MOTOROLA SOLUTIONS

RDS1048

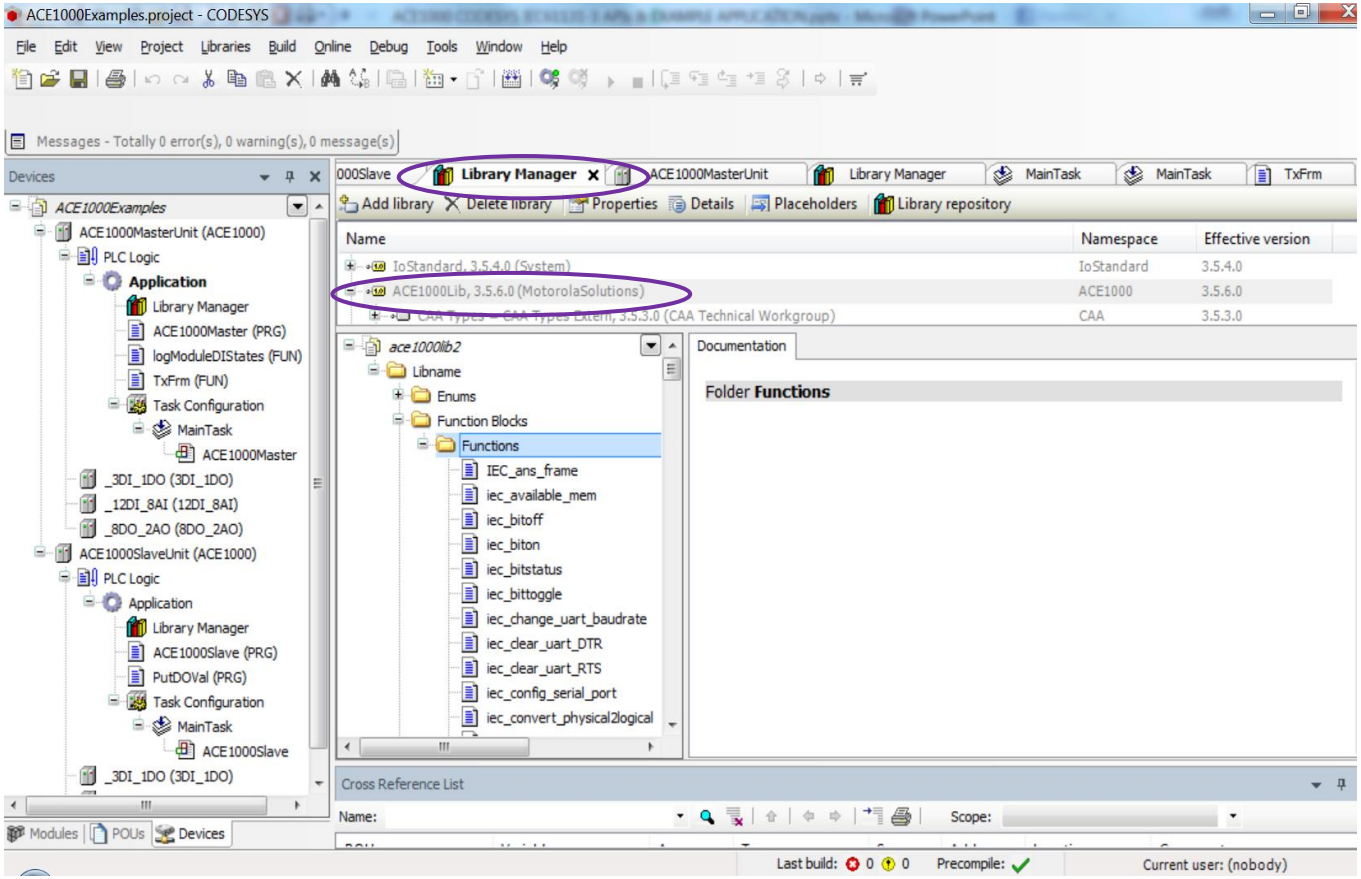
ACE1000 CODESYS IEC61131-3 APIs & EXAMPLE APPLICATION

