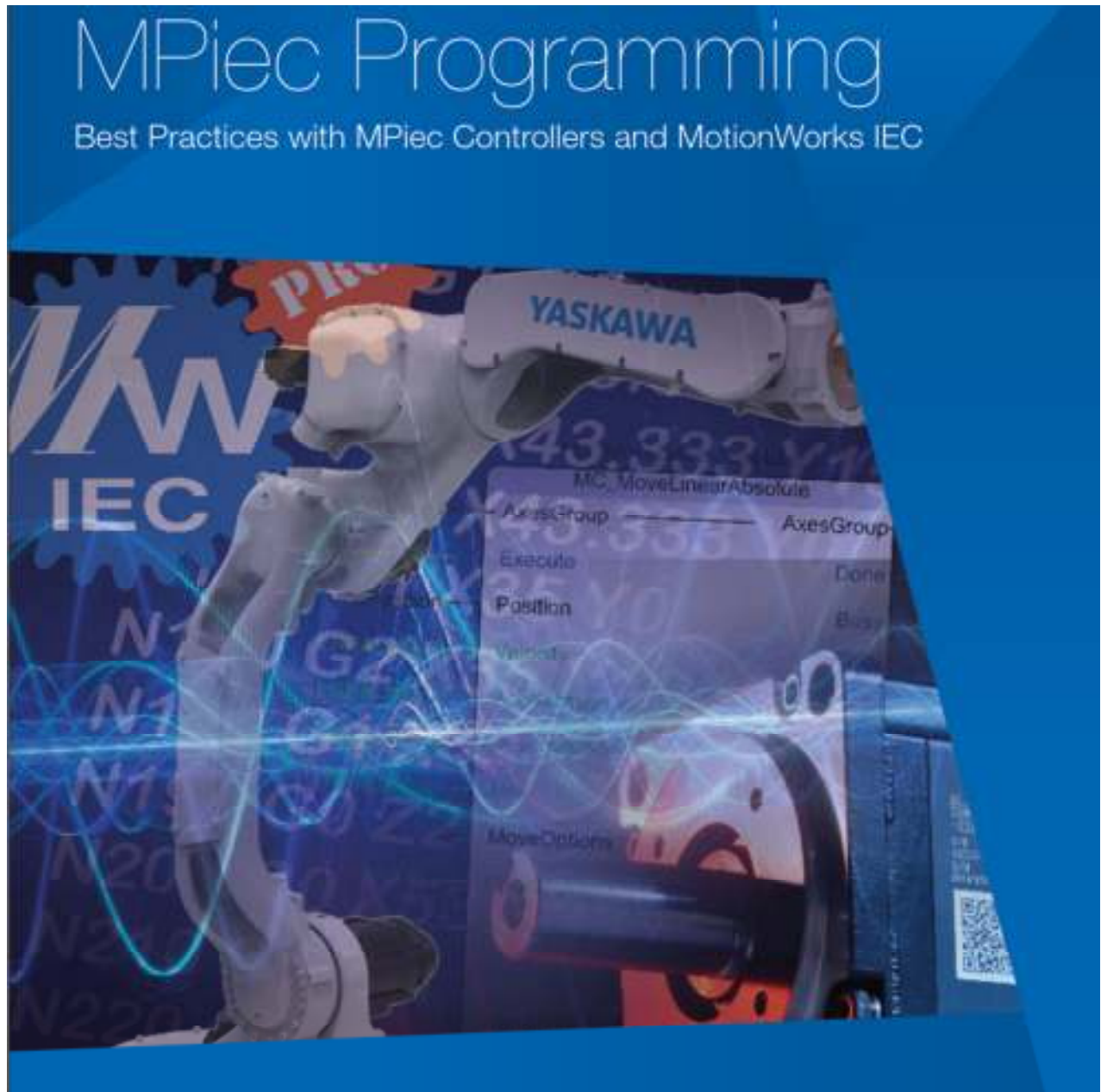


Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03



1	OVERVIEW	4
2	MOTIONWORKS IEC 3 APPLICATION CODE TOOLBOX INSTALLER	4
3	PROJECT TEMPLATES	4
4	INITIALIZATION	5
4.1	Single Element Initialization.....	5
4.2	Multi-Element Initialization.....	5
5	SIGNALS	6
5.1	Momentary	6
5.2	Latch.....	6
6	COMMUNICATION INTERFACE	7
7	RANGE LIMITS	8
8	POWER UP, ALARM MANAGEMENT, E-STOP RECOVERY.....	8
9	AXIS MONITORING AND STATUS	8
10	HOMING	9
11	MAINTENANCE AND UPGRADE.....	10
11.1	Amplifier Replacement:	10
11.2	Servomotor Replacement:.....	10
12	LOCAL VS. GLOBAL VARIABLES	11
13	TASK TIME MONITORING TO OPTIMIZE PROJECT	11
14	USER CREATED FUNCTION BLOCKS	12
15	FUNCTION BLOCK COPY AND PASTE	13
16	STRUCTURES FOR AXIS MOTION PARAMETERS	13
17	OPTIMIZATION	14
18	CODE INSPIRATIONALS AND FAQs.....	15
18.1	Implementing Enable logic for function blocks.....	15
18.2	Are there limits on POU's or variables in a project?	15
18.3	Write a coil at only one location in the program	16
18.4	Set / Reset combinations	16
18.5	Cascading ELSIF	17
18.6	WHILE Loop.....	17
18.7	Handling Errors	17
18.8	Rising Edge and Falling Edge Triggers	17
18.9	Comments.....	18
18.10	Sample Code Sections.....	18

Title: MPiec Programming Best Practices Guideline v3**Product(s):** MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec**Doc. No.** TN.MPIEC.03

18.11	What is the effect of using different power rails for ladder rungs?	19
18.12	The RETURN in MotionWorks IEC 3 programming.....	21
18.13	Rollover in MotionWorks IEC 3	23
18.14	Can variable descriptions be included in the editor window?	24
18.15	Best practice in Structured Text Programming	25
18.16	Best practice in Sequential Function Chart (SFC) programming	26
18.17	Best practice in Ladder Diagram (LD) programming	27

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

1 OVERVIEW

This document is a guide to implement best programming practices during the design and architecture of a MotionWorks IEC 3 project for the MPiec series controller. It is focused mainly on the MP3000iec, Sigma-7Siec and MP2600iec and for users who have completed necessary training.

