RL78 Family

DLMS User Guide

REJxxxxxxx-0100
Rev.1.00
March 13, 2014

Introduction

This document explains the way to integrate DLMS library to project.

Target Device

Energy Meter based on RL78 Family Device

Contents

1. How to integrate DLMS to project

1.1 How to append DLMS library to project

Step 1: Copy dlms folder and DLMSLib.lib to application folder in project folder like below figure.

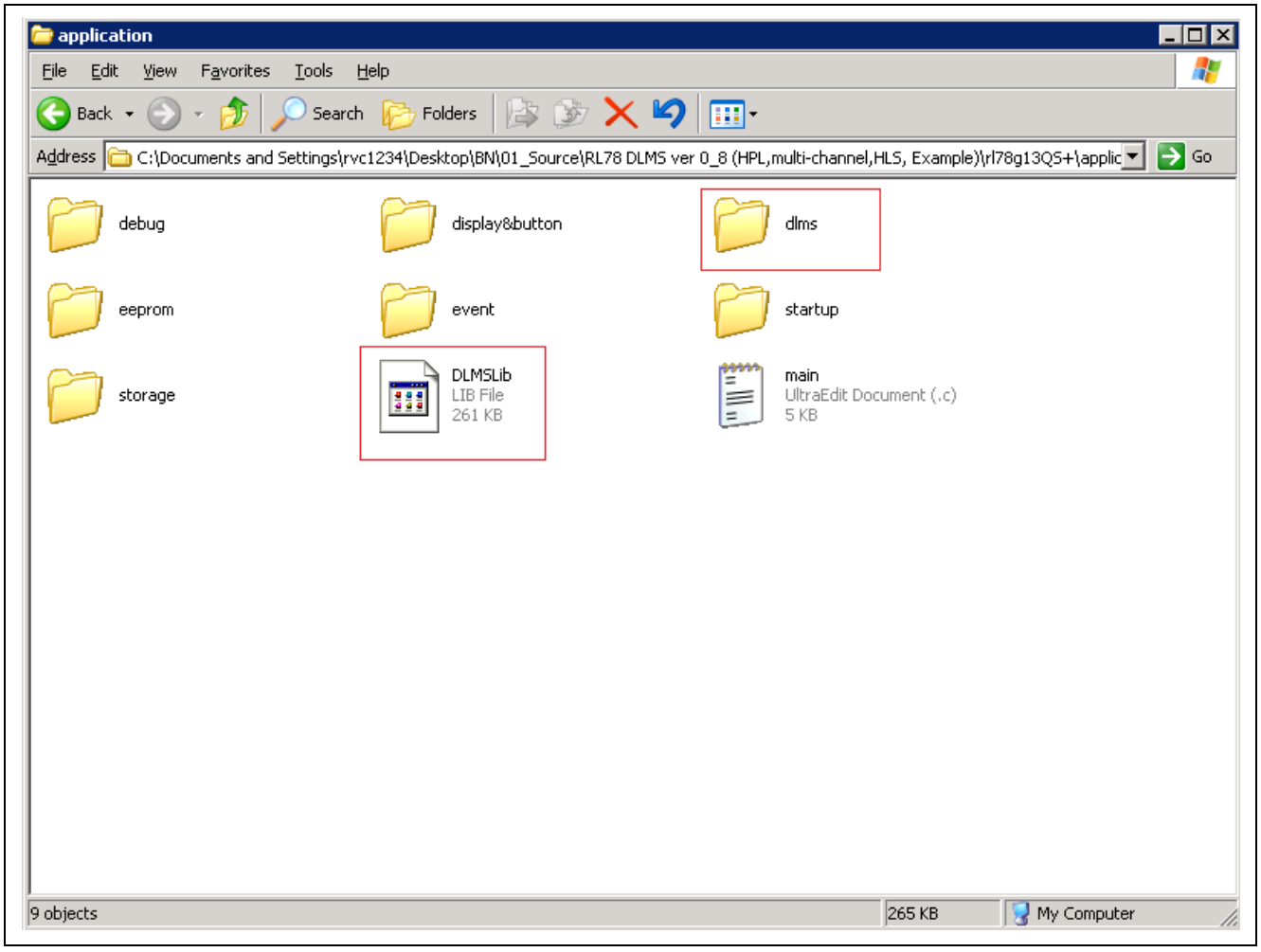


Figure 1 Copy dlms folder and DLMSLib.lib

Step 2: Use CubeSuite+ to open the project file. Then add all files and folders in dlms folder and DLMSLib.lib to project like below figure.

