

Bringing technology to life...  
Machine Interface: NA Series



## TRAINING BOOK

Version 2.4  
OMRON Europe



## Contents

<b>1</b>	<b>Overview.....</b>	<b>6</b>
1.1	Architecture .....	6
1.2	Hardware .....	7
1.3	Connectivity.....	7
<b>2</b>	<b>Integration.....</b>	<b>8</b>
2.1	Handling Projects .....	8
2.2	Framework.....	8
2.3	Search and Replace .....	9
2.4	Cross Reference .....	9
2.5	Global Watch .....	9
2.6	Variable Mapping .....	9
2.7	Security .....	10
<b>3</b>	<b>Configuration and Settings.....</b>	<b>11</b>
3.1	General Settings .....	11
3.2	TCP/IP Network Settings .....	11
3.3	FTP Server Settings .....	12
3.4	NTP Settings.....	12
3.5	FINS Settings.....	12
3.6	VNC Settings.....	12
<b>4</b>	<b>System Menu.....</b>	<b>14</b>
4.1	Project System Menu .....	14
4.2	Device System Menu.....	14
<b>5</b>	<b>Security .....</b>	<b>16</b>
5.1	Users .....	16
5.2	Roles & Access Levels.....	16
5.3	Using Security on Pages.....	17
5.4	Actions.....	17
5.5	Security in code .....	17
5.6	Security at Runtime .....	18
5.7	Limitations in v1.0 .....	18
<b>6</b>	<b>Pages .....</b>	<b>19</b>
6.1	Graphics in the NA .....	19
6.2	Page Basics.....	19
6.3	Getting Started with Pages .....	20
6.4	Using the Property Window .....	21
6.5	Animations.....	21
6.6	Events & Actions.....	22
6.7	Layering Pages .....	22
6.8	Page Toolbar.....	23
<b>7</b>	<b>Toolbox .....</b>	<b>24</b>
7.1	Using the Toolbox.....	24
7.2	Searching the Toolbox.....	24
7.3	Objects: Buttons .....	24
7.4	Objects: Gauges.....	25
7.5	Objects: HMI Controls .....	26
7.6	Objects: Data Display .....	26
7.7	Objects: Data Edit .....	26
7.8	Objects: Media Control .....	27

7.9	Objects: Shapes .....	27
7.10	Objects: Lamps.....	27
7.11	Objects: Standard Controls.....	27
7.12	Objects: Graphics.....	28
7.13	Adding Custom Items To The Toolbox.....	28
<b>8</b>	<b>Variables .....</b>	<b>29</b>
8.1	Global Variables .....	29
8.2	Data Types .....	30
8.3	Mapping to controllers within the Project .....	31
8.4	Limitations:.....	32
8.5	NA System Variables .....	33
8.6	Local NA variables.....	34
8.7	GLOBAL vs LOCAL:.....	34
8.8	Working with External Devices .....	37
8.9	Mapping Variables .....	38
<b>9</b>	<b>VB.NET Code.....</b>	<b>39</b>
9.1	Why would I add code?.....	39
9.2	Where can I use code? .....	39
9.3	Subroutines and Functions .....	39
9.4	VB Variables .....	40
9.5	Private or Public.....	41
9.6	Designing code .....	41
9.7	Code Explorer .....	42
9.8	The Code Editor.....	42
9.9	Building the Code .....	42
9.10	NA VB Methods.....	43
9.11	Using VB to Access Object Properties.....	44
9.12	Standard VB Functions that can be used.....	47
<b>10</b>	<b>Alarms .....</b>	<b>49</b>
10.1	Creating Alarms .....	49
10.2	Alarm Events.....	50
10.3	Using the Alarm Viewer .....	50
10.4	Alarms in Runtime.....	50
10.5	System Variables.....	51
10.6	Alarms in VB .....	51
<b>11</b>	<b>Recipes.....</b>	<b>53</b>
11.1	Overview of Recipes .....	53
11.2	Recipe Templates & Instances .....	53
11.3	Recipe Viewer.....	54
11.4	Working with recipes in VB.....	55
11.5	Recipes & Security .....	56
11.6	Multilanguage support .....	57
<b>12</b>	<b>Data Logging .....</b>	<b>58</b>
12.1	Basics of data logging.....	58
12.2	Limitations of the system .....	58
12.3	Setting up a Data Set.....	58
12.4	Data logging in VB script .....	59
12.5	Trend Graph Object .....	59
12.6	Using the Trend Graph at Runtime .....	62
12.7	Trend Graph in VB, Actions and Events.....	62
<b>13</b>	<b>Localisation.....</b>	<b>64</b>
13.1	<b>Resources.....</b>	<b>64</b>

13.2	What is a Default Language? .....	64
13.3	Resource Types .....	64
13.4	Viewing and Adding Languages.....	64
13.5	Translating Resources .....	65
13.6	Viewing Different Translations on a page .....	66
13.7	Actions & VB .....	66
<b>14</b>	<b>Intelligent Application Gadgets (IAGs) .....</b>	<b>67</b>
14.1	What Is an IAG?.....	67
14.2	What Is an IAG Collection? .....	67
14.3	How to Install IAG Collection.....	67
14.4	Properties on the IAG.....	68
14.5	Changing IAG Objects From Code .....	68
14.6	Using IAG methods From Code.....	68
14.7	Using VB Variables From Code .....	68
<b>15</b>	<b>Creating IAGs .....</b>	<b>70</b>
15.1	Creating an IAG project (Collection) .....	70
15.2	Structure of IAGs.....	70
15.3	IAG Resources .....	71
15.4	How to develop an IAG.....	72
15.5	User variables .....	72
15.6	Debugging an IAG .....	73
<b>16</b>	<b>Simulation .....</b>	<b>74</b>
16.1	Introduction .....	74
16.2	Benefits of Simulation .....	74
16.3	Standalone Simulation .....	74
16.4	Integrated Simulation of NA and NJ .....	74
16.5	Setting Breakpoints .....	75
16.6	Using the Watch Window .....	75
<b>17</b>	<b>Screen Transfer.....</b>	<b>76</b>
17.1	Communication Setup.....	76
17.2	Synchronise with NA Device .....	76
17.3	Source code Synchronisation .....	78
17.4	Synchronise with Media Device .....	79

## 1 OVERVIEW

The Sysmac Machine Interface, NA Series is the next big release within the Sysmac Platform. Firstly with Sysmac we introduced the NJ machine controller and Sysmac Studio. Next we developed IO and Safety control with the platform, and now we introduce this complete new line up of HMI.

The NA Series is first and foremost part of the Sysmac platform, totally integrated inside Sysmac Studio and is the perfect partner for the NJ machine controller. That said, the NA also completely supports communication with Omron PLCs such as CJ via both FINS & EthernetIP, therefore the NA is the next generation of HMI and a new premium HMI panel for CJ and NJ.

But what is HMI? Maybe a silly question! Everyone knows what HMI is! But Omron believes it is changing. Of course HMI is in the end a screen on a machine which provides an interface to start/stop a machine, to handle alarms etc etc, but the core role of today's HMI is **data transformation**. Transforming data from the machine world (NJ controller) into a human world. A machine controller's role is all focused on precision, speed and accuracy, but the HMI is to present data in a way that can be understood at a glance. Not just on the screen, but also in a report, or a web site, or a database, or anywhere!

Thus for the NA we use the strap line: Bringing Technology to Life. The NA is the bridge between Omron technology and the human and IT world.

Nevertheless, as part of the Sysmac platform, the NA continues to be targeted at FAST. In the case of the NA, FAST is about:

1. Intuitive & Dynamic: setting the machine up quickly, operating in a natural way
2. Proactive & Predictive: keep the machine running through security and rich data e.g. pdf, movie etc. Give the operator the rich information they need to understand problems.
3. Simple but Flexible: a simple to use development environment to quickly build HMI, but not limited by functionality of today.

### 1.1 Architecture

The architecture of the NA is a modern Intel Atom based platform running a standard operating system: Windows Embedded Compact 7 (WEC7). WEC7 is the successor to Windows CE and supports the .NET platform including Silverlight (a smaller lighter version of Windows Presentation Foundation (WPF) that is used throughout Sysmac Studio.

#### 1.1.1 Why did Omron choose WEC7?

The key characteristics of an operating system for industrial HMI are fast, robust, dependable. Because of this need, in the past Omron selected a real time OS like OS9 (used in the current NS) for their HMI. The trouble with such an operating system is that the graphics are not so advanced, and every driver has to be developed by us. In the end this is why it took a long time to be able to support a USB keyboard on the NS. Using WEC7 gives also the benefit of many standard features available to the HMI for example support for png, pdf, wmv etc. It gives Omron the opportunity to scale this platform in the future to a desktop environment using the same technology.

Through testing we can say that WEC7 gives us the reliability that is required, but all the flexibility and extensibility to bring new functionality quickly to the market.

The NA Series is not an IPC, it is a panel HMI designed to turn on quickly, does not need a UPS to make sure it shuts down correctly – it has all the standard functionality expected for a real industrial quality product. As such it is a more closed environment, customers can freely develop in VB, but cannot install their own applications on the NA Series.

## 1.2 Hardware

The design of the NA is based on the overall Sysmac platform. This is why there are the two cut outs on the edge (similar to NJ). The screen is an 'edge to edge' design and the LCD itself is bright and clear.

At first release there is a resistive model available, which is perfect for environments where gloves are required, or there is a high moisture atmosphere. Later a multi-touch capacitive screen with gorilla glass (like iPad) will be available.

All NA models are wide screen and will be available in 7", 9", 12" and 15".

At the bottom of the screen are three function buttons which are fully programmable as part of the HMI design.



One of the goals of the NA Series is about flexibility and this is true for the hardware. Once the standard models are in the market, it is possible to customize the front panel of the NA, either a simple customization of the colour/logo of the surround, or a complete change (note: depending on the business case of course).

As an embedded device, there is no fan on the NA.

The NA supports Dimming from 0-100%, particularly important in some applications (marine). The Dimming can be controlled via the software application so can easily be embedded in the design of the HMI.

## 1.3 Connectivity

The connectivity of the NA is gives:

- 2 Ethernet ports
- 2 USB
- 1 Slave USB (Sysmac Studio)
- 1 SD card
- 1 RS232-C
- 1 24V DC connector



Having two Ethernet ports allows the NA to act as a bridge between the machine world (controller and machine networks) and the IT infrastructure at the factory. This isolation allows far better control of machine data, but allows for easy connection to web resources, or databases or printers etc without compromising security.

The USB ports allow connectivity to a wide range of devices including keyboards and mice, as well as bar code scanners or external speakers.

The RS232-C port is provided for connectivity to older devices that still require a serial connection. This port can be accessed directly from the VB and therefore any protocol can be supported.

The NA has space for an option board to be mounted on the back. In v1.0 there are no option boards available but in the future specific non-Ethernet communication boards could be made, or small IO block etc. This again is part of the customisation possibilities with the NA.

## 2 INTEGRATION

The NA Series is totally integrated into the Sysmac platform. As such, Sysmac Studio has been expanded to include all the configuration and programming of for the NA Series. This brings a wealth of features and benefits to the user, accelerating development of the complete machine since the entire project is in one place.

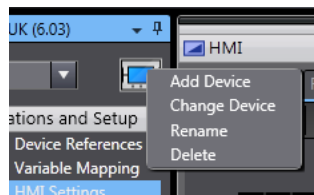
### 2.1 Handling Projects

There are two main ways of using Sysmac Studio to program the NA Series:

1. Creating a dedicated project for NA
2. Adding an NA to an existing project (with the NJ controller for example)

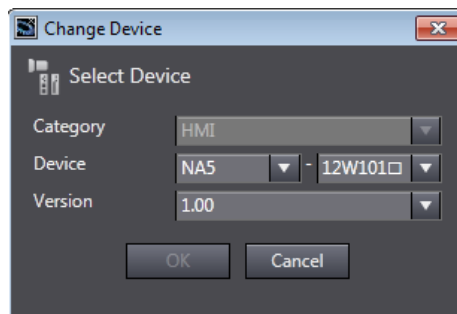
To create a new project for the NA, simply select 'HMI' from the drop down list in the start-up window of Sysmac Studio.

To add an NA to an existing project, having opened the project select Insert | HMI and select the relevant screen size.



When working in a project, it is possible to swap between the devices using the Solution Explorer. When changing devices, all configuration/programming views are closed. This is intentional to prevent later confusion about which device this view is for and to prevent confusion when going online, but can only be online to one device at a time.

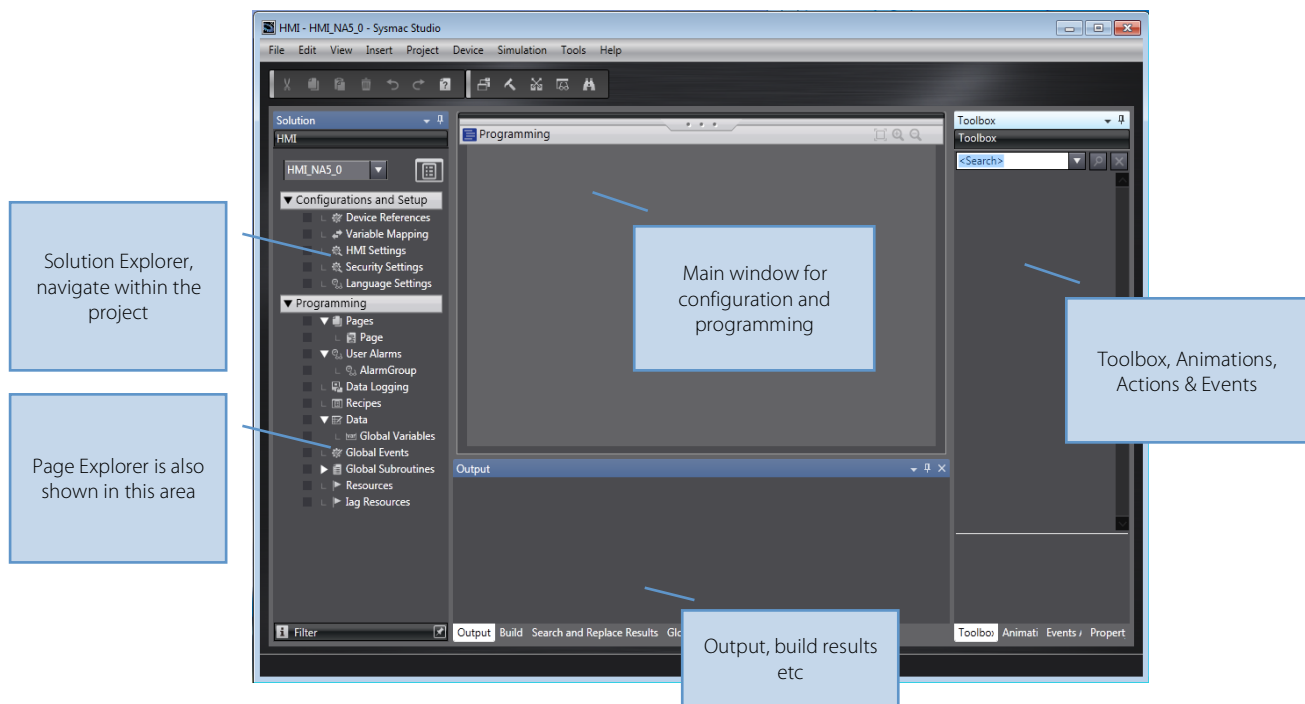
In the same way as a controller, even after an NA is added to a project it is possible to change the device type. Right click on the device icon and select 'Change Device'.



### 2.2 Framework

The diagram below shows the main areas of Sysmac Studio when working with an NA project, it should feel familiar to a user who has already used Sysmac Studio to program the NJ machine controller.





### 2.2.1 Docking Windows

As part of the HMI project docking windows have been added to Sysmac Studio. This is essential since working with HMI requires using many different areas of the screen – often at the same time. Docking windows gives the user the ability to position (almost) all of the windows wherever they choose. The only restrictions are that only some items can be docked within the main window in the centre of the screen.

Docking allows code or pages to be displayed side by side, which makes comparison much easier. It also aids copying and pasting between areas of the project.

The side windows can be unpinned so that they slide and when required simply by moving the mouse into that area. This gives a much greater working area and makes designing pages far simpler.

As well as the flexibility to dock windows anywhere, it is also possible to float windows outside of Sysmac Studio. This makes working on multiple monitors a real benefit.

*Note: the docking feature is not limited to the HMI, docking is now available throughout Sysmac Studio.*

## 2.3 Search and Replace

In v1.0 Search and Replace can only be used to search within VB.net code.

## 2.4 Cross Reference

Not supported in v1.0.

## 2.5 Global Watch

The global watch can be used during simulation and debugging to monitor variable values.

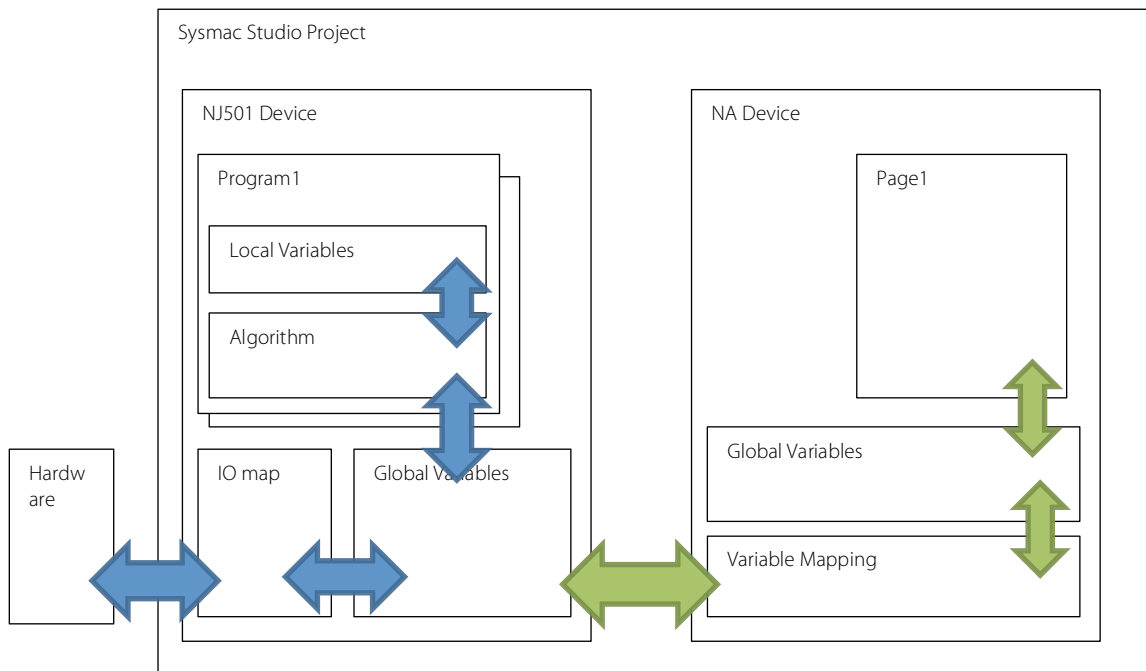
## 2.6 Variable Mapping

This section gives a brief overview of variables and variable mapping within the NA. For more detailed information refer to variables chapter.

### 2.6.1 Variable Mapping between NJ and NA devices in project:

All NJ variables are immediately accessible from the NA, they just need to be mapped in the IO mapping area. NJs within the same project are known as 'internal' devices. No configuration is necessary for an internal device.

The following diagram shows the scope of variables within the Sysmac project:



All NJ global variables can be mapped for reading or writing **regardless** of the 'network publish' setting. It is therefore not necessary to setup network publish in order to access the variables.

Even structures defined in the NJ can be mapped directly in the NA, the data type is automatically converted to the NA. However in v1.0 it is not possible to create user defined data types in the NA.

Besides the NJ global variables, automatically the NA shows a lot of NJ system variables for example the Error\_Table.

NJ local variables (for example in a function block or in a program) cannot be accessed from the NA.

### 2.6.2 External Devices

Further devices can be added as 'external devices', which means devices that are not within this Sysmac project. External devices can be:

1. Additional NJ devices (Ethernet)
2. CJ (FINS or EthernetIP)

Once defined, within the external device configuration, a variable list can be added which can be mapped to the IO. Note: only Ethernet/IP published variables can be used on external devices.

## 2.7 Security

The configuration of HMI users and passwords is not related to the users that are defined within Sysmac Studio to manage the project. For information on the security system in the NA, please refer to the Security chapter.

### 3 CONFIGURATION AND SETTINGS

Within Sysmac Studio all the settings and configuration for the NA Series can be performed. This means that the settings are stored within the project and can be transferred to the device along with the HMI application. This gives the benefit that from opening the box to running an application can require one simple transfer and no further settings need to be made on the NA itself.

The following categories of settings are included in Sysmac Studio:

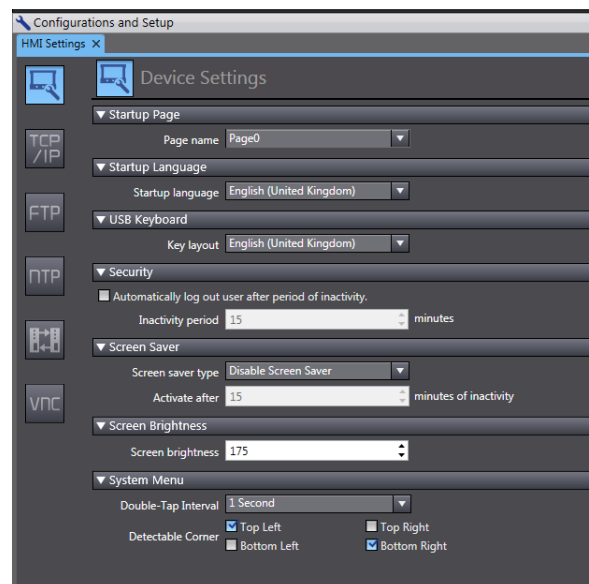
- General Settings
- TCP/IP Network
- FTP Server
- NTP Time Server
- FINS
- VNC

Note: a System Menu exists on the device. Many of the settings in this chapter are configurable via the System Menu. The System Menu will be displayed when no project is found on the device. Alternatively it is accessed by double tapping in a corner (default is top left, but it is configurable in the settings).

#### 3.1 General Settings

There are lots of general settings for the NA:

- The startup page is the page displayed when the HMI is first powered on
- The startup language controls which resources are first used after power on
- A USB keyboard layout to control which keyboard (if any) is connected
- An option to automatically log out after inactivity (and the time)
- An option to setup the screen saver
- The default screen brightness
- The System menu is accessed by double tapping in one of the corners as configured.



#### 3.2 TCP/IP Network Settings

The TCP/IP settings are the usual network settings for a device e.g. IP address, sub net, DNS server. With the NA this needs to be configured for both network ports.

### 3.3 FTP Server Settings

The NA can be configured to run an FTP server which allows external connections to access the file system. A login name and password must be defined.

### 3.4 NTP Settings

NTP allows the NA to synchronise its clock with an Internet time server. Click to enable the option and then configure the server and the update interval.

### 3.5 FINS Settings

Detailed information about FINS is not included in this training book, but the FINS area allows FINS settings to be made for both network ports (network & node number). The port used for FINS communication can be configured as well as a FINS routing table.

Please refer to FINS reference material for further information.

### 3.6 VNC Settings

The NA can run a VNC server to allow VNC clients to connect remotely to the device. By default this option is disabled for security reasons. Click 'use' to enable this option and configure the port, mode and password. There are two modes; either 'view and operate' or 'view only'.



