The purpose of this Application Note is to show a step by step implementation of an IEC61131-3 application to load in a CTI 2500P-ACP1 advanced co-processor. This application will perform Preventive Maintenance calculation for On/Off devices controlled by the CTI 2500® CPU.

INTRODUCTION

In our example, we will have a first motor AM001 with a running contact wired to X1 and a second motor AM102 with a running contact wired to X200. We want to get a Preventive Maintenance (PM) alarm for each motor when its start/stop cycle count exceed 50000 or when its running hour count exceed 10000 hours. The alarm for AM001 will be send to C10 and the alarm for AM102 will be send to C2000.

Device	Running Contact	PM alarm
AM001	X1	C10
AM102	X200	C2000

There will be no change to the CPU Workshop program. C10 and C2000 can be used in CPU to trigger alarm on a HMI/SCADA system.

The CTI CPU and ACP1 will be connected to an Ethernet switch together with the PC where CTI Workbench is installed.
A HMI/SCADA can connect too.

CTI Equipment	IP address	Sub Mask
PC with CTI Workbench	10.55.71.80	255.255.255.0
2500-Cx00	10.55.71.81	255.255.255.0
2500P-ACP1	10.55.71.82	255.255.255.0

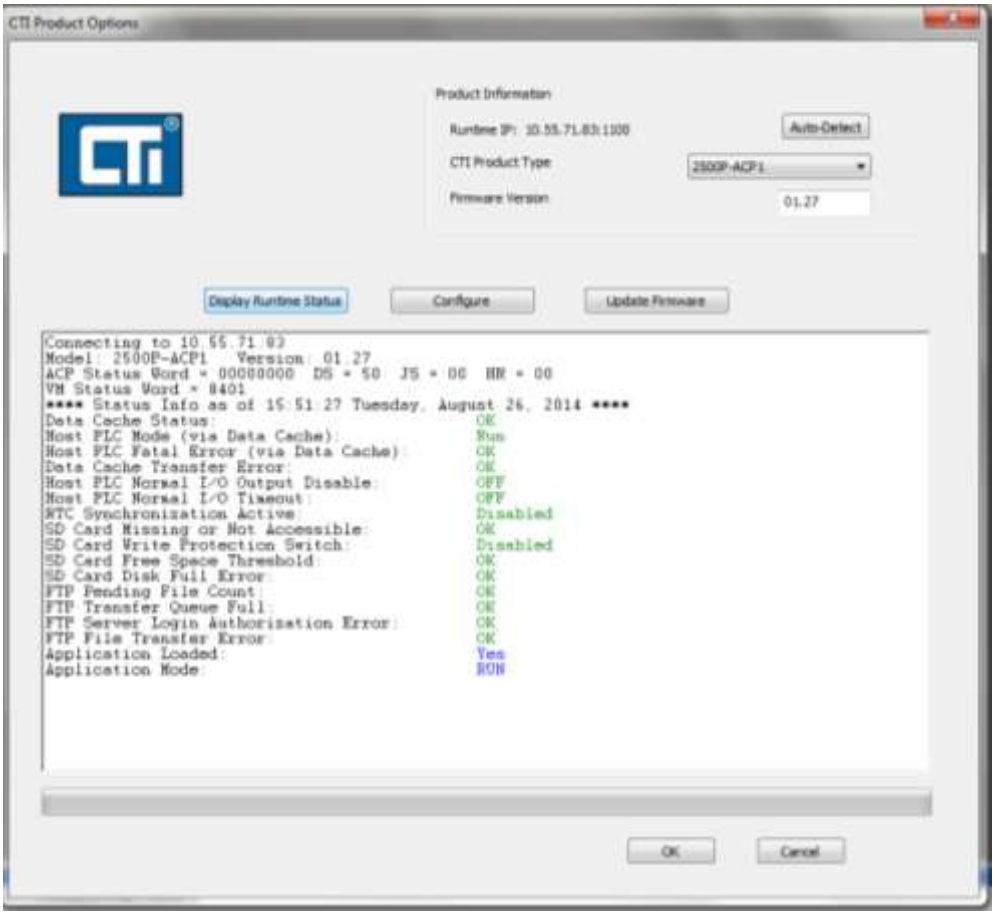
CREATING THE ACP1 PROJECT IN CTI WORKBENCH

