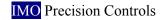
PLC Training Manual



G-Series Basic PLC Training

IMO Precision Controls

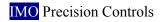
1000 North Circular Rd Staples Corner London NW2 7JP Tel: +44 (0) 208 452 6444 Fax: +44 (0) 208 450 2274 Email: sales@imopc.com or applications@imopc.com Web: www.imopc.com



IMO Precision Controls

1000 North Circular Rd Staples Corner London NW2 7JP Tel: +44 (0) 208 452 6444 Fax: +44 (0) 208 450 2274 Email: sales@imopc.com or applications@imopc.com Web: www.imopc.com

G-SERIES PLC RANGE	1
RANGE OVERVIEW PART NUMBERING	1 6
INSTALLATION OF IMO GMWIN	7
INTRODUCTION TO GMWIN	10
STARTING A NEW GMWIN PROJECT	12
NAVIGATING GMWIN	15
SETTING UP THE PC CONNECTION FOR PROGRAMMING	16
GMWIN MENU'S	18
PROJECT MENU	18
PROGRAM MENU	18
EDIT MENU	19
VIEW MENU	19
COMPILE MENU	19
Online Menu	20
Tools Menu	20
WINDOW MENU	20
HELP MENU	21
PROJECT TREE	22
PARAMETER LIST	22
LIBRARY MENU	23
OUTPUT WINDOW	23
ADDRESSING IN A G-SERIES PLC	24
MEMORY ALLOCATION IN G-SERIES PLC	26
BITS AND BYTES	27
LADDER PROGRAMMING RULES	28
VARIABLE DECLARATION	31
TIME VARIABLES	33
BASIC LADDER LOGIC	34
INPUTS AND OUTPUTS	34
INPUTS	34
OUTPUTS	35



BOOLEAN LOGIC	35
LATCHING CIRCUITS	36
FLIP / FLOP CIRCUIT	36
LADDER PROGRAMMING EXAMPLE	37
BASIC FUNCTIONS	41
MONE EUROPION	42
MOVE FUNCTION	
ARITHMETIC FUNCTIONS ADD FUNCTION	42
	43
SUB FUNCTION	43 44
MUL FUNCTION	
DIV AND MOD FUNCTION	45
BIT FUNCTIONS	46
AND, OR, XOR FUNCTIONS	46
SHIFT AND ROTATE FUNCTIONS	48
COMPARE FUNCTIONS	48
LT, EQ, AND GT FUNCTIONS	49
FUNCTION PROGRAMMING EXAMPLE	51
BASIC FUNCTION BLOCKS	61
TIMER FUNCTION BLOCKS	62
TOF, OFF DELAY TIMER.	63
TON, ON DELAY TIMER.	63
TP, TIMER PULSE.	64
COUNTER FUNCTION BLOCK	65
CTU, UP COUNTER.	65
CTD, DOWN COUNTER	66
CTUD, UP / DOWN COUNTER.	67
FUNCTION BLOCK PROGRAMMING EXAMPLE	68
MULTIPLE-SOURCE PROGRAMMING	74
CLOBER VERSER	74
GLOBAL VARIABLES	74
MULTIPLE LD PROGRAMS	76
SUBROUTINES	78
DOWNLOADING AND ONLINE OPTIONS	80
COMPILING AND BUILDING	80
GMWIN SIMULATION	81
DOWNLOADING	83
BASIC COMMUNICATION	86
G7 PLC COMMUNICATION DIP SWITCH SETTING	88
O / I DC COMMUNICATION DI SWITCH SETTING	00

PROGRAMMING TUTORIAL EXAMPLE – WASHING MACHINE 90

Specification	90
MACHINE I / O	90
WASHING MACHINE CYCLES	90
THE PROGRAM CYCLE	91
BEGINNING TO PROGRAM	91
1. WHITE WASH OR COLOUR WASH IS SELECTED BY A ROTARY SWITCH.	94
2. When the Lid is closed and the start button is pressed, Detergent + $W_{\rm A}$	ATER
ARE PUMPED IN FOR 1 MINUTE	97
3. THE DRUM ROTATES FOR THE SPECIFIED WASH TIME, DEPENDING ON THE TYPE OF	
WASH SELECTED.	99
4. The drum is drained for 1 minute.	100
5. Fresh water is poured in for 1 minute.	100
6. THE DRUM IS ROTATED FOR THE SPECIFIED RINSE TIME, DEPENDING ON THE TYPE	OF
WASH SELECTED.	101
7. THE WATER IS DRAINED FOR 1 MINUTE.	101
8. The drum is rotated for the specified spin time, depending on the type o	F
WASH SELECTED.	102
9. THE LID OPEN LAMP WILL OPERATE IF THE LID IS OPENED, AND THE WASH SEQUEN	ICE
CANNOT START.	102
10. IF THE LID IS OPENED AT ANY TIME DURING THE WASH CYCLE, THE CYCLE IS	
SUSPENDED AND ONLY CONTINUES WHEN THE LID IS CLOSED AGAIN.	103
COMPILE AND BUILD THE PROJECT	105
SIMULATE THE PROGRAM	105
DOWNLOAD THE PROGRAM TO THE PLC	106
	100
TUTORIAL 1 – IMO GREENHOUSE	109
TUTORIAL 2 – IMO CAR PARK	<u>110</u>
TOPICS COVERED IN THE ADVANCED COURSE	111

IMO Precision Controls

1000 North Circular Rd Staples Corner London NW2 7JP Tel: +44 (0) 208 452 6444 Fax: +44 (0) 208 450 2274 Email: sales@imopc.com or applications@imopc.com Web: www.imopc.com

G-Series PLC Range

IMO's G-series PLC range covers the complete scope of automation potential. Within the IMO PLC range there are three levels. What determines these levels generally is the users requirements for the application such amount of I/O required. The IMO range begins with the expandable brick type PLC, the G7 series. To the compact, slot and rack in the mid-range, the G6 series, through to the high-end G4 series. From the 10 I/O brick style G7 to the 1024 I/O Multi-rack and slot G4 PLC, IMO have got it covered!

The complete G-Series range is programmed through the IEC61131-3 compliant software. This is a windows based program allowing the user to program in Ladder diagram (LD), Sequential Function Chart (SFC) and Instruction List (IL).

	G7	G6	G4
Power Supply	110/230Vac or 24Vdc (E-marked for vehicle low voltage (12V) operation)	110/230Vac or 24Vdc (E-marked for vehicle low voltage (12V) operation)	110 Vac or 230Vac or 24Vdc
I/O	Up to 80	Up to 384	Up to 1024
Analogues	Two analogue modules: 2 in, 1 out. Or 4 in. Voltage or Current selectable.	G6F-AD2A 4 analogue inputs. Selectable, 4-20mA, 0-20mA or 0-10V G6F-DA2V voltage input G6FDA2I current input	G4F-AD2A 4 analogue inputs. Selectable, 4-20mA, 0-20mA or 0-10V G4F-DA2V voltage input G4FDA2I current input
Comm. options	2 inbuilt RS232 user configurable ports. RS232 and RS485 options. Modbus, Profibus (slave), DeviceNet (slave) FNet, RNet (IMO comms). User defined comms.	RS232 and RS485 options. Modbus, Profibus, DeviceNet, FNet, RNet (IMO comms). Ethernet. User defined comms.	RS232 and RS485 options. Modbus, Profibus, DeviceNet, FNet, RNet (IMO comms). Ethernet, User defined comms.
Expansion	Up to 3 maximum but only 2 of the 1 kind. I.e. 2 analogue modules + 1 relay module.	Maximum rack size of 12 slots. Non-expandable.	Maximum rack size of 8. 3 expansion racks of 8. Totalling 32 slots.
Additional features	HSC, RTC module, 7k steps, addition memory module & built in PID.	17k steps. Dedicated MPU 0.5μs. 3-axis open loop position control module. 3 options of CPU.	32k steps. Dedicated MPU 0.2µs. Floating-point maths & USB programming port option. PID module with 8 separate PID loops.
Programming Software	GMWin. IEC61131-3 (LD/IL/SFC)	GMWin. IEC61131-3 (LD/IL/SFC)	GMWin. IEC61131-3 (LD/IL/SFC)

Range Overview

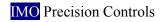
IMO Precision Controls

- All of the IMO G-series range uses the same programming software, the IEC61131-3 compliant: **GMWin**.
- All the PLC's can be interlinked and share data within their own dedicated network.
- The slot and rack PLC's are modular and very variable. They can be specified to meet the application more precisely.
- The I/O count doesn't include the remote I/O units. A maximum network size of 64 stations, with each possibly having 32 points. (I/O = 2048)



The IMO G7 is a very compact block style programmable controller offering extremely high performance. It is ideal for all applications from process control to machine control

- Up to 80 I/O
- High speed processing 0.5µsec per step
- Memory capacity 68K bytes
- Conforms to elements of the IEC61131-3 (IL/LD/SFC)
- Built in PID, interrupt, pulse count and input filters
- High speed counter input 16kHz, 8kHz two phase
- RTC option and memory module options
- Fieldbus options FNet (master/slave), DeviceNet (slave) and Profibus (slave)
- Two built in RS232 serial ports one dedicated the other user configurable
- Two analogue modules: 2 / 1 analogue i/o or 4 analogue inputs
- Analogue potentiometer module
- 10 I/O expansion module
- RS232 and RS485 option modules
- Uses the same programming software as G6 & G4 (GM-WIN)
- Conforms to CE and UL







The IMO G6 is a compact modular programmable controller offering extremely high performance. It is ideal for all applications from process control to machine control

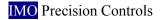
- Up to 384 I/O
- High speed processing with dedicated MPU 0.5µsec per step
- Memory capacity 17Ksteps
- Conforms to elements of the IEC1131-3 standard (IL/LD/SFC)
- PID control, computer link, high speed counter and RTC CPU options are available
- Fieldbus option 1Mbps (FNet), DeviceNet, and Profibus options
- Computer link module for linking up to 32 PLCs to a computer, or to other devices by easily configuring user protocols
- Wide range of digital and analogue I/O
- High speed counter module
- Conforms to CE requirements





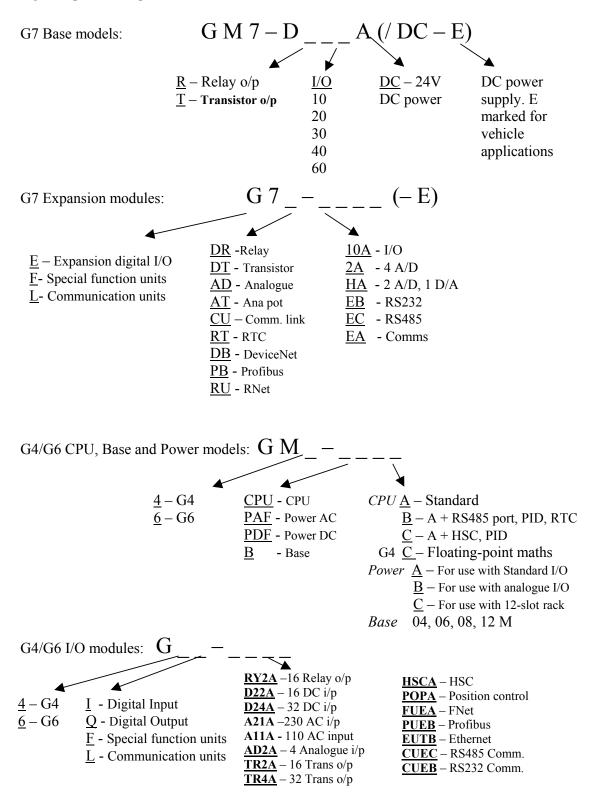
The IMO G4 is a modular programmable controller offering extremely high performance. It is ideal for all applications from process control to machine control.

- Up to 1024 I/O
- Very high speed processing with custom gate array 0.2µsec per step
- Memory capacity 32K steps
- Conforms to elements of the IEC1131-3 standard (IL/LD/SFC)
- Many special function modules PID, Analogue timer, High Speed
- Counter, Positioning, Interrupt and A/D and D/A modules
- Wide range of communication options ENet (Ethernet), FNet Fieldbus),
- Cnet (Computer Link), DNet (DeviceNet) and remote I/O
- Conforms to CE requirements



Part Numbering

At IMO we use an understandable part number where each element has a meaning and is reused through out the part numbering system, thus making it easy to put together product requirements

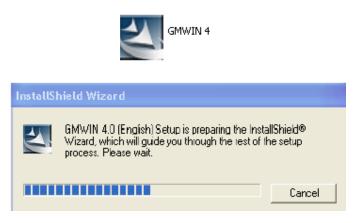


Plus many more options

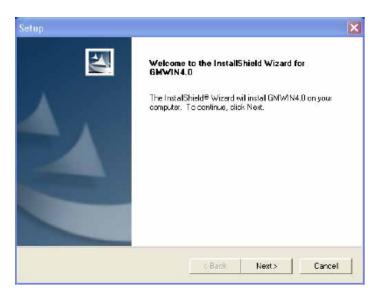
Installation of IMO GMWin

GMWin can be obtained on the IMO PLC Software CD or can be downloaded from the website: <u>www.imopc.com</u> free of charge.

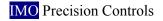
To install double click on the icon GMWIN4.exe and follow the instructions.



A dialogue box showing the welcome message appears. It is required to exit any other windows application programs that may be running during the installation of GMWIN.



Click Next to move to the next screen.



hoose Destination Location		and the second
Select folder where Setup will install files.		
Setup willinstal GMwIN4.Din the following	g folder.	
To install to this folder, click Next. To instal another folder.	l to a different folder, click	Browse and select
Destination Folder		
Destination Folder		Browse
		Browse

Select the destination folder for GMWin to be installed into. If **[Browse]** is clicked then a dialogue box will appear for the path to be inputted. Select the path to install to or manually write the path in the box and click **[OK]**.

Choose Folder	×
Please choose the installation folder. Path	
CAGMWIN 4	
Directories:	
Gravin (C.) Gravin (C.) Gravin Gravin	
OK	Dancel

After selecting the path to install, click [Next].

up			2
Choose Destination Location			And Street of Contractory
Select folder where Setup will install files		1	-
Setup will install GMWIN4.D in the follow	ing folder.		
To install to this folder, click Next. To ins another folder.	stall to a different folde	r, olick Browse an	d select
- Devination Folder			
Destination Folder			Bjøwse

After confirming the path to install into, click [Next] to continue.

GMWin will then begin to install on your computer, in the directory specified.

Installing:
54%
Cancel

When the install is completed, it is required to restart your computer. You have the option to restart your computer now or later from the following screen.

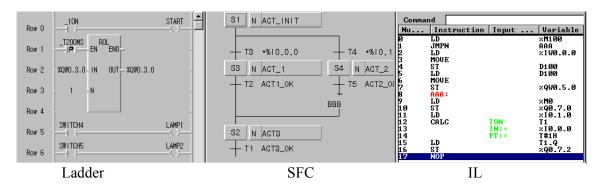
2	InstallShield Wizard Complete
	The Instal Shield Wizard has successfully installed GMWIN4.0. Before you can use the program, you must restart your computer.
	Yes, I want to restart my computer now.
	C No, I wil restart my computer later,
	Remove any daks from their drives, and then click Finish to complete setup.
-	

Now that the installation is complete and after you have restarted your computer, click on the GMWin icon to start programming.



Introduction to GMWin

GMWIN is a programming and debugging tool for the full range of G-Series PLC's (G7, G6 and G4). It is an IEC61131-3 programming compliant environment and with three different programming languages, IL, SFC and Ladder.



Programs can be created with clear symbols for easy understanding and easy to follow menus to create functions and variables alike.