Omron has released an iOS7\* and Android app for Remote Viewing the NA, NS & NB HMI models. This can be downloaded free of charge from the app store (search for Omron HMI Viewer)

This simple app uses VNC technology (for NA) and an integrated web client (for NS & NB). Device configurations can be stored for easy connection at a later date.

IP Address, User name, Password and Port are stored for each connection.

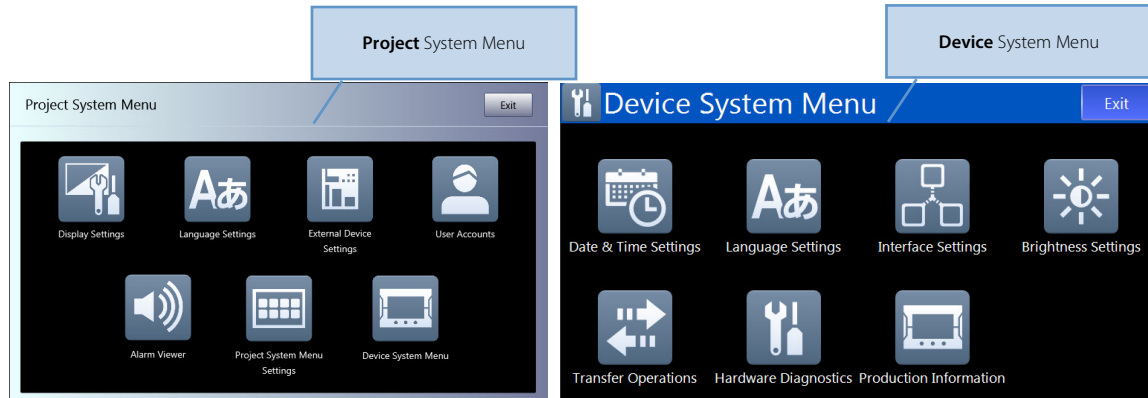
\*iPhone and iPad and Android Tablets (4.2 and above) are supported

## 4 SYSTEM MENU

This section explains the device System Menu that can be accessed on the NA (default by double tapping the top left hand corner of the screen). The System Menu can be used to configure many of the settings from the Configuration & Setup chapter above, but directly on the NA.

There are essentially two main sections to the System Menu.

1. A **project** System Menu
2. A **device** System Menu



When the NA does not contain a project, then when powered on it will automatically show the **Device** System Menu.

### 4.1 Project System Menu

The following areas are the main areas that can be configured from the Project System Menu:

Display Settings	<ul style="list-style-type: none"> <li>• Screen Saver</li> <li>• Brightness</li> </ul>
Language Settings	<ul style="list-style-type: none"> <li>• System Menu Language</li> <li>• User Language</li> <li>• System Language</li> <li>• USB Keyboard layout</li> </ul>
External Device Settings	<ul style="list-style-type: none"> <li>• Communication Driver</li> <li>• Communication Error Indicator</li> <li>• Timeout</li> </ul>
User Accounts	<ul style="list-style-type: none"> <li>• User Accounts &amp; Passwords</li> <li>• Add new accounts</li> <li>• Remove accounts</li> </ul>
Alarm Viewer	<ul style="list-style-type: none"> <li>• View Active Alarms <ul style="list-style-type: none"> <li>◦ View Alarm Details</li> <li>◦ Acknowledge Alarms</li> </ul> </li> <li>• View Historical Alarms <ul style="list-style-type: none"> <li>◦ Export Alarms</li> <li>◦ Clear Alarm Log</li> </ul> </li> </ul>
System Menu Settings	<ul style="list-style-type: none"> <li>• Double tap speed</li> <li>• Detectable corner (default top left)</li> </ul>
NA Device Configuration	<ul style="list-style-type: none"> <li>• Launch the 'Device System Menu'</li> </ul>

### 4.2 Device System Menu

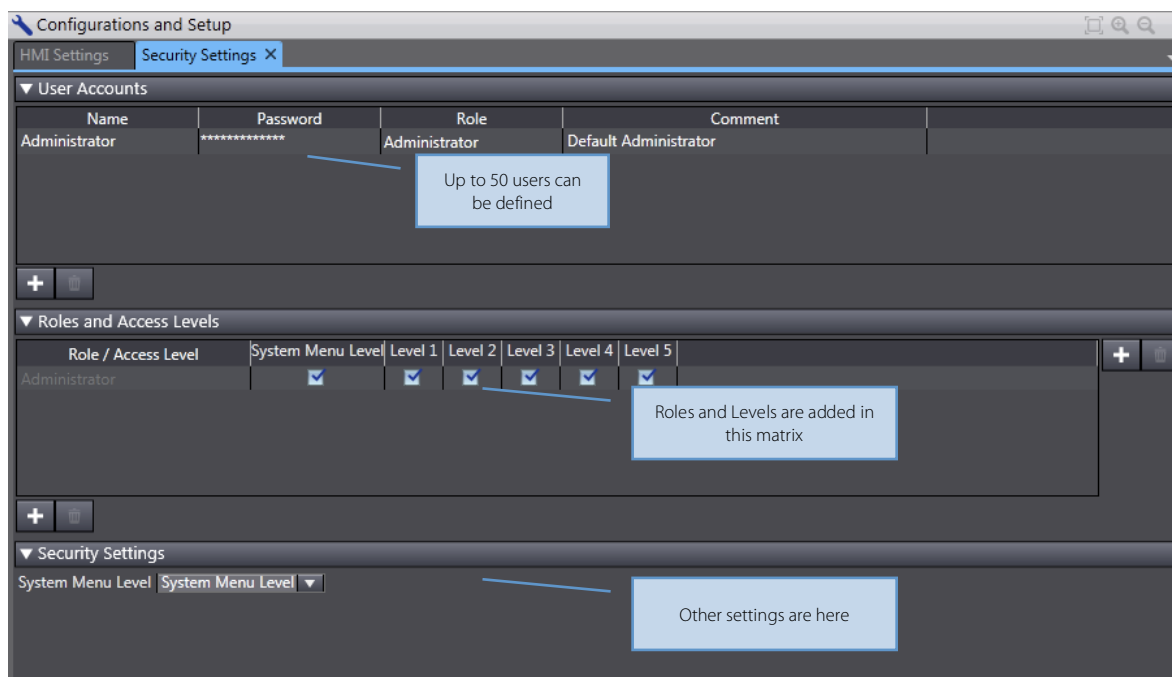
Date & Time Settings	<ul style="list-style-type: none"> <li>• Date</li> <li>• Time</li> <li>• Time zone (including Day light saving)</li> <li>• Use of NTP server</li> </ul>
----------------------	---

Language Settings	<ul style="list-style-type: none"><li>• System Menu Language</li></ul>
Interface Settings	<ul style="list-style-type: none"><li>• TCP/IP settings</li><li>• FTP settings</li><li>• VNC Settings</li><li>• FINS Settings</li></ul>
Brightness Settings	<ul style="list-style-type: none"><li>• Screen Brightness</li></ul>
Transfer Operations	<ul style="list-style-type: none"><li>• Transfer application to HMI</li><li>• Transfer application from HMI</li><li>• Transfer data to HMI</li><li>• Transfer data from HMI</li></ul>
Hardware Diagnostics	<ul style="list-style-type: none"><li>• Screen Calibration</li><li>• Touch panel test</li><li>• Function keys</li><li>• Production information</li><li>• LCD Test</li></ul>
Product Information	<ul style="list-style-type: none"><li>• Production information</li></ul>

## 5 SECURITY

The security settings can be found within the Configuration and Settings area of the solution explorer. Double clicking on 'Security' loads the settings page. The security settings are divided into three main areas:

Users	New users can be configured, passwords set and roles assigned. Up to 50 users can be defined.
Roles & Access Levels	In this area the roles and levels are configured in a kind of matrix. Up to 20 roles and 20 levels can be defined and both roles and levels can be renamed to meaningful names.
Security settings	In v1.0 the other settings include an auto log out following inactivity and a system menu level.



### 5.1 Users

By default an 'Administrator' user is always included in an NA project with the password of 'administrator'. It is possible (and even recommended) to rename this user and change the password. It is even possible to remove this user, but only if further users have been defined.

To add a user simply click the '+' button. To delete a user click the bin icon next to the '+'. A user name can contain up to 40 characters. A password must be defined for each user, which has to be between eight and thirty-two characters (same rules as the NJ).

A comment can be defined to better describe the purpose of this user.

A role must be selected for each user (see below on configuring roles & levels).

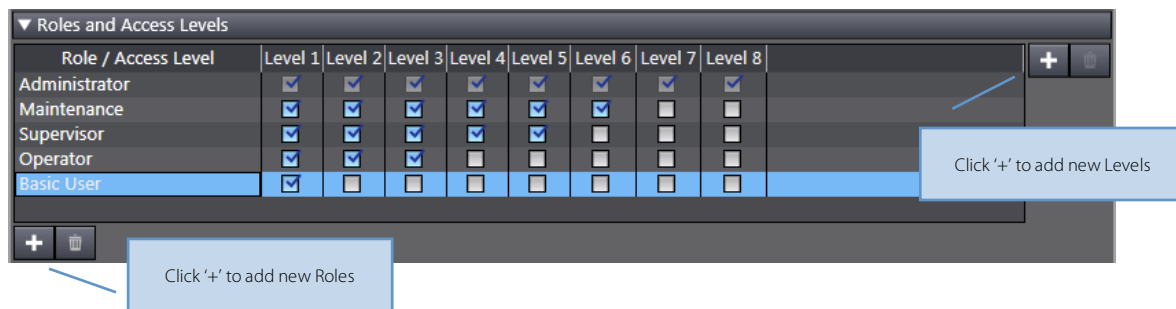
### 5.2 Roles & Access Levels

Roles and levels are configured in a matrix. Each level is distinct – that is to say that if a user has Level 2, he does not automatically have Level 1. Both columns would have to be selected in the Role.

It is not possible to rename or delete the Administrator Role.



New Roles can be added clicking the '+' at the bottom left, new Levels can be added by clicking the '+' on the right hand side. Both Roles and Levels can be renamed to meaningful names e.g. Basic Access, Full Control, Operators, Administrators etc.



It is possible to add up to 20 Roles and 20 Levels.

When assigning these roles to users, bear in mind that when these settings are applied to objects there are two settings: Accessibility Level, and Visibility Level. That is to say that it is possible to configure it so that some users can see some buttons/lamps etc but cannot access them, whereas some users may not even see those buttons/lamps on a page.

In the example shown above, having set the access up as a triangle, it is possible to treat the levels as a hierarchy – but again actually they are all distinct levels.

### 5.3 Using Security on Pages

Security can be used on any of the objects on a page, and there are two independent settings that can be made:

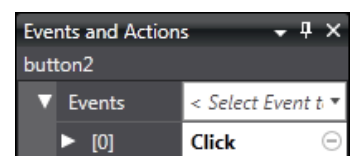


1. **Access Level:** The access level sets the level that is required in order to be able to use the object. Taking a button as an example, it is not possible to press the button and therefore trigger any of the events on that button if the user does not have the access level defined.
  - When a user does not have access level, the button is greyed out and shown as disabled (unless they also do not have visibility level, see below).
2. **Visibility Level:** The visibility level controls the ability for the user to see that object on the page. If they do not have the visibility level for a given object, then it will not be displayed during the runtime operation. Obviously to give them access level, you must also make sure they have visibility level.

### 5.4 Actions

Buttons can be linked to actions to Login & Logout. No parameters can be specified so when linking to the Login action, this brings up the Login window for the user to add his or her details.

For more advanced use of Login, the user has the ability to write some code. See the following section.



### 5.5 Security in code

There are several ways to use the security functions from within code. Firstly there are two routines to use the basics of Login & Logout. There are two ways to use Login. If used without parameters, then the default Login window will be displayed for the user to enter their details. Optionally it is possible to hard code the username and password and get the code to Login automatically.

```
1 'Code behind Page - Add local subroutines for the page.
2
3 Private Sub Button1_Pressed
4     'Script to Login as User1
5     Login("User1", "password")
6 End Sub
7
8 Private Sub Button2_Pressed
9     'Script to Logout
10    Logout()
11 End Sub
```

It is also possible to find out the username of the current user:

```
3 Private Sub Button1_Pressed
4     'Script to Login as User1
5     If _HMI_CurrentUserName = "Operator" Then
6         Login("Supervisor", "password")
7     End If
8 End Sub
```

## 5.6 Security at Runtime

Only one user can be logged in a time. When a new user logs in, any previous user is automatically logged out.

If a user fails to login 5 consecutive times, then it will be impossible to log in (as any user) for a period of 10 minutes.

Users can be modified, and new users added at runtime by using the System Menu. Changes can be uploaded into Sysmac Studio.

## 5.7 Limitations in v1.0

- There is no concept of password aging.
- No ability to get level information from code.
- Security can only be applied to on screen objects, not alarms, recipes, datalogs etc. Therefore the security must be applied simply to prevent access to the areas required using on screen object protection.

## 6 PAGES

Pages are at the heart of any HMI design application. Learning how to use the page editor well will enable great looking HMI's to be created.

Objects can be placed on the page simply by drag & drop from the toolbox (covered in another module). Once these basic objects are placed on the page we need to change their appearance, their behaviour and add any actions.

- Properties control the appearance of objects on the page; changing properties will change the look of the object from the default.
- Behaviour is controlled either by 'Variable/Expression' properties which link specific behaviour of an object to a variable or mathematical expression, or by adding animations to an object.
- Actions are defined to make an object do something when an event occurs such as a button being pressed.

Each of these areas will be considered in detail later on.

### 6.1 Graphics in the NA

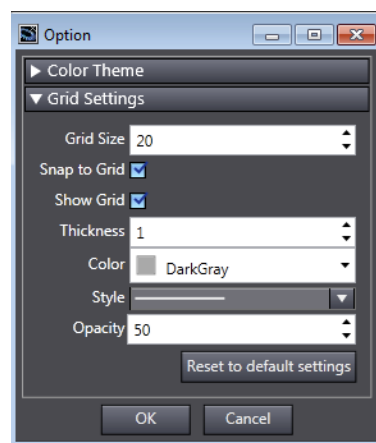
The NA Series HMI uses vector based graphics. Images (bitmaps etc) can be used within the HMI, but all the default objects and graphics are vector based which gives them high quality regardless of the size they are used. That said, Sysmac Studio is not a graphic design package, and should not be compared with Adobe Illustrator, or SolidWorks etc.

Sysmac Studio provides the basics to use high quality vector graphics, bring them to life with animations, and link to machine data to control. It is however possible to draw complex graphics in Sysmac Studio using lines, arcs, polygons etc.

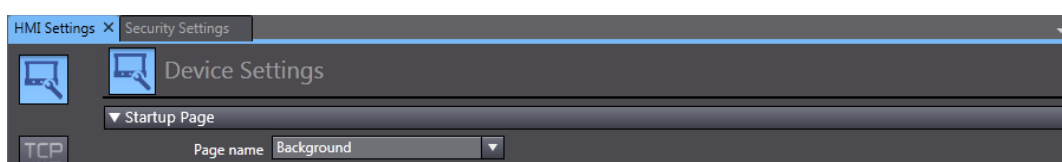
### 6.2 Page Basics

A grid is displayed in the background of the page to help users align objects. Optionally the user can choose to 'snap to grid' so that each time an object is placed or moved it will snap to the nearest grid.

By default the grid is enabled with 'snap to grid'. The size of the grid and colour of display can easily be changed in 'Tools | Options':



Another important setting is the 'Startup Page'. This is where the user configures which is the first page they want to load when the application starts up.



When a page is loaded, in the top right hand corner there are some zoom controls to make it easier to visualise the final HMI, or make precise adjustments the graphics.

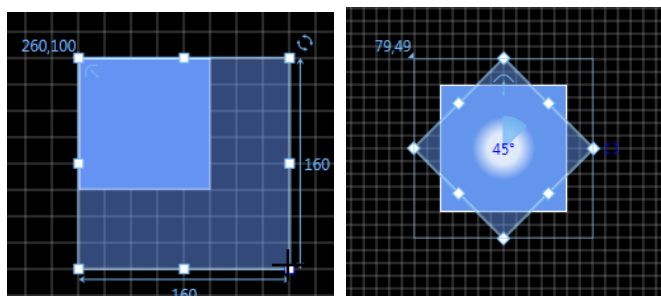


Pages can be inserted in the Solution Explorer, simply right click and select Add Page. Pages can also be grouped to give a hierarchy and structure to the project. By default all pages are full screen.

Optionally pages can be configured as popups that have a custom size. Popup pages can be set to load in different positions of the screen, and even can be loaded in a specific place determined at runtime (see ShowPage action below). Popup pages can also be set to either Modal or Modeless display. When a modal page is displayed, it must be closed before any other objects can be touched. A modeless window is displayed in parallel with the main page, and both can be pressed at the same time.

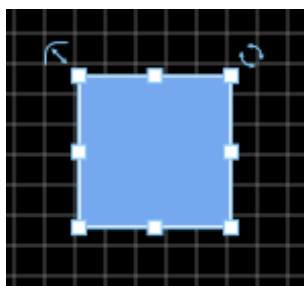
### 6.3 Getting Started with Pages

Once an object is placed on the page, it can be resized using any of the grab handles around the object. When resizing small indicators of size and position are displayed making it easy to get a specific size.



The circle on the corner grab handle can be used to rotate an object. When rotating the angle of rotation is shown in the middle of the object so that precise rotation can be achieved.

The small symbol in the corner of some objects (rectangles, buttons etc) can be used to give a rounded corner to the object. Once again small indicators show the size of the rounding. This is called the CornerRadius and the properties can be changed directly in the properties window if required.



It is also possible to change the size and position of an object using the properties window:

▼ Layout		
▼ Position (Left,Top)	260, 100	
Left	260	
Top	100	
▼ Size (Width,Height)	100, 100	
Width	100	
Height	100	

By expanding the Position & Size section, the user is able to enter an exact position or size.

Objects can be grouped together by selecting them and clicking 'Group' on the right click menu. To select multiple objects either click and drag the mouse over the objects, or select object by object whilst holding Ctrl.

When Objects are grouped together it is possible to change the properties of one of the objects in the group. This can be done using the Page Explorer (View | Page Explorer if not already docked) and selecting the specific object.

Note: the right click menu also allows for standard graphic functionality like 'Bring to Front', Cut/Copy as well as alignment and flipping.

## 6.4 Using the Property Window

To change the appearance of any object the user must use the property window. Properties can either be displayed in categories (default) or alphabetically. It is also possible to search within the property window to show a list of available properties.

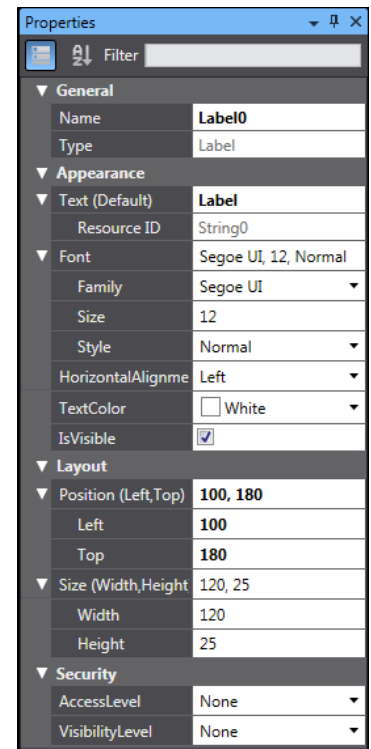
If multiple objects are selected, just the properties that are common to both objects will be displayed. Changing values will affect both objects.

All objects have the following basic property categories:

- **General:** contains the object name and type
- **Appearance:** contains the basic properties relating to the look of the object
- **Layout:** size and position
- **Behaviour:** some objects have built in behaviour such as showing a value of a variable

Unless the property is a Variable or Expression it is not currently possible to link them to variables. In other words, it is not possible to link the scale of a gauge to a variable and then dynamically change the scale. Note: this may be supported in future versions.

All objects have an 'IsVisible' property. This sets the default visibility of the object which could be changed later using an animation or VB code. Objects which are invisible do not respond to any actions. For an invisible object that does respond to actions, use 'Transparent' as the object colour and border colour.



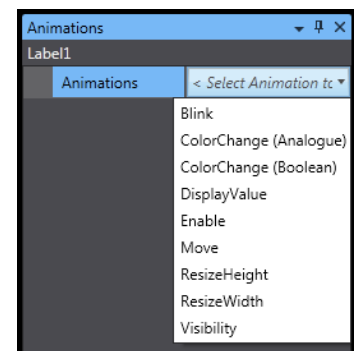
Some properties however can be accessed and changed at the runtime using VB. See the chapter on VB for more information.

## 6.5 Animations

Animations are what bring a page to life. They enable objects to move, colours to change and visibility to be controlled. They are the dynamic behaviour at the runtime.

The animations view (View | Animations if not already docked) controls all the animations that can be applied to any object. To add an animation the user clicks the drop down and selects a specific animation. The following animations are supported:

- **Blink** [Blink the object with a colour]
- **ColourChange** (Analogue) [range of colours]
- **ColourChange** (Boolean) [two state colour change]
- **Enable** [Allow operation of the object e.g. a button]
- **Move** [Move the object x & y]
- **PercentageFill** [Fill the object with its foreground colour]
- **ResizeHeight** [Change the height of the object]
- **ResizeWidth** [Change the width of the object]
- **Visibility** [Show or hide the object]



Each animation is then configured with an expression to control that animation.

An expression can be just a variable name, or any mathematical calculation or logical expression based on variables or fixed values for example: 'myVar \* 5' or 'myBool1 & myBool2'

Some of the animations require more complex configuration, for example Colour Change (Analogue). This animation can be configured with a range of colours for different expressions. These configurations are called thresholds.

## 6.6 Events & Actions

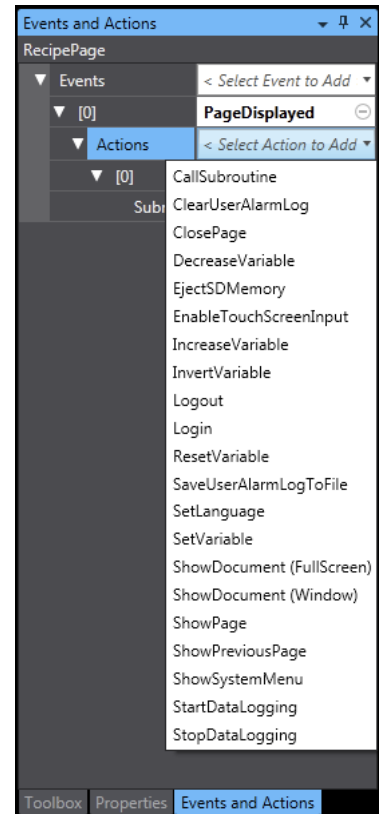
Events are triggered when the user interacts with an object, in the case of a button an event is Click, Press or Release. Actions linked to the event are the thing that is to be done when this event occurs for example 'ClosePage'.

Whilst it is possible to develop all the actions directly in VB, there are a number of standard actions which are built into the system so that in some cases users don't even need to use VB.

It is possible to add multiple actions on one event. They are executed in the order in which they are defined.