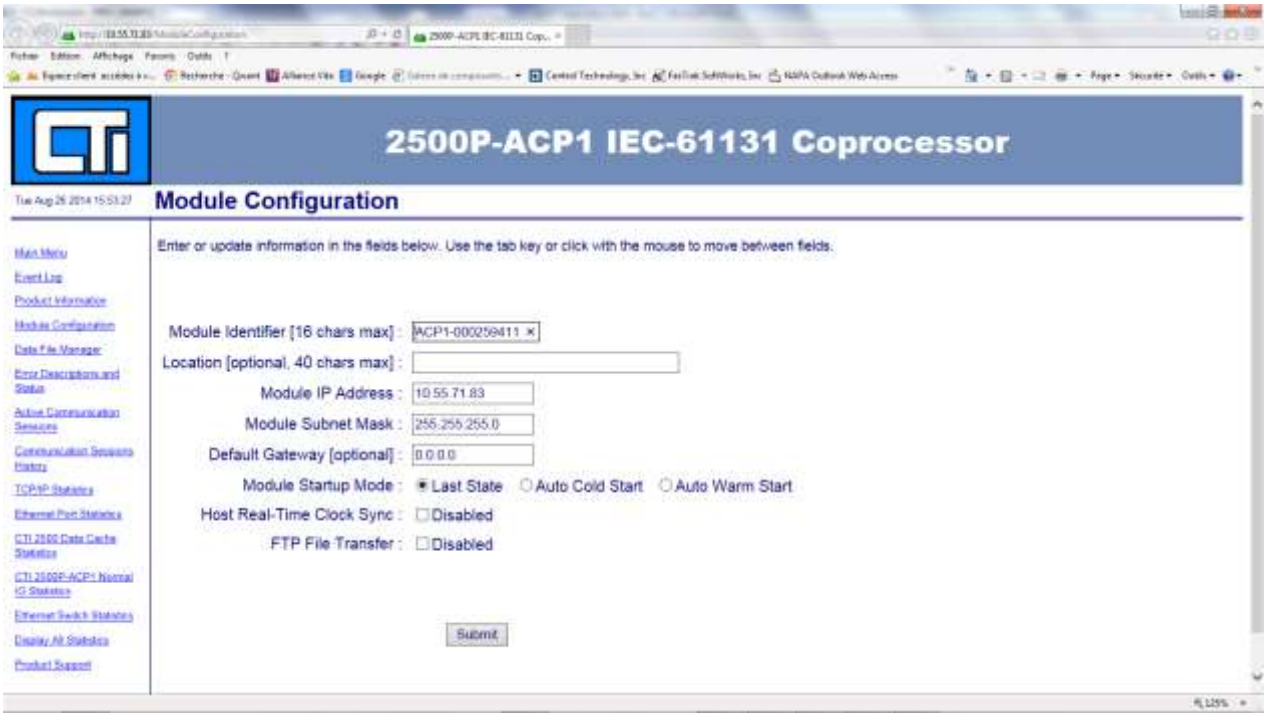
On your engineering PC, where FasTrak WorkShop and CTI Workbench are installed, we will create a new CTI Workbench workspace (PM_workspace) where our project (PREV_MAINT) will be added.

Step	Action	Note
A-1	Open CTI Workbench	From desktop shortcut
A-2	Select “File > New Project List…”	From the CTI Workbench menu bar
A-3	Enter “PM_workspace”	This creates your empty workspace or project list
A-4	Select “File > Add New Project…”	From the CTI Workbench menu bar
A-5	Enter “PREV_MAINT”	This creates your IEC 61131 project
A-6	Make sure “PREV_MAINT” is in **bold** characters	This means it is the Startup project in your workspace
A-7	If it is not the case, right click on “PREV_MAINT” and select “Set As Startup Project”	From the Project context menu

SPECIFYING YOUR ACP1 AS THE TARGET SYSTEM

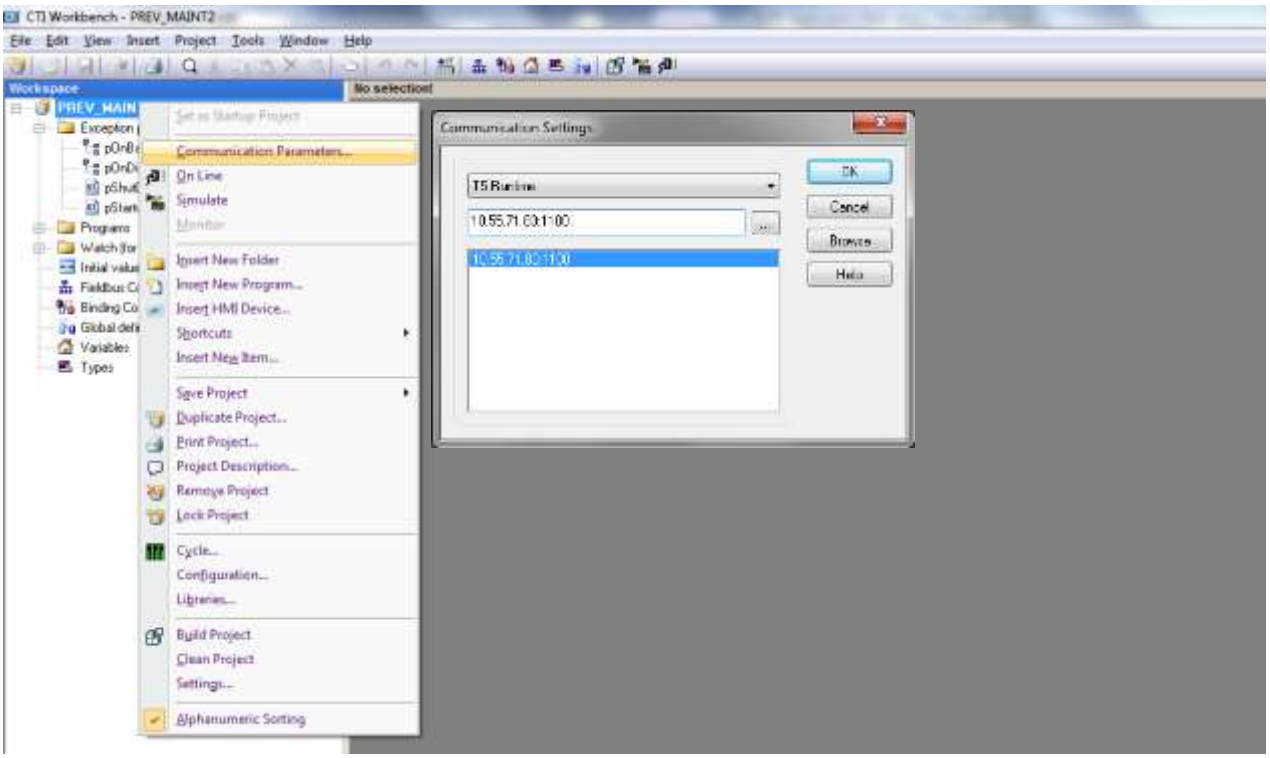
Step	Action	Note
B-1	Select “Tools” then “CTI Product Options”	From the CTI Workbench menu bar
B-2	Click on the “Auto-Detect” button	CTI Product Type field should now read “2500P-ACP1”
B-3	Click on the “Configure” button if the IP address is to be modified	Do not forget to adapt your PC IP address to the same range
B-4	Click on the “Display Runtime Status” button	This shows various information about the ACP1 status