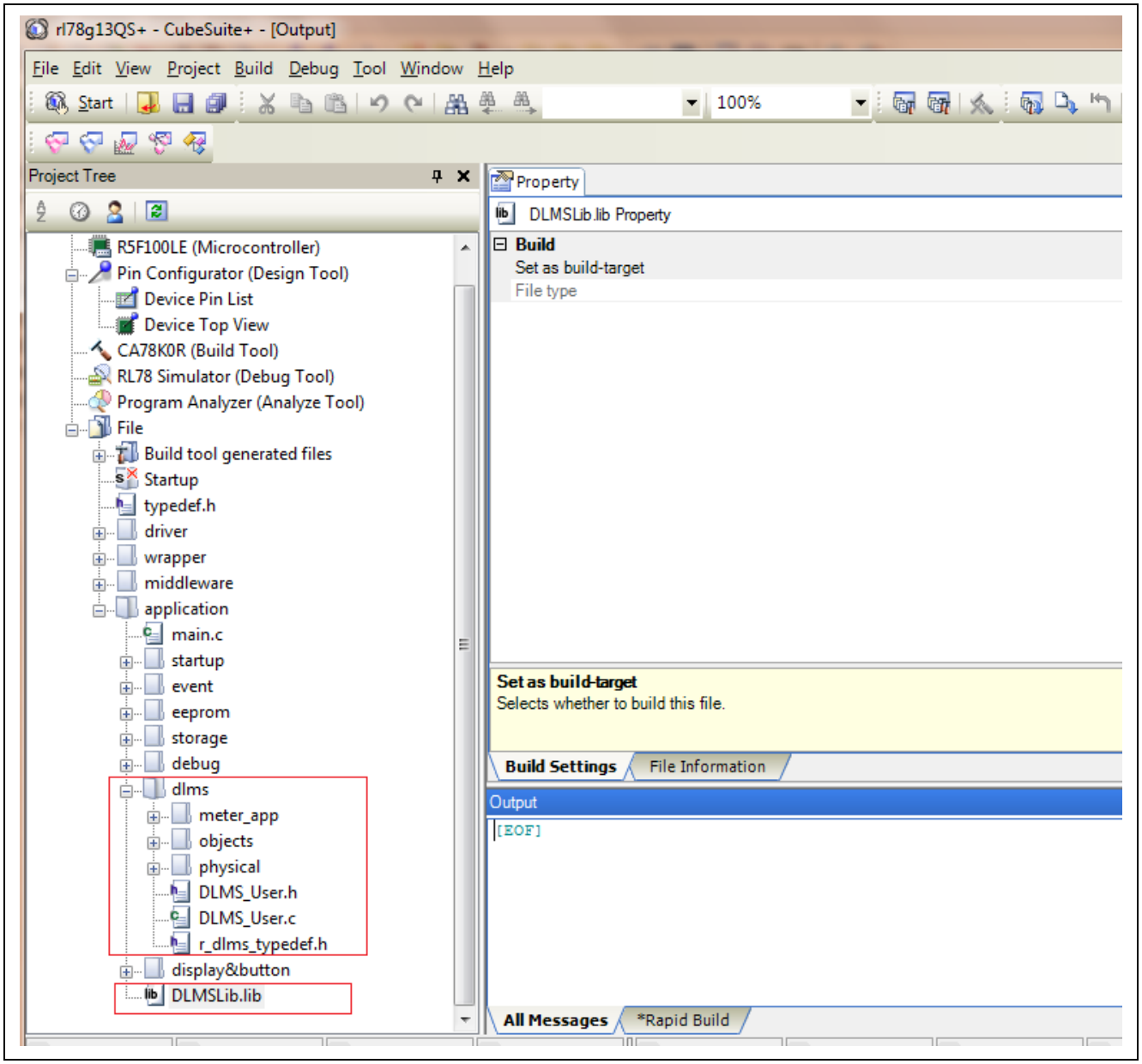


Figure 2 Add all files in dlms folder and DLMSLib.lib to project

1.2 How to link DLMS to main process

The example of basic operation for DLMS in main process is described as follow.

