ENHANCED IEC OVERVIEW & FEATURES

NOTE: Enhanced IEC (EIEC) is available for the following controllers: XL Series (EXCEPT XLE/T models), XL Prime Series, Canvas Series, RCC Series, and Micro X5

Enhanced IEC Overview

The following enhancements are available in Enhanced IEC mode:

1. **Introduction of fully tag-based addressing**
2. **Support for 128kBytes of variable space** - The current IEC system works by mapping allocated variables to the PLC %R space. At the time of development, the OCS was limited to 9999 %R's. The Enhanced IEC development will remove the link between variables and %R's and will simply allocate 128kBytes of variable storage – equivalent to 64000 %R registers.
3. **Reduced scan time**
4. **More powerful Debugging and Simulation features** - Breakpoints, Tracepoints can be set in the IEC logic files. Step in, step out and step over are available for IEC code. In simulation mode user can run the IEC program locally on the PC to test and debug during the development phase.
5. **Support for Online Logic Change** - With the addition of online change in IEC the application running in the unit will never be interrupted, even when making updates. The logic running in the OCS can be modified without putting the unit to idle mode. This is ideal for programs which require long periods of start up or shut down so that logic changes can be made without waiting for the machine to start or stop so that logic can be downloaded.
6. **Introduction of Exception Handling** - Allows user to trap errors which previously was difficult to find and frequently cause technical support calls
7. **Support for LREAL, LINT and LWORD Datatypes** - Enhanced IEC mode supports LREAL (64-bit floating-point number), LINT (64 Bit Signed Integer) and LWORD (64 Bit Unsigned Integer) data types.
8. **Support of Bitfield** - User can give meaningful name to every bits of a particular program variable
9. **Support of Enumerated Datatype** - Added support to enumerated data types.

Enhanced IEC Features

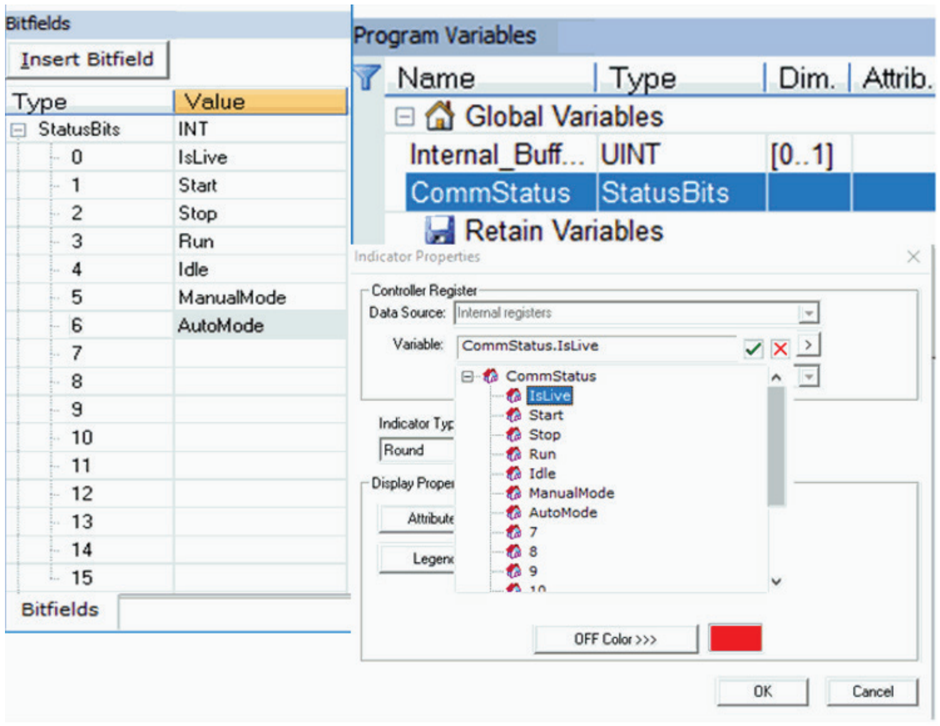
The following enhancements are available in Enhanced IEC mode:

MAN1503 R01
FEB 2024

Enhanced IEC Settings and Data Types

Bitfields

User can give meaningful names to every bit of a particular program variable.
Go to **Tools > Bitfield Window**, a window will appear where in user can insert Bitfield.



Bitfields

Type	Value
StatusBits	INT
0	IsLive
1	Start
2	Stop
3	Run
4	Idle
5	ManualMode
6	AutoMode
7	
8	
9	
10	
11	
12	
13	
14	
15	

Program Variables

Name	Type	Dim.	Attrib.
Global Variables			
Internal_Buff...	UINT	[0..1]	
CommStatus	StatusBits		
Retain Variables			

Indicator Properties

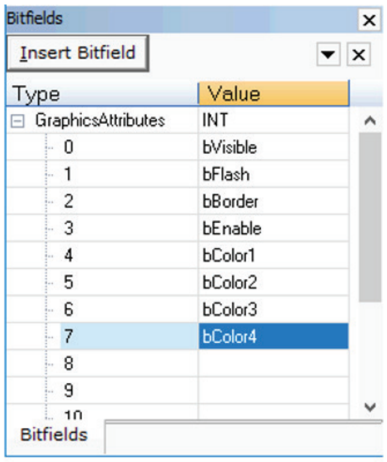
Controller Register
Data Source: Internal registers
Variable: CommStatus.IsLive

CommStatus: IsLive, Start, Stop, Run, Idle, ManualMode, AutoMode, 7, 8, 9, 10

Indicator Typ: Round

Example for Usage of a Bitfield

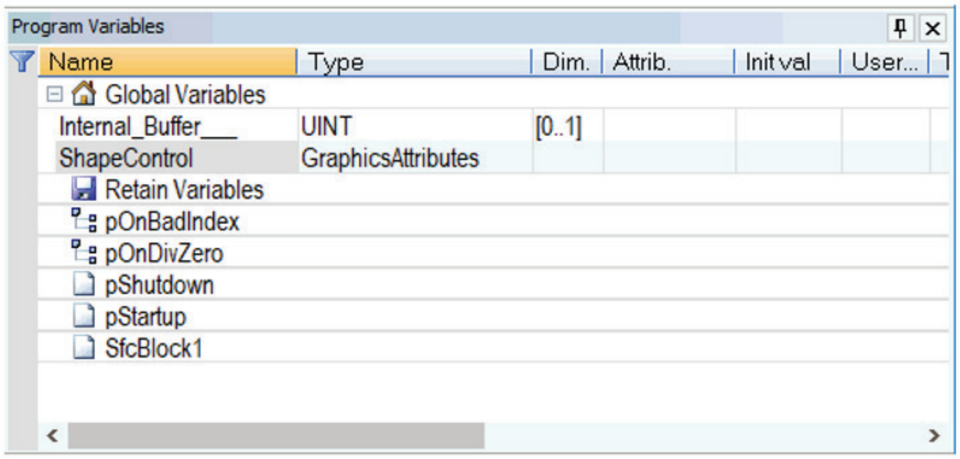
Step 1: Add a bitfield to match the attributes control format for graphical objects (user will need to be out of on line change and/or debug mode):



Bitfields

Type	Value
GraphicsAttributes	INT
0	bVisible
1	bFlash
2	bBorder
3	bEnable
4	bColor1
5	bColor2
6	bColor3
7	bColor4
8	
9	
10	