LD 🔀	Add/Edit Variables
	Variable Name : SWITCH4 OK
K []	Variable Kind Cancel
-1 F1/F	Variable Kind : VAR Help
┤Pト ┤Nト	Data Type Memory Allocation
20	Elementary: BOOL Auto
() ()	C FB Instance : BOOL C Assign (AT) : WORD
(S) (R)	C Array (0,) OF DWORD
(P) (N)	Initial Value INT
	LINT Init. Array
{F} { FB}	
(RET) 📎	Comments
(sc)	
~~~	

GMWin complies with the standardising of expressing direct variables by I, Q, M (Input, Output, Memory). The allocation of program variable memory is carried out automatically or by the user designation. Various data type and kinds can be selected to match the variables requirements. It is also possible to set the initial value, and add documentation to every function, function block, constant and variable.

The IMO G-Series is a Project based PLC system. In GMWin it is possible to create several different dedicated programs in different languages to perform specific routines and have them all related in a Project. As well as Programs there is the option of Tasks, which can be time or event driven.

🔁 c:\gmwin\source\enginelines.prj	
PROJECT ==> PLC Type : GM1 Uriter : Tom Smith	
CONFIGURATION(PLC) ==> Configuration Name : UNNAMED	
CONFIGURATION GLOBALS ==> 3 variables declared	
ACCESS VARIABLES ==> 2 variables declared	
RESOURCE(CPU) 0 ==> Name : RESO	
RESOURCE GLOBALS ==> 0 variables declared	
TASK DEFINITIONS ==> 3 tasks defined	
LD ] PROGRAM ==> WELDING : c:\gmwin\source\robot.src [ IL ] PROGRAM ==> INST1 : c:\gmwin\source\robot2.src	
Survey in State ( In State -> INST1 : c:\gmwin\source\robot2.src	
LD   PROGRAM ==> INST3 with task _H_INIT : c:\gmwin\source\robot4.src	
COMMENTS for DIRECT VARIABLES ==> 6 variables declared	
PARAMETERS	
BASIC PARAMETERS	
I/O PARAMETERS	
Le LINK PARAMETERS	
Included Libraries	
- C:\gmwin\lib\communi.lfb	
- c:\gmwin\lib\remote3.lfb	
- c:\gmwin\lib\remote4.lfb	
└≣c:\gmwin\lib\special.lfb	

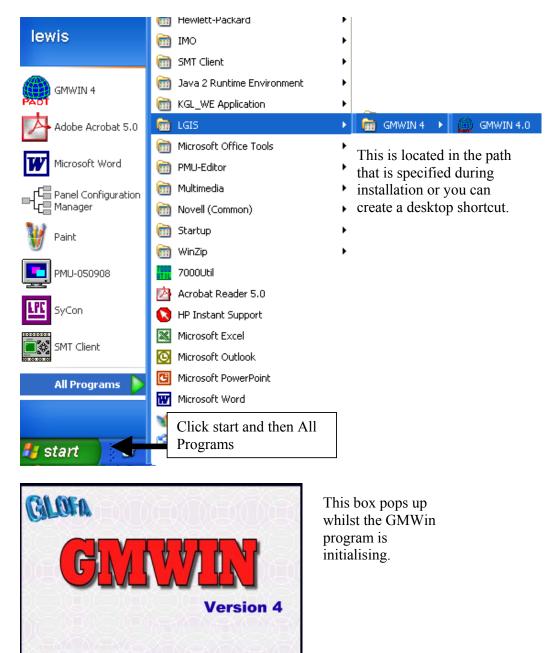
A project tree is shown above that details all libraries, tasks and programs included in a project. Project based programs make duplication and re-use of programs quick and easy.

It is possible to not only download and monitor programs directly connected to a PC but also for PLC's connected in a network. When connected with a PLC it is possible to monitor the operation, change the mode and edit online. Online editing, allows the PLC to remain running and the program can be changed. The full PLC mode, error state and parameters can also be read when connected.

It is also possible to create User-Defined Functions and Function Blocks. This allows for frequently used bits of code being made into an easy to use and re-useable graphical function block. There is also the utility to create User-Defined Communication Protocols, thus allowing obscure and bespoke devices to be communicated with.

## Starting a new GMWin Project

To start a new program click on the GMWin icon.



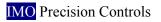
GMWin will appear, with the last project that you were working on, if you did not close the program down. At the moment GMWin will have no project and look rather empty, as this is the first run of the software on your computer.

To begin a New Project, either click the New Project icon or select the option from the drop down Project menu.

GMWIN			
Project Program Edit View Compile Online Debug Tools Window Help			
New Project	XANSSII	<b>ti :</b>	
🚰 Open	a ∧ 991 799 89 89 →11 [3]		
Upload Project From PLC			
Save Save			
Save <u>A</u> s			
Glose	-		
Import Project Bundle			
Export Project Bundle			
Add Item			
M Area Edit			
Pre <u>vi</u> ew			
Print Project			
Print Program Ctrl+P			
Printer Setup			
Option			
1 c:\gmwin 4\source\g7-ana-ad2a\g7-ana-ad2a.prj			
2 c:\gmwin 4\source\junair_51\junair_51.prj			
3 c:\gmwin 4\source\modbus_rtu_master_2\modbus_rtu_master_2.prj			
4 c:\gmwin 4\source\junair_51_buses\junair_51_buses.prj			
Exit			
••• Pr •• Par 22 Lib			
Error/Warning & Cross Reference & NO & Dup	icate Coil λ Find λ Commu	inication /	
	ware con Viring Viconning	Offline	Edit
Create new project.		OTTIME	Edit

In this first menu we can enter the project details and where the project folder is going to be created. We must select the PLC we are going to use and we have some documentation options.

New Project						
Enter project	file name :	first_plc_p	roject	_		
Location :		C:\GMWIN	4\Source\firs	Browse		
DI 0/0		J				
PLC(Configu		-				
		ie in basic p	arameter sei	ting page at	fter project creat	ied. j
- Select PLC	C type ——					
C GR	C G1	C G2	O G3			
⊂ G4	C G4B	C G4C	C G6	🖲 G7	C G7U	
Writer :						
lewis						
Comment :						
Project doo	umentation					-
					_	(
			< Back	Next >	Cancel	Help



Define Program
Enter program file name win 4\source\tfirst_plc_project\ <mark>program_name</mark> src Browse
Enter program instance name INST0 [* Instance is identifier in program memory]
Scan program
C TASK Browse
<back next=""> Cancel Help</back>

The next stage is to define the first program. This can be edited later and even removed but initially this must be set up.

The program will initially be called noname.src, but to modify this, simply highlight the area shown and enter a name that has more meaning to your application.

inter project file name		program_	name.src	
Select language				
⊙SFC ⊙LD	СL	C FBD	C ST	
Select kind of program -				
Program Block	Function Block	🔿 Functi		
			_	
Fun/FB r	name :			
Output d	ata type :		~	
	ered at program.	block. j		
* First program is regist				
* First program is regist Enter program commer				
Enter program commer				
Enter program commer				
Enter program commer				
Enter program commer				

Next we have to select what language we are going to write the program in. The default is ladder.

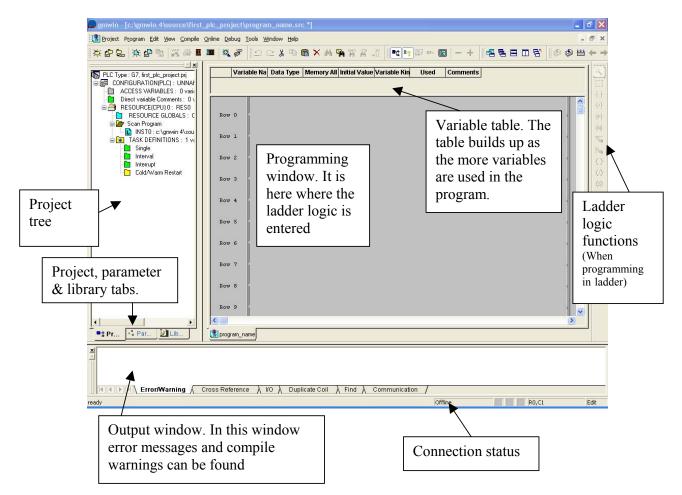
SFC, LD and IL will be discussed in more detail later.

There is also an opportunity to add further documentation notes.

Clicking **[Finish]** will complete the set up process and return you to the main Window with a fuller Project to begin programming.

## **Navigating GMWin**

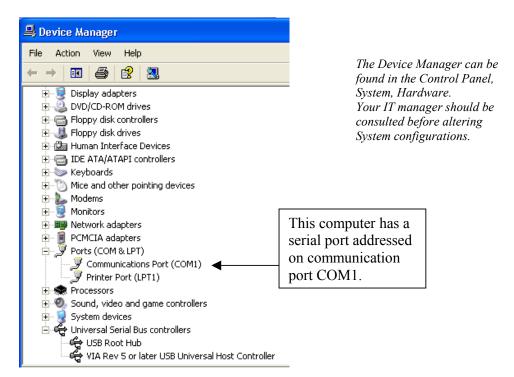
GMWin is a "windows" based programming environment and it can be daunting at first with the amount of information on offer. The programming environment can be customised to your personal preference, with windows an optional buttons being selectable.



#### Setting up the PC connection for programming

The most important thing to do first is to ensure that the correct communication port is selected in GMWin to connect with a PLC. This is particularly important if you are using a USB to Serial converter.

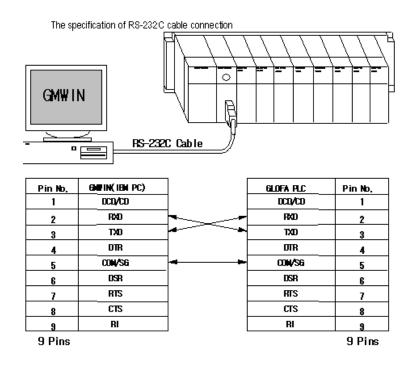
It is possible to locate your computers communication ports using the Device Manager in Windows Control panel. (Control Panel: System, Hardware; Device Manager).

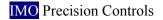


To ensure that the correct port is selected in GMWin go to the option menu in the Project drop down list.

gmwin	Make Option   Monitor/Debug Option   Set Folder Connection Option   General Option
Project Program Edit View Compile Online Debug Tools Window Help	Number of retry: 2
New Project         Popen         Upload Project From PLC         Save         Save As	Method of Connection © RS-232C © Modem Communication port COM1 © GLOFA Fnet for PC © GLOFA Mnet for PC © Ethernet
Import Project Bundle Export Project Bundle Add Item	C USB (GM4C)  Depth of Connection  C Local  Programmi port selection
M Area Edit  Prevjew  Print Project  Print Program  Ctrl+P	C Remote 1 C Remote 2 C Remote 2
Printer Setup Option	Select the Connection Option tab.

The standard method of connection is RS-232C; this is a standard 3 wire, 9 pin D-type connector crossover cable (KIC-50A). Select the communication port that you are going to use from the drop down list.





#### GMWin Menu's

#### **Project Menu**

The project menu has the options required to manage the complete project.

Project Program Edit View Compile Online Debug Tools Window Help	
★ New Project ★ Doen Upload Project From PLC 5 ave Save As Close	Starting a new project, opening an existing project, saving and uploading from a PLC (to be discussed latter).
Import Project Bundle Export Project Bundle Add Item  M Area Edit	Project bundles are the best method for transporting GMWin PLC projects. It collates all the files into one easy to transfer .MUK file.
Pre <u>v</u> jew Print Project Drint Program Ctrl+P	Print options
Printer Setup Option	GMWin options
1 c:\gmwin 4\source\junair_no_drive_52\junair_no_drive_50.prj 2 c:\gmwin 4\source\g7-ismart-board-g71\g7-ismart-board-g71.prj 3 c:\gmwin 4\source\test\test.prj 4 c:\gmwin 4\source\blank\blank.prj	Previous projects created.
E⊻it	

#### Program Menu

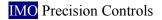
This menu operates the options required for the programs in a project.

Program	<u>E</u> dit	⊻iew	⊆ompile	<u>O</u> nline	<u>D</u> ebug	<u>T</u> ools	<u>W</u> indow	He
<mark>}∯</mark> { <u>N</u> ew	Progr	am					Ctrl+N	
Dpen Ctrl+O								
Save Ctrl+S								
Sav	e <u>A</u> s							
Sa <u>v</u>	e All							
⊆los	е							
Prop	perties							
	al Varia	bles						
In/⊂	)ut Var	iables.						
Acti	on List							
<u>T</u> rar	sition	List						
SEC	Prope	rties						
<u>1</u> c:	gmwin	4\sou	rce\fb_gei	neration	myfunct	ion2.sr	c	

up of two types of files. A .prj file that contains the project information and at least one .src source file that contains the program information.

A GMWin project is made

2 c:\gmwin 4\source\junair_no_drive_50\interlock.src



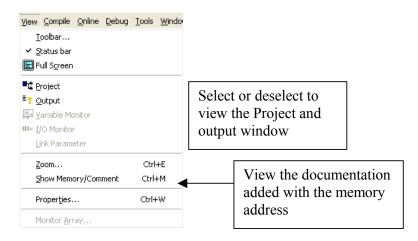
#### Edit Menu

In this menu there are the options to edit the programs.

Edit View Compile C	<u>nline D</u> ebug <u>T</u> o	-   -	F2		
🖸 Undo	Ctrl+Z	-1/1-	F3		
	Ctrl+Y		F4		In this menu the
<b>∦</b> Cu <u>t</u>	Ctrl+X	I.	F5		ontions can be
•••		-( )-	F6		options can be
В⊆ору	Ctrl+C	-(/)-	F7		found to edit
Paste	Ctrl+V	-[FUN]-	F8		found to cure
X Delete	Del	-[FB]-	F9		programs.
Eind	Ctrl+F	- P -	Shft+F1		1 0
Replace	Ctrl+H	- N -	Shft+F2		
Replace Direct Varia	ables	-(5)-	Shft+F3		
Find Next	Ctrl+F3	-(R)-	Shft+F4		
10 Go To		-(P)-	Shft+F5	-	
Find in Files		-(N)-	Shft+F6		Taddan laada
Delete Line	Ctrl+D	<ret></ret>	Shft+F7		Ladder logic
Insert Line	Ctrl+L	>>	Shft+F8	$\sim$	programming tool
-		<scal></scal>	Shft+F9		
Insert Cell	Ctrl+I	To Arrow Mo	de Ctrl+A		(discussed later)
Toolbox	۱.	To Block Moo	le Ctrl+B		()
Calculate M Area				L	

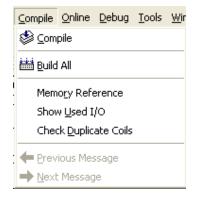
#### View Menu

Using the options in this menu you can customise what windows you want to view.

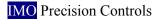


#### **Compile Menu**

This menu is used to build and compile the program written, ready for downloading it into the PLC.



There is also the option to check the program for duplicated coils, memory and I/O locations used.



#### **Online Menu**

This menu has the options for connection with the PLC. In this menu we can read and write programs to the PLC. Check the status of the PLC, enable communications and reset errors.

<u>Online</u> <u>Debug</u> <u>Tools</u> <u>Window</u> <u>H</u> elp		
🍇 Connect+Write+Run+Monitor On	Ctrl+R	Connecting and writing to the PLC
Tonnect		Connecting and writing to the TLC
Read		
<b>₽</b> ↓ <u>W</u> rite		
Monitor On/Off		
PLC Mode	•	Checking the PLC mode. Can relate
Reset	•	to the PLC dipswitch position
Elash Memory	•	to the The upswitch position
PLC Info	•	
I/O Modules	•	PLC information. I/O cards used,
I/O Eorcing	•	communication protocols engaged.
Network	•	communication protocols engaged.
Comm Info		
Online <u>E</u> dit	•	Online editing facility, allows editing
<u>E</u> SM		of the PLC program in the PLC
I/O Skip		whilst still running.
Fault <u>M</u> ask		while still running.
Initialize <u>S</u> pecial Module		

#### **Tools Menu**

The simulation tool can be found in this menu. This is a useful tool to test the project without connecting a PLC.



#### Window Menu

This window allows you to view the different programming windows in different ways.

Į	Window Help
	🔁 New Window
	🖶 Cascade
	Title Horizontally
	Title Vertically
Ì	A <u>r</u> range Icons
:	🖶 Close All
•	1 c:\gmwin 4\source\junair_no_drive_52\junair_no_drive_50.src

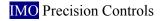
🖌 🖌 2 c:\gmwin 4\source\junair_no_drive_52\communications_3.src

#### Help Menu

In the help menu you can access the help function and IMO website. Please note that the language settings require windows to have the Far East and Asian files installed. (Control Panel: Regional and Language Settings; Languages; Supplemental Language Support).

