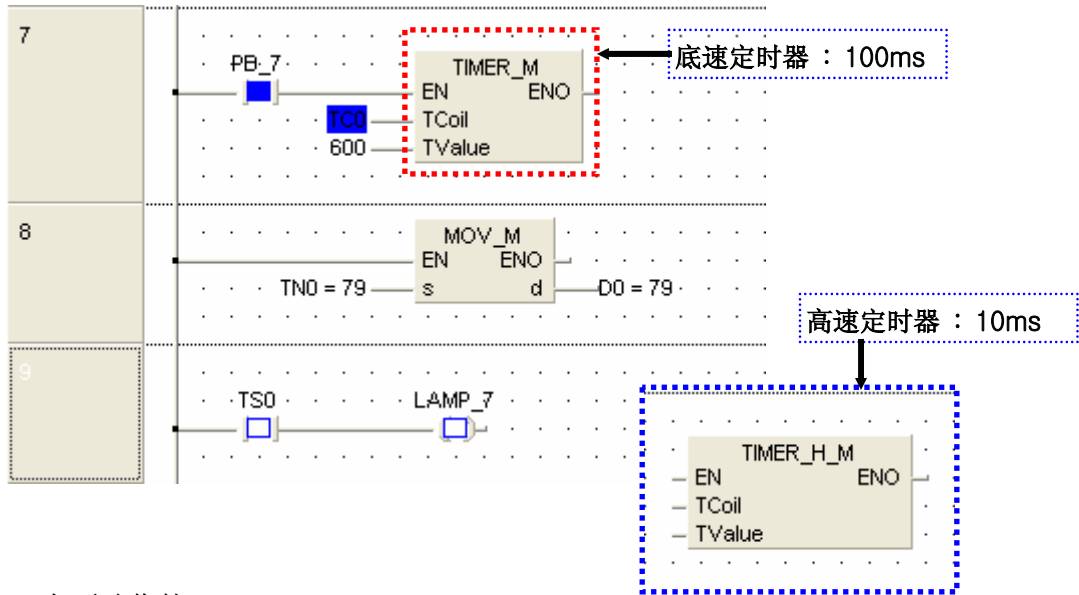
: Q PLC 系统里只有ON-Delay的形态可能

累积定时器的使用与GX Developer一样需要PLC 参数的设定

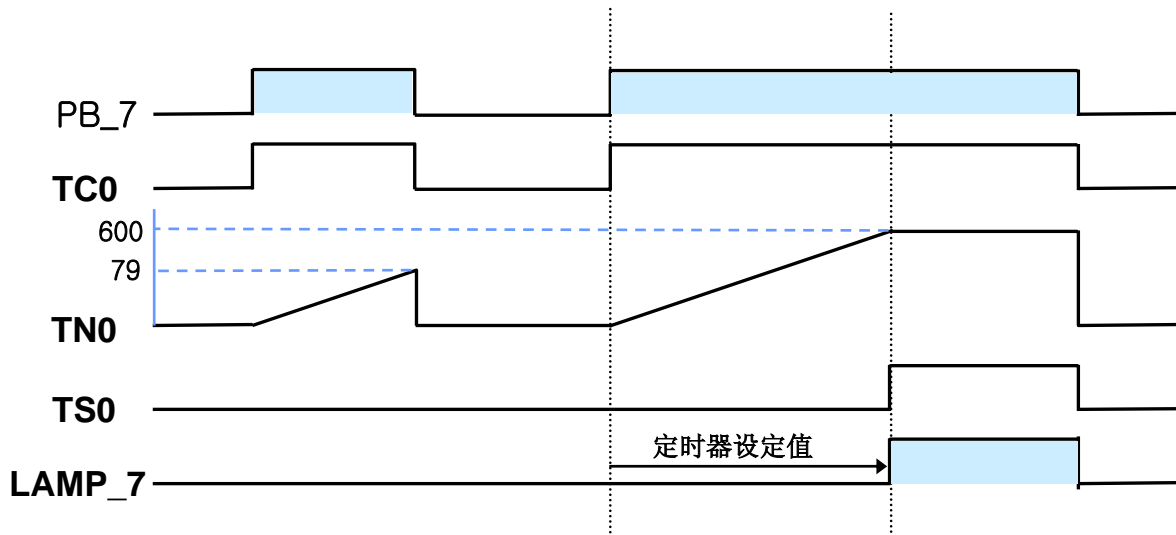
① Timer有以下4种要素构成. (累积定时器 = ST)

- (S)TV : 定时器设定值
- (S)TN : 定时器当前值
- (S)TC : 定时器线圈
- (S)TS : 定时器输出触点

- Ex.) 作成 (监控)



-- 上面动作的Time Chart

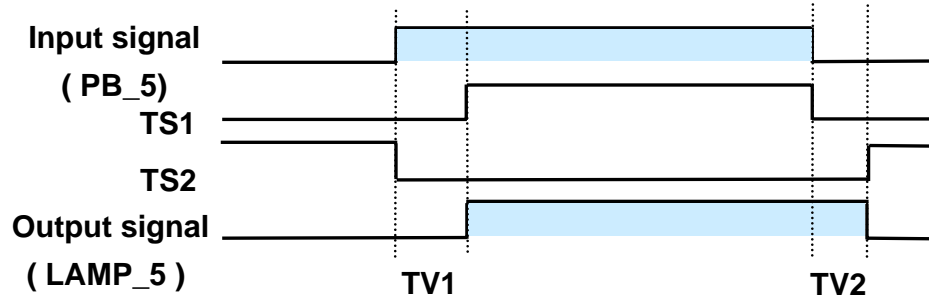




Ex.) 作成以下动作的程序

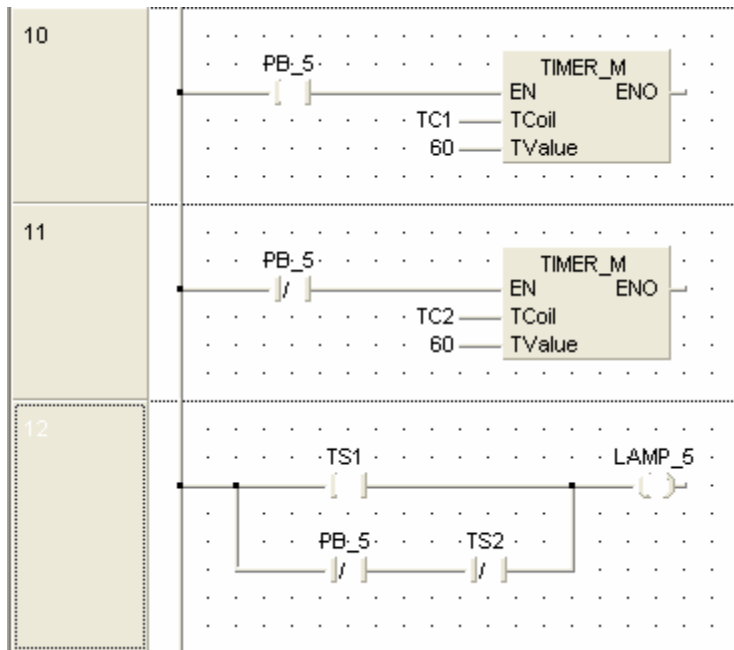
: 输入信号ON以后6秒后灯亮,输入信号OFF后6秒后灯灭

Timing:



Program

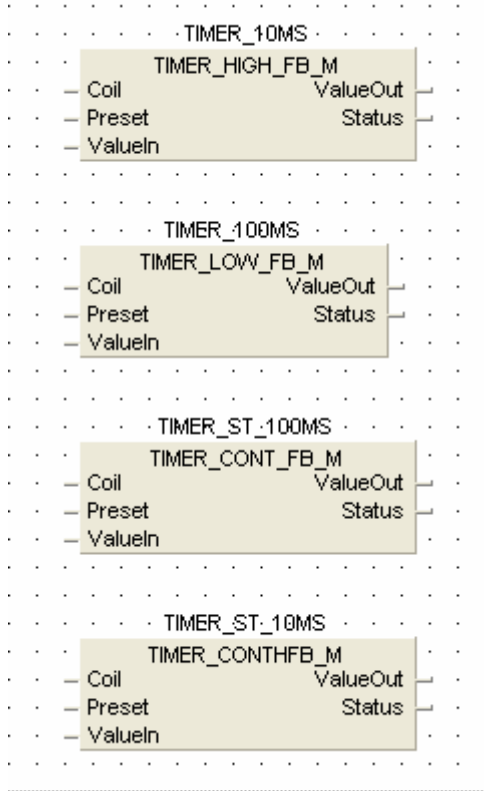
: 作成以下程序并确认.





*** 以FB作成的Timer的使用例

: 以FB来支援的定时器没有必要定时器软元件指定
定时器软元件由系统自动分配.



- 单位时间为10ms的高速定时器

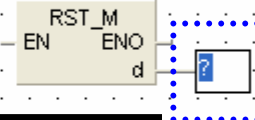
- 单位时间为100ms的底速定时器

- 单位时间为100ms的底速累积定时器
- [Extras]-[Option]-[System Variables]里
设定ST的值才有可能使用

- 单位时间为10ms的高速累积定时器

参考 累积定时器的Reset 处理

- Reset 处理是 把FB的 Coil 进行 Reset处理





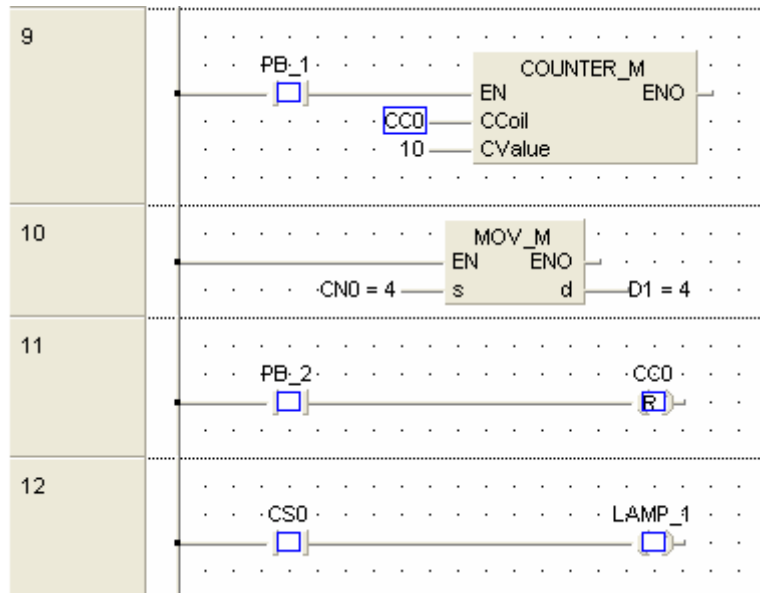
13.5 Counter

: Q PLC 系统里 只有UP Counter 形态适用

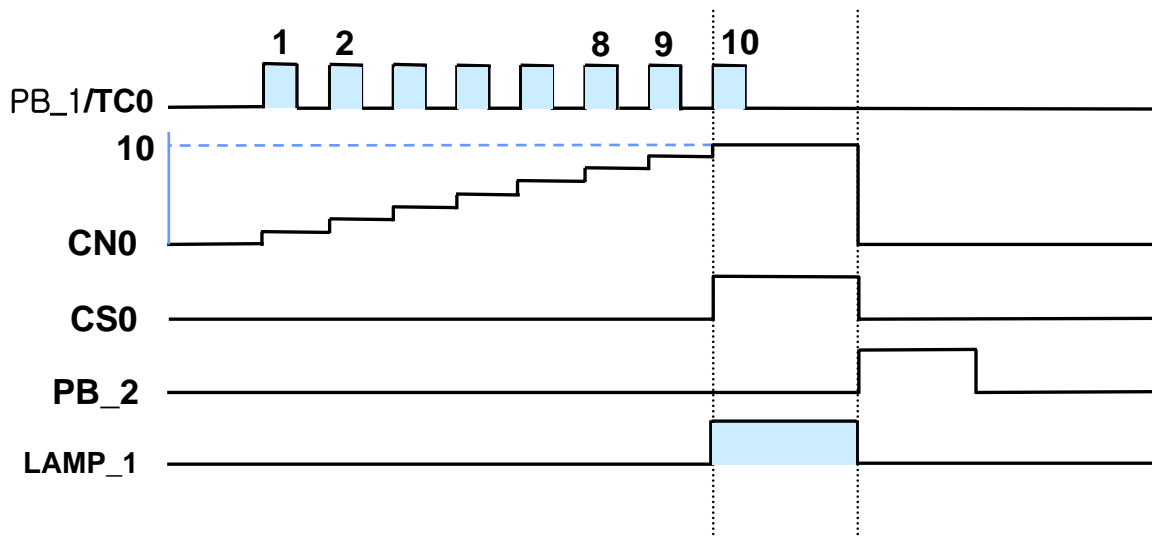
① Counter 由以下4种要素构成.

- CV : 记时器设定值
- CN : 记时器当前值
- CC : 记时器线圈
- CS : 记时器输出触点

- 作成 (监控)



-- 以上动作的Time Chart



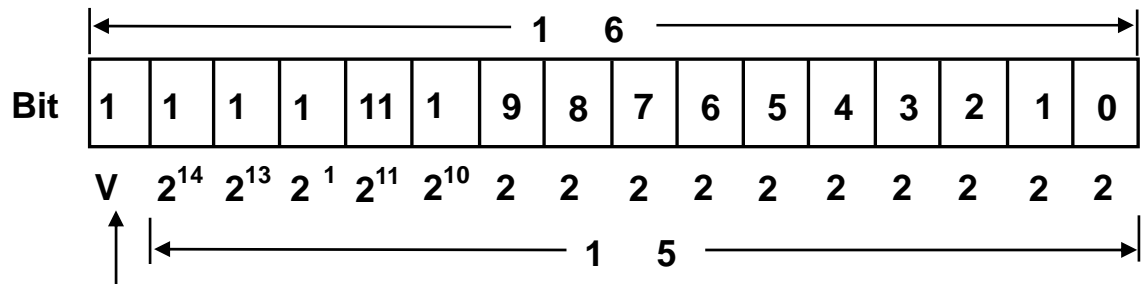


13.6 Data Registers

: Data Register是以数值DATA的存储空间来使用

数值寄存器是以16bit来构成,其中15bit是值, 1bit是符号

以下是构成



Sign

(0: positive value: 1: negative value)