LEARNING OBJECTIVES

ACE1000 CODESYS ‘IEC61131-3’ programming overview about:

- ACE1000 CODESYS APIs library.
- ACE1000 database access APIs.
- ACE1000 properties APIs.
- ACE1000 ‘MDLC’ communication APIs.
- ACE1000 user protocol communication APIs.
- ACE1000 3rd party protocols (DNP,MODBUS) APIs.
- ACE1000 IEC61131-3 mini ‘I/O’ application examples.
- ACE1000 IEC61131-3 ‘I/O and ‘MDLC” communication application example.

ACE1000 CODESYS ‘ACE1000Lib’ LIBRARY (version 3.5.6.0)



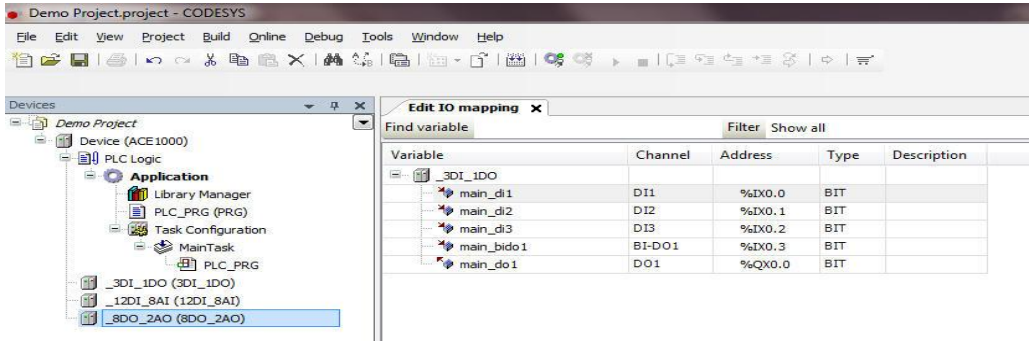
The library and its APIs are documented in the **‘ACE1000 IEC61131-3 Developer's Guide’**

ACE1000 DATABASE REGULAR SET OF TABLES

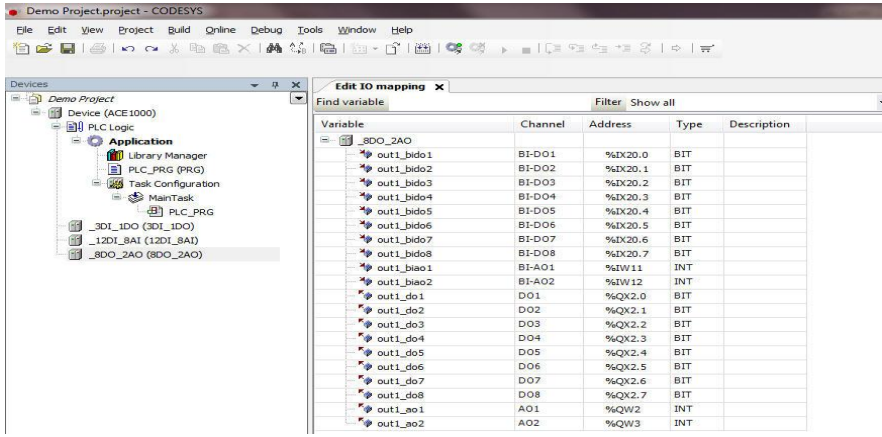
Table Index	Table Index	Number of values in table
System Values	0	11 (detailed in a later slides)
Discrete Inputs	2	Depends on the RTU's actual number of digital inputs
Discrete Outputs	3	Depends on the RTU's actual number of digital outputs
Analog Inputs	4	Depends on the RTU's actual number of analog inputs
Analog Outputs	5	Depends on the RTU's actual number of analog outputs
User Bits	6	2000 values
User Integers #1	7	2000 values
User Floats	8	1000 values
User Integers #2	9	2000 values
User Integers #3	10	2000 values
User Longs	11	2000 values