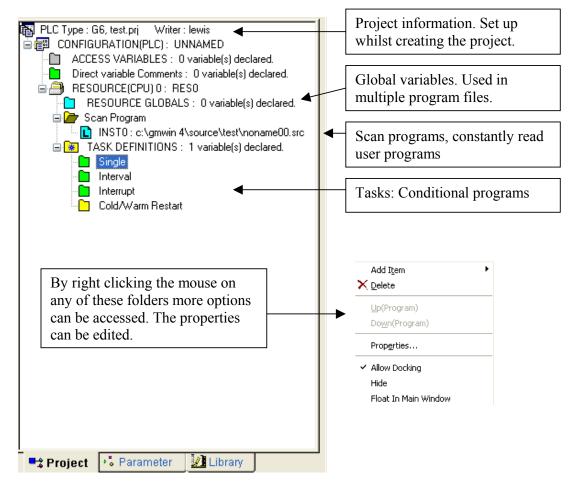
Help
GMWIN Hel <u>p</u>
IMO <u>H</u> ome Page <u>A</u> bout GMWIN

Gmwin_Eng	
File Edit Bookmark Options Help	
Help <u>T</u> opics <u>Back</u> <u>Print</u> <u>&lt;</u> <	≥>
Contents       Index       Search         Image: Search       Image: Search         Iman	Welcome Thank you for using a G-series PLC and GMWin version 4 of IMO Precision Controls Ltd This help is a User Manual of GMWIN version 4.



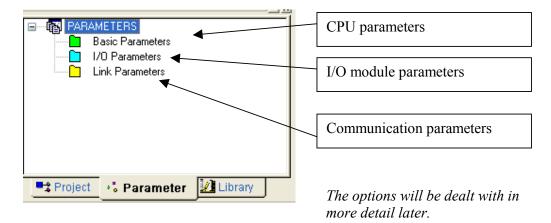
#### Project Tree

The project tree contains the complete detail of the project. In this menu you can view the variables used, Programs included and task definitions.



#### Parameter list

Access the PLC's basic parameters, I/O parameters (the different I/O cards in the slot and rack PLC's) and Link parameters (set up the communication protocols).



#### Library Menu

In this menu you can view, add and delete the libraries that contain the functions and function blocks used in programming.

Library - P Function - Standard Function	Add Item     Resource       X Delete     Program       Up(Program)     Library       Iask
i → 行詞 Function Block ・ □ Standard Function Block	Down(Program)  Properties  Allow Docking Hide
To view what functions are in the library_click on the + icon	Float In Main Window To add a library right click on a folder and select: Add item, Library.

Select the library required, and click the open button.

Open			<u>?</u> ×
Look in: 🔀	Lib		<b>-</b>
APP.6fb APP.6fu COMMUNI. Mknewlib.6 MKNEWLIB Mkstdlib.6f	ifb 💼 STDLIB.6fb 8.6fu 😇 Stdlib.6fu		
File name:	SPECIAL.6fb	01	pen
Files of type:	Library File(*.6f*)	▼ Ca	ncel

#### **Output Window**

The output window displays messages. In it you can find connection information, error messages and other details.

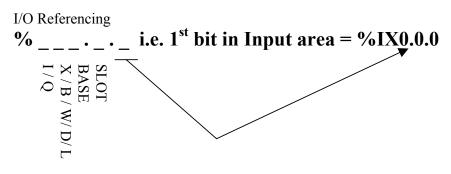


The error messages direct you to where the issue lies within the program.

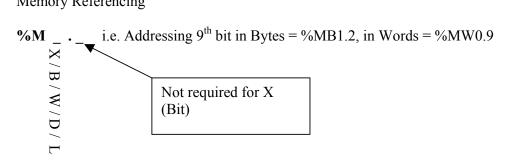
## Addressing in a G-Series PLC

The addressing format adheres to the IEC61131-3 standard for the complete range on G-series PLC, from the G7 through to the G4.

In the IEC61131-3 the addressing is standardised for all PLC manufacturers. It states that to address an input the prefix I must be used. Likewise for an Output, prefix Q and Memory, prefix M.

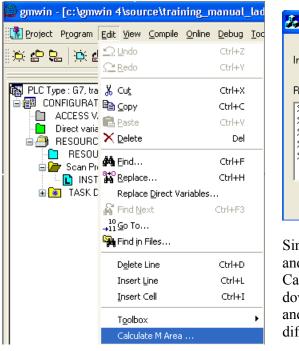


Memory Referencing



In the G-series PLC we can address individual bits, bytes, words and double words and this is done in the structure of the address. When directly addressing I/O or memory a percent symbol must be put before the address structure "%", this signifies that you intend to directly address an input, output or memory location and not a variable. The next symbol is to signify whether it is an input, output or memory location and then the second symbol indicates whether it is a bit, byte, word or double word. Finally it is the location address and this format can differ, depending on if it is an input / output or a memory location.

In the Edit Menu there is a drop down option that will aid in calculating memory locations.



Simply enter the address and click Execute. The M Calculator will then break down the memory address and how it can be addressed differently.

## Memory Allocation in G-series PLC

For a better understanding of addressing and programming in GMWin it is important to know how the memory in the PLC is organised.

The memory is split into two main areas:

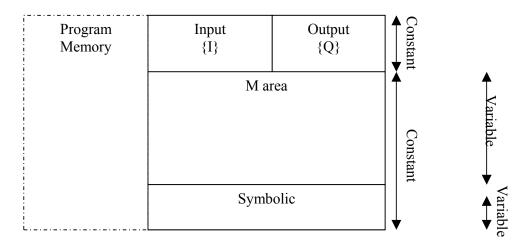
1. The Program Memory

This is where the user created program is stored.

2. The I/O, internal memory and Symbolic memory.

In this part of the memory the I/O reference particular to the type of PLC is stored, along with the internal memory used to store variable data. Lastly the symbolic area is used to store the Functions and Function Blocks used in the Program.

When you enter a Function Block, you are asked to give it an instance name. This is required for storage in the symbolic region of the PLC and it can be referenced again and again but use of its Instance Name.

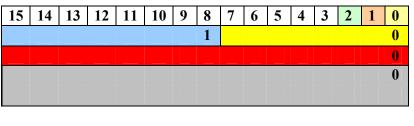


As previously mention in the addressing section, it is possible to address the memory in several different chunks: Bits, Bytes, Words, etc. The M area of memory should be considered as a blank sheet of paper and by using coordinates you locate the memory area required. However the same location can be easily addressed in different ways by using a different coordinate system.

#### i.e. %MB1.0 = %MX8 = %MW0.8

Therefore it is a good idea to use (where possible) the one consistent addressing structure. This means that if you are most likely t be mostly addressing words then use the structure: %MW_ (and the ._ if you require to address a bit in a word.)

M area (exploded view)



- <- Bits
- <-1 byte = 8 bits
- <-1 word = 2 bytes
- <- 1 double word = 2 words

{...and 2 double words = 1long word}

To address in G-Series:

X = bitB = byteW = wordD= Double word M = Memory areaI = Input Q = Output

L = long word

## **Bits and Bytes**

Different types of variable information require a different amount of bits to store it. Below is a guide to what is used for different types of variable information.

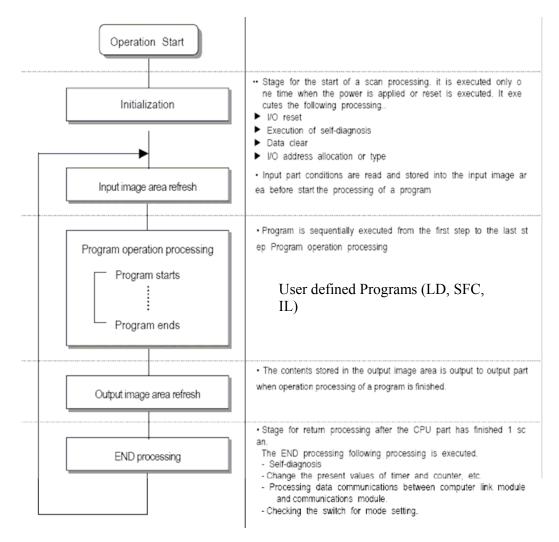
Type	Name	# of bits	Comment	
BOOL	Bit	1	Used for I/O addressing	
BYTE	Byte	8	Used for memory allocation	
WORD	Word	16	Used for memory allocations	
DWORD	Double word	32	Used for memory allocations	
LWORD	Long word	64	Used for memory allocations	
INT	Integer	16	Used in calculations	
SINT	Small integer	8	MSB signals the polarity $(0 = +ve)$	
USINT	Unsigned Small integer	8	Used in calculations	
UINT	Unsigned integer	16	Used in calculations	
ULINT	Unsigned long integer	64	Used in calculations	
DINT	Double integer	32	Used in calculations	
UDINT	Unsigned Double integer	32	Used in calculations	
REAL	Real	8	Floating-point maths	
LREAL	Long real	64	Floating-point maths	

## Ladder Programming Rules

Before programming it is important to know a few basic rules of the ladder diagram structure. In this section we will discuss the basic programming rules of ladder logic.

It is important to be aware of a PLC's scan cycle. This is the order of processing that the PLC performs. The Scan cycle of a GMWin project is as follows:

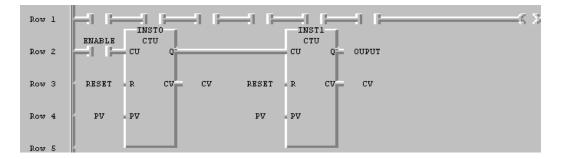
- 1. On operation start the variables are initialised, and a self-diagnosis is performed.
- 2. The input conditions are then read in and stored
- 3. The user program is then sequentially executed.
- 4. Communications with modules is checked and data shared, variables are up dated and a self diagnoses is performed again
- 5. Return to read the input conditions and refresh stored memory.



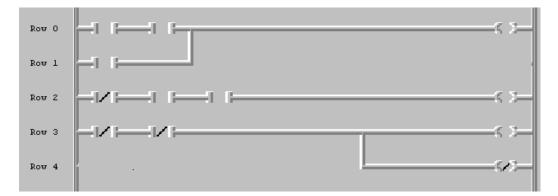
This process is known as the scan cycle and various factors can effect how quick it is. I.e. Larger and more complex programs have bigger scan cycle times.

The flow of a GMWin program is sequential from the first row to the last; this means that Code in Row 1 will be operated on before Row 2. The time difference will be fractional however it is important to note when using latching circuits and sub routines.

Up to 30 contacts can be placed on one run, but only one coil per line. Function Blocks and Functions use 3 contact spaces on a line and can be put anywhere on a run apart from the first space in a line.



In the GMWin programming environment contacts and coils can be inputted without naming them. This means that the program structure can be drawn out in full before assigning variables.

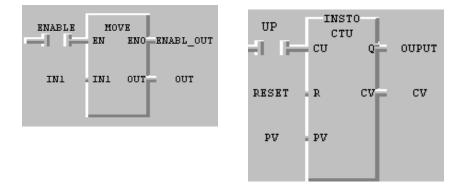


All Rows that have been started must be completed, this means that there must be a complete horizontal connection between the vertical power-rails otherwise the program will not compile.

No output coil No input coil / connection to rail	( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )( )
Incomplete connection	

A Function only has one output and requires an enable signal whereas a Function Block doesn't require an enable and can have several outputs and internal data. When entering a Function Block the user must define an instance name.

Each Function / Function Block requires a Contact in the first input that a named Variable or constant at the other inputs. It is possible to create user defined Functions and Function Blocks.



It is possible to duplicate coils and this will not bring up an error. This could have been intended however it is an easy mistake to make. With a duplicated coil it will never be active until all conditions are met.

For example in the ladder logic below, A and B on would usually switch Z on. However Z has been duplicated and so required all conditions (A, B, C and D) to be on before being switch positive.



# Variable Declaration

Variables can be expressed in one of two ways. The first is to give a Name to a data element using an identifier and the second is to directly assign a memory address / input or output address to a data element.

Variables can either be entered as contacts/coils/functions or function blocks are entered or in the variable table.

Right click on the mouse when the arrow is in the table area at the top of the ladder logic program.

Variable Na	Data Type	Memory All	Initial Value	Variable Kin	Used	Comments
	Add New	Variable	l			
	Cut					
	Del					

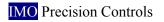
	Variable Name	Data Type	Memory All	Initial Value	Variable Kind	Used	Comments
1	DOWN	BOOL	<auto></auto>		VAR		down input
2	INSTO	FB Instance	<auto></auto>		VAR		up counter
3	INST1	FB Instance	<auto></auto>		VAR		down counter
4	INST2	FB Instance	<auto></auto>		VAR		up down counter
5	LOAD	BOOL	<auto></auto>		VAR		load input, loads pv value
6	PV	INT	<auto></auto>	10	VAR		preset value
7	RESET	BOOL	<auto></auto>		VAR		reset
8	UP	BOOL	<auto></auto>		VAR		up input

Variable table with variables assigned in the program.

Directly addressing uses the percent sign % followed by the location prefix. The location prefix contains a location identifier (I, Q, M), followed by a size prefix (X, B, W, D, L) and completed with an expression indicator (for I and Q, base . slot. Data; for M, data (according to size prefix). bit of data).

Name : %IX0.2.6	Direct Variable Comment
	Direct Variable Comment
When directly addressing	Variable Name : %IX0.2.6
it is possible to add Direct Variable comments.	Input bit 6 on slot 2, base 0
	OK Cancel Help

i.e. % I X 0 . 2 . 6 : Input, Bit size, Base 0, Slot 2, Bit 6. % M W 32 . 7 : Memory, Word size, word 32, bit 7.



When naming a variable there is a little more information that has to be inserted.

Add/Edit Variables			×
Variable name: 16 characters, with no spaces Variable Kind Variable Kind :	VAR	OK Cancel	
Normal, Constant, Retainable, Global © Elementary :	BOOL	Memory Allocation	Assigning
Data type: whether it is binary, numeric, time, array, etc	) OF BOOL	C Assign(AT):	memory to a location
An initial value to be loaded into variable on power up		Init. Array	Documentation comments
-Variable Kind Variable Kind : VAR VAR_CONS VAR_RETA VAR_EXTE	IN	VAR: General read / VAR_CONSTANT: VAR_RETAIN: VAR_EXTERNAL:	
Data Type     Elementary :     BOOL     FB Instance :     BYTE     WORD     WORD     LWORD     LWORD     INT		i.e. Integer value, time is important, as the di	ariable is representing. e, binary, etc. This also
DINT LINT USINT UINT	V		
Memory Allocation Auto Assign(AT) : MVV32.7		allocate the memory location. like the direct	

Initial Value	Init. Array	Data can be set with an initial value instead of zero
Comments Variable comments		Comments for documentation

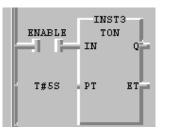
### Time Variables

Time variables are used in Timers for the preset value and need special attention. Data intended to be entered into timers must use advanced functions to convert to a Time Variable. This will be discussed further in the Advanced Function section.

Time variables have the prefix of: t# The value is the entered before ending with the variable with the timing constraint.

i.e. A Timing Value of two days, seven hours, three minutes, twenty seconds and sixteen milliseconds, would become: t# 2 d 7 h 3 m 20 s 16 ms

The maximum time allowed is T#49d17h2m47s295ms (32 bits of ms unit)



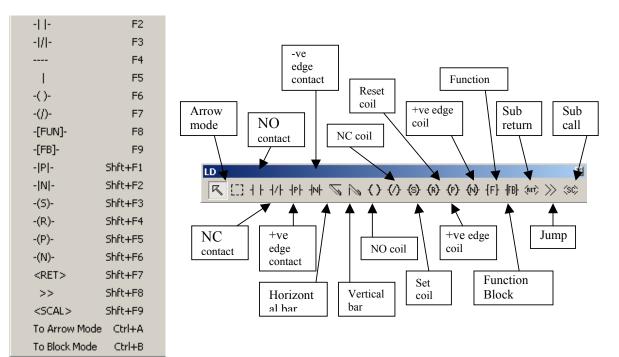
This is a 5s On-delay timer