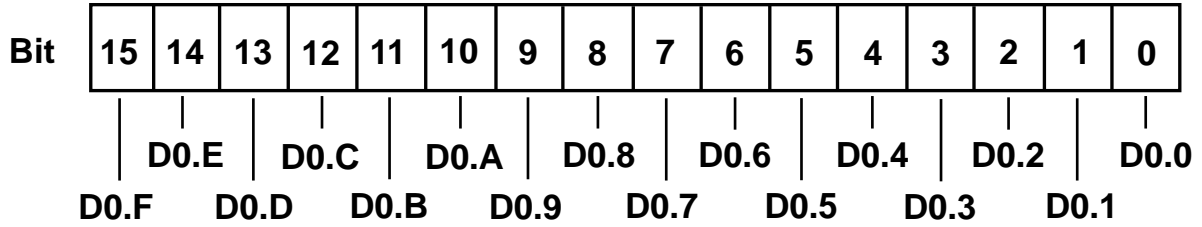
- 范围 : $-32768 \sim 32767$



13.7 Word 软元件的 BIT 指定

: 数值寄存器位软元件的各个bit可以以bit形态来使用可能

- Ex.) 数值寄存器的构成



- 全域变数里的选择形态

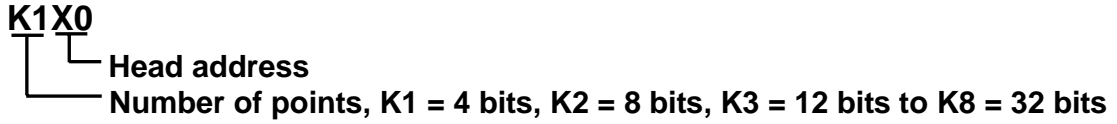
Global Variable List						
	Class	Au	Identifier	MIT-Addr.	IEC-Addr.	Type
0	VAR_GLOBAL	☞	Bit_9_of_D0	D0.9	%MX0.0.9	BOOL
1	VAR_GLOBAL	☞	Bit_10_of_D0	D0.A	%MX0.0.10	BOOL
2	VAR_GLOBAL	☞	Bit_11_of_D0	D0.B	%MX0.0.11	BOOL



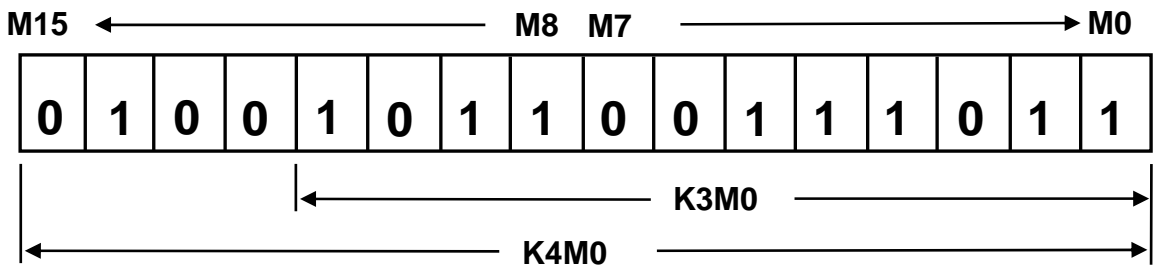
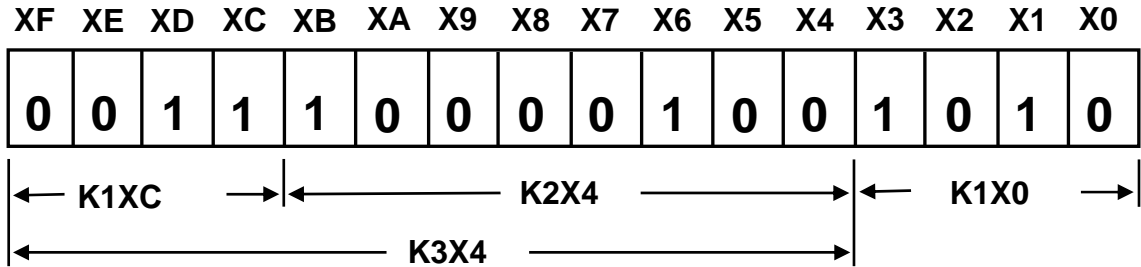
13.8 BIT 软元件的模块化

: BIT软元件可以以4bit单位来指定并使用.

- 由以下形式使用



Examples:



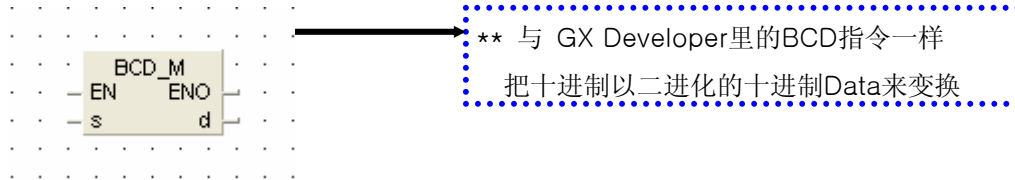
- 全域变数(Global Variable)里的选择形态

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type
0	VAR_GLOBAL	Display_1	K4Y40	%QW19.4.64	INT
1	VAR_GLOBAL	Display_2	K4Y50	%QW19.4.80	INT
2	VAR_GLOBAL	Display_3	K4Y60	%QW19.4.96	INT

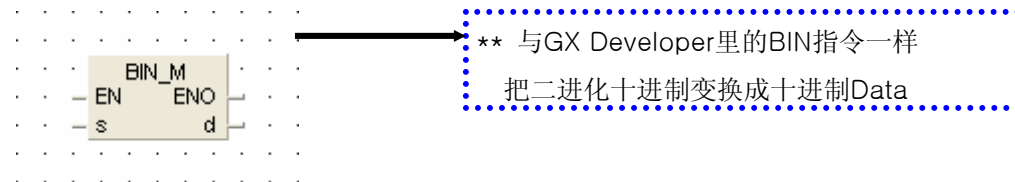


13.9 Data Code 变换

① BCD \Rightarrow BIN



② BIN \Rightarrow BCD

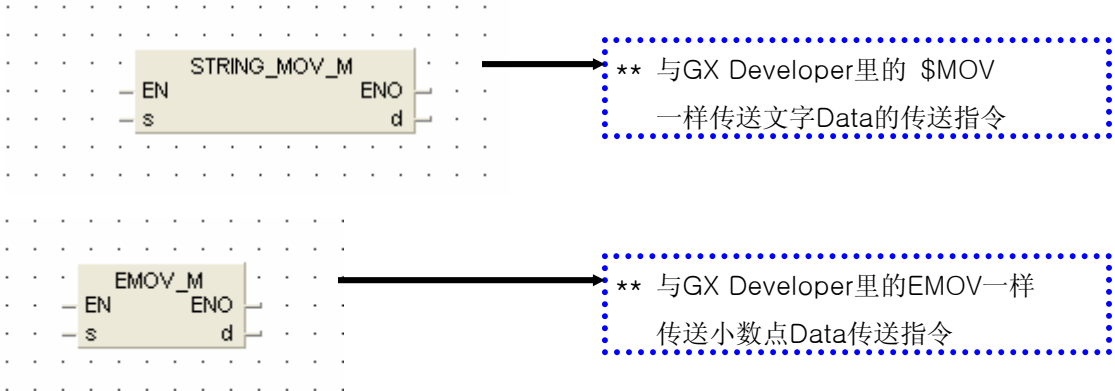
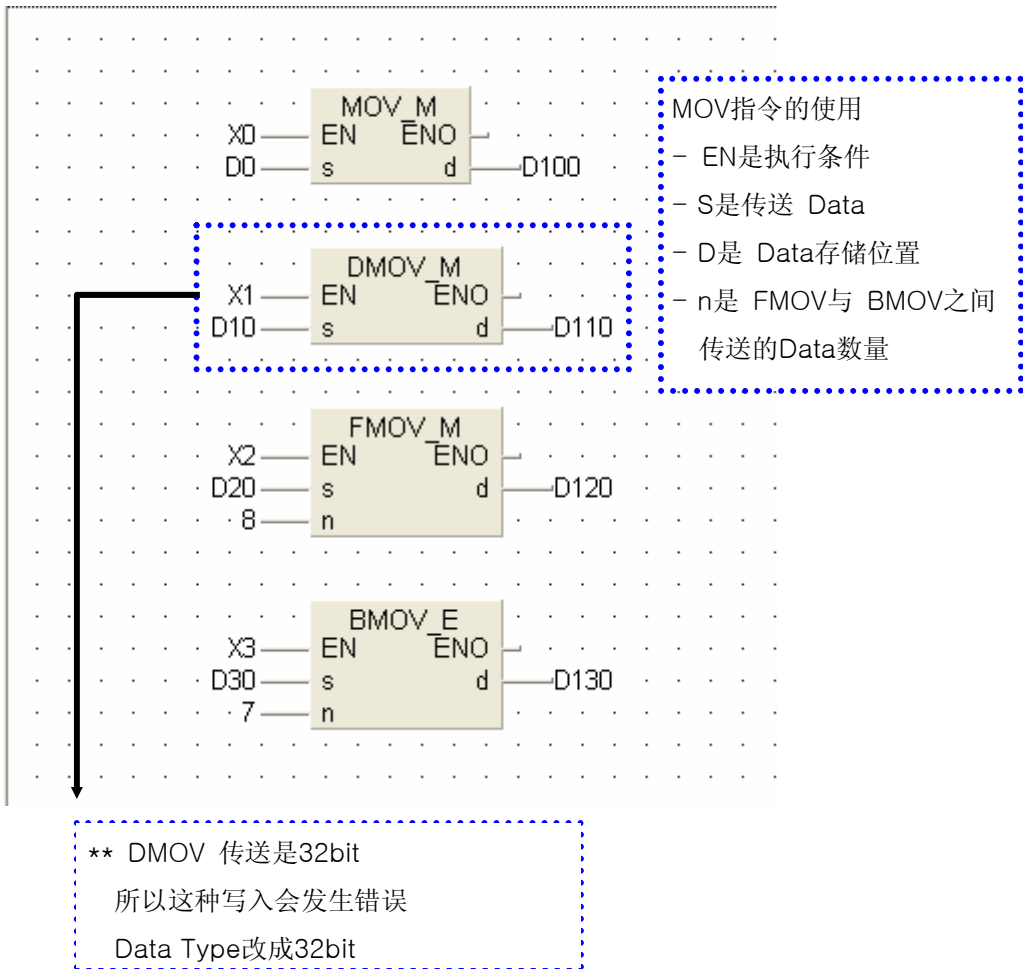




13.10 Data 传送

: 基本上在 GX Developer里使用的指令以 Library的 Manufacturer Library形态来表现.

- 说明基本的几种例子





13.11 比较运算指令

： 以下是比较运算指令的构成

比较演算内容 : $IN1 = IN2$



比较演算内容 : $IN1 > IN2$



比较演算内容 : $IN1 \geq IN2$



比较演算内容 : $IN1 \neq IN2$



比较演算内容 : $IN1 < IN2$

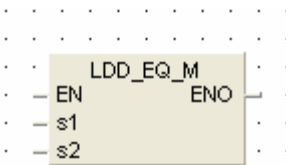


比较演算内容 : $IN1 \leq IN2$

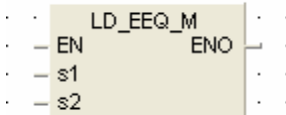


Comparison	Output = 1 if	Output = 0 if
Equal	$IN1 = IN2$	$IN1 \neq IN2$
Not equal	$IN1 \neq IN2$	$IN1 = IN2$
Greater than	$IN1 > IN2$	$IN1 \leq IN2$
Less than	$IN1 < IN2$	$IN1 \geq IN2$
Greater equal	$IN1 \geq IN2$	$IN1 < IN2$
Less equal	$IN1 \leq IN2$	$IN1 > IN2$

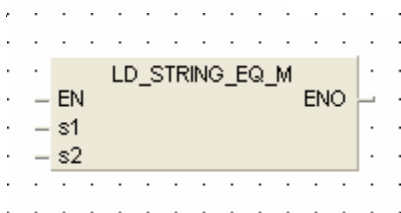
*** 其他比较运算指令 ***



* 32bit的 比较运算



* 实数值 比较运算



* 文字列 比较运算



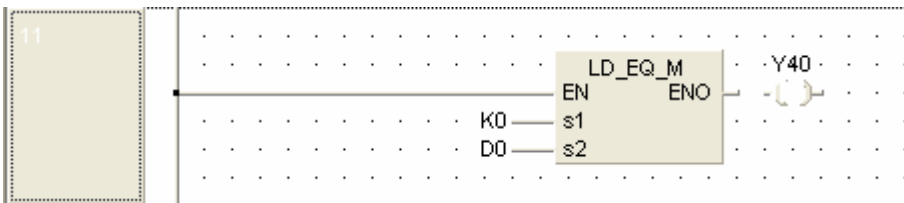
*** 比较运算指令例子 ***

㊸ LD 连接

- GX Developer

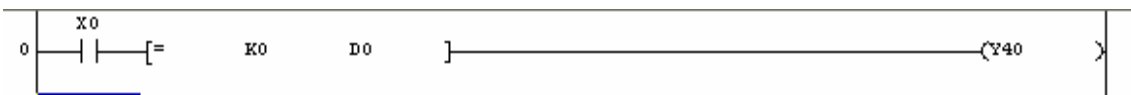


- GX IEC Developer

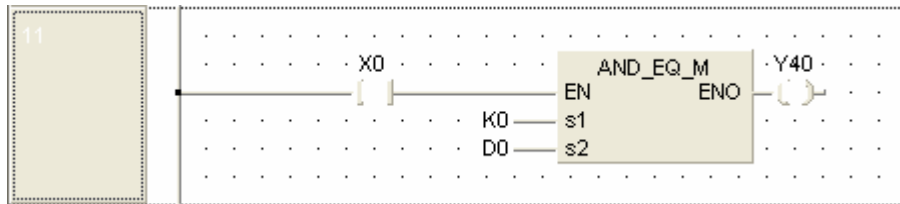


㊸ AND 连接

- GX Developer

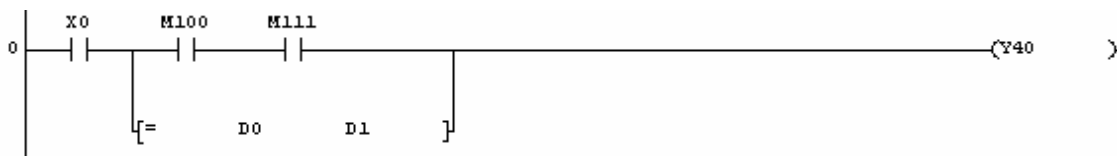


- GX IEC Developer

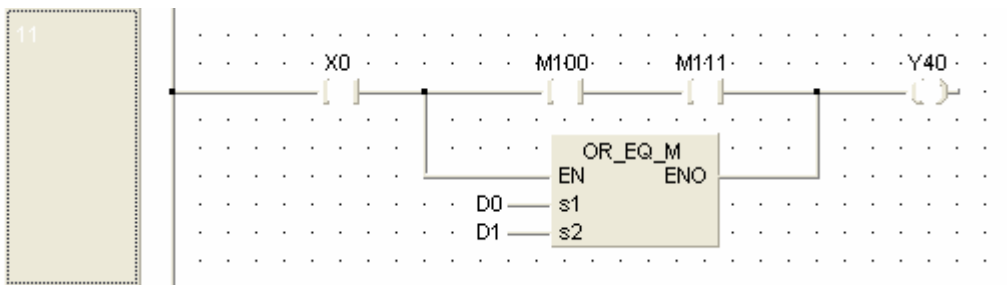


㊸ OR 连接

- GX Developer



- GX IEC Developer



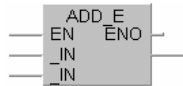


13.12 算术运算指令

：以下是基本的算术运算指令构成.