SETTING THE COMMUNICATION PARAMETERS

Step	Action	Note
B-5	Right click on the “PREV_MAINT” program	This shows the Project context menu
B-6	Select “Communication Parameters…”	From the Project context menu
B-7	Select “T5 Runtime” in the top field	From the field list
B-8	Enter “10.55.71.82:1100” in the other field	This is the IP address of your ACP1 on port 1100



CREATING GLOBAL AND RETAIN VARIABLES AND INITIALIZE VARIABLE ON ACP1 POWER UP

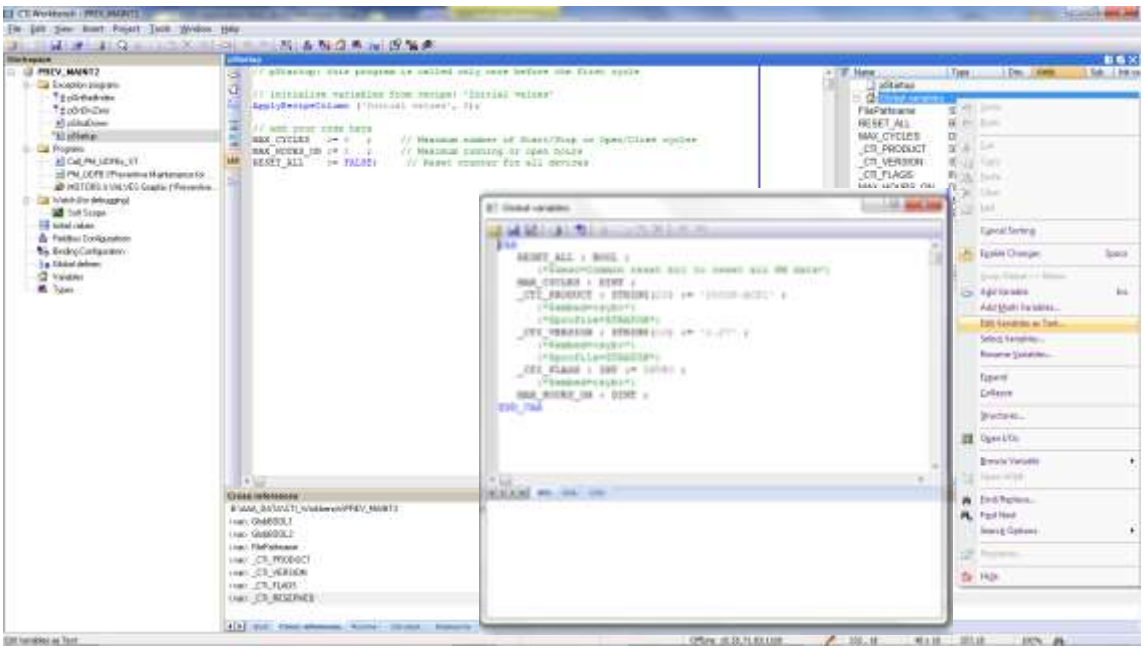
Let's create some global variables and initialize a few variables when the ACP1 starts up.

Step	Action	Note
C-1	Double click on the pre-existing program "pStartup" in the "Exception programs" folder of your project	This opens the pStartup Program as it is pre-build by CTI Workbench

C-2	Under the line
	`// add your code here`
	Add the following lines:
	`MAX_CYCLES := 50000; // Maximum number of Start/Stop or Open/Close cycles`
	`MAX_HOURS_ON := 10000; // Maximum Running or Open hours`
	`RESET_ALL := FALSE; // Reset counter for all devices`	

C-3	Right click on "Global variables" in the variable list on the right hand side of the screen and select "Edit Variables as Text…" and paste the following text:
	`VAR`
	`RESET_ALL : BOOL ;`
	`(*$desc=Common reset bit to reset all PM data*)`
	`MAX_CYCLES : DINT ;`
	`(*$desc=Max number of On/Off cycles to trigger PM alarm*)`
	`MAX_HOURS_ON : DINT ;`
	`(*$desc=Max number of hours On to trigger PM alarm*)`
	`END_VAR`	

C-4	Then right click on "RETAIN variables" in the variable list, on the right hand side of the screen and select "Edit Variables as Text…" and paste the following text:
	`VAR`
	`AM001_ON : BOOL ;`
	`AM001_CYCLES : DINT ;`
	`(*$embed=<syb>*)`
	`(*$profile=STRATON*)`
	`(*$prop6=350*)`
	`AM001_HOURS_ON : DINT ;`
	`(*$embed=<syb>*)`
	`(*$profile=STRATON*)`
	`(*$prop6=350*)`
	`AM001_MAINT_ALM : BOOL ;`
	`(*$embed=<syb>*)`
	`(*$profile=STRATON*)`
	`AM102_ON : BOOL ;`
	`AM102_MAINT_ALM : BOOL ;`
	`(*$embed=<syb>*)`
	`(*$profile=STRATON*)`
	`END_VAR`	