# **Basic Ladder Logic**

All ladder diagram programs contain basic Boolean logic and interlocking circuitry. It is important to master the basics as complicated programs are made of these simple programs.

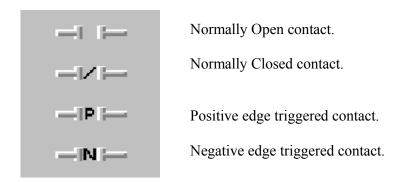
### Inputs and Outputs

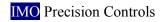
The inputs and outputs can be inserted by using the Icon bar and click on the desired icon or by using a hot key.



The inputs and outputs are of a Boolean type and operate as either ON or OFF.

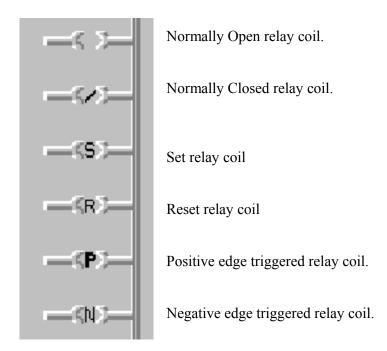
#### Inputs





Ζ

#### Outputs



#### **Boolean Logic**

Boolean logic deals with binary data, (ON or OFF), with knowledge of some basic Boolean logic circuits most events can be accounted for and controlled.

AND Gate: Z = A AND B

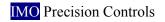


OR Gate: Z = A OR B



NOT Gate: Z = NOT A





Or 

## Latching Circuits

A latching circuit holds an output high when the input has been removed. There are two methods of latching circuit: Circuitry latch and Set / Reset coils.

Circuitry latch using the output as an interlock.

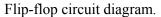


Using S R relay coils.



## Flip / Flop Circuit

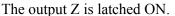
A flip-flop circuit toggles the output. The first input pulse will set the output high and the second input pulse will reset.





1st Pulse – Switches On the output Z.







The  $2^{nd}$  input pulse resets the output and switches Off Z.



## Ladder Programming Example

Design a simple ladder circuit using a logical AND gate and an OR Gate to create a basic interlock.

First open GMWin and start a new project.

Select the Normally Open (NO) contact icon  $\frac{|+|}{|}$  in the toolbox and click the right button of the mouse into position row '0' and column '1' in the LD window

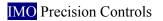


Next Select a Normally Closed (NC) contact icon 4/1 in the toolbox and click the right button of the mouse on position row '0' and column '2' in the LD window



Next Select a normally open coil icon  $\bigcirc$  in the toolbox and click the right button of the mouse on position row '0' at the far right hand side. You will see that it the line is automatically created so that it is connected across to the inputs.





Select the Normally Open (NO) contact icon  $4^{+}$  in the toolbox and click the right button of the mouse on position row '1' and column '1' in the LD window

Row O	-
Row 1	

Select the Up line connection icon in the toolbox and click the right button of the mouse between row '1' and row '0' and column '1' in the LD window.

Row O	<u>,</u>	
1.000 0		
Row 1		

Now that the diagram has been entered we can enter the variables. Double click on the contacts and coils with the right mouse button to enter the Variables menu.

<mark>Variables</mark> Name :	8		Direct	Variable Comm	ent	ок	
- Variables	List Var. Kind	Allocat Us	ed Data Type	Initial value	Comments	Cancel Flag Global Direct Variable Add Delete Edit Help	Click Add to enter new variables

Variable			OK	
Variable Kind			Cancel	
Variable Kind :	VAR	-	Help	
Data Type			Memory Allocation	Enter t
Elementary:	BOOL	•	C Auto	variabl data. T
C FB Instance :	CTD	~	<ul> <li>Assign(AT) :</li> </ul>	click d
C Array (0	) OF BOOL	~	%	when
Initial Value				comple
			Init Away	1
			Init. Array	
Comments				

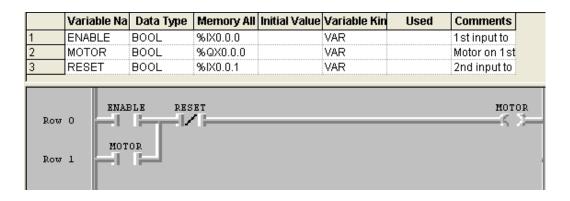
Name	Туре	Address	Comment
Enable	BOOL	%IX0.0.0	1 st input to enable
Reset	BOOL	%IX0.0.1	2 nd input to reset
Motor	BOOL	%QX0.0.0	Motor on 1 st output

Keep adding variables until you have a variable menu like below. To assign variable to the contacts and coils, Double click on the contact or coil and simply select the variable from the menu and press OK.

ame :	MOTOR		Direct Varia	ble Comment	ок
Variables Lis	:t				Cancel
Name	Var. Kin			tial value Comments	Flag
ENABLE MOTOR	VAR VAR	%IX0.0.0 %QX0.0.0	BOOL BOOL	1 st input to enable Motor on 1 st output	
RESET	VAR	%IX0.0.1	BOOL	2nd input to enable	Global
					Direct Variat
					Add
					Delete
					Edit
					Help

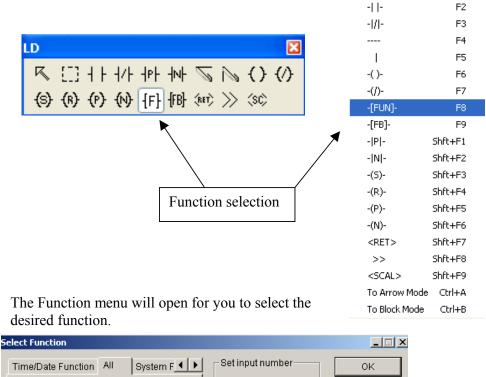
The program should now look like the one below.

#### IMO Precision Controls



# **Basic Functions**

There are two methods of entering a function into a program: Selecting the Function icon or pressing the hotkey. Both methods attached the Function icon onto the mouse arrow and then simply click it into the diagram.

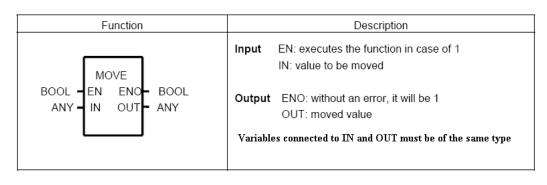


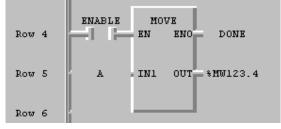
Time/Date Function All System F	Set input number	ОК
MOVE	Max number:	Cancel
MCS MCSCLR MID MIN MOD MOVE	Required number:	Help
Function information		
Comment: Copy data		
MOVE EN ENO NOTYPE OUT	×.	

Select the function you want and enter the settings required. Press OK once complete.

## Move Function

A frequently used function. Used to move data from one memory location into another location.





When the Enable input goes high, the data in A will be moved to memory location %MW123.4.

#### Arithmetic Functions

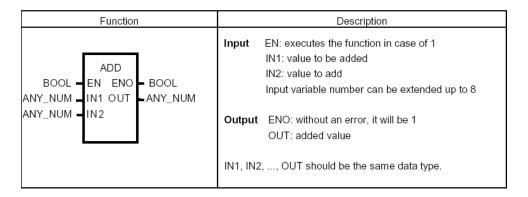
The PLC can perform all mathematical functions. However the only PLC CPU that can perform true floating-point maths is the G4 CPUC. This means that the result of an arithmetic function will return an integer, unless using a GM4-CPUC.

Arithmetic functions are entered as any other Function, however you can select the number of inputs required. You can select the desired function easier by selecting the arithmetic tab, which lists only those particular functions.

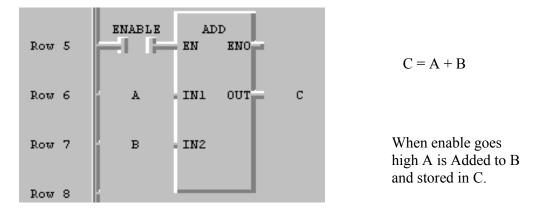
Set input number	Arithmetic Function   Bit Function   Cor	All arithmetic
Max number: 8	1	functions are available in a sub menu for
Required number:	ABS ADD DIV MCS MCSCLR MOD V	ease of selection. Available functions include Add /
		Sub / Mul / Div /Mod

#### Add Function

The Add function performs addition on up to 8 variables.



The Addition Function can have up to 8 eights. Select the number of inputs required for the function and click ok. Below is an example of a 2 input Add function.

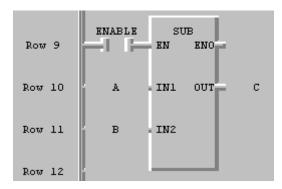


#### Sub Function

The Sub function, subtracts one variable from another.

Function	Description
SUB BOOL – EN ENC- BOOL ANY_NUM – IN1 OUT- ANY_NUM ANY_NUM – IN2	Input       EN: executes the function in case of 1         IN1: the value to be subtracted         IN2: the value to subtract         Output       ENO: without an error, it will be 1.         OUT: the subtracted result value
	The variables connected to IN1, IN2 and OUT should be all the same data type.

The Subtraction function can only have two inputs.



C = A - B

When Enable is set high, B is Subtracted from A and stored in C.

## **Mul Function**

The multiplication function can up to eight inputs selected.

Function	Description		
MUL BOOL – EN ENO – BOOL ANY_NUM IN1 OUT – ANY_NUM ANY_NUM IN2	Input       EN: executes the function in case of 1         IN1: multiplicand         IN2: multiplier         Input is available to extend up to 8.         Output       ENO: without an error, it will be 1         OUT: multiplied value         Variables connected to IN1, IN2,, OUT are all the satisfied to the satisfier	ame	

The example below is a three input is multiplication function.

Row 9	ENABLE	MU EN	ENO	
Row 10	A	IN1	OUT	D
Row 11	в	IN2		
Row 12	· c	INS		
Row 13				

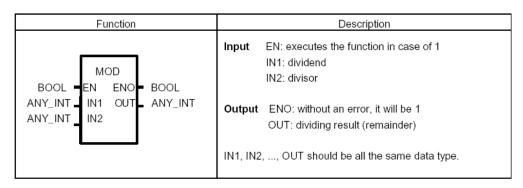
 $D = A \times B \times C$ 

When the enable signal is active the calculation is performed (A x B x C) and stored in D.

#### **Div and Mod Function**

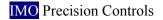
The divide function can only have two inputs and returns only a whole number. The Mod function calculates the remainder and returns the value as in integer.

Function	Description		
DIV	<ul> <li>Input EN: executes the function in case of 1</li></ul>		
BOOL – EN ENO – BOOL	IN1: the value to be divided (dividend)		
ANY_NUM – IN1 OUT – ANY_NUM	IN2: the value to divide (divisor) <li>Output ENO: without an error, it will be 1.</li>		
ANY_NUM – IN2	OUT: the divided result (quotient) <li>The variable connected to IN1, IN2 and OUT should be all the same data type.</li>		



C equals A divided B and D equals the remainder of A divided by B.

Row 10	ENABLE DIV	ENO	EN	MOD I ENO
Row 11	A IN1	оυт с	A IN	11 OUT D
Row 12	B IN2		B IN	12
Row 13			_	



### **Bit Functions**

Bit functions and functions that manipulate the fundamental bits of data.

Arithmetic Fun	ction Bit I	Function	Cor 🔸 🕨
AND			~
OR			
ROL			
ROR			
SHL			
SHR			
I YOR			<b>×</b>

Depending on the type of Function, they can have a minimum of two inputs and an enable signal.

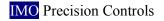
The bit functions include Boolean Logic Functions like AND, OR, and XOR but also include Rotate and Shift functions like ROR, ROL, SHR and SHL.

#### AND, OR, XOR Functions

*These functions can also be performed using ladder logic. See Basic Ladder Logic, Boolean Logic.* 

Function	Description
AND BOOL - EN ENO - BOOL ANY_BIT - IN1 OUT - ANY_BIT	Input EN: executes the function in case of 1 IN1: input 1 IN2: input 2 Input variables can be extended up to 8.
ANY_BIT - IN2	Output ENO: without an error, it will be 1 OUT: AND result IN1, IN2, and OUT should be all the same data type.

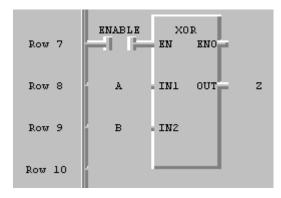
Function	Description		
OR BOOL EN ENO BOOL ANY_BIT IN1 OUT ANY_BIT	Input EN: executes the function in case of 1 IN1: input 1 IN2: input 2 Input variables can be extended up to 8.		
ANY_BIT - IN2	Output ENO: without an error, it will be 1. OUT: OR result		
	IN1, IN2, OUT should be all the same data type.		



Function		Description
XOR BOOL - EN ENC - BOOL ANY_BIT - IN1 OU - ANY_BIT ANY_BIT - IN2	Input Output	EN: executes the function in case of 1 IN1: the value to be XOR IN2: the value to be XOR Input variable number can be extended up to 8. ENO: without an error, it will be 1. OUT: the result of XOR operation
	IN1, IN2,	OUT should be all the same data type.

#### 4 Input AND Gate Z = A AND B AND C AND D

Row O	ENABLE	AND EN EI	по
Row 1	A	IN1 01	ut z
Row 2	в	IN2	
Row 3	c	IN3	
Row 4	D	IN4	
Row 5			



2 Input Exclusive OR Z = A XOR B

Row 12	ENABLE	OR EN ENO	
Row 13	A	IN1 OUT	2
Row 14	в	IN2	
Row 15	c	ING	
Row 16			

3 Input OR Gate Z = A OR B OR C

#### **Shift and Rotate Functions**

Shift and rotate functions operate on byte and word variable kinds. They perform bit multiplication and division.

	ENABLE ROR	Function	Description
Row O Row 1	A IN OUT 2	ROR BOOL = EN ENO = BOOL ANY_BIT = IN OUT _ ANY_BIT	Input EN: executes the function in case of 1 IN: the value to be rotated N: bit number to rotate
Row 2	- 2 - N		Output ENO: without an error, it will be 1. OUT: the rotated value
Row 3			
Row 5	ENABLE SHL EN ENO	Function	Description
Row 6	B IN OUT	z	Input EN: If EN is 1, function is executed. IN: bit string to be shifted
KOW 6	B HIN OUT	SHL BOOL EN ENO BOOL	N: bit number to be shifted
Row 7	1 N	ANY_BIT IN OUT ANY_BIT	Output ENO: without an error, it will be 1 OUT: the shifted value
Row 8	i		

#### **Compare Functions**

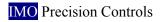
Comparison functions are used to evaluate data.

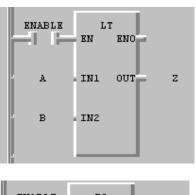
Bit Function	Comparison Function
GE GT LE LT MID	
NOT	

- GE Greater or Equal
- GT Greater Than
- LE Less or Equal
- LT Less Than
- EQ Equal
- MID Takes the middle part of a character string.
- MUX Selection of 1 from a multiple of inputs.
- SEL Select one of two inputs

# LT, EQ, and GT Functions

Function		Description
LT BOOL = EN ENC = BOOL ANY= IN1 OUT =BOOL ANY= IN2	Input	EN: executes the function in case of 1 IN1: the value to be compared IN2: the value to compare Input variable number can be extended up to 8. IN1, IN2,should be the same data type.
	Output	ENO: without an error, it will be 1. OUT: comparison result value
Function		Description
GT	Input	EN: executes the function in case of 1 IN1: the value to be compared IN2: the value to compare
BOOL – EN ENO – BOOL ANY – IN1 OUT – BOOL ANY – IN2		Input variable number can be extended up to 8. IN1, IN2, should be the same data type.
	Output	ENO: without an error, it will be 1. OUT: comparison result value
Function		Description
EQ BOOL – EN ENO– BOOL ANY – IN1 OUT – BOOL ANY – IN2	Input	EN: executes the function in case of 1 IN1: the value to be compared IN2: The value to compare Input variable number can be extended up to 8. IN1, IN2, should be the same type.
	Output	ENO: without an error, it will be 1. OUT: comparison result value
ENABLE GT EN ENO A IN1 OUT Z B IN2		is Greater Than B then the at Z will be high.