V1.0 supports the following actions:

- CallSubroutine [calls a VB method]
- ClearUserAlarmLog [clears the history of alarms]
- ClosePage [closes the specified page]
- DecreaseVariable [reduce a variable by a specified amount]
- EjectSDMemory [eject the SD memory]
- EnableInputOperation [enable/disable the touch screen]
- IncreaseVariable [increase a variable]
- Logout [log the current user out]
- Login [display the login screen]
- ResetVariable [set a variable to false/initial value]
- SaveUserAlarmLogToFile [save the alarm log to a storage device]
- SetBrightness [change the brightness of the screen]
- SetLanguage [change the current language]
- SetVariable [set a variable to a specified value]
- ShowDocument [show a document (pdf) full screen or windowed]
- ShowPage [show a new page, optional paramters can set the position of a popup page]
- ShowPreviousPage [load the previously loaded page]
- ShowSystemMenu [display the system menu]
- StartDataLogging [start the data logging]
- StopDataLogging [stop the data logging]



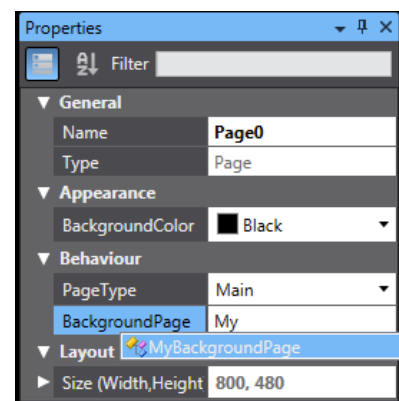
## 6.7 Layering Pages

In the NA it is possible to layer pages, this done by defining a 'Background Page' for a given page. The background page is then loaded in the behind the current page. In the IDE it will be displayed faintly in the background so it is clear which objects are on the current page.

Layering pages allows the user to place all their navigation buttons, login/logout, date and time etc on a master page, which is then used as a background for all other pages.

Any page can be used as a background page, as long as it does not cause a circular reference.

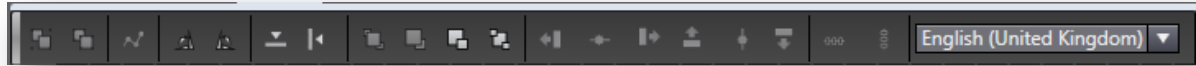
Popup pages cannot use Background pages in v1.0.



*Note: a page must be configured to have either a background colour or a page, but not both. If a colour is specified the background page will not be displayed.*

## 6.8 Page Toolbar

A page toolbar with frequently used graphic operations such as flip, align and rotate is displayed at the bottom of each page. This can be used to quickly access these common features.



One of the features of this page toolbar is the language dropdown. This dropdown can be used to change the currently shown language. This can make it easy and quick to validate translations and ensuring the localisation works on all the objects. More information about translation is handled in the translation module.

## 7 TOOLBOX

When editing a page within the project, the toolbox becomes the source of all the visible items that can be put the page. The toolbox appears as a pane on the right side by default. If the toolbox is not visible, select toolbox from 'View' Menu.

The Toolbox Contains:

- Shapes – e.g. polygons and circles
- Objects – e.g. buttons and gauges
- Graphics – complex shapes such as pipes or motors
- IAGs – pre-configured graphics with functionality
- Custom Objects – user customised objects with properties, animations & events

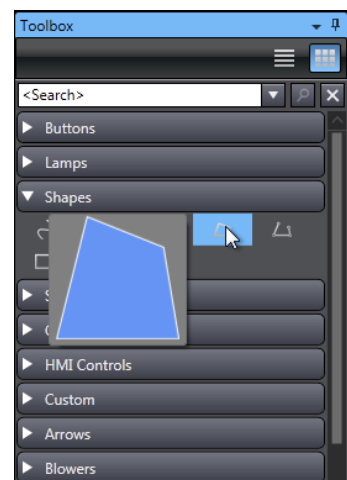
These items in the toolbox can be added to the page using a drag & drop operation.

### 7.1 Using the Toolbox

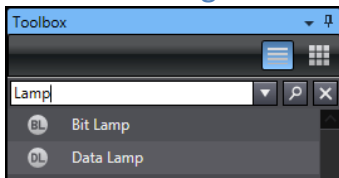
The toolbox is a hierarchical menu system where each section can be opened up to reveal the contents. As each section is opened the previous section closes. To open up a section you must click on the section header. When you hover over each object or shape you are given a preview.

To add an item to the page, click on the shape or item and drag it to the page. Once on the page you can configure the item using the properties window. For more information on general properties, see the Pages chapter.

There are two main ways of viewing the items in the toolbox. By default, the objects are presented in a list view with a series of small icons with text to describe them. By pressing the 'Grid View' item in the top right hand corner of the toolbox it is possible to view the toolbox as a set of small icons to represent the objects.



### 7.2 Searching the Toolbox



The search bar at the top of the toolbox allows you to search the entire toolbox for an object or shape

Type the search term e.g. 'Pipe' into the search bar and press return. Drag the required object and place on the page.

Press the 'X' to clear the search.

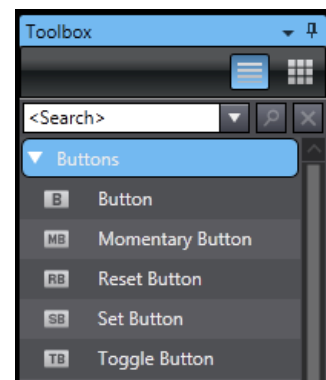
### 7.3 Objects: Buttons

Buttons are some of the most commonly used objects within an HMI application. The NA series supports five basic types of buttons. These five buttons cover all the basic needs for a button in HMI, as well as being completely programmable to tailor the function even further.

- Button – simple button with no special action
- Momentary Button – Sets a bit while the button is held
- Reset Button – Resets a bit to false when button is pressed
- Set Button – Sets a bit to True when button is pressed
- Toggle Button – Toggles a bit between True and False with each press

Besides the standard functionality of the buttons above, each button supports the following events so that further actions can be defined:

- Click - When you press and release over the button (use when cancel operation is required)
- Press – When you press down on the object
- Release – When you release the button/object





*Note: It is not possible to change the type of button after it is on the page, therefore selection should be made carefully.*

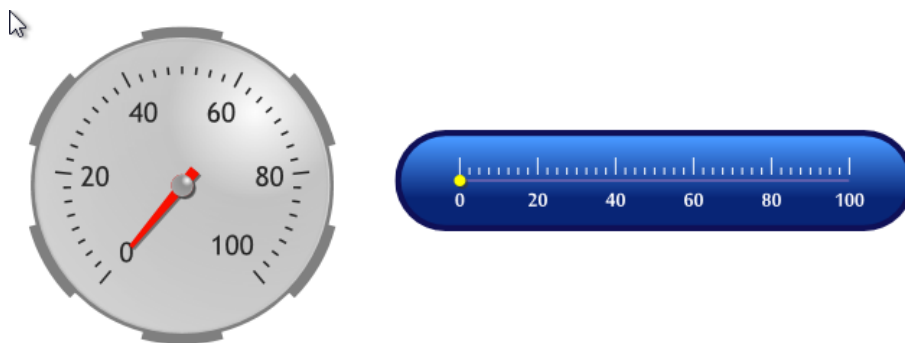
Buttons can also be defined as to use an image (from the resources area) rather than a colour. It is then possible to define both an Up and Down image to completely change the appearance of the button. In this way complex button styles can be used.

## 7.4 Objects: Gauges

A gauge is a visual way to display a single value. As such the gauge object has an expression property that allows a variable to be linked to the object to display the value at runtime. Gauges support all number data types.

Two basic types of gauges are included with Sysmac Studio, namely a rotational gauge and a linear gauge. The linear gauge can be configured to be either vertical or horizontal.

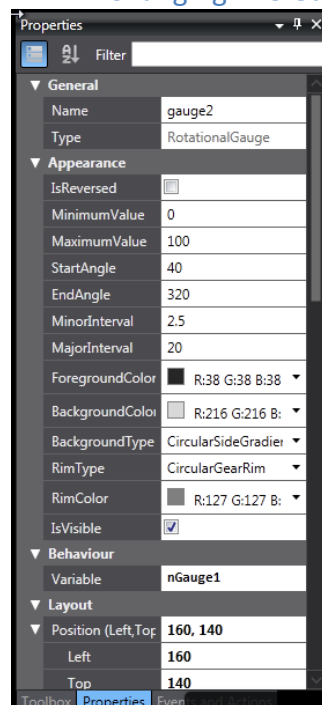
Gauges have many properties so that the appearance can be changed to the desired look. To help the user, a whole set of themed gauges are available for both rotational and linear gauges.



Gauges use the following terms in their properties:

- Ascent** – The X axis of the gauge
  - Ticks and Needle thicker on the Rotary gauge
  - Ticks and Needle longer on linear gauge
- Extent** – This is the width of the tick, line or needle
  - Ticks and Needle longer on the Rotary gauge
  - Ticks and Needle thicker on linear gauge
- Offset** – The distance from the scale bar

### 7.4.1 Changing The Gauge Appearance



**Is Reversed** – clockwise or anticlockwise gauge

**Minimum and Maximum Values** – scale of the gauge

**Start and End Angles** – the angle of the needle at the top and bottom of the range

**Minor intervals** – frequency of small ticks on the gauge

**Major intervals** – frequency of large ticks and the labels on the gauge

**Gauge text foreground colour** – colour of the Text

**Is Rim Effect Enabled** – a graphical effect on the Rim

**Rim Type and Colour** – shape and colour around the edge

**Background Colour and Type** – colour and type of the background

### 7.4.2 Ranges

A range is a colour on the gauge that can be configured to be multiple colours at various points, start at a certain place and end. For example the optimum running temperature may be 95 – 105 degrees so this can have a Green range on the gauge.

Multiple ranges can be defined for a gauge just by clicking the add button in the ranges section of the properties.

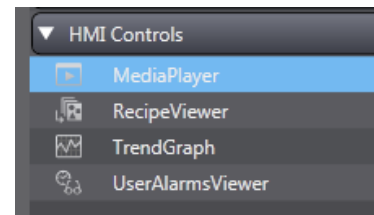
The ranges can be stepped away from the scale bar by setting the Scale Offset.

Range marker width can be changed and graduated by altering Start and End Ascent/Extend.

## 7.5 Objects: HMI Controls

This section contains more complex controls specific to HMI.

The Media Player can be embedded on pages to play videos (see below), the Recipe Viewer displays recipe templates and instances (see the Recipes chapter), the Trend Graph is linked to a data set and displays the values for analysis (see the Data Logging chapter) and User Alarms Viewer shows either current or historical information on alarms (see the Alarms chapter).



## 7.6 Objects: Data Display

This object is used for displaying data. Much like the Label object can display text, this object allows you to display the value of a variable or the result of an expression.

In the properties, the data type can be changed between Boolean, Numeric and Text.

When using Boolean, you can choose what is displayed when the value is 0 or 1 e.g. Stopped or Running.

The numeric value can be formatted in one of the following ways:

```
#
#.#
#.#.#
#.#.###
##
##.#
##.#.#
##.#.###
###
###.#
###.##
###.###
Custom
```

You can also choose whether to display the number in Binary, Hex or Decimal. When using Text you can choose the encoding type.

## 7.7 Objects: Data Edit

This object is used for data entry. It is similar to the data display object, but this object allows the user to enter a new value. It is linked to the value of a variable.

The data edit object supports the same formats as the data display object.

It is possible to set a background colour and border colour/thickness for the data edit object. This makes it easy to highlight the object for users to enter new values.

## 7.8 Objects: Media Control

This control is a simple media player allowing videos or animations to be played on the HMI. It is ideal for playing videos to help the operator deal with problems or servicing on the machine.

From the properties the user chooses which video will play from the HMI resources.

The following formats are supported in v1.0: MPEG 1, MPEG 4 or WMV Format.

Using the resources area, it is possible to automatically play different videos depending on the current language of the HMI. See the chapter on Localisation.

Global functions are provided to start and stop the videos, but it is also possible to call 'MediaPlayer1.Play/Stop/Pause'

## 7.9 Objects: Shapes

Shapes provide the user with the ability to draw complex graphics for parts of their machines. Vector graphics enable the HMI to easily achieve a high quality look regardless of the size of the graphic.

The following objects are supported in v1.0:

Ellipse	Drag object on page and resize or change height/width.
Rectangle	Basic rectangle or square
Triangle	Simple Triangle
Polygon	Fillable user defined shape with as many points as defined
Polyline	A user defined line
Line	A straight line
Curve	A multi point curve

All shapes have settings to change the appearance including colours and line styles.

The Polygons & Polylines can be further edited by right clicking and selecting 'Edit'. Whilst in edit mode further points can be added, or points moved to create a new shape.

## 7.10 Objects: Lamps

There are two types of lamp available to use: Bit lamp and Data Lamp. A Bit lamp will switch on and off based on the value of a bit. A Data Lamp will change its behaviour based on an expression it may have multiple states with different appearances (colour or bitmap) depending on the result of the expression.

To configure a data lamp, use a variable in the expression property and then set thresholds in the Animation tab.

Bit Lamps and Data Lamps allow 3 different types:

- Ellipse
- Rectangle
- Image

Additionally, you can change the on and off colours and supply labels to display text based on the current state.

Lamps also have events that allow you to trigger an action at the same time the lamp changes or when the user touches the lamp.

## 7.11 Objects: Standard Controls

These are standard controls used in HMI and will be familiar to VB.NET users. Most standard controls will change a variable and also have ability to trigger events

**Checkbox** toggles a Boolean variable

**Date time** – Shows a date from a variable

**Dropdown and Listbox** – Choose from a list

**Image** – Displays a picture

**Label** – Shows a word or a value

**Radio button** – Select a single button from a group

**Slider** – Change the value of a numerical variable

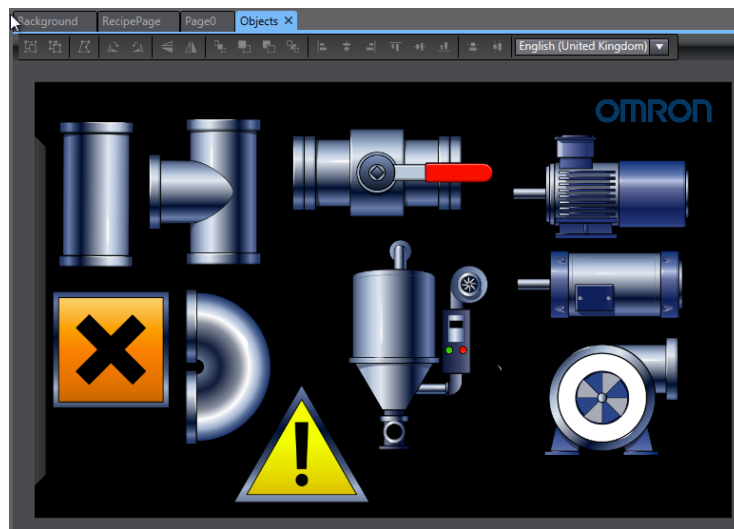
**Textbox** – Freeform text entry

## 7.12 Objects: Graphics

In addition to primitive objects and shapes there are some graphic objects that depict real items such as pipes and boilers, pumps, blowers etc. These are also located in the toolbox. Simply drag the item onto the page. There is a large choice of graphics so you can browse or make use of the search.

The graphics are all vector based and built up of many grouped parts. Once on the page, using the page explorer you can easily modify the look of any of the objects. Additionally you can 'ungroup' the object and remove or animate parts if needed.

In the future it will be possible for users to import their own vector-based images and convert them into Sysmac Studio objects. In v1.0 this is possible, but only with the help of Omron.



## 7.13 Adding Custom Items To The Toolbox

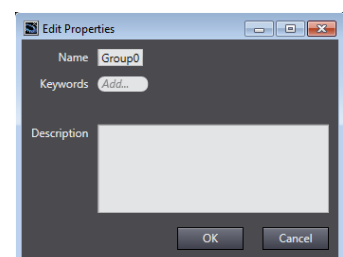
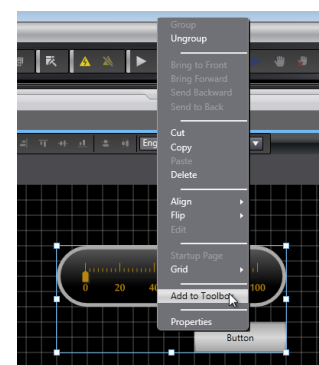
It is possible to add further items to the toolbox (for example objects with specific properties, animations or actions & events). This is ideal for objects that are required to be used frequently. It saves the time to customise the objects. Even groups of objects can be added to the toolbox. To add an object or group of objects, right click on the item and choose 'Add to Toolbox'.

Expressions, variables animations and events & actions used by these items are also saved into the toolbox. Therefore when used again, these variables must exist within the project, or the object will require to be linked to new variables.

Note: code used by these objects (e.g. subroutines) will **not** be stored in the toolbox. This is a key difference between custom objects and IAGs.

Once an item is added to the toolbox, right click it and change the properties as follows:

- Name – the name as it appears in the toolbox
- Keywords – search keywords to help users find this custom object
- Description – the long description shown at the bottom of the toolbox when the item is selected



## 8 VARIABLES

Variables are at the heart of the NA Series Machine Interface. The NA is a variable based system, so learning how to configure and use variables properly is very important.

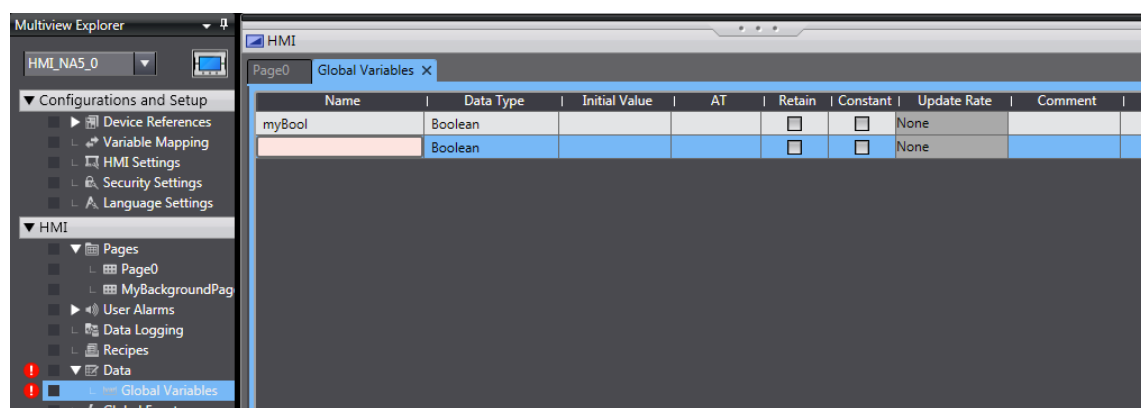
There are several different kinds of variables that exist in the NA:

- System variables
- Global HMI variables
- Internal device variables (other controllers within the Sysmac project)
- External device variables (other NJ/CJ controllers)

Like in a controller, to use device variables they have to be mapped. This then creates variables for this mapping and they appear in the global variable list.

### 8.1 Global Variables

Global variables editor can be found in "Programming – Data – Global Variables" section. The user can create new variables in this section:



The following properties are available for mapped and not mapped variables:

1. Initial Value  
A variable can have an initial value applied upon runtime initialisation. This can be applied to all basic data types, but cannot be applied to a Network Variable mapped to an NJ Global variable.
2. Retain  
The Retain property will cause the last known value of a variable in runtime to be retained and automatically re-applied when the HMI re-starts, in the following list of circumstances.
  - Power off \ power on cycle
  - Stop and Restart runtime execution

*NOTE: If variables have both the Retained property enabled and an Initial Value applied, the retained value takes precedence. On runtime initialisation the retained value overwrites any Initial Value attributed to the variable.*

The Retain property can be applied to all basic data types. When applied to an array variable, the Retain property retains and applies the value of each array element individually.

The Retain property cannot be applied to a mapped Variable.

The user can choose to not initialise the retained values when downloading a project.

## 3. Constant

The constant property makes the variable a constant, with the value as defined in the 'Initial Value Field'. The value of the variable cannot be changed **programmatically** and any attempt to change the value in the application by assignment should result in a compilation error.

The Constant property can be applied to all basic data types, and to array variables.

The Constant property cannot be applied to a mapped Variable.

## 4. Comment

A description can be defined up to a max length of 2048 characters

## 5. AT Field

The AT field describes the mapping relationship to a device and the data location in that device. The user should not need to understand the format.

## 6. Read-Only Property

Field defining the Read-Only property of a Network Variable. Read-Only variables can be read from an external device, but are never written to it.

## 7. Update Rate (mapped variables only)

Mapped variables have their values **read** at regular intervals from the registered devices (by default this is every 500ms). The Update Rate defines the regular Interval at which the value is read. The update rates that can be applied are fixed within the system and limited to values in the following list.

```
None (i.e. the variable is write-only)
100 Milliseconds
500 Milliseconds
1 Second
2 Seconds
5 Seconds
10 Seconds
30 Seconds
1 Minute
5 Minutes
10 Minutes
30 Minutes
1 Hour
```

In Version 1.0 there is no direct means for forcing an update (e.g. via a script "Read" method).

## 8.2 Data Types

The following data types are available (**and arrays of those types**) for the NA Series.

The table below demonstrate ALL NA data types and its specific currently supported device equivalents.

If a controller data type is entered (IEC61131), then it will be automatically converted to the relevant NA data type.

*Please note the variable data types will/can be different to the PLC/Machine Controller, as NA is VB not IEC based system. NA is a 32bit data type system.*

NJ501 (and Sysmac Studio variables)				
Data Types	Alignment	HMI Data Types (VB.Net)	CJ2 Data Types	Comment
INT	2 bytes	Short	INT	
DINT	4 bytes	Integer	DINT	
LINT	8 bytes	Long	LINT	
UINT	2 bytes	UShort	UINT	
WORD			WORD	See Note 1
			UINT-BCD	

UDINT	4 bytes	UInteger	UDINT	See Note 1
DWORD			DWORD	
			UDINT-BCD	
ULINT	8 bytes	Ulong	ULINT	See Note 1
LWORD			LWORD	
			ULINT-BCD	
REAL	4 bytes	Single	REAL	
LREAL	8 bytes	Double	LREAL	
No equivalent data type		Decimal	No equivalent data type	
BOOL	2 bytes	Boolean	BOOL	
STRING	1 byte	String	STRING	
No equivalent data type		Char	No equivalent data type	
SINT	1 byte	SByte	No equivalent data type	
USINT	1 byte	Byte	No equivalent data type	
BYTE	1 byte		No equivalent data type	
TIME	8 bytes	System.TimeSpan	Not supported	See Note 3
DATE	8 bytes	DateTime	Not supported	
DATE_AND_TIME			Not supported	
TIME_OF_DAY			Not supported	
		Not supported	TIMER	See Note 2
		Not supported	COUNTER	
		Not supported	CHANNEL	
		Not supported	NUMBER	

**Note 1:** VB.Net does not have a BCD data type. However unsigned BCD data types values can be represented by unsigned integer data types

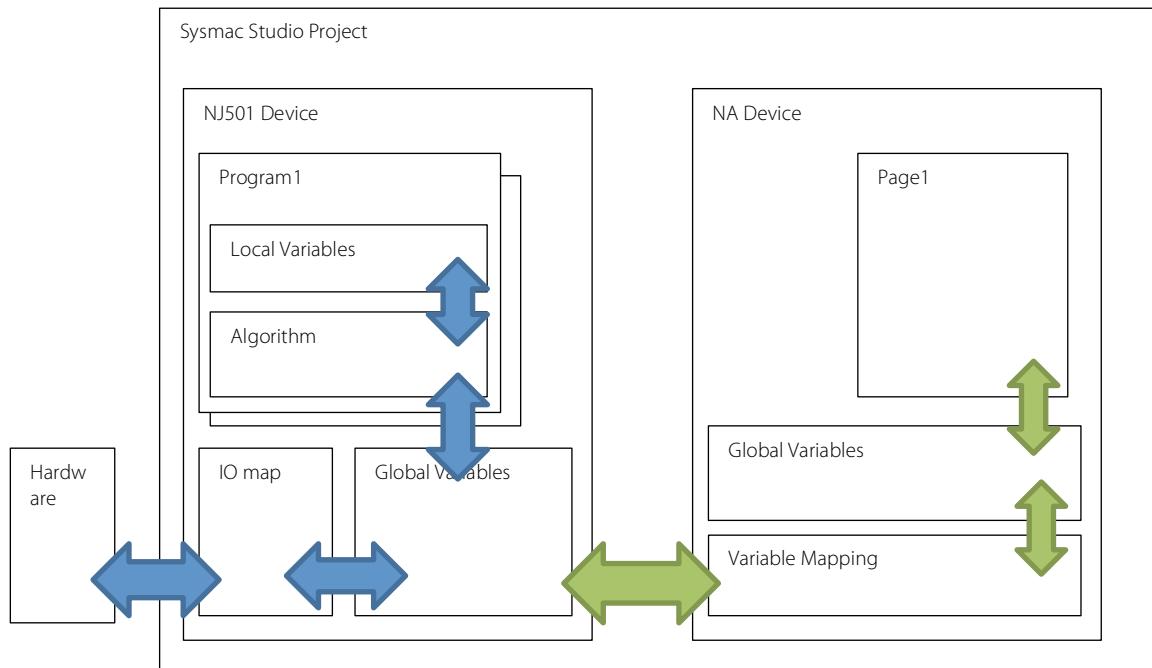
**Note 2:** CJ device Timer and Counter values, (PV) cannot be represented as published symbols, (unsupported for these data types), therefore they cannot be addressed using symbolic addressing- However they can be accessed by using physical memory addressing with the appropriate data type. (Unsigned BCD or Unsigned Binary data type dependent upon configuration in device).