加法

Example:

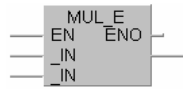


ADD, ADD_E

$$d = IN1 + IN2$$

乘法

Example:

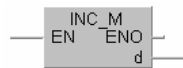


MUL, MUL_E

$$d = IN1 \times IN2$$

增法

Example:

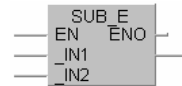


INC_M d

$$d = d + 1$$

减法

Example:



SUB, SUB_E

$$d = IN1 - IN2$$

除法:

Example:

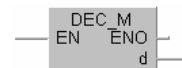


DIV, DIV_E

$$d = IN1 / IN2$$

减法

Example:



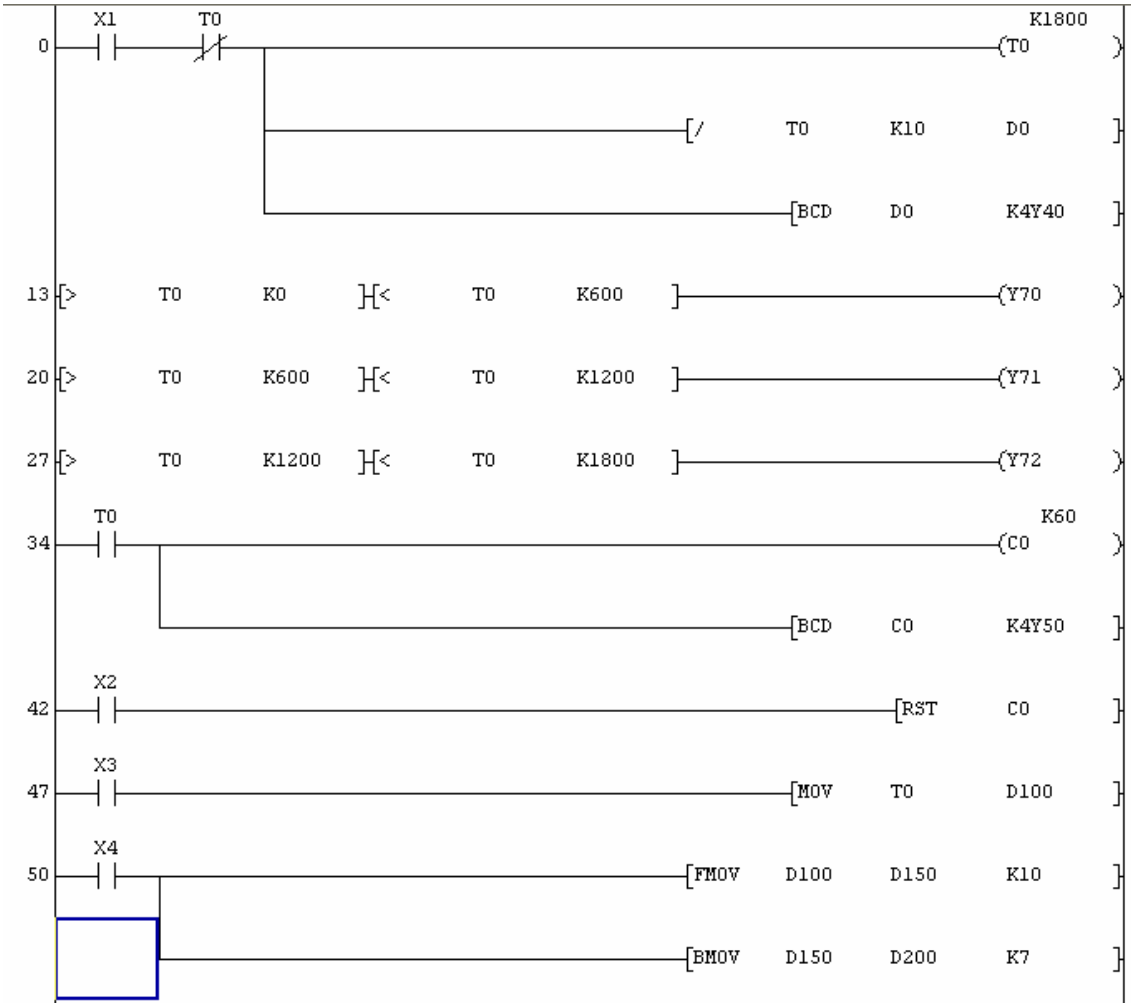
DEC_M d

$$d = d - 1$$



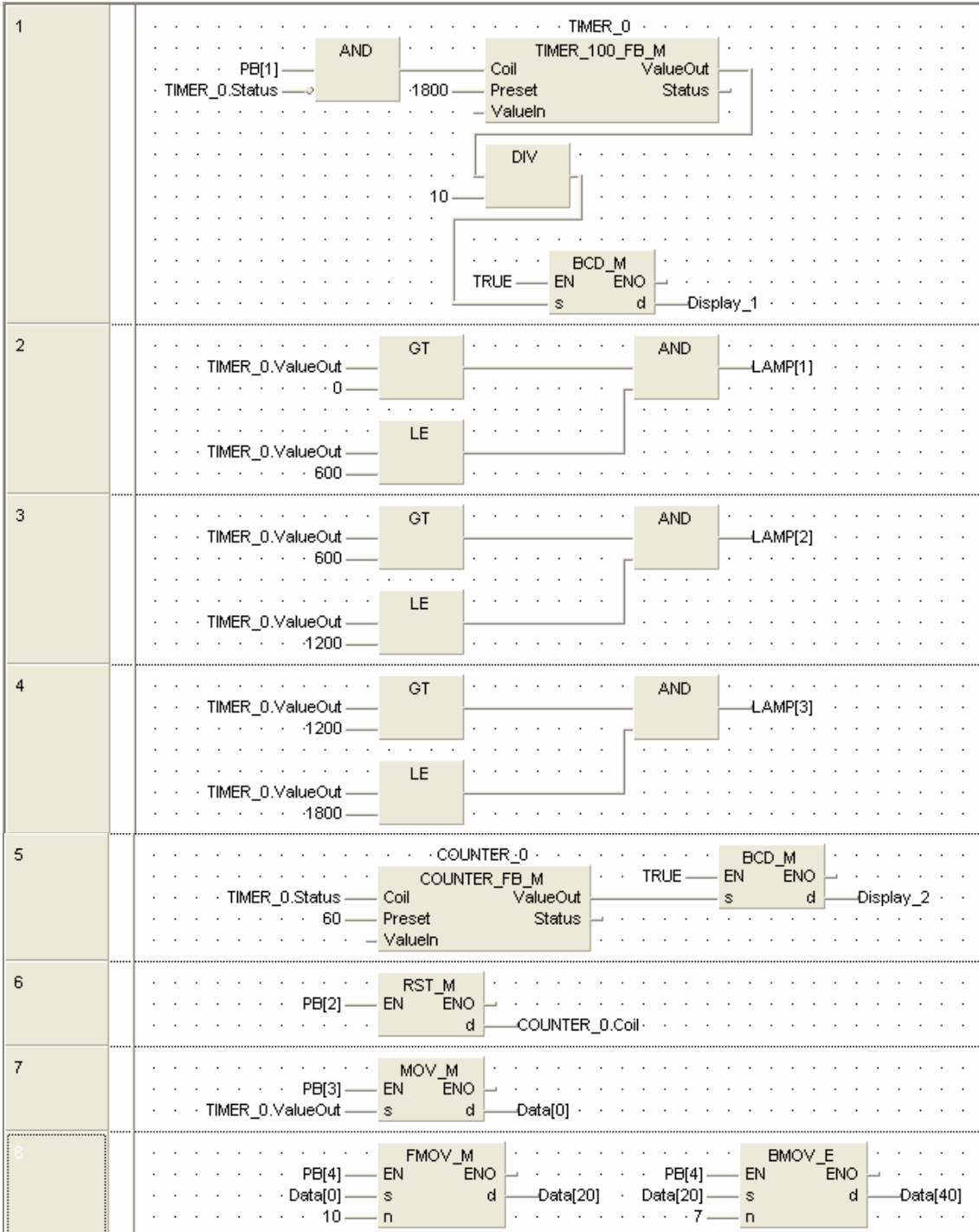
**** 下面程序是用 GX Developer来作成的程序
 简单用GX IEC Developer来作成

参考) 作成是所有地址都进行变数处理。
 “D” 地址是系统自动分配或则可以直直接输入





*** 下面是简单改变作成的程序
供参考





13.13 FROM & TO 指令

: 读写 Buffer Memory的Data时使用.

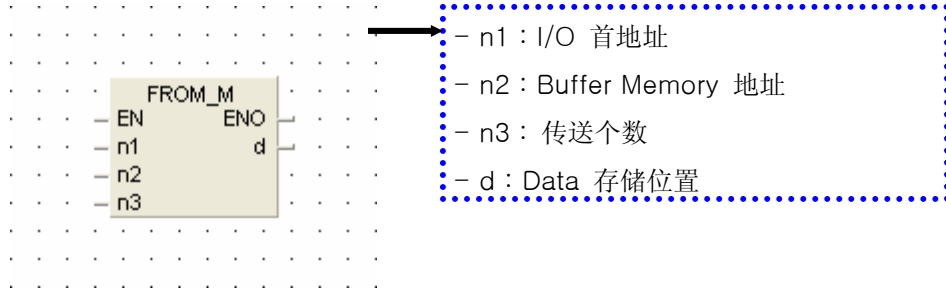
- FROM 指令

: Buffer Memory Data 读

GX Developer



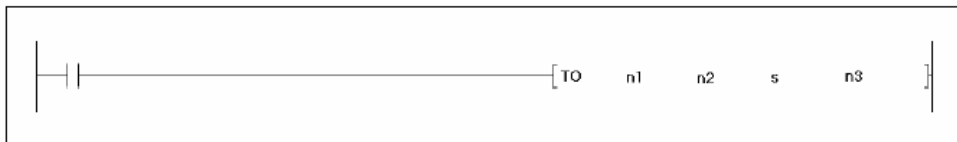
GX IEC Developer



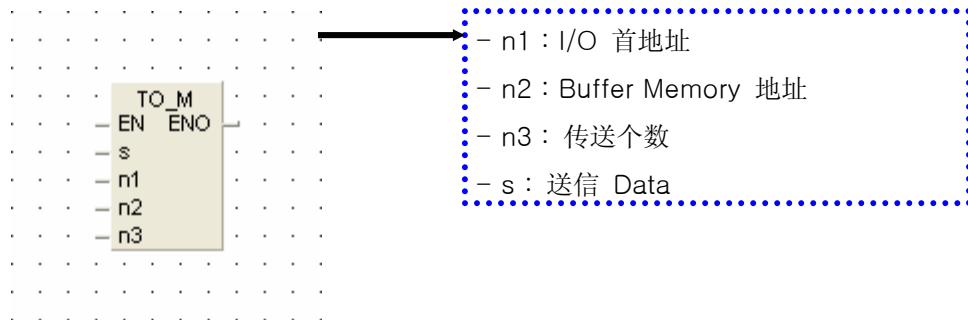
- TO 指令

: Buffer Memory Data 写

GX Developer



GX IEC Developer



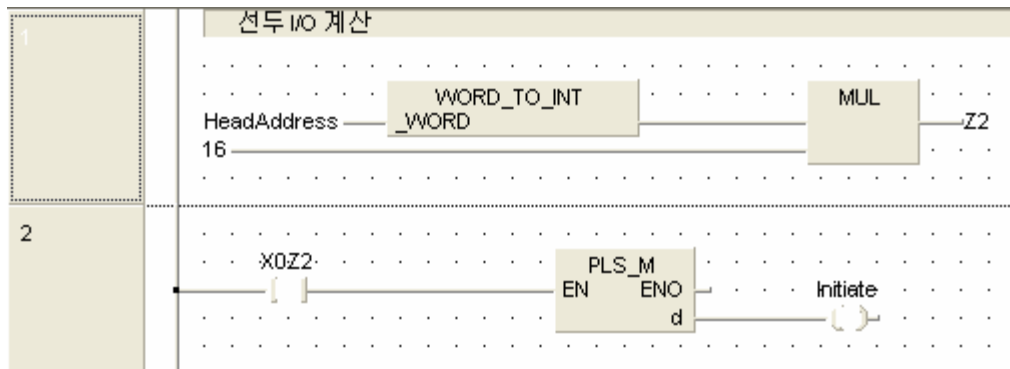


13.14 变址寄存器

: 变址寄存器是间接指定软元件号的时候使用
即, 在直接指定的软元件号上加上变址寄存器的内容后
再指定软元件号.

Ex.) 简单程序

- “X0Z0” 是 根据Z0的值软元件号就会变化.





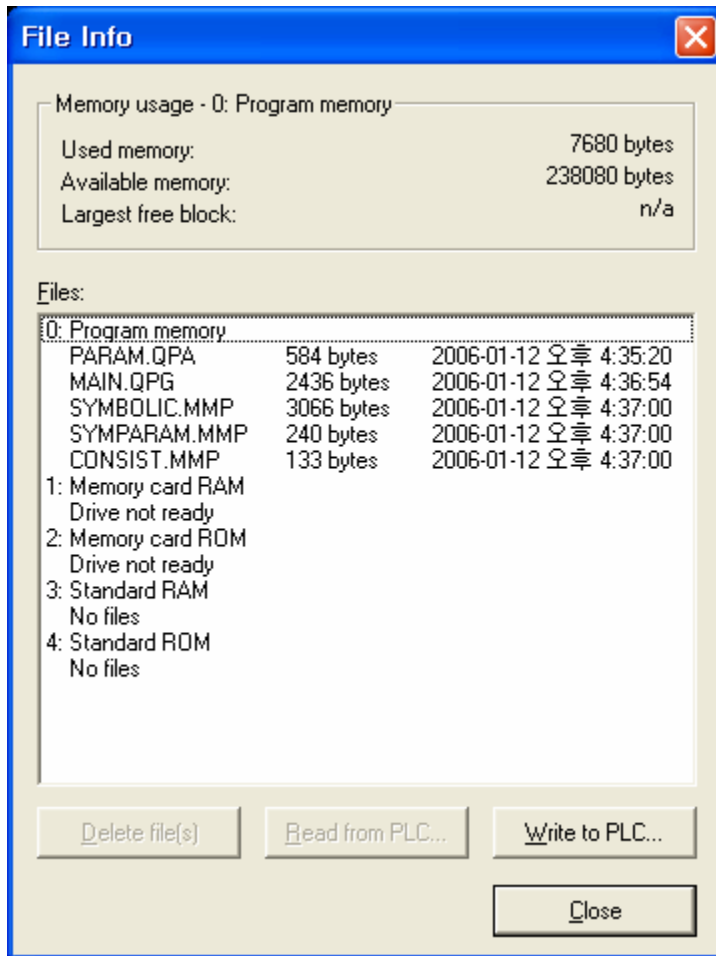
14. GX IEC Developer 使用 - (Debug / 监控功能)

: 关于PLC在线情况下功能的说明.

14.1 File Information

: 可以确认写入PLC里的各File的情报.

⇒ 选择路径 : [Online] - [File Info]



- 在CPU的各Drive里所使用的file容量的情报
全部使用内存容量及
可用内存容量的确认

- 使用方法
先选择所确认内存的文件, 在
Memory usage里确认全部容量
现在所使用的容量..



