MoxGRAF User Guide

0809-901-2302

Preface

Scope of the User Guide

This guide has been designed to assist the user to configure and create program code for a MOX system application. It is expected that the reader is an engineer or similar with an understanding of the operating and programming requirements of the intended MOX system components.

MoxGRAF can be used to create program code for the following MOX devices:

- MOX Open Controller
- MOX Unity Field Controller
- MOX IoNix Controller

Related Documents

A MOX system contains a collection of MOX equipment and several software packages. For this reason, a number of related documents should be read in conjunction with this guide.

The related documents are noted below:

- MOX Open Controller User Guide
- MOX Unity User Guide
- MOX IoNix User Guide
- MoxIDE User Guide

Conventions Used

When you see the "exclamation mark" icon in the left-hand margin, the text to its immediate right will be a special note. Please ensure that you read this information to increase your understanding of the systems operation.

When you see the "stop sign" icon in the left-hand margin, the text to its immediate right will be a warning. This information could prevent injury loss of property or even death (in extreme cases). It is very important that you stop and read this information and ensure that you have complete understanding before continuing with the procedures.

Contents

Figures

Tables

1 Introduction

1.1 Concepts

MoxGRAF's main use is for configuring the MOX Controller for execution of program code during run time. This includes the creation of the program code, its structure of execution and how the program code interacts with I/O (internal or external).

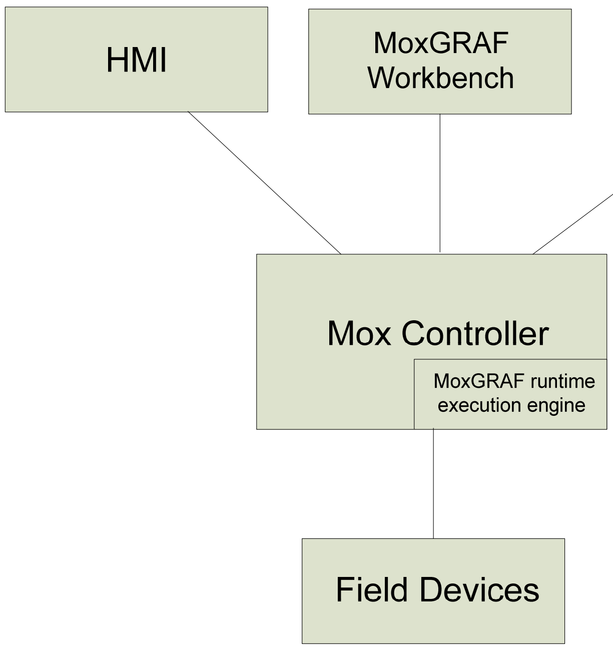


Figure 1 Concept Model

Using the concept model as a guide, it can be seen that there are multiple configuration points that are required for a MOX System.

MoxGRAF is used to create program code, which incorporates a program structure, an order of execution and communication with I/O modules, either onboard or external.

MoxIDE is used to configure the operational characteristics of the controller and I/O modules. This involves the configuration of communication ports, additional operational extra such as security, web server access, communication protocol configurations. MoxIDE is also used to configure the connectivity parameters between MOX controllers and connected I/O modules. This enables the user to manage the overall MOX system architecture.

1.2 Overview

MoxGRAF consists of a set of Soft Logic Programming Tools - called the Workbench.

MoxGRAF is based on the only internationally recognized industrial standard for industrial automation control languages, the IEC61131-3, and fully supports all five of the languages:

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

MoxGRAF also supports the Flow Chart programming language. Additionally, for the ultimate in power and flexibility, MoxGRAF supports functions and function blocks written in IEC61131-3 languages.

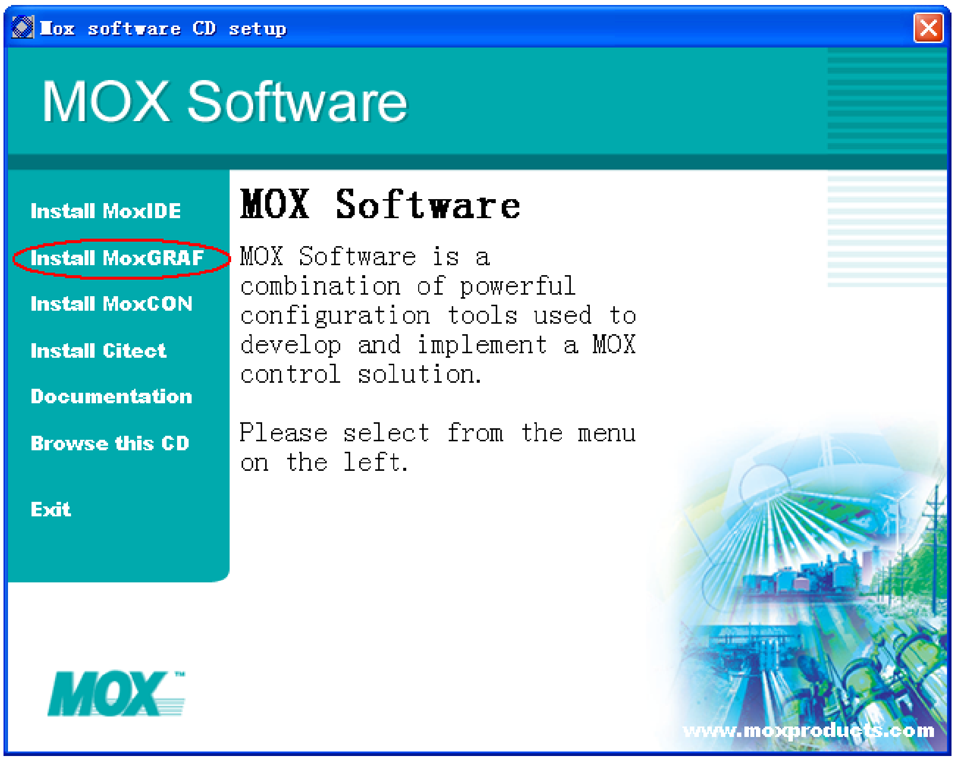
The following chapters describe how to get a MOX controller up and running quickly with MoxGRAF. The information contained in this document should provide adequate information to configure a basic system, but if further information is required please consult the comprehensive MoxGRAF Online Help.

1.3 Installation

Insert the MOX Software CD into the disc drive. This action should initiate the Autorun functions on the CD to bring up the screen shown below. *Please be patient as it may take a few moments*.

If the Autorun is not initiated then use Explorer to browse the CD and run the *Autorun.exe* file in the root directory of the CD.

From the installation options on the menu, select the option to Install MoxGRAF and follow the pop-up windows. Once MoxGRAF is installed you will need to restart your PC.



MOX Software

MOX Software is a combination of powerful configuration tools used to develop and implement a MOX control solution.

Please select from the menu on the left.

2 MoxGRAF Project

2.1 Project Management Workbench

The first step in using MoxGRAF is to start the MoxGRAF application from within **Start | Programs | MOX Products.**

When starting MoxGRAF, the Project Management window is the first screen displayed, refer to Figure 2.

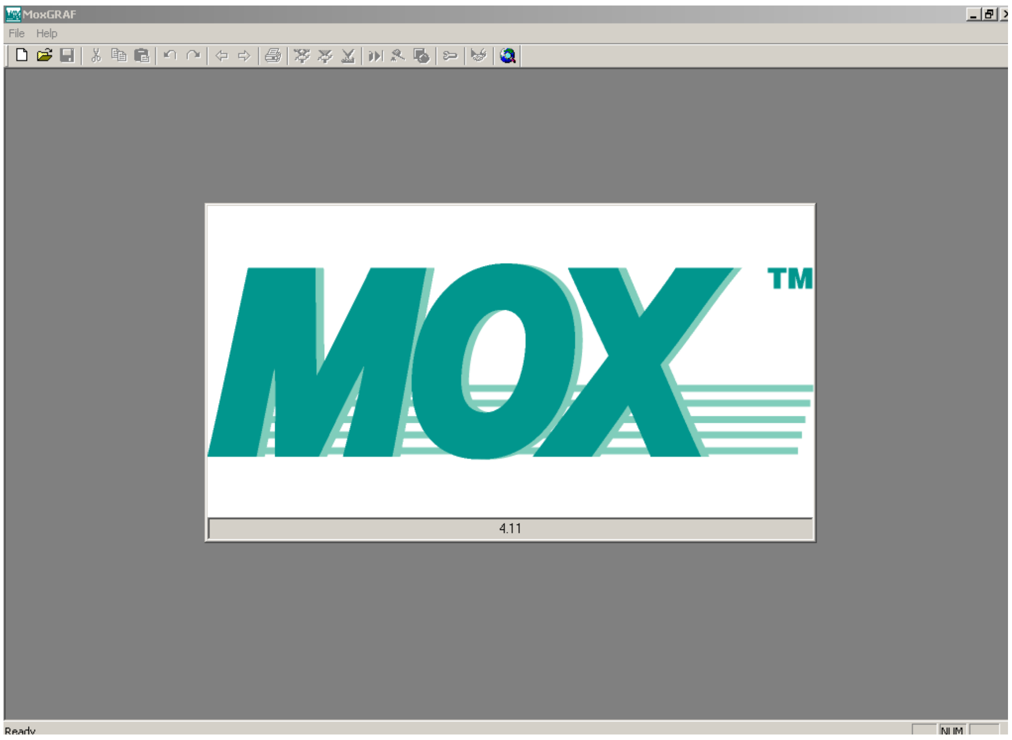


Figure 2 MoxGRAF Project Management Workbench

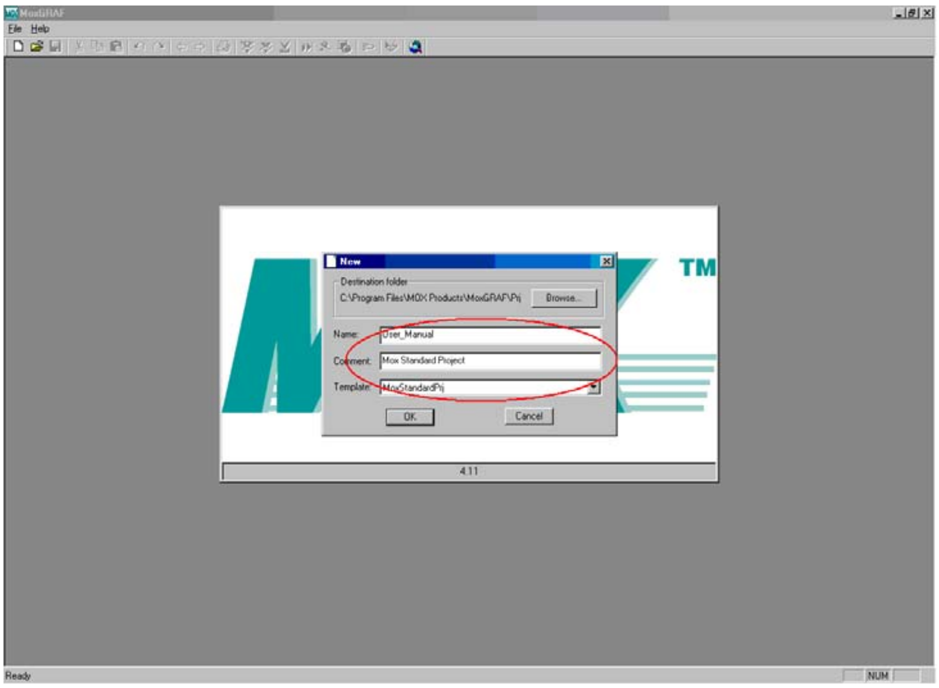
2.2 Create a New Project

To create a new project, select **File | New** from within the MoxGRAF Projects Management window.

Enter a name for the new project you wish to create. The name must be less than 32 characters and consist only of alphanumeric characters. It is also recommended you use a meaningful name and one that follows a naming standard.

*You must select a **Template** before you can continue with programming. Among the template options, MoxRTUStandardPrj is used for MOX Unity and IoNix project while MoxStandardPrj is used for MOX Open Controller (OC) project. Ensure that you have changed to the correct template for the desired controller type before selecting **OK**.*

Upon creation of this new project, a directory entitled the same as the project title will be created and placed under the MoxGRAF directory structure where it can be easily accessed.



Destination folder
C:\Program Files\MOX Products\MoxGRAF\Prj

Name: User_Manual

Comment: Mox Standard Project

Template: MoxStandardPrj

Figure 3 MoxGRAF Project Creation

A MoxGRAF project is a collection of programs (programs, functions, function blocks, etc.) used to control a process. A project corresponds to one complete process run on a target MOX controller.

2.3 Open an Existing Project

From the **File** menu within the MoxGRAF Projects Management window, select the **Open Project/Library** option. The pop up window will show the directory of the project opened last time, or **"My Documents"** which is the default directory.

To open another available project, return to parent directory or change to **"/Prj"** of MoxGRAF installation folder, and open the directory of desired project. This process will display all project data in the window below the prompt. Select the **PRJlibrary.mdb** file from this window and click on the **"Open"** button.

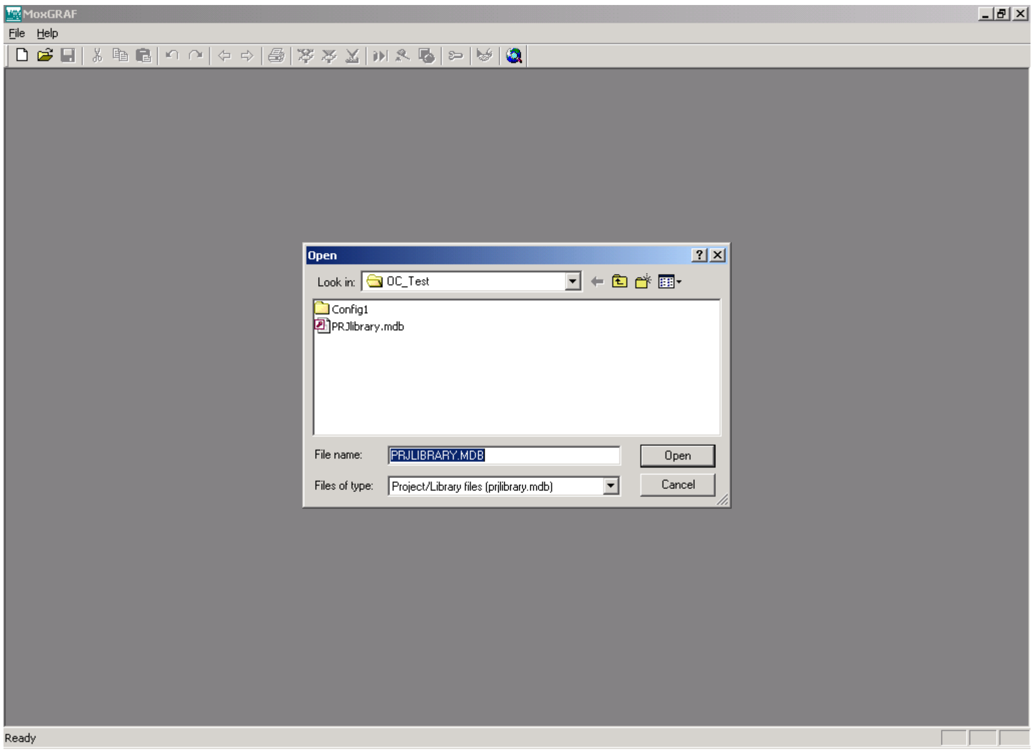


Figure 4 MoxGRAF Program Definition

2.4 Define Variables

Defining the variables is achieved by selecting option **Project | Variables** refer to Figure 5. Open out the variable tree in the far left window. This will display a list of options for allocating variables.