2 MOTIONWORKS IEC 3 APPLICATION CODE TOOLBOX INSTALLER

Yaskawa recommends using the function blocks created and documented from the installer below to save code development time:

<http://www.yaskawa.com/iectb>

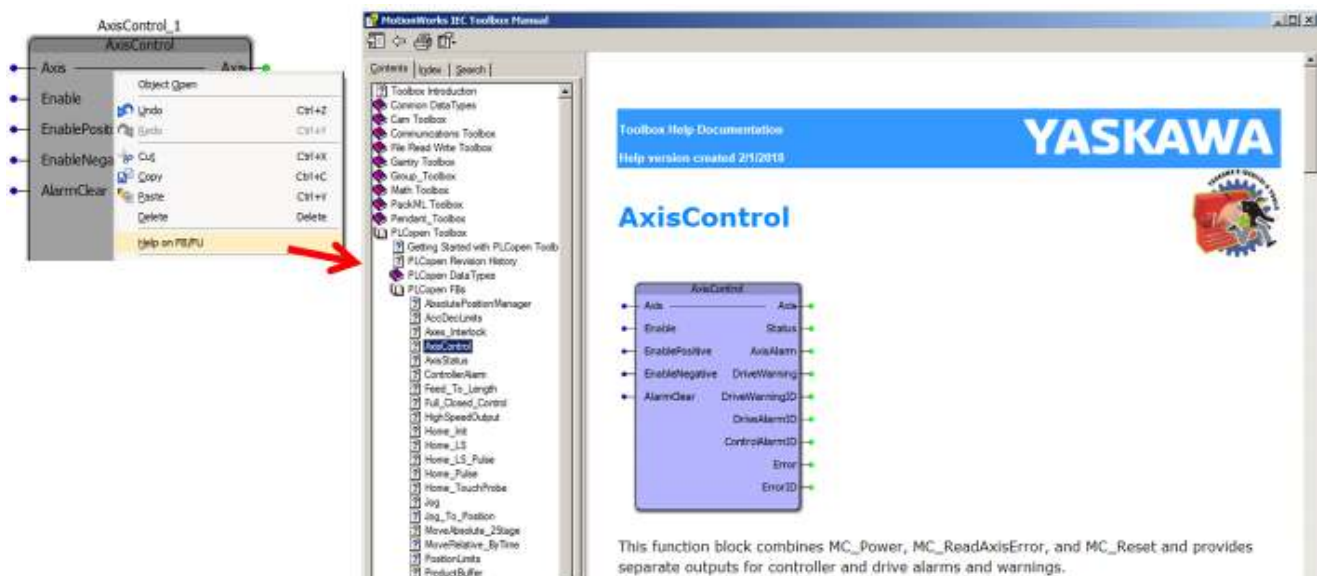


FIGURE 1 CHM HELP FILE

3 PROJECT TEMPLATES

The user can choose between a standard MotionWorks IEC 3 template or the MP2600iec Project Template as reference for getting started with a project:

** Link for Full MP2600iec Template: Here **

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

4 INITIALIZATION

4.1 Single Element Initialization

If a variable is not written every time before it is used, or written every scan, then it likely needs to be initialized. Text is the best way to initialize.

```
(*Data for Registration based Rotary Knife*)

(*ProductBufferStruct for Registration Data *)
(*=====*)
Products.BufferSize           := INT#20;
Products.LockoutDistance     := LREAL#4.0;
Products.SensorDistance      := LREAL#23.25;
Products.ProductAwayDistance := LREAL#23.25;

Products.Sensor.Bit           := UINT#1;
```

FIGURE 2 ONE TIME INITIALIZE

Variables being used in a FOR loop should be initialized right before entering the loop.

```
PairTotal:=UDINT#1; (* Initialized to one to start off the first pair at 0,0 *)
LastSegment:=CamData.LastSegment;
FOR Segment:=1 TO LastSegment DO
  PrevSegment:=Segment - INT#1;
```

FIGURE 3 INITIALIZE VARIABLE BEFORE ENTERING FOR LOOP

4.2 Multi-Element Initialization

For complex user-defined data types, such as structures or arrays, there is a special syntax required, as several elements are to be initialized. The multi-element variable window simplifies this handling by entering the desired initial values for the STRUCT or ARRAY elements in the 'Init' column of the multi-element variable window.

Name	Type	Description	Init. value
HomeDataStruct	HomeStruct		(Direction := 1, SwitchMode := 1, TorqueLimit := 20.0, ApproachVelocity := 15.0, ApproachTimeLimit := 40.0, Ap
Direction	INT	Used in conjunction with MC_StepU...	1
SwitchMode	INT	Configuration for action of the home...	1
TorqueLimit	LREAL	Default if unused [ZERO] is 300.00...	20.0
ApproachVelocity	LREAL	In user units / second	15.0
ApproachTimeLimit	LREAL	In seconds	40.0
ApproachDistanceLimit	LREAL	In user units	65.0
AccDec	LREAL	In user units / second 2	150.0
LimitBackOffDistance	LREAL	In user units. Added in v021 for HD...	24.5
CreepVelocity	LREAL	In user units / second	5.0
CreepTimeLimit	LREAL	In seconds	15.0
CreepDistanceLimit	LREAL	In user units	10.0
Offset	LREAL	Position offset to MOVE after findi...	24.5
OffsetVelocity	LREAL	In user units / second	2.0
Position	LREAL	This is the location that will be defin...	0.0
Homed	BOOL	Flag to indicate that the axis was su...	

FIGURE 4 MULTI-ELEMENT INITIALIZATION

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

5 SIGNALS

5.1 Momentary

All command bits to the controller should be programmed as latched momentary signals. This prevents motion from restarting unintentionally in the event of E-Stop or power off conditions.

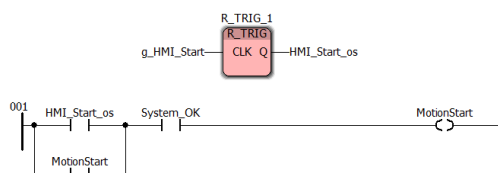


FIGURE 5 RISING EDGE SIGNAL FROM HMI

5.2 Latch

It is recommended to have the execute input latched with output bits of the same function block to demonstrate the status of that function block's impact on the axis. If MC_Stop has an error or aborts, the ErrorID will be shown since the Execute bit is latched. To clear this alarm, toggle the normally closed 'AlarmClear' bit.

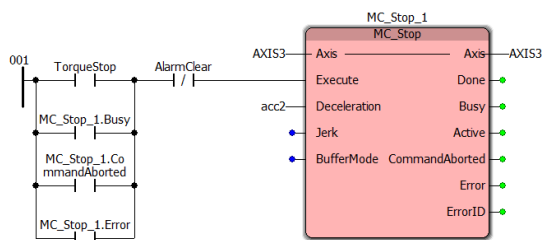


FIGURE 6 LATCHING THE 'EXECUTE' BIT

Use seal in circuits for latching when available.

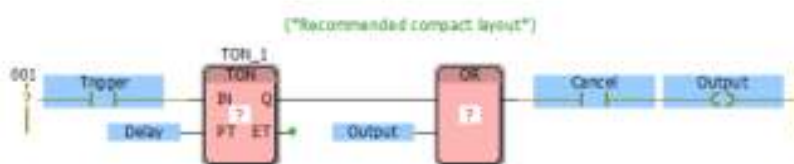


FIGURE 7 SEAL IN THE 'OUTPUT' BIT

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

6 COMMUNICATION INTERFACE

A POU for communication or network variable interfacing is very important to help trouble shoot, test and install communication between the MPiec controller and the network device. With such an interface POU in place, testing for proper communication can be done in a centralized location. Shown below is an example of transferring HMI variables to global variables that can be used in the project.