14.2 回路监控及Online Change Mode

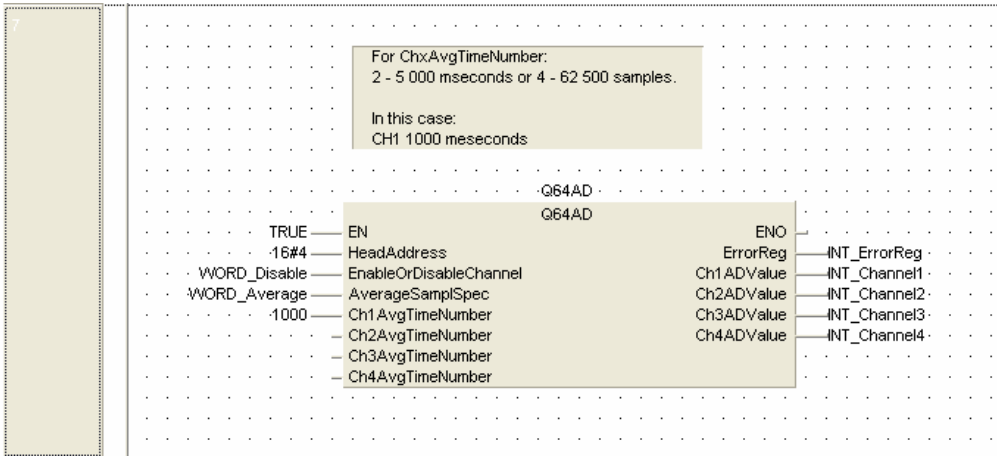
: 确认作成程序的动作状态及PLC运行中变更回路时所使用

⇒ 选择路径

⇒ 回路监控画面



⇒ Online Change Mode 使用



- 在回路监控前状态下
选择功能图表
- 修整要换的变数
- [Check] 后施行自动
写入



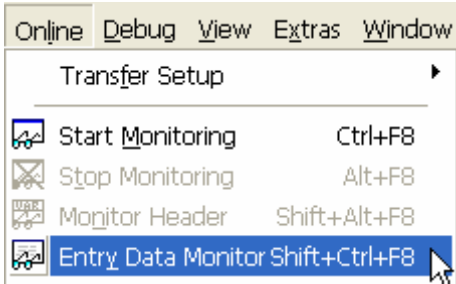
14.3 登录Data 监控

：表示要确认的Data的当前状态

可以指定实际软元件或指定变数的情况.

14.3.1 指定实际软元件的方法

- 顺序 -



[Online] - [Entry Data Monitor] 选择

登录窗被激活.

Pos	Address (MIT)	Name	Value (dec)
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

加入要确认的软元件.

Pos	Address (MIT)	Name
1	K2 X10	
2		
3		

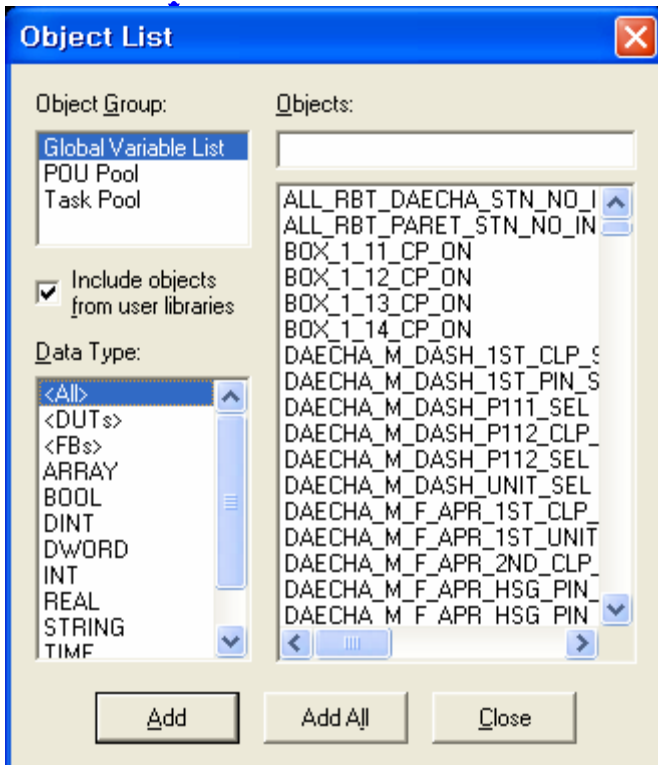
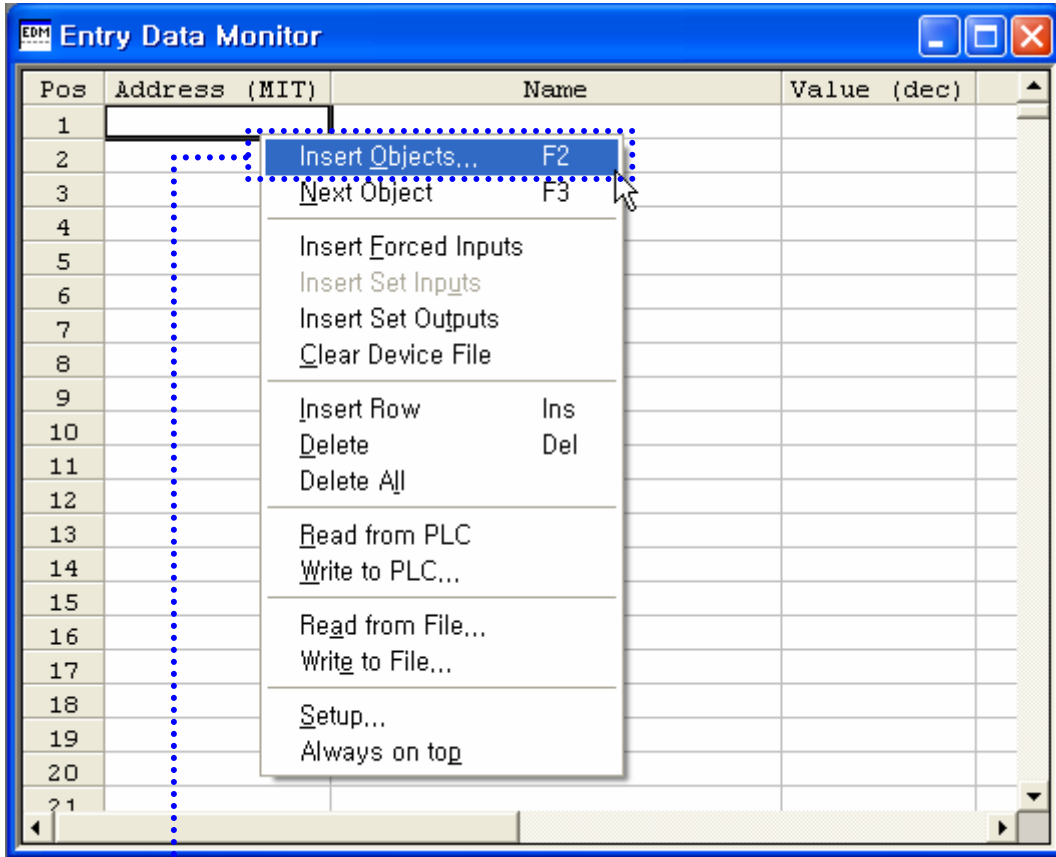
追加别的软元件

Pos	Address (MIT)	Name	Value (dec)
1	K2 X10	K2 X10	10
2	DO	DO	10
3	Y25	Y25	0
4			



14.3.2 指定变数并监控

- 顺序 -



- Object List里的设定顺序
1. Object Group 选择.
(在3个项目当中选择
根据各选择项目的
设定而所监控的
内容就不一样
 2. Data Type 选择
(Global 选择的时候
被激活.)
 3. Object 选择.
(选择要监控的对象)



** 详细说明 **

- ① Global Variable List 设定
 - 选择要监控的对象并选择Add
 - 要指定全部的时候选择Add_All.
 - 监控选择画面

Pos	Name	Type	Address (MIT)	i
1	ALL RBT DAECHA STN NO INTER	BOOL	M6206	*
2	ALL RBT PARET STN NO INTER	BOOL	M6205	*
3	BOX 1 11 CP ON	BOOL	X61F	*
4	BOX 1 12 CP ON	BOOL	X65F	*
5	BOX 1 13 CP ON	BOOL	X69F	*
6	BOX 1 14 CP ON	BOOL	X6CF	*
7	DAECHA M DASH 1ST CLP SEL	BOOL	M6204	*
8	DAECHA M DASH 1ST PIN SEL	BOOL	M6203	*
9	DAECHA M DASH P111 SEL	BOOL	M6202	*
10	DAECHA M DASH P112 CLP SEL	BOOL	M6201	*
11	DAECHA M DASH P112 SEL	BOOL	M6200	*
12	DAECHA M DASH UNIT SEL	BOOL	M6199	*
13	DAECHA M F APR 1ST CLP SEL	BOOL	M6198	*
14	DAECHA M F APR 1ST UNIT SEL	BOOL	M6197	*
15	DAECHA M F APR 2ND CLP SEL	BOOL	M6196	*

② POU Pool 选择

- POU Pool的 List中选择要确认的POU
- 参考下面

Pos	Name	Type	Address (MIT)	i
1	+DEACHA PARAMETER SETTING	STRUCTURE		
2				

* 选择+时在选择的POU里使用的内部结构

及关于变数内容的确认可能.

Pos	Name	Type	Address (MIT)	Address
1	-DEACHA PARAMETER SETTING	STRUCTURE		
2	+CC LINK 05	STRUCTURE		
	+Station Information	STRUCTURE		
	+Reserve Information	STRUCTURE		
5	+Invalid Information	STRUCTURE		
6	+Send Receive Information	STRUCTURE		
7	Setting Information	INT	D10090	%MWO.1009
8	Connect Information	INT	D10091	%MWO.1009
9	Headaddress Information	INT	D10092	%MWO.1009

Open 状态

Close 状态



- 项目追加

Pos	Address (MIT)	Name
1	K2 X10	K2 X10
2	D0	D0
3	Y25	
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		

- Insert Objects... F2
- Next Object F3
- Insert Forced Inputs
- Insert Set Inputs
- Insert Set Outputs
- Clear Device File
- Insert Row Ins
- Delete Rows Del
- Delete All
- Read from PLC
- Write to PLC...
- Read from File...
- Write to File...
- Setup...
- Always on top

需要更详细内容的时候

- 点击鼠标右键
- Setup 选择
- 跟着箭头方向进行

Pos	Field
1	Address (MIT)
2	Name
3	Value (dec)
4	
5	
6	
7	
8	
9	
10	
11	

Buttons: Close, Cancel, Read Setup..., Write Setup..., Password..., Security on

Don't search variables in GVL

Monitor only visible objects in window

Pos	Field
1	A
2	N
3	V
4	
5	
6	
7	
8	
9	
10	
11	

Field List

- Address (IEC)
- Address (MIT)
- Comment
- Name
- Status
- Type
- Value (bin)
- Value (oct)
- Value (dec)
- Value (hex)

Buttons: OK, Cancel

Don't search variables in GVL

Monitor only visible objects in window

Pos	Field
1	Address (MIT)
2	Name
3	Value (dec)
4	Value (bin)
5	
6	
7	
8	
9	
10	
11	

Buttons: Close, Cancel, Read Setup..., Write Setup..., Password..., Security on

Don't search variables in GVL

Monitor only visible objects in window



14.4 Buffer Memory 监控

: 监控特殊模块的内部存储的Data.

(与GX Developer的监控画面一样.)

- 使用 -

「Debug」 - 「Buffer Memory Batch...」 里表示画面.

Address	+FEDC	+BA98	+7654	+3210	
0000	0000	0000	0000	1110	14
0001	0000	0000	0011	0010	50
0002	0000	0000	0000	0000	0
0003	0000	0000	0000	0000	0
0004	0000	0000	0000	0000	0
0005	0000	0000	0000	0000	0
0006	0000	0000	0000	0000	0
0007	0000	0000	0000	0000	0
0008	0000	0000	0000	0000	0
0009	0000	0001	0000	0000	256
000A	0000	0000	1111	0001	241
000B	0000	0101	1011	1110	1470
000C	0000	0000	0000	0000	0
000D	0000	0000	0000	0000	0
000E	0000	0000	0000	0000	0
000F	0000	0000	0000	0001	1
0010	0000	0000	0000	0000	0
0011	1111	1111	1111	1110	-2
0012	0000	0000	0000	0000	0
0013	0000	0000	0000	0000	0
0014	0101	0101	0101	0101	21845
0015	0101	0101	0101	0101	21845
0016	0000	0000	0000	0000	0
0017	0000	0000	0000	0000	0
0018	0000	0000	0000	0000	0
0019	0000	0000	0000	0000	0
001A	0000	0000	0000	0000	0

- * 追加功能的使用
- Option Setup
- Device Test



14.5 Device 编辑

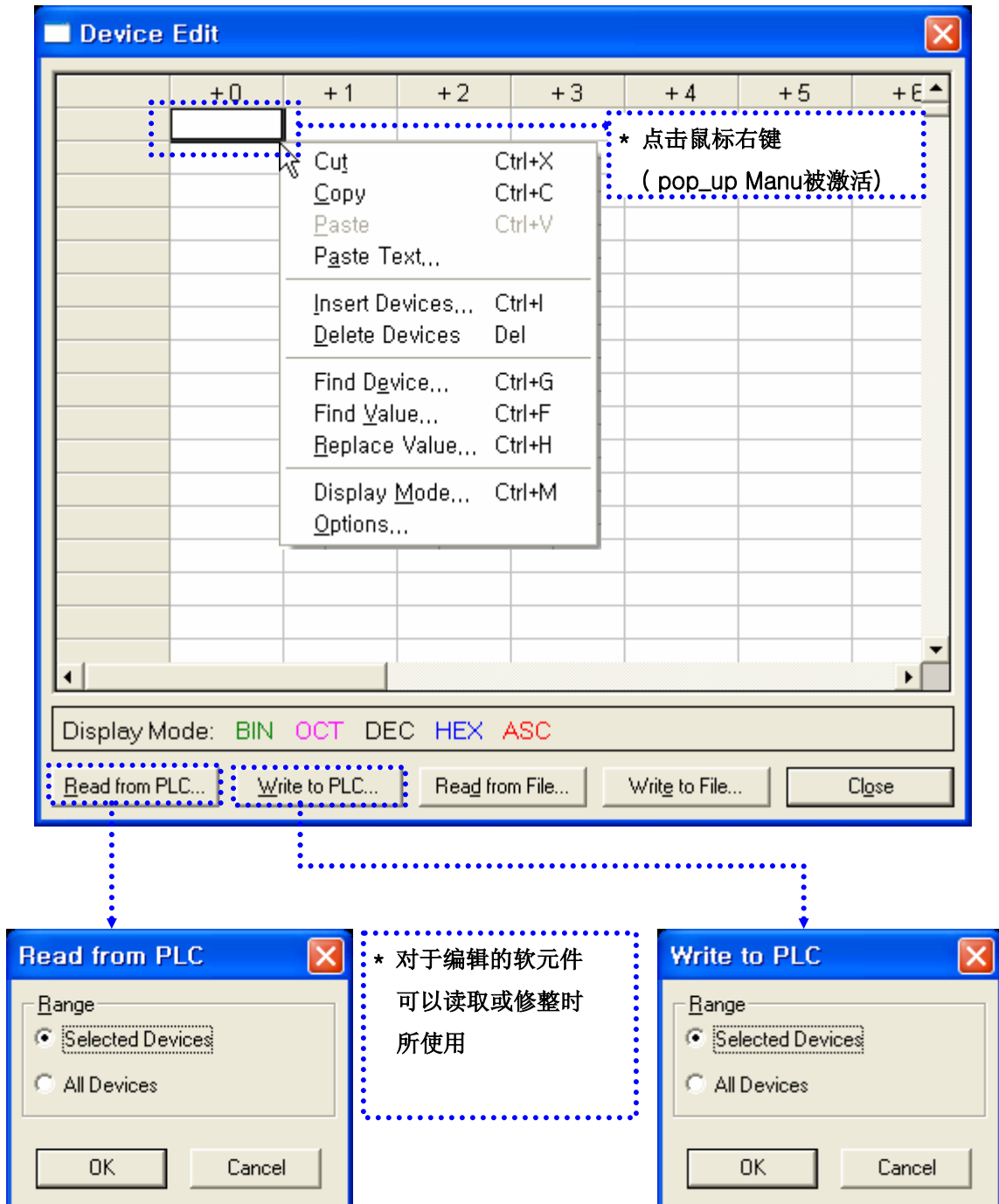
: 为了PLC CPU的 Word 及 bit 软元件的表示及编辑的功能.