If A is Less Than B then the output Z will be High.

1 OUT Z
z

If A Equals B then the output Z will be High.

### Function Programming Example

Write a simple Move, Comparison and Add function program.

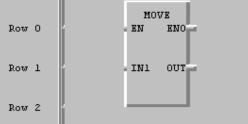
Select a Function by clicking the icon *[F]* in the toolbox with the right button of the mouse and put into position row '0' in from the first column.

Row O	1		
Row 1			

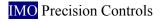
The Function menu will then pop up.

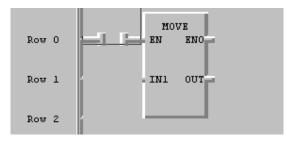
	Select Function		
	Time/Date Function All System F	Set input number	ОК
Either search through the functions or if you know the desired function name, enter it in the box.	MOVE	Max number: Required number:	Cancel Help
Type MOVE and then click OK.	MOVE EN ENO NOTYPE OUT		

The Function is now in the ladder diagram. Note this particular function requires three rows.



Now insert a NO contact in row 0, column 1.

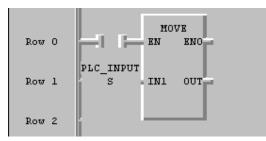




To insert a new variable at the IN1 of the MOVE function, double click the right mouse button in the space to the left of IN1. This will open up the variables menu.

Variables		
Name : Direct Variable Comment Variables List Name Var: Kind Allocat Used Data Type Initial value Co	OK Cancel Flag Global Direct Variable Add Delete Edit Help	Click Add to insert a new variable.
Add/Edit Variables Variable PLC_INPUTS Variable Kind Variable Kind : VAR	OK Cancel Help	We are going to read the inputs of the PLC as a word. So we
Data Type       Image: Elementary :       Image: Elementary : <td>Memory Allocation Auto Assign(AT) : WV0.0.0</td> <td>need to change the elementary to WORD.</td>	Memory Allocation Auto Assign(AT) : WV0.0.0	need to change the elementary to WORD.
C Array (0,) OF BOOL	Init. Array	Click OK when the details have been entered
Comments 1st 16 inputs on the PLC		

The variable is now entered at the function. Note that the Functions Function Blocks only require the first input to be a contact and the rest of the inputs are named variables.



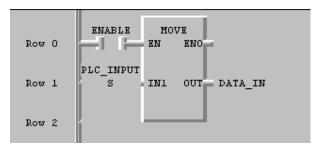
Double click the right mouse button on the input contact and enter the variable "Enable" as a Boolean and Auto allocated.

The input side is now set up on the function.

Now to set up the output of the MOVE function. Double click the right button on the mouse in the space to right of the OUT on the MOVE function. This will bring up the variable menu and we need to add a variable for the output, for data to be moved to.

Add/Edit Variables	×	
Variable DATA_IN Variable Kind Variable Kind : VAR	OK Cancel Help	Enter the details and click OK.
Data Type	Memory Allocation C Auto Auto Assign(AT) : MVV0	
Initial Value Comments Inputs move to a internal memory	Init. Array	

Row

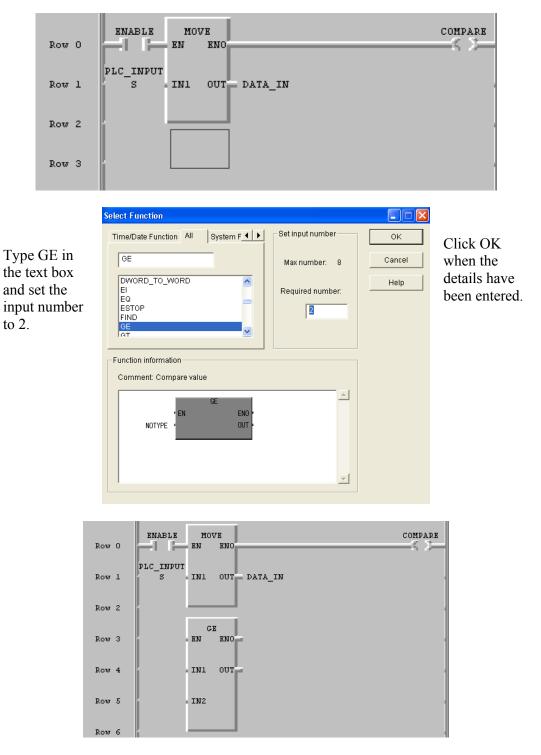


Insert a NO coil off the ENO output of the MOVE function. It is not always necessary to do this however for this example we require it. Name the variable Compare, the type will be Boolean and auto allocate it.

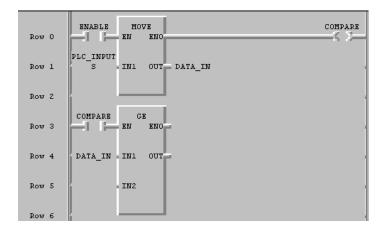
Add/Edit Variables		
Variable COMPARE Variable Kind Variable Kind : VAR	ок Enter th Cancel data and Help click O	d
C Array (0 ) OF BOOL	Memory Allocation Auto Assign(AT) : %	
Initial Value Comments	Init. Array	
this o/p will trigger the compare		_
O ENABLE MOVE EN ENO	COMPARE	
S INI OUT DATA_IN		

Now that the move function has been set up we can continue the program with setting up a compare function. We want to compare the DATA IN to a fix value, if DATA IN is greater or equal to the value 100 we want the output to go high. In this case we are going to use the function GE.

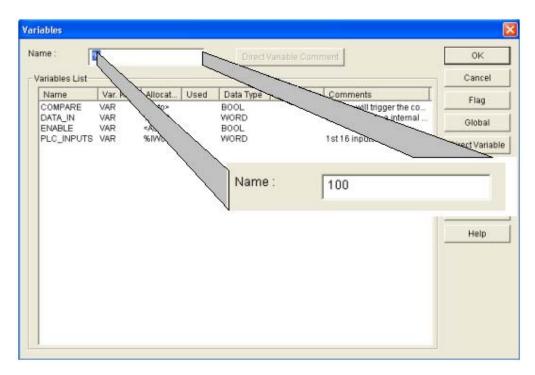
Select a Function by clicking the icon [F] in the toolbox with the right button of the mouse and put into position row '3' in from the first column.



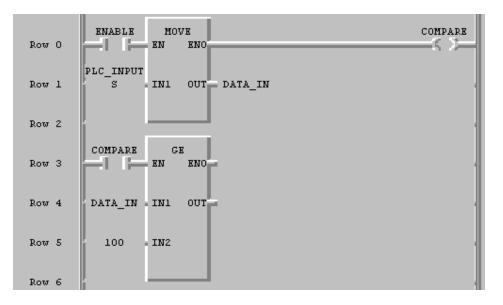
Enter a NO contact at EN and assign it to the variable COMPARE. Using the right mouse button and double click in the space to the left of IN1, assign the variable DATA_IN.



Double click the right mouse button in the space to the left of IN2 to bring up the variables menu.

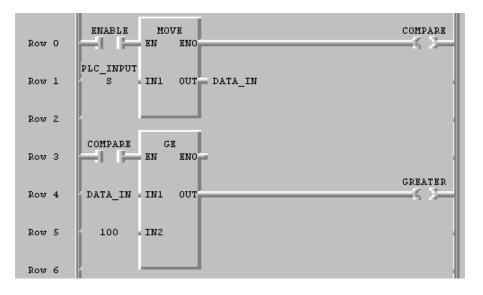


Highlight and delete the % symbol to enter a constant value of 100. Click OK when complete.

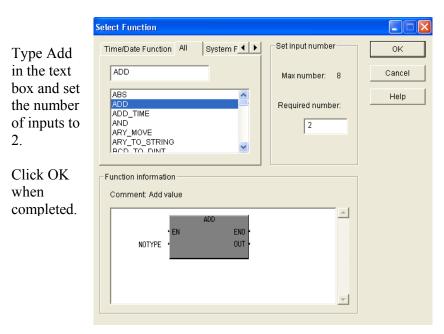


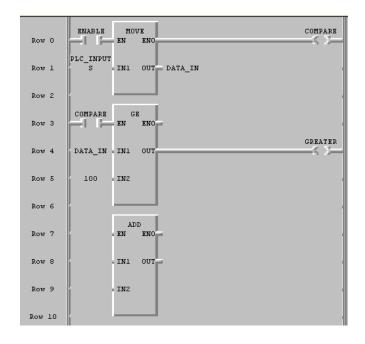
Now assign a NO coil to the output OUT of the GE function, and add a new variable to assign to the coil.

Add/Edit Variables		
Variable GREATER Variable Kind Variable Kind : VAR Data Type © Elementary : BOOL C FB Instance : OTD C Array (0 ) OF BOOL	OK Cancel Help Memory Allocation • Auto • Assign(AT) : %	Enter a new variable named GREATER, of element type BOOL and auto allocate it. Click OK,
Comments	Init. Array	when done.



Now we want to set up the Add function select a Function by clicking the icon [F] in the toolbox with the right button of the mouse, put the function into position row '7' in from the first column.



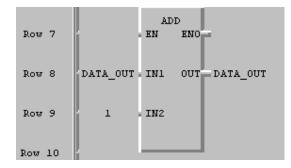


With the Add function we want to Add 1 to the outputs of the PLC every time the GREATER output is high.

Double click the right mouse button to the left of the input IN1 of the ADD function. Add a new variable called DATA_OUT, make the element type an INT (integer) and allocate the memory to %QW0.0.0. If we make the output the same as the input, each time the output is triggered the input will increment. Therefore double click the right mouse button in the space to the right of the OUT output of the ADD function.

Add/Edit Variables		
Variable DATA_OUT Variable Kind Variable Kind : VAR	OK Cancel Help	Click OK when done
Data Type       INT       FB Instance :       C Array (0       ) OF	Memory Allocation Auto Assign(AT) : % QVV0.0.0	
Comments	Init. Array	

Double click the right mouse button in the space to the left of the input IN2 and enter the constant 1.



Finally we need to set up the enable to trigger the ADD function when variable GREATER is high. However we only want one pulse per GREATER trigger, therefore we will use a leading edge pulse contact.

Select the rising edge pulse contact icon in the toolbox and click the right button of the mouse on position row '1' and column '1' in the LD window

Row O	ENABLE MOVE EN ENO	COMPARE
Row 1	PLC_INPUT S IN1 OUT DATA_IN	
Row 2		
Row 3	COMPARE GE EN ENO	
Row 4	DATA_IN INI OUT	GREATER
Row 5	- 100 - IN2	
Row 6		
Row 7	GREATER EN ENO	
Row 8	DATA_OUT IN1 OUT DATA_OUT	
Row 9	· l IN2	
Row 10		

Now the program is complete, try simulating the program to see how it works.

# **Basic Function Blocks**

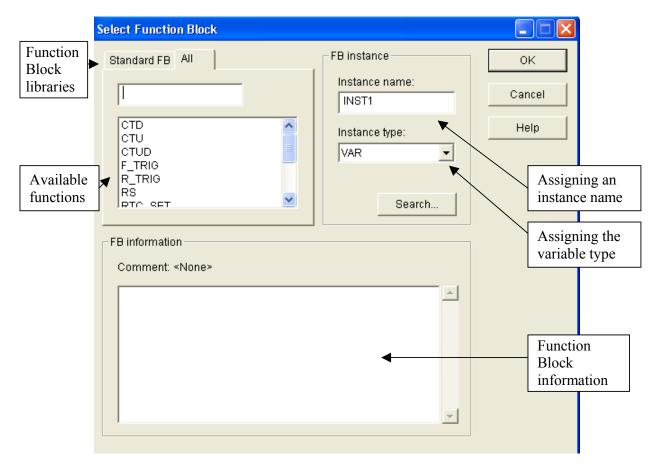
A Function Block (FB) can have several outputs and contains data inside. A Function Block is required to be declared on entry.

Enter a Function Block by clicking on the hotkey or selecting the icon.

-   -	F2
-1/1-	F3
	F4
I	F5
-( )-	F6
-(/)-	F7
-[FUN]-	F8
-[FB]-	F9

LD									X
R	[]]	+ +	<del>1</del> /ŀ	┨╒┠	- ∾ -	$\mathbb{Z}$	$\square$	()	<b>(</b> )
(s)	(R)	(P)	(Ŋ)	{F}	- <b> </b> № - -{FB}-	(RET)	$\gg$	<sc></sc>	

When you select the Function Block, it attaches to the mouse arrow. Click it into the programming area and the Function Block selection menu pops up.



## **Timer Function Blocks**

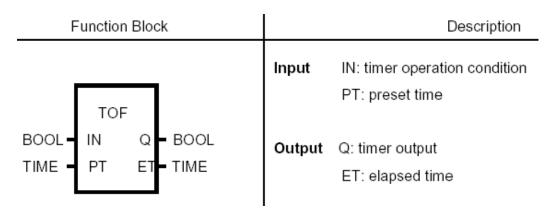
The three basic Timer Function Blocks are: TOF, Off-delay timer, TON, On-delay timer and TP, Pulse timer.

When the timers have been inserted GMWin creates the outputs to use elsewhere in the program.

′ariables List	,						Cancel
			le e d	Data Tura	In it also has	Comments	 Cancer
Name A	Var. Kind VAR	Allocat U	Jsed	Data Type SINT	Initial value	Comments	Flag
B C	VAR VAR	<auto> <auto></auto></auto>		SINT SINT			Global
D ENABLE IN	VAR VAR VAR	<auto> <auto> <auto> *</auto></auto></auto>		SINT BOOL BOOL			Direct Variable
INST1 INST2	VAR	<auto></auto>		FB Instance			Add
INST3 OUT	VAR VAR	<auto> %QX0.5.0 *</auto>		FB Instance BOOL			Delete
PT Z	VAR VAR	<auto> <auto></auto></auto>		TIME BOOL			Edit
INST1.Q INST1.ET INST2.Q	VAR VAR VAR	<auto> <auto> <auto></auto></auto></auto>		BOOL TIME BOOL			Help
INST2.ET INST3.Q INST3.ET	VAR VAR VAR	<auto> <auto> <auto></auto></auto></auto>		TIME BOOL TIME			
INOTS.ET	1013	-000-		T TWIE			

Timer Function Blocks create the outputs .Q and .ET. The .Q represents the output state of the timer and the .ET represents the Elapsed Time.

#### TOF, Off Delay timer.



When enable goes high and the timer output is set high, when the enable then goes low the timing begins and the output remains high until the timers' time has elapsed.

ENABLE	INST TON IN	
PT	PT	ET

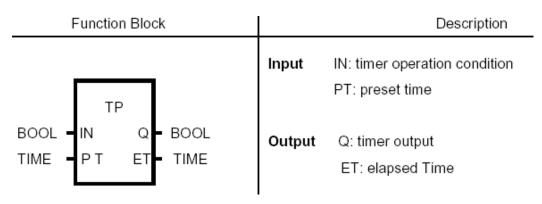
## TON, On delay Timer.

Function Block		Description
TON	Input	IN: timer operation condition PT: preset time
BOOL IN Q BOOL TIME PT ET TIME	Output	Q: timer output ET: elapsed Time

ENABLE	INST TOP IN	-
PT	PT	ET

When the enable goes high the timer begins to count. The output will go high when the timing has elapsed. The output will remain high until the Enable input has been removed.

#### TP, Timer Pulse.



When the enable signal is set high the timing begins, the output will be set high for the duration of the timing. Once the timing has been completed the output goes low and the timing begins again. The output will pulse high-low whilst the Enable signal is on.

ENABLE	INS TP IN	
PT	PT	ET

## **Counter Function Block**

There are three basic types of Counter Function Blocks: CTU, CTD, CTUD. The CTU Counter is an Up counter, the CTD is a Down Counter and the CTUD is an Up / Down counter (it is a combination of a CTU and CTD).

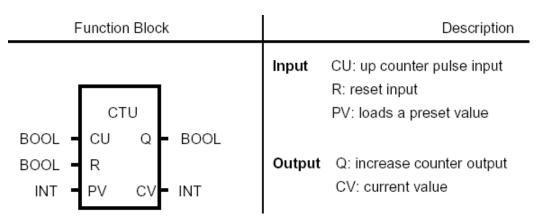
When a Counter Function Block is inserted GMWin creates the outputs for use in the program.

	,						Convert
/ariables List						/	Cancel
Name	Var. Kind	Allocat	Used	Data Type	Initial value	Comments	Flag
DOWN INST0	VAR VAR	≺Auto> ≺Auto>		BOOL FB Instance			
INST1	VAR	<auto></auto>		FB Instance			Global
INST2	VAR	<auto></auto>		FB Instance			
LOAD	VAR	≺Auto≻		BOOL			Direct Variable
PV	VAR	≺Auto≻		INT			Add
RESET	VAR	≺Auto≻		BOOL			Auu
INST0.Q	VAR VAR	<auto></auto>		BOOL BOOL			Delete
INST0.CV	VAR	<auto></auto>		INT			
INST1.Q	VAR	≺Auto≻		BOOL			Edit
INST1.CV	VAR	≺Auto≻		INT			
INST2.QU	VAR	≺Auto≻		BOOL			Help
INST2.QD INST2.CV	VAR VAR	≺Auto> ≺Auto>		BOOL INT			
114012.00	100	-Autor					

All counters have a .CV output, this is the Current Value. The Up and Down counters have a .Q output for when the counter is output is high. Finally the Up / Down Counter has an output state for both Low count condition and a High output condition.

#### CTU, Up counter.

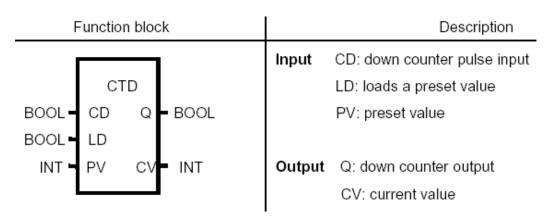
#### IMO Precision Controls



The Counter will count up every Up pulse input. The output of the counter will go High when, CV is equal to PV. The Current value will continue to go up on every Up input pulse and the output .Q will remain high until the Reset input is triggered. This then resets the Counter CV back to zero and the output will go off.

UP	INST CTU CU	0 Q
RESET	R	cv
· pv	ΡV	

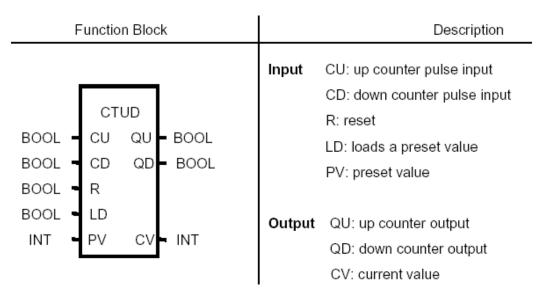
## CTD, Down Counter



DOWN	INS: CTI CD	
LOAD	LD	cv
. PA	- PV	

The down counter works in the opposite way to the Up counter. There is no reset input but a Load input, when high this loads the counter with the Preset Value. The down input will then count down from the PV value loaded and the counter will go high when CV is equal to zero.

### CTUD, Up / Down Counter.

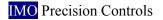


The Up / Down Counter is a combination of the two previous counters (CTU and CTD).

The CTUD has all the inputs of the CTD and CTU, plus it has an Up condition output .QU and Down condition output .QD. The Up condition output will go high when CV is equal to the PV value. It will then remain high whilst CV is greater than or equal to PV. The Down condition output .QD will go high when CV is equal to zero.

The Up input increments CV, whilst the Down input decrements CV. The Reset input, resets CV back to zero, whilst the Load input, loads the Preset Value (PV) into the Current Value (CV).

	INS CTU CU	
DOWN	- CD	QD-
RESET	= R	cv
LOAD	LD	
PΔ	PV	



# Function Block Programming Example

Design a simple timer and counter program that will count the timer output.

Select a Function Block by clicking the icon in the toolbox with the right button of the mouse and put into position row '0' in from the first column.