To create a new variable, simply double click in the larger window and enter in the correct information at the prompts, ensure the desired variable position is selected in the left window. All variable properties are changeable by selecting them and choosing the ones desired, i.e. Scope-Global, Internal, Constant, etc. Continue to change the allocated attributes of this newly created variable by clicking on the associated parameter and selecting the desired option.

The following chapter describes the process of addressing variables in more details.

Although it is not essential, it is highly recommended that all of the physical variables be defined at this stage, prior to configuring the I/O channels. This will enable all variables to be wired at the same time and will also enable quick and easy program creation.

It is not necessary to define all of the variables associated with the application at this time. If desired, variable definition may take place at any time the programmer chooses.

Ensure that you save your configuration information to the project file correctly.

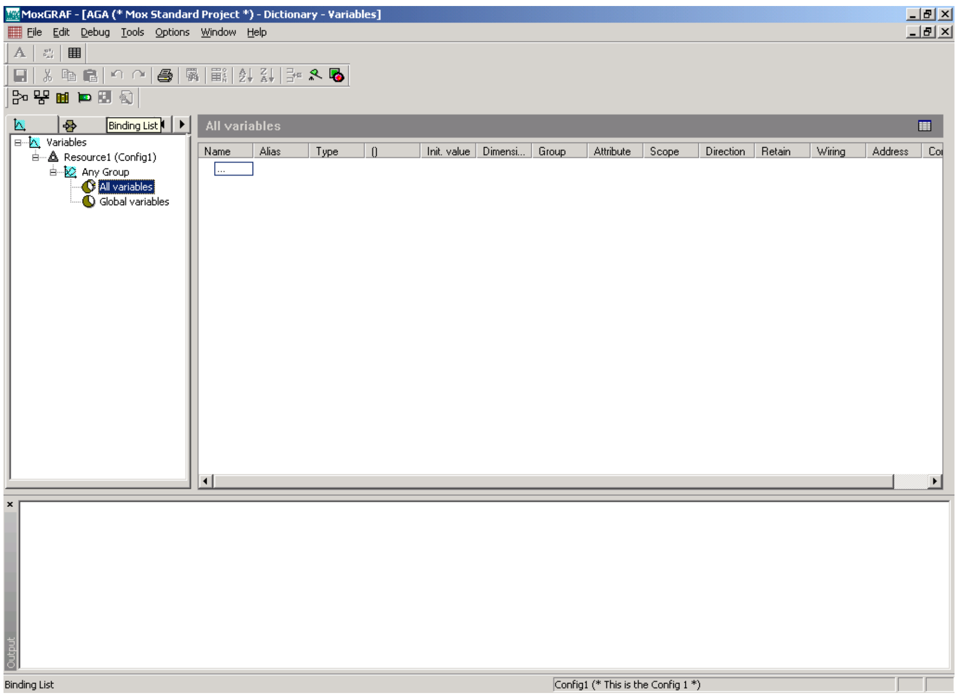


Figure 5 MoxGRAF Dictionary (Variable Definition)

Always give a meaningful variable name and description (comment) to each variable. This will aid in use, debugging and for future updates of the application.

2.5 Variable Addressing

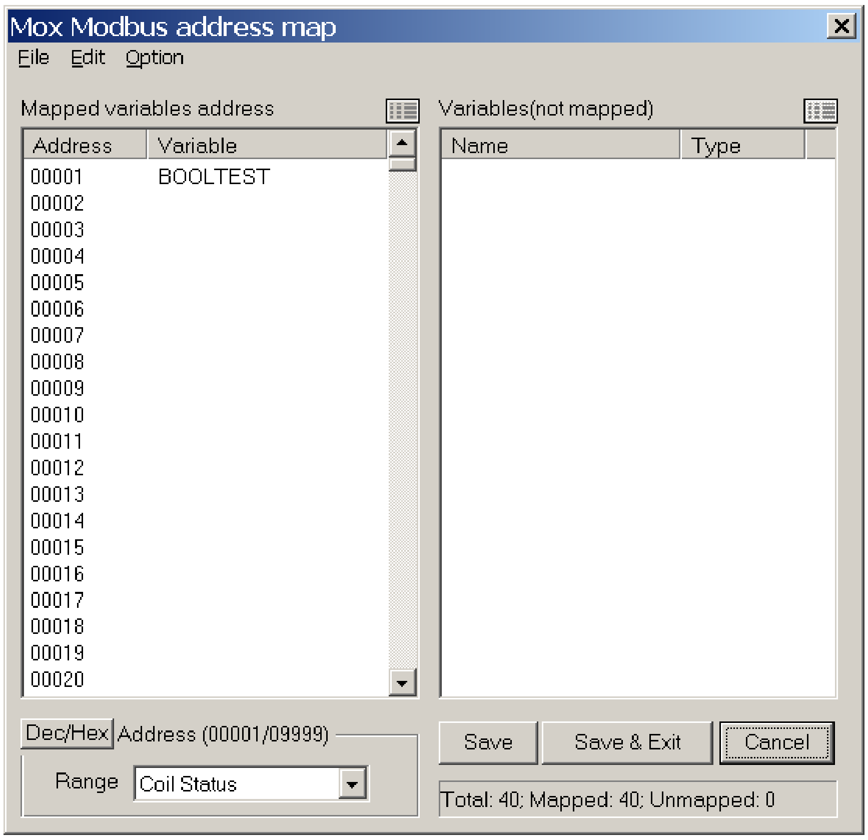
The address is used when a master device, e.g. HMI, is used to monitor the connected slave controller. There are a number of ways of allocating an address to newly created variables.

1) Use the MOX MODBUS Address Map to allocate variables for MODBUS communications. For more information on how to map MODBUS variables to correct addresses, please refer to *MODBUS Configuration Guide*.
2) Use the DNP 3.0 Setting address utility to allocate variables for DNP3.0 communications. For further information, please refer to *DNP 3.0 Configuration Guide*.

2.5.1 MOX MODBUS Address Map

Ensure the Variables Dictionary is closed before using MOX MODBUS Address Map utility.

The MOX MODBUS address map is to assign each variable with a MODBUS address. To open the MOX MODBUS address map utility, select **Tools | Mox Modbus address map**, a window similar to that displayed in Figure 6 will appear.



Address	Variable
00001	BOOLTEST
00002	
00003	
00004	
00005	
00006	
00007	
00008	
00009	
00010	
00011	
00012	
00013	
00014	
00015	
00016	
00017	
00018	
00019	
00020	

Name	Type

Address (00001/09999)

Range: Coil Status

Total: 40; Mapped: 40; Unmapped: 0

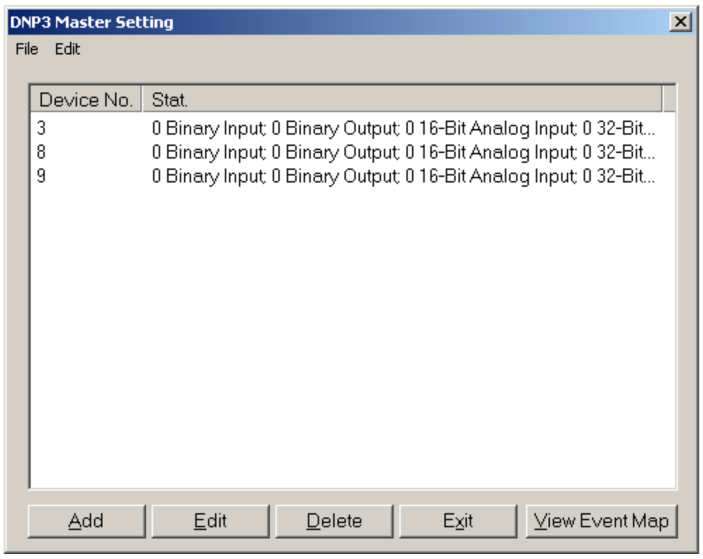
Figure 6 MOX MODBUS Address Map

If the focus is not on the Resource window, the **"Mox Modbus address map"** option will be greyed out. To ensure that the option is accessible, please make sure the resource window is selected.

2.5.2 DNP 3.0 Setting

2.5.2.1 DNP3 Master Address Settings

To open the DNP3 Master Setting, select **Tools | DNP3 Master Setting**, a window similar to that displayed in Figure 7 will appear.



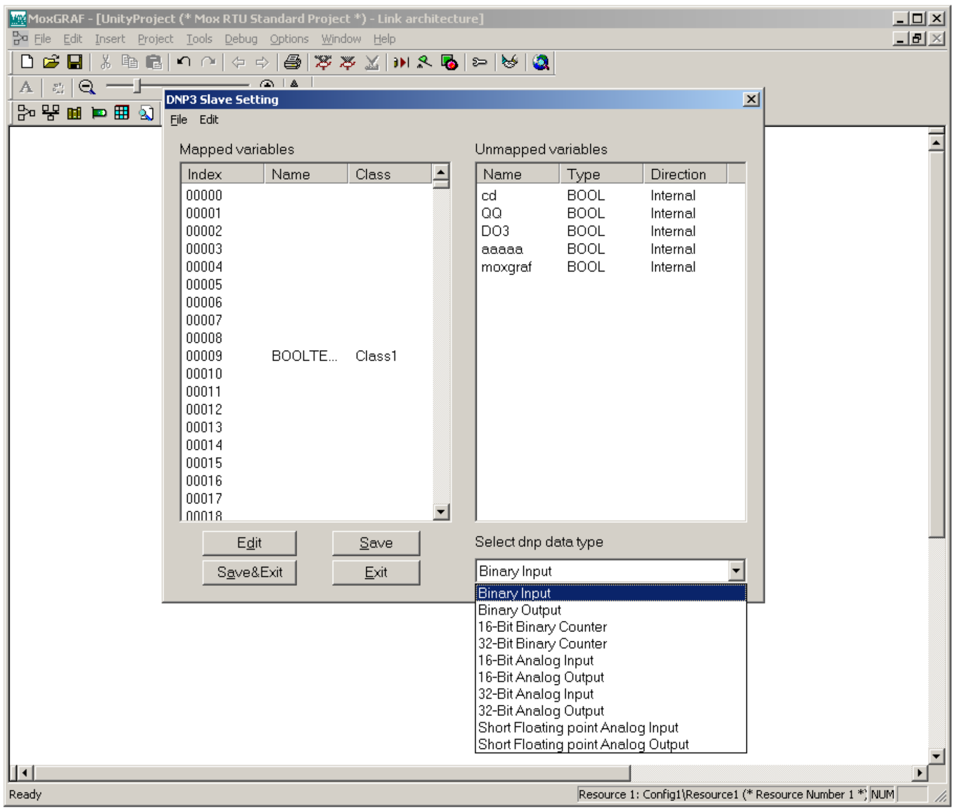
Device No.	Stat.
3	0 Binary Input; 0 Binary Output; 0 16-Bit Analog Input; 0 32-Bit...
8	0 Binary Input; 0 Binary Output; 0 16-Bit Analog Input; 0 32-Bit...
9	0 Binary Input; 0 Binary Output; 0 16-Bit Analog Input; 0 32-Bit...

Figure 7 DNP3 Master Devices Summary

2.5.2.2 DNP3 Slave Address Settings

To configure the MOX DNP3 slave addressing, these are the required configuration steps to be completed in MoxGRAF:

- Assign MoxGRAF internal variables to DNP3 index
- Assign data Classes to the individual DNP3 index
- Configure individual DNP3 address object and variation specifics and Event characteristics
- Build the project database and download the compiled project configuration to the MOX controller.


MoxGRAF - [UnityProject (* Mox RTU Standard Project *) - Link architecture]
File Edit Insert Project Tools Debug Options Window Help
DNP3 Slave Setting
File Edit
Edit
Save
Save&Exit
Exit
Ready
Resource 1: Config1\Resource1 (* Resource Number 1 *) NUM

Mapped variables

Index	Name	Class
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009	BOOLTE...	Class1
00010		
00011		
00012		
00013		
00014		
00015		
00016		
00017		
00018		

Unmapped variables

Name	Type	Direction
cd	BOOL	Internal
QQ	BOOL	Internal
DO3	BOOL	Internal
aaaaa	BOOL	Internal
moxgraf	BOOL	Internal

Select dnp data type

Binary Input

- Binary Input
- Binary Output
- 16-Bit Binary Counter
- 32-Bit Binary Counter
- 16-Bit Analog Input
- 16-Bit Analog Output
- 32-Bit Analog Input
- 32-Bit Analog Output
- Short Floating point Analog Input
- Short Floating point Analog Output

Figure 8 DNP3 Slave Address Settings

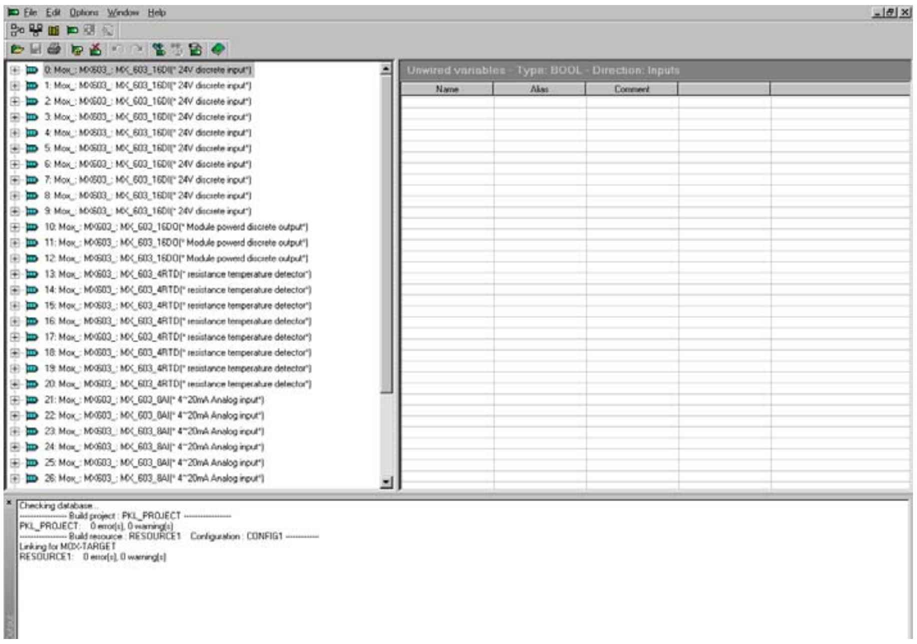
2.6 Configure I/O Wiring

The aim of the I/O wiring and configuration operation is to establish a logical link between the variables of the application and the physical channels of the external I/O devices. To make this link, the user has to identify and set up all the I/O devices and bind the previously defined variables with the corresponding I/O channels.