IEC61131-3 APPLICATION ACCESS TO ACE1000 RTU I/O VALUES (1)

- The ACE1000 RTU physical I/Os are:
  Digital inputs (**DI**)
  Digital outputs **(DO**) / back indications(**BI**)
  Analog inputs (**AI**)
  Analog outputs (**AO**) / back indications(**BI**)



- Access the ACE1000 physical I/O values from your application, using the logical variable names you have assigned them at the time of initial configuration I/O mapping.



For example:

IsDoorOpened**:=main_di1; Out1_ao1:=**100**;**

Note: the logical variable names can be modified at any time when working on your CODESYS project.

IEC61131-3 APPLICATION ACCESS TO ACE1000 RTU I/O VALUES (2)

ACE1000 'IEC' API Name	Description
iec_get_num_of_modules()	Read the actual number of I/O modules in the RTU (1-only main CPU module, 2-main and 1 expansion module, 3-main and 2 expansion modules)
iec_get_num_of_dis()	Read the number of digital inputs in a certain module
iec_get_num_of_dos()	Read the number of digital outputs in a certain module
iec_get_num_of_ais()	Read the number of analog inputs in a certain module
iec_get_num_of_aos()	Read the number of analog outputs in a certain module
Iec_get_di_val() / iec_get_ai_val()	Read the value of a certain digital / analog input in a certain module
iec_get_di_counter_val()	Read the value of a certain digital input counter in a certain module
iec_get_do_val() / iec_get_ao_val()	Read the value of a certain digital / analog output in a certain module
iec_get_do_bi_val()	Read the back indication value of a certain digital output in a certain module
iec_put_do_val() / iec_put_ao_val()	Write desired value to a certain digital / analog output in a certain module

IEC61131 APPLICATION ACCESS TO ACE1000 RTU SYSTEM VALUES (1)

System Values (direct access to database table #0)	Description
device_Is_Error_on	0 – No (no error) ; 1 – Yes (error exists)
device.LTR_Input_Voltage	Last reading of the input power level in mill volts
device.LTR_Temperature	Last measured temperature of the board in Celsius
device.Is_PB_on	0 – No (pushbutton not pressed) ; 1 – Yes (pushbutton pressed)
device.Conf_Change	0 – No (configuration not changed); 1– Yes (configuration changed)
device. Is_Init_Completed	0 – the unit has not completed its database buildup. 1– the unit has completed its database buildup and the user's application is allowed to access any desired variable in the database tables.

IEC61131 APPLICATION ACCESS TO ACE1000 RTU SYSTEM VALUES (2)

System Values (direct access to database table #0)	Description
device.IO_Main_Board_Status	0 – OK 1 – Comm. failure (between CPU and expansion/s 2 – Config. mismatch (actual expansion is different than the one specified in the configuration). 4 – General error. 255 – Module is not configured.
device.Io_Exp_1	Same as above
device.Io_Exp_2	Same as above
device. Actual_Io_Exp_1_Type	1 - module type unknown 2 - Inputs expansion module: 12 DIs & 8 AIs 3 - Outputs expansion module: 8 DOs & 2 Aos
device. Actual_Io_Exp_2_Type	1 - module type unknown 2 - Inputs expansion module: 12 DIs & 8 AIs 3 - Outputs expansion module: 8 DOs & 2 Aos

IEC61131 APPLICATION ACCESS TO ACE1000 RTU SYSTEM VALUES (3)