CREATING THE UDFB FOR THE GENERIC DEVICE PM DATA COMPUTATION

Step	Action	Note
D-1	Right click on the “Programs” folder	This shows the Program context menu
D-2	Select “Insert New Program…”	From the Program context menu
D-3	Enter “PM_UDFB” in the Name field Enter “Preventive Maintenance” in the Description field Select “ST” for the Programming language Select “UDFB” for the Execution style Click OK You can skip the Parameters window by clicking OK again.	This creates an empty User Defined Function Block program to be programmed in Structured Text where you are going to write the code for the Preventive Maintenance data calculation.
D-4	Double click on the “PM_UDFB” program	This opens the program with the Structured Text editor

This PM_UDFB is handling the logic for a generic On/Off device (it will then have to be instantiated for each real device). Copy the following code in the Program window:

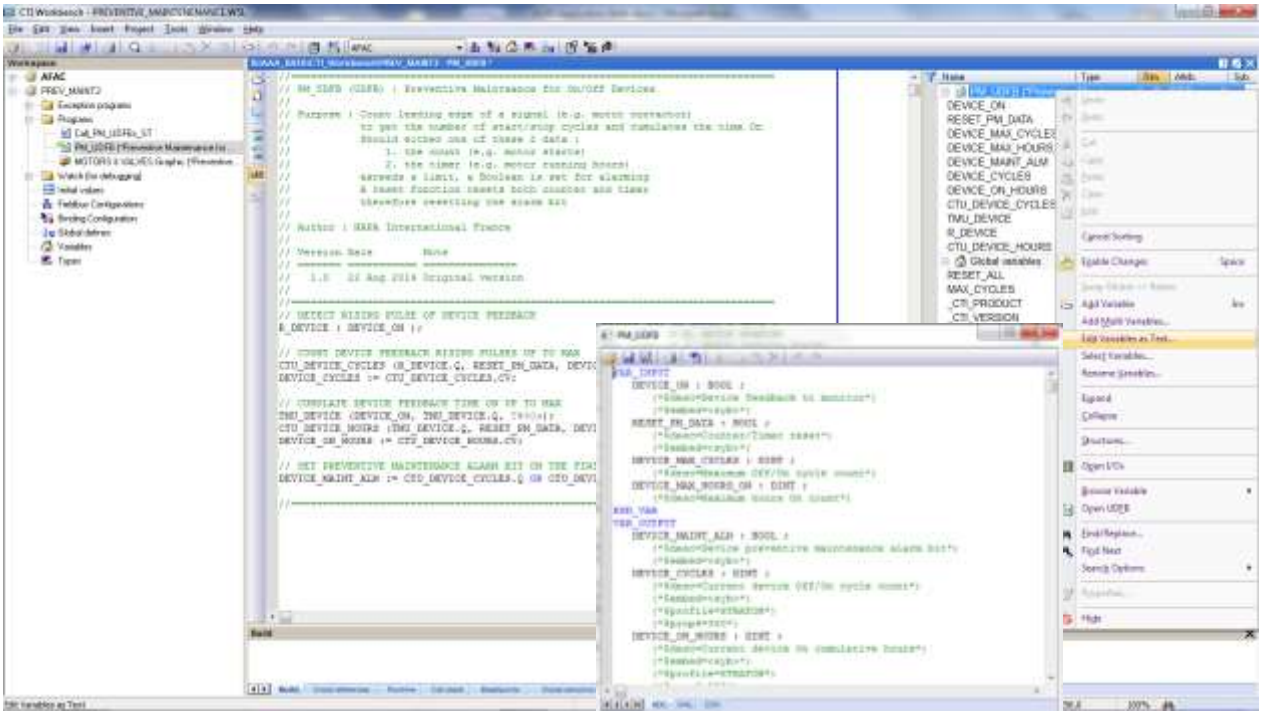
```
//=============================================================================
// PM_UDFB (UDFB) : Preventive Mainteance for On/Off Devices
//
// Purpose : Count leading edge of a signal (e.g. motor contactor)
//           to get the number of start/stop cycles and cumulates the time On
//           Should either one of these 2 data :
//               1. the count (e.g. motor starts)
//               2. the timer (e.g. motor running hours)
//           exceeds a limit, a Boolean is set for alarming
//           A reset function resets both counter and timer
//           therefore resetting the alarm bit
//
// Author : NAPA International France
//
// Version Date        Note
// ======= =========== ===============
//   1.0   22 Aug 2014 Original version
//=============================================================================

// DETECT RISING PULSE OF DEVICE FEEDBACK
R_DEVICE ( DEVICE_ON );

// COUNT DEVICE FEEDBACK RISING PULSES UP TO MAX
CTU_DEVICE_CYCLES (R_DEVICE.Q, RESET_PM_DATA, DEVICE_MAX_CYCLES);
DEVICE_CYCLES     := CTU_DEVICE_CYCLES.CV;

// CUMULATE DEVICE FEEDBACK TIME ON UP TO MAX
TMU_DEVICE (DEVICE_ON, TMU_DEVICE.Q, T#60s);
CTU_DEVICE_HOURS (TMU_DEVICE.Q, RESET_PM_DATA, DEVICE_MAX_HOURS_ON);
DEVICE_ON_HOURS   := CTU_DEVICE_HOURS.CV;

// SET PREVENTIVE MAINTENANCE ALARM BIT ON THE FIRST LIMIT REACHED
DEVICE_MAINT_ALM  := CTU_DEVICE_CYCLES.Q OR CTU_DEVICE_HOURS.Q;
//=============================================================================
```