The whole information package of MOX I/O devices connected to one controller can be exported to MoxGRAF automatically using MoxIDE. For detailed information, please refer to *MoxIDE User Guide*.

To manually configure a connected MOX I/O network refer to the following information.

From within the MoxGRAF Programs window, select **Project | I/O Wiring**. Ensure that the Resource 1 (*Resource Number 1*) window is selected otherwise this option cannot be selected.



```
0: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
1: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
2: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
3: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
4: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
5: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
6: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
7: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
8: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
9: Mox_: MX603_: MX_603_16DI(* 24V discrete input*)
10: Mox_: MX603_: MX_603_16DO(* Module powerd discrete output*)
11: Mox_: MX603_: MX_603_16DO(* Module powerd discrete output*)
12: Mox_: MX603_: MX_603_16DO(* Module powerd discrete output*)
13: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
14: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
15: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
16: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
17: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
18: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
19: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
20: Mox_: MX603_: MX_603_4RTD(* resistance temperature detector*)
21: Mox_: MX603_: MX_603_8AI(* 4~20mA Analog input*)
22: Mox_: MX603_: MX_603_8AI(* 4~20mA Analog input*)
23: Mox_: MX603_: MX_603_8AI(* 4~20mA Analog input*)
24: Mox_: MX603_: MX_603_8AI(* 4~20mA Analog input*)
25: Mox_: MX603_: MX_603_8AI(* 4~20mA Analog input*)
26: Mox_: MX603_: MX_603_8AI(* 4~20mA Analog input*)
```

Unwired variables - Type: BOOL - Direction: Inputs

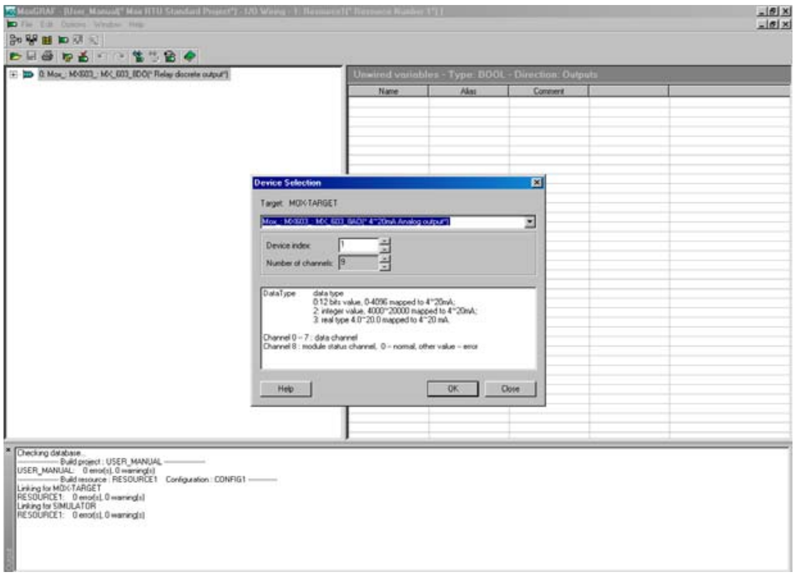
Name	Alias	Comment		

```
Checking database...
---------------- Build project : PKL_PROJECT ----------------
PKL_PROJECT:    0 error(s), 0 warning(s)
---------------- Build resource : RESOURCE1    Configuration : CONFIG1 -----------
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
```

Figure 9 MoxGRAF I/O Wiring Configuration

Within the MoxGRAF I/O Wiring window, select **Edit | Add Device**. You may also click on the corresponding icon on the toolbar, refer to Figure 9. At the first prompt use the down arrow tab to display a list of all devices.

Scroll down and select the desired device. Once selected, all information on that device will be displayed in the window underneath the prompt, refer to Figure 10.



Target: MOX-TARGET

Mox_: MX603_: MX_603_8AO(* 4~20mA Analog output*)

Device index: 1

Number of channels: 9

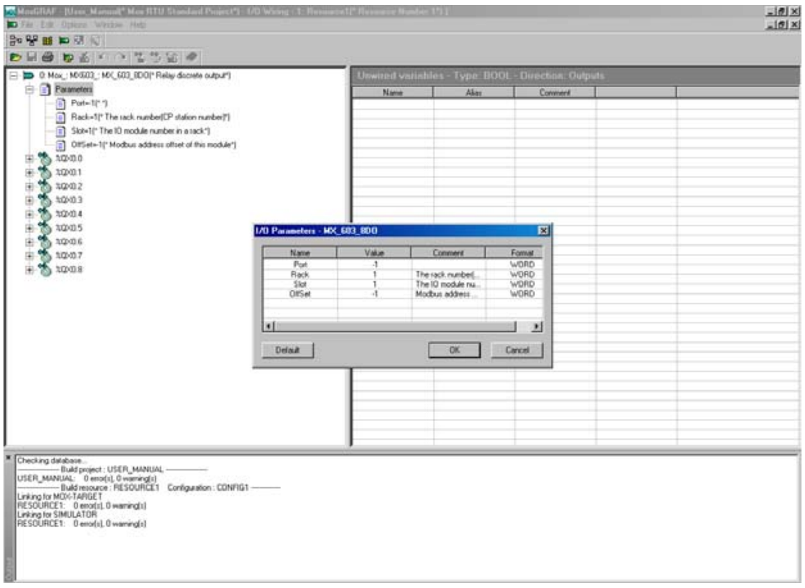
```
DataType      data type
              0:12 bits value, 0-4096 mapped to 4~20mA;
              2: integer value, 4000~20000 mapped to 4~20mA;
              3: real type 4.0~20.0 mapped to 4~20 mA.

Channel 0 -- 7 : data channel
Channel 8 : module status channel,  0 - normal, other value - error
```

```
Checking database...
------------ Build project : USER_MANUAL ------------
USER_MANUAL:    0 error(s), 0 warning(s)
------------ Build resource : RESOURCE1    Configuration : CONFIG1 --------
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
```

Figure 10 I/O Information

Once the I/O device is selected it will appear in the left window. Double clicking on the I/O device will display a **Parameter** option and channels for variable connections.

Opening the **Parameter** option, will display the parameters window, refer to Figure 11.



```
0: Mox_: MX603_: MX_603_8DO(* Relay discrete output*)
   Parameters
      Port=-1(* *)
      Rack=1(* The rack number(CP station number)*)
      Slot=1(* The IO module number in a rack*)
      OffSet=-1(* Modbus address offset of this module*)
   %QX0.0
   %QX0.1
   %QX0.2
   %QX0.3
   %QX0.4
   %QX0.5
   %QX0.6
   %QX0.7
   %QX0.8
```

I/O Parameters - MX_603_8DO

Name	Value	Comment	Format
Port	-1		WORD
Rack	1	The rack number(...	WORD
Slot	1	The IO module nu...	WORD
OffSet	-1	Modbus address ...	WORD

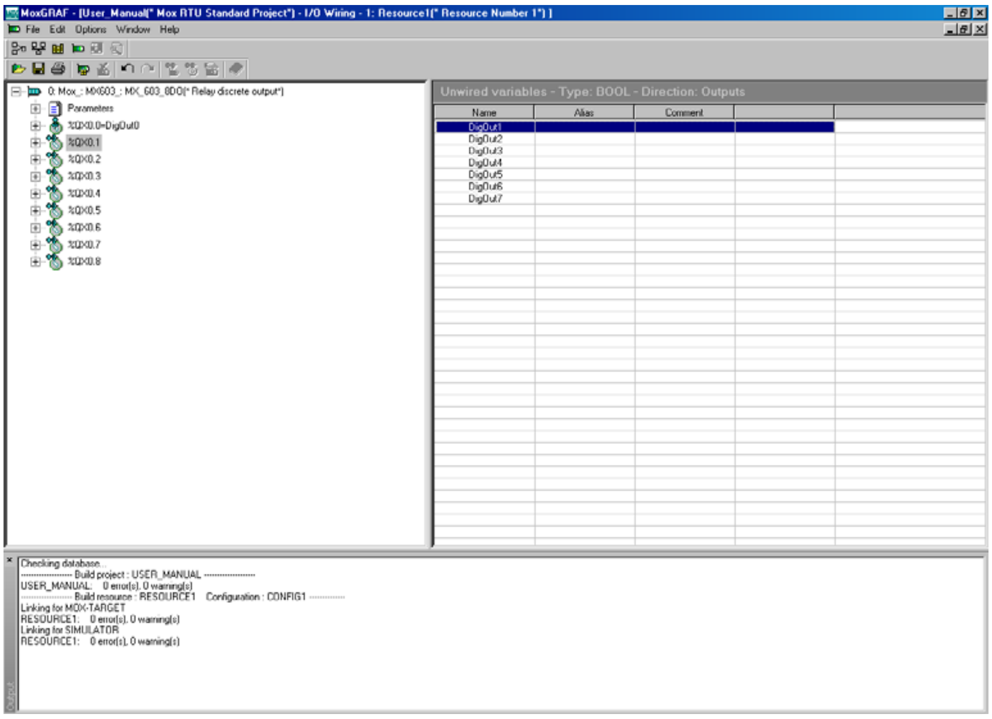
```
Checking database...
------------ Build project : USER_MANUAL ------------
USER_MANUAL:    0 error(s), 0 warning(s)
------------ Build resource : RESOURCE1    Configuration : CONFIG1 --------
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
```

Figure 11 I/O Parameters Window

Close the parameters window and allocate the variables in the right window to the desired channels of the device selected. To do this select the desired device, refer to Figure 12. This will display the channels of the I/O device and all possible variables that can be connected to this device type.

Select a desired channel of the I/O device and double click on the variable name in the Unwired variables window. This will move that variable out of that window and connect it to the selected channel.

The selected variable should now be connected to the channel. The variable can be unwired by double clicking on it in the connected window. It will then disconnect from the channel and appear in the unwired window.


MoxGRAF - [User_Manual(* Mox RTU Standard Project*) - I/O Wiring - 1: Resource1(* Resource Number 1*)]

```
0: Mox_: MX603_: MX_603_8DO(* Relay discrete output*)
    Parameters
    %QX0.0=DigOut0
    %QX0.1
    %QX0.2
    %QX0.3
    %QX0.4
    %QX0.5
    %QX0.6
    %QX0.7
    %QX0.8
```

Unwired variables - Type: BOOL - Direction: Outputs

Name	Alias	Comment	
DigOut1			
DigOut2			
DigOut3			
DigOut4			
DigOut5			
DigOut6			
DigOut7			

```
Checking database...
------------------- Build project : USER_MANUAL -------------------
USER_MANUAL:    0 error(s), 0 warning(s)
------------------- Build resource : RESOURCE1    Configuration : CONFIG1 -----------
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
```

Figure 12 Assigning Variables

Ensure that you save your configuration information to the project file before continuing.

2.7 Create Program Sections

A MoxGRAF project is divided into several programming units called programs. A program is a logical programming unit that describes operations between variables of the process. The programs of the project are linked together in a tree-like architecture. A program can be described with any of the following graphic or literal languages:

- **Sequential Function Chart** (SFC) for high level programming
- **Flow Chart** (FC) for high level programming
- **Function Block Diagram** (FBD) for cyclic complex operations
- **Ladder Diagram** (LD) for Boolean operations only
- **Structured Text** (ST) for any cyclic operations
- **Instruction List** (IL) for low level operations

Before you begin programming, you must create your program sections. Select the node ***Programs*** in the tree displayed within the *Resource* window, select **Insert | Add Program** from the menu. This will display a list of the above application types. Select on the application type that you desire and an untitled program will be created under the programs section in the **Resource1** window. Enter in a new title name and press enter.

It is recommended you use a meaningful name and one that follows a naming standard. You should also enter an appropriate comment for the program section.