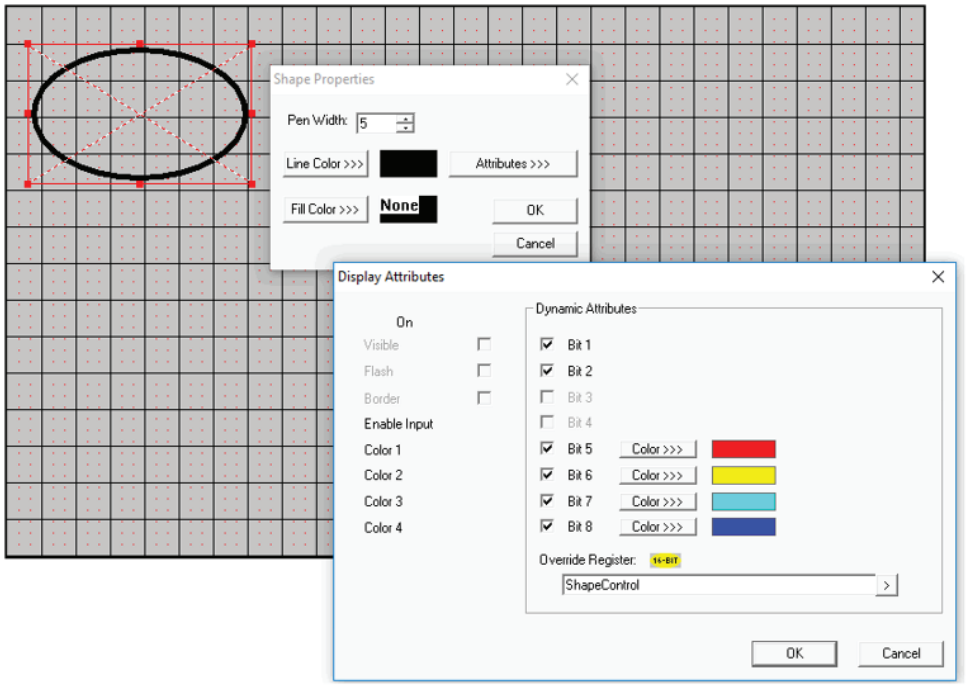
Step 2: Then create a variable of this type:



Program Variables

Name	Type	Dim.	Attrib.	Init val	User...
Global Variables					
Internal_Buffer___	UINT	[0..1]			
ShapeControl	GraphicsAttributes				
Retain Variables					
pOnBadIndex					
pOnDivZero					
pShutdown					
pStartup					
SfcBlock1					

Step 3: Now in the graphics editor create a shape object with attributes controlled by this variable:



Shape Properties

Pen Width: 5

Line Color >>> | Attributes >>>

Fill Color >>> None | OK | Cancel

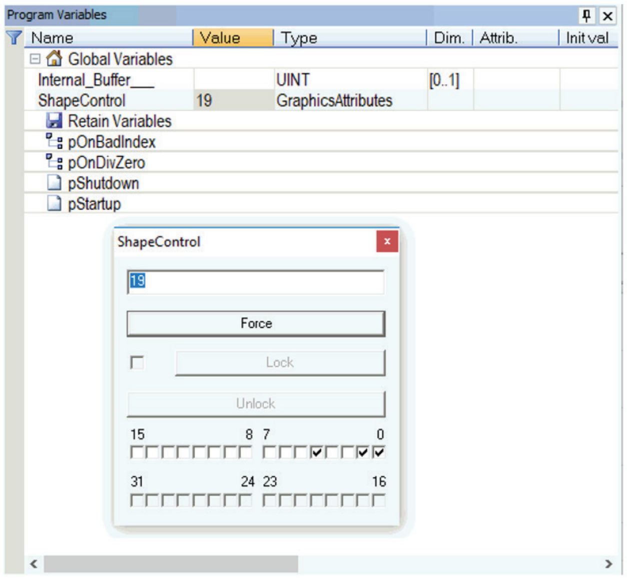
Display Attributes

	On	Dynamic Attributes	
Visible	☐	☑ Bit 1	
Flash	☐	☑ Bit 2	
Border	☐	☐ Bit 3	
Enable Input		☐ Bit 4	
Color 1		☑ Bit 5	Color >>> 🟥
Color 2		☑ Bit 6	Color >>> 🟨
Color 3		☑ Bit 7	Color >>> 🟦
Color 4		☑ Bit 8	Color >>> 🟦

Override Register: 16-BIT ShapeControl

OK | Cancel

Step 4: Download to the OCS and enter debug mode and double click the value of ShapeControl to force the various bits and see the on screen shape change. For example here we set the visible, flash and color bits.



Name	Value	Type	Dim.	Attrib.	Init val
Internal_Buffer___		UINT	[0..1]		
ShapeControl	19	GraphicsAttributes			

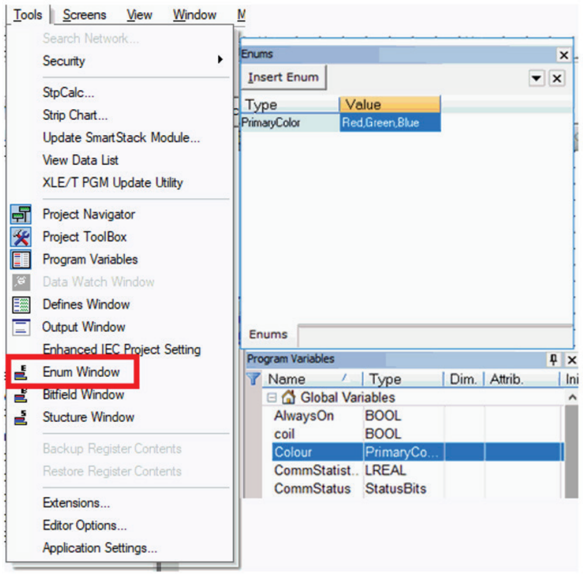
NOTE: Bits such as ShapeControl.bColor4 become valid BOOL variables, for example

```
ShapeControl.bFlash := TRUE;
```

is a valid ST statement.

Enumerated Data Type

User can insert new Enum using "Insert Enum" button.
Go to **Tools > Enum Window**, a window will appear where in user can insert Enum.

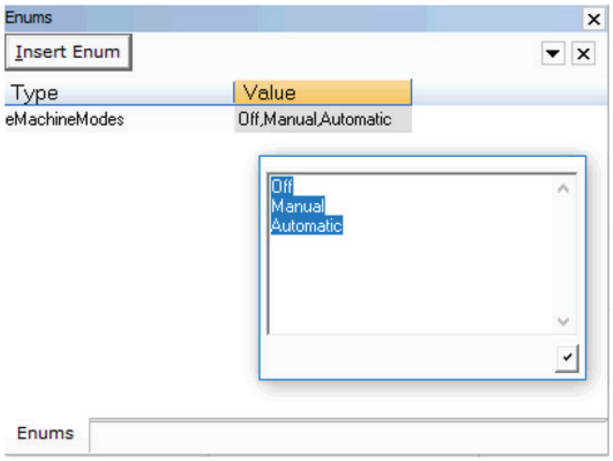


Type	Value
PrimaryColor	Red,Green,Blue

Name	Type	Dim.	Attrib.
AlwaysOn	BOOL		
coil	BOOL		
Colour	PrimaryCo...		
CommStatist...	LREAL		
CommStatus	StatusBits		

Enum Example

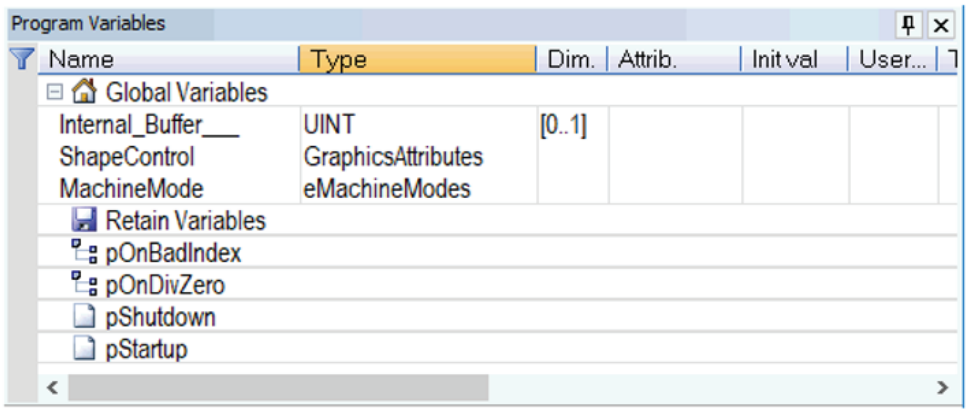
Step 1: Add a new enum eMachineModes with entries Off, Manual and Automatic.