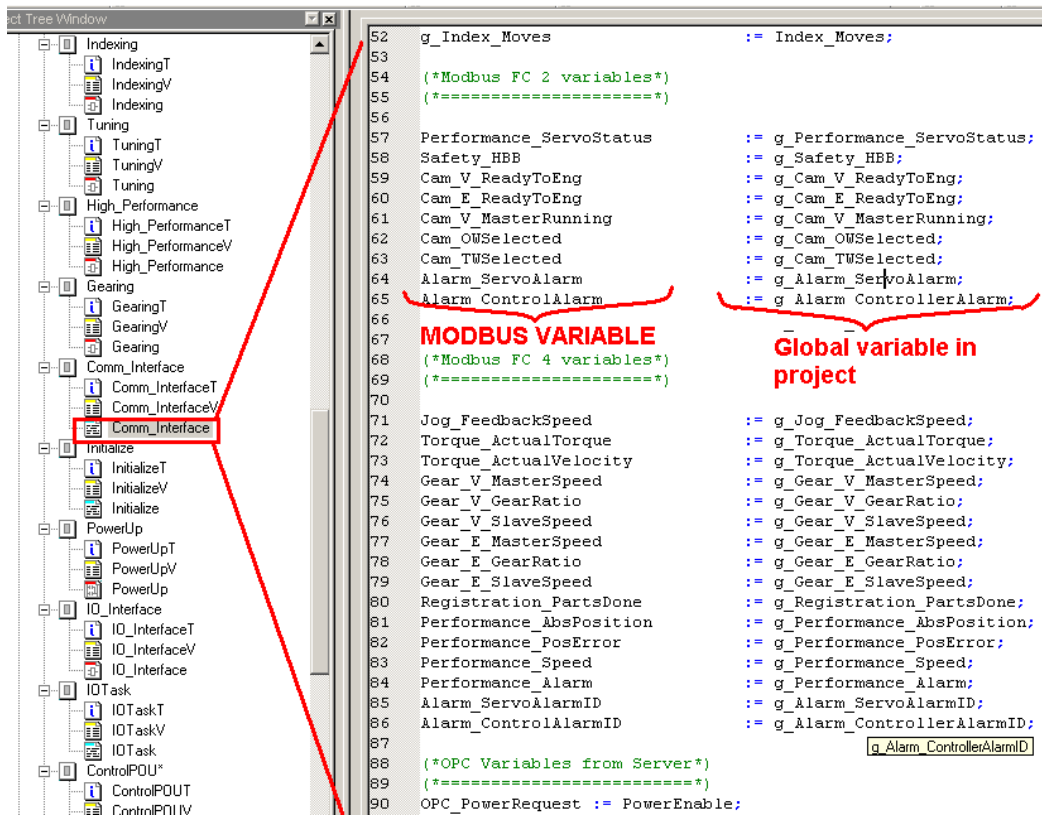


FIGURE 8 COMMUNICATION INTERFACE POU

Title: MPiec Programming Best Practices Guideline v3**Product(s):** MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec**Doc. No.** TN.MPIEC.03

7 RANGE LIMITS

Limit ranges of motion parameters like position, velocity, acceleration that are commanded from an HMI within the program to limit user errors causing damage to the system.

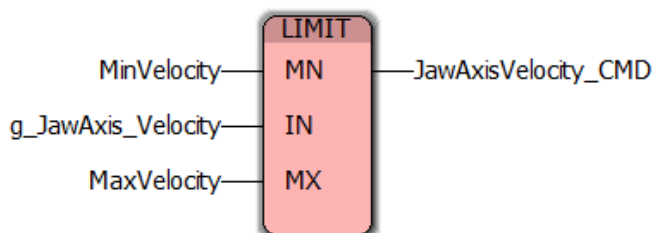


FIGURE 9 SAFETY LIMITS ON USER INPUT VARIABLES

8 POWER UP, ALARM MANAGEMENT, E-STOP RECOVERY

Power Up:

Enable multiple axes in the following way to prevent the MPiec controller from watchdogs:

- In a slow task
- In a staggering fashion with a scan delay or timer delay

Alarm clearing:

Alarms or errors can be cleared through the AlarmClear input on the Axis Control function block.

E-Stop Recovery

It is recommended that an E-Stop recovery (either using HBB or using main power on the servopack) be carried out by clearing all alarms on controller and all axes, then clearing HBB or servopack main power loss conditions

9 AXIS MONITORING AND STATUS

The ReadAxisParameters function block from the PLCopen Toolbox lists all the axis parameters in a structure format. This block does take up a lot of resources, but can be a quick way to access all parameters should the need arise for logic analyzing or in the watch window.

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

10 HOMING

There are four PLCopen specified homing methods currently supported by MotionWorks IEC 3

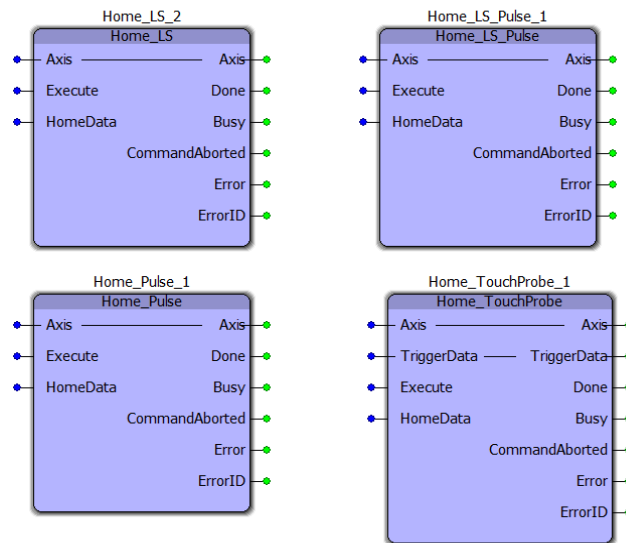


FIGURE 10 FOUR HOMING METHODS

The following shows an example of interlocks to include for homing:

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

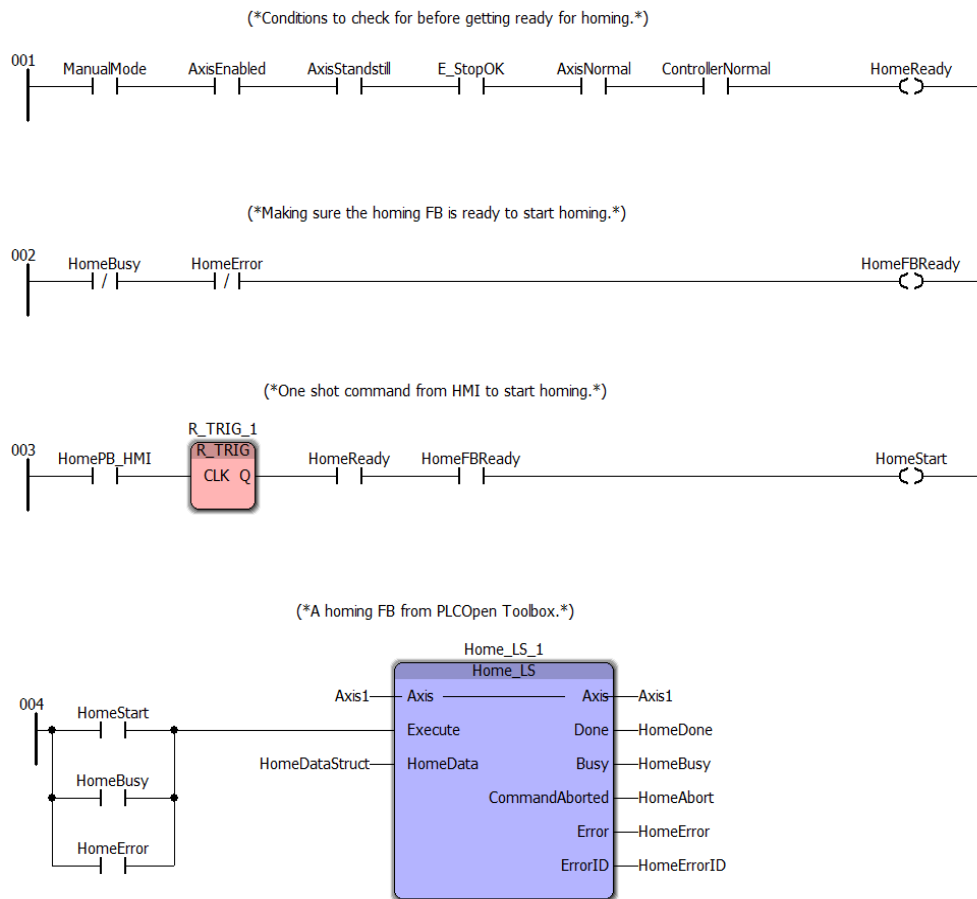


FIGURE 11 HOME LOGIC

11 MAINTENANCE AND UPGRADE

11.1 Amplifier Replacement:

[PD.MP2000iec.10.105](#) – This document explains how to replace an amplifier

11.2 Servomotor Replacement:

[Elv.MP2000iec.01.Maint-MtrRplce](#) – This e-learning video explains how to replace a servomotor

Title: MPiec Programming Best Practices Guideline v3**Product(s):** MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec**Doc. No.** TN.MPIEC.03

12 LOCAL VS. GLOBAL VARIABLES

If a variable does not have to be a global variable keep it local. It is recommended to keep the number of global variables to the bare minimum. One may use a distinguishing feature when naming global variables. (eg: G_newvariable)

Assign a group for every new global variable in the global variable worksheet.

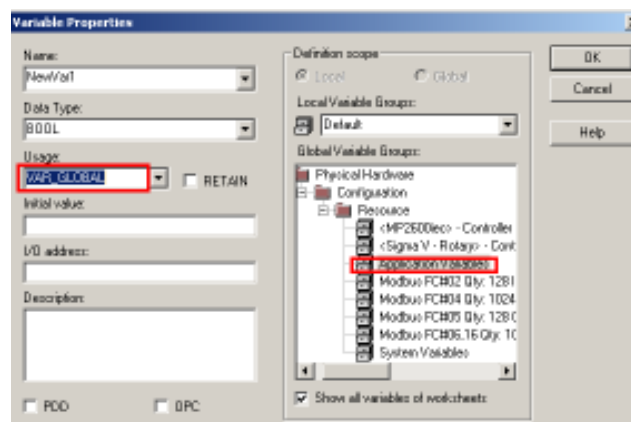


FIGURE 12 VARIABLE DECLARATION

13 TASK TIME MONITORING TO OPTIMIZE PROJECT

For the MP3000iec, MP2600iec and Sigma-7Siec series controllers, the task interval can be monitored with microsecond precision using variables in the global variable list.

MaxDuration_us is the maximum time that particular task took to execute. CurDuration is the time that task is taking to run in real time. These variables can be used to decide on whether a task requires to be given more time to run or if some other non urgent tasks can be moved to lower priority levels.

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

Variable	Value	De
PLC_TASK_4		
TaskStack	0	
TaskPrio	0	
TaskPeriod_us	10000	
TaskWatchdog_u	10000	
TaskPeriod	10	
TaskWatchdog	10	
MinDuration_us	399	
MaxDuration_us	2417	
CurDuration_us	483	
MinDelav us	0	

FIGURE 13 TASK TIME REAL TIME MONITORING FOR TASK 4

14 USER CREATED FUNCTION BLOCKS

User-defined function blocks should be stored in a separate library that can be reused. General rules for inputs and outputs for function blocks can be obtained at

www.plcopen.org

Always have comments within the code about the functionality of user defined function blocks.

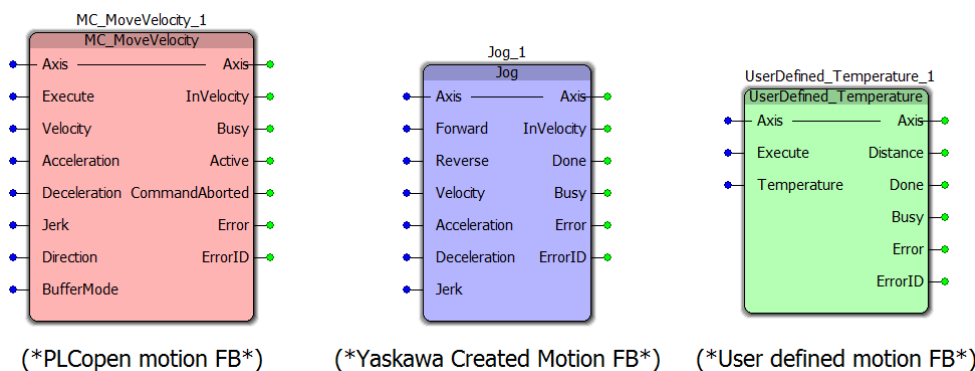


FIGURE 14 USER CREATED FUNCTION BLOCK: USERDEFINED_TEMPERATURE

Title: MPiec Programming Best Practices Guideline v3**Product(s):** MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec**Doc. No.** TN.MPIEC.03

17 OPTIMIZATION

The Optimize Axis is a new feature in MotionWorks IEC 3 which can be used to set tuning related parameters to their optimum value, reducing torque ripple and motor heating.

Parameter #	Parameters	Current Value
Pr109	Feed Forward Gain	100
Pr10A	Feed Forward Filter Time Constant	4.00
Pr811	Exponential Function Accel/Decel Time Constant	4.0
Pr812	Movement Average Time	4.0
1310	Velocity Feed Forward Enabled/Disabled	Disabled
1311	Drive Motor Command DPTION ACCFIL Value	2 - S-curve acceleration/deceleration

All parameters shown on this tab can improve performance for following applications:
 1) Axis with very high acceleration and deceleration rates
 2) Axis following cam master or gear master

Parameters 1311, Pr811, Pr812 can improve performance by reducing motor heating for all applications

Recommended Settings:
 * 1310 setting to disable VFF
 * 1311 setting to use S-curve acceleration/deceleration
 * Pr109 setting to use full (100%) feed forward gain
 * Pr10A, Pr811, Pr812 settings to equal servo network communication cycle time

Refer to the following supplemental information at www.yaskawa.com
 eLV MP2000iec 01 Parameters Webstar: MPiec - Controller Parameters for Performance

For details, refer to the eLearning Video below:

[eLM.MotionWorksIEC.13.MWiec3_Optimize](#)