**Note 3:** TIME has no direct equivalent data type and in VB.Net is represented by the TimeSpan structure

For version 1.0 the structures (and Arrays of structures) are expanded to the basic data type level. Mapping a structured variable will automatically create an NA structure to match the external device. This structured data can then be used within the NA.

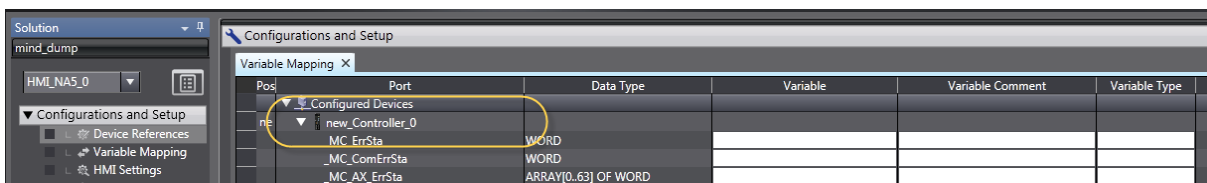
### 8.3 Mapping to controllers within the Project

The diagram below shows how Sysmac Studio maps/shares GLOBAL variables between different devices, within **one** Sysmac Studio project:



All variables created in the NJ **within the Sysmac Studio project** are accessible by the NA. They do not have to be defined as a network publish variable. Note: external devices can only use published variables.

To map to NJ variables, use the 'Variable mapping' window in the 'Configuration and Setup' area:



When expanding the desired controller that is inside the overall project, all the variables will be listed. This includes the global variables (including user defined data types), as well as many system variables for that device.

To map to the variable, right click on the desired variable, 'Create device variable' action, will create NA variable automatically, and map it the NJ variable (Note: Multi select, multi auto create is supported.)

Once mapped, global variables will be available in programming tab

new_Controller_0_global_var1	Boolean				
------------------------------	---------	--	--	--	--

In addition:

- Mapping NJ variables to already existing NA variables is possible.
- Mapping NJ System variables is supported
- Mapping NA system variables to NJ variables is supported.

_EIP_TagAdrErr	BOOL	A_test_bool
_EC_ErrSta	WORD	test_bool1
_EC_PortErr	WORD	_HMI_IsBatteryLow
_EC_MstrErr	WORD	_HMI_IsScreenSaverActive
_EC_SlavErr	WORD	_HMI_IsSDPowered
_EC_SlavErrTbl	ARRAY[1]	_HMI_RunSignal
_EC_MacAdrErr	BOOL	CJ2H_test_bool
_EC_LanHwErr	BOOL	new_Controller_0_global_var1

## 8.4 Limitations:

- No scaling is supported.



- The maximum number of variables is limited to 90000.
- The maximum number mapped variables connected to an external device is limited to 10000.
- The maximum number of external device references allowed in an HMI application is 32.
- The maximum number of variables that can have the Retain property applied (i.e. have their values preserved during a power-loss situation) is limited to 5000.
- Mapping to **Union** and **Enumerator** data types is **not** supported.
- The mapped Variable is always a map to the entire derived data type.

## 8.5 NA System Variables

The following NA System Variables will be available for the user: These can be accessed within NA project like any other GLOBAL VARIABLE.

System Variable Name	Description	Data Type	Read Only?
_HMI_Brightness	The current backlight brightness. This is a control variable, so setting it to a valid value will cause the current backlight brightness to be changed.	Integer	Read-Write
_HMI_IsScreenSaverActive	Whether the screen saver is active. The ActivateScreenSaver action has to be used to make the screen saver active.	Boolean	Read-Write
_HMI_DateTime	The current local time, read from the internal clock. The SetDateTime action has to be used to adjust the time.	Date	Read-Only
_HMI_CanEjectSDCard	Whether the power for SD memory card is currently supplied. To remove the SD memory card, the EjectSDMemory action has to be used.	Boolean	Read-Only
_HMI_RunSignal	Whether HMI is operating normally. The value of this variable should be toggled periodically while HMI is normal. Even when displaying any menu or a message box, this variable should be toggled.	Boolean	Read-Only
_HMI_IsPageSwitching	Whether the page is being switched. When switching a page, this variable should be turned 'ON'. After the operation is complete, this variable should be turned 'OFF'.	Boolean	Read-Only
_HMI_IsDataInput	Whether any data input object on page is being used to enter data. When any data input object on page has a focus for data input, this variable should be turned 'ON', otherwise it should be turned 'OFF'.	Boolean	Read-Only
_HMI_IsBatteryLow	The HMI internal battery voltage state. When the voltage level is below the warning threshold, this variable should be turned 'ON', otherwise it should be turned 'OFF'.	Boolean	Read-Only
_HMI_CurrentPage	The name of the currently displayed page. This is a control variable – setting it to a valid page will cause the displayed page to be changed. Invalid page names will be ignored.	String	Read-Write
_HMI_Millisecond	Milliseconds, based on the system clock value.	Integer	Read-Only
_HMI_Second	Seconds, based on the system clock value.	Integer	Read-Only
_HMI_Minute	Minutes, based on the system clock value.	Integer	Read-Only
_HMI_Hour	Hours, based on the system clock value.	Integer	Read-Only
_HMI_RAMTotal	The amount of RAM in the NA	Integer	Read-Only
_HMI_RAMInUse	The amount of RAM currently in use in the NA	Integer	Read-Only
_HMI_ManagedRAMInUse	The amount of Managed RAM in use the NA	Integer	Read-Only

### Alarm Variables

System Variable Name	Description	Data Type	Read Only?
_HMI_AlarmCount	Total number of alarms raised since the beginning of the current runtime session	Integer	Read-Only
_HMI_AlarmsRaised	Total number of alarms currently in the raised state	Integer	Read-Only
_HMI_AlarmsNotAck	Total number of alarms currently requiring acknowledgement (whether raised or cleared)	Integer	Read-Only
_HMI_AlarmsRaisedNotAck	Number of currently raised alarms requiring acknowledgement	Integer	Read-Only
_HMI_AlarmsClearedNotAck	Number of alarms cleared but still requiring acknowledgement	Integer	Read-Only

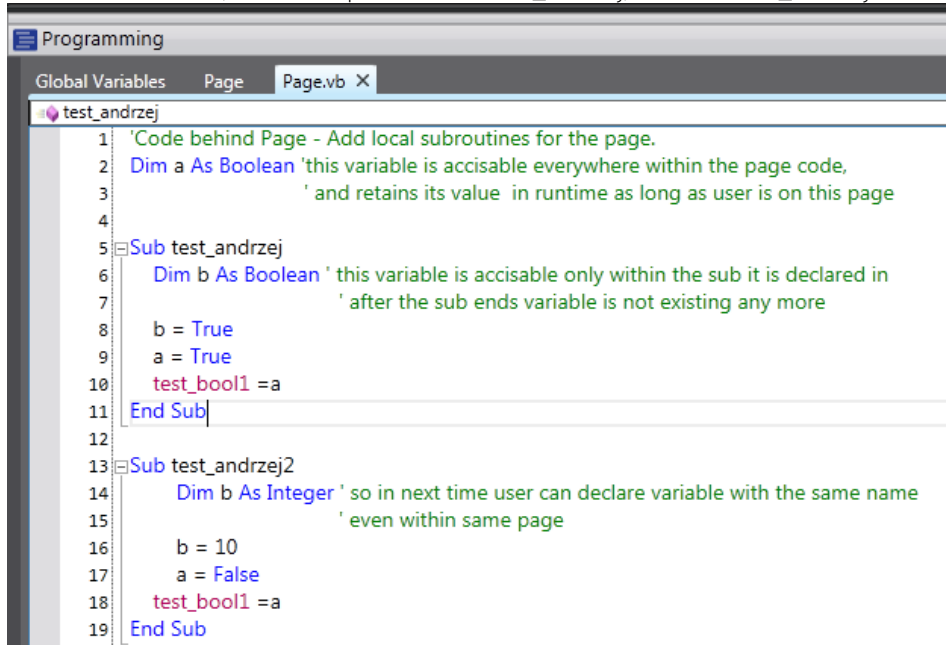
### Security Variables

System Variable Name	Description	Data Type	Read Only?
_HMI_CurrentUserName	Contains the name of the currently logged in user.	String	Read-Only
_HMI_CurrentUserRole	Contains the security role of the currently logged in user (e.g. Administrator)	String	Read-Only

## 8.6 Local NA variables

Further variables can be created and are known as 'local variables'. These only exist in VB and within the scope of the declaration. They can be declared as a page variable (at the top of the Page.vb), or within a sub routine.

Example: two buttons on the screen, one when pressed does test\_andrzej, the other test\_andrzej2:



```

Programming
Global Variables Page Page.vb X
test_andrzej
1 'Code behind Page - Add local subroutines for the page.
2 Dim a As Boolean 'this variable is accisable everywhere within the page code,
3 ' and retains its value in runtime as long as user is on this page
4
5 Sub test_andrzej
6     Dim b As Boolean ' this variable is accisable only within the sub it is declared in
7     ' after the sub ends variable is not existing any more
8     b = True
9     a = True
10    test_bool1 =a
11 End Sub
12
13 Sub test_andrzej2
14     Dim b As Integer ' so in next time user can declare variable with the same name
15     ' even within same page
16     b = 10
17     a = False
18     test_bool1 =a
19 End Sub
  
```

## 8.7 GLOBAL vs LOCAL:

Local variables can be used only within the VB code. They cannot therefore be used for animations, events even within the page they were declared. Page Local variables keep their values at all times during runtime.

Local variables declared within the sub, will keep its value only within the sub they were declared in.

Global variables can be used and referred to anywhere, across the whole project.

The following table shows which variables can be used on specific objects, including the relevant data types:

Category	Object Type	Global Variable - in properties
Buttons	Button	NO
	Toggle Button	YES (Boolean)
	Momentary Button	YES (Boolean)
	Reset Button	YES (Boolean)
	Set Button	YES (Boolean)
Lamps	Bit Lamp	YES - Expression (must equate to a Boolean value)
	Data Lamp	YES - Expression (must equate to a numeric value)
Shapes	Line	NO
	Curve	NO
	Ellipse	NO
	Triangle	NO
	Rectangle	NO
	Polyline	NO
	Polygon	NO
Standard Controls	Label	NO
	Textbox	NO
	Date/Time	YES (DateTime)
	Data Display	YES (All)
	Data Edit	YES (All)

	Image	NO
	Checkbox	YES (Boolean)
	Radio Button	YES ( Short, Integer, Long, Ushort, Uinteger, Ulong, Single, Double, Decimal)
	Slider	YES ( Short, Integer, Long, Ushort, Uinteger, Ulong, Single, Double, Decimal)
	Dropdown	YES ( Short, Integer, Long, Ushort, Uinteger, Ulong, Single, Double, Decimal)
	Listbox	YES (Short, Integer, Long, Ushort, Uinteger, Ulong, Single, Double, Decimal)
Gauges	Linear Gauge	YES ( Short, Integer, Long, Ushort, Uinteger, Ulong, Single, Double, Decimal)
	Rotational Gauge	YES ( Short, Integer, Long, Ushort, Uinteger, Ulong, Single, Double, Decimal)
HMI Controls	Trend Graph	YES (analogue and digital traces)
	User Alarms Viewer	NO
	Recipe Viewer	NO
	Media Player	NO

The Following table shows the list of Actions where the user can use Global Variables:

Category	Actions	Global Variable
User administration	Login	NO
	Logout	NO
Pages	ShowPage	YES (String)
	ClosePage	Yes (String)
	ShowPreviousPage	NO
	ShowSystemMenu	NO
Variables	InvertVariable	YES (Boolean)
	IncreaseVariable	YES(Numeric)
	DecreaseVariable	YES(Numeric)
	SetVariable	YES (Any)
	ResetVariable	YES (Any)
User Alarms	ClearUserAlarmLog	NO
	SaveUserAlarmLogToFile	YES(String)
Data Logging	StartDataLogging	YES(String)
	StopDataLogging	YES(String)
Document Viewers	ShowDocument	YES(String)
	ShowDocument	YES(String)
Dimming	SetBrightness	YES(Integer)
Multi-Language	SetLanguage	YES(String)
External Storage	EjectSDMemory	NO
User operation	EnableTouchScreenInput	Yes (Boolean)
General	CallSubroutine	NO

The Following table shows the list of Actions (Code Functions) where the user can use Global Variables:

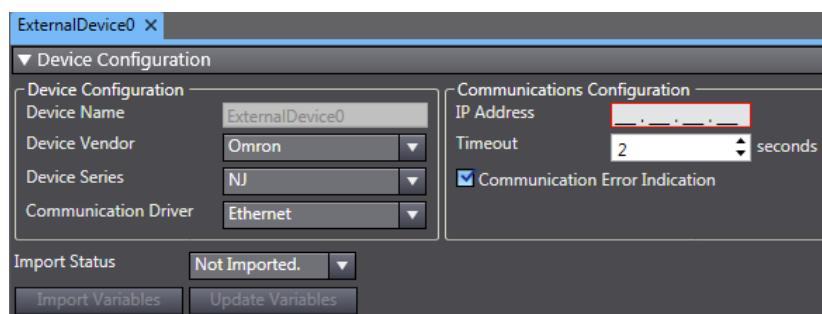
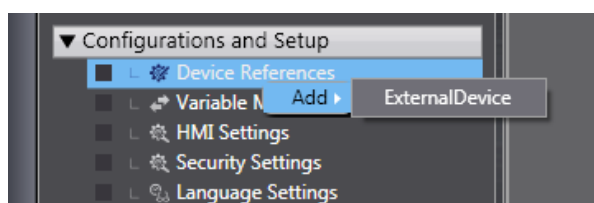
Category	Actions ( Script functions )	Global Variable
User Administration	Login	NO
	Login	YES (String)

	Logout	YES (String)
	GetCurrentUserName	NO
	GetCurrentUserPassword	NO
	GetCurrentUserRole	NO
<b>Pages</b>	ShowPage	YES (String)
	ClosePage	YES (String)
	ShowPreviousPage	NO
	ShowSystemMenu	NO
<b>Variables</b>	InvertVariable	YES (Boolean)
	IncreaseVariable	YES(Numeric)
	DecreaseVariable	YES(Numeric)
	SetVariable	YES (Any)
	ResetVariable	YES (Any)
	EditVariable(value)	YES (String)
<b>User Alarms</b>	AcknowledgeUserAlarm	NO
	AcknowledgeLatestUserAlarm	NO
	AcknowledgeAllUserAlarms	NO
	IsUserAlarmActive	NO?
	IsUserAlarmRaised	NO?
	ClearUserAlarmLog	NO
	SaveUserAlarmLogToFile	YES (String)
	GetAlarmInfo	YES (String)
	ScrollAlarmViewerList	YES (String)
	GetSelectedAlarm	YES (String)
	GetAlarmInfo	NO
<b>Data Logging</b>	StartDataLogging	YES (String)
	StartDataLogging	NO
	StopDataLogging	YES (String)
	StopDataLogging	NO
	ClearDataLogBuffer	YES (String)
	ClearDataLogBuffer	NO
	ExportDataLogBuffer	YES (String)
<b>Recipes</b>	GetRecipeTemplates	NO?
	GetRecipes	YES(String)
	SaveRecipe	YES(String)
	SaveRecipeAs	YES(String)
	AddRecipe	YES(String)
	DeleteRecipe	YES(String)
	ImportRecipes	YES(String)
	ReadRecipeFromController	YES(String)
	WriteRecipeToController	YES(String)
	GetRecipeIngredients	YES(String/Object)
	SetRecipeIngredients	YES(String/Object)
	GetRecipeIngredientValue	YES(String/Object)
	SetRecipeIngredientValue	YES(String/object)
	GetSelectedRecipeTemplate	YES (String)
	GetSelectedRecipe	YES (String)
<b>Movie</b>	OpenMovieFile	YES(String)
	PlayMovie	YES(String)
	StopMovie	YES(String)
	PauseMovie	YES(String)

<b>Trend Graph</b>	ScrollTrendToStart	YES(String)
	ScrollTrendToEnd	YES(String)
	ScrollTrendForward	YES(String)
	ScrollTrendBackward	YES(String)
	ScrollTrendUp	YES(String)
	ScrollTrendDown	YES(String)
	ScrollTrendOverlayForward	YES (String/Integer)
	ScrollTrendOverlayBackward	YES (String/Integer)
	ZoomIntoTrend	YES (String/Integer)
	ZoomOutOfTrend	YES (String/Integer)
	ShowTrendCursor	YES (String/Integer)
	MoveTrendCursor	YES (String/Integer)
	AddOverlayGraph	YES (String)
	RemoveOverlayGraph	YES (String)
	SetTrendDisplayMode	YES (String)
<b>Multi-Language</b>	SetLanguage	YES(String)
	GetLanguage	YES(String)
	GetResourceString	YES(String)
<b>Date &amp; Time</b>	SetDateTime	YES(DateTime)
<b>External storage</b>	EjectSDMemory	NO
	EjectUSBDevice	YES(String)
	GetConnectedUSBDevices	YES (String/Integer)
<b>User operation</b>	EnableTouchScreenInput	NO
<b>Document Viewers</b>	ShowDocument	YES(String)
	ShowDocument	YES(String/Integer)
<b>Others</b>	StartApplication	YES(String/String)

## 8.8 Working with External Devices

As well as working with devices that are within the same Sysmac project, it is possible to connect the NA to 'External Devices'. External simply means not included in the same project. Therefore external devices can be other NJs or CJs. In the future, with the development of multi-vendor drivers, it will be possible to connect to 3<sup>rd</sup> party external devices.



### 8.8.1 Configuring an External Device

In the 'Configuration and setup' area, select Device references and then right click to add an 'ExternalDevice'. The user can then select the type of device. In v1.0, only Omron devices are supported, but this will be expanded in the future.

Selecting the communication driver will reveal the specific settings for that connection. The NA supports EthernetIP, FINS or a special connection to NJ. All are Ethernet based connections.

Once the communication configuration is complete (including IP Address, timeout etc), the variables can be added.

There are several ways to work with variables on external devices, which will be explained in the following sections:

1. Online, importing variables/tags direct from the device
2. Copy and paste from Sysmac Studio
3. Copy and paste to/from Excel
4. Manually enter the tags (valid for FINS connections)

#### Import from real device

To import from a real device that is already connected to the network, click 'Import'. The 'Update' button can be used later to refresh the list of variables known for this device.

Import is available for EthernetIP and direct NJ communications. It is not possible for a FINS based connection.

#### Copy and paste from another Software

Using an instance of Sysmac Studio (separate project), it is possible to copy a list of variables/symbols and then paste directly into the variable list in the External Device.

#### Manually adding tags (FINS)

For FINS devices it is possible to add device variables manually. Right click and select 'Create New'. The AT field must be used to defined the address e.g. D0.

## 8.9 Mapping Variables

All device variables need to be mapped (which creates an NA global variable linked to the device variable) to be used within the NA. This is similar to the IO variables in the NJ.

Select 'Variable Mapping' and expand the device will display a complete list of variables on that device. Variables can be created immediately for all the variables by right clicking on the device name and selecting 'Create Device Variables', or they can be created individually by clicking on each variable and selecting 'Create Device Variable'.

Position	Port	Data Type	Variable	Variable Comment
	▼ Configured Devices			
192.168.	▼ new_Controller_0			
	▶ System Variables			
	User Variables			
	ExternalDevice0			

Once the variables are mapped, clicking on 'Global Variables' in the Data section can see the complete list of NA global variables. Notice that the data types have automatically being mapped to the equivalent NA (.NET) data types. This includes structures, which are automatically linked to the NJ user defined data types.

## 9 VB.NET CODE

### 9.1 Why would I add code?

Adding code allows more power and flexibility behind an action than simple getting and setting of bits. It allows complex logic expressions and mathematical calculations to determine precisely the action to take. Code can also be used to validate values entered and allow the user to correct mistakes. Code may perform calculations to ensure what the user sees is meaningful. Code can be used to initialise values when pages are opened/closed.

#### 9.1.1 Why VB.Net?

The NA is running Windows Embedded Compact 7 (WEC7) and therefore is using the .NET Compact Framework. Using .NET enables the NA to benefit from a huge amount of flexibility and standard functionality that otherwise would take years to develop.

BASIC was developed back in 1964! **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. As the name suggests, it's easy to use. Since its development there have been many versions and changes. Microsoft Developed Visual BASIC (VB) to allow users to easily create user interfaces for Windows. In the last 10 years Microsoft has developed VB to be used under its .NET framework. Far from its 'Beginners' or Hobby beginnings, VB.NET is now widely used in a commercial setting

VB.NET is now an established language that is widely used. Novice developers will be quickly become familiar with the syntax which is easy to use and forgiving when compared with C#, C++ or Java.

Unlike Visual BASIC from many years ago, VB.NET is a compiled language (not interpreted) which means the quality and speed is comparable to any other .NET language.

### 9.2 Where can I use code?

In the NA Series there are several main areas that code can be used:

- Global Subroutines
- Page Code
- IAG Code

Global subroutines are useful to contain all the common functionality for the whole project. They can be linked to global events such as scheduled events that run at a particular time of day (perhaps export a file to the USB device or something).

Each Page has a code behind. Here the user can program functionality specific to that page. All actions of page objects that call subroutines would result in a call to a function in the page code. It is possible to call a subroutine on another page, for example a routine on the background page.

IAG Code will be covered in another chapter (Creating IAGs), but IAGs can contain code and expose methods to allow code to be encapsulated within a reusable object (IAG).