ACE1000 'IEC' API Name	Description
iec_get_v_in_level ()	Read the last reading of the input power level in mill volts (same as reading device.LTR_Input_Voltage)
iec_get_board_temperature ()	Read the last measured temperature of the board in Celsius (same as reading device.LTR_Temperature)
iec_get_io_status ()	Read the selected I/O board status (same as reading device.IO_Main_Board_Status / device.Io_Exp_1 / device.Io_Exp_2)
iec_is_error_exist ()	Read the status of the system error flag (same as reading device.Is_Error_on)
iec_is_PB_on ()	Read the status of the RTU's push button (same as reading device.Is_PB_on)
Iec_is_conf_changed ()	Read the status of the system configuration changed flag (same as reading device.Conf_Change)
iec_is_init_completed ()	Read the status of the system initialization completed (same as reading device.Is_Init_Completed)

IEC61131 APPLICATION ACCESS TO ACE1000 RTU USER VALUES

User Values	Data Type	Description
Read: MyBVal:=**device.UserValBit250;** Write: **device.UserValBit1**:=TRUE;	**BIT**	Direct access to database table#6 , 1st column's cells: UserValBit1 – UserValBit250
Read: MyIVal:=**device.UserValInt250;** Write: **device.UserValInt1**:=100;	**INTEGER**	Direct access to database table#7 , 1st column's cells: UserValInt1 – UserValInt250
Read: **iec_get_user_int_value** (row,column,&MyIVal) Write: **iec_set_user_int_in_table** (row,column,MyIVal)	**INTEGER**	API access to database table #7, #9 , #10 - all 8 column cells
Read: MyFloatVal:=**device.UserValFloat250;** Write: **device.UserValFloat1**:=2.5;	**FLOAT**	Direct access to database table#8 , 1st column's cells: UserVaFloat1 – UserVaFloat250
Read: **iec_get_user_long_value** (row,column,&MyLVal) Write: **iec_set_user_long_value** (row,column,MyLVal)	**LONG**	API access to database table #11 - all 8 column cells

IEC61131-3 APPLICATION – ACE1000 Properties APIs

API	Description
iec_product_type()	Returns ‘ACE1000’ as the unit’s product type.
iec_cpu_type()	Obtains ‘80’ as the ACE1000 CPU type number
iec_version()	Obtains the version of the ACE1000 firmware
iec_kernel_version()	Obtains the version of the ACE1000 kernel firmware
iec_system_build_number()	Obtains the ACE1000 firmware build number
iec_cpu_serial_number()	Obtains the serial number given to the CPU at factory
Iec_site_id()	Obtains the Site ID of the ACE1000 unit
Iec_get_hardware_board_type()	Obtain the unit’s board types: main / inputs / outputs

ACE1000 CODESYS ‘IEC61131-3’ APPLICATION

ACE1000 ‘IEC’ API	Description	Parallel ‘C’ Toolkit API
IEC_tx_frame()	Sends a frame to a peer RTU and expect a peer’s acknowledgement.	MOSCAD_SndFrm
IEC_tx_frame_noack()	Send a frame to a peer RTU without expecting a peer’s acknowledgement.	MOSCAD_TxFrm_siteid_linkid
IEC_rx_frame()	Receive a frame sent from peer RTU.	MOSCAD_RcvFrm_siteid_linkid
IEC_ans_frame()	Answers a previously received frame from a peer RTU.	
IEC_frameseq_tx_frame()	send a frame to a peer RTU in a sequenced manner.	MOSCAD_MDLC_txframe
IEC_frameseq_rx_frame()	Receive a frame sent from a peer RTU in a sequenced manner.	MOSCAD_MDLC_rxframe
IEC_tx_burst()	Send a burst of data to the ACE IP Gateway expecting it to acknowledge the burst.	MOSCAD_tx_burst
IEC_tx_burst_noack()	Send a burst of data to the ACE IP Gateway not expecting it to acknowledge the burst.	MOSCAD_tx_burst_noack

IEC61131-3 APPLICATION – ACE1000 Messages Logging APIs

During its run, an application may log applicative messages, warnings and errors in the RTU's 'Error Logger' .