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

18 CODE INSPIRATIONALS AND FAQs

18.1 Implementing Enable logic for function blocks

EN/ENO functionality is available within MotionWorks IEC 3 for enabling the evaluation of the function. This feature is available by default; however, the user has the ability to disable it by clicking on Objects in the Menu Bar > Unselecting Insert Block with EN/ENO:

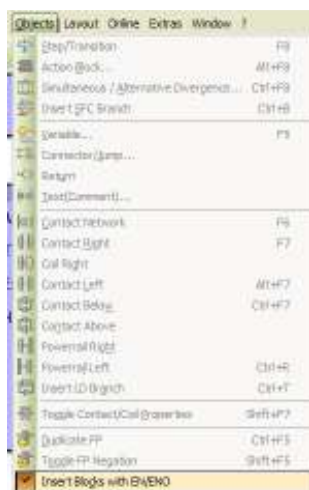


FIGURE 17 MENU FOR EN/ENO

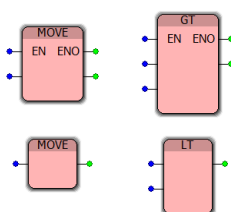


FIGURE 18 EN/ENO BLOCKS

For more details, refer to the e-learning video: [eLV.MotionWorksIEC.16.Basic_FunFbk](https://www.yaskawa.com/e-learning/video/eLV.MotionWorksIEC.16.Basic_FunFbk)

18.2 Are there limits on POU's or variables in a project?

Yes, the limits are 2,000 POU's and 40,000 global variables per project. Please search 'Project limits for ProConOS targets' in PLC help.

Title: MPiec Programming Best Practices Guideline v3**Product(s):** MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec**Doc. No.** TN.MPIEC.03

18.3 Write a coil at only one location in the program

It is recommended that a coil or a register be written only at one instance in a program because the last coil written to will take effect. Use cross referencing to check if there are multiple instances where a coil or a register is getting written.

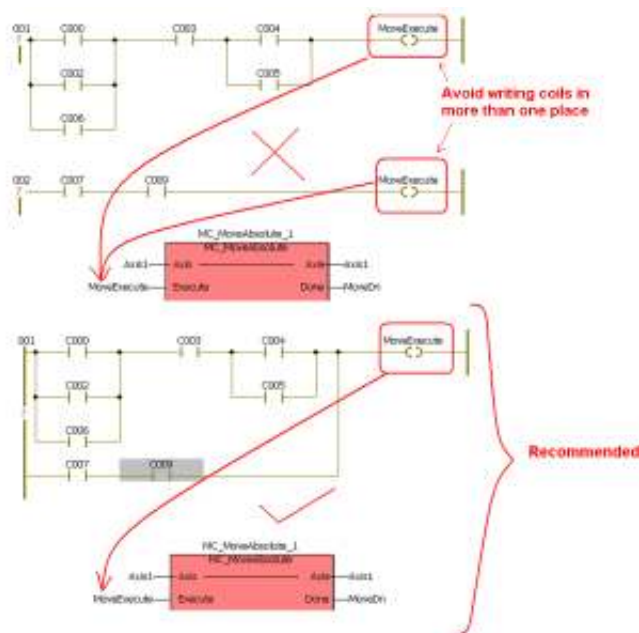


FIGURE 19 WRITE TO A COIL ONLY IN ONE PLACE

18.4 Set / Reset combinations

It is recommended to keep set-reset combinations as low as possible. Utilize the seal in circuit from [Section 5.2](#) when possible. If unavoidable, use set-reset coils physically close in a POU such that they can be verified in one debug window. Do not use only a set coil. Make sure every coil that is set has a corresponding reset.

Title: MPiec Programming Best Practices Guideline v3**Product(s):** MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec**Doc. No.** TN.MPIEC.03

18.5 Cascading ELSIF

When using a cascaded ELSIF THEN, is there a final ELSE?

```

1      FALSE IF      Input01
2      999   THEN    Output01 := 123;
3      FALSE ELSIF  Input02
4      999   THEN    Output01 := 456;
5      FALSE ELSIF  Input03
6      999   THEN    Output01 := 789;
7      FALSE ELSIF  Input04
8      999   THEN    Output01 := 159;
9      999   ELSE    Output01 := 999;
10     END_IF;

```

FIGURE 20 CASCADING ELSIF AND FINAL ELSE

18.6 WHILE Loop

Avoid infinite loops as this will cause the loop to keep running until the MPiec controller watchdogs. In the below example, if Input05 was not set to 'false' within the While loop, a watchdog would occur. When using a WHILE loop, ensure there is an unconditional increment of the variable that is controlling the loop.

```

FALSE WHILE Input05 (*EXPRESSION (must return a boolean value)*)
      DO (*Repeats the STATEMENTS until returned value of EXPRESSION = FALSE*)
5      a := 0;
5      a := 1 + 1 (*STATEMENTS*);
5      a := a + 1 (*STATEMENTS*);
5      a := a + 1 (*STATEMENTS*);
5      a := a + 1 (*STATEMENTS*);
FALSE Input05 := false;
      END_WHILE;

```

18.7 Handling Errors

Are all errors handled in a way to combine multiple errors into a single structure of an array? Verify all necessary errors are handled together before the Error is used in any logical statement.

18.8 Rising Edge and Falling Edge Triggers

Avoid R_TRIGs and F_TRIGs inside of conditional statements because if the conditional statement is not running, the triggers will not be seen. Oftentimes, these can be moved to the beginning of the code and their results checked conditionally.

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

```

6  (*****
7  (*****                               Event Handling Section   *****
8  (*****
9
10 R_TRIG_Enable(CLK:=Enable);
11 R_TRIG_Execute(CLK:=Execute);
12
13
14 (* This line causes the function block to exit if the execute, and all outputs are off. (For efficiency.) *)
15 (* IMPORTANT, be sure to include CommandAborted in the interlock logic if a motion block will be used. *)
16 IF NOT(Enable OR iActive OR Valid OR Done OR Busy OR Error) THEN RETURN; END_IF;
17
18 iActive:= (Execute OR iActive) AND NOT(Done OR Error);
19
20
21 (*****
22 (*****                               Initialization Section   *****
23 (*****
24
25 (*****                               This is a hybrid block that has an Enable AND an Execute input   *****
26 IF R_TRIG_Enable.Q OR ClearBuffer THEN
27     CircularByteBuffer.StorePointer:=INT#0;
28     CircularByteBuffer.UsePointer:=INT#0;
29     CircularByteBuffer.Size:=UDINT_TO_INT(UPPER_BOUND(CircularByteBuffer.Data,DINT#1));
30     x:=BytesRead; (* Stop the WHILE loop from attempting to
31 END_IF;
32
33 IF R_TRIG_Execute.Q THEN
34     Complete:=FALSE;
35     x:=UDINT#0;
36 END_IF;
    
```