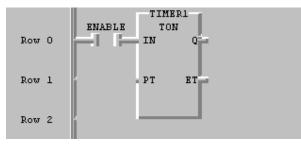


The Function Block menu will pop up.

	Select Function Block	
Either search through the functions or if you know the desired function block name, enter it in the box. Type TON into the text box for an on delay timer	Standard FB All     TON   RS   RTC_SET   SEMA   SR   TOF   TON   TP     FB information   Comment: ON delay timer     FB information   Comment: ON delay timer     BOOL   IN   BOOL   TIME     BOOL     TIME     BOOL     TIME     BOOL     TIME     BOOL     TIME     BOOL     TIME     BOOL     TIME     BOOL     TIME     BOOL     TON     Search     ON     Standard FB     All     BOOL     TON     BOOL     TON     Constant     Constant <	Give the timer the instance name timer1 and click OK.
	Row 0 Row 1 Row 2	

Insert a NO contact at the IN input to the timer and Add a variable. Name the variable Enable and allocate the address to %IX0.0.0 . You will notice that there are some variables that have been created already, TIMER1.Q and TIMER1.CV. These are the output and current value of the timer. They can be used elsewhere in the program simply by assigning them to inputs and contacts.

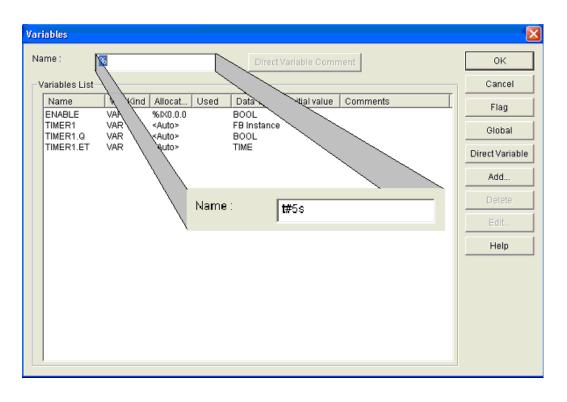
ame :	ENABLE		Direct	Variable Comm	nent		OK
Variables Lis	t						Cancel
Name	Var. Kin		Jsed Data Type	Initial value	Comments		Flag
ENABLE TIMER1	VAR VAR	%IX0.0.0 <auto></auto>	BOOL FB Instance				Global
TIMER1.ET	VAR	<auto></auto>	TIME			-	Ciobai
TIMER1.Q	VAR	<auto></auto>	BOOL				Direct Variable
						[	Add
							Delete
							Edit
							Help

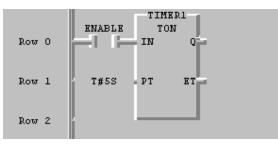


Double click the right mouse button in the space to the left of the input PT. This brings up the Variable menu. We are now going to enter a Preset Time for the timer.

Delete the % sign and insert the data, t#5s. This is the prefix for a five seconds time.

### IMO Precision Controls





There is no need to allocate output variables as they have already been created: TIMER1.CV and TIMER1.Q

We now want to insert a Counter, select the Function Block icon by click on it in the toolbox with the right button of the mouse and putting it three columns to the right if the timer function block.

Row O	ENABLE TON IN Q
Row 1	T#5S PT ET
Row 2	

Either search through the functions or if you know the desired function block name, enter it in the box.

Type CTU into the text box for an UP counter

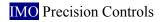
Select Function Block			
Standard FB All CTU CTD CTU CTUD F_TRIG R_TRIG RS PTC SET FB information Comment: increase counter BOOL • CU BOOL • CU U U U U U U U U U U U U U	FB Instance Instance name: IP_COUNT Instance type: VAR • BOOL INT	OK Cancel Help	Give the timer the instance name IP_COUNT and click OK.

As with the timer function block the counter function block automatically creates outputs, in the case: IP_COUNT.Q and IP_COUNT.CV

Var	iables							X
Na	ame:	%			Direct	/ariable Comn	nent	ОК
٦	/ariables List-							 Cancel
	Name	Var. Kind	Allocat	Used	Data Type	Initial value	Comments	Flag
	ENABLE IP_COUNT TIMER1	VAR VAR VAR	%IX0.0.0 <auto> <auto></auto></auto>	*	BOOL FB Instance FB Instance			Global
	IP_COUNT.Q IP_COUNT.C	VAR V VAR	<auto> <auto></auto></auto>		BOOL INT			Direct Variable
	TIMER1.Q TIMER1.ET	VAR VAR	<auto> <auto></auto></auto>	*	BOOL TIME			Add
								Delete
								Edit
								Help

These outputs can be used elsewhere in the program.

We now need to connect the output TIMER1.Q to the UP input of the counter. Select the horizontal line connection icon in the toolbox and click the right button of the mouse to connect the spaces between the timer function block and counter function block.



	TIMER1	IP_COUNT
Row O	ENABLE TON Q	cu ci d-
Row 1	T#55 PT ET	R CV
Row 2		- PV
Row 3		

Now we need to assign variables to the inputs on the counter function block. Input R is a reset input of type BOOL and input PV is the preset value that the counter will count up to, it is of type INT.

Double click in the space to the left of input R, with the right mouse button and add a new variable called RESET.

Add/Edit Variables	×	
Variable  Variable Kind Variable Kind  Variable Kind : VAR  Data Type  C Elementary : BOOL	OK Cancel Help Memory Allocation	Assign the variable to the memory address IX0.0.0 and click OK
C FB Instance : CTD	<ul> <li>Assign(AT) :</li> </ul>	CHICK OK
C Array (0	% X0.0.1	
Comments	Init. Array	

Double click in the space to the left of input PV, with the right mouse button and add a new variable called PRESET.

Add/Edit Variables	×	
Variable PRESET Variable Kind Variable Kind Variable Kind : VAR  Data Type  Elementary : INT  FB Instance : CTD  Array (0., ) OF BOOL	OK Cancel Help Memory Allocation Auto Assign(AT) : %	Leave the variable as Auto allocated bu give it an initial value of 10. Click
Initial Value I I I Comments I I I I I I I I I I I I I I I I I I I	Init. Array	OK to complete.

Next Select a normally open coil icon  $\bigcirc$  in the toolbox and click the right button of the mouse at the output Q of the counter. Directly address this output to %QX0.0.0 by double clicking the right mouse button on the coil to open the variable menu.

Variables	
Name : %QX0.0.0	Direct Variable Comment OK
Variables List	Cancel
ENABLE VAR %IX0.0.0 * BOO	ta Type   Initial value   Comments Flag
PRESET VAR «Auto» INT	
	nstance Direct Variable
IP_COUNT.Q VAR <auto> BOO IP_COUNT.CV VAR <auto> INT TIMER1.Q VAR <auto> * BOO</auto></auto></auto>	Add
TIMERT.Q VAR «Auto» * BOU TIMERT.ET VAR «Auto» * TIME	
	Edit
	Help
TIMER1 ENABLE TON	IP_COUNT CTU %QX0.0.
7 0 - I - IN Q	cu o:( );
σl T#5S PT ET	RESET R CV
σ 2	PRESET PV
σ3 [']	

When the ENABLE input is switch ON the timer will count to five seconds. When 5s has elapsed the timer output Q will go high, this will trigger the counter input and the counter current value will be incremented by one. When the counter current value reaches ten, the counter output Q will go high and trigger the coil output %QX0.0.0

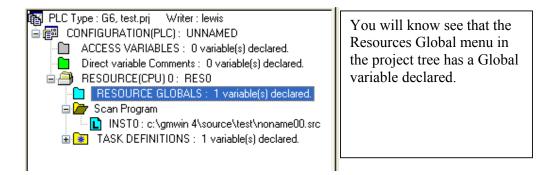
# Multiple-Source Programming

## **Global Variables**

Global Variables can be used in multiple Programs within a Project, unlike regular variables.

One way to declare a Global Variable is to assign it to VAR_EXTERNAL type, when entering the variable in a program.

Add/Edit Variables				×
Variable UP				OK Cancel
Variable Kind :	VAR VAR VAR_CONSTANT		Ē	Help
Data Type	-VAR_RETAIN VAR_RETAIN		lemory Alloc	ation
Elementary :	BOOL	- 6	Auto	
C FB Instance :	CTD	<b>I</b>	Assign(AT	Ŋ:
C Array (0.,	) OF BOOL	<u> </u>	%	
Initial Value				
			Init.	Array
Comments				
up input				



Another method is to double click on the RESOURCES GLOBALS menu in the project tree and 'Add' a Variable as usual.

obal Varia	bles						
Name	Var. Kind	Allocat	Used	Data Type	Initial	Commen	Close
UP	VAR_GLO	<auto></auto>	*	BOOL		up input	
							Add
							Delete
							Edit
							Help

Add/Edit Global V	ariables		×
Variable name : Variable Kind :	VAR_GLOBAL		OK Cancel
Data Type C Elementary : C FB Instance :	VAR_GLOBAL VAR_GLOBAL_CONSTANT VAR_GLOBAL_RETAIN BOOL CTD ) OF BOOL	•	Help mory Allocation Auto Assign(AT) :
Comments			Init. Array

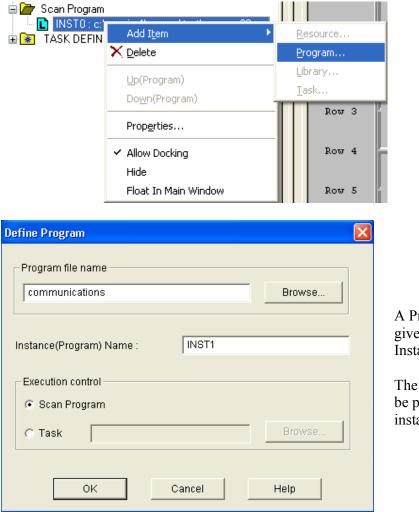
Enter a Variable as normal, however there are a couple of different Variable Kinds to select from. Var_Global: normal variable. Var_Global_Constant: Constant global variable Var_Global_Retain: Global variable with power down retain.

## **Multiple LD Programs**

It is possible to have more than one program in a GMWin Project. This allows the programmer to have a dedicated program to perform a specific function, for example one program can control the communications and another to control the logic of a machine process.

Variables declared in one program will not be available in other programs unless declared as Global.

To enter a new program right click on the project tree and select Add Item, Program.

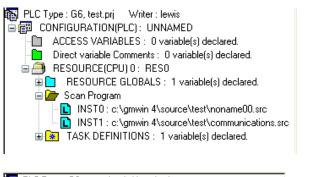


A Program must be given a name and an Instance name.

The programs will then be put in order of instance name.

<ul> <li>PLC Type : G6, test.prj Writer : lewis</li> <li>CONFIGURATION(PLC) : UNNAMED</li> <li>ACCESS VARIABLES : 0 variable(s) declared.</li> <li>Direct variable comments : 0 variable(s) declared.</li> <li>RESOURCE(CPU) 0 : RESO</li> <li>RESOURCE GLOBALS : 1 variable(s) declared.</li> <li>Scan Program</li> <li>INST0 : c:\gmwin 4\source\test\noname00.src</li> <li>INST1 : c:\gmwin 4\source\test\communications.</li> <li>TASK DEFINITIONS : 1 variable(s) declared.</li> </ul>	Now a blank program has been created, to select it as a Ladder Diagram, double click on it and select the option.
New Program         Program file name :       Communications.src         Language       Cancel         SFC < LD < IL < FBD < ST	<ul> <li>On selection of Language, click OK</li> <li>Then continue programming as before.</li> </ul>

The scan cycle of the Project will process the programs linearly. In this case INST0 will be processed before INST1. However the order can be changed.



PLC Type : G6, test.prj
Writer : lewis

CONFIGURATION(PLC) : UNNAMED

ACCESS VARIABLES : 0 variable(s) declared.

Direct variable Comments : 0 variable(s) declared.

RESOURCE(CPU) 0 : RES0

RESOURCE GLOBALS : 1 variable(s) declared.

Scan Program

INST0 : c:\gmw

Add Item

TASK DEFINITION

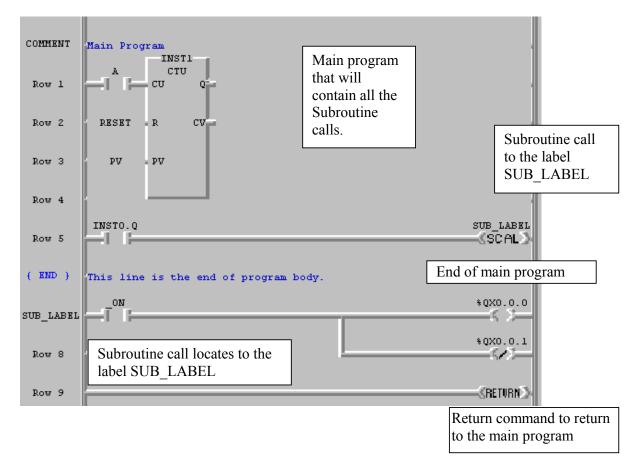
Delete

Up(Program)

To Change the order of the Programs, right click and select Up or Down (Program)

## Subroutines

A Subroutine is a grouping of logic that must be performed given a certain condition. To use Subroutines the programmer needs to use Sub Call and Return from commands. A Sub Call command is similar to a Jump command however there are some subtle differences. A Sub Call function is required to return to where it was called from and then it the PLC will continue linearly to process code. Whereas a Jump command skips code to and processing is continued at a given location.



To enter a Subroutine call, select the icon or use the hot-keys short cut.

<ret></ret>	Shft+F7
>>	Shft+F8
<scal></scal>	Shft+F9
To Arrow Mode	Ctrl+A

LD									×
R	[]]	+ +	杍	┨╒┠	╢	2	$\square$	{}	(/)
(5)	<b>(</b> R)	<b>(</b> P <b>)</b>	(N)	{F}	<del>[</del> FB}	(RET)	$\gg$	(s¢)	



Then double click on the SCAL function call and enter the Label to go to.

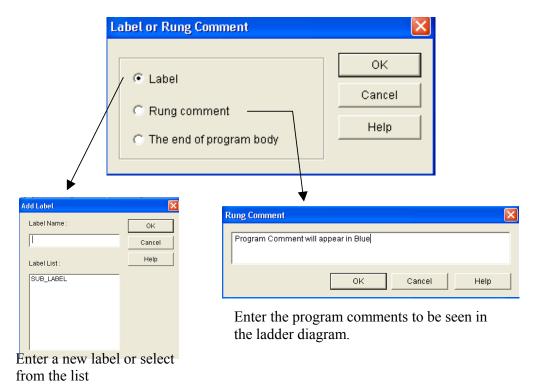
Either select a pre-registered label or enter a Label directly

To enter a Return call, select the icon or use the hot-keys short cut. A return call doesn't require a coil to operate it, simply enter it at the end of the subroutine (as shown in the example).

<ret> Shft+F7</ret>	LD
>> Shft+F8 <scal> Shft+F9 To Arrow Mode Ctrl+A</scal>	(c)

To enter Comments, an End of Main Program statement or a Label, double click on a Row number and a menu will pop up.

Select the desired option and enter the relevant data.



# **Downloading and Online Options**

# Compiling and Building

Prior to downloading the Project needs to be compiled and built. This prepares the Project for downloading to the PLC and checks for any errors.

Compile	Compile				
Hemory Reference Show Used I/O Check Duplicate Coils	The compile of	options prepare the programs for The programs are checked for errors			
← Previous Message ↓ Next Message	Compile: Makes just the one program you are currently working on. Build All: Makes the complete project.				
Clicking the Compile option brings up this message	Confirm Compile Would you like to compile? If you compile the program, onli after changing mode to stop an Do not show this dialog box	d writing program to PLC.			
Compile Source File : mixing_prog.src Lines compiled : 4 Lines compiled. Variables : 60 Bytes Status : 0 Errors, 18 warning OK		When the program has compiled it informs the user of any errors or warnings. Errors will stop the program from operating, whereas warnings just inform you of potential issues but do not impact on downloading the program.			
<ul> <li>Warning :Unreferenced local variable: : Warning : Unreferenced local variable: : Warning : Warning</li></ul>	e WATER_VALVE' is found. e 'YEAST' is found. e 'YEAST_ADDED' is found. e 'YEAST_INPUT' is found. e 'YEAST_VALUE' is found.				
	Cross Reference λ I/Ο λ D	uplicate Coil $\lambda$ Find $\lambda$ Communication /			

The compile messages are shown in the output window and can be cycled through using the message buttons in the Compile drop down menu.

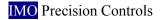
	Confirm Compile	All 🔀	