Type	Value
eMachineModes	Off,Manual,Automatic

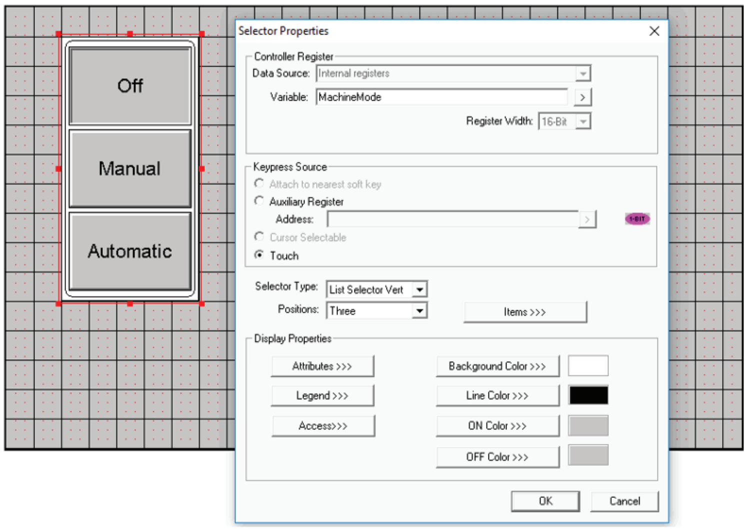
Off
Manual
Automatic

Step 2: Create a new Variable MachineMode of this type:



Name	Type	Dim.	Attrib.	Init val	User...
Global Variables					
Internal_Buffer___	UINT	[0..1]			
ShapeControl	GraphicsAttributes				
MachineMode	eMachineModes				
Retain Variables					
pOnBadIndex					
pOnDivZero					
pShutdown					
pStartup					

Step 3: Go to the Graphics Editor and create a new selector object.



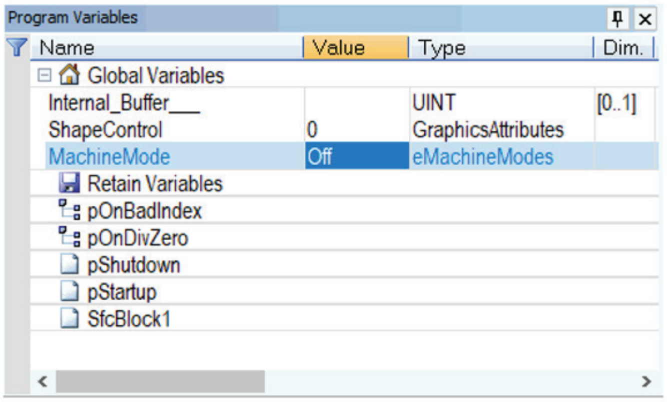
Selector Properties

Controller Register
Data Source: Internal registers
Variable: MachineMode
Register Width: 16-Bit

Keypress Source
( ) Attach to nearest soft key
( ) Auxiliary Register
Address:
( ) Cursor Selectable
(•) Touch

Selector Type: List Selector Vert
Positions: Three

Step 4: Download the Program to the OCS, and then start the debugger.



Program Variables

Name	Value	Type	Dim.
Global Variables			
Internal_Buffer___		UINT	[0..1]
ShapeControl	0	GraphicsAttributes	
MachineMode	Off	eMachineModes	
Retain Variables			
pOnBadIndex			
pOnDivZero			
pShutdown			
pStartup			
SfcBlock1			

Observe How

- Changing the 'value' of MachineMode in the debug window changes the selection in the graphics object.
- Selecting a different item in the on-screen selector item changes the value in the program variables window.

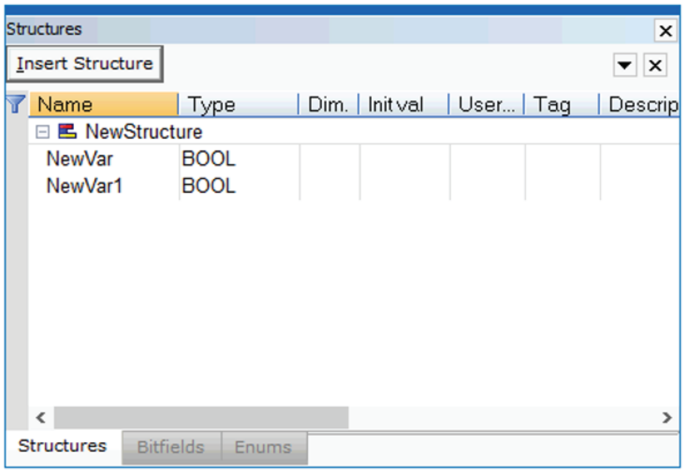
Used in Logic

The values of "MachineMode" can be used in the control logic using (for example in ST):

```
MachineMode := Manual;
If MachineMode = Manual THEN
any operation code
END_IF;
```

Structures

User can insert new structure using **Insert Structure** button.
Go to **Tools > Structure Window**, and a window will appear where in user can insert structure.



Name	Type	Dim.	Init val	User...	Tag	Descrip
NewStructure						
NewVar	BOOL					
NewVar1	BOOL					

"Enhanced IEC Settings and Data Types" on page 2 | "Debugger Windows " on the next page | "Online Change " on page 11
"Simulation " on page 19 | "Stepping " on page 20

Debugger Windows

NOTE: User can debug any IEC program by setting **breakpoints / tracepoints** to the program.

Breakpoints

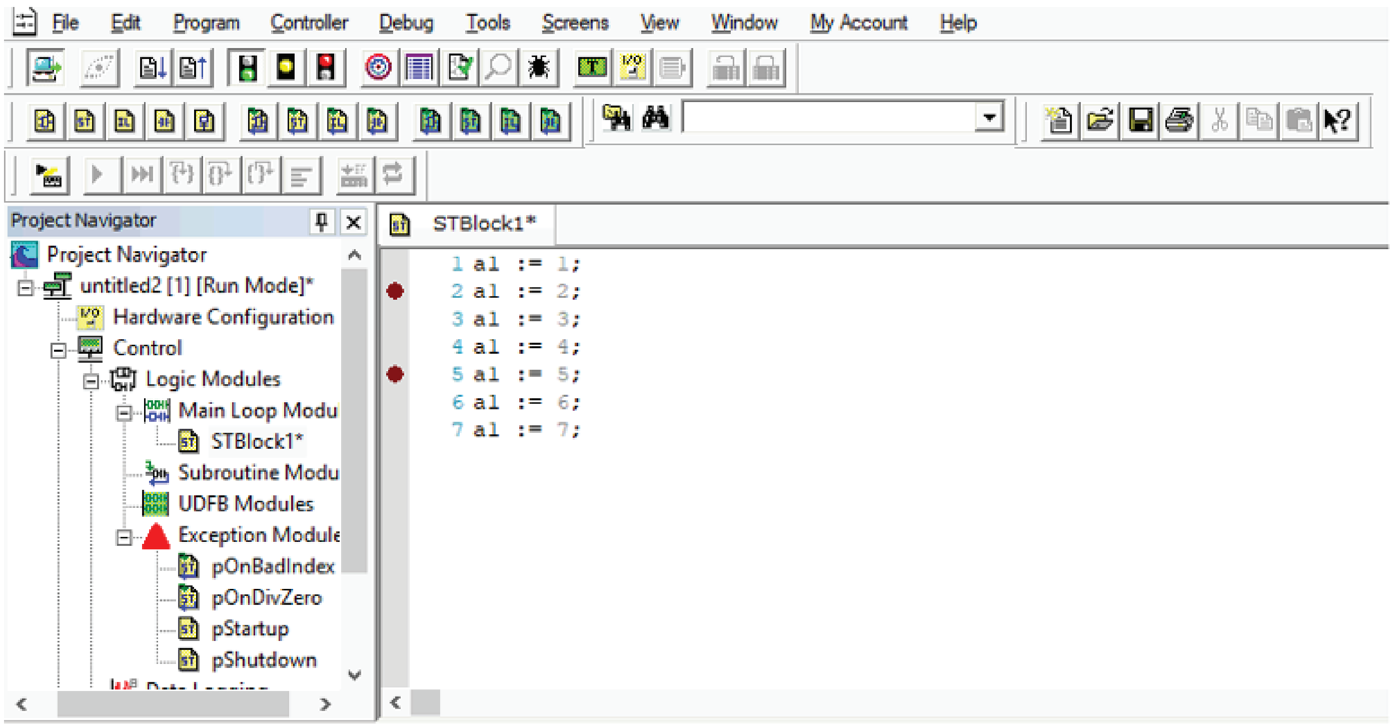
A breakpoint is an intentional stop marked in the code of an application where execution pauses for debugging. A breakpoint helps to speed up the debugging process in a large program by allowing the execution to continue up to a desired point before debugging begins.

Pressing **F9** will set / reset break points on Editor with red circle mark as shown below.

NOTE: Pressing **F9** will not set / reset breakpoints on Ladder Language of Enhanced IEC and this is a limitation.