### 9.3 Subroutines and Functions

In VB.Net there are two ways of defining methods:

- A subroutine can take parameters but does not return a value. Subroutines are used to link to objects on a page, for example, a button could call a subroutine when it is pressed.
- A function also takes parameters, but can return a value. Functions are normally used to perform an operation, such as converting data and returning in a new format. Functions cannot be coupled up to page events, but can be called from subroutines.

Subroutines and functions should be used to break the code down into smaller 'bite sized' chunks. You can repeatedly call a function or subroutine from more than one place allowing you to re-use something rather than duplicating the same code many times in the project.

The syntax for a new subroutine is:

```
Sub mysubname(myParameter As Integer)
    'add code here
End Sub
```

The syntax for a new function is:

```
Function myfuncname(myParameter As Integer) As Integer
    'add code here
    myfuncname = myParameter * 2 'the result
End Function
```

A subroutine or function local to the current page can be called by just its name or using:

```
me.subroutineName
```

Functions and subroutines that are global can be called using the group name or page where they are located. A Subroutine defined in a global group can called in this way:

```
SubroutineGroup.mySubroutine
```

Background page subroutines can also be called the following way (this cannot be used to call other pages, only background page methods):

```
Pagename.mySubroutine
```

The syntax for calling subroutines and functions when you are using return values is different:

```
returnedValue = me.myFunction(myParameter, myOtherParameter) 'using brackets
me.mySubroutine myParameter, myOtherParameter 'no return so no brackets
```

## 9.4 VB Variables

VB variables can be defined within a subroutine or function, these are perfect for holding temporary values needed to perform the operation and reduce unnecessary global variables.

To define a VB variable, the following syntax can be used:

```
Dim myVar as Integer
```

Any VB type can be used for these VB Variables. The following table defines the usable data types:

Visual Basic type	Size	Value range
<a href="#">Boolean</a>		<b>True or False</b>
<a href="#">Byte</a>	1 byte	0 through 255 (unsigned)
<a href="#">Char</a> (single character)	2 bytes	0 through 65535 (unsigned)
<a href="#">Date</a>	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
<a href="#">Decimal</a>	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) <sup>†</sup> with no decimal point; 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal;  smallest nonzero number is +/-0.00000000000000000000000000000001 (+/-1E-28) <sup>†</sup>
<a href="#">Double</a> (double-precision floating-point)	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324 <sup>†</sup> for negative values;  4.94065645841246544E-324 through 1.79769313486231570E+308 <sup>†</sup> for



		positive values
<a href="#">Integer</a>	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
<a href="#">Long</a> (long integer)	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807 (9.2...E+18 <sup>†</sup> ) (signed)
<a href="#">SByte</a>	1 byte	-128 through 127 (signed)
<a href="#">Short</a> (short integer)	2 bytes	-32,768 through 32,767 (signed)
<a href="#">Single</a> (single-precision floating-point)	4 bytes	-3.4028235E+38 through -1.401298E-45 <sup>†</sup> for negative values; 1.401298E-45 through 3.4028235E+38 <sup>†</sup> for positive values
<a href="#">String</a> (variable-length)	-	0 to approximately 2 billion Unicode characters
<a href="#">UInteger</a>	4 bytes	0 through 4,294,967,295 (unsigned)
<a href="#">ULong</a>	8 bytes	0 through 18,446,744,073,709,551,615 (1.8...E+19 <sup>†</sup> ) (unsigned)
<a href="#">UShort</a>	2 bytes	0 through 65,535 (unsigned)

To define a VB variable as an array, the following syntax would define an array of 10 elements:

```
Dim myArray(10) as Integer
```

To access the individual array elements, the user would type: `x = myArray(1)`

It is possible to define VB Variables outside of a subroutine or a function. For example, a page VB variable can be defined by declaring a Variable at the top of the page.vb. Such a variable can then be accessed from anywhere in the page code. Values in such variables are maintained whilst the page is on display. Equally it is possible to define a global VB variable by declaring it in the global subroutine area.

*Note: whilst technically global & page VB variables are possible, they are not recommended. It is usually better to consider a global variable defined within Sysmac Studio.*

For further information on using variables go to the chapter on variables.

## 9.5 Private or Public

When defining a variable, subroutine or function in VB, it is possible to make it private or public.

Methods defined as 'Private' can only be used from within the same page or group so is protected from any unexpected changes from elsewhere in the application, a value of a private variable could be shared through a public function.

Conversely, Public methods can be used from everywhere in the program by using the page or group.

## 9.6 Designing code

The important thing to design a program well is to break it down into smaller 'bite sized' pieces of functionality. This makes the development and debugging process.

The programmer's basic rule is 'a function should do one thing, and one thing well'. Therefore when designing the structure of a program, the functionality should be divided down into small sections and subroutines used to contain that functionality.

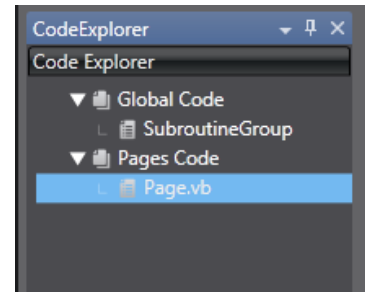
Also by using subroutines for common code, when there is a problem it is easy to solve it in one place, rather than having to change everywhere that that functionality is programmed.

Note: it is one of the most common mistakes in programming to not correctly plan the structure of the functions that are needed. The result is routines with huge numbers of IF ELSE statements that are trying to handle far too many things in one place, or code repeatedly copied throughout the application. Take the time and ensure a better program design.

## 9.7 Code Explorer

There are several ways to view the code within the project. Right clicking on any page and selecting 'View Code' will display the code for that specific page. Equally on the global subroutines area in the Solution Explorer will open the global code for that section.

In addition to these methods to view code, there is a Code Explorer (like Solution Explorer and Page Explorer) that makes it easy to navigate through the code included in the project. If it is not displayed, select 'View | Code Explorer'.



## 9.8 The Code Editor

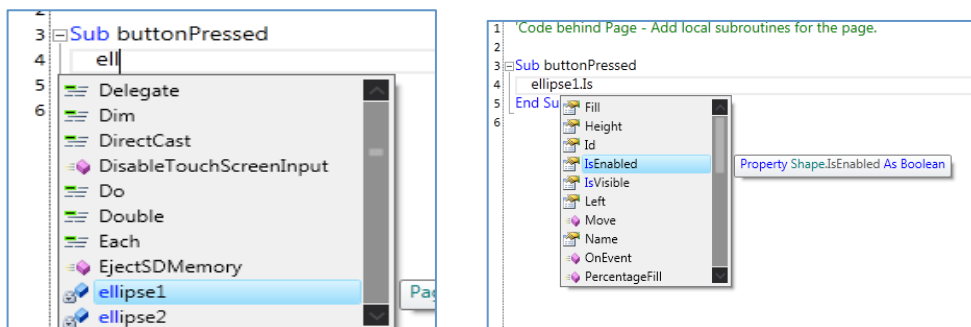
The code editor is started from Solution Explorer or Code Explorer. The code editor behaves the same for page or global code. Subroutines and functions can be expanded or collapsed using the + or – button next to the code block.

The Code Editor uses colour coding to help you with your syntax (default colours can be changed in the options):

- Blue - Reserved Words for VB e.g. Sub, Dim, Call etc
- Red - Known variable name
- Green - Comment preceded by '
- Green underline - Warning
- Red underline - Syntax error

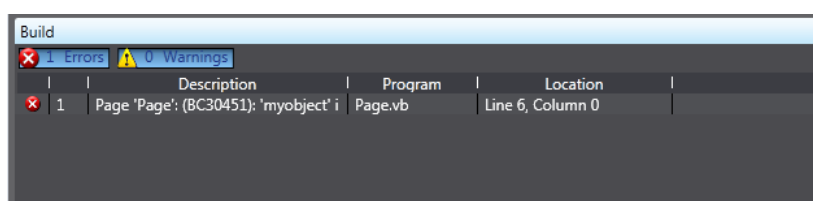
The code editor will offer suggestions as you are typing; this reduces mistakes and the need for using a manual.

Autocomplete will pop up automatically when in the right context e.g. 'me.' Or 'mediaplayer.'. You can also get auto complete to pop up at any point in Sysmac Studio by pressing <Ctrl> + <Enter> to display all possible options.



## 9.9 Building the Code

To build the project, and associated code, click on: Project | HMI Build. The output from the build process will appear at the bottom of Sysmac Studio. Double clicking on any errors or warnings will jump directly to the code that needs correcting:



## 9.10 NA VB Methods

The following NA functions can be called from VB:

Category	Action
User Administration	Login
	Login
	Logout
	GetCurrentUserName
	GetCurrentUserPassword
	GetCurrentUserRole
Pages	ShowPage
	ClosePage
	ShowPreviousPage
	ShowSystemMenu
Variables	InvertVariable
	IncreaseVariable
	DecreaseVariable
	SetVariable
	ResetVariable
	EditVariable
User Alarms	AcknowledgeUserAlarm
	AcknowledgeLatestUserAlarm
	AcknowledgeAllUserAlarms
	IsUserAlarmActive
	IsUserAlarmRaised
	ClearUserAlarmLog
	SaveUserAlarmLogToFile
	GetAlarmInfo
	ScrollAlarmViewerList
	GetSelected
Data Logging	StartDataLogging
	StartDataLogging
	StopDataLogging
	StopDataLogging
	ClearDataLogBuffer
	ClearDataLogBuffer
	ExportDataLogBuffer
Recipes	GetRecipeTemplateNames
	GetRecipeNames
	GetRecipeIngredientNames
	SaveRecipe
	SaveRecipeAs
	SaveAllRecipes
	AddRecipe
	DeleteRecipe
	ImportRecipes
	ReadRecipeFromController
	WriteRecipeToController
	GetRecipeIngredients
	SetRecipeIngredients
	GetRecipeIngredientValue
	SetRecipeIngredientValue
	GetSelected
Movie	OpenMovieFile
	PlayMovie
	StopMovie
	PauseMovie
Trend Graph	ScrollTrendToStart
	ScrollTrendToEnd
	ScrollTrendForward
	ScrollTrendBackward
	ScrollTrendUp
	ScrollTrendDown
	ScrollTrendOverlayForward

	ScrollTrendOverlayBackward
	ZoomIntoTrend
	ZoomOutofTrend
	ShowTrendCursor
	MoveTrendCursor
	AddOverlayGraph
	RemoveOverlayGraph
	SetTrendDisplayMode
Dimming	SetBrightness
Multi-Language	SetLanguage
	GetLanguage
	GetResourceString
Date & Time	SetDateTime
ScreenSaver	ActivateScreenSaver
	DeactivateScreenSaver
External storage	EjectSDMemory
	EjectUSBDevice
	GetConnectedUSBDevices
User operation	EnableTouchScreenInput
Document Viewers	ShowDocument
	ShowDocument

### 9.11 Using VB to Access Object Properties

Using code it is possible to access some properties of objects that are on the screen. In v1.0 this is limited mostly to appearance properties such as position or size. In the future this will be expanded to support further access from code during the runtime. The VB.Net names of properties sometimes differ slightly from the property table shown in Sysmac Studio (which is translated into many different languages). The following table shows what can be accessed, and the VB property names:

Object	Property Grid Name	VB.NET Name
<b>Checkbox</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Date/Time</b>	IsVisible	IsVisible
	<del>IsEnabled</del>	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Dropdown</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Listbox</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Image</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Label</b>	> Text (Default)	Text
	IsVisible	IsVisible

	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Radio Button</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
<b>Slider</b>	Height	Height
	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
<b>Textbox</b>	Width	Width
	Height	Height
	> Text (Default)	Text
	IsVisible	IsVisible
	IsEnabled	IsEnabled
<b>Data Display</b>	Left	Left
	Top	Top
	Width	Width
	Height	Height
	IsVisible	IsVisible
<b>Data Edit</b>	Left	Left
	Top	Top
	Width	Width
	Height	Height
	IsVisible	IsVisible
<b>Group</b>	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
<b>IAG</b>	Height	Height
	Left	Left
	Top	Top
	Width	Width
<b>Line</b>	Height	Height
	LineColor	Stroke
	LineThickness	StrokeThickness
	IsVisible	IsVisible
	n/a	IsEnabled
	Left	Left
	Top	Top
	Width	Width
<b>Ellipse</b>	Height	Height
	FillColor	Fill
	LineColor	Stroke
	LineThickness	StrokeThickness
	IsVisible	IsVisible
	n/a	IsEnabled
	Left	Left
	Top	Top
<b>Triangle</b>	Width	Width
	Height	Height
	FillColor	Fill
	LineColor	Stroke
	LineThickness	StrokeThickness
	IsVisible	IsVisible

	<i>n/a</i>	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Rectangle</b>	FillColor	Fill
	LineColor	Stroke
	LineThickness	StrokeThickness
	IsVisible	IsVisible
	<i>n/a</i>	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Polyline</b>	LineColor	Stroke
	LineThickness	StrokeThickness
	IsVisible	IsVisible
	<i>n/a</i>	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Polygon</b>	FillColor	Fill
	LineColor	Stroke
	LineThickness	StrokeThickness
	IsVisible	IsVisible
	<i>n/a</i>	IsEnabled
	Left	Left
	Top	Top
	Width	Width
<b>Curve</b>	Height	Height
	LineColor	Stroke
	LineThickness	StrokeThickness
	IsVisible	IsVisible
	<i>n/a</i>	IsEnabled
	Left	Left
	Top	Top
	Width	Width
<b>Bit Lamp</b>	Height	Height
	IsVisible	IsVisible
	Left	Left
	Top	Top
<b>Data Lamp</b>	Width	Width
	Height	Height
	IsVisible	IsVisible
	Left	Left
<b>Trend Graph</b>	Top	Top
	Width	Width
	Height	Height
	IsVisible	IsVisible
	IsEnabled	IsEnabled
<b>User Alarm Viewer</b>	Left	Left
	Top	Top
	Width	Width
	Height	Height
	IsVisible	IsVisible
<b>Recipe Viewer</b>	IsEnabled	IsEnabled
	Left	Left
	IsVisible	IsVisible

	Top	Top
	Width	Width
	Height	Height
<b>Media Player</b>	IsVisible	IsVisible
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Linear Gauge</b>	IsVisible	IsVisible
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Rotational Gauge</b>	IsVisible	IsVisible
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Button</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Momentary Button</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Reset Button</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Set Button</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height
<b>Toggle Button</b>	IsVisible	IsVisible
	IsEnabled	IsEnabled
	Left	Left
	Top	Top
	Width	Width
	Height	Height

## 9.12 Standard VB Functions that can be used

VB.NET has a huge library of functionality that is useable inside an NA project. It is impossible to define here everything that is supported, and the best resource to help with programming is probably the Internet. Online there are many examples and documentation that explain what is possible in VB.NET.

One of the limitations of the NA is that it is not possible to 'import' other assemblies. Therefore sometimes to access the functionality the full path to the methods must be defined. For example:

```
System.IO.File.Exists(PathName)
```

The NA does however import some assemblies automatically which are designed to make programming easier. The following are imported:

- System
- System.Net
- System.Net.Sockets
- System.ComponentModel
- System.Diagnostics
- System.IO
- System.IO.Compression
- Sysmac.IO.Ports
- System.Text.RegularExpressions
- System.Collections
- System.Text
- System.Globalization
- System.Resources

For more information on what is available in the .NET Compact Framework, see the following web link:  
[http://msdn.microsoft.com/en-us/library/cc838194\(v=vs.95\).aspx](http://msdn.microsoft.com/en-us/library/cc838194(v=vs.95).aspx)



## 10 ALARMS

Like any Machine Interface, the NA Series has an alarm management function that can alert the operator to problems, or matters that require their attention.

Alarms are configured in Sysmac Studio.

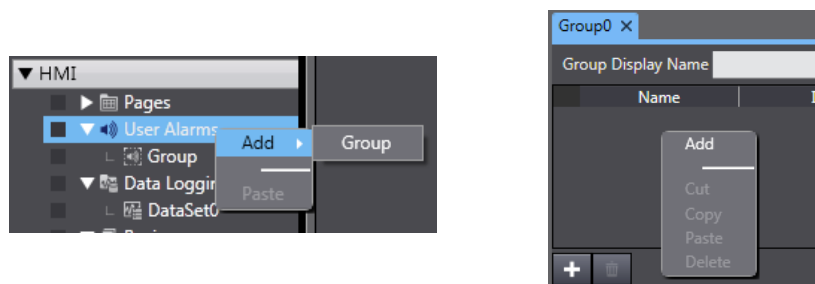
### 10.1 Creating Alarms

Alarms are created in groups, this enables the designer to put group alarms either by type, severity or based on the area of the machine. This gives structure to the alarm definitions, makes the project easier to understand, and is shown to the user at runtime in the Alarm Viewer.

To add the Alarm group, go to "Programming – User Alarms – (right click)-Add". Groups can be nested.

The user can specify a Group Display Name (which can be translated in resources). The Group Display Name will be shown in the Alarm Viewer at runtime.

When an Alarm Group is selected, the alarms page is shown, To add a new alarm to that group, right click – add, or use "+" or use "insert" key to add alarms to the group.



Alarms can be copied and pasted into excel, which makes large edits quicker and easier. Copy one alarm from Sysmac Studio into Excel to see the format, manipulate the data and then paste back into Sysmac Studio.

Each Alarm has the following properties:

1. **Name** (string, 128 characters max): The name of the User Alarm. The name is unique within a group.
2. **Id** This cannot be changed by the user, but is used to identify the alarm in the system
3. **Alarm Code.** An optional number, 0 to 60000. The 'Event Code' field is an optional numeric field. This field can be empty or specify any number. In future versions, when the Controller User Event functionality is integrated with the HMI alarm functionality, the Event Code should be obtained automatically from the Controller User Event information (when integrating User Events into the HMI alarm system). It should also be possible to define HMI alarms that have the same Event Code. For example, where two different Controllers use the same Event Code and are both being displayed (integrated) in an HMI device.
4. **Expression** (string): A logical condition (or Boolean variable) that when true the Alarm will be raised. Standard Intellisense functionality will be supported.
5. **Priority** (string): This can have the following values:
  - a. Level 1 Highest Alarm Priority
  - b. Level 2
  - c. Level 3
  - d. Level 4
  - e. Level 5
  - f. Level 6
  - g. Level 7
  - h. Level 8 Lowest Alarm Priority (apart from "Information Level")
  - i. Information Level
6. **Message** (string, 128 characters max): The event message, which appears in the runtime status and log viewer objects. (Max 128 characters)

7. **Details** (string, 2048 characters max): The full event description, which can optionally be displayed in the runtime status and log viewer objects.
8. **Display popup message**: If set, this causes a popup alarm status to appear in the runtime when an alarm occurs and requires acknowledgement.
9. **Requires acknowledgement**: This is automatically set for alarms that display the popup message. If not set, then the alarm will automatically be treated as if it is acknowledged as soon as it occurs.

## 10.2 Alarm Events

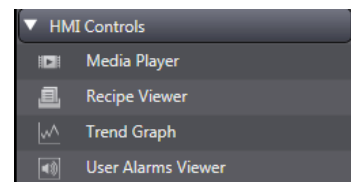
Each alarm can be configured to have actions on the following three events:

- UserAlarmRaised
- UserAlarmCleared
- UserAlarmAcknowledged

These events can trigger various actions/events from the default actions list (See Events & Actions for more information on actions).

## 10.3 Using the Alarm Viewer

The NA Series has a special Alarm Object that can be placed on the page to display the status information about alarms – both live and historical information. To use the Alarm Viewer, expand the 'HMI Controls' section of the toolbox and drag&drop the 'User Alarms Viewer' onto the page.



There are two object display modes:

- **Active Display mode** – 'Active' User Alarms only are displayed
- **Historical Display mode** – The User Alarm Log is displayed showing the history of all User Alarm events

The columns that are shown in the Alarm Viewer are configurable using the properties window of the object. The following are the possible columns:

- Date & Time
- Name
- Message
- Priority (Level)
- Status
- Group
- Event Code
- Details
- Logged-In User

There is also a property for the title of these columns, which appears in the resources; therefore it is possible to translate the columns titles.

The colours for each of the states of an alarm can be customised to match a specification.

## 10.4 Alarms in Runtime

The runtime engine includes an Alarm Handler process that continuously monitors the User Alarms that are configured for the project. The Alarm Handler detects state changes in User Alarms and responds to the state changes by producing Operator notifications and by creating data entries into a User Alarm Log MRAM.

Each time a User Alarm is either raised, acknowledged or cleared is captured in runtime by the Alarm Handler. On each event the Alarm Handler creates a data entry into the User Alarm Log. The sum of the data entries in the User Alarm Log constitutes the User Alarm History. All logging occurs in the MRAM, which acts as a circular buffer with at least 2048 log entries. When the maximum size is exceeded then the oldest log entry will be overwritten by new data. System Variables can be used to monitor the number of alarms in the system, and methods are provided to save the alarm log (see below).

Each User Alarm can exist in one of five states:

1. Acquiescent
2. Raised – Requiring acknowledgement, but not acknowledged
3. Raised – Acknowledged or not requiring acknowledgement
4. Cleared – Requiring acknowledgement, but not acknowledged
5. Cleared - Acknowledged or not requiring acknowledgement

**Acquiescent** - The User Alarm has never been raised in the current runtime session

**Raised** – Raised state for a User Alarm is achieved when the expression associated with the User Alarm resolves to True

**Cleared** – Cleared state for a User Alarm is achieved when the expression associated with the User Alarm resolves to False (having previously been True)

**Acknowledged \ Not Acknowledged** – Acknowledgment is a process applied by the runtime user confirming their recognition that a User Alarm has been raised. Whether User Acknowledgement is required or not can optionally be applied for each individual User Alarm.

Time	Name	Priority	Message	Status	Group	Event Code	Details	Logged-In User
23/07/2013 17:3	NewUserAlarm0	Level4	bit1	Alarm Raised(Unacknowledged)	AlarmGroup0	1009		
23/07/2013 17:3	NewUserAlarm2	Level4	bit2	Alarm Raised	AlarmGroup1	1009		
23/07/2013 17:3	NewUserAlarm1	Level3	bit3	Alarm Raised	AlarmGroup0	1009		

Time	Name	Priority	Message	Status	Group	Event Code	Details	Logged-In User
23/07/2013 17:4	NewUserAlarm0	Level4	bit1	Alarm Cleared(Unacknowledged)	AlarmGroup0	1009		

## 10.5 System Variables

The following “alarms” system variables are available:

System Variable Name	Description	Data Type	Read Only?
_HMI_AlarmCount	Total number of alarms raised since the beginning of the current runtime session	Integer	Read-Only
_HMI_AlarmsRaised	Total number of alarms currently in the raised state	Integer	Read-Only
_HMI_AlarmsNotAck	Total number of alarms currently requiring acknowledgement (whether raised or cleared)	Integer	Read-Only
_HMI_AlarmsRaisedNotAck	Number of currently raised alarms requiring acknowledgement	Integer	Read-Only
_HMI_AlarmsClearedNotAck	Number of alarms cleared but still requiring acknowledgement	Integer	Read-Only

## 10.6 Alarms in VB

The following methods are supported:

- AcknowledgeHMIUserAlarm
- AcknowledgeHMIUserAlarmsAll
- HMIClearUserAlarmLog
- AcknowledgeHMILatestUserAlarm
- IsHMIUserAlarmActive
- IsHMIUserAlarmRaised
- SaveUserAlarmLogToFile (copies log from MRAM to specified file)
- GetAlarmInfo (allows users to define their own alarm viewer functionality).
- ScrollAlarmViewerList (allows users to define their own UI for scrolling the Alarm Viewer list).