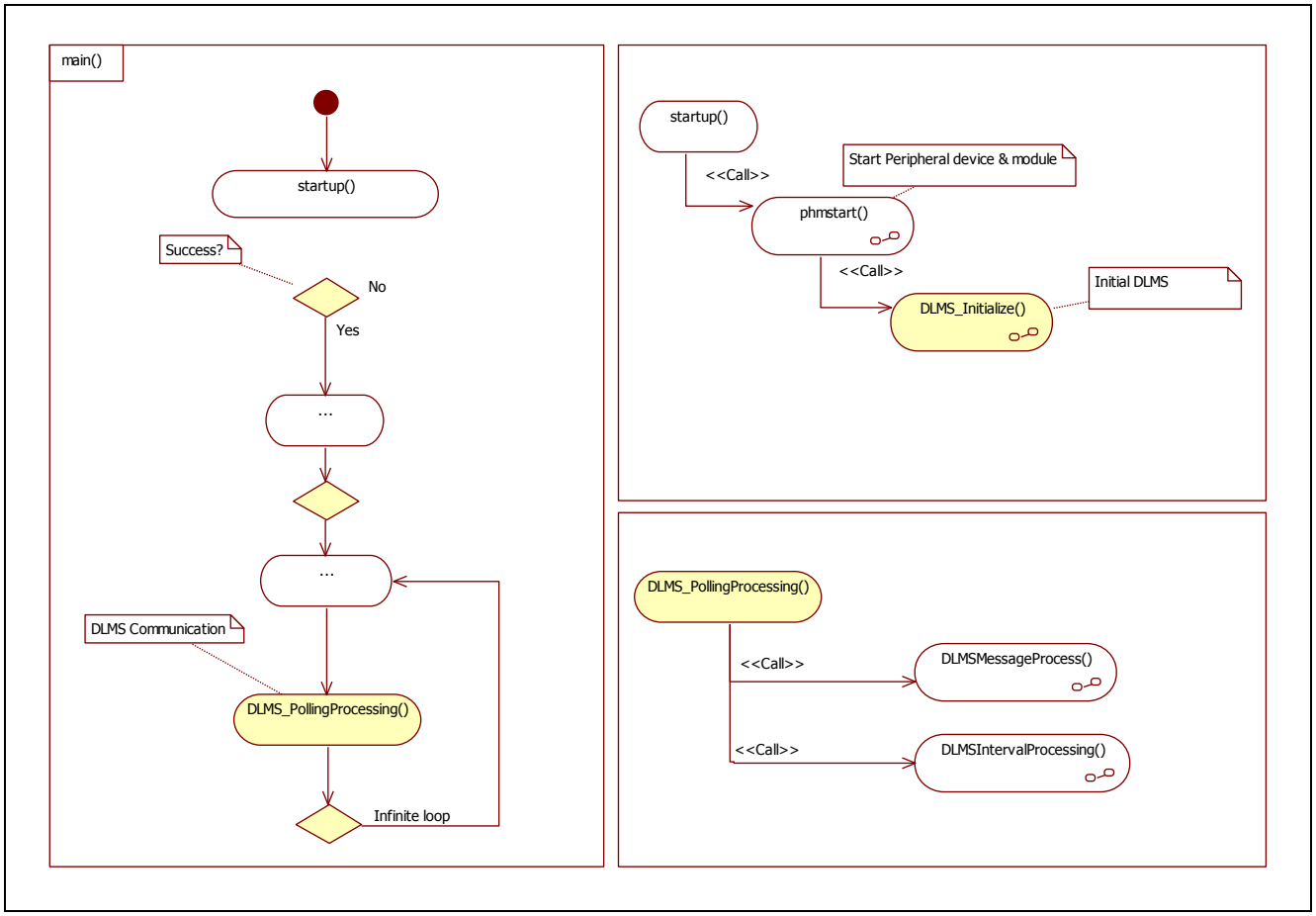


Figure 3 DLMS basic main operation

1.2.1 DLMS_PollingProcessing

Synopsis	DLMS Communication
Prototype	void **DLMS_PollingProcessing** (void);
Description	DLMS Communication - Decode the request form client. - Response to client based on the request.
Return value	None
Program example	Append this API to main() function (In \application\main.c)

```
#include "DLMS_User.h"

void main()
{
…
      /* Start-up OK */
      while (1)
      {
      …
       /* DLMS Communication */
       DLMS_PollingProcessing();
      …
      }
…
}
```