If the current position is not on a valid line for stepping, the breakpoint is automatically moved to the nearest valid position. Breakpoints can be placed in programs, sub-programs or UDFBs. They are not available in SFC programs.

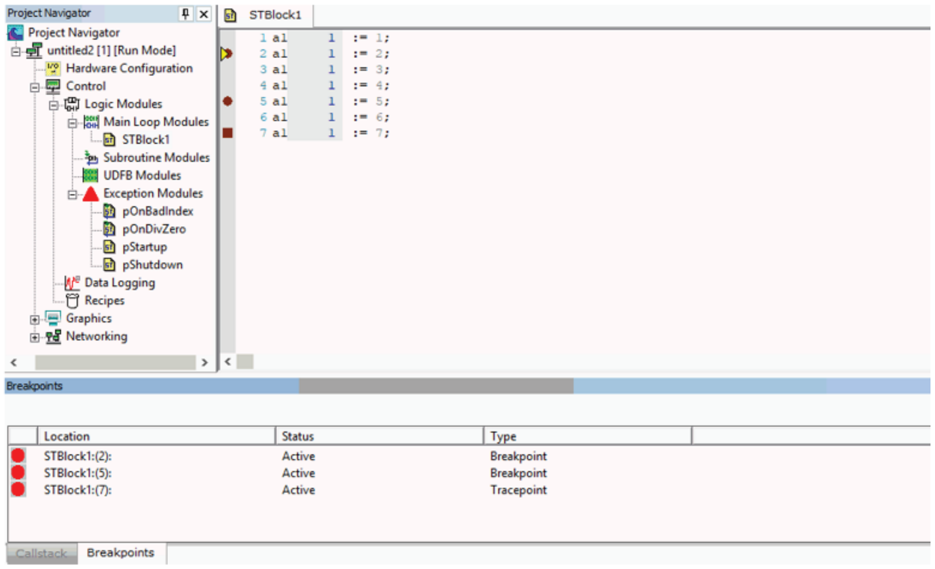
User can see the list of active breakpoints in the Log Window or in the output window of an editor. From here user have the possibility to remove all breakpoints in a single command. At any moment user can double click on a breakpoint in the Log window to open the corresponding program at the breakpoint location.



```
al := 1;
● al := 2;
al := 3;
al := 4;
● al := 5;
al := 6;
al := 7;
```

Breakpoints Window

When program is in Debug / Simulation mode and breakpoints set then it will hit break points with yellow tri-angle and user can see corresponding break / trace points. Only the point which hits the breakpoints will become as active breakpoints other display as Inactive.



```
1 al      1  := 1;
2 al      1  := 2;
3 al      1  := 3;
4 al      1  := 4;
5 al      1  := 5;
6 al      1  := 6;
7 al      1  := 7;
```

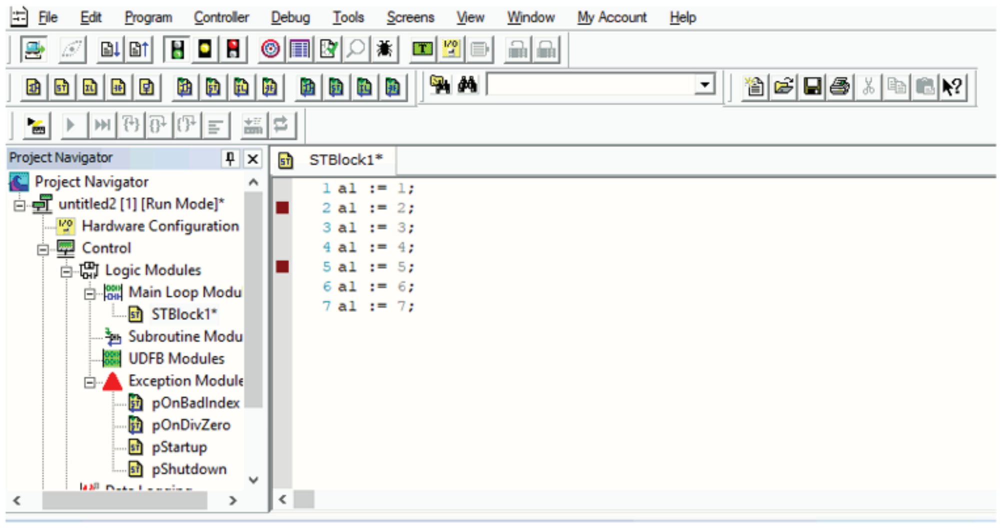
	Location	Status	Type
●	STBlock1:(2):	Active	Breakpoint
●	STBlock1:(5):	Active	Breakpoint
●	STBlock1:(7):	Active	Tracepoint

Tracepoints

Tracepoints are similar to breakpoints but rather than stopping the PLC scan, a message will be added in the runtime window at the time of execution. When a tracepoint is hit, the debugger performs the specified trace-point action instead of, or in addition to, breaking program execution.

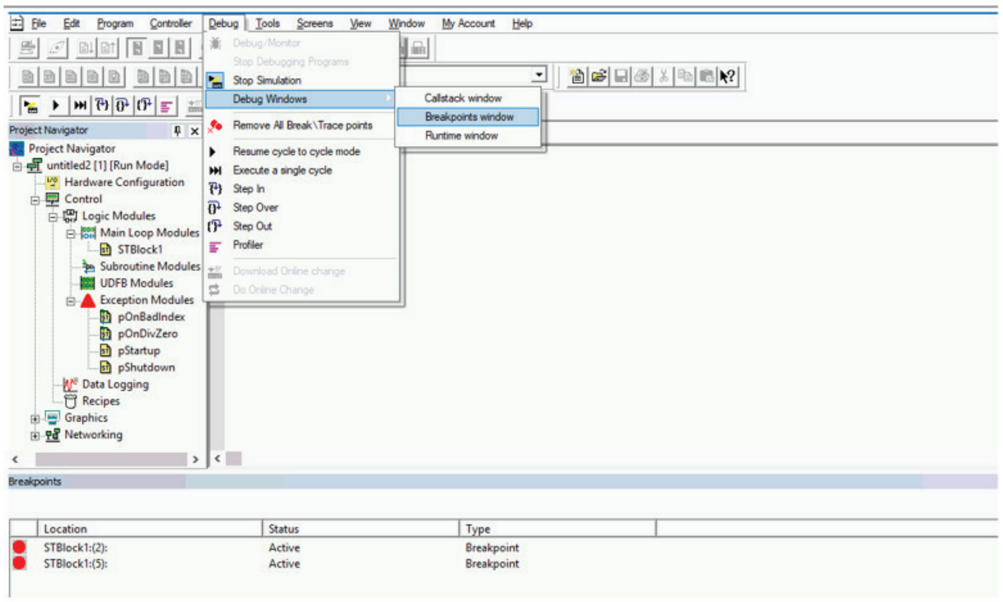
Pressing **F8** will set / reset trace points on Editor with red square mark as shown below.

NOTE: Pressing **F8** will not set / reset trace points on Ladder Language of Enhanced IEC and this is a limitation.



```
1 al := 1;
2 al := 2;
3 al := 3;
4 al := 4;
5 al := 5;
6 al := 6;
7 al := 7;
```

NOTE: User can check set breakpoints / tracepoints on the **Breakpoint Window**. Callstack and Runtime windows are used during debug / simulation for showing callstack and traced points.