ACE1000 'IEC' API Name	Description
iec_moscad_message ()	Logs a text message (which is not classified as error or warning) in the RTU's Error Logger.
iec_moscad_warning ()	Logs a warning message in the RTU's Error Logger.
iec_moscad_error ()	Logs an error message in the RTU's Error Logger.

IEC61131-3 APPLICATION – User Protocol APIs

- A serial ACE1000 port configured as a user port, allows an IEC61131-3 automation application to control the port and the protocol using related ‘iec_uport_...()’ APIs.

- The ACE1000 ‘User Port’ may be used to connect to a computer, a printer, or any other device that requires serial communication.

- These APIs allow an IEC61131-3 application to:
  - Initialize a configured user port
  - Set the port signals (DTR - Data Ready, CTS - Clear to Send)
  - Clear the port signals (DTR - Data Ready, CTS - Clear to Send)
  - Send a user protocol buffer to the port
  - Receive characters/buffers from the port

IEC61131-3 APPLICATION – DNP3 Communication APIs

The ACE1000 IEC61131-3 API library includes:

- **DNP Master services (**each API starts with ‘iec_dnp_’ prefix)**:**

  Allow the application to access and control each DNP remote Slave device and its elements.

- **DNP Slave services (**each API starts with ‘iec_dnp_slave_’ prefix)**:**

  Allow the application to handle DNP Slave data objects.

IEC61131-3 APPLICATION – DNP3 Master APIs

- The ACE1000 DNP3 Master supports the following data types (‘I/O points’):
  - Discrete Input / Discrete Output
  - Analog Input / Analog Output
  - Counter
  - Frozen counter

- All the **Master’s** I/Os are **virtual** (not physical) I/Os represented in the ACE1000 database.

- These ‘iec_dnp_...()’ APIs allow an application running in the ACE1000 to:
  - Get Master device and Slave devices properties and I/O quantities.
  - Perform an integrity/event poll of a Slave device (statuses and data)
  - Get a Slave device specific I/O data and status.
  - Set a Slave device specific I/O.
  - Control a certain Slave device cold/warm restart, counter freeze, etc.

IEC61131-3 APPLICATION – DNP3 Slave APIs

- The ACE1000 DNP3 Slave supports the following data types (‘I/O points’):
  - Discrete Input / Discrete Output
  - Analog Input / Analog Output
  - Counter
  - Frozen counter

- The **Slave** supports physical I/Os and virtual I/Os represented in the ACE1000 database.

- These ‘iec_dnp_slave_...()’ APIs allow an application running in the ACE1000 to:
  - Get a Slave device I/O quantities and properties .
  - Get a Slave device specific I/O data and status.
  - Set a Slave device specific I/O data.
  - Set a Slave device specific I/O data and generate a DNP3 event.

IEC61131-3 APPLICATION – MODBUS Master APIs

- Allow the application to access and control each MODBUS remote **Slave** device and its elements (each API starts with 'iec_modbus_' prefix).

Supported MODBUS Data Type	ACE1000 Equivalent
Logic Coils	Discrete Output
Discrete Inputs	Discrete Input
Holding Registers	Value Input (Read) / Value Output (Write)
Input Registers	Value Input

- These APIs allow an application running in an ACE1000 MODBUS **Master** to:
  - Poll a MODBUS Slave device
  - Get a Slave device MODBUS I/O data.
  - Set a Slave device MODBUS I/O data.
  - Get the last known communication status with a Slave device.