软元件编辑画面是Excel的构成相似并使用同样的方法

- 顺序 -

[Debug] - [Device Edit...] 选择

- 操作画面 -

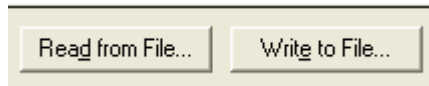




- 设定方法 -

1. 选择软元件
2. 软元件的范围设定
3. 设定表示方法

*** Excel 操作 ***



- 操作

Read from File : 读取 用Excel操作的数据.

Write to File : 用GX IEC作业的data以 Excel 文件来保存.

- 画面

Device Edit		GX IEC 画面						
		+0	+1	+2	+3	+4	+5	+E
X0		0	0	0	0	0	0	
D0		0	0	0	0	0	0	
D10		0	0	0	0	0	0	
D20		0	0	0	0	0	0	

register.xls		Excel 画面								
	A	B	C	D	E	F	G	H	I	
1	X0	0	0	0	0	0	0	0	0	
2	D0	0	0	0	0	0	0	0	0	
3	D10	0	0	0	0	0	0			
4	D20	0	0	0	0	0	0	0	0	

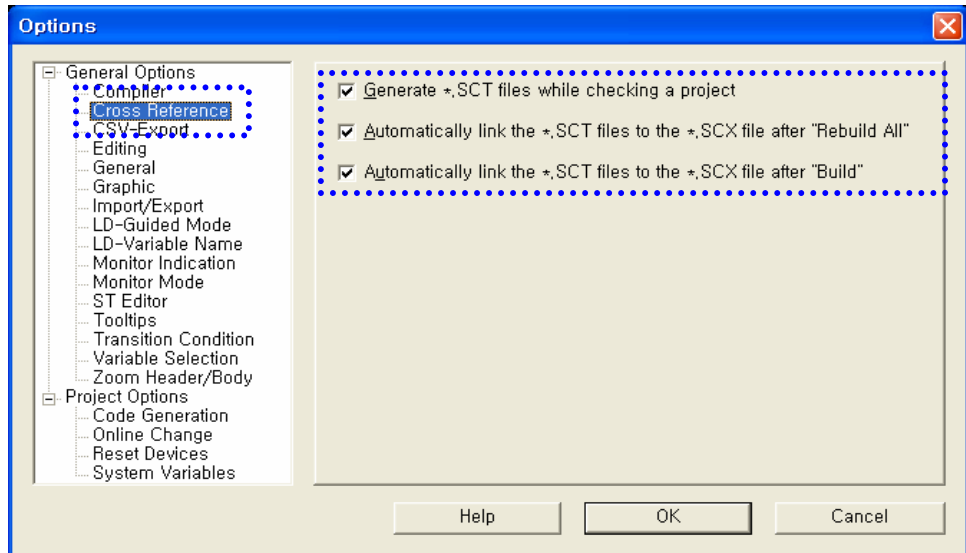


14.6 Cross Reference

: 把所有工程的的变数以 database的List来制作.

- 制作顺序

① [Extras] - [Option]

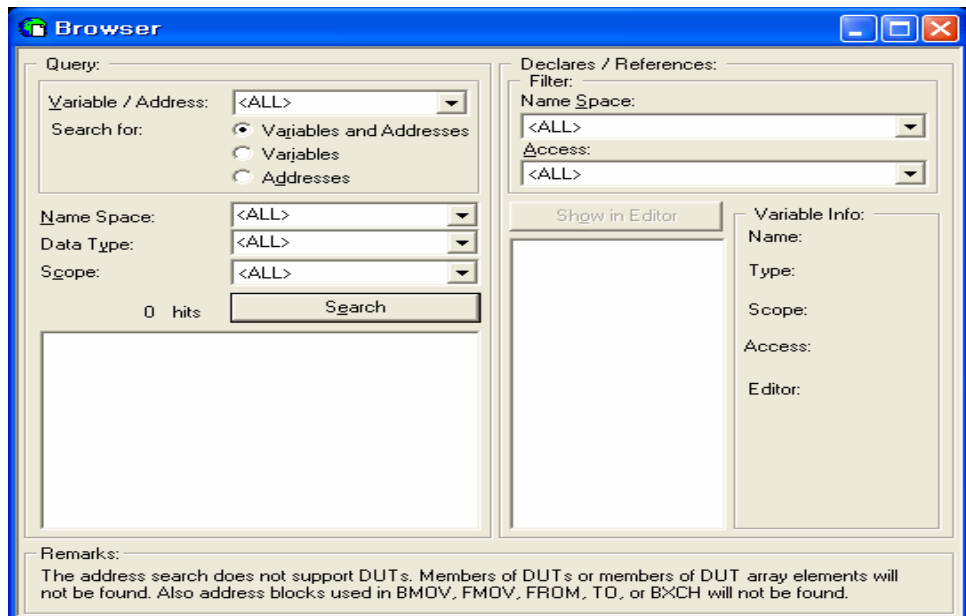


② [Project] - [Make Cross Reference]

: 作成 *.SCT, *.SCX 形式的文件.

③ [Project] - [Browse]

: 在下面画面里确认各项目里的变数.



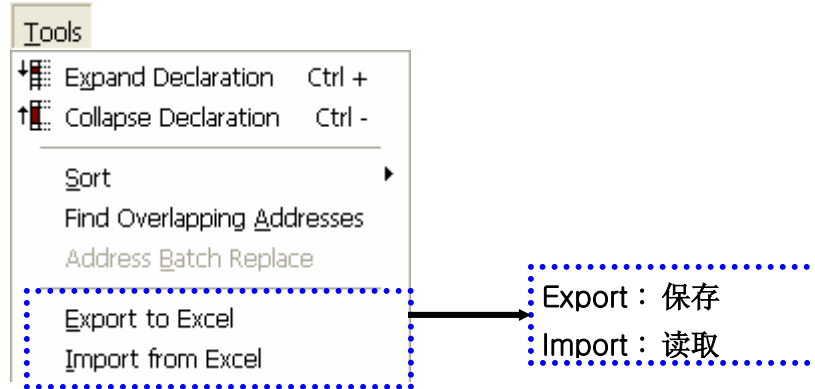


14.7 变数的 Import / Export

： 作成的变数Data以 Excel文件形式来保存及用 Excel文件来制作并可以进行读取。

- 路径-

： 在打开变数窗状态下选择跟下面一样的路径.



- Export的画面 -

① GX IEC Developer 画面

PRG_TEST [PRG] Header				
	Class	Identifier	Type	Initial
0	VAR	CC_LINK_05	PRG_CC_LINK	...
1	VAR	Station_Information	ARRAY [1..64] OF INT	[64(0)]
2	VAR	Reserve_Information	ARRAY [1..4] OF INT	[4(0)]
3	VAR	Invalid_Information	ARRAY [1..4] OF INT	[4(0)]
4	VAR	Send_Receive_Information	ARRAY [1..3,1..26] OF INT	[78(0)]
5	VAR	Setting_Information	INT	0
6	VAR	Connect_Information	INT	0
7	VAR	Headaddress_Information	INT	0

② Excel 画面

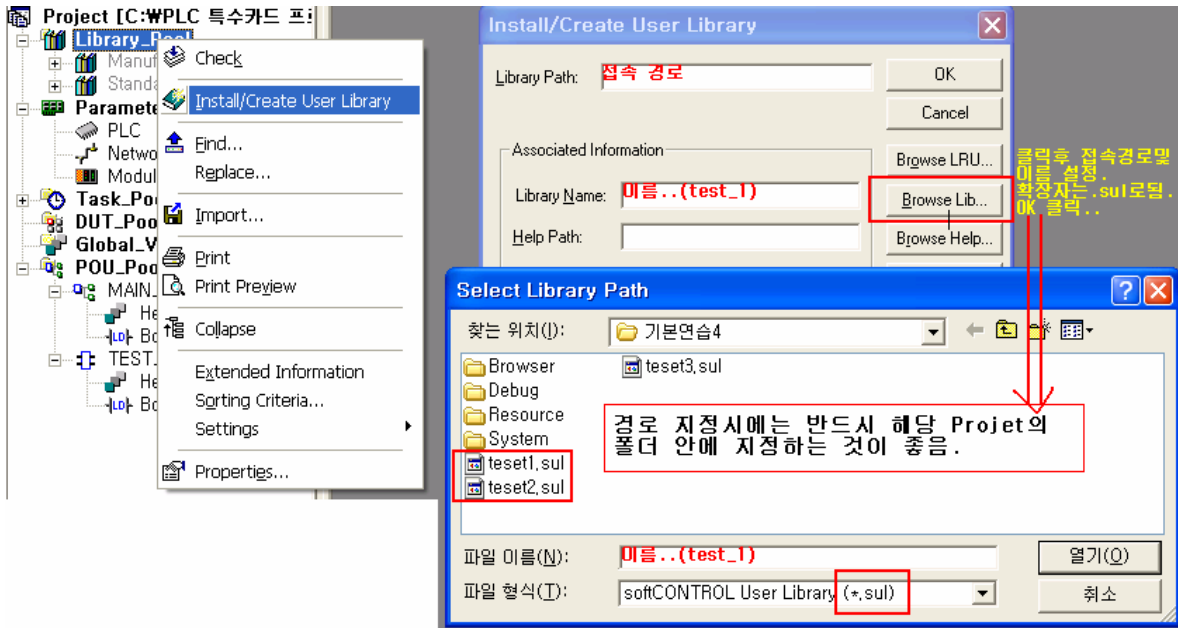
head.xls					
	A	B	C	D	E
1	Class	Identifier	Type	Initial	Comment
2	VAR	CC_LINK_05	PRG_CC_LINK		
3	VAR	Station_Information	ARRAY [1..64] OF INT	[64(0)]	
4	VAR	Reserve_Information	ARRAY [1..4] OF INT	[4(0)]	
5	VAR	Invalid_Information	ARRAY [1..4] OF INT	[4(0)]	
6	VAR	Send_Receive_Information	ARRAY [1..3,1..26] OF INT	[78(0)]	
7	VAR	Setting_Information	INT	0	
8	VAR	Connect_Information	INT	0	
9	VAR	Headaddress_Information	INT	0	



14.8 User Library 登录

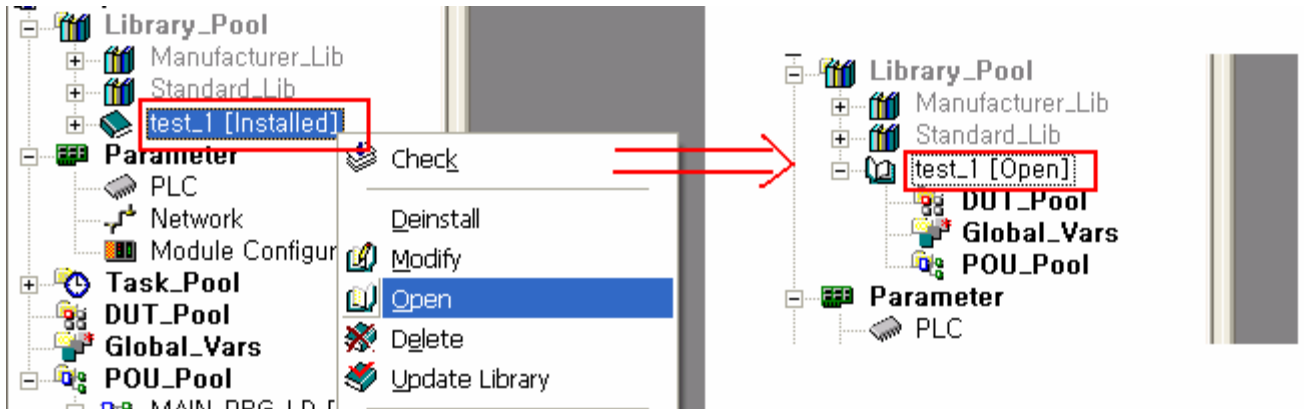
14.8.1 Library的制作

- ① Library_Pool里点击鼠标右键
- ② Install/Create User Library 选择.
- ③ 根据以下方法进行.



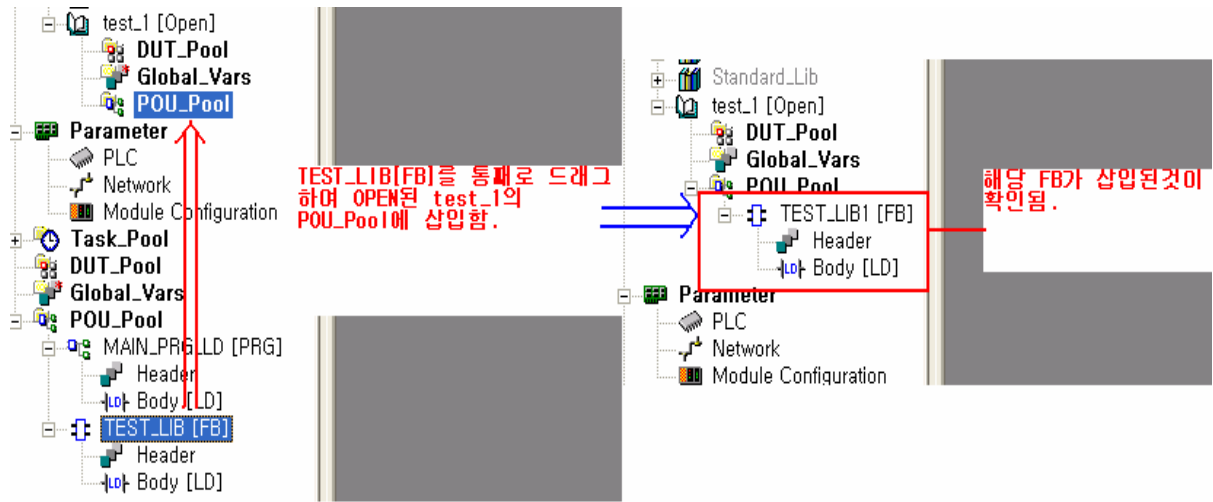
14.8.2 登录Library

- ① 在作成的Library里加入作成的FB
- ② 点击对应Library的右键 => OPEN.
- ※在执行之前必须先把所有窗口进行关闭.