1.2.2 DLMS_Initialize

Synopsis	DLMS Initialization
Prototype	void **DLMS_Initialize** (void);
Description	DLMS Initialization - Initialize Physical layer. - Initialize Logical device. - Setup server station. - Initialize the stack library - Initialize Object layer
Return value	None
Program example	Append this API to start up process (In \application\startup\startup.c)

```
#include "DLMS_User.h"

static uint8_t phmstart(void)
{
…
/* Initial DLMS */
DLMS_Initialize();

/* Success */
return PHM_START_OK;
}
```

1.3 How to link DLMS to driver APIs

The basic interrupt operation for DLMS is described as follow.

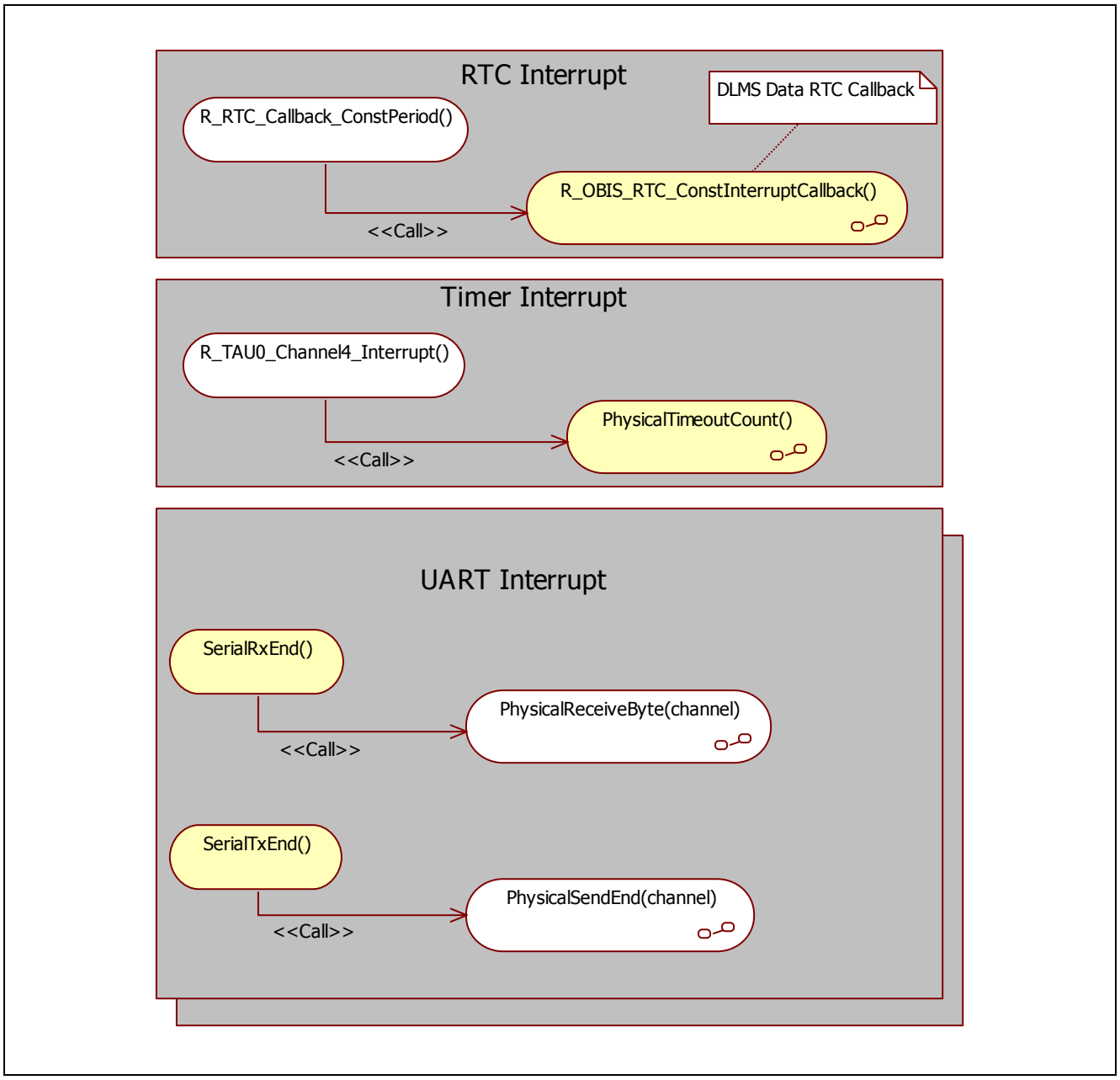


Figure 4 DLMS basic interrupt operation

Table 1 List of macros for driver APIs need to define by user

No.	Function name	Description
1	E2PR_OK	Return value of successful result
2	E2PR_READ	EEPROM read
3	E2PR_WRITE	EEPROM write

Example: In \dlms\meter_app\r_dlms_data_meter.h, define these macros as below:

```
/* EEPROM MW Layer */
#include "eeprom.h"

/* Wrapper defination for ouside memory access */
#define E2PR_OK                          EPR_OK
#define E2PR_READ(addr,buffer,size)      EPR_Read(addr,buffer,size)
#define E2PR_WRITE(addr,buffer,size)     EPR_Write(addr,buffer,size)
```

Below is list of functions in DLMS need to link to driver APIs:

1.3.1 R_OBIS_RTC_ConstInterruptCallback

Synopsis	RTC Callback for internal processing
Prototype	void **R_OBIS_RTC_ConstInterruptCallback** ( );
Description	RTC Callback for internal processing in DLMS object layer This function is call-back function, called by the meter for every interval period of RTC interrupt
Return value	None
Program example	Append this API to R_RTC_Callback_ConstPeriod (In \driver\mcu\r_rtc_user.c).

```
#include "r_dlms_data.h"