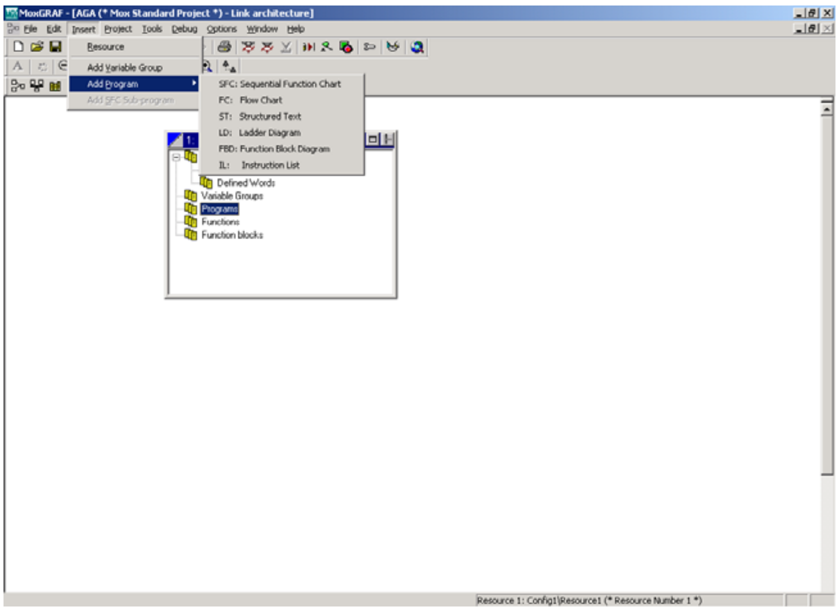


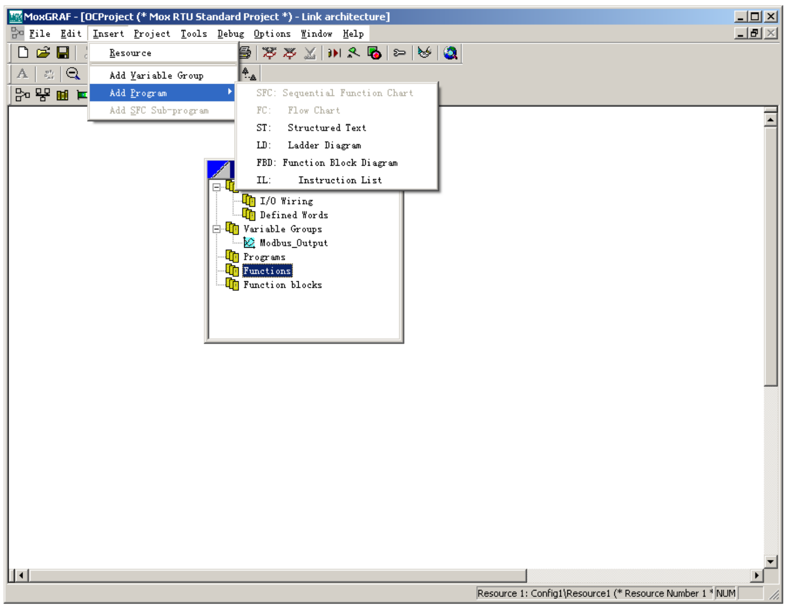
Figure 13 MoxGRAF Program Section Creation

The process of correctly creating and naming program sections will be greatly simplified by preparing a program design prior to commencing. This will also aid in ongoing development and maintenance of your completed project.

2.8 Functions and Function Blocks

2.8.1 MoxGRAF Functions

A function can be written in ST, LD, FBD and IL, which can be called by other program, function or function block. A function has no instance. It means that local data are not stored, and are generally lost from one call to the other.



MoxGRAF - [OCProject (* Mox RTU Standard Project *) - Link architecture]
File Edit Insert Project Tools Debug Options Window Help
Resource
Add Variable Group
Add Program
Add SFC Sub-program
SFC: Sequential Function Chart
FC: Flow Chart
ST: Structured Text
LD: Ladder Diagram
FBD: Function Block Diagram
IL: Instruction List
I/O Wiring
Defined Words
Variable Groups
Modbus_Output
Programs
Functions
Function blocks
Resource 1: Config1\Resource1 (* Resource Number 1 *
NUM

Figure 14 MoxGRAF Functions Selection

The interface of a function must be explicitly defined, with a type and a unique name for each of its calling (or input) or return (or output) parameter. In order to support the ST language convention, the return parameter must have the same name as the function. There is only one output parameter.

2.8.2 MoxGRAF Function Blocks

A function block can be written in ST, LD and FBD. Function blocks are instantiated. It means **local variables** of a function block are copied for each instance. When calling a function block in a program, you actually call the instance of the block: the same code is called but the data used are the one which have been allocated for the instance. Values of the variables of the instance are stored from one cycle to the other.

An instance of a function block can be called by a program or another function block, while it can not be called by a function (no internal data for a function).

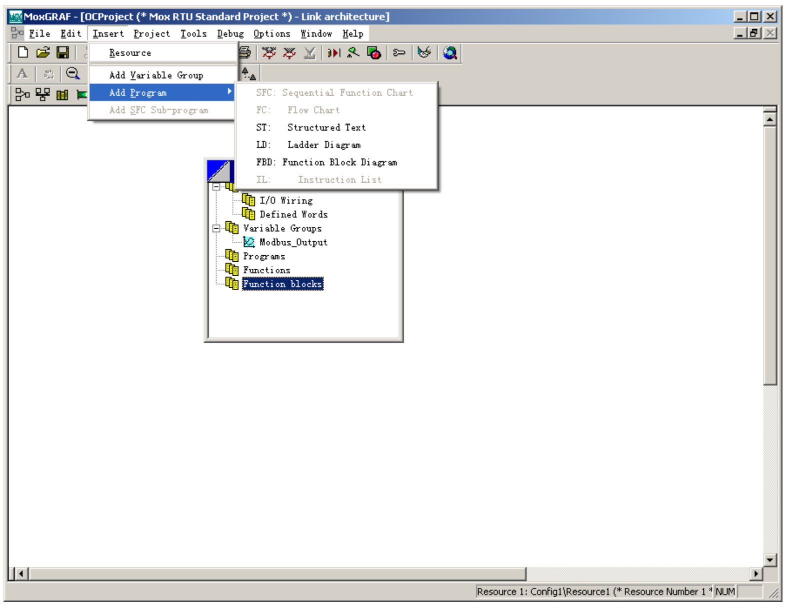

MoxGRAF - [OCProject (* Mox RTU Standard Project *) - Link architecture]
File Edit Insert Project Tools Debug Options Window Help
Resource
Add Variable Group
Add Program
Add SFC Sub-program
SFC: Sequential Function Chart
FC: Flow Chart
ST: Structured Text
LD: Ladder Diagram
FBD: Function Block Diagram
IL: Instruction List
I/O Wiring
Defined Words
Variable Groups
Modbus_Output
Programs
Functions
Function blocks
Resource 1: Config1\Resource1 (* Resource Number 1 *
NUM

Figure 15 MoxGRAF Function Blocks Selection

The interface of a function block must be explicitly defined, with a type and a unique name for each of its calling (or input) or return (or output) parameter. A function block can have more than one output parameter.

When you need a loop in your function block, you must use local variable before doing the loop.

2.9 Commence Programming

Once the program sections are created, open the appropriate program section as shown below. Select the **File | Open** from the menu. This action will open a separate programming window and then you can commence programming.

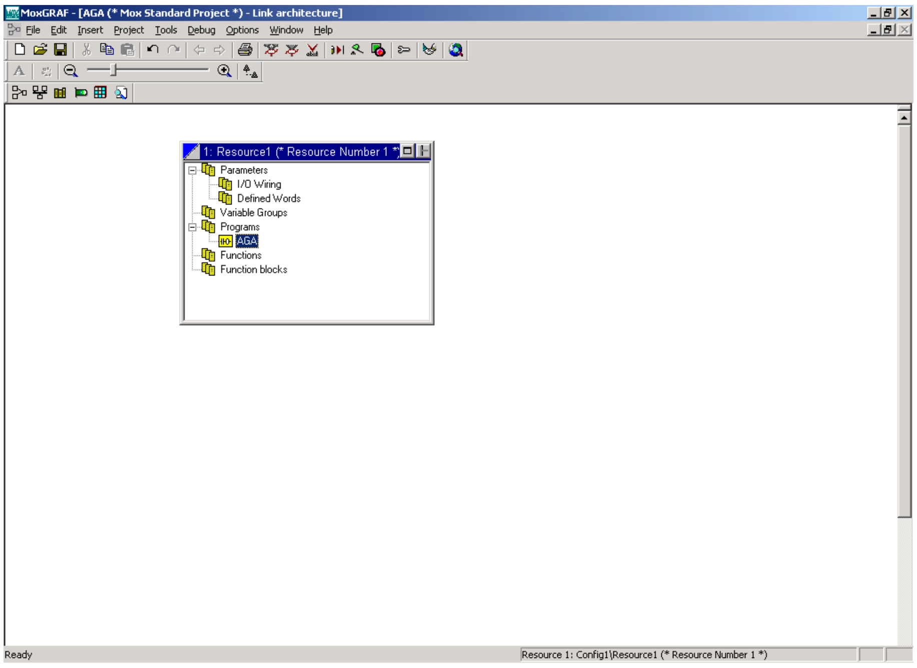

MoxGRAF - [AGA (* Mox Standard Project *) - Link architecture]
File Edit Insert Project Tools Debug Options Window Help
1: Resource1 (* Resource Number 1 *)
Parameters
I/O Wiring
Defined Words
Variable Groups
Programs
AGA
Functions
Function blocks
Ready
Resource 1: Config1\Resource1 (* Resource Number 1 *)

Figure 16 MoxGRAF Successful Program Creation

2.10 MoxGRAF Project Execution

MoxGRAF is a **synchronous** system and programs describe either **sequential** or **cyclic** operations. Sequential programs describe sequential operations, where the time variable explicitly synchronizes basic operations. Cyclic programs are executed during each cycle of the MOX device. All the operations are triggered by a clock. The basic duration of the clock is called the cycle timing:

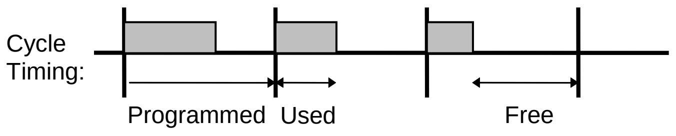


Figure 17 MoxGRAF Project Cycle Timing

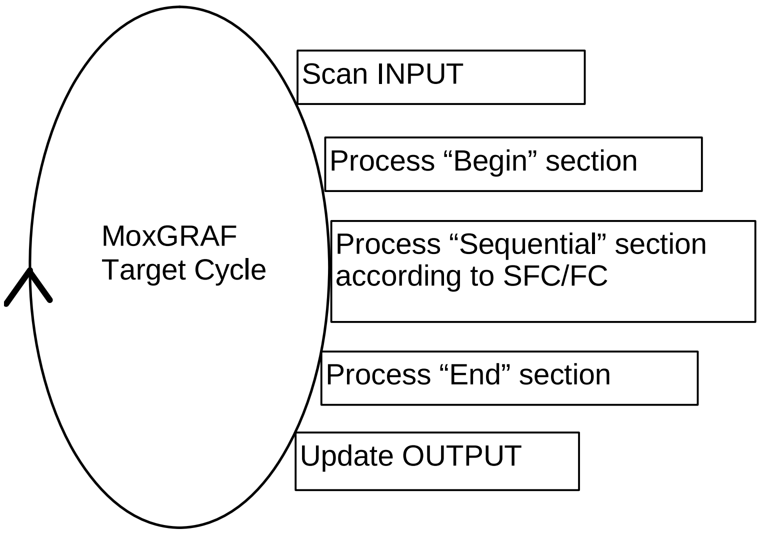


Figure 18 MoxGRAF Basic Cycle Operations

This system makes it possible to:

- Guarantee that an input variable keeps the same value within a cycle,
- Guarantee that an output device is not updated more than once in a cycle,
- Work safely on the same global variable from different programs,
- Estimate and control the response time of the complete application.

2.11 Link & Hardware Architecture

There are two kinds of architecture screen within MoxGRAF, one is **Link Architecture** and the other is **Hardware Architecture**.

The **Link Architecture** window graphically displays the resources of a project and the data links between them. This is the default view of MoxGRAF, which provides a main entry point to all the Editors.

The **Hardware Architecture** is basically used to create networks.

To show the Link Architecture screen, select the Link Architecture icon located on the toolbar; please refer to Figure 19 (identified by the green circle).

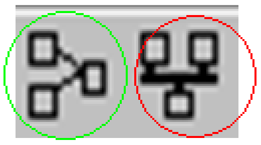


Figure 19 Link & Hardware Architecture Icons

To show the Hardware Architecture screen, select the Hardware Architecture icon located on the toolbar; please refer to Figure 19 (identified by the red circle).

Upon opening your project the link architecture page will be displayed, refer to Figure 20. From this page you are able to create new sub-programs, functions, etc.

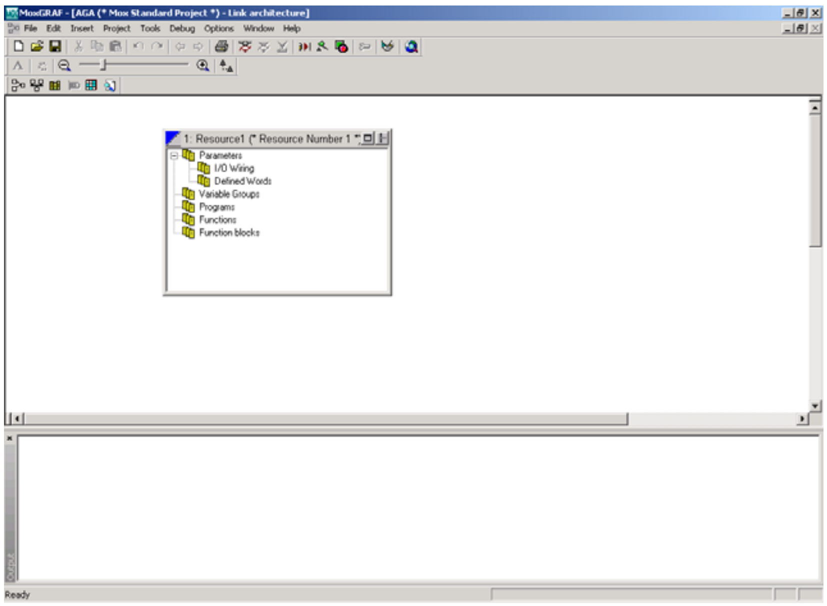


Figure 20 MoxGRAF Link Architecture

Configuration of the hardware architecture involves setting communication properties between MoxGRAF and the MOX controller. This can be performed from the Hardware Architecture screen.