FIGURE 21 R_TRIG AND F_TRIG OUTSIDE OF 'IF' STATEMENTS

18.9 Comments

Are all code sections of reasonable complexity commented? Please use comments!

```

(*****
(*****                               Set OutputFlags   *****
(*****
PathData.Segment[p].OutputFlags := DWORD#0;
PathData.Segment[p].OutputFlags.X0 := (GCodeData.Mode.Coolant = INT#7); (* Coolant On (mist) (Turns off if M Code was 9) *)
PathData.Segment[p].OutputFlags.X1 := (GCodeData.Mode.Coolant = INT#8); (* Coolant On (flood) (Turns off if M Code was 9) *)
    
```

FIGURE 22 COMMENTS

18.10 Sample Code Sections

Please add comments over any sections marked with 'TODO' or 'Add your code here'. Shown below is an example within the Enable_FB_Template function block:

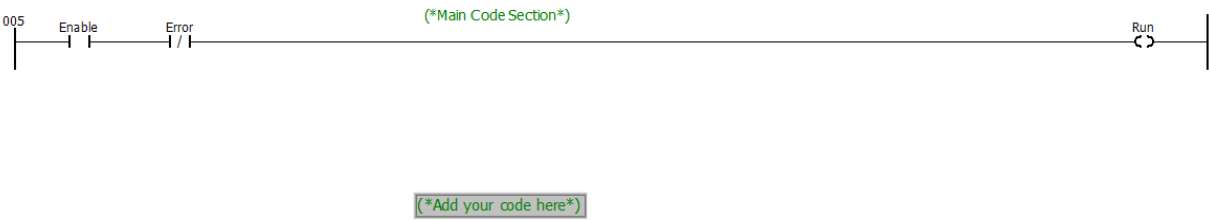


FIGURE 23 ADD COMMENTS

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

18.11 What is the effect of using different power rails for ladder rungs?

The effect of different power rails is shown in the example below.

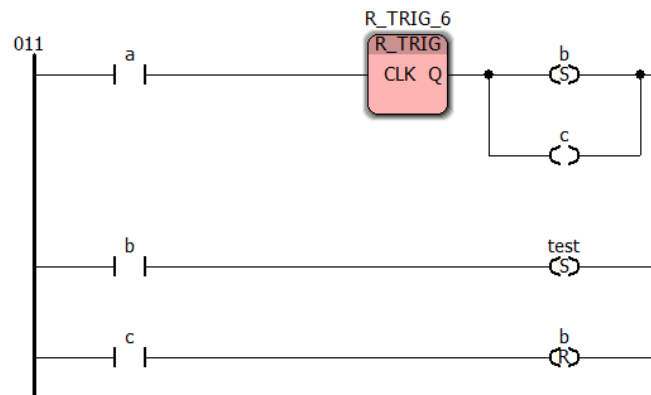


FIGURE 24 COILS ON THE SAME POWER RUNG

The coils b and c which go high in scan 1, act as contacts only in scan 2 if they are used in the same power rail as shown from the plot below. **Note that there is only one power rail in the above logic.**

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

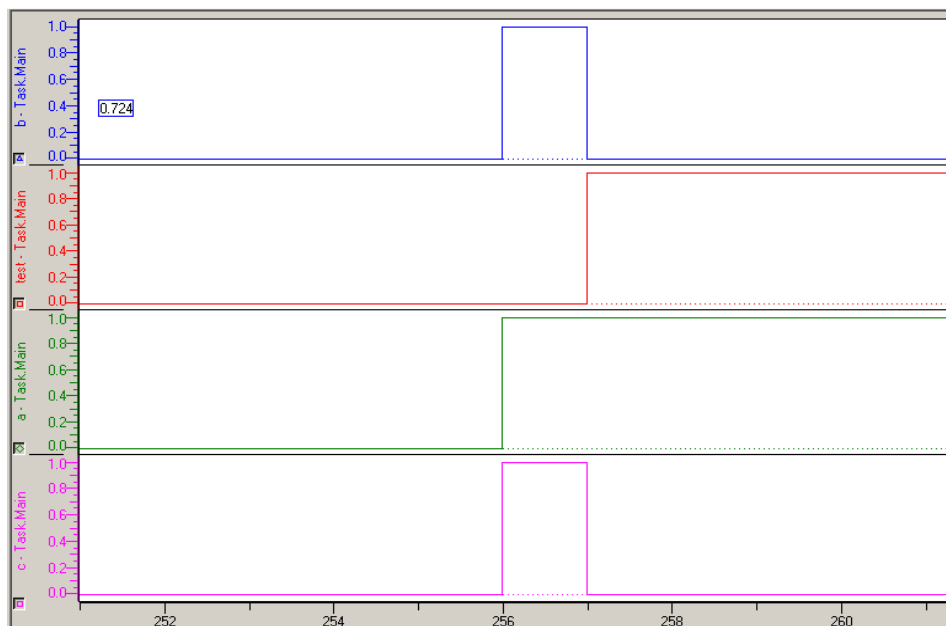


FIGURE 25 RESULTS OF COILS ON THE SAME POWER RUNG

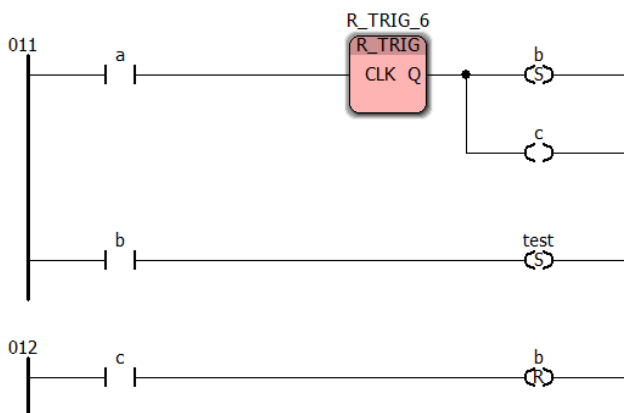


FIGURE 26 COILS ON SEPARATE POWER RUNGS

Note that there are two power rails in this above logic. In the above scenario, coil c which goes high in scan 1 acts as a contact in scan 1 to reset b in scan 1 (last rung). Coil b is used as a contact in the same power rail. So, b would get a chance to be a high contact only in the next scan (scan 2). Since b is reset by c in scan 1, b never becomes high in scan 1 or 2. As b never becomes high, test never goes high either.

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

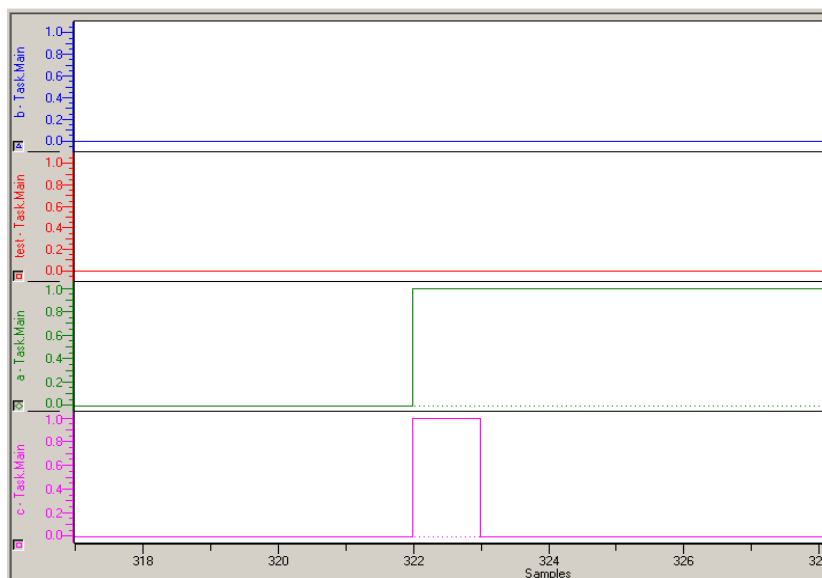


FIGURE 27 RESULTS OF COILS ON SEPARATE POWER RUNGS

18.12 The RETURN in MotionWorks IEC 3 programming

A return object in a ladder rung within a POU will make the scan control exit from the particular POU and go to the next POU in the task. For example in a project where the POU order of execution is as shown in the following figure, POU 'first' gets executed first and 'second' gets executed second.