Then right click on “PM_UDFB” in the variable list, on the right hand side of the screen and select “Edit Variables as Text…” and paste the following text:

```
VAR_INPUT
   DEVICE_ON : BOOL ;
      (*$desc=Device feedback to monitor*)
      (*$embed=<syb>*)
   RESET_PM_DATA : BOOL ;
      (*$desc=Counter/Timer reset*)
      (*$embed=<syb>*)
   DEVICE_MAX_CYCLES : DINT ;
      (*$desc=Maximum Off/On cycle count*)
   DEVICE_MAX_HOURS_ON : DINT ;
      (*$desc=Maximum hours On count*)
END_VAR
VAR_OUTPUT
   DEVICE_MAINT_ALM : BOOL ;
      (*$desc=Device preventive maintenance alarm bit*)
      (*$embed=<syb>*)
```

```
   DEVICE_CTR : DINT ;
      (*$desc=Current device Off/On cycle count*)
      (*$embed=<syb>*)
      (*$profile=STRATON*)
      (*$prop6=350*)
   DEVICE_ONHOURS : DINT ;
      (*$desc=Current device On cumulative hours*)
      (*$embed=<syb>*)
      (*$profile=STRATON*)
      (*$prop6=350*)
END_VAR
VAR
   CTU_DEVICE_CYCLES : CTU ;
      (*$desc=Up Counter*)
   TMU_DEVICE : TMU ;
      (*$desc=Up-Counting Stop Watch*)
   R_DEVICE : R_TRIG ;
      (*$desc=Rising Pulse Detection*)
   CTU_DEVICE_HOURS : CTU ;
      (*$desc=Up Counter*)
END_VAR
```

Now your PM_UDFB program is complete. You still need to create a main program to call one instance of this UDFB for each On/Off device you need.



CREATING THE MAIN PROGRAM CALLING THE UDFB INSTANCES

Step	Action	Note
E-1	Right click on the “Programs” folder	This shows the Program context menu
E-2	Select “Insert New Program…”	From the Program context menu
E-3	Enter “Call_PM_UDFBs” in the Name field Enter “For each device” in the Description field Select “ST” for the Programming language Select “Main” for the Execution style Click OK You can skip the Parameters window by clicking OK again.	This creates a main program to be programmed in Structured Text where the previous UDFB will be instantiated for each On/Off device.
E-4	Double click on the “Call_PM_UDFBs” program	This opens the program with the Structured Text editor

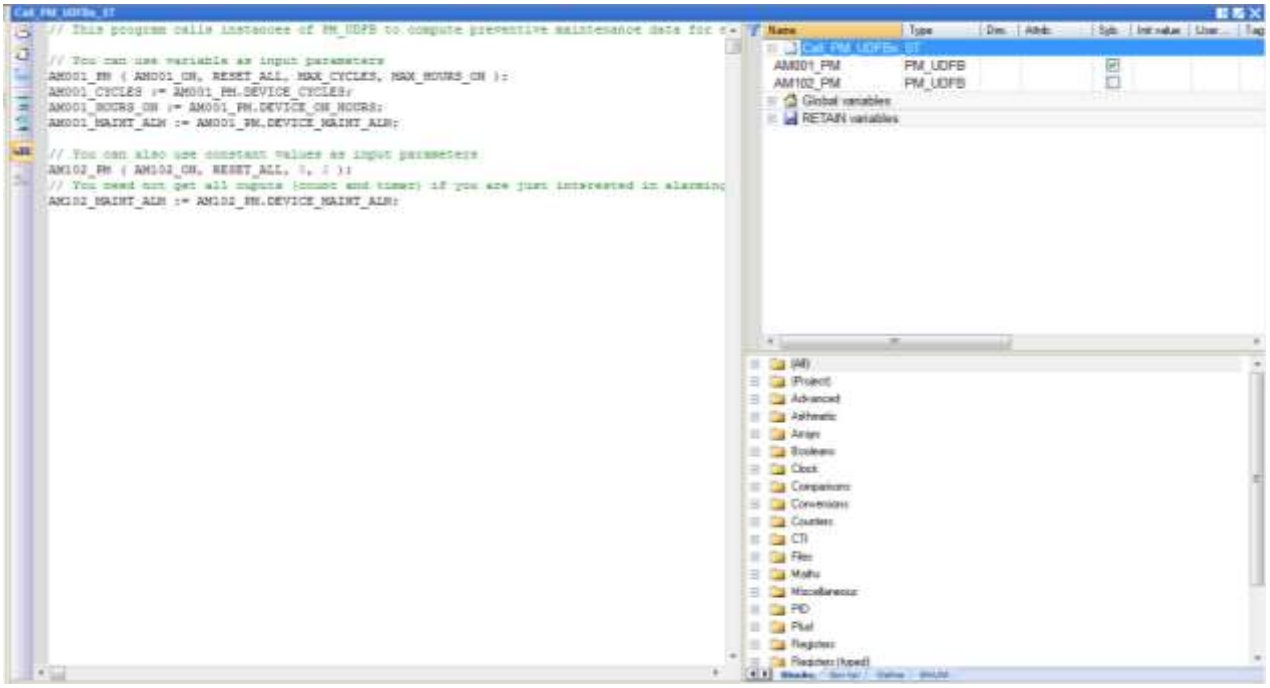
This Call_PM_UDFBs is calling the previous UDFB for each generic On/Off device. Copy the following code in the Program window:

```
// This program calls instances of PM_UDFB to compute PM data for each On/Off device

// You can use variable as input parameters
AM001_PM ( AM001_ON, RESET_ALL, MAX_CYCLES, MAX_HOURS_ON);
AM001_CYCLES := AM001_PM.DEVICE_CYCLES;
AM001_HOURS_ON := AM001_PM.DEVICE_ON_HOURS;
AM001_MAINT_ALM := AM001_PM.DEVICE_MAINT_ALM;

// You can also use constant values as input parameters
AM102_PM ( AM102_ON, RESET_ALL, 50000, 10000 );
// You need not get all ouputs (count and timer) if you are just interested in alarming bit
AM102_MAINT_ALM := AM102_PM.DEVICE_MAINT_ALM;
```