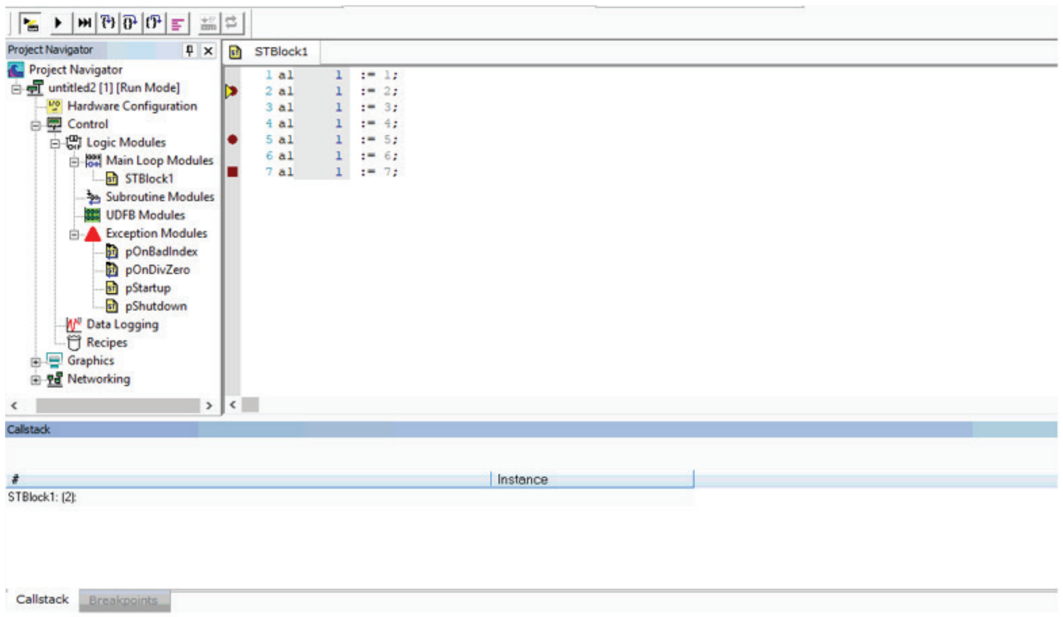
	Location	Status	Type
●	STBlock1:(2):	Active	Breakpoint
●	STBlock1:(5):	Active	Breakpoint

Call Stack Window

When program is in debug / simulation mode and breakpoints set then it will hit break points with yellow tri-angle and user can see corresponding call stack.

When stepping, the Call Stack is available in the Log Window or in the output window of editors. It shows the stack of called function blocks and/or sub-programs, from the top level calling program up to the current stepping position. For a UDFB, the Call Stack also indicates which instance of the UDFB is executed.

At any moment user can double click on a line of the Call Stack to open the corresponding program

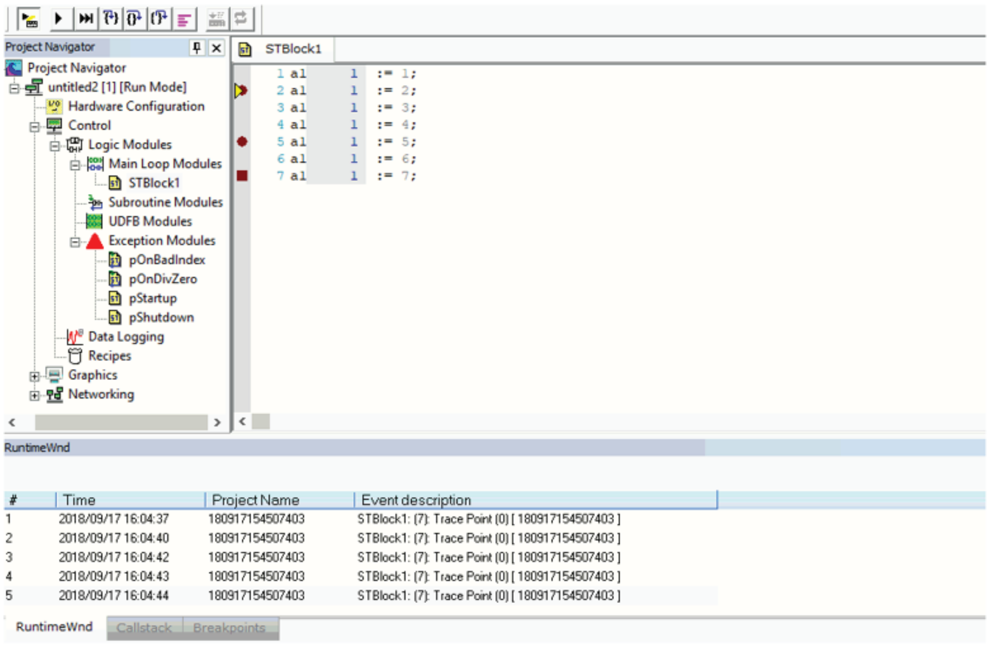


```
1 al      1   := 1;
2 al      1   := 2;
3 al      1   := 3;
4 al      1   := 4;
5 al      1   := 5;
6 al      1   := 6;
7 al      1   := 7;
```

#	Instance
STBlock1: (2):	

Runtime Window

When program is in debug / simulation mode and trace points set then it will hit trace points and user can see corresponding traced points.



```
1 al      1   := 1;
2 al      1   := 2;
3 al      1   := 3;
4 al      1   := 4;
5 al      1   := 5;
6 al      1   := 6;
7 al      1   := 7;
```

```
#   Time                  Project Name       Event description
1   2018/09/17 16:04:37   180917154507403    STBlock1: (7): Trace Point (0) [ 180917154507403 ]
2   2018/09/17 16:04:40   180917154507403    STBlock1: (7): Trace Point (0) [ 180917154507403 ]
3   2018/09/17 16:04:42   180917154507403    STBlock1: (7): Trace Point (0) [ 180917154507403 ]
4   2018/09/17 16:04:43   180917154507403    STBlock1: (7): Trace Point (0) [ 180917154507403 ]
5   2018/09/17 16:04:44   180917154507403    STBlock1: (7): Trace Point (0) [ 180917154507403 ]
```

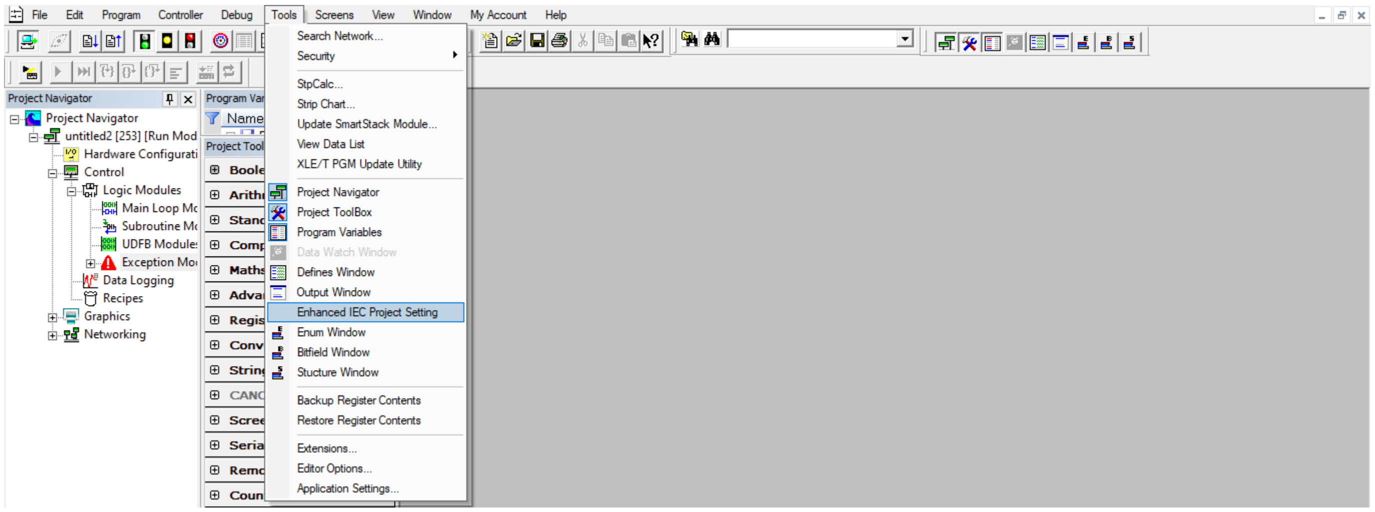
"Enhanced IEC Settings and Data Types" on page 2 | "Debugger Windows " on page 7 | "Online Change " on next page | "Simulation " on page 19 | "Stepping " on page 20

Online Change

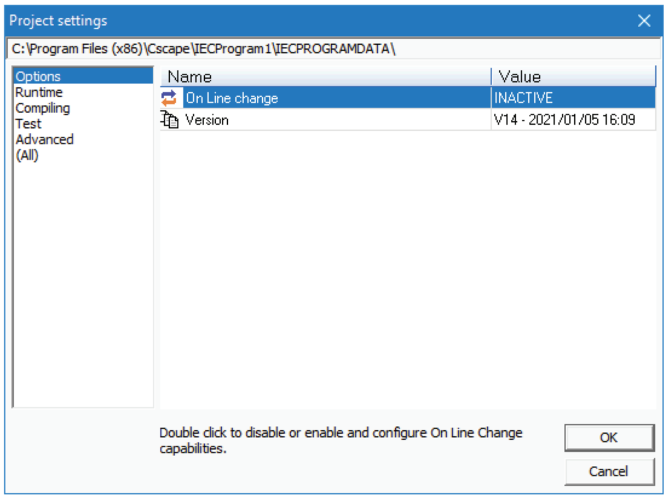
On Line change enables user to change the application "on the fly" while it is running in the target system. In order to enable most kinds of changes, user needs to configure the programming environment and enable the On Line changes.