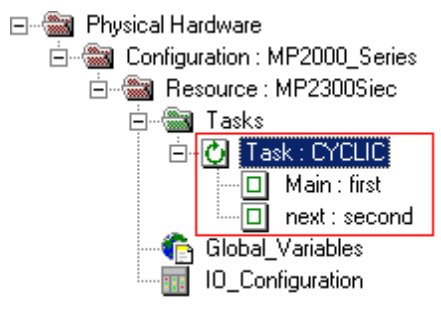


FIGURE 28 SEQUENCE OF EXECUTION

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

A return encountered in 'first' will make the scan ignore the ladder rungs and logic that is waiting to be scanned in 'first' and go to 'second'. This is shown using an example below. POU 'first' has the following logic. (All variables are of type BOOL)

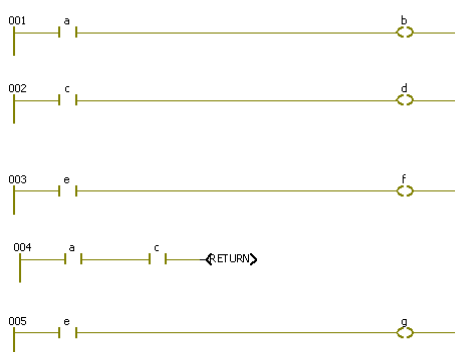


FIGURE 29 RETURN IN LOGIC IN POU 'FIRST'

If 'a' and 'c' are TRUE, 'g' will not be TRUE even if 'e' is true because the return interrupts the scan of POU 'first' and transfers the scan to 'second' whose logic is as follows.



FIGURE 30 LOGIC IN POU 'SECOND'

Variable 'h' in POU 'second' will become TRUE since 'e' (a global variable) is TRUE.

The result of running the project is shown below:

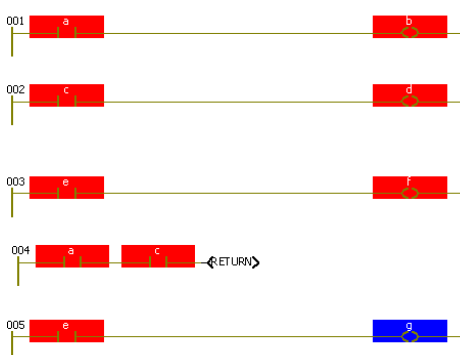


FIGURE 31 POU 'FIRST' IN DEBUG MODE

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

It can be seen that 'g' did not get triggered because the scan jumped to the next POU 'second' after line 4.



FIGURE 32 POU 'SECOND' IN DEBUG MODE

The sequence of bits is shown in the following figure.

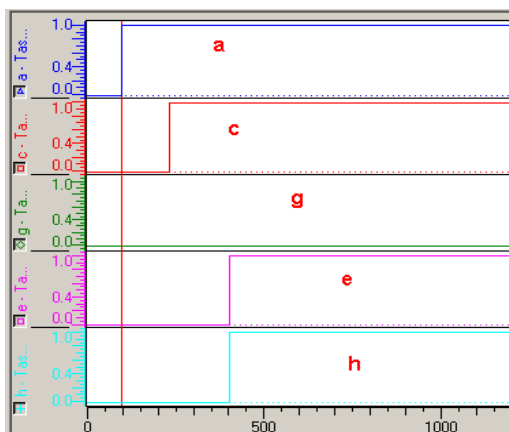


FIGURE 33 RESULTS OF RETURN

18.13 Rollover in MotionWorks IEC 3

Integers defined in MotionWorks IEC 3 automatically rollover

The rollover limit is dependent on the data type of the variable being used. For example, a variable defined as an Integer (INT: 16 bit) in MotionWorks IEC 3 has an operating range of -32768 to 32767. Other limits can be seen in the table provided below

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

TABLE 1 ELEMENTARY DATA TYPES AND LIMITS

Data type	Description	Size	Range	Default initial value
BOOL	Boolean	1	0...1	0
SINT	Short integer	8	-128...127	0
INT	Integer	16	-32,768...32,767	0
DINT	Double integer	32	-2,147,483,648 up to 2,147,483,647	0
USINT	Unsigned short integer	8	0 up to 255	0
UINT	Unsigned integer	16	0 up to 65,535	0
UDINT	Unsigned double integer	32	0 up to 4,294,967,295	0
REAL	Real numbers	32	-3.402823466 E+38 up to -1.175494351 E-38 and +1.175494351 E-38 up to +3.402823466 E+38	0.0
LREAL	Long real numbers	64	-1.7976931348623158 E+308 up to -2.2250738585072014 E-308 and +2.2250738585072014 E-308 up to +1.7976931348623158 E+308	0.0
TIME	Duration	32	0... 4,294,967,295 ms	t#0s
BYTE	Bit string of length 8	8	0...255 (16#00...16#FF)	0
WORD	Bit string of length 16	16	0...65,535 (16#00...16#FFFF)	0
DWORD	Bit string of length 32	32	0...4,294,967,295 (16#00...16#FFFFFF)	0

Real numbers do not rollover.

A variable defined as an LREAL (64 bit) in MotionWorks IEC 3 has a maximum operating range of -1.7E308 to 1.7E308. The variable will have to be reset before these limits are reached. If not, the value of the variable beyond these limits goes to - Inf or + Inf depending on how the count was progressing. Realizing these limits on LREAL variables that command position should not be of any practical concern to users because the MPiec controller series uses double precision floating point format for position definition and calculations.

18.14 Can variable descriptions be included in the editor window?

The description option can be selected by going to Extras>Options> ToolTips as shown below:

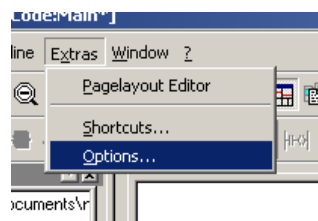


FIGURE 34 EDITOR OPTIONS

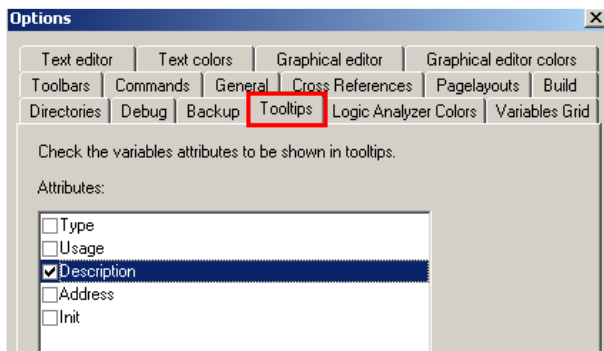
Title: MPiec Programming Best Practices Guideline v3**Product(s):** MP3300iec, MP3200iec, Sigma-7Siec,
MP2600iec, MotionWorks IEC 3, MP2300Siec**Doc. No.** TN.MPIEC.03

FIGURE 35 TOOLTIP ATTRIBUTES FOR EDITOR WINDOW

The description will not be displayed at all times. It will be displayed when the tool tip is hovered around the variable in question as shown below.