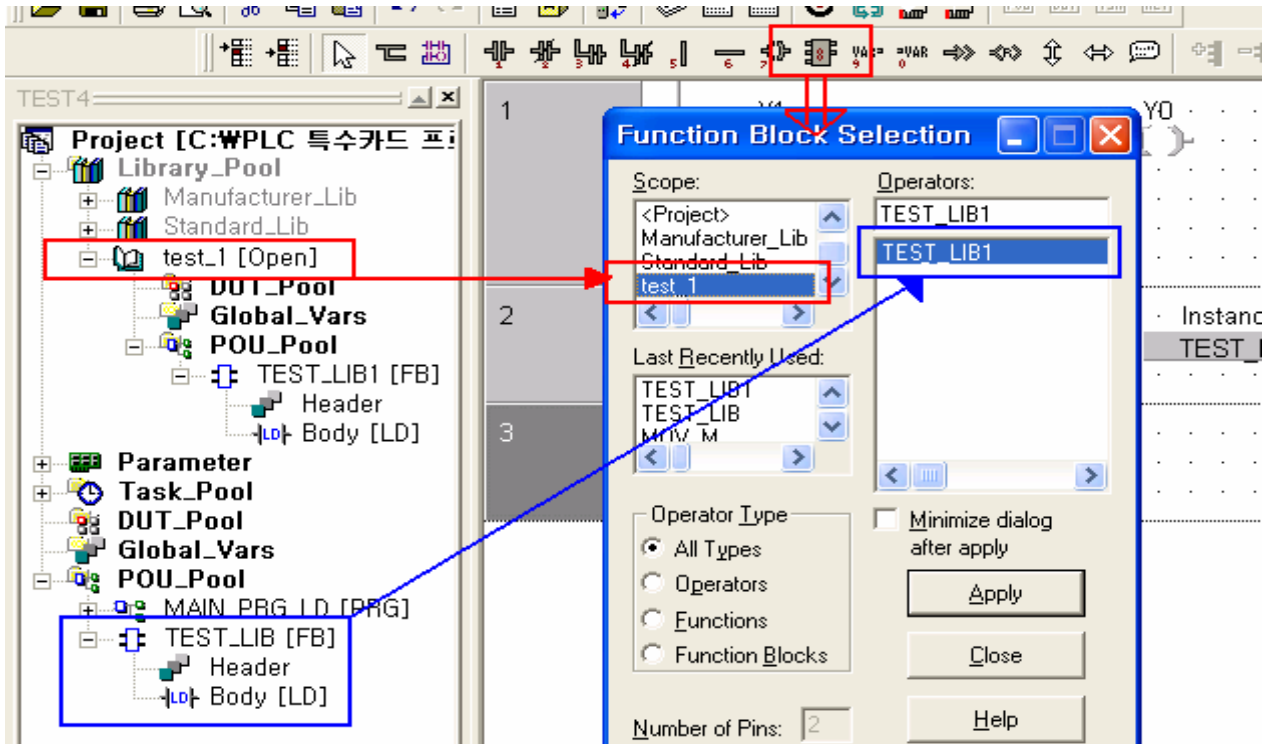


㉔ 把作成的FB的POU_Pool全部选择并插入到文件夹里.



- ㉕ 插入完了后点击鼠标右键后Close
- ㉖ 保存并完了.
- ㉗ 以后再打开并可以进行修整及追加.

※施行FB输出的话把User Library在Project内可以简单施行.

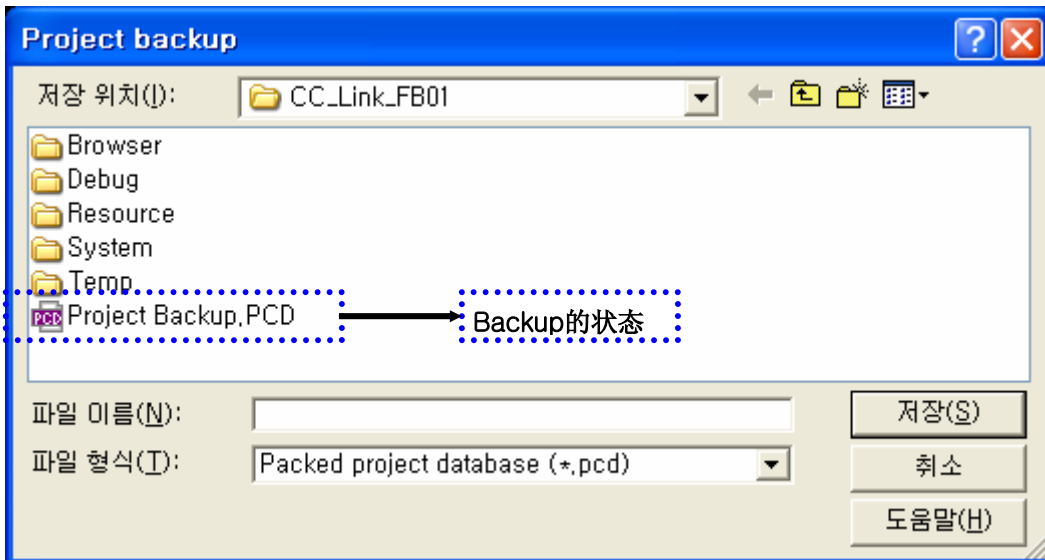
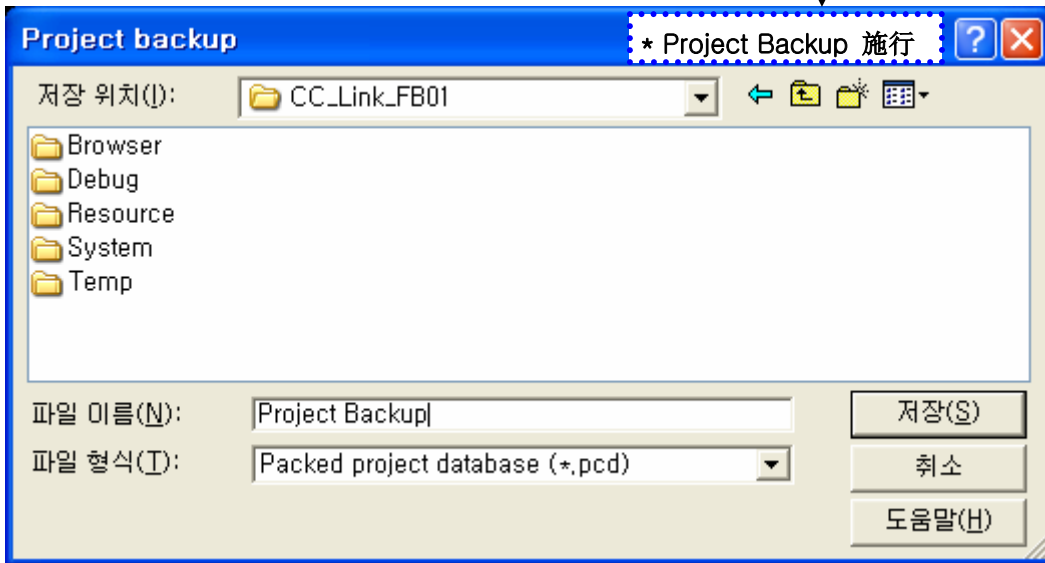
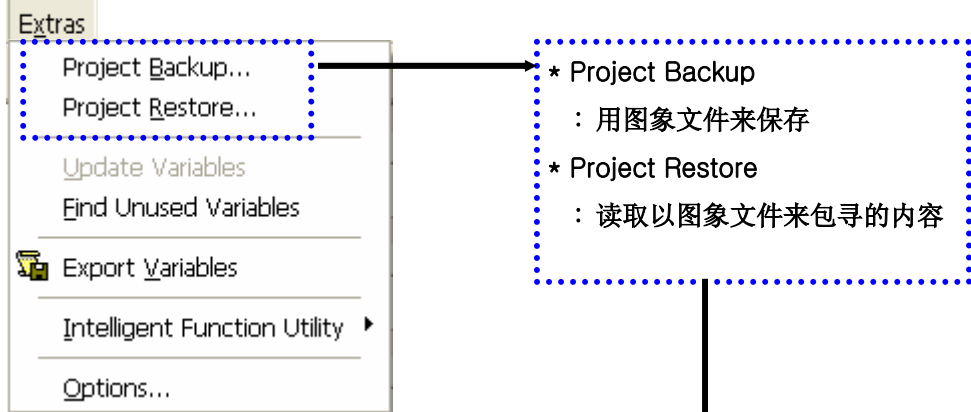




14.9 Project File 管理

: 用GX IEC Developer来作成的程序根据程序结构的特性而在PC里占据大量容量. 为了更好的管理以上的程序而另做成图象文件后使用

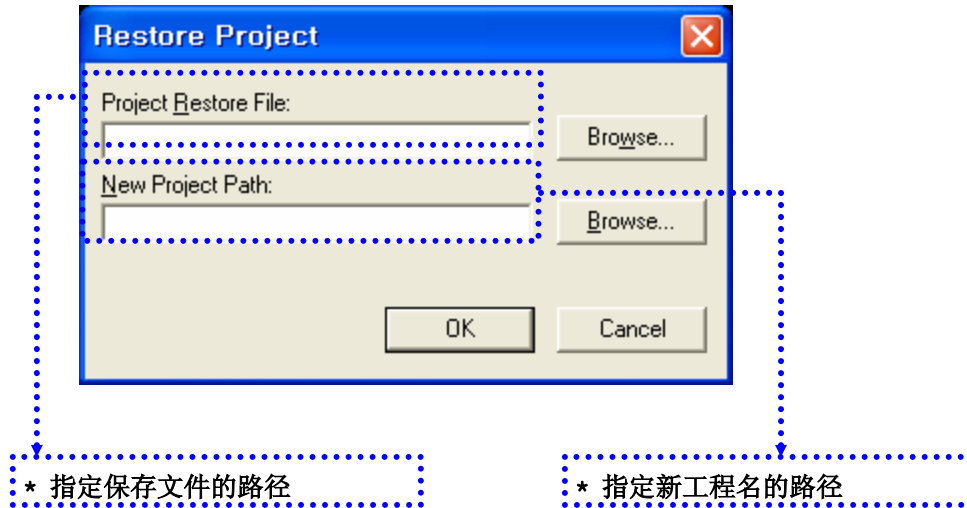
- 使用 -





- Project Restore
: Backup的 Project的读取.