ACE1000 IEC61131-3 MINI 'I/O' APPLICATION Examples - I/O Mapping

Variable	Mapping	Channel	Address	Type	Unit	Description
di1		DI1	%IX0.0	BIT		
di2		DI2	%IX0.1	BIT		
di3		DI3	%IX0.2	BIT		
bido1		BI-DO1	%IX0.3	BIT		
do1		DO1	%QX0.0	BIT		

Main (CPU)

Variable	Mapping	Channel	Address	Type	Unit	Description
bido2		BI-DO1	%IX20.0	BIT		
bido3		BI-DO2	%IX20.1	BIT		
bido4		BI-DO3	%IX20.2	BIT		
bido5		BI-DO4	%IX20.3	BIT		
bido6		BI-DO5	%IX20.4	BIT		
bido7		BI-DO6	%IX20.5	BIT		
bido8		BI-DO7	%IX20.6	BIT		
bido9		BI-DO8	%IX20.7	BIT		
biao1		BI-AO1	%IW11	INT		
biao2		BI-AO2	%IW12	INT		
do2		DO1	%QX2.0	BIT		
do3		DO2	%QX2.1	BIT		
do4		DO3	%QX2.2	BIT		
do5		DO4	%QX2.3	BIT		
do6		DO5	%QX2.4	BIT		
do7		DO6	%QX2.5	BIT		
do8		DO7	%QX2.6	BIT		
do9		DO8	%QX2.7	BIT		
ao1		AO1	%QW2	INT		
ao2		AO2	%QW3	INT		

Outputs Module (DOs+AOs)

Variable	Mapping	Channel	Address	Type	Unit	Description
di4		DI1	%IX2.0	BIT		
di5		DI2	%IX2.1	BIT		
di6		DI3	%IX2.2	BIT		
di7		DI4	%IX2.3	BIT		
di8		DI5	%IX2.4	BIT		
di9		DI6	%IX2.5	BIT		
di10		DI7	%IX2.6	BIT		
di11		DI8	%IX2.7	BIT		
di12		DI9	%IX3.0	BIT		
di13		DI10	%IX3.1	BIT		
di14		DI11	%IX3.2	BIT		
di15		DI12	%IX3.3	BIT		
ai1		AI1	%IW2	INT		
ai2		AI2	%IW3	INT		
ai3		AI3	%IW4	INT		
ai4		AI4	%IW5	INT		
ai5		AI5	%IW6	INT		
ai6		AI6	%IW7	INT		
ai7		AI7	%IW8	INT		
ai8		AI8	%IW9	INT		

Inputs Module (DIs+AIs)

ACE1000 IEC61131-3 MINI 'I/O' APPLICATION Examples

Example#1:

```
IF di1=1 AND di2=1 AND di3=0 THEN //Check the values of the 3 DIs on the RTU's CPU board
    do9:=1;                       // set the last DO on the RTU's 1st outputs expansion
    ao1:=4095; END_IF             //Send command to AO1 to change value to 20MA/10V
```

Example#2:

```
IF (ai1>19640 AND ai1<19680) OR device.Is_PB_On=1 THEN  //if AI1 is between 19.98 and 20.02
    device.UserValInt100:=device.UserValInt1+1; END_IF  // increase user value #100 (in DB table 7) by 1
```

Example#3:

```
do3:=bido3;                        // Pre-requisite step for the next rule
IF di5 <> di5old AND di5=1 THEN // if di5 changed from the last IEC cycle and if it's turned on
   do3:=0; END_IF                  // set the 2nd DO on the RTU's 1st outputs expansion
di5old:=di5;                       // store the last value of di5 in di5old
```

‘ACE1000_Master_and_Slave_units-signaling_app’
IEC61131-3 ‘I/O and Rtu-to-Rtu Comm.’ Project – Example (1)

There are two ACE1000 devices, Master and Slave, each running an IEC61131-3 structured text application.