Then right click on “Call_PM_UDFBs” in the variable list, on the right hand side of the screen and select “Edit Variables as Text…” and paste the following text:

```
VAR
   AM001_PM : PM_UDFB ;
   AM102_PM : PM_UDFB ;
END_VAR
```

Now your Call_PM_UDFBs program is complete.



FIELD BUS CONFIGURATION: DATA EXCHANGE WITH THE CTI CPU

You now need to link a few project variables with the ACP1 cache to/from the CTI CPU. Follow these steps:

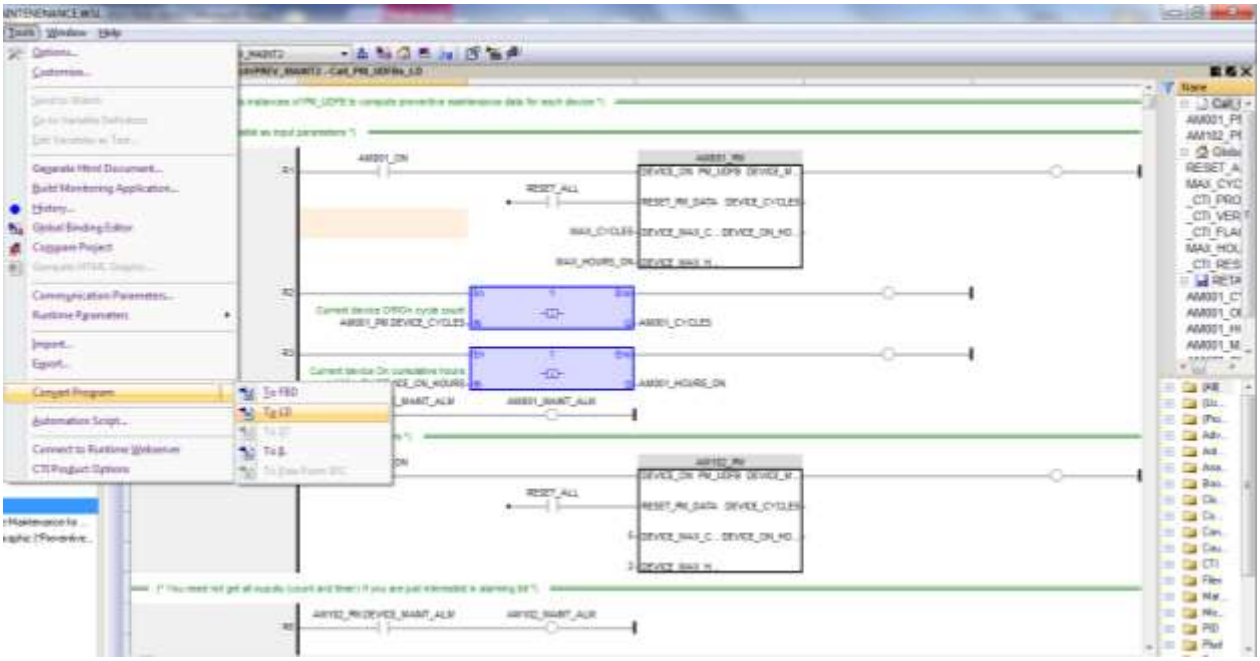
Step	Action	Note
F-1	Click on the “Fieldbus Configurations” folder	This open the Fieldbus IO driver editor
F-2	Right click in the editor	This calls the editor context menu
F-3	Select “Insert Configuration…” then select “All” then select “CTI 2500 Data Cache”	From the editor context menu
F-4	Right click on “CTI 2500 Data Cache”	This calls the configuration context menu
F-5	Select “Insert Master/Port…”	From the configuration context menu
F-6	Enter “10.55.71.81” in the PLC IP address field	Enter the IP address of the CTI 2500 PLC CPU
F-7	Leave the other entries to their default values	5, LAN, not used
F-8	Click “OK”	This create an entry : CTI 2500 IP =”10.55.71.81”
F-9	Right click on “CTI 2500 IP = “10.55.71.81””	This calls the master/port context menu
F-10	Select “Insert Slave/Data Block…”	From the master/port context menu
F-11	Select “Common”, then “Discrete I/O (XY) [BOOL]” then enter “1” then select “Read From PLC”, then click “OK”	This creates a read entry starting at X1
F-12	Right click on “Discrete I/O (XY) [BOOL] (1)”	This calls the slave/data context menu
F-13	Select “Insert Variables…”	From the slave/data context menu
F-14	Click on the “…” button if the Edit Variable window then scroll to select AM001_ON then enter “0” in the Offset field, then click OK	This copies the value of X1 to AM001_ON variable.
F-15	Select again “Insert Variables…”	From the slave/data context menu
F-16	Click on the “…” button if the Edit Variable window then scroll to select AM102_ON then enter “199” in the Offset field, then click OK	This copies the value of X200 to AM102_ON variable. (1+199)
F-17	Select “Insert Slave/Data Block…”	From the master/port context menu
F-18	Select “Common”, then “Control Relay (C) [BOOL]” then enter “10” then select “Write to PLC”, then click “OK”	This creates a write entry starting at C10
F-19	Right click on “Control Relay (C) [BOOL] (10)”	This calls the slave/data context menu
F-20	Select “Insert Variables…”	From the slave/data context menu
F-21	Click on the “…” button if the Edit Variable window then scroll to select AM001_MAINT_ALM then enter “0” in the Offset field, then click OK	This copies the value of AM001_MAINT_ALM variable to C10.
F-22	Select “Common”, then “Control Relay (C) [BOOL]” then enter “2000” then select “Write to PLC”, then click “OK”	This creates a write entry starting at C2000
F-23	Right click on “Control Relay (C) [BOOL] (2000)”	This calls the slave/data context menu
F-24	Select “Insert Variables…”	From the slave/data context menu
F-25	Click on the “…” button if the Edit Variable window then scroll to select AM102_MAINT_ALM then enter “0” in the Offset field, then click OK	This copies the value of AM102_MAINT_ALM variable to C2000.

