Project Documentation

File:	WebApp_PLC_Logic.ecp
Date:	12/7/2021
Profile:	e!COCKPIT

Table of contents

1 POU: CommandHandler 3
2 POU: PRG_VariableLogger 5

1 POU: CommandHandler

```
(* Purpose of the program: Receive and handle commands aka "metric-change-requests" from the Sparkplug primary application
Ignition has functionality for "write tag", so within the commands our example program will define exacly one parameter,
which is actually the writable metric from ignitions point of view.
The data protocol Sparkplug doesn't support response messages. (FbCommandResponse)  *)

PROGRAM  CommandHandler
VAR
    (* Use the function block FbCommandConfigurator to send the command configuration to the cloud *)
    oFbCmdConfigurator  :   WagoAppCloud . FbCommandConfigurator ;
    (* Define two commands in the list *)
    aCommandDescriptions  :      ARRAY  [ 0 .. 1 ]  OF  WagoAppCloud . typCommandDescription ;
    (*        Listen to command calls from the cloud using the FbCommandListener.
           It will set the xCommandReceived flag to true if a command is called. *)
    oFbCmdHandler  :  WagoAppCloud . FbCommandListener ;
    xCommandReceived :  BOOL ;
    IncomingCommand  :  WagoAppCloud . typCommandRequest ;
    dwReceivedCmdId  :  DWORD ;
    response             :  WagoAppCloud . typCommandResponse ;
    oFbCmdResponder      :  WagoAppCloud . FbCommandResponder ;
    (* Parameter to get the received value from the cloud *)
    rParameter1  :  STRING ;
    xResponseTrigger :  BOOL ;
END_VAR