Device#1 -‘**ACE1000MasterUnit**’:

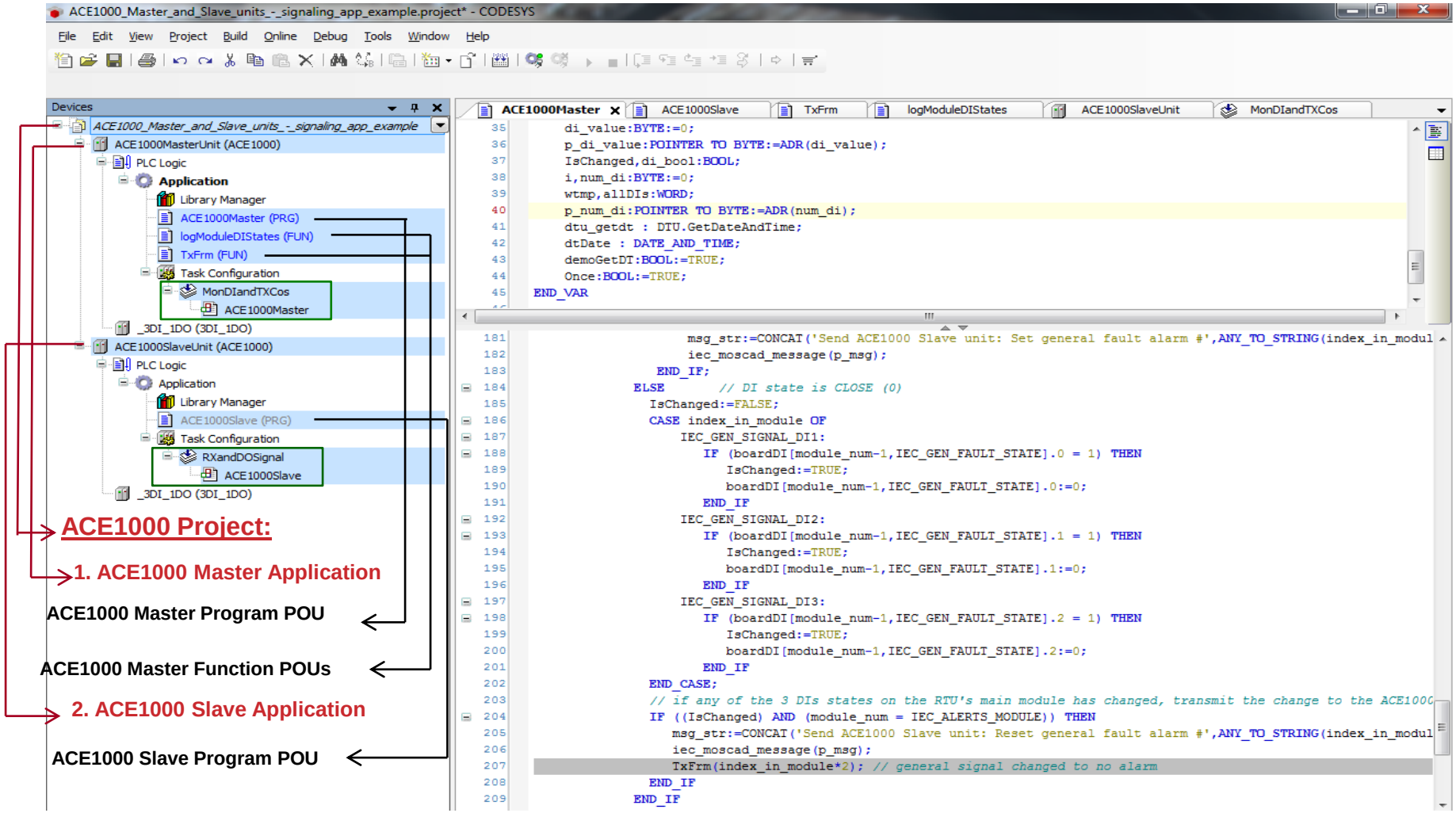
- logs the main board Serial Number in the RTU’s error logger
- logs the RTU date and time in the RTU's error logger
- identifies and logs all the RTU’s IO module types and the DI quantities and states in each IO module in the RTU’s error logger
- if any of the three DIs state on the RTU's main module is changed, transmits the change to the ACE1000 Slave RTU.