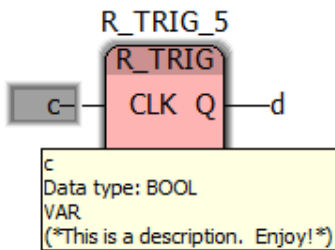


FIGURE 36 TOOLTIP OVER VARIABLE 'C'

18.15 Best practice in Structured Text Programming

Care should be exercised if a function block is enabled inside an IF ELSE structure. The state transition may never be scanned if the logic does not sweep through the IF ELSE structure.

Care should be exercised while implementing any one shot trigger based logic inside an IF-ELSE loop in structured text programming.

Put each input on its own line, more debug information will be available.

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

```

247 |
248 | | Y_CamStructSelect_1
249 | | (
250 | |   CamTable:=XCamTable,
251 | |   Execute:=Active AND (F_TRIG_SelectX.Q OR Y_CamStructSelect_1.Busy OR Y_CamStructSelect_1.Error),
252 | |   BlockSize:=UDINT#1024
253 | | );
254 | | XCamTable:=Y_CamStructSelect_1.CamTable;
255 | | F_TRIG_SelectY(CLK:=Y_CamStructSelect_1.Done);
256 | | FALSE IF F_TRIG_SelectY.Q THEN
257 | |   PathID.XAxisTable:=Y_CamStructSelect_1.CamTableID;
258 | | END_IF;
    
```

FIGURE 37 F_TRIG OUTSIDE OF IF THEN STATEMENT

18.16 Best practice in Sequential Function Chart (SFC) programming

Use of the 'Step.X' bit in action instances is recommended for SFC programming. This will ensure that the action is not held in an undetermined state if the step is not active. Use of the Step.X bit ('FwdStep.X' in this example) will ensure that when FwdStep stops being active, the contact FwdStep.X in the action block will be low and will prevent that particular rung from being stuck in one state. If another variable were being used in place of FwdStep.X, there is a possibility of that variable being in a stuck state if it did not change state while FwdStep was active.

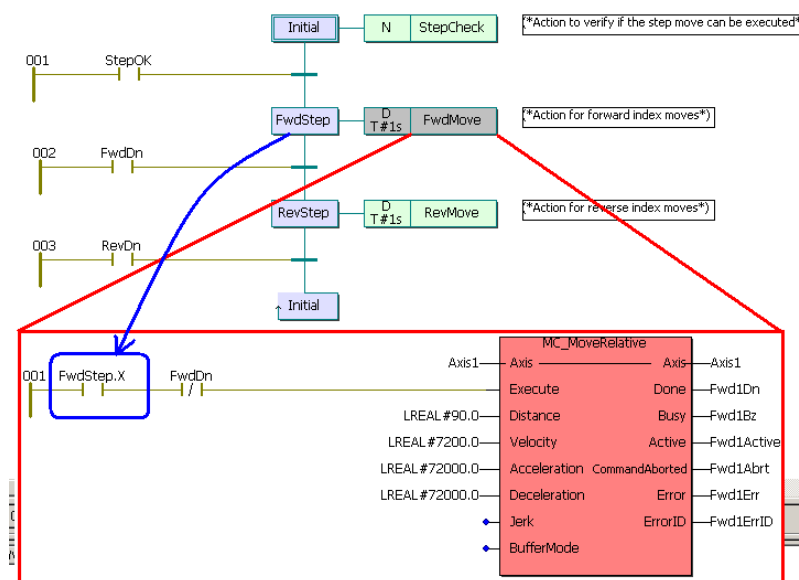


FIGURE 38 SFC PRACTICES

Title: MPiec Programming Best Practices Guideline v3

Product(s): MP3300iec, MP3200iec, Sigma-7Siec, MP2600iec, MotionWorks IEC 3, MP2300Siec

Doc. No. TN.MPIEC.03

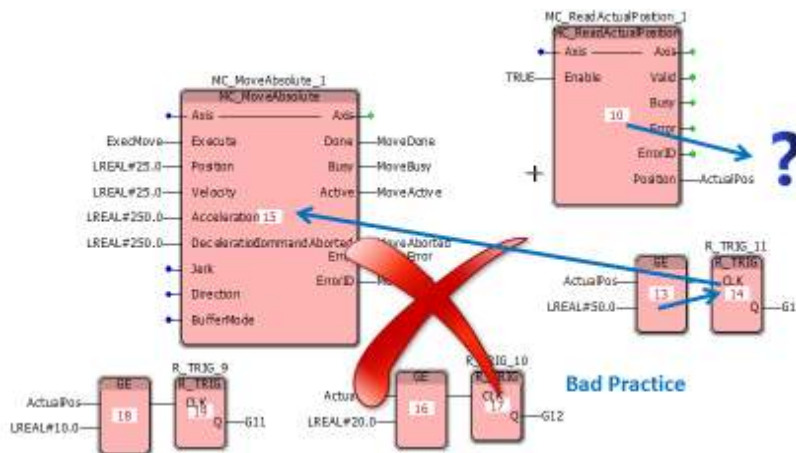


FIGURE 40 INCORRECT FLOW

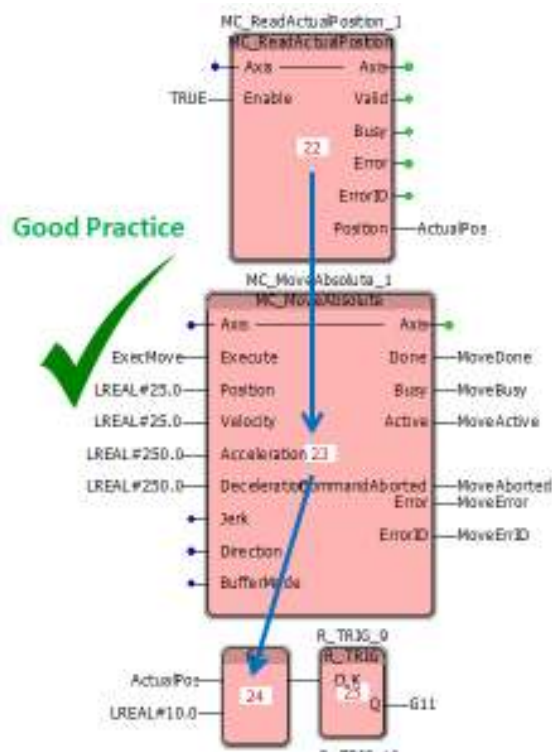


FIGURE 41 CORRECT FLOW