=> 使用

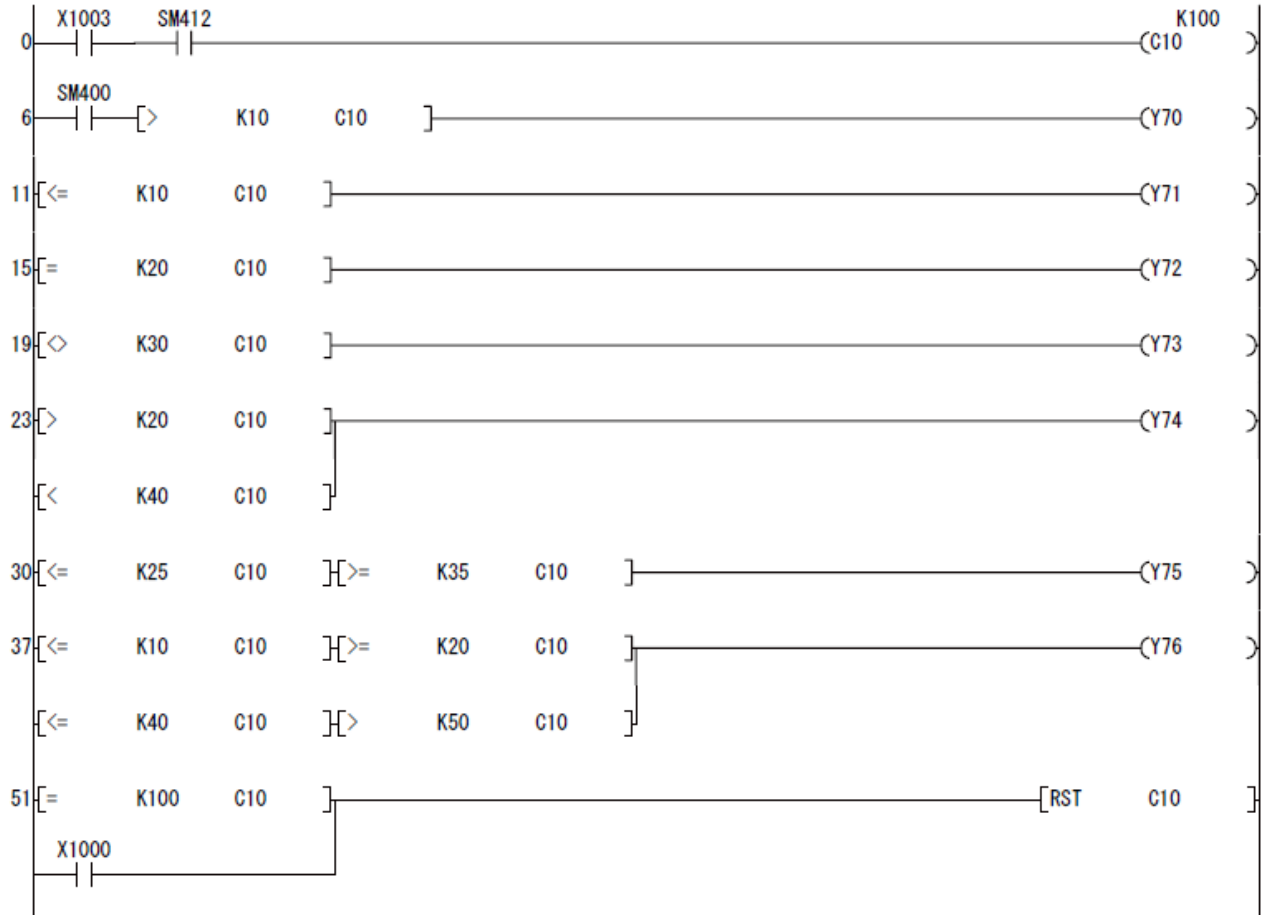




15. GX IEC Developer 使用 - (Sample Program)

15.1 课题程序 1

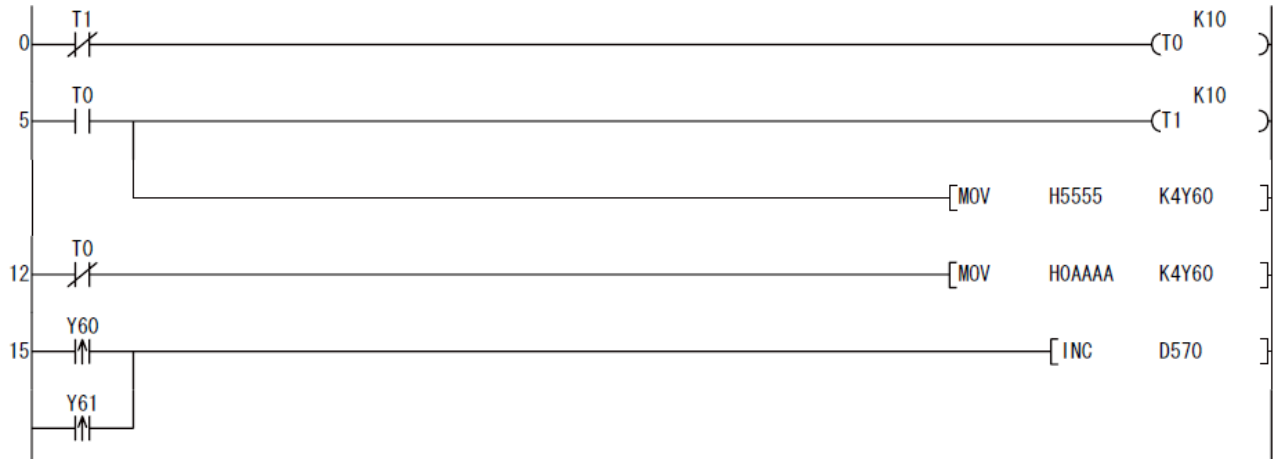
: 把下面的程序用 GX IEC来作成





15.2 课题程序 2

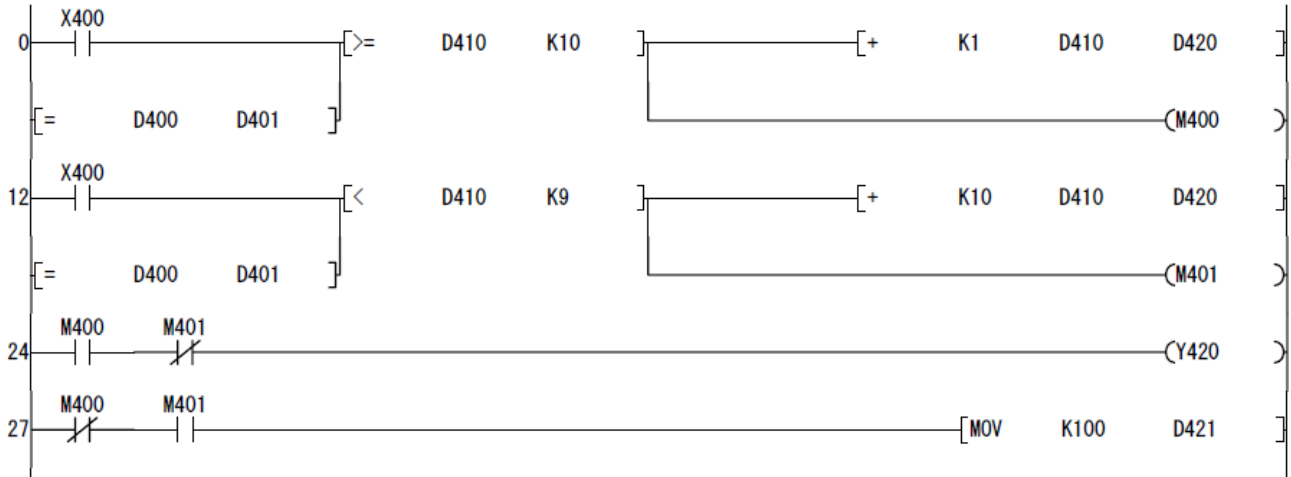
: 把下面的程序用GX IEC来作成





15.3 课题程序 3

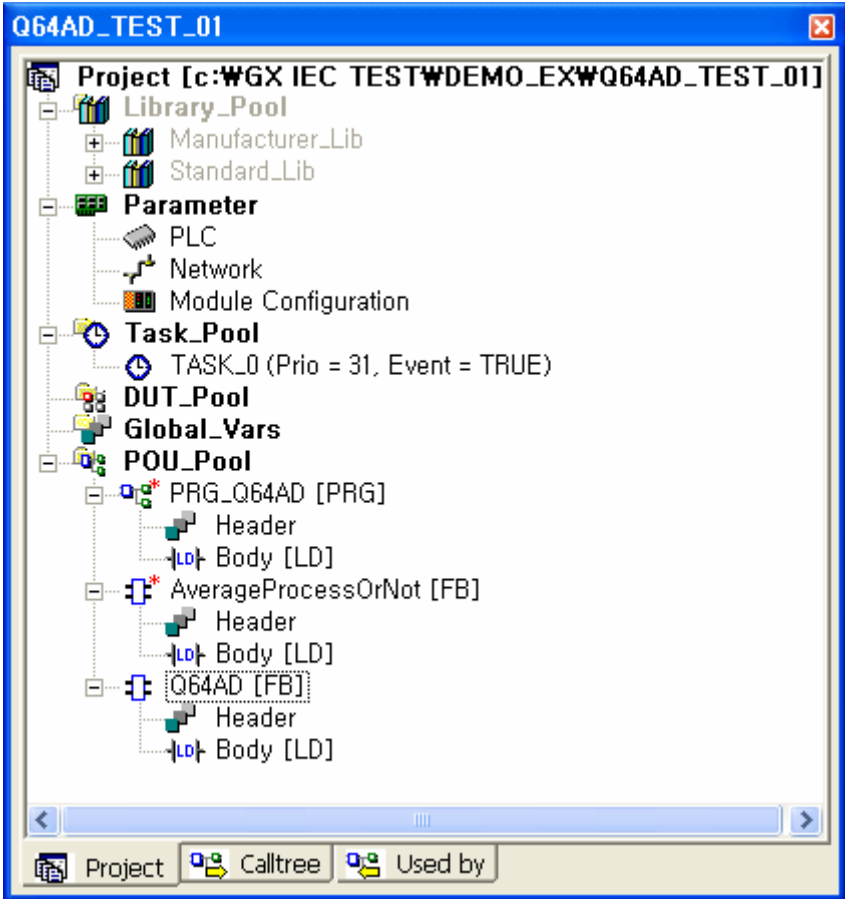
: 把下面的程序用GX IEC来作成





15.4 Q64AD 例题程序

Fig15-1: Sample Program的 Project 构成



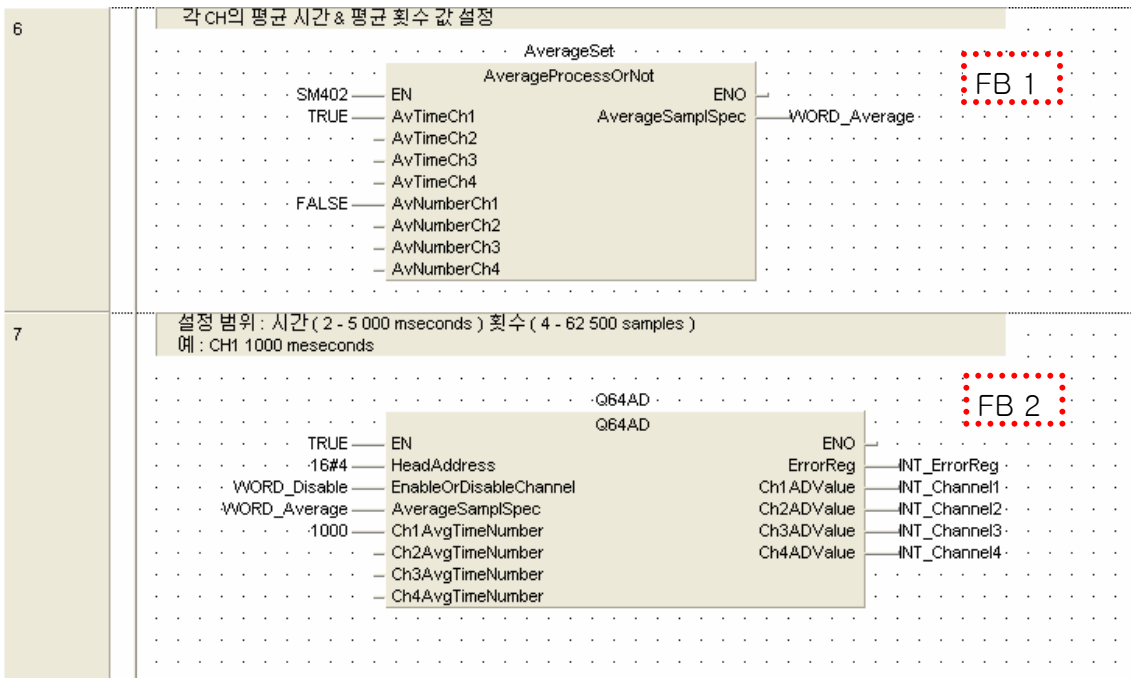
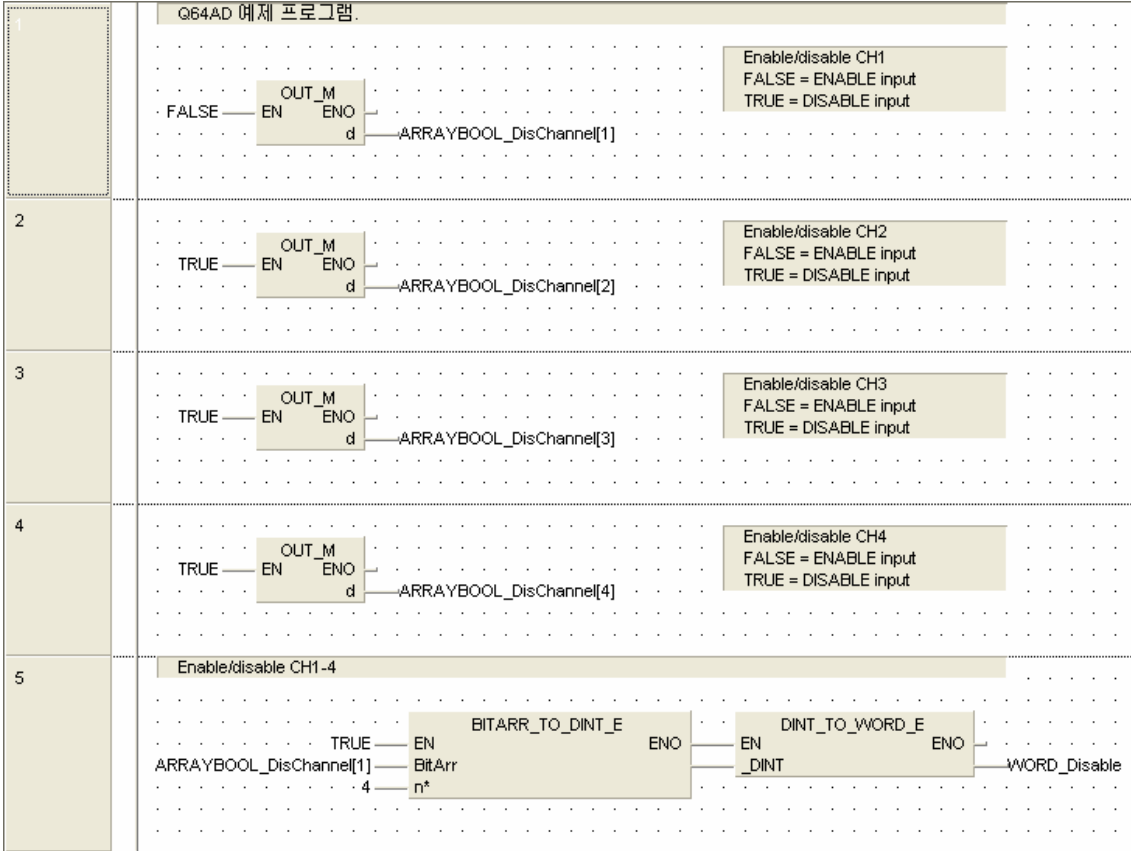
15.4.1 Q64AD的 PRG程序

☞ PRG 程序的Local变数选定

	Class	Identifier	Type	Initial	Comment
0	VAR	INT_ErrorReg	INT	0	
1	VAR	INT_Channel1	INT	0	
2	VAR	INT_Channel2	INT	0	
3	VAR	INT_Channel3	INT	0	
4	VAR	INT_Channel4	INT	0	
5	VAR	ARRAYBOOL_DisChannel	ARRAY [1..8] OF BOOL	[8(FALSE)]	모듈의 Enable/disable 설정
6	VAR	WORD_Disable	WORD	0	
7	VAR	WORD_Average	WORD	0	
8	VAR	AverageSet	AverageProcessOrNot		
9	VAR	Q64AD	Q64AD		



PRG程序的body程序





15.4.2 在PRG里所用的各FB程序

☞ [FB1]是 把Q64AD的 各Channel的平均处理指定的内容用FB来作成的.