Device#2 -‘**ACE1000SlaveUnit**’:

- logs the main board Serial Number in the RTU’s error logger
- identifies and logs all the RTU’s IO module types and the DO quantities in each IO module in the RTU’s error logger
- set or reset one of the three DOs selected to signal the fault state received from the ACE1000 Master RTU

‘ACE1000_Master_and_Slave_units-signaling_app’

IEC61131-3 ‘I/O and Rtu-to-Rtu Comm.’ Project – Example (2)



‘ACE1000_Master_and_Slave_units-signaling_app’
IEC61131-3 ‘I/O and Rtu-to-Rtu Comm.’ Project – Example

API Name	ACE1000Master	ACE1000Slave	API Category
Iec_moscad_message()	+	+	Logging
Iec_moscad_warning()	+	+	Logging
Iec_moscad_error()	+	+	Logging
Iec_get_num_of_modules()	+	+	I/O values
Iec_get_hardware_board_type()	+	+	Unit properties
Iec_cpu_serial_number()	+	+	Unit properties
Iec_get_num_of_dis()	+	-	I/O values
Iec_get_di_val()	+	-	I/O values
IEC_tx_frame_noack()	+	-	RTU-to-RTU comm.
Iec_get_user_int_from_table()	+	-	User values
Iec_get_num_of_dos()	-	+	I/O values
Iec_get_do_val()	-	+	I/O values
Iec_put_do_val()	-	+	I/O values
IEC_rx_frame()	-	+	RTU-to-RTU comm.

‘ACE1000_Master_and_Slave_units-signaling_app’
IEC61131-3 ‘I/O and Rtu-to-Rtu Comm.’ Project – Example (4)

The required setup to run the example project and monitor its run:

Device	Minimal Requirement	Recommended
ACE1000 Master RTU	Main CPU with a communication port configured to communicate over RTU to RTU connection type	To observe the logging of I/O types and quantities not just on the CPU board, add one or two expansion boards, at least one of them is an inputs board.
ACE1000 Slave RTU	Main CPU with a communication port configured to communicate over RTU to RTU connection type	To observe the signaling of all three DIs by three DOs and the logging of I/O types and quantities not just on the CPU board, add one or two expansion boards, at least one of them is an outputs board.

LEARNING OBJECTIVES

- **ACE1000** CODESYS 'IEC61131-3' programming overview about:

  - ACE1000 CODESYS APIs library.
  - ACE1000 database access APIs.
  - ACE1000 properties APIs.
  - ACE1000 'MDLC' communication APIs.
  - ACE1000 user protocol communication APIs.
  - ACE1000 3rd party protocols (DNP,MODBUS) APIs.
  - ACE1000 IEC61131-3 mini 'I/O' application examples.
  - ACE1000 IEC61131-3 'I/O and 'MDLC" communication application example.

MOTOROLA SOLUTIONS

Thank you

KNOWLEDGE CHECK

1. The full collection of ACE1000 CODESYS APIs is documented in:
   a) CODESYS online help
   b) ACE1000 Easy Configurator manuel under ‘APPENDIX E: CODESYS IEC61131-3 PROGRAMMER APPLICATIONS‘
   c) All the answers apply
2. The CODESYS IEC61131-3 application may access values of I/Os, user and system in the ACE1000 data base tables in two ways:
   a) True
   b) False
3. The ACE1000 CODESYS library offers APIs for RTU to RTU communication solely without acknowledgment of the peer RTU:
   a) True
   b) False

KNOWLEDGE CHECK

4. The ACE1000 CODESYS library offers APIs to work with the following 3rd part protocols:
   a) DNP3
   b) MODBUS
   c) Both answers apply
5. The ACE1000 CODESYS library does not provide APIs for conducting communication over serial link with a computer, printer or other devices:
   a) True
   b) False