AcknowledgeUserAlarm	1. AlarmID	1. Alarm	Acknowledges the specified user alarm.
AcknowledgeLatestUserAlarm	<None>	<None>	Acknowledges the latest user alarm.
AcknowledgeAllUserAlarms	<None>	<None>	Acknowledge all user alarms.
IsUserAlarmActive	1. AlarmID	1. Alarm <b>Return Type:</b> Boolean	Returns TRUE if the specified alarm is active, otherwise FALSE will be return. Active states are Raised & Acknowledged, Raised & Unacknowledged and Cleared & Unacknowledged.
IsUserAlarmRaised	1. AlarmID	1. Alarm <b>Return Type:</b> Boolean	Returns TRUE if the specified alarm is raised or FALSE if it is cleared.
ClearUserAlarmLog	<None>	<None>	Removes all current alarm data stored in MRAM (the data will be lost and is unrecoverable).
SaveUserAlarmLogToFile	1. File Name	1. String 1. String Variable	Copies the current user alarm data from MRAM and saves it into the specified file.
GetAlarmInfo	1. ID 2. Name 3. Group 3. Code 4. Priority 5. Message 6. Details 7. Date & Time 8. Status 9. User	1. Integer 1. Integer Variable 2. String Variable (ByRef) 3. String Variable (ByRef) 4. String Variable (ByRef) 5. String Variable (ByRef) 6. String Variable (ByRef) 7. String Variable (ByRef) 8. String Variable (ByRef) 9. String Variable (ByRef)	Gets the alarm information (alarm name, group name, code, priority, message and details) associated with the specified Active Alarm ID.
ScrollAlarmViewerList	1. Page Name 2. Alarm Viewer Object Name 3. Scroll Direction 4. Scroll Amount	1. String 1. String Variable 2. String 2. String Variable 3. Integer 3. Integer Variable 4. Integer 4. Integer Variable	Scrolls the specified alarm viewer object up or down by the amount defined in the 'Scroll Amount' parameter. When the value 1 is passed to the 'Direction' parameter then alarm viewer object will be scrolled upwards (to scroll downwards, the value 2 must be passed). The 'Scroll Amount' parameter defines how many will be scrolled each time the function is called.
GetSelectedAlarm	1. Page Name 2. Alarm Viewer Object Name <b>Returns:</b> AlarmID	1. String 2. String Return Type: String	Gets the selected alarm ID from the specified viewer.
GetAlarmInfo	1. AlarmID 2. Name 3. Group 4. Alarm Code 5. Level 6. Message 7. Details	1. String 2. String (ByRef) 3. String (ByRef) 4. String (ByRef) 5. String (ByRef) 6. String (ByRef) 7. String (ByRef)	Returns the Alarm information for the specified Alarm ID.

**Note:** the AlarmViewer also has a property called 'SelectedAlarm' (ID) which enables VB to find the currently selected alarm for some of the functions above.

## 11 RECIPES

### 11.1 Overview of Recipes

A recipe is a quick way to download a set of parameters for the batch of the products. Templates are configured with sets of ingredients. Each ingredient is linked to a variable and then instances of the recipe can be defined. Selecting a specific instance will load these values into the variables and download to the controller where relevant.

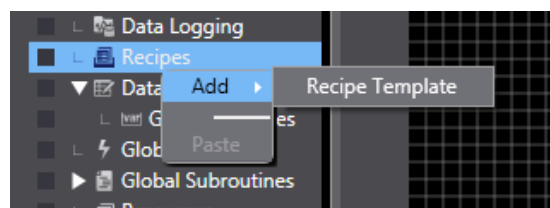
NA Recipe features:

- the ability to load a pre-configured recipe from a database csv file
- the ability to manipulate the ingredient values of a recipe once loaded
- the ability to save a modified recipe back to the database
- the ability to create new recipe in the recipe database
- the ability to transfer a recipe to the variables
- the ability to upload a recipe from the variables

The runtime HMI device uses the recipe data that was pre-configured in the IDE and allows further updates to be made.

### 11.2 Recipe Templates & Instances

To create new Recipe template go to Programming -> Recipe (right click), Add | Recipe.



This will create a new blank recipe. Recipes consist of 'templates', 'ingredients' and 'instances'.

**Templates:** are used to configure the recipe ingredients.

**Ingredients:** are linked to variables, can have default values as well as maximum and minimum values.

**Instances/Recipes:** are the sets of distinct values of ingredients as part of the overall template.

Double clicking on a recipe will open the recipe editor:

Ingredient Name	Variable	Default Value	Min Value	Max Value	Visible	Editable
Rate	nRate	10	1	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

When viewing the template of a recipe it is possible to add ingredients (use the '+'). Each ingredient will contain the following attributes:

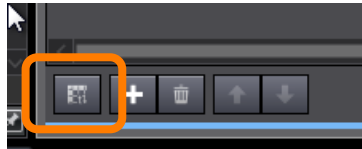
- **Name** - name of the ingredient.
- **Variable** - name of the GLOBAL tag that this ingredient is associated with in the Controller.
- **Default Value** – the default value for this ingredient.
- **Minimum Value** – the minimum value that this ingredient can be set to during runtime operation.
- **Maximum Value** – the maximum value that this ingredient can be set to during runtime operation.
- **Visibility** – determines if this ingredient is displayed to the user during runtime operation.
- **Editable** – determines if this ingredient value can be modified during runtime operation.

#### 11.2.1 Workflow

When working with recipes, it is suggested to start with creating global variables that will be used with the recipes. Then in the recipe template view create the template:

Ingredient Name	Variable	Default Value	Min Value	Max Value	Visible	Editable
sugar	test_int	55	10	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
salt	test_real	15	11	20	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
quality	test_bool				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
description	test_string				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Once ingredients are added to the template, the user can declare instances of the recipes and switch between the instance and template view.



The Recipe Instance data can be created using the Sysmac Studio (i.e. a Table control that contains a column for Recipe name, plus a column for each ingredient):

HMI					
Recipe0 X Global Variables					
Recipe Name	sugar	salt	quality	description	
GoodProduct	20	11	true	good soup	
AverageProduct	55	20	false	average soup	
	55	15	true		

Show Ingredients  
Add  
Delete  
Cut  
Copy  
Paste

Each recipe template will be associated with a database file (.CSV), which will contain the full list of recipe instances and recipe data. This information is pre-configured in the IDE along with name of the database, its format. In runtime, this database is simply accessed (read from or written to) when any of the recipe operations are carried out by the user.

The user will be able to “export” csv file from the NA to USB stick (using system menu) and edit the recipe data, adding complex instances of the templates in excel rather than in runtime on the HMI.

	A	B	C	D	E
1	<b>Name</b>	<b>Width</b>	<b>Height</b>	<b>Depth</b>	<b>Thickness</b>
2	Box_A	300	200	100	5
3	Box_B	500	300	200	10
4	Box_C	1000	1000	1000	20

It is possible to copy & paste recipe data to and from excel. This makes large scale modifications easier and quicker.

In order to restrict the size of the recipe database files, each database can hold up to 1000 recipe instances.

### 11.3 Recipe Viewer

The NA Series includes a dedicated ‘Recipe Viewer’ object that is able to be used to easily display recipe templates & instances during the runtime. It allows the user to interact with the system recipes in runtime.

Templates	Name	Value
Recipe		

There are three available areas within the Recipe Viewer object (all of which can be configured in the property area of Sysmac Studio):

**Templates:** Shows all appropriate recipe templates (recipe types), configured in Sysmac Studio. The user can select the required recipe template by touching on it in the list.

**Recipes:** Shows all recipe instances configured for the item selected in the 'Recipe Template List'. The user can select the required recipe instance by touching on it in the list.

**Recipe Data:** Shows a list of all ingredients associated with the item selected in the 'Recipe Instance List'. This includes the name of the ingredient and the value for that ingredient. The user can select a specific ingredient and edit its value using a popup keyboard.

### 11.3.1 Customising the appearance

It is possible to customise the appearance of the Recipe Viewer in order to work well on a touch screen (depending on the screen size, maybe the user would choose different font sizes etc). These can all be configured using the properties area of Sysmac Studio:

HorizontalSplitterPosition1	0
▼ Font	Segoe UI,12,Normal
Family	Segoe UI ▼
Size	20
Style	Normal ▼
▼ HeaderFont	Segoe UI,12,Normal
Family	▼
Size	20
Style	Normal ▼
IsVisible	<input checked="" type="checkbox"/>
▼ Misc	

## 11.4 Working with recipes in VB

Besides using the recipe viewer, it is also possible to interact fully with the recipe system using code. The following functions are available for the user to use within VB:

GetRecipeTemplateName	Returns a list of all the recipe template names defined for the current HMI application.
GetRecipeNames	Returns a list of all the recipe instance names for the specified recipe template.
GetRecipeIngredientNames	Returns a list of all the recipe ingredient names for the specified recipe template.
SaveRecipe	Saves the specified recipe instance (in memory) to the specified recipe database file. The database filename is determined by the recipe template name passed.  <b>NOTE:</b> only the specified recipe instance will be saved into the recipe database file (not all instances held in memory).

SaveRecipeAs	Copies the specified recipe instance (in memory) and appends it to the end of the recipe database file, using the new recipe instance name specified. The new recipe instance name will be validated to confirm it is unique. An error will be raised if the recipe instance name is not unique and a new recipe instance will not be created.  <b>NOTE:</b> <i>only the new recipe instance will be written to the database file (not all instances held in memory).</i>
SaveAllRecipes	Saves all recipe instances (in memory) to the specified recipe database file. The database filename is determined by the recipe template name passed.
AddRecipe	Appends a new recipe instance to the end of the recipe instance list (in memory) using the specified instance name. The ingredient values will be set to their default values (originally defined when the recipe template was created). The new recipe instance name will be validated to confirm it is unique. An error will be raised if the recipe instance name is not unique and a new recipe instance will not be added.  <b>NOTE:</b> <i>The new recipe instance will not be saved to the recipe database file until the 'SaveRecipe' function is called.</i>
DeleteRecipe	Deletes the specified recipe instance from the specified recipe database file.
ImportRecipes	Imports all of the recipe instances, held in the specified file, into the list of recipe instances in memory. Validation on the ingredients names will be performed to ensure they match - if they do not match then no data will be imported and an error will be raised.  <b>NOTE:</b> <i>The new recipe instances will not be saved to the recipe database file until the 'SaveAllRecipes' function is called. Alternatively, 'SaveRecipe' could be called for each new recipe instance but could be very cumbersome.</i>
ReadRecipeFromController	Reads the recipe ingredient values, from the controller variables that are associated with the specified recipe template, and transfers them to the specified recipe instance in memory.
WriteRecipeToController	Writes the recipe ingredient values, from the specified recipe instance (in memory), to the controller variables associated with the specified recipe template.
GetRecipeIngredients	Gets a list of the ingredient names, and their associated values, from memory, for the specified recipe template, and writes them into the array variables passed in.
SetRecipeIngredients	Using the ingredient names (and associated values) passed in, this function sets the specified recipe instance (in memory), for the specified recipe template, to those values.
GetRecipeIngredientValue	Gets the value associated with the specified ingredient name, from memory, for the specified recipe template.
SetRecipeIngredientValue	Sets the value associated with the specified ingredient name (in memory) for the specified recipe template.
GetSelectedRecipeTemplate	Returns the currently selected recipe template from the specified recipe viewer
GetSelectedRecipeInstance	Returns the currently selected recipe instance from the specified recipe viewer

**Note:** None of the functions listed above can be called directly from the Action Editor. They can only be used within VB.net code. In addition there are several object methods that can be used to get the selected recipe template/instance e.g. `RecipeViewer1.SelectedTemplate`

## 11.5 Recipes & Security

Operational security can be applied to the recipe viewer as a whole, access and Visibility control can be configured.



## 11.6 Multilanguage support

There is currently no support for localisation of recipe templates or instances. This will be an area for future improvement.

## 12 DATA LOGGING

### 12.1 Basics of data logging

The NA Series includes a Data Logging feature that allows variables to be stored in a data log, and later viewed on the screen as a trend. A dataset is required in order to log data, a dataset contains a list of variables as well as the configuration of the trigger to log (interval or on a condition).

To start data logging for a dataset, there are two options, either to automatically start logging when starting up, or a manual start-up using an action or code.

It is possible to configure for each data set where it will be stored, either on the SD card, or on a mass storage device connected via USB.

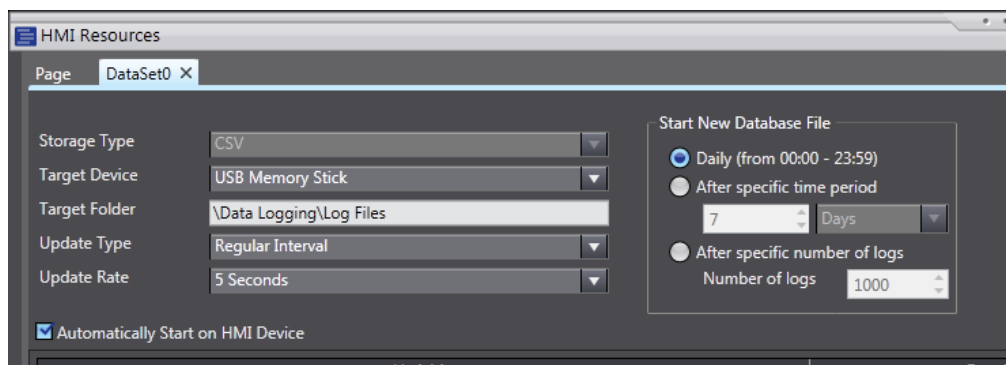
Data logging makes use of the MRAM, a special battery backed memory area in the NA. This reduces the number of writes to an SD/USB device, and is managed correctly during power off.

### 12.2 Limitations of the system

- Maximum number of characters per Data Set name: 64
- Maximum number of Data Sets: 100
- Maximum number of Variables per Data Set: 128
- Maximum number of Variables per Project: 512
- Logging of Structures & Array Variables limited to members/elements only
- It is not possible to set the 'Target Device' as a Network Drive.
- Only CSV format supported

### 12.3 Setting up a Data Set

To create a new data set, in the solution explorer click on Data Logging (right click) -> Add -> Data Set. The following will appear:

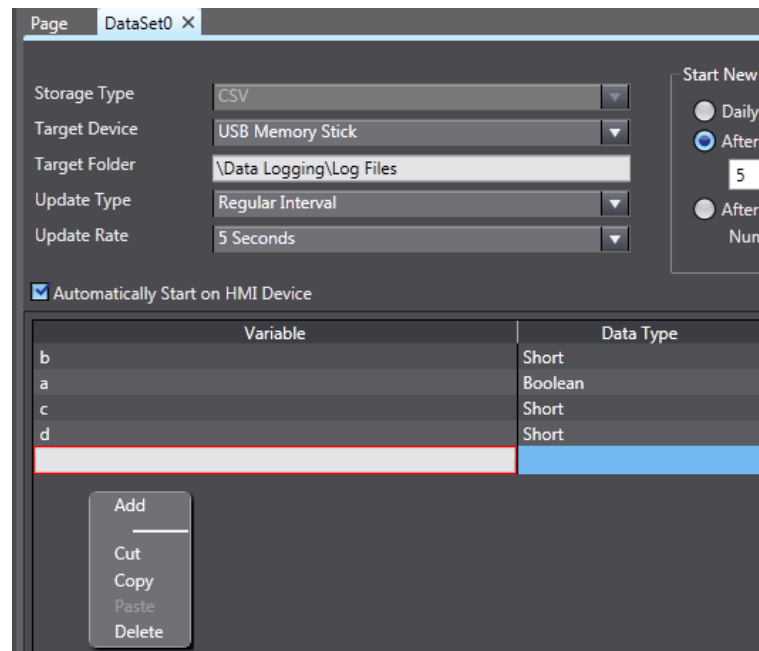


Each Data Set has the following properties:

- **Name** – a unique name for this Data Set (Sysmac Studio naming rules apply).
- **Storage Type** – fixed as CSV in version 1.0.
- **Target Device** – SD Card or USB Memory Stick.
- **Target Folder** – Location on the target device (refer to Notes below).
- **Update Type** – Determines when Data Set values are logged (Regular Interval or On Condition).
- **Update Rate** – Interval rate (only when Update Type is 'Regular Interval').
- **Expression** – Trigger Expression (only when Update Type is 'On Condition').
- **Automatically Start on HMI Device** – automatically begin logging when device is powered on.
- **Start New Database File**
  - Daily – Specifies new Data Set files should be started daily.
  - After Specific Time Period – Specifies new Data Set files should be started after a specific time period.

- Time Period – Specifies the time period.
- After Specific Number of Logs - Specifies new Data Set files should be started after a specific number of logs have been written to the existing one.
  - Number of Logs – Specifies the number of logs.

Once the basics of the data set have been configured, variables can be added to it. Up to 128 variables can be added to a dataset.



The user can change the order of variables on the list. This will change the order in which the variable and its values are being written to the csv file.

## 12.4 Data logging in VB script

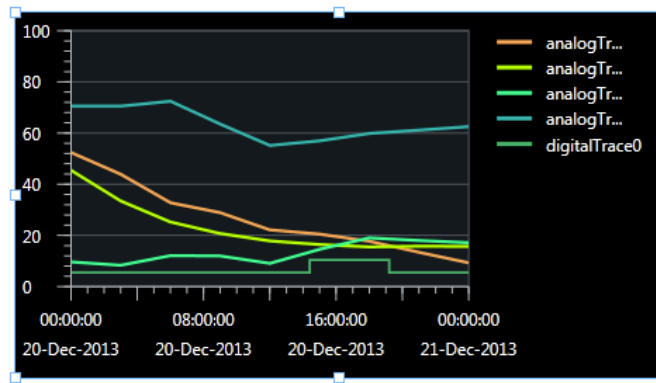
The following actions and events are available for user, this are also accessible through the VB script:

StartDataLogging	1. DataSet Name	1. String 1. String Variable	Starts data logging for the specified dataset.
StartDataLogging	<None>	<None>	Starts data logging for all datasets.
StopDataLogging	1. DataSet Name	1. String 1. String Variable	Stops data logging for the specified dataset.
StopDataLogging	<None>	<None>	Stops data logging for all datasets.
ClearDataLogBuffer	1. DataSet Name	1. String 1. String Variable	Clears the current data logging data stored in MRAM.
ClearDataLogBuffer	<None>	<None>	Clears the current data logging data, stored in MRAM, for all datasets.
ExportDataLogBuffer	1. DataSet Name	1. String 1. String Variable	Exports the current data logging data, stored in MRAM, to the currently active data logging log file.

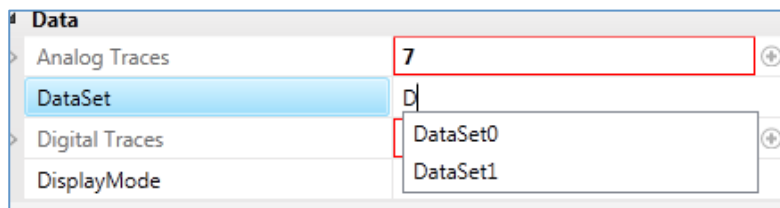
## 12.5 Trend Graph Object

The NA Series includes a Trend Graph object designed to show values from a dataset. The primary purpose of the Trend Graph is to show the profile of values of a particular dataset.

*Note: the trend graph is supported in v1.11 and above of Sysmac Studio*



Once the Trend Object is on a page the user can set up its properties. The most important property is the DataSet. This determines which of the datasets this trend viewer is associated with. Note: it is not currently possible to change the dataset at runtime via code.



Once the DataSet is specified, the user can choose the traces that will be displayed in runtime. It is possible to use with Analogue traces (for example for integer values), or digital traces (for Booleans). Up to 16 analogue traces and 32 digital traces can be displayed on a trend viewer object.

Each trace has a set of individual properties to choose from. There are a lot of other properties that enables user to set up detailed appearance of the Trend Viewer object

Some more important miscellaneous properties:

1. **Background Colour:** sets the colour to be used for the outer area of the Trend graph
2. **Graph Colour:** sets the colour to be used for the actual graph area of the Trend.
3. **Default Display Mode:** startup display mode the default is 'Analogue & Digital'.
4. **Show Current Log File:** determines whether the name of the data log file currently being accessed is displayed on the Trend.
5. **Show Breaks in Logged Data:** determines whether periods of time where data logging is inactive should still be shown on the Trend graph, or whether it should be filtered out using stop/start markers.
6. **Enable Gesture control**

### 12.5.1 Cursor Properties

#### Cursor 1 Status

The 'Cursor 1 Status' property determines whether Cursor 1 is displayed or hidden when the Trend is first displayed. This is a dropdown list field that contains the following options:

- Show
- Hide

The default is 'Hide'.

#### Cursor 2 Status

The 'Cursor 2 Status' property determines whether Cursor 2 is displayed or hidden when the Trend is first displayed. This is a dropdown list field that contains the following options:

- Show
- Hide

The default is 'Hide'.

## 12.5.2 Data Scale Properties (Left & Right)

### Visibility

The 'Visibility' property determines whether the value scale (left scale) is displayed or hidden. This is a dropdown list field that contains the following options:

- Show
- Hide

The default is 'Show'.

### Colour

The 'Colour' property sets the colour to be used for the value scale. This is a colour selection property. The default colour will be black.

### Font

The 'Font' property (and associated sub-properties) determine the font settings that should be used for the value scale values. This will be standard font fields. The default colour will be the same as the value scale colour property.

### Format

The 'Format' property determines the format that that value scale values will be displayed in. This is a dropdown list field that contains the following options:

- #
- #.#
- #.##

### Minimum Value

The 'Minimum Value' property specifies the lowest value on the left scale. This is an edit box field that allows a numeric value to be entered.

### Maximum Value

The 'Maximum Value' property specifies the highest value on the left scale. This is an edit box field that allows a numeric value to be entered.

### Minor Ticks Visibility

The 'Minor Ticks Visibility' property determines if the minor tick markers should be displayed on the value scale. This is a dropdown list field that contains the following options:

- Show
- Hide

### Number of Major Ticks

The 'Number of Major Ticks' property determines how many value fields will be shown on the value scale. This is a dropdown list field.

### Grid Line Visibility

The 'Grid Line Visibility' property determines whether additional grid lines will be drawn horizontally across the Trend graph at each of the major tick points. This is a dropdown list field that contains the following options:

- Show
- Hide

## 12.5.3 Time Scale Properties

### Visibility

The 'Visibility' property determines whether the time scale is displayed or hidden. This is a dropdown list field that contains the following options:

- Show
- Hide

The default is 'Show'.

### Colour

The 'Color' property sets the colour to be used for the time scale. This is a colour selection property. The default colour will be black.

### Font

The 'Font' property (and associated sub-properties) determine the font settings that should be used for the time scale information. This will be standard font fields. The default colour will be the same as the time scale colour property.

### Format

The 'Format' property determines the format that that time scale information will be displayed in. This is a dropdown list field that contains the following options:

- DD/MM/YYYY  
HH:MM:SS
- MM/DD/YYYY  
HH:MM:SS
- DD/MM/YY  
HH:MM:SS
- MM/DD/YY  
HH:MM:SS
- HH:MM:SS

### Number of Time-Stamps

The 'Number of Time Stamps' property determines how many time stamps will be displayed on the time scale. This is a dropdown list.