Clicking the Build All option brings up this message	changing mode	nline editing will be possible after to stop and writing program to PLC. v this dialog box next time	
	Yes	No	
		Build All	
Building the who program, links th project and show	ne entire	Output File : g7-baking-bread-161205.BN0	
memory used.		Lines compiled :	
It shows the Prog memory size, Da and upload file s along with the maximums it sho	ita size ize,	Status : Make completed	
percentage used.		Program : 4932 Bytes(8%, Max.68KB) Data : 4420 Bytes(14%, Max.32KB) Upload file : 3452 Bytes(5%, Max.68KB)	
		The size of program + upload file should be less than 68KB.	
		ОК	

### **GMWin Simulation**

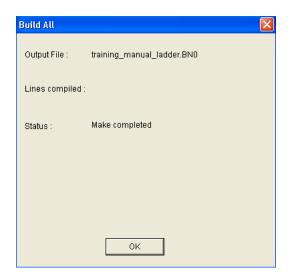
In GMWin there is a powerful simulation tool to verify your program without connecting a PLC. As GMWin is designed for use with the complete range of G-series PLC's the simulator looks like a slot and rack type PLC.

To open the PLC Simulator click on the icon with the right mouse button or select the simulator option from the Tools menu.

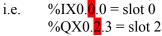
👫 Project Program Edit View Compile Online Debug	<u>T</u> ools <u>W</u> indow <u>H</u> elp
) 🔆 🕒 🔁 💾 📲 🗮 🗮 🗮 🗮 🗮	🁖 Library Manager
	Start Simulation
R PLC Type : G7. training manual ladder.pri Writer : lewis	🕈 Data Share



When the simulation software is started the project is built. Click OK to continue with the simulation.

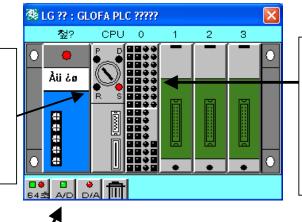


The simulation PLC is similar to a G4 or G6 PLC. When simulating a G4 or G6 program it is important to insert the I/O modules as addressed in the program. With a G7 program there will be a default 60 I/O card in slot zero and then the user can add modules where necessary.





To put the PLC in RUN mode use the right mouse button to switch the dial from STOP (S) to RUN (R). Click on the circle above R.

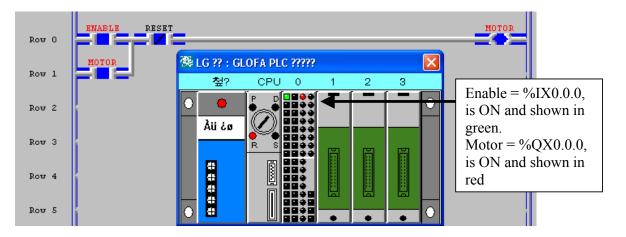


The inputs in the slot are square and the outputs are round. When ON the inputs go green and outputs go red

To insert a digital input, digital output, analogue input or analogue output module click the right mouse button and drag the module to the desired slot location. To remove a module click the right mouse button and drag it to the trash icon The simulated PLC will stay on top of all program, however you can change this option by left click the mouse on the blue bar at the top.



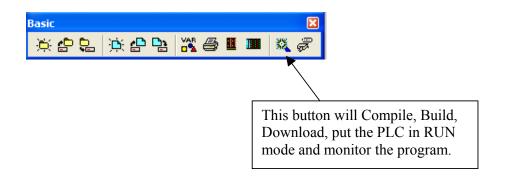
When the PLC is in RUN mode the program power rails are in blue as are the contacts and coils that are ON.



When the variable is ON the variable name also changes colour from black to RED.

# Downloading

Once the program has been compiled it is ready to be downloaded. However there is a button that will compile the program, build the project and download to the PLC.



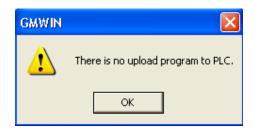
### IMO Precision Controls

🍇 Connect+Write+Run+Monitor On	Ctrl+R
F Connect	
<u>R</u> ead	
■J <u>W</u> rite	
🛄 Monitor On/Off	
PLC M <u>o</u> de	•
Rese <u>t</u>	•
Elash Memory	•
<u>P</u> LC Info	•
I/O <u>M</u> odules	•
I/O Eorcing	•
<u>N</u> etwork	•
Comm Info	
Online <u>E</u> dit	•
<u>E</u> SM	
I/O S <u>ki</u> p	
Fault <u>M</u> ask	
Initialize Special Module	

To connect to the PLC, either use the Connect option on the Online menu or press the connect button on the Basic buttons.

When connected the greyed out options will become available.

If there is no program in the PLC or the security option has been selected then this message will pop up when the user connects to the PLC.





The connect button in the toolbar at the top will now be appear pressed

<b>8</b> 2	Connect+Write+Run+Monitor On	Ctrl+R
Ş	Disconnect	
	<u>R</u> ead	
₽,	Write	
41-	Monitor On/Off	
	PLC M <u>o</u> de	+
	Reset	+
	<u>F</u> lash Memory	•
	<u>P</u> LC Info	•
	I/O <u>M</u> odules	+
	I/O <u>F</u> orcing	+
	Network	+
	Comm Info	
	Online <u>E</u> dit	•

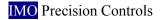
The grey-out online options are now available.

Once connected to the PLC it is now possible to:

- Write and read to the PLC.
- Switch the mode between RUN, STOP and PAUSE.
- Write and Read to the Flash memory.
- Check the PLC information, useful for if an error occurs in the PLC.
- Check the I/O modules connected.
- Enable the High Speed Networks.
- Edit the program ONLINE. This allows the user to make

Once the program is ready to be downloaded, select the Connect + Write + RUN + Start monitor button to program the PLC.

The user is then prompted for what they want to download.	Write Area © Basic Parame © I/O Parameter © HB Link Para © Redundancy I © Communicati © Special Parar (Special PID,	neter Parameter on Parameter Positioning Module and punter Module) rogram d Program rogram	For security, if yo do not wish for anybody to be abl to upload your program back off PLC, un-tick the	e
	ОК	Cancel Help	upload option.	
Build All         Output File :       g7-baking-bread-161205.BND         Lines compiled :         Status :       Make completed         Link success         Program :       4932 Bytes(8%, Max.68KB)         Data :       :4420 Bytes(14%, Max.32KB)         Upload file :       3448 Bytes(5%, Max.68KE)         The size of program + upload file should be less tha         OK	beir	g downloaded to t		3
ADD_MIX 0 B B 1 B A SALT F Row 0 A B Row 1 A B	BOOL <auto> BOOL %MW/100.6 BOOL <auto> B Instance <auto></auto></auto></auto>	VAR VAR VAR VAR	Used Comments * * * gram we can	



# **Basic Communication**

For communications with the IMO HMI's range (both PMU and MI-text terminal), the basic communication parameters will need to set. The communication parameters menu can be found in the parameter tab.

🖃 📲 👪 PAF	RAMETERS	
·····	Basic Parameters	
·····	Communication Parameters	

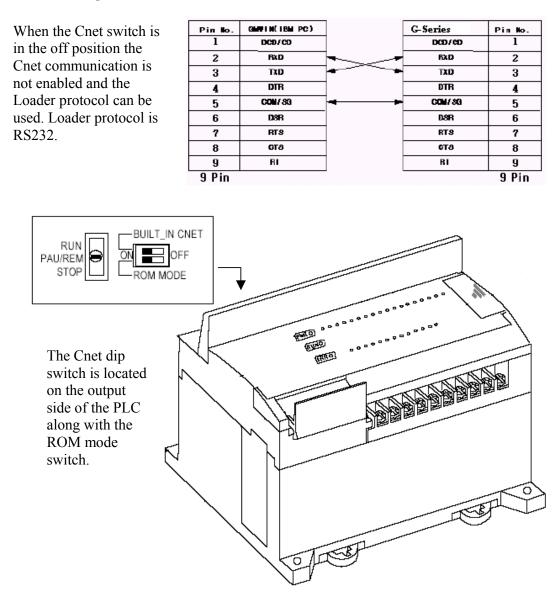
The parameters need to be set to match the HMI. Generally the HMI will be the master and the PLC the slave.

The key settings to ensure communication with the HMI are that the Baud rate, Stop bit, Start bit, Data bit and Station number. They all must match what has been entered in the HMI's communication settings.

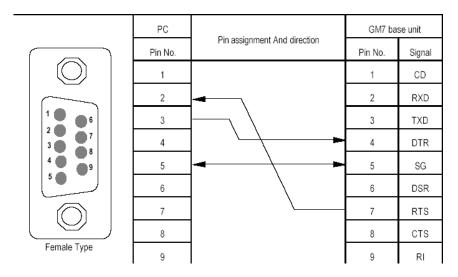
	Com	munication Parameter	
		Communication method	
		Station No.:	
		Baud rate: 19200 - Data bit: 8 -	
		Parity bit: None  Stop bit: 1	
		Communication channel     RS232C Null Modern or RS422/485	For communication
			with the HMI's this
		RS232C Modem (Dedicated Line) Initial command:     RS232C Dial-up Modem ATZ	is always set as
			RS232 / 422 /485
	F	Protocol and mode	
		Timeout in master mode: 500 ms	
		Dedicated	
HMI is		🔿 Master 📃 Read Status of Slave PLC List	
general the	Iy	Slave	
master,		Modbus	-
so ensu PLC is	re	C Master Transmission mode: ASCII	-
slave		C Slave	
		User defined	
		C Master	
		C Slave TIELDBUS	
		C Master	
		C Slave	
		OK Cancel Help	

# G7 PLC Communication Dip Switch Setting

On the G7 PLC the 9 pin socket is two communication ports in one, RS232 and at the flick of a dip switch RS422/RS485.

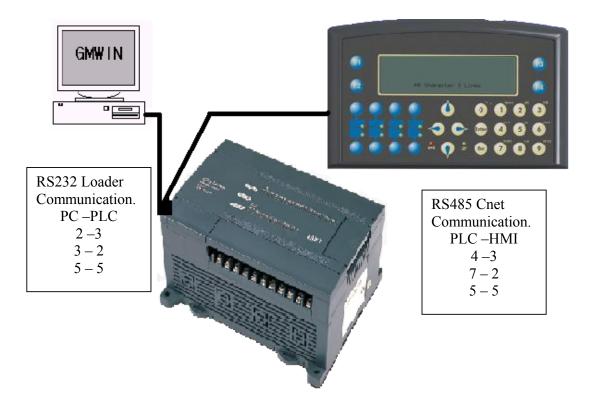


When the Cnet dip switch is in the on position the G7 PLC can communicated on a Cnet protocol. This is a RS422 / 485 protocol and has the following wiring



With the G7 PLC it is possible to communicate to both the HMI and PC. The Cnet dip switch needs to be in the on position and the HMI will communicate on Cnet protocol, whilst the PC will remain on Loader.

The advantage of this setting is that the PLC can be monitored whilst it is still 'live' communicating with the HMI.



# **Programming Tutorial Example – Washing Machine**



# Specification

Design a GMWin project with a ladder program to run a simple washing machine program. The washing machine has four inputs and six outputs. The washing machine will run two wash cycles: White wash and Colour wash.



#### Machine I / O

	INPUTS	OUTPUTS
1.	Start	Detergent + Water In
2.	White Wash	Fresh Water In
3.	Colour Wash	Water Drain
4.	Lid Closed	Drum Rotate
5.		Spin Dry
6.		Lid Open Lamp

### Washing machine cycles

The White Wash or Colour Wash is selected by a rotary switch (i.e. only one type can be selected).

	Wash Time	<b>Rinse Time</b>	Spin Time
White Wash	10 minutes	5 minutes	4 minutes
<b>Colour Wash</b>	7 minutes	4 minutes	3 minutes

# The Program Cycle

- 1. White Wash or Colour Wash is selected by a rotary switch (i.e. only one type can be selected).
- 2. When the lid is closed and the start button is pressed, Detergent + Water are pumped in for 1 minute.
- 3. The drum rotates for the specified wash time, depending on the type of wash selected.
- 4. The drum is drained for 1 minute.
- 5. Fresh water is poured in for 1 minute.
- 6. The drum is rotated for the specified rinse time, depending on the type of wash selected.
- 7. The water is drained for 1 minute.
- 8. The drum is rotated for the specified spin time, depending on the type of wash selected.
- 9. The lid open lamp will operate if the lid is opened, and the wash sequence cannot start.
- 10. If the lid is opened at any time during the wash cycle, the cycle is suspended and only continues when the lid is closed again.

# Beginning to Program

It is a good programming practice to map the known variables to addresses in the PLC before starting to program.

INPUTS	Address	OUTPUTS	Address
Start	%IX0.0.3	Detergent + Water In	%QX0.0.0
White Wash	%IX0.0.5	Fresh Water In	%QX0.0.1
Colour Wash	%IX0.0.6	Water Drain	%QX0.0.2
Lid Closed	%IX0.0.7	Drum Rotate	%QX0.0.3
		Spin Dry	%QX0.0.4
		Lid Open Lamp	%QX0.0.5

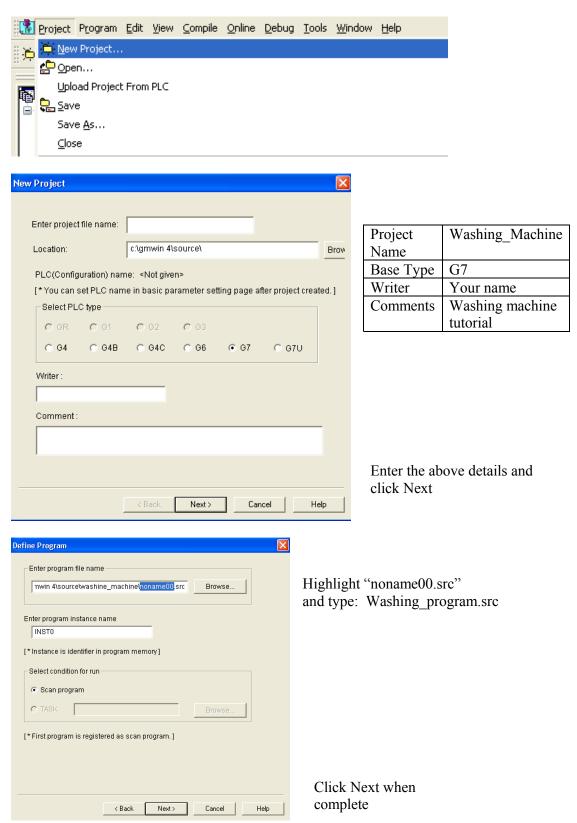
Open GMWin and create a new project.

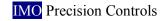


Double click on the icon **GMWIN**⁴ to start GMWin.

#### IMO Precision Controls

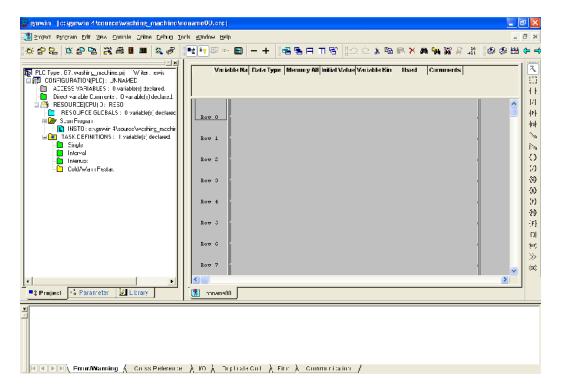
Set up a new project

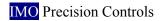




Program	
Enter program file noname00.src	
Select language	
C SFC @ LD C IL C FBD C ST	
Select kind of program	
Program Block     O Function Block     O Function	
Fun/FB name :	
Output data type :	
* First program is registered at program block. ]	
Enter program comments	
<b>1</b> '	
	Select Ladder (Ll

Now the project has been made we are ready to create a Program.





### 1. White Wash or Colour Wash is selected by a rotary switch.

Select the NO contact icon **H** in the toolbox and click the left button of the mouse on position row '0' and column '1' in the LD window.

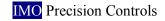
🏦 c:\program files\gmwin\source\noname00.src	_ 🗆 ×
Row 0	
Row 1	
Row 2	
Row 3	
Row 4	-
Row 5	

Assign a variable name to the NO contact and allocate the relevant IP address to the variable name. (As mapped out before programming)

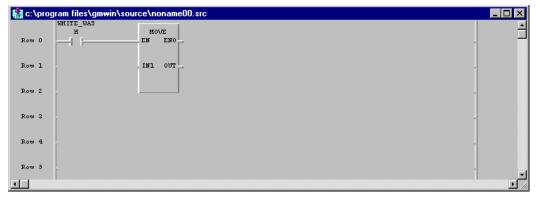
Variable Kind : VAR Help Data Type O Elementary : BOOL O FB Instance : CTD O FB Instance : CTD O Array (0) OF BOOL O F BOOL O FB Instance : Init. Array Comments	ol	OK Cancel			WHITE_WASH	Edit Variables ariable Name : WH ariable Kind
Elementary: BOOL     Auto     FB Instance:     CTD     Array     (0. ) OF     BOOL     S      Mitial Value      Initial Value      Init. Array				•	VAR	ariable Kind :
FB Instance :     CTD     Array     (0. ) OF BOOL     Solution     Initial Value     Init. Array		cation ——	- Memory Allo			ata Type
Array (0			C Auto	Ŧ	BOOL	Elementary :
Initial Value		AT) :	<ul> <li>Assign (</li> </ul>	v	CTD	🔿 FB Instance :
Init. Array		i)	%IX0.0.9	7	) OF BOOL	O Array (0
						itial Value
Comments		Array	Init			
Commente						omments
	-					

The same timer can operate from different preset times. This can be achieved by moving a time literal into a variable, which can then be used as a preset time.

Select [F] in the toolbox using the mouse and insert a **MOVE** function after the NO contact.

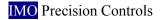


ime/Date Function All	System F 📕 🕨	- Set input number	ок
MOVE		Max number:	Cancel
MCS MCSCLR MID MIN MOD		Required number:	Help
MOVE	<b>•</b>		
unction information Comment: Copy data			
Comment: Copy data	OVE ENO •	<u>ح</u>	
Comment: Copy data		*	
Comment: Copy data	ENO •	*	



Double Click on the area to the left of **IN1**. The Variables dialog box will be activated. Enter the time expression for 10 minutes.

Variables	
Name : Direct Variable Comment	ОК
Variables List	Cancel
Name         Var. Kind         Allocat         Used         Data Type         Initial value         Comments           WHITE_WASH VAR         %IX0.0.5         BOOL         BOO	Flag
	Global
	Direct Variable
	Add
	Delete
	Edit
	Help