This is ideal for programs which require long periods of start up or shut down so that logic changes can be made without waiting for the machine to start or stop so that logic can be downloaded.

Online change option can be enabled from **Tools > Enhanced IEC Project Settings**.



Double clicking on the option displays the following window where user can Activate / Deactivate On Line Change functionality.

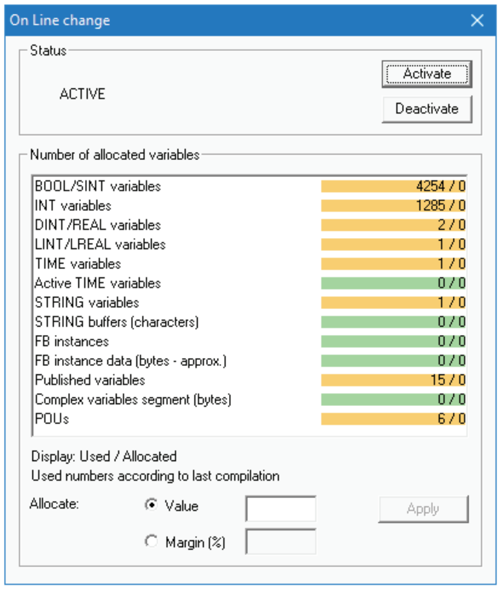

Project settings

C:\Program Files (x86)\Cscape\IECProgram1\IECPROGRAMDATA\

Options
Runtime
Compiling
Test
Advanced
(All)

Name	Value
On Line change	INACTIVE
Version	V14 - 2021/01/05 16:09

Double click to disable or enable and configure On Line Change capabilities.


On Line change

Status

ACTIVE

Number of allocated variables

BOOL/SINT variables	4254 / 0
INT variables	1285 / 0
DINT/REAL variables	2 / 0
LINT/LREAL variables	1 / 0
TIME variables	1 / 0
Active TIME variables	0 / 0
STRING variables	1 / 0
STRING buffers (characters)	0 / 0
FB instances	0 / 0
FB instance data (bytes - approx.)	0 / 0
Published variables	15 / 0
Complex variables segment (bytes)	0 / 0
POUs	6 / 0

Display: Used / Allocated
Used numbers according to last compilation

Allocate: [✓ Value] Margin (%)

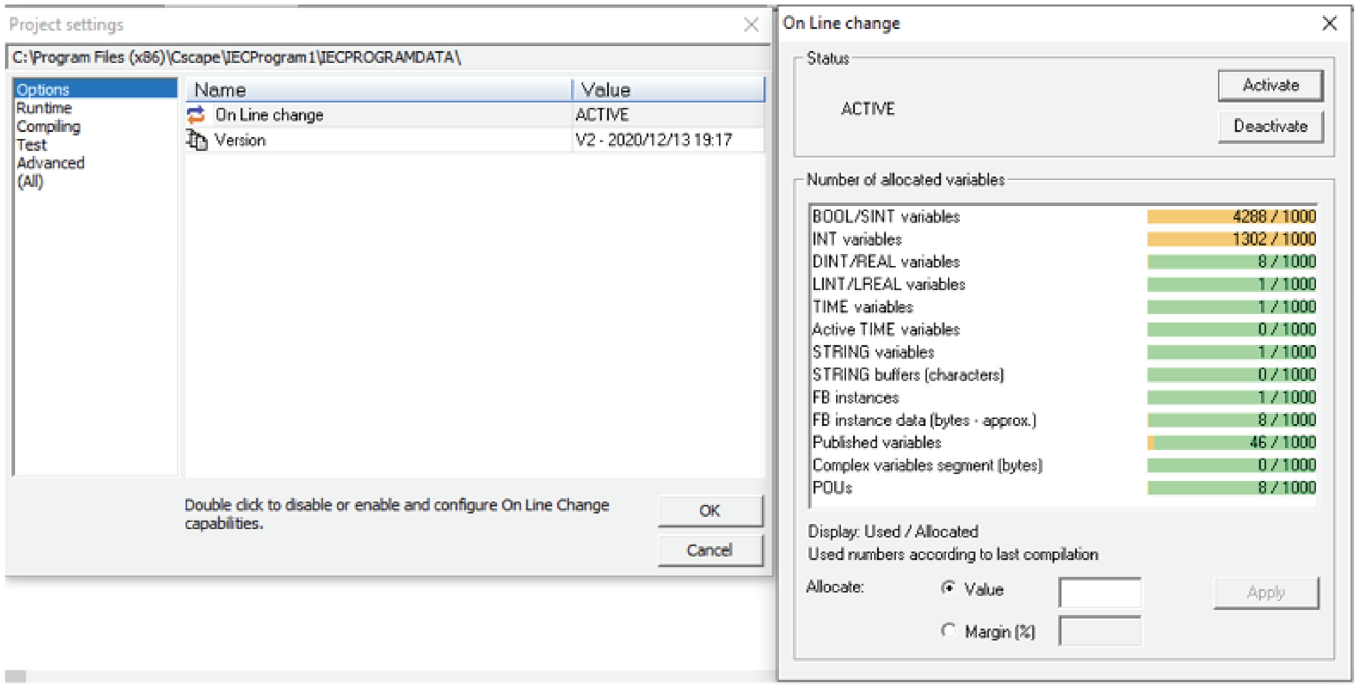
Selecting Options displays the following items:

To allow the declaration of new variables and blocks, the user has to define the amount of memory to be allocated in the target for each type of data. This includes:

1. The number of variables for each type (8, 16, 32 or 64 variables, character strings).
2. The number of function block instances.
3. The amount of memory for storing character strings.
4. The amount of memory for private data of function block instances.

The number of variables of each type used in the application is given as a report at the end of each build. Changes are allowed until user exceed the sizing for at least one type of data. At this time, if user need to declare new variables, user have to change the configuration of memory sizing, rebuild the application and perform a full download stopping the target application.

Select the required variable and give a value or % Margin as to how much is required to create online change. In the below screen shot, for each variable 1000 nos. memory is allocated.



Name	Value
On Line change	ACTIVE
Version	V2 - 2020/12/13 19:17

Double click to disable or enable and configure On Line Change capabilities.

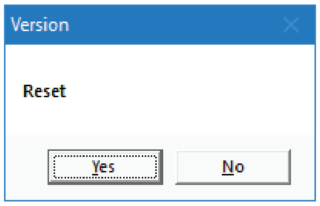
ACTIVE

BOOL/SINT variables	4288 / 1000
INT variables	1302 / 1000
DINT/REAL variables	8 / 1000
LINT/LREAL variables	1 / 1000
TIME variables	1 / 1000
Active TIME variables	0 / 1000
STRING variables	1 / 1000
STRING buffers (characters)	0 / 1000
FB instances	1 / 1000
FB instance data (bytes - approx.)	8 / 1000
Published variables	46 / 1000
Complex variables segment (bytes)	0 / 1000
POUs	8 / 1000

Display: Used / Allocated
Used numbers according to last compilation

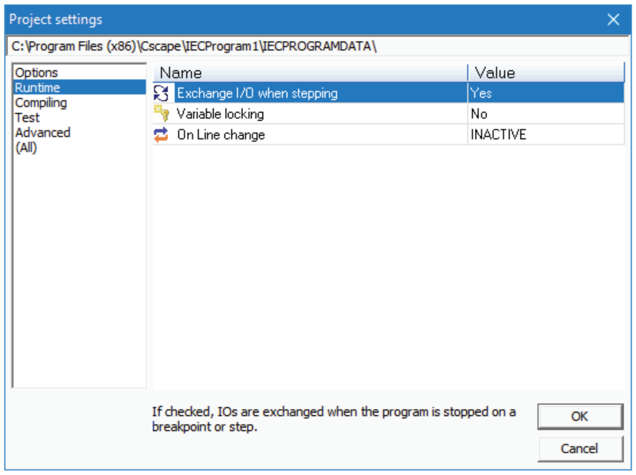
Reset the Version

Doubling clicking on the option displays the following window so that user can reset the version.