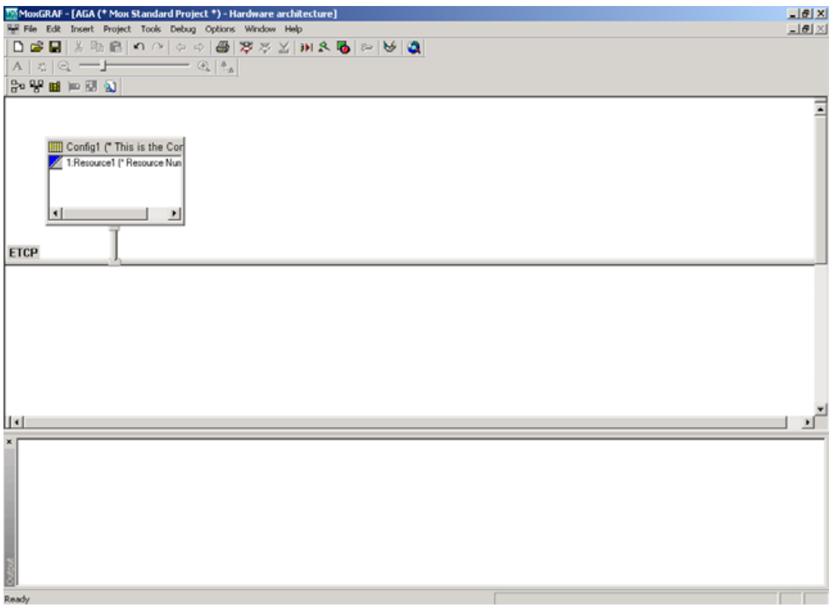

MoxGRAF - [AGA (* Mox Standard Project *) - Hardware architecture]
File Edit Insert Project Tools Debug Options Window Help
Config1 (* This is the Cor
1:Resource1 (* Resource Nun
ETCP
Ready

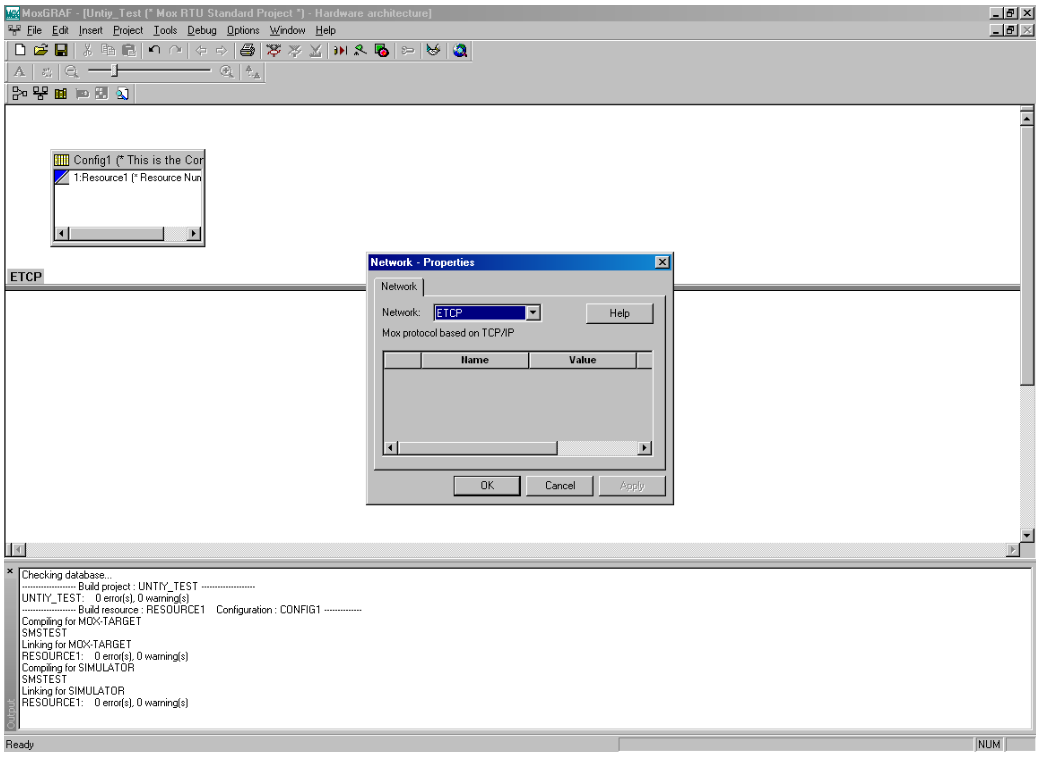
Figure 21 MoxGRAF Hardware Architecture

2.12 Create a Network Connection

To establish a network connection, open the Hardware Architecture screen and select **Insert -> Network** from the menu bar.

Meanwhile, the other way to insert a network is to open the Hardware Architecture screen and **right click** below the small window entitled **Config1**. This will display a prompt where you are able to insert either a network or a configuration, select **Insert Network**.

A window displaying the network properties will now be displayed.


MoxGRAF - [Untiy_Test (* Mox RTU Standard Project *) - Hardware architecture]
File Edit Insert Project Tools Debug Options Window Help
Config1 (* This is the Cor
1:Resource1 (* Resource Nun
ETCP
Network - Properties
Network
Help
Name
Value
OK
Cancel
Apply
Output
Ready
NUM

Network: ETCP

Mox protocol based on TCP/IP

```
Checking database...
------------------ Build project : UNTIY_TEST ------------------
UNTIY_TEST:    0 error(s), 0 warning(s)
------------------ Build resource : RESOURCE1    Configuration : CONFIG1 -------------
Compiling for MOX-TARGET
SMSTEST
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Compiling for SIMULATOR
SMSTEST
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
```

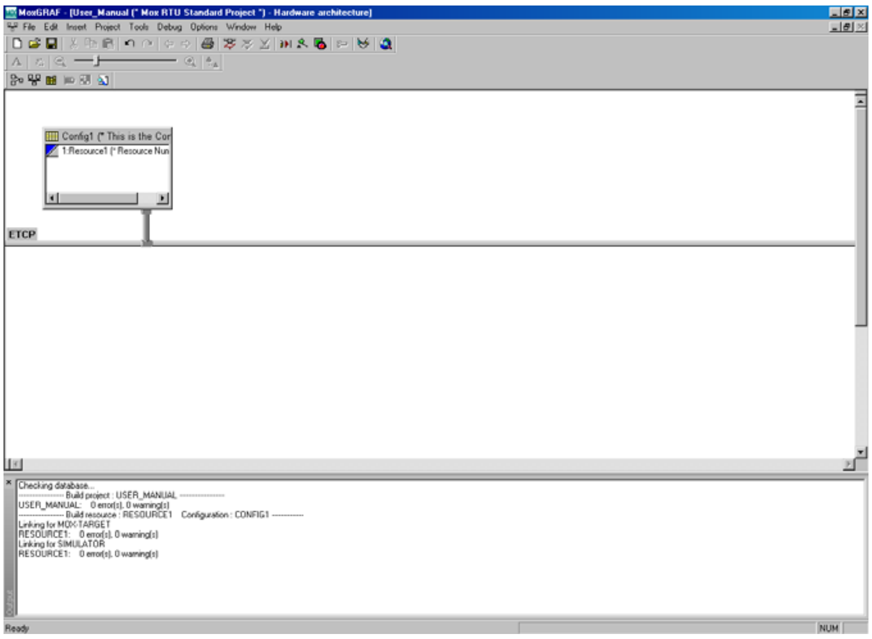
Figure 22 Inserting Connection Network

2.12.1 Ethernet

For an Ethernet connection, select **ETCP** as the desired **Network.**

2.12.2 Establish Network Link

To link the newly created network to the MoxGRAF application left click on the network line with ETCP tag, hold down the button and drag the cursor up to the middle of the **Config1** window. A link should now be displayed between the network line and the **Config1** window.

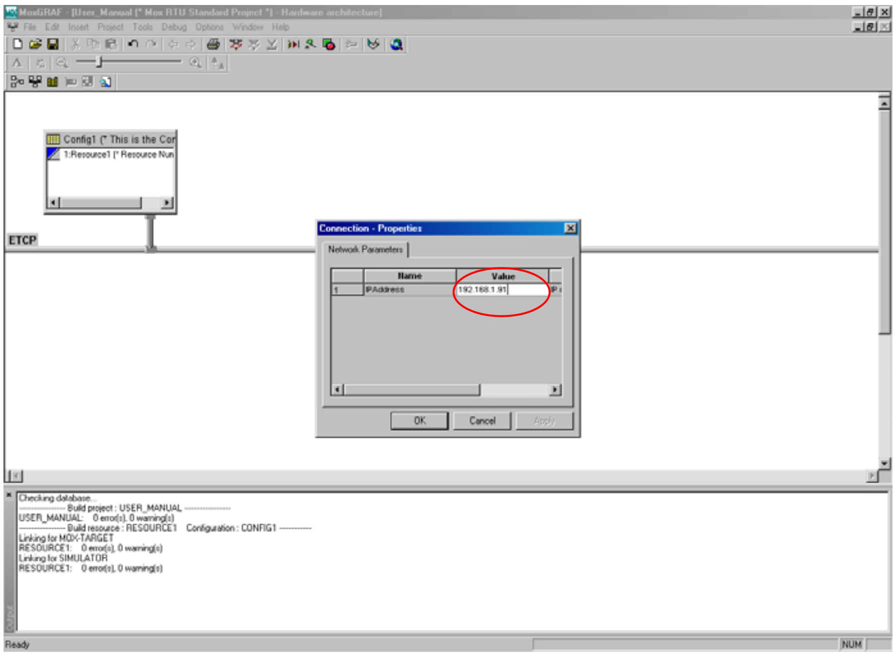

MoxGRAF - [User_Manual (* Mox RTU Standard Project *) - Hardware architecture]
File Edit Insert Project Tools Debug Options Window Help
Config1 (* This is the Cor
1:Resource1 (* Resource Nun
ETCP
Ready
NUM

```
Checking database...
---------------- Build project : USER_MANUAL ----------------
USER_MANUAL:    0 error(s), 0 warning(s)
---------------- Build resource : RESOURCE1    Configuration : CONFIG1 -----------
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
```

Figure 23 Network Connection Link

If you have just created the link the **Connection Properties** window will display automatically otherwise double click on the **link** between the network and the Config1 window. This will produce a window entitled - **Connection Properties**, refer to Figure 24.

You must enter the IP Address of the connected MOX controller to make MoxGRAF know whom to communicate with, before clicking **"OK"**.


MoxGRAF - [User_Manual (* Mox RTU Standard Project *) - Hardware architecture]
File Edit Insert Project Tools Debug Options Window Help
Config1 (* This is the Cor
1:Resource1 (* Resource Nun
ETCP
Connection - Properties
Network Parameters
OK
Cancel
Apply
Ready
NUM

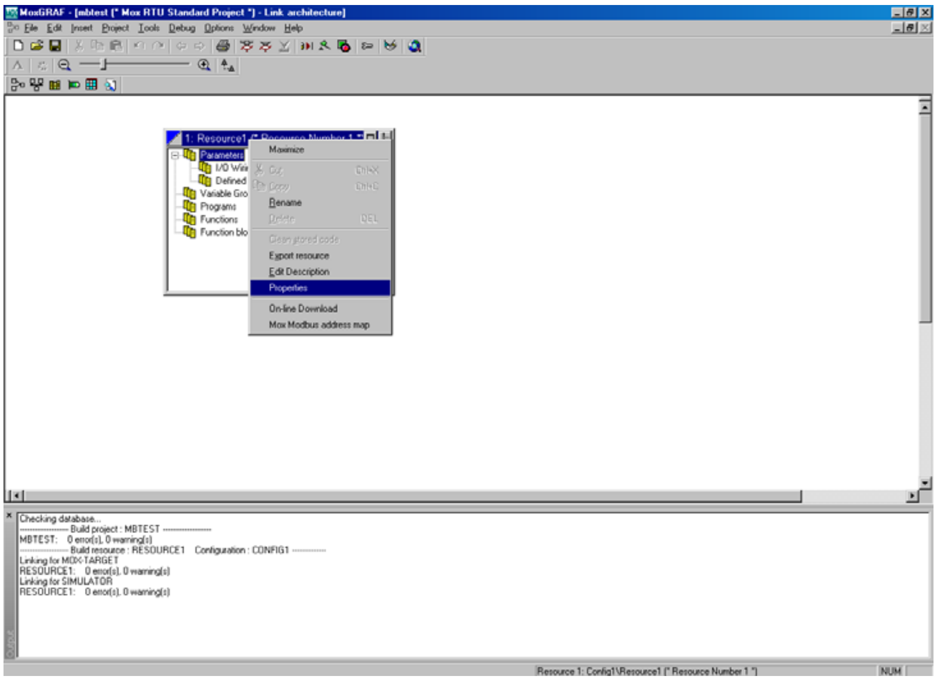
	Name	Value
1	IPAddress	192.168.1.91

```
Checking database...
---------------- Build project : USER_MANUAL ----------------
USER_MANUAL:    0 error(s), 0 warning(s)
---------------- Build resource : RESOURCE1    Configuration : CONFIG1 -----------
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
```

Figure 24 Configure the IP address of the Target device

2.13 Configure Redundancy in MoxGRAF

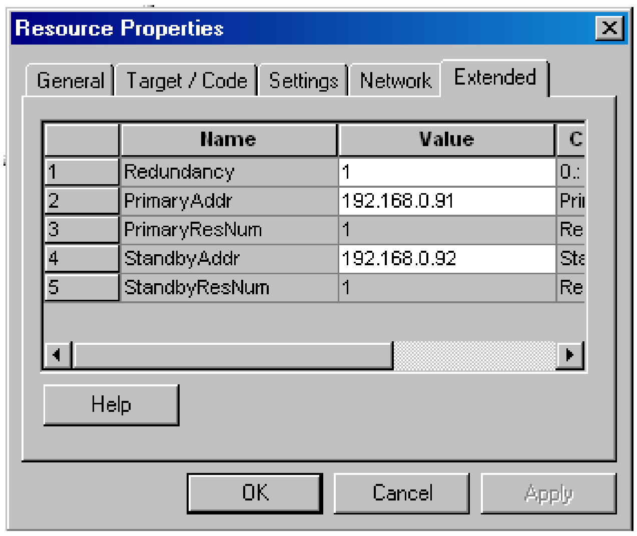
The primary and standby controllers must be configured when programming. In the **Link Architecture** screen right click on the top of the **Resource1 (*Resource Number 1*)** window and select **Properties**, please refer to Figure 25.