```

```
(* Init command1 and its Request Parameter *)
aCommandDescriptions [ 0 ] . bCommandId  :=  1 ;
aCommandDescriptions [ 0 ] . bNumberOfRequestParameters  :=  6 ;
aCommandDescriptions [ 0 ] . bNumberOfResponseParameters  :=  1 ;
aCommandDescriptions [ 0 ] . sName  :=  'Commands' ;


aCommandDescriptions [ 0 ] . aRequestParameters [ 0 ] . sParameterName  := 'BTN_Supply_input_pump_Outer_Tank' ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 0 ] . eParameterType  := WagoAppCloud . eCommandParameterType . CPT_BOOL ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 1 ] . sParameterName  := 'BTN_Water_output_pump_Outer_Tank' ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 1 ] . eParameterType  := WagoAppCloud . eCommandParameterType . CPT_BOOL ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 2 ] . sParameterName  := 'Customer_High_Level_Inner_tank' ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 2 ] . eParameterType  := WagoAppCloud . eCommandParameterType . CPT_BOOL ;

aCommandDescriptions [ 0 ] . aRequestParameters [ 3 ] . sParameterName  := 'BTN_Heater' ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 3 ] . eParameterType  := WagoAppCloud . eCommandParameterType . CPT_BOOL ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 4 ] . sParameterName  := 'Set_Point_Heater' ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 4 ] . eParameterType  := WagoAppCloud . eCommandParameterType . CPT_REAL ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 5 ] . sParameterName  := 'External_temperature' ;
aCommandDescriptions [ 0 ] . aRequestParameters [ 5 ] . eParameterType  := WagoAppCloud . eCommandParameterType . CPT_REAL ;

// The request parameters and the response parameter are independently of each other - they can be different parameter types
aCommandDescriptions [ 0 ] . aResponseParameters [ 0 ] . sParameterName  := 'Booleans of command 1 received' ;
aCommandDescriptions [ 0 ] . aResponseParameters [ 0 ] . eParameterType  := WagoAppCloud . eCommandParameterType . CPT_BOOL ;

(* Configurate the commands and send the commands configuration to the cloud *)
oFbCmdConfigurator (     pSupportedCommands  := ADR ( aCommandDescriptions ) ,
                    bNumberOfSupportedCommands  := 1 ) ;

(* Receive data to get the command from the cloud *)
oFbCmdHandler ( pCommand := ADR ( IncomingCommand ) ,
                xCommandReceived => xCommandReceived ) ;

(* Handle command *)
IF  xCommandReceived  THEN
    dwReceivedCmdId  := IncomingCommand . dwCommandId ;

    CASE  dwReceivedCmdId  OF
        1 :
            (* received command with the command ID 1 *)
            GVL . Set_point_room  := TO_REAL ( IncomingCommand . aRequestParameters [ 4 ] . sParameterValue ) ;
            GVL . External_temperature_Room  := TO_REAL ( IncomingCommand . aRequestParameters [ 5 ] . sParameterValue ) ;

            IF  ( IncomingCommand . aRequestParameters [ 0 ] . sParameterValue ) =  '1'  OR  ( ( IncomingCommand . aRequestParameters [ 0 ] . sParameterValue )  = 'TRUE' ) THEN
                GVL . In_BTN_input_pump_Outer_Tank  := TRUE ;
            ELSE
                GVL . In_BTN_input_pump_Outer_Tank  := FALSE ;
            END_IF

            IF  ( IncomingCommand . aRequestParameters [ 1 ] . sParameterValue ) =  '1'  OR  ( ( IncomingCommand . aRequestParameters [ 1 ] . sParameterValue )  = 'TRUE' )  THEN
                GVL . In_BTN_Water_output_pump_Outer_Tank  := TRUE ;
                ELSE
                GVL . In_BTN_Water_output_pump_Outer_Tank  := FALSE ;
            END_IF

            IF ( IncomingCommand . aRequestParameters [ 2 ] . sParameterValue ) = '1' OR ( ( IncomingCommand . aRequestParameters [ 2 ] . sParameterValue ) = 'TRUE' ) THEN
                GVL . In_BOOL_Customer_HLevel_Inner_Tank := TRUE ;
                ELSE
                GVL . In_BOOL_Customer_HLevel_Inner_Tank := FALSE ;
            END_IF

            IF ( IncomingCommand . aRequestParameters [ 3 ] . sParameterValue ) = '1' OR ( ( IncomingCommand . aRequestParameters [ 3 ] . sParameterValue ) = 'TRUE' ) THEN
                GVL . In_BOOL_BTN_Heater := TRUE ;
                ELSE
                GVL . In_BOOL_BTN_Heater := FALSE ;
            END_IF

            response . dwCommandId := dwReceivedCmdId ;
            response . dwInvokeId := IncomingCommand . dwInvokeId ;
            response . bNumberOfResponseParameters := 1 ;
            response . aResponseParameters [ 0 ] . eParameterType := aCommandDescriptions [ 0 ] . aResponseParameters [ 0 ] . eParameterType ;
            response . aResponseParameters [ 0 ] . sParameterName := aCommandDescriptions [ 0 ] . aResponseParameters [ 0 ] . sParameterName ;
            response . aResponseParameters [ 0 ] . sParameterValue := TO_STRING ( rParameter1 ) ;
            xResponseTrigger := TRUE ;
    END_CASE
END_IF

oFbCmdResponder ( pCommand := ADR ( response ) ,
                xTrigger := xResponseTrigger ) ;

```

2 POU: PRG_VariableLogger

```
(* Send data from the PFC to the cloud *)
PROGRAM PRG_VariableLogger

VAR RETAIN
    (* Remanent variables *)
    tSampleInterval1 : TIME := T#1S ;
    tPublishInterval1 : TIME := T#2S ;
    tSampleInterval2 : TIME := T#2S ;
    tPublishInterval2 : TIME := T#4S ;
END_VAR

VAR
    (* Define two collections*)
    aCollections : ARRAY [ 0 .. 1 ] OF WagoAppCloud . typCollection ;

    (* Define two variables for the collections*)
    aVariableDescriptions1 : ARRAY [ 0 .. 4 ] OF WagoAppCloud . typVariableDescription ;
    aVariableDescriptions2 : ARRAY [ 0 .. 4 ] OF WagoAppCloud .
typVariableDescription ;

    (* Function block to log the values to the cloud *)
    oFbCollectionLogger : WagoAppCloud . FbCollectionLogger ;

END_VAR