Runtime

Selecting Runtime option displays the following items:



C:\Program Files (x86)\Cscape\IECProgram1\IECPROGRAMDATA\

Name	Value
Exchange I/O when stepping	Yes
Variable locking	No
On Line change	INACTIVE

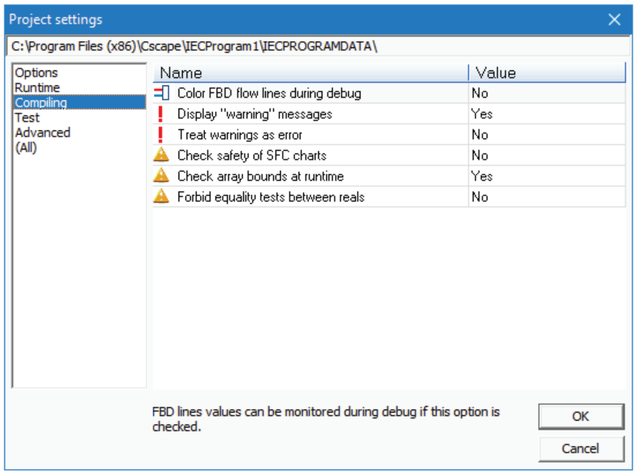
If checked, IOs are exchanged when the program is stopped on a breakpoint or step.

Exchange I/O when stepping - User can change the value to Yes/No by double click. If this option is selected as Yes, IO's are exchanged when the program is stopped on a breakpoint or step.

Variable locking - User can change the value to Yes/No by double click. Locked variables can be forced to a fixed value from the debugger. This option would be time consuming. This option is not available when On Line Change is in active mode.

Compiling

Selecting the Compiling option displays the following items:



C:\Program Files (x86)\Cscape\IECProgram1\IECPROGRAMDATA\

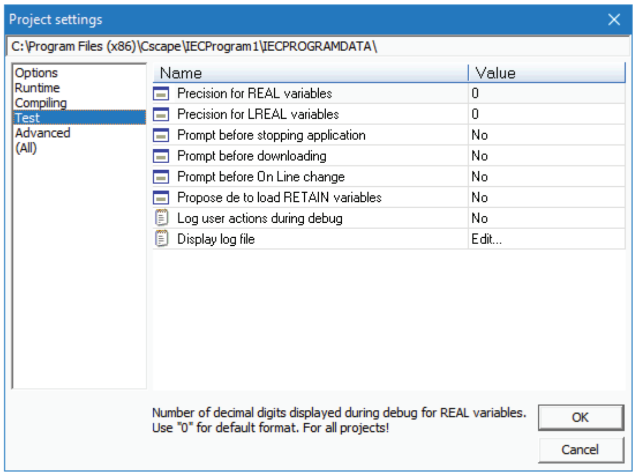
Name	Value
Color FBD flow lines during debug	No
Display "warning" messages	Yes
Treat warnings as error	No
Check safety of SFC charts	No
Check array bounds at runtime	Yes
Forbid equality tests between reals	No

FBD lines values can be monitored during debug if this option is checked.

- **Color FBD flow lines during debug** - User can change the value to Yes/No by double click. FBD lines values can be monitored during debug if this option is set to yes.
- **Display "warning" messages** - User can change the value to Yes/No by double click. Reports dangerous or unclear operations at compiling time if this option is set to yes.
- **Treat warnings as error** - User can change the value to Yes/No by double click. Setting this option to yes produces an error if warnings are reported.
- **Check safety of SFC charts** - User can change the value to Yes/No by double click. Setting this option to Yes checks possible unsafe or blocking structures in SFC charts at compiling time.
- **Check array bounds at runtime** - User can change the value to Yes/No by double click. Setting this option to Yes checks indexes used for array read/write operations at runtime.
- **Forbid equality tests between reals** - User can change the value to Yes/No by double click. Errors will be output if equality tests are performed between REALs or LREALs, if this option is set to yes.

Test

Selecting Test option displays the following items.



Project settings

C:\Program Files (x86)\Cscape\IECProgram1\IECPROGRAMDATA\

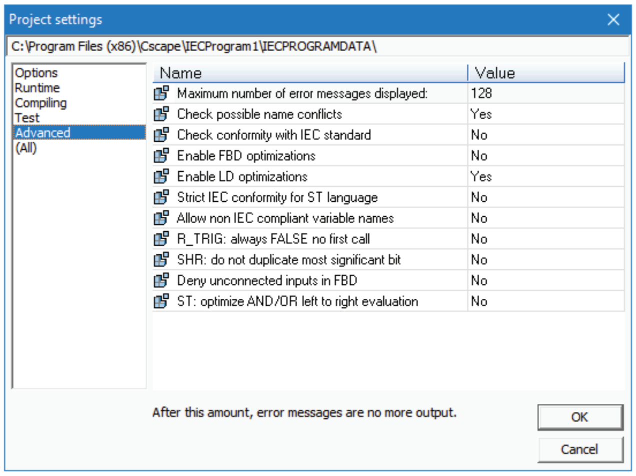
Name	Value
Precision for REAL variables	0
Precision for LREAL variables	0
Prompt before stopping application	No
Prompt before downloading	No
Prompt before On Line change	No
Propose de to load RETAIN variables	No
Log user actions during debug	No
Display log file	Edit...

Number of decimal digits displayed during debug for REAL variables. Use "0" for default format. For all projects!

- **Precision for REAL variables** - Using this option user can set number of decimal digits that needs to be displayed during debug for REAL variables. Use "0" for default format.
- **Precision for LREAL variables** - Using this option user can set number of decimal digits that needs to be displayed during debug for LREAL variables. Use "0" for default format.
- **Prompt before stopping application** - User can change the value to Yes/No by double click. Setting this option to Yes, displays a confirmation window before a stop operation.
- **Prompt before downloading** - User can change the value to Yes/No by double click. Setting this option to Yes, displays a confirmation window before a download operation.
- **Prompt before On Line Change** - User can change the value to Yes/No by double click. Setting this option to Yes, displays a confirmation window before an online change operation.
- **Propose to load RETAIN variables** - User can change the value to Yes/No by double click. Setting this option to Yes, starts the runtime with reloading RETAIN variables as a default choice.
- **Log user actions during debug** - User can change the value to Yes/No by double click. Setting this option to Yes, logs the user operations performed during simulation or On Line test.
- **Display log file** - Double click on this option to open the log file.

Advanced

Selecting advanced option displays the following items.



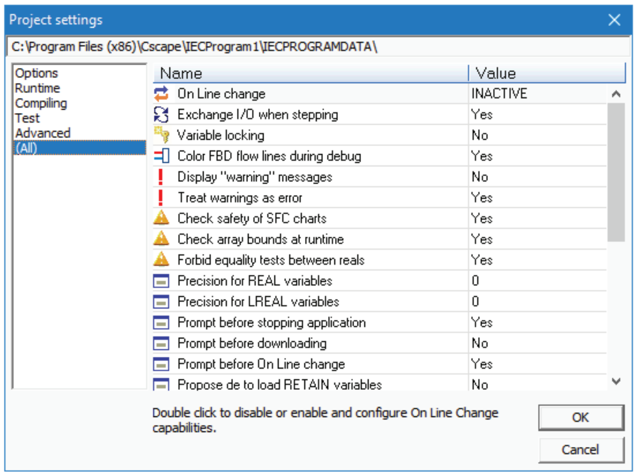
C:\Program Files (x86)\Cscape\IECProgram1\IECPROGRAMDATA\

Name	Value
Maximum number of error messages displayed:	128
Check possible name conflicts	Yes
Check conformity with IEC standard	No
Enable FBD optimizations	No
Enable LD optimizations	Yes
Strict IEC conformity for ST language	No
Allow non IEC compliant variable names	No
R_TRIG: always FALSE no first call	No
SHR: do not duplicate most significant bit	No
Deny unconnected inputs in FBD	No
ST: optimize AND/OR left to right evaluation	No

After this amount, error messages are no more output.