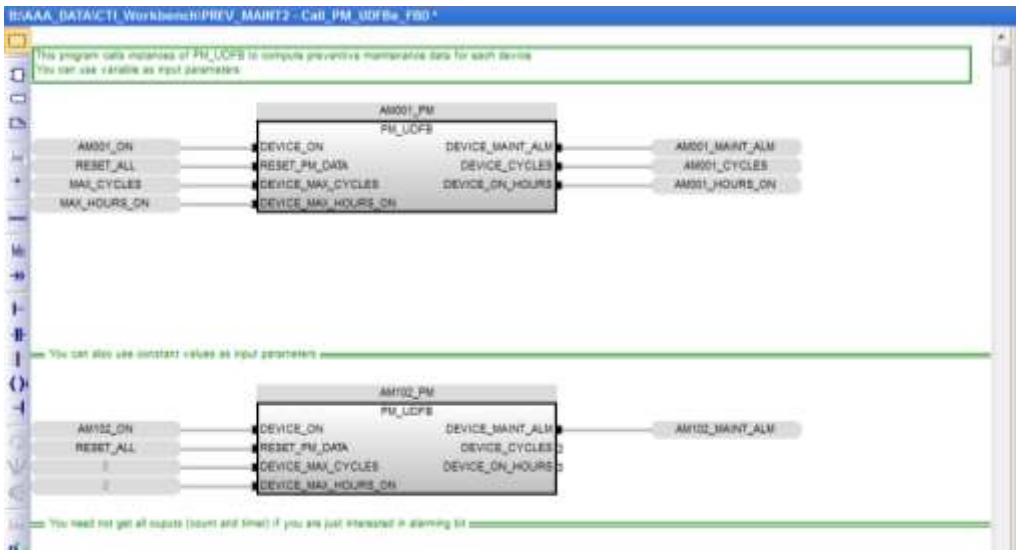
COMPILING AND LOADING

You are now ready to compile and load the project to the target ACP1:

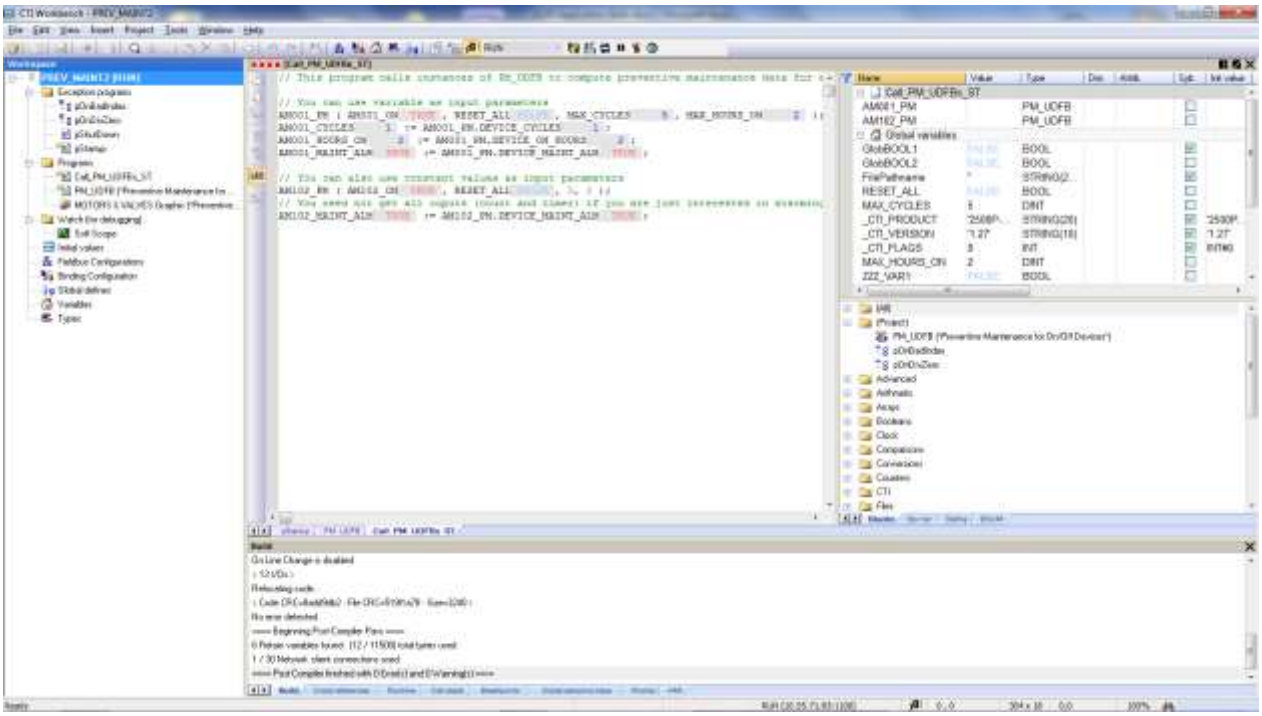
Step	Action	Note
G-1	Click on the "Build Startup Project" icon in the menu bar	This compile your project
G-2	Click on the "On Line" icon in the menu bar	This will connect CTI Workbench to the ACP1
G-3	Click on the "Download changes" icon in the menu bar	This will load your project in the ACP1 memory
G-4	If the ACP1 status shows "STOP" or "No application", Then click on the "Restart" icon in the menu bar and answer "Cold Start" to the prompt	This will restart the ACP1 with the new project