```

```
(* Init collection 1 *)
aVariableDescriptions1 [ 0 ] . pAddress := ADR ( Tank_control_POU_ST .
Output_Supply_Input_Pump ) ;
aVariableDescriptions1 [ 0 ] . eValueType := WagoAppCloud . VVT_BOOL ;
aVariableDescriptions1 [ 0 ] . dwTypeId := 1 ;
aVariableDescriptions1 [ 0 ] . sTag := 'OUT_Outer_tank_supply_input_pump' ;
aVariableDescriptions1 [ 0 ] . sUnit := 'BOOL' ;

aVariableDescriptions1 [ 1 ] . pAddress := ADR ( Tank_control_POU_ST .
Output_Water_Pump ) ;
aVariableDescriptions1 [ 1 ] . eValueType := WagoAppCloud . VVT_BOOL ;
aVariableDescriptions1 [ 1 ] . dwTypeId := 2 ;
aVariableDescriptions1 [ 1 ] . sTag := 'OUT_Outer_tank_water_output_pump' ;
aVariableDescriptions1 [ 1 ] . sUnit := 'BOOL' ;

aVariableDescriptions1 [ 2 ] . pAddress := ADR ( GVL .
In_BOOL_Customer_HLevel_Inner_Tank ) ;
aVariableDescriptions1 [ 2 ] . eValueType := WagoAppCloud . VVT_BOOL ;
aVariableDescriptions1 [ 2 ] . dwTypeId := 3 ;
aVariableDescriptions1 [ 2 ] . sTag := 'Customer_HLevel_Inner_Tank' ;
aVariableDescriptions1 [ 2 ] . sUnit := 'BOOL' ;

aVariableDescriptions1 [ 2 ] . pAddress := ADR ( GVL . tank_level ) ;
aVariableDescriptions1 [ 2 ] . eValueType := WagoAppCloud . VVT_REAL ;
aVariableDescriptions1 [ 2 ] . dwTypeId := 4 ;
aVariableDescriptions1 [ 2 ] . sTag := 'Tank_level' ;
aVariableDescriptions1 [ 2 ] . sUnit := '%' ;

aVariableDescriptions1 [ 3 ] . pAddress := ADR ( GVL .
In_BTN_input_pump_Outer_Tank ) ;
aVariableDescriptions1 [ 3 ] . eValueType := WagoAppCloud . VVT_BOOL ;
aVariableDescriptions1 [ 3 ] . dwTypeId := 5 ;
aVariableDescriptions1 [ 3 ] . sTag := 'Status_Supply_input_pump_BTN' ;
aVariableDescriptions1 [ 3 ] . sUnit := 'BOOL' ;

aVariableDescriptions1 [ 4 ] . pAddress := ADR ( GVL .
In_BTN_Water_output_pump_Outer_Tank ) ;
aVariableDescriptions1 [ 4 ] . eValueType := WagoAppCloud . VVT_BOOL ;
aVariableDescriptions1 [ 4 ] . dwTypeId := 6 ;
aVariableDescriptions1 [ 4 ] . sTag := 'Status_Water_Output_Pump_BTN' ;
aVariableDescriptions1 [ 4 ] . sUnit := 'BOOL' ;



//-------------------------------------------------------------------------------
```

```
    aCollections [ 0 ] . dwCollectionId  :=  1 ;
    aCollections [ 0 ] . sName  :=  'Collection1' ;
    aCollections [ 0 ] . pSampleInterval  :=  ADR ( tSampleInterval1 ) ;
    aCollections [ 0 ] . pPublishInterval  :=  ADR ( tPublishInterval1 ) ;
    aCollections [ 0 ] . pVariableDescriptions  :=  ADR ( aVariableDescriptions1 ) ;
    aCollections [ 0 ] . dwVariablesCount  :=  5 ;


    //-----------------------------------------------------------------------
    (* call FbCollectionLogger  (WagoAppCloud)  with 2 collections*)
    oFbCollectionLogger ( pCollections  :=  ADR ( aCollections ) ,
                          dwCollectionsCount  :=  1 ) ;
```