### Grid Line Visibility

The 'Grid Line Visibility' property determines whether additional grid lines will be drawn vertically down the Trend graph at each of the major time-stamp markers. This is a dropdown list field that contains the following options:

- Show
- Hide
- 

## 12.6 Using the Trend Graph at Runtime

There are several modes in which the Trend Graph Object can operate:

- a) Show analogue and digital data
- b) Show analogue data only
- c) Show digital data only
- d) Show "live" data
- e) Show "historical" data

### 12.6.1 Scaling and zooming

The Trend Graph contains two scales. Both scales are user configurable in Sysmac Studio and will remain fixed when running on the HMI device. Each analogue trace will be assigned to either the Left or Right scale and during runtime operation, the position of each plot will be relative to the scale it is associated with.

When zooming operations are carried out on the Trend graph then both scales will be zoomed.

## 12.7 Trend Graph in VB, Actions and Events

The following actions are supported:

Actions (Script Functions)		Description	Location	
			Action Editor	VB.NET Code
1	ScrollTrendToEnd	Scrolls the specified trend graph to the latest point in time.	No	Yes
2	ScrollTrendToStart	Scrolls the specified trend graph to the start of the logged data (for the current log file being viewed).	No	Yes
3	ScrollTrendForward	Scrolls the specified trend graph by one display width forward (to show newer data).	No	Yes
4	ScrollTrendBackward	Scrolls the specified trend graph by one display width backward (to show older data).	No	Yes
5	ScrollTrendUp	Scrolls the specified trend graph upwards to	No	Yes

		the next major tick marker (to show data with higher values). This function only works when the Trend graph is zoomed.		
6	ScrollTrendDown	Scrolls the specified trend graph downwards to the next major tick marker (to show data with lower values). This function only works when the Trend graph is zoomed.	No	Yes
7	ScrollTrendOverlayForward	Scrolls the overlaid data, on the specified trend graph, by the specified time offset in the right direction.	No	Yes
8	ScrollTrendOverlayBackward	Scrolls the overlaid data, on the specified trend graph, by the specified time offset in the left direction.	No	Yes
9	ZoomIntoTrend	Zooms into the specified trend graph by 100% from the centre point of the graph. It will be possible to zoom into the X axis, Y axis or both based on the parameters passed.	No	Yes
10	ZoomOutofTrend	Zooms out of the specified trend graph by 100% from the centre point of the graph. It will be possible to zoom out of the X axis, Y axis or both based on the parameters passed.	No	Yes
11	ShowTrendCursor	Shows or hides the specified cursor for the specified trend graph.	No	Yes
12	MoveTrendCursor	Moves the specified cursor on the specified trend graph.	No	Yes
13	RemoveOverlayGraph	For the specified trend graph, removes the overlaid data from the Trend graph.	No	Yes
14	SetTrendDisplayMode	Sets the trend graph display mode to the specified setting (analogue and digital, analogue only or digital only).	No	Yes

## 13 LOCALISATION

### 13.1 Resources

Each string, image, movie and document used within a project is known as a resource. Resources are kept in a central library and can be used throughout the project. Multiple objects can use the same resource e.g. a 'Back' button may be used many times. Once using a resource, if the resource in the central library changes, all uses of this resource will reflect this change.

Therefore resources are central to understanding how localisation is achieved in the NA Series.

### 13.2 What is a Default Language?

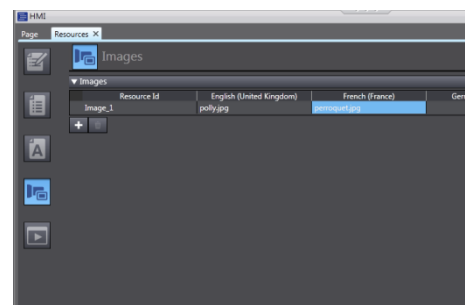
Every project has a Default Language that can be configured to be any language. All resources in the resource library use this default language as a point of reference, the default language is the point from where translations will be taken and the language you develop your HMI in. When no other localised resource exists for a particular string, document or video, the default resource will be used. Therefore the default language is best to be defined as the most common language in your machine.

### 13.3 Resource Types

There are several things within a project that can be language dependent, these are:

- General Strings
- Alarm Strings
- Documents
- Images
- Videos

New resources can be added by pressing the + button. Alternatively, whenever a new string, image or movie is first used on a page it is created in the resource library. It is possible to edit the resource even after one or more objects are using it – all objects using this resource will automatically reflect the changes made.



#### 13.3.1 General Strings and Alarms

The most common resources within a project are the General strings. General strings are the text on buttons, labels and other items throughout the HMI. When creating the project, any text used will automatically be added as a General String resource. As with other resources, they can be added in advance using the + button and then when configuring the text on an object the resource will be offered from the General strings to use in the text field.

Each time text is added to a new object one of the existing 'Default language' resources can be selected or a new one created. A list of available resource strings will be shown when text is entered, the more text entered the shorter the list. If the string entered does not a new resource will be created in the resource library.

Alarm strings are exactly the same as general strings, except they can be much longer. The Alarm Message and Alarm Details are stored as resources.

#### 13.3.2 Pictures, Movies and Documents

All media such as pictures, movies and documents are stored in the resource library. Pictures and movies are automatically added when you select the file on a page object or, in the same way as text resources, you can add in advance using the + button. Documents can only be added in advance as they are opened from a script command rather than a page object.

### 13.4 Viewing and Adding Languages

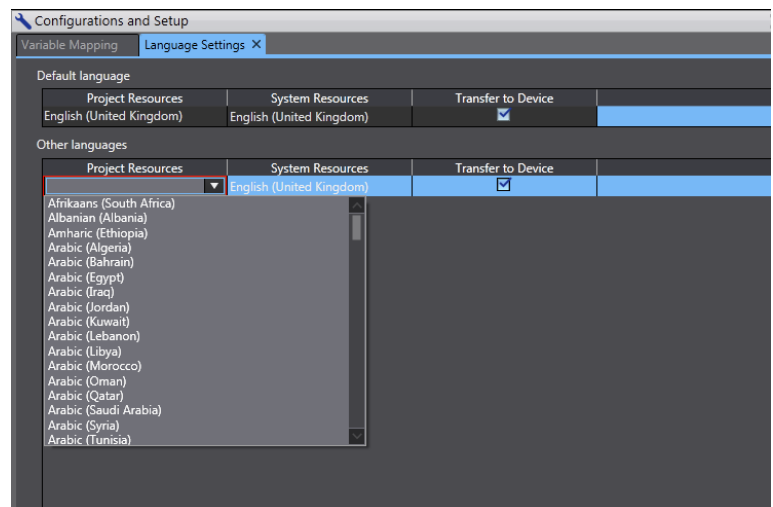
Language settings are located under the Configurations and Setup menu in the project window. A new project has just a default language configured; this default language can be set to any language.



The start-up language is the language the HMI loads when switched on. This can be changed to be different to the default.

New languages are added using the + button. When you add a new language, you have a number of things to set:

- Project Resources are the language you want the user to see and use. This is selected from a dropdown list of all the possible languages that WEC7 lists (137)
- System Resources are the language used in areas not specific to the project such as error messages, pop up keyboards and decimal separation (dot/comma). Version 1.0 supports nine system languages.
- Transfer to device. Not all languages need to be transferred to the device, this can be of use if there are lots of videos or pictures in the project which could use too much space on the HMI.



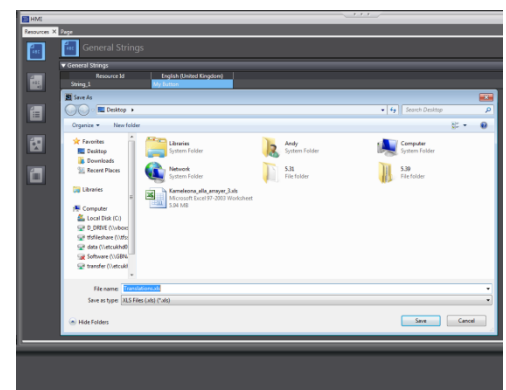
The 'Trash' button can be used to delete a language. Note: this will delete any translations that have been added for that language.

## 13.5 Translating Resources

Each resource can have alternative translations for each language. Using a central library of resources makes it easier to use professional translator as all resources are in the same place, corrections and changes are easy and quick to implement.

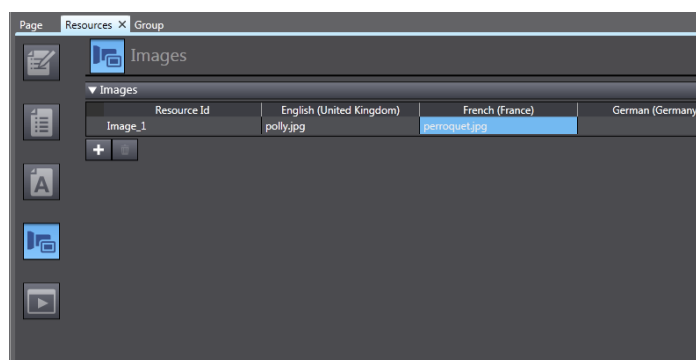
The General Strings in the Resource Table can be exported to a file that can be sent to an external company for translating. The resulting translated file can be imported back into Sysmac Studio.

Alarm strings (both message and detail) can be translated. Again it is possible to export the tables and translate in excel.



### 13.5.1 Images, Videos and Documents

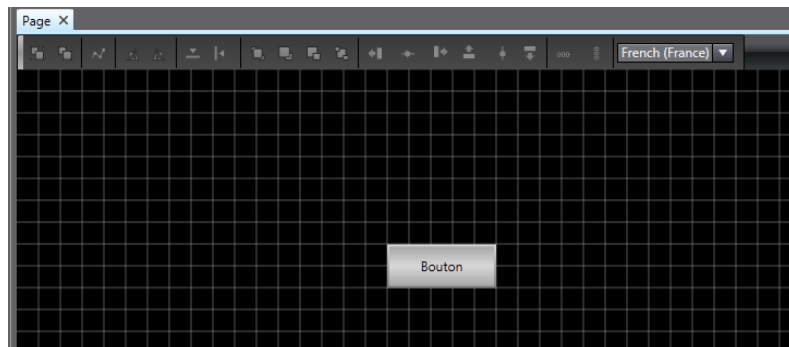
Clicking in each cell in the images, documents or videos allows the user to click a button that shows a file open dialog. Using this a different file can be selected for each language. As with all resources if you don't supply an alternative, the default will be used.



### 13.6 Viewing Different Translations on a page

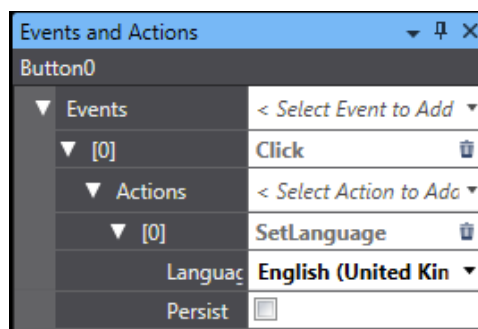
When viewing the pages, it is possible to check how other languages will appear on the page. Using the language drop, select a different language. This enables the developer to:

- Check the translations fit the space
- Confirm translations look correct with the layout



### 13.7 Actions & VB

There are standard functions in the NA that can be used to change the language during the runtime. Actions can be used to get and set the current language. This could be associated with a dropdown box, or a button.



The option to 'Persist' ensures that, when the NA is restarted, the chosen language will be loaded automatically.

In addition two VB methods can be used to change the language:

GetLanguage(ReturnString) - Gets the current language  
 SetLanguage(Language) - Sets the current language

Note that both of these functions use a 2-2 code e.g. en-GB. For more information on the language codes in .NET see the following MSDN page:

[http://msdn.microsoft.com/en-us/library/ms533052\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms533052(v=vs.85).aspx)

## 14 INTELLIGENT APPLICATION GADGETS (IAGs)

### 14.1 What Is an IAG?

An IAG (Intelligent Application Gadget) is an object designed either by Omron or by a user to enhance structure and reuse within a program. It is similar to the concept of PLC Function Blocks. IAG objects can be designed to fulfill a specific purpose. An IAG can contain a mixture of one or more graphics and objects and can also contain functionality such as methods.

An IAG cannot be edited and it is not possible to see anything inside the IAG. This makes IAGs ideal for protecting intellectual property whilst still allowing the user the flexibility to configure their use of the object and the rest of the application. Note: At v1.0 IAG files are not encrypted so more expert users will be able to view code within IAGs.

### 14.2 What Is an IAG Collection?

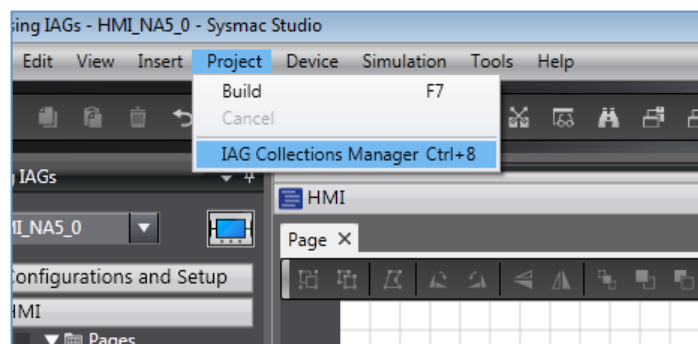
Multiple IAGs can be packaged into an IAG collection distributed as a single file. The collection file includes the names of the categories that the IAGs are split into, when installed the IAGs will be found within the Toolbox.

The installation of IAGs is computer based so when installed; an IAG is available to all projects on the computer. Likewise if deleted, the IAG objects will be removed from the toolbox and no longer available for use in projects (note: projects already using the IAG will continue to function correctly).

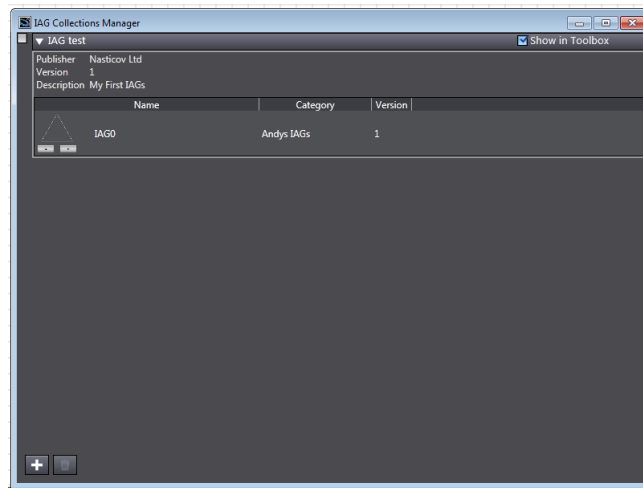
If an IAG is used in a project, the entire Object is copied into the project. If you subsequently transfer the project to a computer without the IAG installed, the IAG in the project will not be affected.

### 14.3 How to Install IAG Collection

To import an IAG collection to Sysmac Studio, use Project | IAG Collections Manager:



Using the '+' button it is possible to select the collection file to install.



Once installed, the IAG collection will be added to the manager. In order to show these IAGs in the current project make sure 'Show in Toolbox' is checked. This allows you to control which objects are shown in each project.

Once an IAG collection is added to the toolbox for this project all the IAGs in the collection will be shown in them. To use them drag the objects onto a page.

#### 14.4 Properties on the IAG

IAG properties show various aspects of the IAG including the version number and who created the IAG. The properties also include appearance and layout, which enable the user to change the background colour, its size and position on screen. Besides these, user interface variables added by the IAG designer will be shown. These must be linked to NA variables within the project.

Once an IAG is added to the page, it is possible to customise the appearance of the objects within the IAG. To view the properties of the objects within an IAG, use the Page Explorer.

In addition to the appearance, any user variables exposed by the IAG developer will be shown in the behaviour section. This includes 'Input' variables where the user can enter expressions, constants or leave them blank, as well as 'Input/Outputs' which must be linked to variables.

#### 14.5 Changing IAG Objects From Code

It is possible to customise objects within an IAG from code by referencing the object name from outside the IAG:

```
IAGName.ObjectName.ObjectProperty = NewValue
```

This is considered an advanced use, and only limited properties are accessible from VB.

#### 14.6 Using IAG methods From Code

In addition IAGs can provide methods (Functions and Subroutines) that can be called from the IAG. The names and parameters of these methods will be set by the IAG developer and their use and properties must be communicated to the user.

The syntax to call an IAG subroutine is:

```
iag_name.subroutineName parameters
```

Or, for a function:

```
ReturnValue = iag_name.functionName(parameters)
```

#### 14.7 Using VB Variables From Code

It is possible that the IAG contains VB variables that are accessible from code.

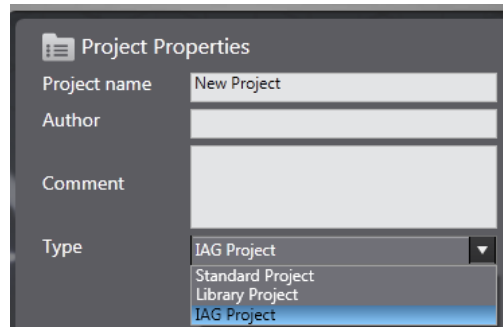
```
iag_name.variableName = newValue
```

The names of these public variables would have to be documented separately, but will be shown by auto complete in the code editor.

## 15 CREATING IAGS

### 15.1 Creating an IAG project (Collection)

IAGs are created in Sysmac Studio by creating IAG projects. Start Sysmac Studio, click to create a New Project and select 'IAG Project' from the type dropdown menu.



### 15.2 Structure of IAGs

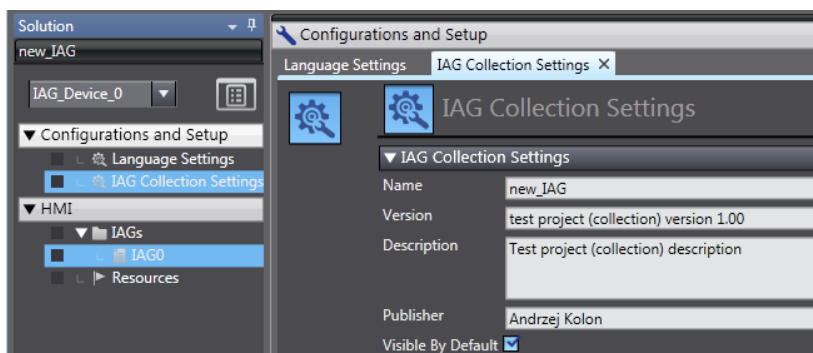
The structure of an IAG Project (Collection) is different to a standard HMI project.

In Language settings the user can:

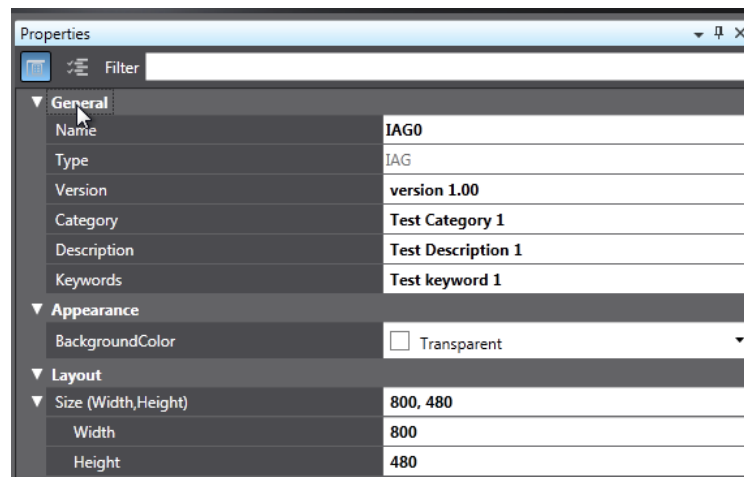
- Set up Default Language
- Add Defined Languages

**Note:** The languages selected here will be the ones that IAG will support when it is used.

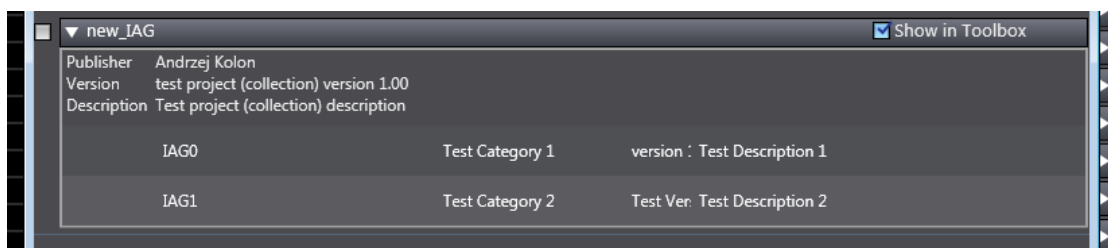
In IAG Collection Settings the user can include Basic IAG Collection information (This is the information that will appear in the IAG Collection Manager when installing):



IAGs are created in a similar way to pages in a normal project. By default there will be one IAG (IAG0) in the new IAG project. To create further IAGs go to HMI | IAGs | (Right Click) | Add IAG. To get to individual IAGs properties right click on the background and select properties.



Using these properties the IAG can be described and later recognised in the IAG Collection Manager.



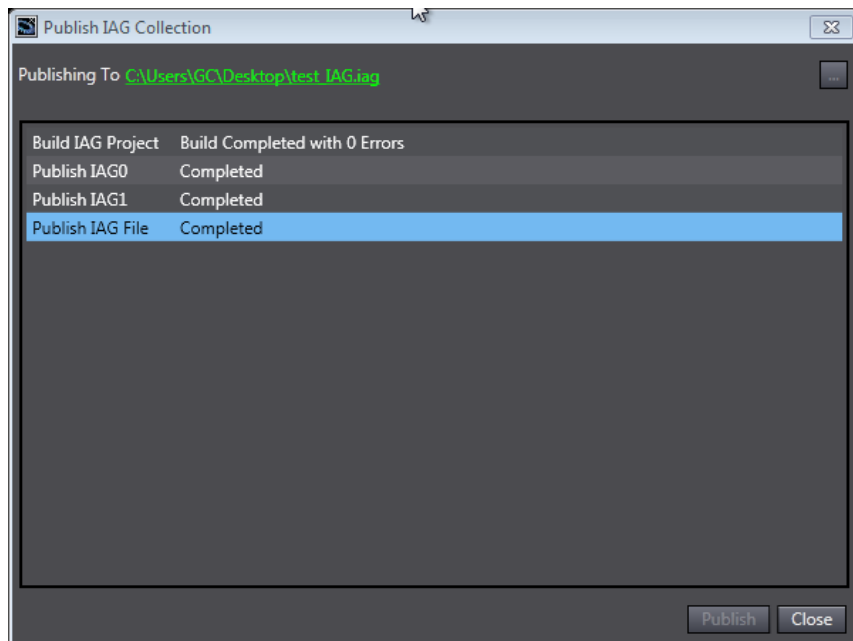
### 15.3 IAG Resources

IAG Project (Collection) Resource tables are shared across all IAGs in the Project (collection), they work in the same manner as in a standard NA HMI project.



An IAG project has to be published in order to use it in the 'standard' NA project. The project with all the IAGs inside will be saved as \*.IAG file which can then be installed on any other computer using the IAG Collection Manager.