NOTE: Once successfully compiled any program in your project can be translated from ST to LD or FBD if you are mode familiar with these languages.





Now you can monitor the behavior of your programs with CTI Workbench in Online mode.



You can use FasTrak WorkShop® and create a Data Window to modify X1 and X200 and see what happens to C10 and C2000.

ACP1 INTERNAL GRAPHICS

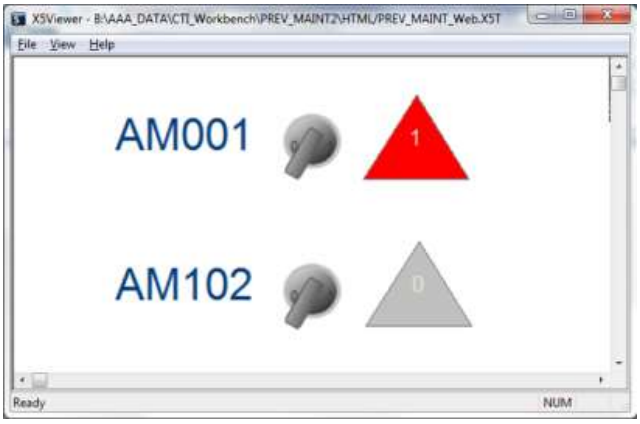
Suppose you don't have a HMI/SCADA system to get the Preventive Maintenance alarms (C10 and C2000). You can create a special Graphic with CTI Workbench and load it in the ACP1. This will allow some simple, local monitoring.

Step	Action	Note
H-1	Right click on the "Programs" folder	This shows the Program context menu
H-2	Select "Insert New Item…"	From the Program context menu
H-3	Select "Graphics" in the "Available Items" column Then click OK Enter "PM_VIEW" in the Name field Enter "Preventive Maintenance" in the Description field Then click OK	This will create a new Graphic PM_VIEW in your project
H-4	Double click on the "PM_VIEW" graphic	This will open your empty graphic in the Graphic editor of CTI Workbench
H-5	Drag and drop graphical item "Grey Round Switch" from the list on the right hand side under "Switches" to your graphic	This will add a switch (to show the device On feedback)
H-6	Double click on this switch item in your graphic	This will display its Graphic Item Properties
H-7	Double click on the "Variable symbol"	This will call the CTI Workbench variable browser
H-8	Select "AM001_ON" from the list	This will attach the switch to AM001 On feedback
H-9	Drag and drop graphical item "Triangle" from the list on the right hand side under "Shapes" to your graphic	This will add a switch (to show the device On feedback)
H-10	Double click on this triangle item in your graphic	This will display its Graphic Item Properties
H-11	Double click on the "Variable symbol"	This will call the CTI Workbench variable browser
H-12	Select "AM001_MAINT_ALM" from the list Select red in the "TRUE color" property Select grey in the "FALSE color" property	This will attach the triangle to AM001 PM alarm
H-13	Right click on your graphic background	This will display the Graphic context menu
H-14	Select "Generate HTML Graphic…"	From the Graphic context menu
H-15	Click on "Next" Then navigate to the folder where you want to store the generated HTML files (you can keep the default location)	You can use the "Create Folder" to create a new folder
H-16	Click on "Next" Then enter your HTML file name Leave "K5NET5.DLL" in the Target name field Use "10.55.71.83:1100" for Connection settings Leave the other fields to their default values	This specify the name of the file in the target (ACP1) system and how to connect to it, as well as other attributes such as its size…
H-17	Click on "Next" Tick the "Display HTML graphic" box Then click on "Generate" button	This will create the HTML file from your PM_VIEW graphic

The graphic now needs to be downloaded in the ACP1:

Repeat step G-1 to G-4.

The PM_VIEW graphic can now be used in Workbench on line as well as with the X5Viewer tool. See how the graphic animates when you change values in a FasTrak WorkShop® Data Window.



CONCLUSION

Suggestions if you want to go further:

- Have the RESET_ALL signal read from the CPU (ex: C3)
- Have the AM001_CYCLES and AM001_HOURS_ON written to the CPU (ex: V2000 and V2002 long integers)
- Add these data on the PM_VIEW Graphic

This completes this ACP1 step by step introduction illustrated by the Preventive Maintenance application.

If you have question about this Application Note, or if you want this CTI Workbench “Preventive Maintenance” application project backup sent to you by e-mail, do not hesitate to contact support@NAPA.fr

www.napa.fr
support@napa.fr
Tel :+33 4 93 20 93 93
Tel :+33 4 92 02 44 07

NAPA International France
Marina 7, 1545 Route Nationale 7
06270 – Villeneuve-Loubet
France