MoxGRAF - [mbtest (" Mox RTU Standard Project ") - Link architecture]
File Edit Insert Project Tools Debug Options Window Help
1: Resource1 (" Resource Number 1 ")
Parameters
I/O Wir
Defined
Variable Gro
Programs
Functions
Function blo
Maximize
Cut Ctrl+X
Copy Ctrl+C
Rename
Delete DEL
Clean stored code
Export resource
Edit Description
Properties
On-line Download
Mox Modbus address map
Resource 1: Config1\Resource1 (" Resource Number 1 ")
NUM

```
Checking database...
------------------ Build project : MBTEST ------------------
MBTEST:    0 error(s), 0 warning(s)
------------------ Build resource : RESOURCE1    Configuration : CONFIG1 ------------
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
```

Figure 25 Resource1 Properties

Open **Extended** tab and set the **Redundancy Value** to **"1"** to enable the redundancy. Set the **Primary Addr** and **Standby Addr** to the IP address of the desired controllers' redundancy port. Select **"Apply"** and exit the **Resource Properties** window.



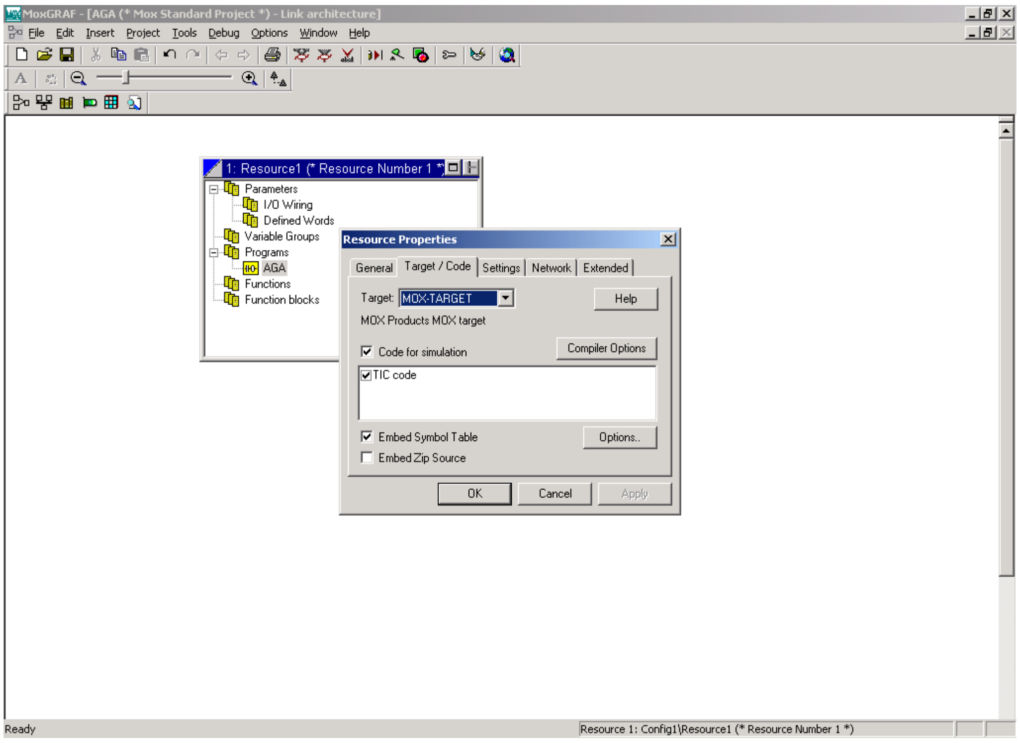
	Name	Value	C
1	Redundancy	1	0.:
2	PrimaryAddr	192.168.0.91	Pri
3	PrimaryResNum	1	Re
4	StandbyAddr	192.168.0.92	Sta
5	StandbyResNum	1	Re

Figure 26 Resource Properties

Downloading of the program is required to be performed only on the currently active controller, as it will seek out the standby controller and download the program to it, by itself.

2.14 Compiler Options

Change to the **Link Architecture** screen and right click on the top of the **Resource1** window to display a list of options. Select the **Properties** option to open the Resource Properties window, refer to Figure 27. Ensure that the target is set to the desired option and that the **TIC (Target Independent Code) Code** is selected.



MoxGRAF - [AGA (* Mox Standard Project *) - Link architecture]

1: Resource1 (* Resource Number 1 *)

- Parameters
  - I/O Wiring
  - Defined Words
- Variable Groups
- Programs
  - AGA
- Functions
- Function blocks

Resource Properties

Target: MOX-TARGET

MOX Products MOX target

☑ Code for simulation

☑ TIC code

☑ Embed Symbol Table

☐ Embed Zip Source

Ready

Resource 1: Config1\Resource1 (* Resource Number 1 *)

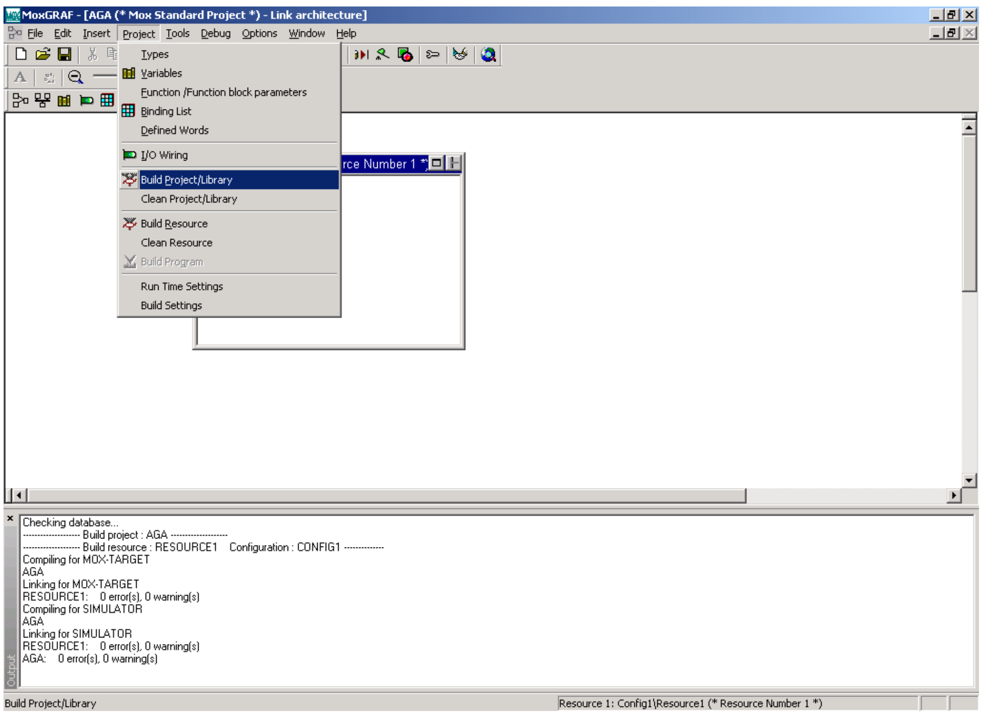
Figure 27 MoxGRAF Compiler Options

2.14.1 Build Application

*Ensure that the MoxGRAF **license hardware key** is installed in your PC. **You will not be able to perform** any of the following operations without a licensed hardware key.*

Before selecting to build the application, all old conflicting information must be cleared from the project library to ensure that the built program code is correct. To perform this operation, select **Project | Clean Project/Library**.

Select the **Project | Build Project/Library**. MoxGRAF will commence compiling the application code and will report the status of the compilation. Any errors that are reported must be corrected before a successful compilation is completed.



```
Checking database...
------------------ Build project : AGA ------------------
------------------ Build resource : RESOURCE1    Configuration : CONFIG1 -----------
Compiling for MOX-TARGET
AGA
Linking for MOX-TARGET
RESOURCE1:    0 error(s), 0 warning(s)
Compiling for SIMULATOR
AGA
Linking for SIMULATOR
RESOURCE1:    0 error(s), 0 warning(s)
AGA:    0 error(s), 0 warning(s)
```

Build Project/Library   Resource 1: Config1\Resource1 (* Resource Number 1 *)

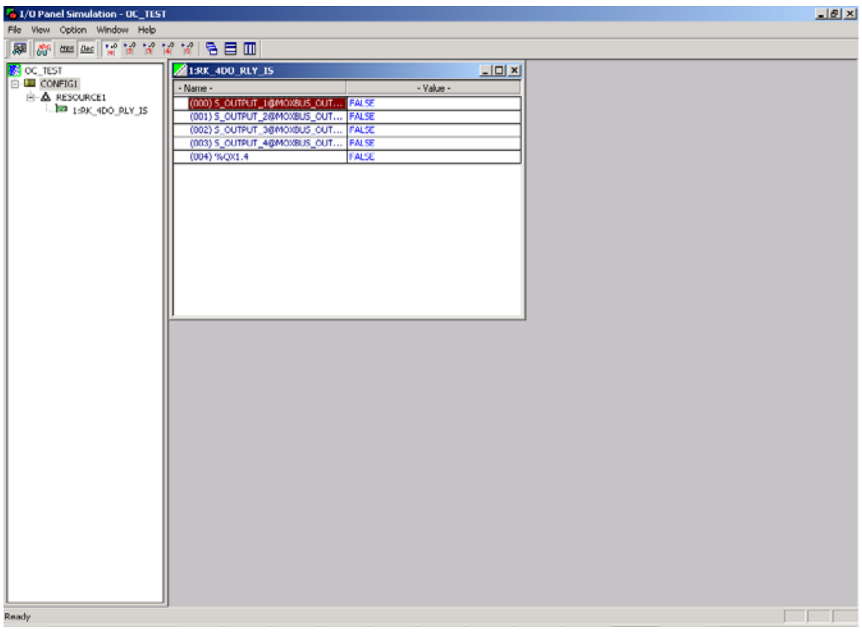
Figure 28 MoxGRAF Build Application

2.15 Program Simulation

After you have completed your program and before you proceed to download it to the device, it is beneficial to ensure that your program performs the way you want. This can be achieved by simulating your program “offline” through MoxGRAF.

Before compiling your program, ensure the compiler option "code for simulation" is selected; please refer to chapter 2.13 and Figure 27.

Once your program has been compiled successfully you are ready to simulate it. MoxGRAF will simulate all input and output information that your program produces or requires. Select the **Debug | Simulation** to start the simulation application, refer to Figure 29.



- Name -	- Value -
(000) S_OUTPUT_1@MOXBUS_OUT...	FALSE
(001) S_OUTPUT_2@MOXBUS_OUT...	FALSE
(002) S_OUTPUT_3@MOXBUS_OUT...	FALSE
(003) S_OUTPUT_4@MOXBUS_OUT...	FALSE
(004) %QX1.4	FALSE

Figure 29 MoxGRAF Variable Simulation Status

This step can cut down the amount of time used for error checking within the program before the final download step is required.

If your program does not perform the way you thought that it should, ensure that you have compiled the latest version.

If you alter the program prior to downloading without compiling it, then the previous compiled version will be executed, not the current program.

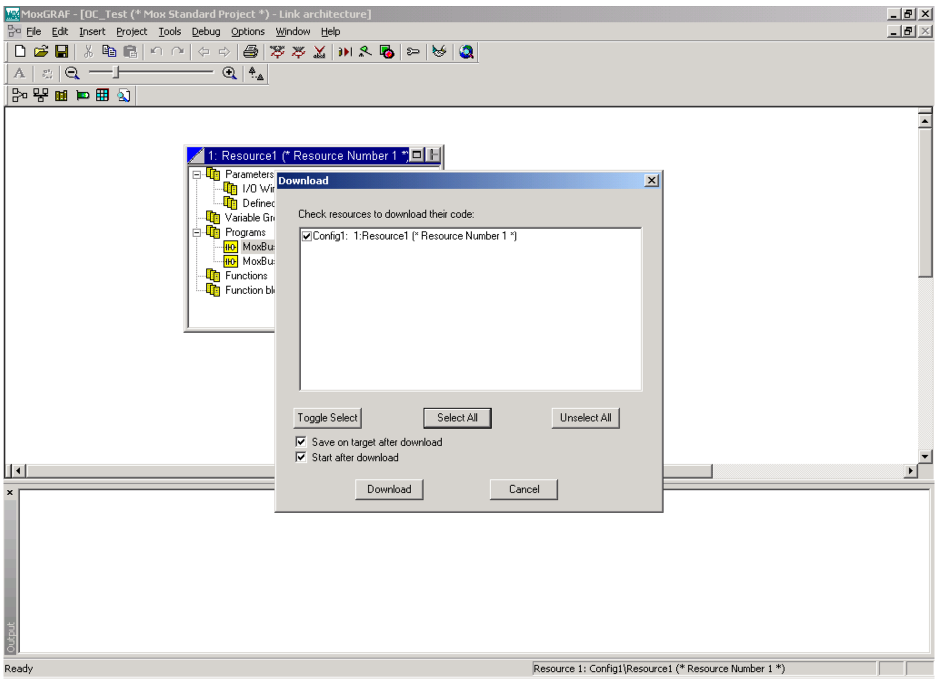
The following table gives function blocks that are not fully supported in MoxGRAF Simulator. Their inputs do not have any influence on the outputs while their outputs can be modified according to users' requirements. Other than these, the rest of the function blocks are fully supported in MoxGRAF Simulator.

FB Name	Description
Ammeter645	Ammeter645 Data Access
AmmeterWS	Ammeter Data Access for WeiSheng
ComX	Comm port process
EmailRx	Email Recv
EmailTx	Email Send
Eth2Com	Ethernet to serial port process
F_Control	Open, close and delete the file
F_RW	Read from or write to the file
FileTrans	File Transfer
FlowAdjust	Flow adjust of nature gas
FrameGrab	Frame Grabber
GPRS	GPRS
ModBusM	ModBus Master
Modem	Modem Process
ModNetM	ModNet Master
MoxCmp	Compare data
MoxGetTime	Read system time
MoxLog	Log data
MoxRxTX	Read or write remote data
MoxSetTime	Set system time
MXPower	Detect the MOX Unity power supplier
Ping	Ping Destination IP
PPP	ppp connection management
SysInfo	System Information
Temperature	Detect the CPU board temperature
UPSConf	UPS Configuration
UPSMonitor	UPS monitoring
WatchDog	WatchDog

Table 1 Function Blocks with Limited Simulation Support

2.16 Download the Application

You may now download your application to the MOX controller by selecting **Debug | Download**, refer to Figure 30. Individually select the programs that you wish to download to the target device or simply click on the **“Select All”** button. Finally click on the **“Download”** button.