- **Maximum number of error message displayed** - Using this option user can set maximum number of error messages that needs to be displayed after which the error messages will not be displayed in output window.
- **Check possible name conflicts** - User can change the value to Yes/No by double click. Setting this option to Yes, reports warnings if the same symbol is used for many purposes.
- **Check conformity with IEC standard** - User can change the value to Yes/No by double click. Setting this option to Yes, reports warnings if non IEC compliant features or blocks are used.
- **Enable FBD optimizations** - User can change the value to Yes/No by double click. Setting this option to Yes, produces smaller and faster FBD code.
- **Enable LD optimizations** - User can change the value to Yes/No by double click. Setting this option to Yes, produces smaller and faster LD code.
- **Strict IEC conformity for ST language** - User can change the value to Yes/No by double click. If this option is set to No, equality comparison of TIME variables will be forbidden.
- **Allow non IEC complaint variable names** - User can change the value to Yes/No by double click. This option is not recommended to be set to Yes as some limitations may occur in the editors.
- **R_TRIG: always false on first call** - User can change the value to Yes/No by double click. Setting this option to Yes, makes R_TRIG output as always FALSE on the first call to an instance.
- **SHR : do not duplicate most significant bit** - User can change the value to Yes/No by double click. Setting this option to Yes, the most significant bit of a word shift to the right is set to FALSE.
- **Deny unconnected inputs in FBD** - User can change the value to Yes/No by double click. Setting this option to Yes, errors will be generated when some block inputs are not wired in FBD diagrams.
- **ST: optimize AND/OR left to right evaluation** - User can change the value to Yes/No by double click. Setting this option to Yes, the evaluation of AND/OR expressions from left to right will be stopped as soon as the result is known.

All
Selecting this option will displays all the items that are available in Enhanced IEC project settings.


Project settings

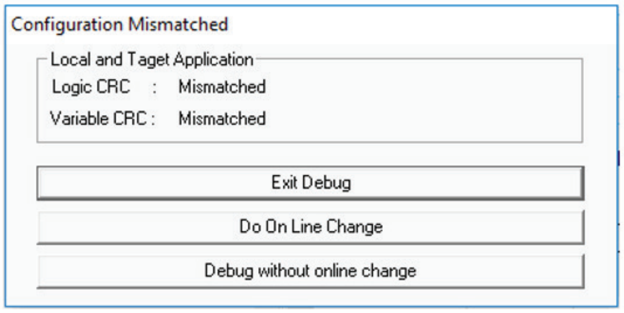
C:\Program Files (x86)\Cscape\IECProgram1\IECPROGRAMDATA\

Name	Value
On Line change	INACTIVE
Exchange I/O when stepping	Yes
Variable locking	No
Color FBD flow lines during debug	Yes
Display "warning" messages	No
Treat warnings as error	Yes
Check safety of SFC charts	Yes
Check array bounds at runtime	Yes
Forbid equality tests between reals	Yes
Precision for REAL variables	0
Precision for LREAL variables	0
Prompt before stopping application	Yes
Prompt before downloading	No
Prompt before On Line change	Yes
Propose de to load RETAIN variables	No

Double click to disable or enable and configure On Line Change capabilities.

Download the Program
Download the program (which has online change enabled) into the controller.

To download any modified program to the unit, go to **Debug Mode.** The following window appears:


Configuration Mismatched

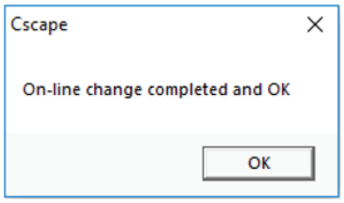
Local and Taget Application

Logic CRC : Mismatched

Variable CRC : Mismatched

- **Exit Debug** - Comes out of debug.
- **Do On Line Change** - downloads the updated application and applies the change to the unit.
- **Debug without online change** - Enters debugging but no change will be done in the unit.

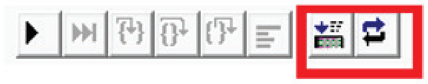
If user selects "Do On Line Change", Cscape will download the modified application to the unit and applies the change to the unit. A confirmation message of On Line change completed will be displayed after successfully modifying the changes as shown below:


Cscape

On-line change completed and OK

Online Change Menu

The following options can be used to separately download modified program and to apply changes to the unit.



Download Changes

This option is used to download the changes to the unit.

Do Online Changes

This option is used to apply the changes to the unit.

When **Online Change** is enabled, all newly created variables are marked in blue in the variable editor. Deleted variables are kept as ghosts, marked in red and renamed with a "_del_" prefix. Users can manually rename a ghost variable if they want to make it alive again.

NOTE: When On Line change is enabled, user can perform following changes on the fly.

1. Change the code of a program.
2. Change the condition of a SFC transition or the actions of a SFC step.
3. Create, rename or delete global and local variables.
4. Create, rename or delete global and local function block instances.

Limitations of Online Change

1. Create, delete or rename a variable.
2. Changing SFC charts
3. Changing the local parameters and variables of a UDFB.
4. Changing the type or dimension (or string length) of a variable or function block instance.
5. Changing the set of I/O boards
6. Changing the definition of RETAIN variables
7. The Embedded HMI data (screens, bitmaps, fonts, strings) are not updated during an On Line Change.

Unsafe During Online Change

The following options can be used to separately download modified program and to apply changes to the unit.

1. **Pulse (P or N) contacts and coils** (edge detection). Instead, user must use declared instances of R_TRIG and F_TRIG function blocks.
2. **Loops in FBD** with no declared variable linked. User need to explicitly insert a variable in the loop.

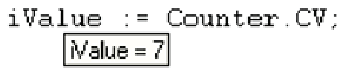
On Line Change will be denied by the runtime if some system resources are currently open when the change is performed. A system resource can be either a file open by file management functions, or a socket open by TCP/UDP functions, or an dynamic array created by ARCREATE functions.

"Enhanced IEC Settings and Data Types" on page 2 | "Debugger Windows " on page 7 | "Online Change " on page 11 | "Simulation " on next page | "Stepping " on page 20

Simulation

The Simulator function is similar to Debugging, but logic is evaluated within Cscape, and OCS is not needed. **Breakpoints** and **Single-Stepping Debugging** facilities are all supported in the Simulation Mode.

During test (connected mode or simulation) of the program the ST / IL editor shows in a Tool tip the current value of the variable pointed to by the mouse cursor without a special command by the user. Simply hover the cursor on the variable symbol and wait for one second to open the Tool tip. See example below:



```
iValue := Counter.CV;
    iValue = 7
```

The value shown in the tool tip is automatically refreshed while the tool tip is open.

NOTE: Function blocks that have hardware dependency like Networking Blocks, Serial Ports, etc., cannot be simulated.

"Enhanced IEC Settings and Data Types" on page 2 | "Debugger Windows " on page 7 | "Online Change " on page 11 | "Simulation " on page 19 | "Stepping " on next page

Stepping

In addition to the cycle-by-cycle execution mode, the debugger has a rich collection of powerful features for making step by step debugging in the source code of user's application. The step-by-step feature is completed by the possibility to place breakpoints in the source code of the application. The debugger also shows the call stack of called UDFBs and sub-programs when stepping. Step by step debugging is available:

- **In ST and IL text programs** (a step is a statement).
- **In LD program** (a step is a rung).
- **In FBD** (a step is a graphic symbol corresponding to an action).

Entering the step-by-step debugging mode:

- **Place a breakpoint in a program**. When the breakpoint is reached, the execution stops at the specified location and User can step further in the program.

The following commands are available for stepping:

Step In

Similar to single step but if any subroutines / UDFBs need to be executed to get to the next function block the step will move inside the function block, i.e., if the line contains a call to a subroutine or UDFB then that sub-routine / UDFB is opened and a break occurs at the first line of the subroutine / UDFB. If there is no subrou-tine / UDFB on the line, then the debugger steps to the next line.

Step Over

If the next instruction is a call of a function block or a sub-program, the execution continues up to the end of the called block.

Step Out

Will complete execution of the current subroutine / UDFB and then break at the next line in the calling code, i.e., If currently debugging in a subroutine or UDFB then the remainder of that module is executed and exe-cution returns to the calling module. This assumes no further breakpoints are hit in the remainder of the sub-routine / UDFB.

In addition to these commands, user can at any time:

- **Execute the cycle** (from the current position up to the end of the last program).
- **Restart the target** in normal execution mode (RUN).

Apart from the above-mentioned Stepping Functions, these are also supported:

Resume Cycle-to-Cycle Mode

▶

Resumes Execution

Execute a Single Cycle

⏭

Executes the next line of code/rung/function block.

"Enhanced IEC Settings and Data Types" on page 2 | "Debugger Windows " on page 7 | "On Line Change " on page 11 | "Simulation " on page 19 | "Stepping " on page 20