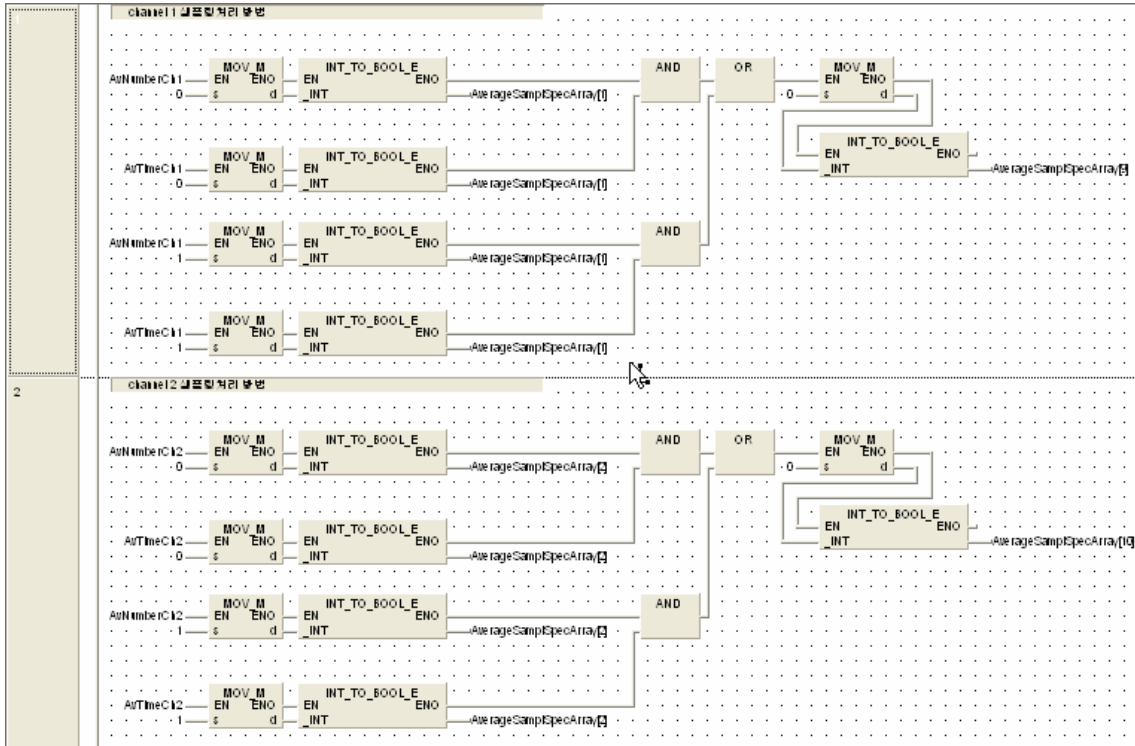
15.4.2.1 [FB1] 程序

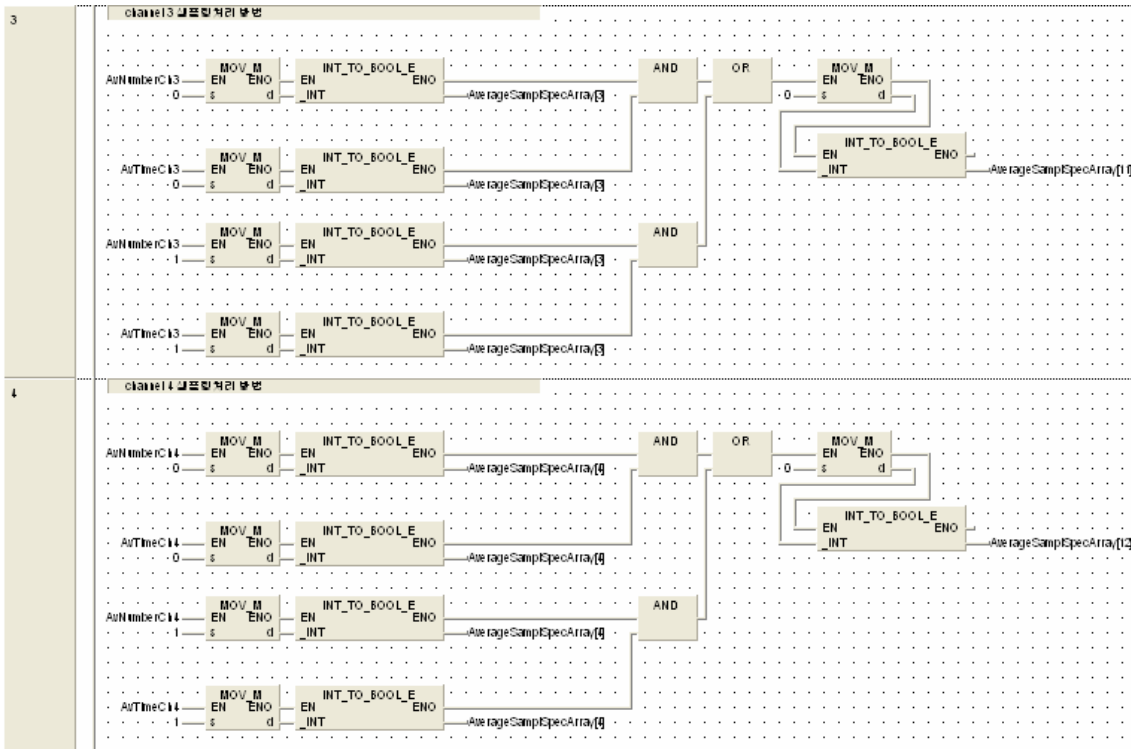
☞ [FB1]的 Local 变数指定

	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	AvTimeCh1	BOOL	... FALSE	
1	VAR_INPUT	AvTimeCh2	BOOL	... FALSE	
2	VAR_INPUT	AvTimeCh3	BOOL	... FALSE	
3	VAR_INPUT	AvTimeCh4	BOOL	... FALSE	
4	VAR_INPUT	AvNumberCh1	BOOL	... FALSE	
5	VAR_INPUT	AvNumberCh2	BOOL	... FALSE	
6	VAR_INPUT	AvNumberCh3	BOOL	... FALSE	
7	VAR_INPUT	AvNumberCh4	BOOL	... FALSE	
8	VAR_OUTPUT	AverageSamplSpec	WORD	... 0	
9	VAR	AverageSamplSpecArray	ARRAY [1..16] OF BOOL	... [16(FALSE)]	

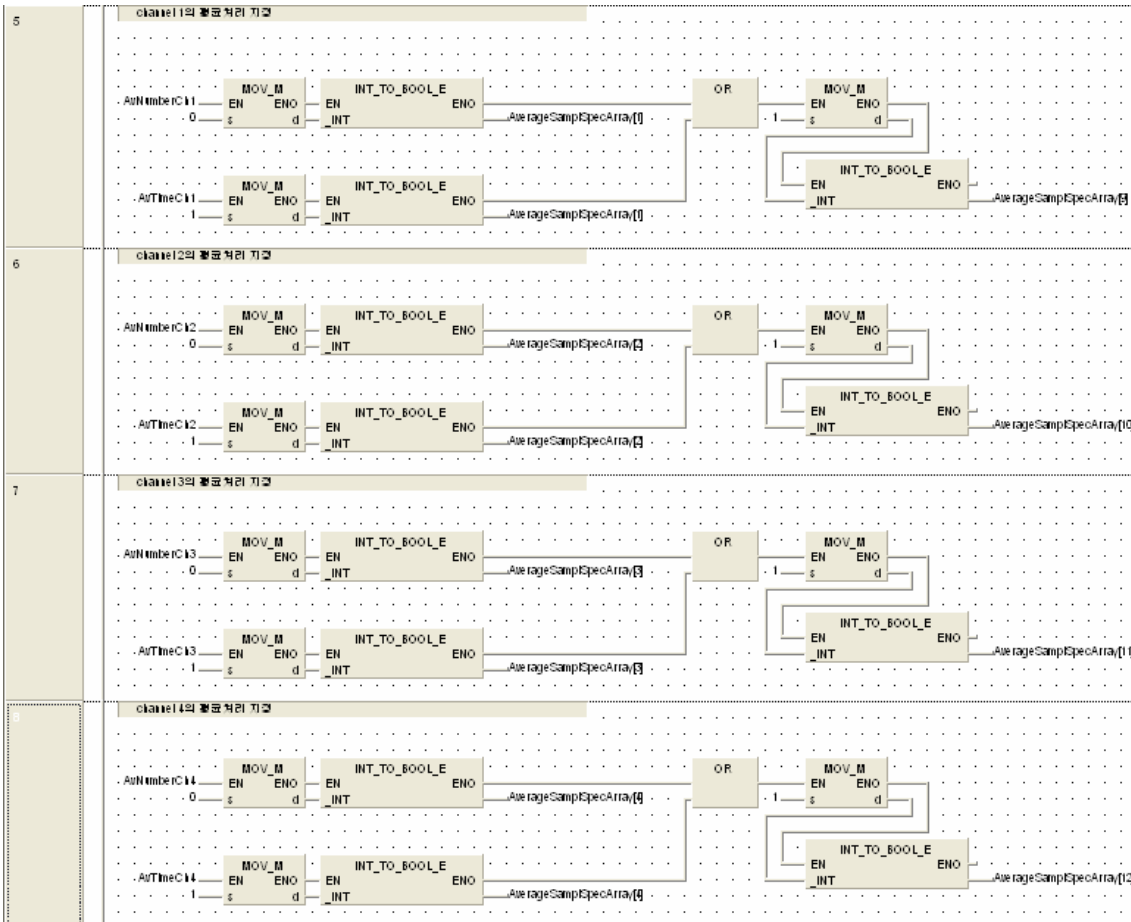
☞ [FB1]的 body程序

① 设定各Channel的平均次数或平均时间



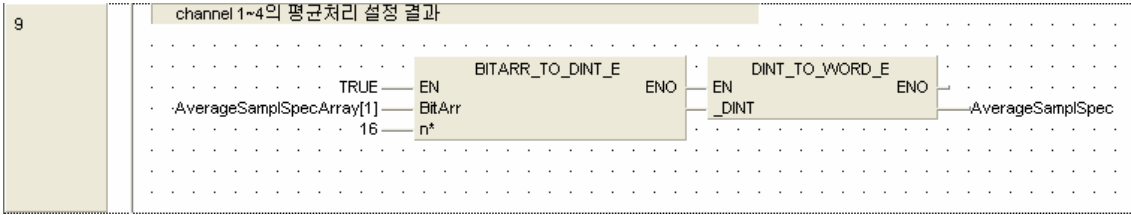


㉔ 指定平均处理Channel





㉔ 把(㉑+㉒)的结果用一个word data形态来作成.





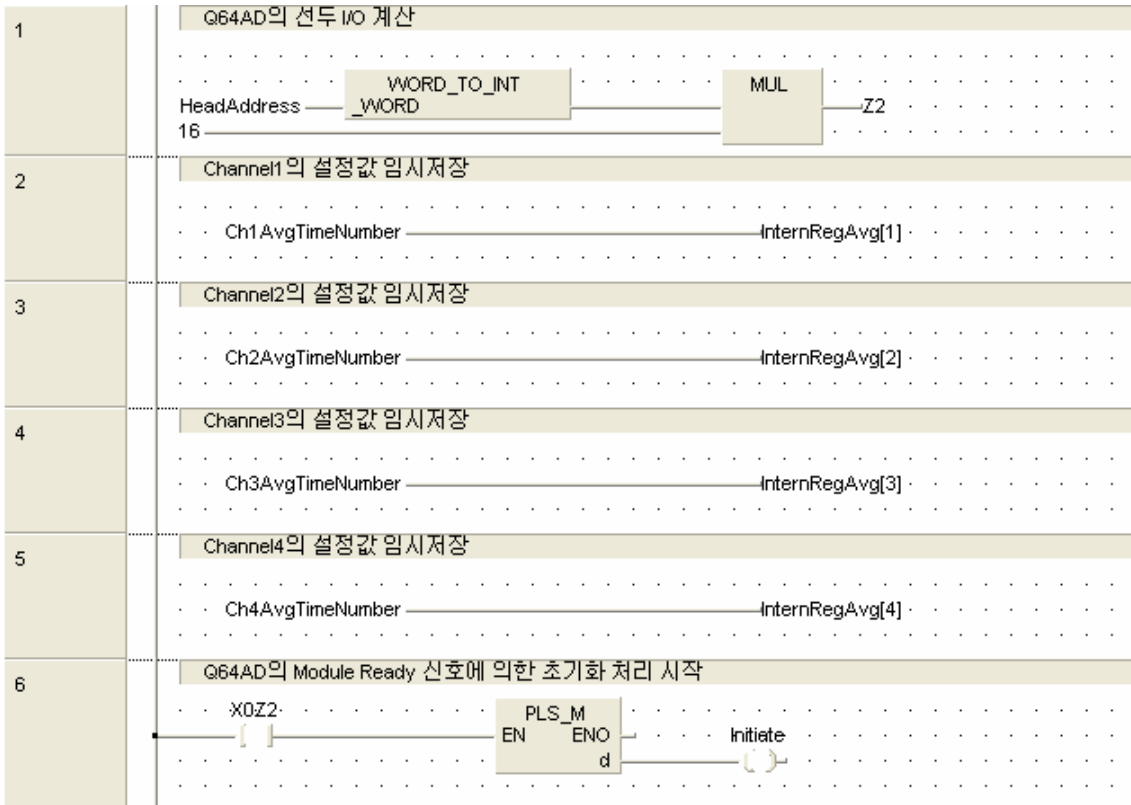
15.4.2.2 [FB2] 程序

[FB2] 的 Local 变数指定

	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	HeadAddress	WORD	...0	
1	VAR_INPUT	EnableOrDisableChan	WORD	...0	
2	VAR_INPUT	AverageSamplSpec	WORD	...0	
3	VAR_INPUT	Ch1AvgTimeNumber	INT	...0	
4	VAR_INPUT	Ch2AvgTimeNumber	INT	...0	
5	VAR_INPUT	Ch3AvgTimeNumber	INT	...0	
6	VAR_INPUT	Ch4AvgTimeNumber	INT	...0	
7	VAR_OUTPUT	ErrorReg	INT	...0	
8	VAR_OUTPUT	Ch1ADValue	INT	...0	
9	VAR_OUTPUT	Ch2ADValue	INT	...0	
10	VAR_OUTPUT	Ch3ADValue	INT	...0	
11	VAR_OUTPUT	Ch4ADValue	INT	...0	
12	VAR	InternReg	ARRAY [1..4] OF INT	...[4(0)]	
13	VAR	InternRegAvg	ARRAY [1..4] OF INT	...[4(0)]	
14	VAR	Initiate	BOOL	...FALSE	

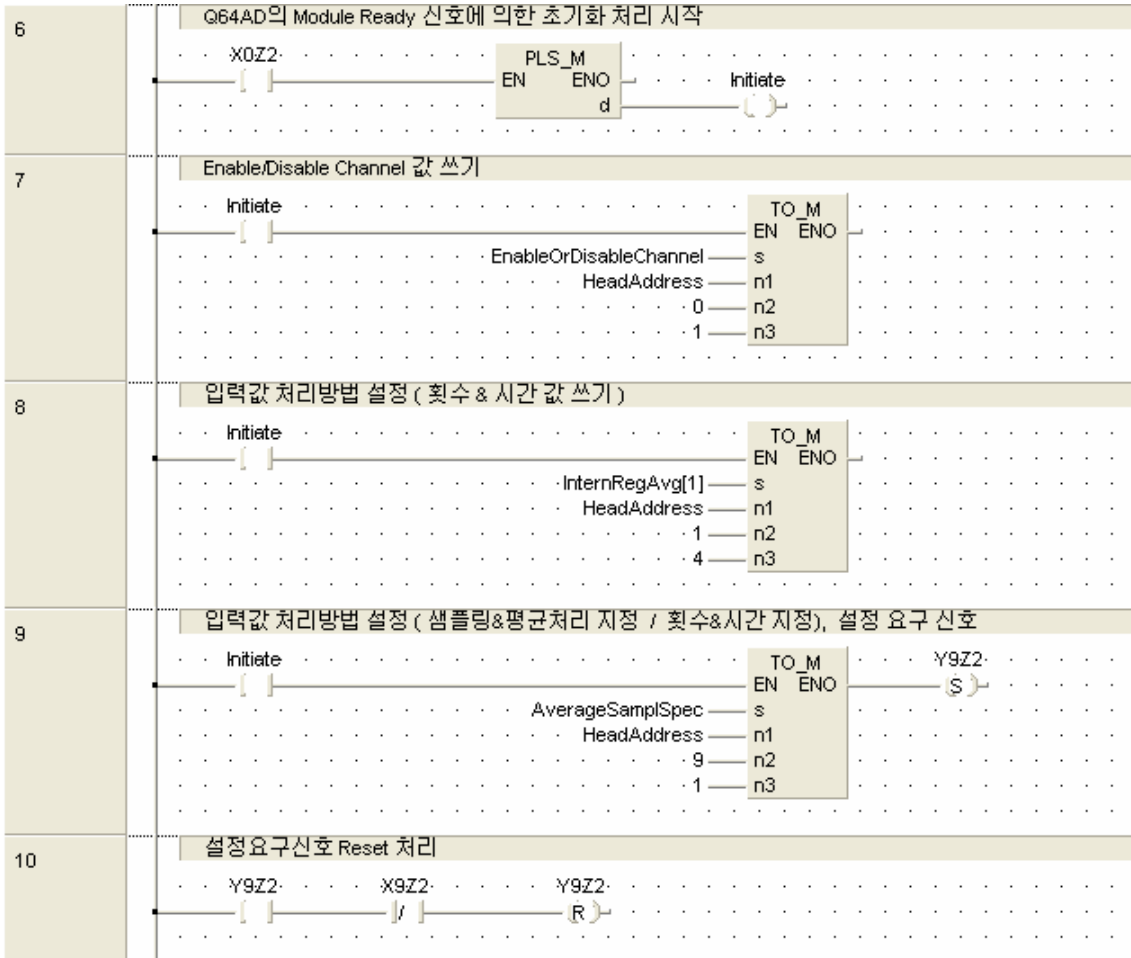
[FB2] 的 body程序

① 地址处理及输入值的处理





㉞ Q64AD의 초기화设定处理程序





㉔ 读取Digital输出值及处理错误编码