MoxGRAF - [OC_Test (* Mox Standard Project *) - Link architecture]
File Edit Insert Project Tools Debug Options Window Help
1: Resource1 (* Resource Number 1 *)
Parameters
I/O Wir
Definec
Variable Gr
Programs
MoxBu
MoxBu
Functions
Function bl
Download
Toggle Select
Select All
Unselect All
Download
Cancel
Output
Ready
Resource 1: Config1\Resource1 (* Resource Number 1 *)

Check resources to download their code:

☑ Config1: 1:Resource1 (* Resource Number 1 *)

☑ Save on target after download

☑ Start after download

Figure 30 MoxGRAF Application Download

Once the **“Download”** button has been selected a small window will be displayed indicating the status of the download to the target. If any warnings or errors occur during this time, attend to them individually and progressively until your application has been successfully downloaded.

2.16.1 Project Verification

After you have downloaded the application to the MOX controller, the application will automatically start. Once the application is running, there are 3 possible ways to ensure that the MOX controller is performing correctly and that your program is correct, whilst the debugger window is still active.

Dictionary Project Verification

The first option is to open the dictionary of your program. This will display all variables used in your program and will also identify their current state, i.e. whether that state is TRUE or FALSE or holds a particular value.

I/O Connection Project Verification

The second option is to open up the I/O wiring. This will display all the devices that you have configured and all inputs and outputs that you have attached to those devices. All information concerning the inputs and outputs will be displayed next to them, i.e. whether the current value of a particular Digital Input or Output is TRUE or FALSE, or Analog Input or Output holds a particular value. Manual manipulation of these values is possible within MoxGRAF and the displayed information will reflect the changes.

Source Code Project Verification

The third option to check program functionality is to open up the source code. Depending on the language used to write the program, functionality at different steps in the program code will be displayed with different colours or an indicator that will show the current state of the program.

Please review the MoxGRAF online help for more detailed information on how to configure and program a MOX controller.

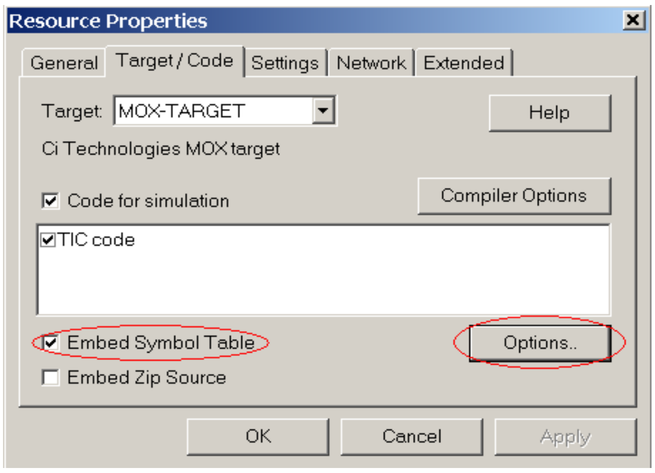
2.17 On-line Download

The on-line download feature enables the user to modify a resource when it is running, however there are some limitations. If modifications are made that exceed the limitations, the on-line download feature will still work, but the modified code will run as a new start program instead of updating the old application.

This function should be used with care. MoxGRAF may not be able to detect all possible conflicts generated by the user-defined operations as a result of the on-line download.

2.17.1 On-line Download Operation

In order to use on-line download, the complete symbol table must be compiled and downloaded. Please confirm that the options in following images are checked.


Resource Properties

General | Target / Code | Settings | Network | Extended

Target: MOX-TARGET

Ci Technologies MOX target

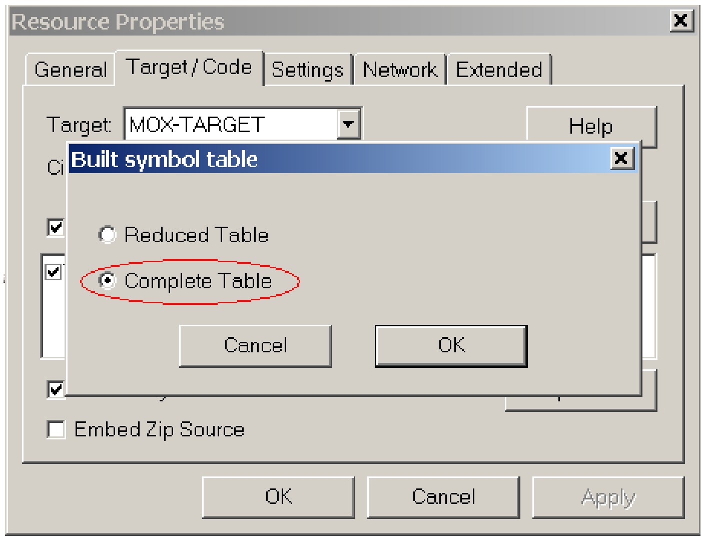
☑ Code for simulation

☑ TIC code

☑ Embed Symbol Table

☐ Embed Zip Source

Figure 31 Resource Properties for On-line Download


Resource Properties

Built symbol table

○ Reduced Table

● Complete Table

☐ Embed Zip Source

Figure 32 Symbol Table Build Properties

Modifying a running resource consists of the following operations:

- Modifying the resource source code on the workbench.
- Generating the new resource code.
- Downloading the new resource code. Select **Tools | On-line Download**. Then the progress bar will indicate the download progress.

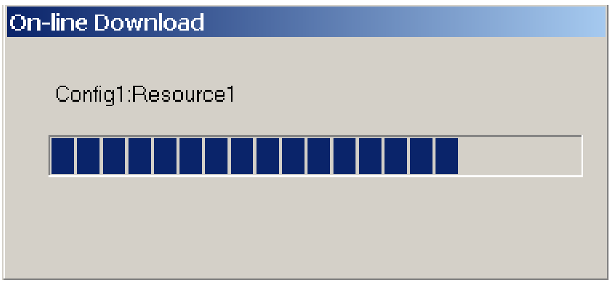


Figure 33 On-line Download Progress Bar

2.17.2 On-line Download Limitations

SFC (Sequential Function Chart) and FC (Flow Chart) code will be reset after on-line download. It's recommended not to use these programming languages if you want to perform on-line download.

Changing the Program Organization Unit (POU) name is not allowed if use of the online download function is intended. If the POU name is changed, the code and its local variables will be reset.

3 Programming Languages

3.1 Instruction List (IL) Language

Instruction List (IL) is a low level language. It is highly effective for smaller applications or for optimising parts of an application. Instructions always relate to the current result (or IL register). The operator indicates the operation that must be made between the current value and the operand. The result of the operation is stored again in the current result.

Below are examples of instruction lines:

Label	Operator	Operand	Comments
Start:	LD	IX1	(* push button *)
	ANDN	MX5	(* command is not forbidden *)
	ST	QX2	(* start motor *)

3.2 Structured Text (ST) Language

Structured Text (ST) is a high level structured language designed for automation processes. This language is mainly used to implement complex procedures that cannot be easily expressed with graphic languages.

ST is the default language for the description of the actions within the steps and conditions attached to the transitions of the **SFC** language.

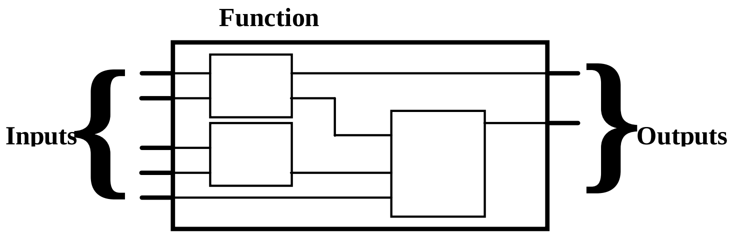
```
(* imax : number of iterations *)
(* i: FOR statement index *)
(* cond: process validity *)
imax := max_ite;
cond := X12;

if not (cond) then
        return;
end_if;

(* process loop *)
for i := 1 to max_ite do
        if i <> 2 then
                Spcall ();
        end_if;
end_for;
```

3.3 Function Block Diagram (FBD) Language

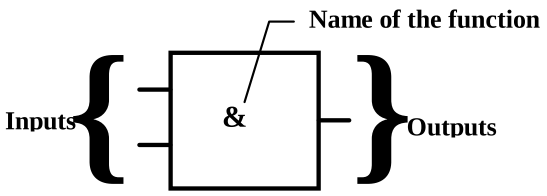
The Functional Block Diagram (FBD) is a graphic language. It allows the programmer to build complex procedures by taking existing functions and function blocks from the MoxGRAF library and wiring them together in the graphic diagram area.

FBD diagram main format

FBD diagram describes a function between input variables and output variables. A function is described as a set of elementary blocks. Input and output variables are connected to blocks by connection lines. An output of a block may also be connected to an input of another block.



An entire function operated by an FBD program is built with standard elementary blocks from the MoxGRAF library. Each block has a fixed number of input connection points and a fixed number of output connection points. A block is represented by a single rectangle. The inputs are connected on its left border. The outputs are connected on its right border. An elementary block performs a single function between its inputs and its outputs. The name of the function to be performed by the block is written in its rectangle symbol. Each input or output of a block has a well-defined type.



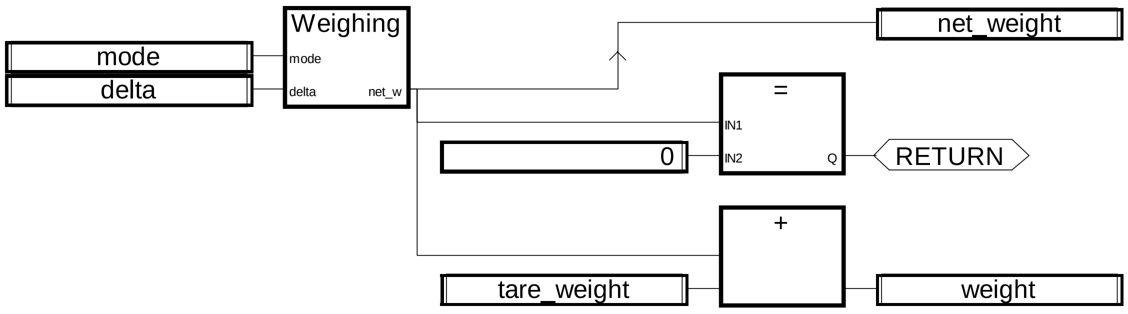
Input variables of an FBD program must be connected to input connection points of blocks. The type of each variable must be the same as the type expected for the associated input. An Input for FBD diagram can be a constant expression, any internal or input variable, or an output variable.

Output variables of an FBD program must be connected to output connection points of blocks. The type of each variable must be the same as the type expected for the associated block output. An Output for FBD diagram can be any internal or output variable, or the name of the function (for functions only). When an output is the name of the currently edited function, it represents the assignment of the return value for the function (returned to the calling program).

Input and output variables, inputs and outputs of the blocks are wired together with connection lines. Single lines may be used to connect two logical points of the diagram.

The connection is oriented, meaning that the line carries associated data from the left extremity to the right extremity. The left and right extremities of the connection line must be of the same type.

(* Example of an FBD program *)



(* ST Equivalence *)

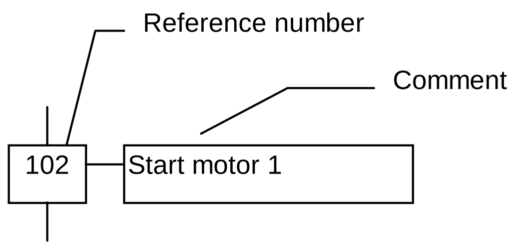
```
net_weight := Weighing (mode, delta); (* call sub-program *)
If (net_weight = 0)
        Then Return;
End_if;
weight := net_weight + tare_weight;
```

3.4 Sequential Function Chart (SFC) Language

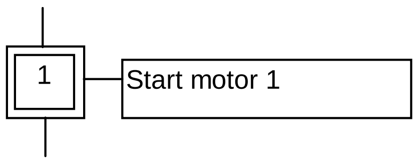
Sequential Function Chart (SFC) is a graphic language used to describe sequential operations. The process is represented as a set of well-defined steps, linked by transitions. A Boolean condition is attached to each transition. Actions within the steps are detailed by using other languages (ST, IL, LD and FDB).

Steps and Initial Steps

A step is represented by a single square. Each step is referenced by a number, written in the step square symbol. A main description of the step is written in a rectangle linked to the step symbol. This description is a free comment (not part of the programming language). The above information is called the Level 1 of the step:

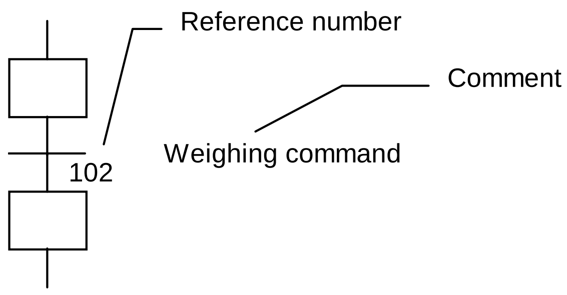


The initial situation of an SFC program is expressed with initial steps. An initial step has a double-bordered graphic symbol. A token is automatically placed in each initial step when the program is started. An SFC program must contain at least one initial step.



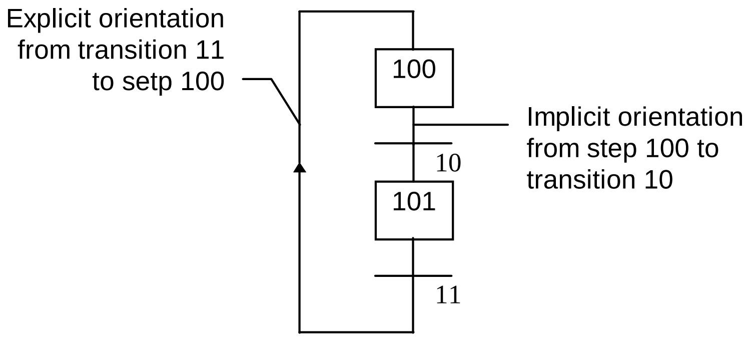
Transitions

Transitions are represented by a small horizontal bar that crosses the connection link. Each transition is referenced by a number, written next to the transition symbol. A main description of the transition is written on the right side of the transition symbol. This description is a free comment (not part of the programming language). The above information is called the Level 1 of the transition:



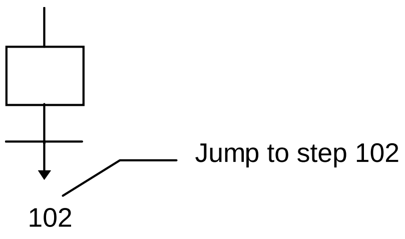
Oriented links

Single lines are used to link steps and transitions. These are oriented links. When the orientation is not explicitly given, the link is oriented from the top to the bottom.