To publish an IAG, go to Project | Publish IAG Collection. Configure the path and file name for your IAG project file. Click publish following compilation and preparation it will display the following:

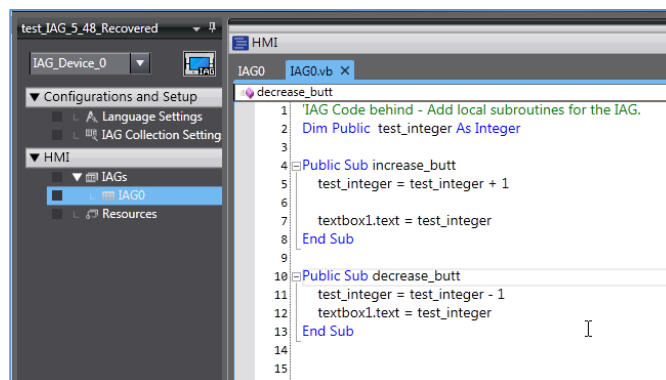


## 15.4 How to develop an IAG

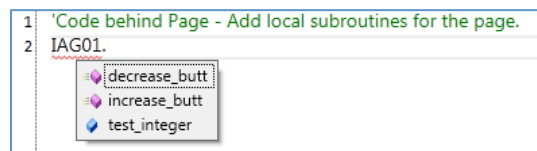
In principle developing an IAG is no different than developing a “standard” HMI project and page. The most important thing is to remember what parts of the IAG will be “exposed” outside of the IAG, once it is published.

Every IAG has its own Design, Code and User Variables. User Variables will eventually be displayed as properties when the IAG is used in a standard project. They allow data to be linked to the instance of the IAG when used. Appearance properties of all the object used in the IAG will be exposed outside of the IAG so that the look can be tailored to suit the needs of the standard HMI project.

It is possible to add VB code behind the IAG and declare functions & methods that can be called by the user of the IAG.



When using the IAG in a standard project, these methods (and any parameters) will be visible to the user through auto complete.



## 15.5 User variables

User variables are used to define the data interface for the IAG. When used in a standard project these variables will link to data in the standard HMI.



IAG0 User Variables X				
Internals	Name	Data Type	Initial Value	Comment
In/Out	myInternalVar	Integer		

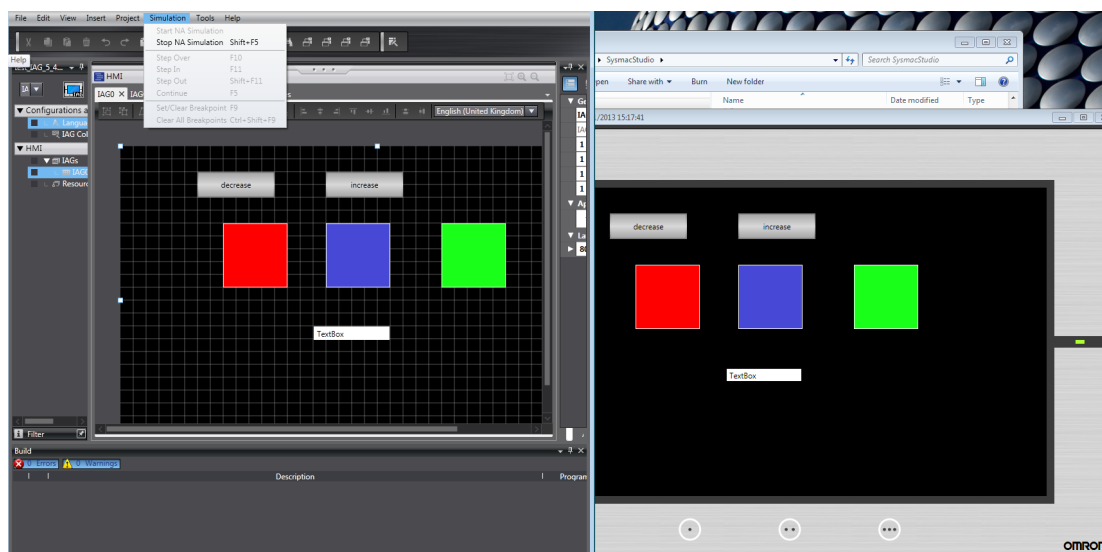
User Variables can be 'internals' which are only seen within the IAG and not exposed to the user of the IAG, 'Input' variables which can be assigned expressions, constants or left blank when the IAG is used, or 'In/Out' variables which must be linked to a variable when the IAG is used.

Variables can be given an initial value, and these will be shown to the user of the IAG as the 'default' value when the IAG is added to the page.

<b>Layout</b>	
Position (Left,Top)	71, 55
Size (Width,Height)	221, 214
<b>Behavior (Input)</b>	
myInput	
<b>Behavior (In/Out)</b>	
myOutput	

## 15.6 Debugging an IAG

It is possible to debug an IAG project like any other NA HMI project (with all the restrictions of the NA project simulation)



Additional buttons could be added to the IAG to allow testing of methods. The watch window can be used to simulate user variable data.

Note: it is not possible to call methods on the IAG directly from the debugger.

## 16 SIMULATION

### 16.1 Introduction

Simulation is the representation of the behaviour or characteristics of one system through the use of another system, especially a computer program designed for the purpose.

- The NA Simulator displays a window which looks like the HMI and has the same functionality
- The NA Simulator uses the same 'runtime' as the actual device so will behave the same
- Can be used alongside the NJ simulator for a complete machine simulation
- Simulation allows use of a debugger to step through VB Code

### 16.2 Benefits of Simulation

Simulation is an essential part of efficient development. When developing its great to be able to quickly see what something looks like or try out some VB Code without having to spend time sending the project to the hardware device.

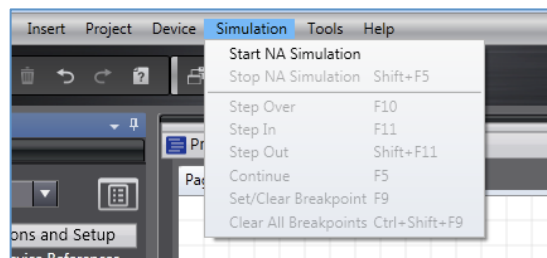
- Quick to try out or test new functionality or code
- No need to have the hardware nearby
- Possibility to debug code and single step through routines
- See things as they will look on the NA
- Can communicate with NJ Simulator to simulate the machine without using any hardware

#### 16.2.1 Limitations with Simulation

- Not always an accurate gauge of performance
  - Development PC will have more powerful CPU than NA
  - Development PC may be running other tasks
  - Communication with device is often a bottleneck
- Cannot communicate with actual NJ/CJ hardware
- Can only communicate with a single simulated NJ
- Some NA Features are supported during simulation by the Operating System which can differ between Simulation (on Win7) and WEC7 such as: PDF Viewer
- Some NA Features are not supported during simulation such as Trend Graph, Movies

### 16.3 Standalone Simulation

To start simulation, click on 'Simulation | Start NA Simulation'. If necessary this will build the project and (if successful) will launch the simulator. Once launched it is possible to use the computer mouse to simulate clicks on the touch screen.



Text entry can be made either using the popup keyboards, or by using the computer keyboard.

In simulation, all variables start with their default values.

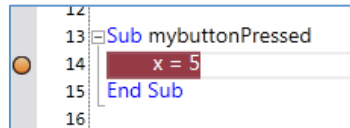
### 16.4 Integrated Simulation of NA and NJ

To start integrated simulation, click on 'Simulation | Start NA Integrated Simulation'. Like standalone simulation this will build the project if necessary and then launch both simulators. The user is asked which NJ controller they would like to simulate (if there are multiple controllers in the same project).

Integrated simulation allows for testing of the relationship between the HMI and the controller, data can be more realistically controlled to reflect the real machine.

## 16.5 Setting Breakpoints

A breakpoint causes the simulator to pause during execution. It is only possible to use breakpoints when using the simulator (not a real NA). Breakpoints can be set on any line of VB.net code. These can be set either before launching the simulator, or whilst running the simulator. To set a breakpoint there is an option in the simulation menu, or it is possible to click on the left hand margin within the code (or just press F9 whilst on a line of code).



There is an option to 'Clear All Breakpoints' which can be used when required.

V1.0 does not support any conditional breakpoints e.g it is not possible to break when a variable reaches a certain value.

When a breakpoint is reached during runtime, the execution stops and control returns to Sysmac Studio. The user is then able to control the program execution manually using the Simulation menu:

- Step Over – Run the highlighted line of code and move onto the next line
- Step In – If the line of code is a subroutine, step into the subroutine and keep debugging
- Step Out – If currently in a subroutine, run until you have finished the subroutine.
- Continue – Run until you reach another break point

## 16.6 Using the Watch Window

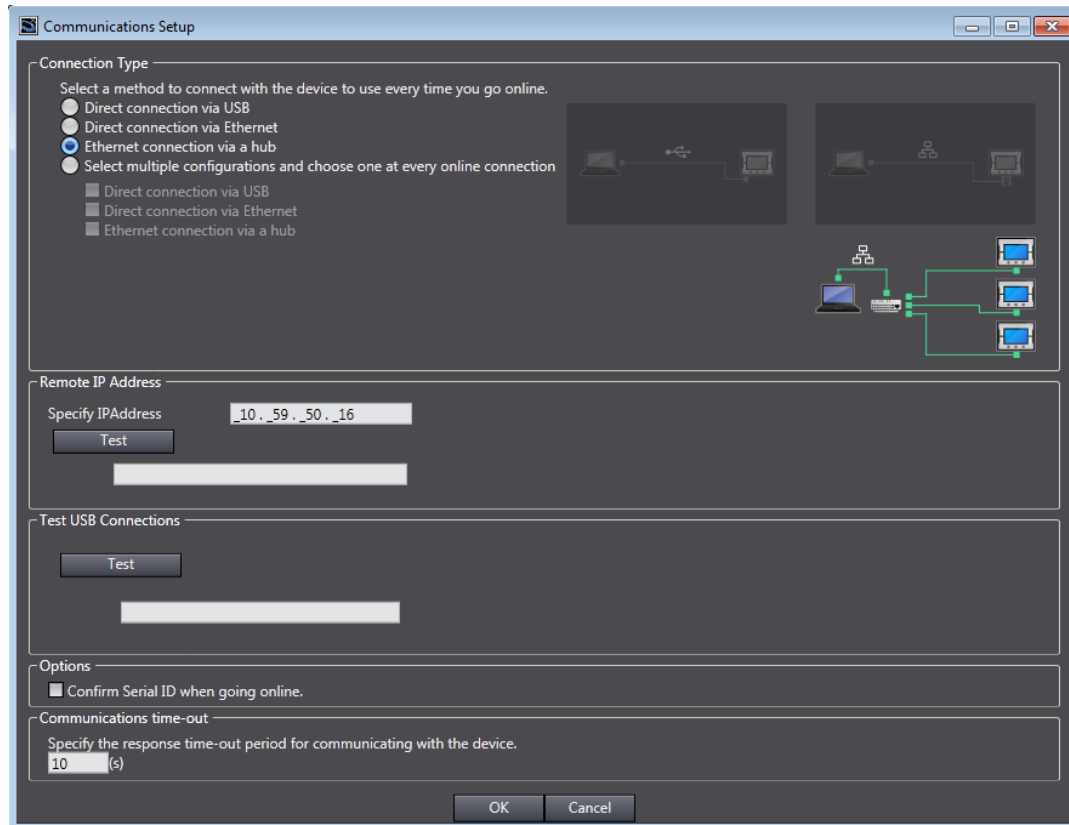
When execution is paused (e.g. a breakpoint is reached) variables can be added to the watch window by typing the variable name and using the intelli-sense prompt. For arrays, square brackets must be used (note: multi-dimension arrays are not supported in v1.0).

Watch (Project)							
Controller name	Name	Online value	Modify	Comment	Data type	AT	Display for
HMI_NA5_0	FloatArray[0]				Double		Real
HMI_NA5_0	z	20			Object		
HMI_NA5_0	intNumber				Integer		Decimal
HMI_NA5_0	Input Name...						

## 17 SCREEN TRANSFER

### 17.1 Communication Setup

When user selects “HMI | Communication Setup” the following screen will popup (Similar to NJ communication setup):



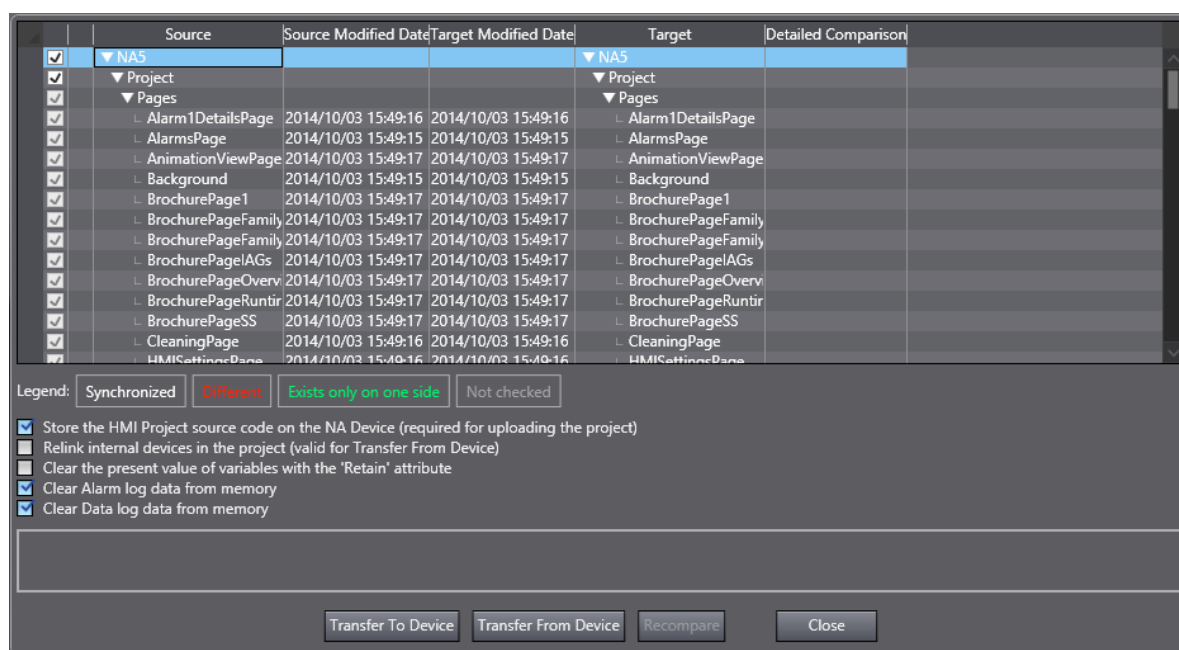
- **Connection type:**
  - **Direct connection via USB** – direct connection from PC to HMI device, via standard USB cable. Requires no additional settings.
  - **Direct connection via Ethernet** – direct connection from PC to HMI device, via Ethernet cable. Cross-over or standard Ethernet cable can be used. Does not require IP Address or any other additional settings.
  - **Ethernet connection via a hub** – connection from a PC to HMI device via an Ethernet network. IP Address must be specified in this case.
  - **Select one method from these options at every online connection** – In addition to the previous options it is possible to specify that multiple connections are configured and a choice should be offered every time the transfer operation is initiated (as shown in the image above). Those options that are selected will appear as a choice when Synchronisation initiated.
- **Remote IP Address** – Specifies the IP Address to be used each time the transfer operation is initiated. This is only relevant when using a remote Ethernet connection. The test buttons allow the connection, using the specified IP Address, to be validated.
- **Confirm the serial ID when going online** – forces the Serial Number of the HMI device to be validated against that specified in the Sysmac Studio HMI project (not applicable for SC/USB, if ID is different the standard Sysmac Studio dialog will be displayed asking for confirmation on how to proceed)
- **Communications Timeout** specifies the time-out period if communications with the specified device fails.

### 17.2 Synchronise with NA Device

When the Synchronise option is invoked the following will happen:

1. If only one connection is configured then the system will go directly to the ‘Synchronisation’ window allowing the user to select the required transfer operation

2. If multiple connections are configured then the user will be presented with the window requesting which transfer method should be used. Once the connection type is selected and OK'd then the 'Synchronisation' window will be displayed

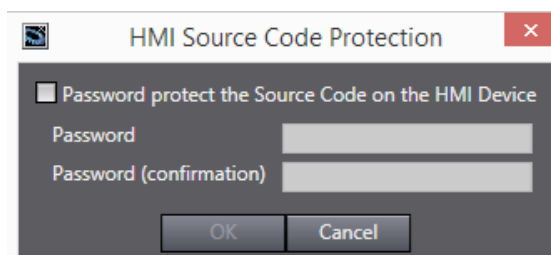


The 'Synchronisation' window provides a comparison of the project data in the source PC against the project data in the actual device, based on the date and time the files were generated.

	Source Files (PC)	Build Date	Build Time	Build Date	Build Time	Target (Device)	Files
<input type="checkbox"/>	Project	12/10/2012	12:10.13	12/10/2012	12:10.13	Project	
<input type="checkbox"/>	Settings	12/10/2012	12:10.14	12/10/2012	12:10.14	Settings	
<input type="checkbox"/>	User Data	12/10/2012	12:10.15	12/10/2012	12:10.15	User Data	
<input type="checkbox"/>	Runtime Files	12/10/2012	12:10.16	12/10/2012	12:10.16	Runtime Files	

When synchronising the following actions are possible:

- a) **'Store the HMI Project source code...'** this option informs Sysmac Studio to store the source code on the internal storage of the NA. This can be protected with a password using 'HMI | Security | HMI Source Code Protection...'



Source will only be transferred to the device if the 'project' tick box is also checked. It is not transferred if the only changes are HMI Settings, Recipes or Resources. If source is found on the device this option is automatically ticked, and if unticked then the source will be removed during transfer (a warning dialog is shown).

- b) **Relink internal devices in the project (valid for Transfer From Device)**  
When this option is checked, the upload process will relink each internal device reference to an internal NJ Controller in the project with the same name. A device reference will only be relinked to an NJ Controller if all the mappings can be applied successfully. If any of the mappings cannot be applied, the relinking attempt is aborted and the device reference remains unlinked. The user will be informed of the outcome of each relinking attempt in the message area of the Synchronisation View
- c) **Option to delete stored HMI data**

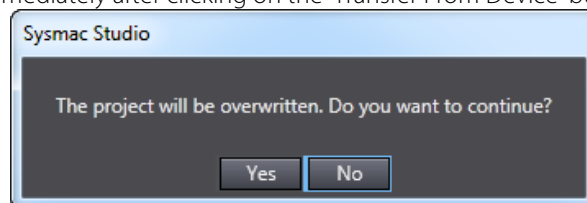
- This allows the user to specify whether retained variables and logged data stored on the HMI device should be automatically deleted after a successful transfer. It will be possible to select which of the following is deleted:
  - Retained Variables
  - Historical Alarms
  - Data Logging Data
  - Event/Error Log
- The warning message will be displayed prior to any of the above being deleted.

The **'Transfer to HMI Device'** option initiates the transfer of all selected files to the HMI device. The following sequence will commence:

- If the 'Confirm the serial ID when going online' option is set then the serial number setting in the Sysmac Studio HMI project will be compared against the serial number of the HMI device. If they match then the process will continue as follows
- The 'Synchronise' window will remain on display and a Sysmac Studio progress indicator will display the percentage complete.
- The HMI device will be put into 'Transfer' mode, ready to receive the new HMI application.
- The HMI device will display a transfer progress page indicating the percentage complete. This will continue to be updated as the transfer operation progresses.
- The relevant files, based on the checked state of each checkbox, will start to be copied to the HMI device. The Sysmac Studio progress indicator should be updated accordingly.
- When all files have been successfully transferred a status message will be displayed in the 'Synchronisation' window. If any errors occur during the transfer process then a suitable message should also be displayed in the 'Synchronisation' window.

The **'Transfer from HMI Device'** option initiates the transfer of all selected files to the HMI device. The following sequence will commence:

- If the 'Confirm the serial ID when going online' option is set then the serial number setting in the Sysmac Studio HMI project will be compared against the serial number of the HMI device. If they match then the process will continue as follows
- When uploading protected source code from the NA device, the following 'Verification' dialog will be displayed immediately after clicking on the 'Transfer From Device' button, but prior to uploading anything:

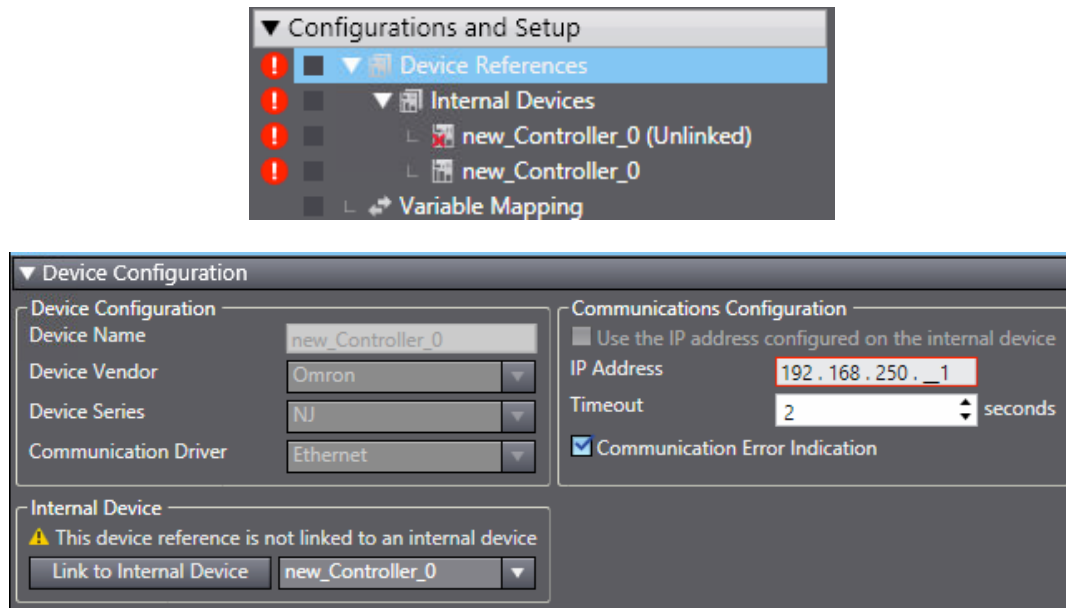


- After confirming the dialog, The 'Synchronise' window will remain on display and a Sysmac Studio progress indicator will display the percentage complete.
- Upload process will not be indicated in any way on the NA Device (3 simultaneous uploads are allowed)
- The relevant files, based on the checked state of each checkbox, will start to be copied from the HMI device. The Sysmac Studio progress indicator should be updated accordingly.
- When all files have been successfully transferred a status message will be displayed in the 'Synchronisation' window. If any errors occur during the transfer process then a suitable message should also be displayed in the 'Synchronisation' window.

### 17.3 Source code Synchronisation

When connecting to an NA, if source is detected, then during synchronise it will be possible to upload the source into the open project. This will replace all the current HMI area of the project with the source found on the NA. If the source on the NA is protected with a password, then the user will be prompted to enter this during the transfer (unless the open project has the same password in Sysmac Studio – HMI | Security | HMI Source Code Protection).

Following the upload, any device references to internal devices need to be remapped. There is an option during synchronise to attempt this automatically during the process. If it is not possible to identify the correct internal device, then an icon will show an 'unlinked' device in the internal devices section. The user can then manually select which is the correct device at which point all the variable mappings will be validated.



## 17.4 Synchronise with Media Device

As well as synchronising with an NA device, it is possible to synchronise straight to USB memory and then use the system menu on the device to transfer the application to the NA.

The following sequence of events will occur:

- The standard Windows 'File Open' dialog is displayed allowing the folder location and file name to be specified on the target media device.



- The compressed and encrypted HMI application executable will then be copied to the media device. A suitable progress indicator will be displayed during this time.
- The user will be prompted when the file has been successfully copied to the target media device.
- Once process successfully finished, it is possible to upload the project from the media device to NA using NA System menu
- It is possible to Synchronise to the same \*.nabin bin file
- It is possible to Synchronise and upload the project form Media Device.







***SYSTMAC***  
always in control