Double Click on the area to the right of **OUT**. Enter the Variable name **WASH_TIME**, as a **TIME** Data type.

Variable Kind	WASH_TIME			Cancel
Variable Kind :	VAR	<u> </u>		Help
Data Type			Memory All	ocation ——
Elementary:	TIME	•	<ul> <li>Auto</li> </ul>	
O FB Instance :	SINT	-	C Assign	(AT) :
O Array (0.,	) OF USINT			
Initial Value				
	TIME_	OF_DAY AND_TIME G	Ini	it. Array
Comments				

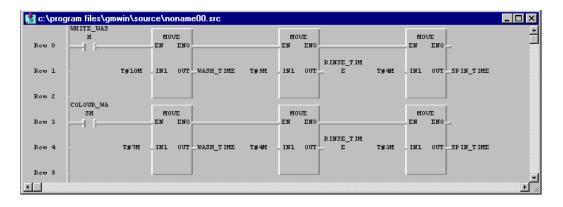
On the rising edge of the NO contact activation the time value of 10 minutes (T#10M) is moved into the variable named **WASH_TIME** with a time data type.

🎇 c:Apro	am files\gmwin\source\noname00.src	- 🗆 ×
	HITE_WAS H MOVE	-
Row 0	EN ENO	
Row 1	T#10M IN1 OUT WASH_TIME	
Row 2		
Row 3		
Row 4		
Row 5		J
		• //

The other two variables for the White Wash Cycle can be linked in series to the NO contact.

🕌 c:\p	orogram files\gmwin\sou	urce\noname00.src		_ 🗆 ×
Row O	WHITE_WAS H	NOVE EN ENO	MOVE EN ENO	MOVE EN ENO
Row 1	L T#10M	INI OUT WASH_TIME	T#5M IN1 OUT E	T#4M IN1 OUT SPIN_TIME
Row 2	:			
Row 3	1			
Row 4	•			
Row 5	;			

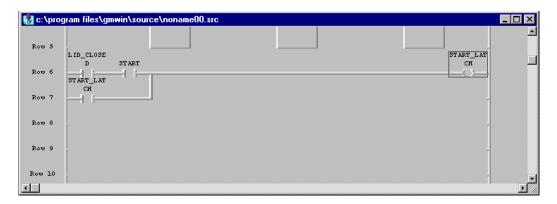
Repeat for the Colour Wash Cycle. Move the relevant time values into the WASH_TIME, RINSE_TIME and SPIN_TIME Variables.



# 2. When the lid is closed and the start button is pressed, Detergent+ Water are pumped in for 1 minute

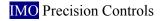
Create a latch using the lid closed and start inputs. Create three variables, LID_CLOSED, START and START_LATCH. LID_CLOSED and START are allocated to the relevant inputs and START_LATCH is auto allocated.

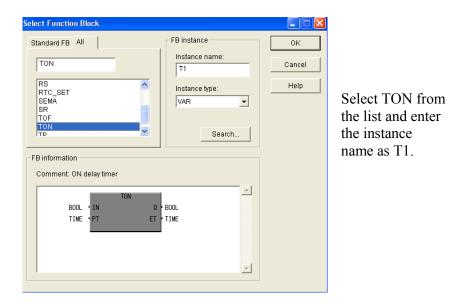
Insert an output coil by clicking on the icon with the mouse and inserting it to the program on the right hand side. The same Add / Edit Variables menu will pop up so that you can name the variable.



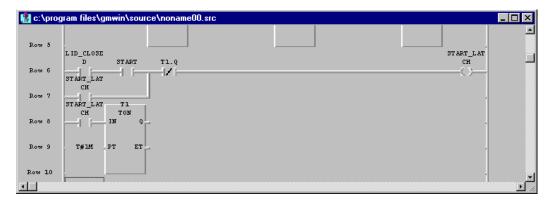
Use the **START_LATCH** bit to operate an ON delay timer for 1 minute.

To insert an On Delay Timer select the Function Block icon ^[f] using the mouse.



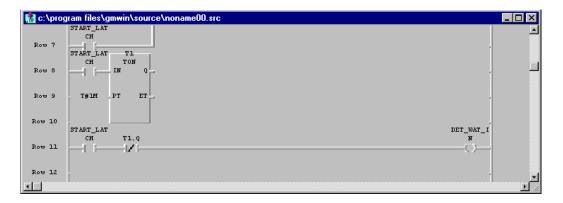


Name the **TON** timer as **T1**. Enter the time value **T#1M** as the Preset Time (PT). Use the T1 timer output bit, **T1.Q** to break the **START_LATCH**.



Also use the **START_LATCH** bit, in series with a NC output bit of the **T1** timer, **T1.Q**, to operate the Detergent and Water in output, **DET_WAT_IN** assigned to **%QX0.0.0**.

To insert a Normally Closed (NC) coil click on the icon 4/1 with the mouse and insert it into the program where required.

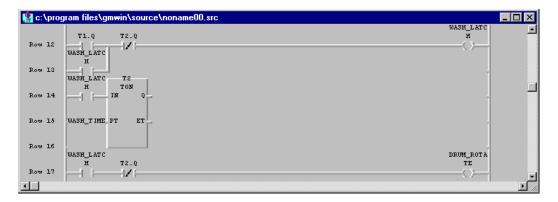


Link the **START** bit in series with the colour/white wash selection at the start of the program. The selection of program will only happen at one point in the wash cycle. If the wash selector is changed during the program, the settings for the current cycle will remain the same and not change until the next cycle.

	WHITE_WAS			_	e00.src			_			-	_		
	н [—]	ST ART	MO	VE			MO	VE			M	)VE		
Row O		-1 1-	EN	EN0			EN	ENO			EN	ENO	-	
Row 1		T#10M	INI	0.1777		T# 5M	INL	OUT	RINSE_TIM	T# 4M	ואו	0.1777	SPIN TIME	
KOW I	j	T#IOM	TNT	UUT	_wash_t ime	T# 5M	TNT	OUT	E	T# 491	TNT	UUT	TIME	
Row 2														
	COLOUR_WA												1	
	SH	ST ART	MO				MO					IVE		
Row 3			EN	EN0			EN	ENO			EN	ENO	7	
									RINSE_TIM					
Row 4		T#7M	INI	our	WASH_T IME	T# 414	INL	OUT		T#3M	INI	OUT	SPIN TIME	
Row 5	-						_						1	

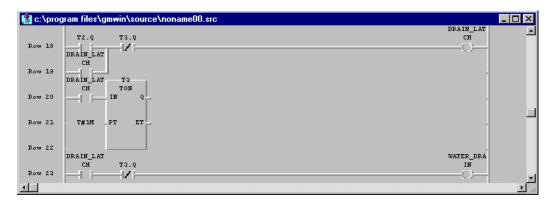
# 3. The drum rotates for the specified wash time, depending on the type of wash selected.

After the detergent and water have been pumped in for 1 minute, the output bit from the **T1** timer, **T1.Q**, can be used to start the drum rotation timer and output. Create a latch called **WASH_LATCH**, and use this latch to operate the timer and the output. With two options for the length of time that the drum rotates, Colour and White wash, the **WASH_TIME** variable is used as the Preset Time (PT) for the On delay timer.



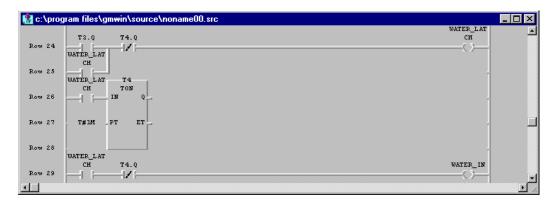
### 4. The drum is drained for 1 minute.

The resulting bit from the wash timer, **T2.Q**, is used to latch a bit to operate the drum drain output and a 1 minute delay.



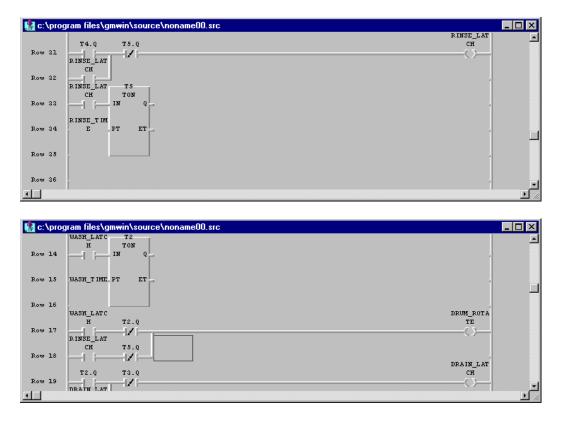
### 5. Fresh water is poured in for 1 minute.

The resulting bit from the drain timer, **T3.Q**, is used to latch a bit to operate the fresh water in output and a 1minute delay.



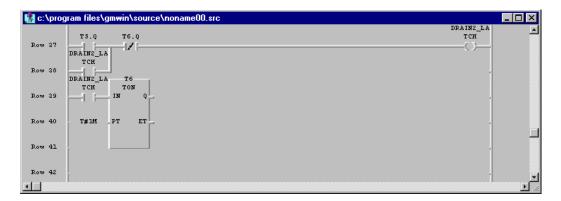
# 6. The drum is rotated for the specified rinse time, depending on the type of wash selected.

The resulting bit from the fresh water timer, **T4.Q**, is used to latch a bit to operate the drum rotate output and the **RINSE_TIME** delay. As the **DRUM_ROTATE** output is already used in row 17, and can only be assigned to an output coil once in the program, the drum rotate condition is linked in parallel with the output on row 17.

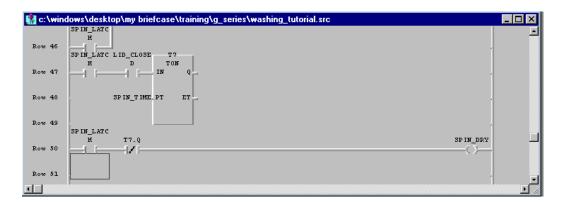


### 7. The water is drained for 1 minute.

The in the program resulting bit of the rinse timer, **T5.Q**, is used to initiate the drain timer and output. Again the drain output has been used previously so the initiating bit is used in parallel in row 24.



8. The drum is rotated for the specified spin time, depending on the type of wash selected.



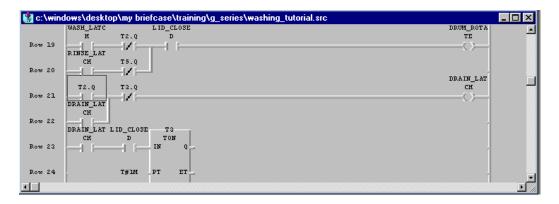
# 9. The lid open lamp will operate if the lid is opened, and the wash sequence cannot start.

Use a NC condition of the lid switch input to operate an output lid open indicator.



# 10. If the lid is opened at any time during the wash cycle, the cycle is suspended and only continues when the lid is closed again.

The lid switch input could also be placed in series with any line where an action is performed. This will stop the cycle at the point where the lid is opened and only restart again when the lid is closed.

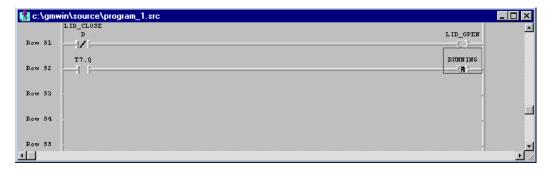


To ensure the **START** button does not restart the program again during the cycle, insert a NC contact on row 8 after the **T1.Q** NC contact. Create a new variable **RUNNING** and assign to the contact.

Insert a line and add a NO contact with variable **T1.Q**. Add a set coil with the variable **RUNNING**.

😭 c:\gmv	vin\source\program_1.src		_ 🗆 🗡
Row 7	LID_CLOSE	START_LAT	•
Row 8	D START T1.0 RUNNING	<del>сн</del> С	
Row 9		RUNN ING	
Row 10	START_LAT LID_CLOSE T1 CH D TON	(S)	
Row 11			<u>ت</u>

After the last time, **T7**, add a NO contact **T7.Q** operating a reset coil with the variable **RUNNING**.



Place the **RUNNING** bit in series with the recipe selection, to inhibit changes to the preset time during the cycle.

### IMO Precision Controls

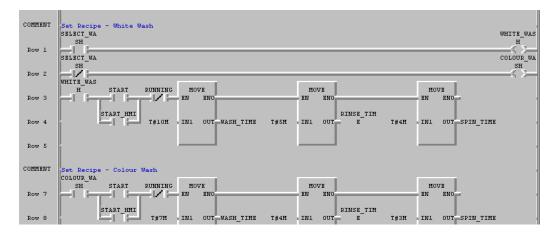
Rung comments can be added by double clicking on the row number on the left power rail, selecting the rung comment option and entering the comment in the dialog box.

Double clicking brings up this menu option. Select Rung comment and click OK

Label or Rung Comment	×
Class	ОК
C Label	Cancel
C The end of program body	Help
· · · · · · · · · · · · · · · · · · ·	

#### Enter the comment and press ok

Cot Desinia - White Week			
Set Recipie - White Wash			
,			
	ок	Cancel	Help



# Compile and Build the Project

Once the program has been written has been written Compile the program by clicking on the icon with the mouse.

Confirm Compile	×					
Would you like to compile?						
If you compile the program, online editing will be possible after changing mode to stop and writing program to PLC.						
🥅 Do not show this dialog box next time						
Yes No						
Compile						
Source File : washing_program.src						
Lines compiled : 154 Lines compiled. Variables : 192 Bytes						
Status : Compiled Successfully.						
ок						

### Simulate the Program

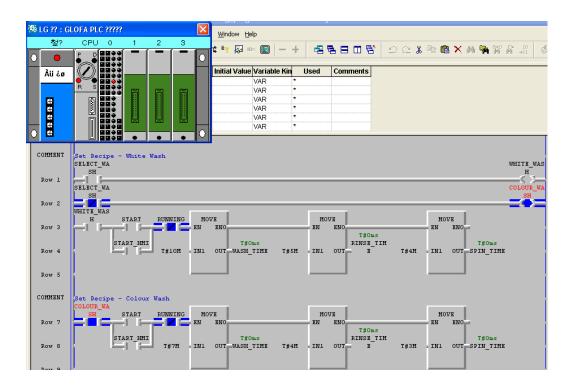
To simulate the program click on the icon with the mouse. This will automatically build the program and if there are no errors open up the simulation software.

The simulated PLC looks represent the G4 PLC. This is because GMWin is used for the complete range of G-series PLCs.

Switch the Simulation PLC from Stop (S) to Run (R) and now you can simulate the program. Input and output cards can be added to the simulation PLC, the inputs are square and outputs round, when active they are red.

On the program the power rails are active and shown in blue, as are active coils and contacts. The active variable names also go red. Timer and counter current values are shown in green.

### IMO Precision Controls



# Download the Program to the PLC

Connect the PLC to the computer serial port using the KIC-50A programming cable. Click on the connect + write + run + monitor  $\frac{1}{2}$  icon with the mouse. This will then bring up the write to PLC options.

W	/rite 🛛 🔀								
	Area								
	C Basic Parameter								
	C I/O Parameter								
	🔿 HS Link Parameter								
	C Redundancy Parameter								
	C Communication Parameter								
	C Special Parameter								
	(Special, PID, Positioning Module and High-Speed Counter Module)								
	C Program								
	🔽 Upload Program								
	Parameter and Program								
	🔽 Upload Program								
	C Upload Program								
[	OK Cancel Help								

The upload option enables the program to be read from the PLC by a user without the original project. If you do not want other user to have access to your program, un-tick the box

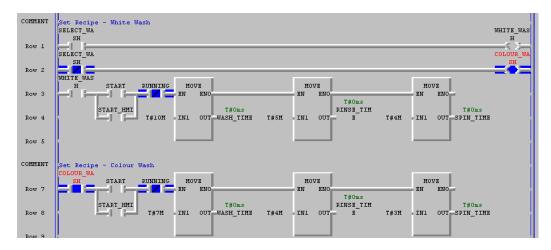
The Project is then built before downloading to the PLC.

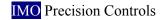
	Build All 🛛 🔀	
	Output File : washing_machine.BN0	
Once the	Lines compiled :	
program is built a menu	Status : Make completed	
pops up displaying the size of the	Link success Program : 5552 Bytes(8%, Max.68KB) Data : 4552 Bytes(14%, Max.32KB) Upload file : 5656 Bytes(9%, Max.68KB)	
project.	The size of program + upload file should be less than 68KB.	Click OK to download project

The project then downloads to the PLC.

Write to PLC	
Percentage of Frames Sent(%):	
36%	
Cancel	

Just like in the simulator, when monitoring the live program, it can now be seen with power active to the rails.





### **IMO Precision Controls**

1000 North Circular Rd Staples Corner London NW2 7JP Tel: +44 (0) 208 452 6444 Fax: +44 (0) 208 450 2274 Email: sales@imopc.com or applications@imopc.com Web: www.imopc.com

# Tutorial 1 – IMO Greenhouse

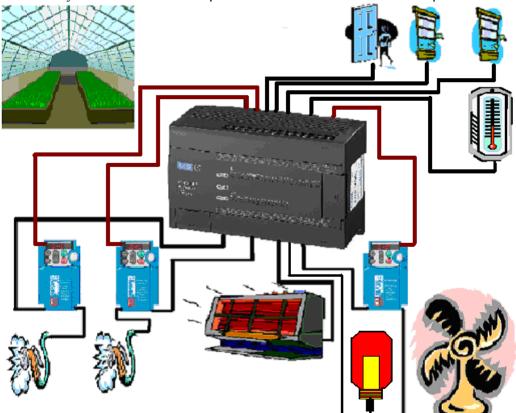
The plants in the greenhouse must be watered every 3 hours. There are two IMO Jaguar drives which control two pumps supplying the plants with water. They must not come on at the same time.

If the greenhouse gets too hot then a cooling fan must be switched on. Likewise if the greenhouse gets too cold then the heater must be switched on. The fan and heater cannot be on at the same time. The PLC has an On/Off input from a temperature controller to regulate the temperature.

The water sprinkler must not switch on if the door is open and the iSmart must display if either of the two windows are open.

The customer in time would like in time, to add a HMI to the system but for now he would like to log how many times the door has been opened.

There is also a feed back system from the drives to inform the PLC when a drive is down. If any of the drives are not operational then this will flash the lamp.

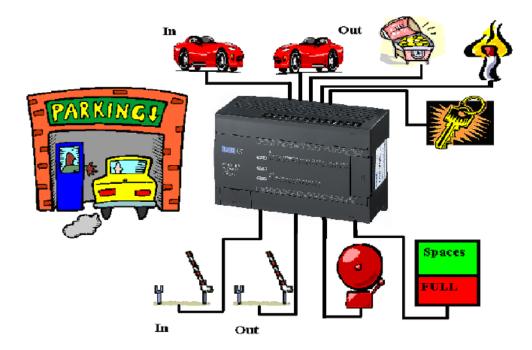


# Tutorial 2 – IMO Car park

Design a simple car park system to monitor and control the number of cars in the IMO Car Park.

The maximum capacity of the car park is 20 cars. The entrance is separate to the exit at which there will be barriers. The entrance barrier will rise when the driver has pushed the entrance button and there are available spaces. The exit barrier will rise on the car approaching and having settled an appropriate fee. The exit barrier will automatically rise when there is a fire alarm, the entrance gate will not open and the bell will sound.

To reset the fire alarm the car park attendant must use a key.



# **Topics Covered in the Advanced Course**

- SFC and LD Programming
- Tasks
- Inserting Libraries
- Communications
  - DeviceNet
  - $\circ$  Modbus
  - o RNet
  - o Dedicated
  - o Profibus
- Expansion Units
  - o Analogue
  - Expansion I/O
  - Communication Modules
- Slot and Rack PLCs
- Creating User defined Functions / Function Blocks
- Smart I/O

#### **IMO Precision Controls**

1000 North Circular Rd Staples Corner London NW2 7JP Tel: +44 (0) 208 452 6444 Fax: +44 (0) 208 450 2274 Email: sales@imopc.com or applications@imopc.com Web: www.imopc.com