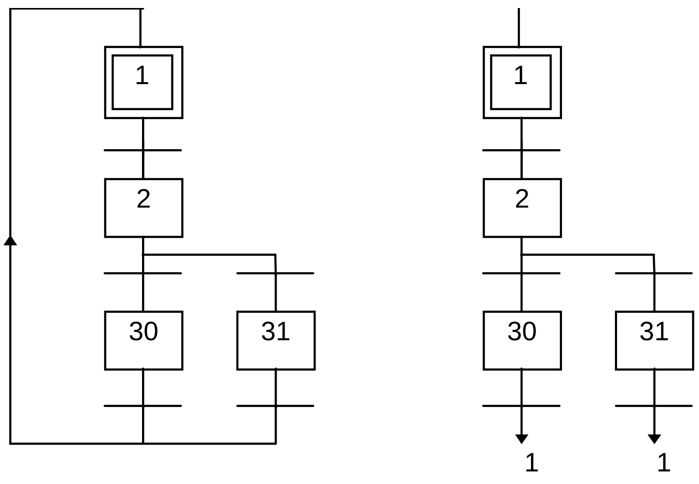

Explicit orientation from transition 11 to setp 100
100
Implicit orientation from step 100 to transition 10
10
101
11

Jump to a step

Jump symbols may be used to indicate a connection link from a transition to a step, without having to draw the connection line. The jump symbol must be referenced with the number of the destination step:



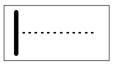
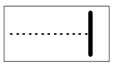
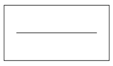
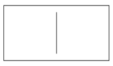
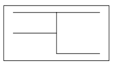
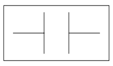
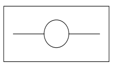
A jump symbol cannot be used to represent a link from a step to a transition. Examples of jumps - the following charts are equivalent:



3.5 Ladder Diagram (LD) Language

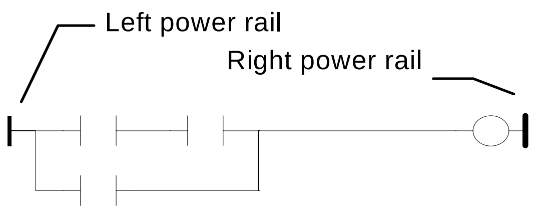
Ladder Diagram (LD) is a graphic representation of Boolean equations, combining contacts (input arguments) with coils (output results). The LD language enables the description of tests and modifications of Boolean data by placing graphic symbols into the program chart. LD graphic symbols are organized within the chart exactly as an electric contact diagram.

These are basic graphic components of an LD diagram:

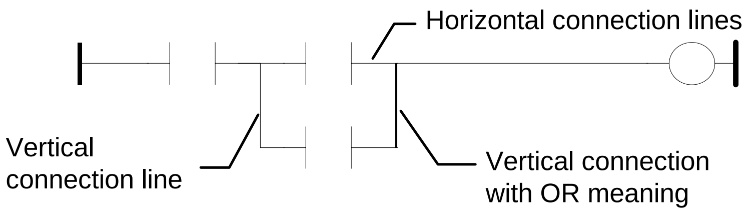
	Left vertical power rail
	Right vertical power rail
	Horizontal connection line
	Vertical connection line
	Multiple connection lines (all connected together)
	Contact associated with a variable
	Coil associated to an output or to an internal variable

Power rails and connection lines

An LD diagram is limited on the left and right side by vertical lines, named left power rail and right power rail respectively.



LD diagram graphic symbols are connected to power rails or to other symbols by connection lines. Connection lines are horizontal or vertical.


Horizontal connection lines
Vertical connection line
Vertical connection with OR meaning

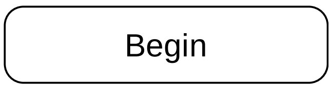
Each line segment has a Boolean state FALSE or TRUE. The Boolean state is the same for all the segments directly linked together. Any horizontal line connected to the left vertical power rail has the TRUE state.

3.6 Flow Chart (FC) Language

Flow Chart (FC) is a graphic language used to describe sequential operations. A Flow Chart diagram is composed of Actions and Tests. Between Actions and tests are oriented links representing data flow. Actions and Tests can be described with ST, LD or IL languages. Functions and Function blocks of any language (except SFC) can be called from actions and tests. A Flow Chart program can call another Flow Chart program. The called program is a sub-program of the calling FC program.

Beginning of FC chart

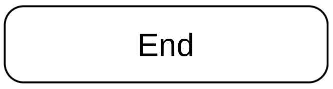
A "begin" symbol must appear at the beginning of a Flow Chart program. It is unique and cannot be omitted. It represents the initial state of the chart when it is activated. Below is the drawing of a "begin" symbol:



The "Begin" symbol always has a connection (on the bottom) to the other objects of the chart. A flow chart is not valid if no connection is drawn from "Begin" to another object.

Ending of FC chart

An "end" symbol must appear at the end of a Flow Chart program. It is unique and cannot be omitted. It is possible that no connection is drawn to the "End" symbol (always looping chart), but "End" symbol is still drawn anyway at the bottom of the chart. It represents the final state of the chart, when its execution has been completed. Below is the drawing of an "end" symbol:



The "End" symbol generally has a connection (on the top) to the other objects of the chart. A flow chart may have no connection to the "End" object (always looping chart). The "End" object is still visible at the bottom of the chart in this case.

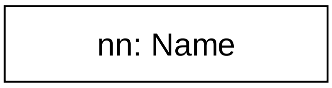
FC flow links

A flow link is a line that represents a flow between two points of the diagram. A link is always terminated by an arrow. Below is the drawing of a flow link:

Two links cannot be connected to the same source connection point.

FC actions

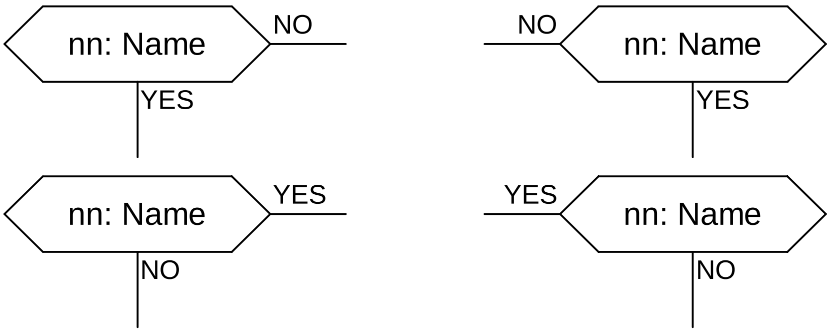
An action symbol represents actions to be performed. An action is identified by a number and a name. Below is the drawing of an "action" symbol:



Two different objects of the same chart cannot have the same name or logical number. Programming language for an action can be ST, LD or IL. An action is always connected with links, one arriving to it, and one starting from it.

FC conditions

A condition represents a Boolean test. A condition is identified by a number and a name. According to the evaluation of attached ST, LD or IL expression, the flow is directed to "YES" or "NO" path. Below are the possible drawings for a condition symbol:



4 Programming Instructions

4.1 Standard Operators

The following are standard operators of the IEC61131-3 languages.

	Symbol	Operation
Data Manipulation	1 Gain	Assignment
	NEG	Analog Negation
Boolean Operations	& (AND)	Boolean AND
	>=1 (OR)	Boolean OR
	=1 (XOR)	Boolean Exclusive OR
	NOT	Boolean Negation
Arithmetic Operations	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
Comparison Tests	<	Less than
	<=	Less or equal to
	>	Greater than
	>=	Greater or equal to
	=	Is equal to
	<>	Is not equal to
Data Conversion	BOO	Convert to Boolean
	ANA	Convert to Integer Analog
	REAL	Convert to Real Analog
	TMR	Convert to Timer
	MSG	Convert to Message
Logical Operators	AND_MASK	Analog bit to bit AND mask
	OR_MASK	Analog bit to bit OR mask
	XOR_MASK	Analog bit-to-bit Exclusive OR mask
	NOT_MASK	Bit to bit negation
Other	CAT	Message concatenation
	SYSTEM	System access
	OPERATE	Operate I/O channel

Table 2 MoxGRAF Standard Operators of the IEC61131-3 Languages

4.2 Instruction List Operators

The following table summarizes the standard operators of the IL language:

Operator	Modifiers	Operand	Description
LD	N	Variable, Constant	Load Operand
ST	N	Variable	Stores current result
S		BOOL Variable	Sets to TRUE
R		BOOL Variable	Sets to FALSE
CAL	C, N	FB Instance Name	Calls a function
JMP	C, N	Label	Jumps to a label
RET	C, N		Returns from sub-program
)			Executes delayed operation
Function			Calls a function or sub-function
AND	N	BOOL	Boolean AND
&	N	BOOL	Boolean AND
OR	N	BOOL	Boolean OR
XOR	N	BOOL	Boolean Exclusive OR
ADD		variable, constant	Addition
SUB		variable, constant	Subtraction
MUL		variable, constant	Multiplication
DIV		variable, constant	Division
GT		variable, constant	Test: >
GE		variable, constant	Test: >=
EQ		variable, constant	Test: =
LE		variable, constant	Test: <=
LT		variable, constant	Test: <
NE		variable, constant	Test: <>

Table 3 MoxGRAF Instruction List Operators

4.3 Standard Function Blocks

These are standard function blocks supported by the MoxGRAF system. Such function blocks are pre-defined and do not have to be declared in the library.

	Symbol	Operation
Booleans	SR	Set dominant bistable
	RS	Reset dominant bistable
	R_TRIG	Rising edge detection
	F_TRIG	Falling edge detection
	SEMA	Semaphore
Counting	CTU	Up counter
	CTD	Down counter
	CTUD	Up-down counter
Timers	TON	On-delay timing
	TOF	Off-delay timing
	TP	Pulse timing
Integer Analogs	CMP	Full comparison function block
	STACKINT	Stack of integer analogs
Real Analogs	AVERAGE	Running average over N samples
	HYSTER	Boolean hysteresis on difference of reals
	LIM_ALRM	High/low limit alarm with hysteresis
	INTEGRAL	Integration over time
	DERIVATE	Differentiation according to time
Signal Generation	BLINK	Blinking Boolean signal
	SIG_GEN	Signal generator

Table 4 MoxGRAF Standard Function Blocks

4.4 Standard Functions

These are standard functions supported by the MoxGRAF system. Such functions are pre-defined and do not have to be declared in the library.

	Symbol	Operation
Math	ABS	Absolute value
	EXPT	Exponent
	LOG	Logarithm
	POW	Power calculation
	SQRT	Square root
	TRUNC	Truncate decimal part
Trigonometric	ACOS	Arc cosine
	ASIN	Arc sine
	ATAN	Arc tangent
	COS	Cosine
	SIN	Sine
	TAN	Tangent
Register Control	ROL	Rotate Left
	ROR	Rotate Right
	SHL	Shift Left
	SHR	Shift Right
Data Manipulation	MIN	Minimum
	MAX	Maximum
	LIMIT	Limit
	MOD	Modulo
	MUX4	Multiplexer (4 entries)
	MUX8	Multiplexer (8 entries)
	ODD	Odd parity
	RAND	Random value
	SEL	Binary selector
Data Conversion	ASCII	Character ASCII code
	CHAR	ASCII code Character
String Management	DELETE	Delete sub-string
	INSERT	Insert string
	FIND	Find sub-string
	MLEN	Get string length

	Symbol	Operation
	LEFT	Extract left
	MID	Extract middle
	REPLACE	Replace sub-string
	RIGHT	Extract right
	DAY_TIME	Time of day

Table 5 MoxGRAF Standard Functions

Appendix A Product Support

Warranty Information

All MOX manufactured products are warranted to be free from defects in material and workmanship. Our obligation under this warranty will be limited to repairing or replacing, at our option, the defective parts within 1 year of the date of installation, or within 18 months of the date of shipment from the point of manufacture, whichever is sooner. Products may only be returned under authorisation. The purchaser will prepay all freight charges to return any products with a valid return authorisation number to the designated repair facility.

This limited warranty does not cover loss or damage that may occur in shipment of the goods or due to improper installation, maintenance, misuse, neglect or any cause other than ordinary commercial or industrial use. This limited warranty is in lieu of all other warranties whether oral or written, expressed or implied.

Liability associated with all MOX products shall not exceed the price of the individual unit that is the basis of the claim. In no event will there be liability for any loss of profits, loss of use of facilities or equipment or other indirect, incidental or consequential damages.

Contact Details

To obtain support for MOX products, call MOX Group on the following numbers or your designated support provider and ask for MOX Support.

E-mail addresses:

support@mox.com.au

sales@mox.com.au

Visit our web page at:

http://www.mox.com.au

Service Information

If you require service, contact your local MOX Group representative. A trained specialist will help you to quickly determine the source of the problem. Many problems are easily resolved with a single phone call. If it is necessary to return a unit, an RMA (Return Material Authorization) number will be provided.

All returned materials are tracked with our RMA system to ensure speedy service. You must include this RMA number on the outside of the box so that your return can be processed immediately.

Your MOX Group authorised applications engineer will complete an RMA request for you. If the unit has a serial number, we will not need detailed financial information. Otherwise, be sure to have your original purchase order number and date purchased available.

We suggest that you provide a repair purchase order number in case the repair is not covered under our warranty. You will not be billed if the repair is covered under warranty.

Please supply us with as many details about the problem as you can. The information you supply will be written on the RMA form and supplied to the repair department before your unit arrives. This helps us to provide you with the best service, in the fastest manner. Most repairs are completed within two days. During busy periods, there may be a longer delay.

If you need a quicker turnaround, ship the unit to us by airfreight. We give priority service to equipment that arrives by overnight delivery. Many repairs received by midmorning (typical overnight delivery) can be finished the same day and returned immediately.

We apologize for any inconvenience that the need for repair may cause you. We hope that our rapid service meets your needs. If you have any suggestions to help us improve our service, please give us a call. We appreciate your ideas and will respond to them.

For Your Convenience:

Please fill in the following information and keep this manual with your MOX system for future reference:

P.O. #: ____________________ Date Purchased: ____________________

Purchased From: ____

MOX Group

Web: www.mox.com.au
Email: info@mox.com.au