void R_RTC_Callback_ConstPeriod(void)
{
        /* Start user code. Do not edit comment generated here */
        EI();
        …
        /* DLMS Data RTC Callback */
        R_OBIS_RTC_ConstInterruptCallback();

        /* End user code. Do not edit comment generated here */
}
```

R_OBIS_Timer_ConstInterruptCallback();

1.3.2 TimerRCinit

Synopsis	Initialises RC timer
Prototype	Unsigned8 **TimerRCinit** (void);
Description	Initializes RC timer for DataLinkTimeoutCount()
Return value	None
Program example	Implement this API to connect to Timer driver (In \application\dlms\physical\Timer.c)

```
#include "r_cg_timer.h"

void TimerRCinit(void)
{
        R_TAU0_Channel4_Start();
}
```

1.3.3 PhysicalTimeoutCount

Synopsis	Physical time out callback
Prototype	Unsigned8 **CONNMGR_ChannelCount**(void);
Description	Physical 1ms time out callback
Return value	None
Program example	Append this API to timer driver (i.e: In driver\mcu\r_timer_user.c)

```
#include "DLMS_User.h"

__interrupt void R_TAU0_Channel4_Interrupt(void)
{
        /* Start user code. Do not edit comment generated here */
        EI();

        PhysicalTimeoutCount();
        /* End user code. Do not edit comment generated here */
}
```

1.3.4 InitSerial

Synopsis	Initialization of serial
Prototype	void **InitSerial** (Unsigned8 channel_id,);
Description	Initialization of UART unit to enable serial receive/transmit operations
Return value	None
Program example	Implement this API to connect to Timer driver (In \application\dlms\physical\serial.c)

```
#include "wrp_user_uart.h"

void InitSerial(Unsigned8 channel)
{
    switch (channel)
    {
        case CHANNEL_PRIMARY:
            /* Start UART */
            WRP_UART0_Init();
            WRP_UART0_Start();
            break;
        case CHANNEL_SECONDARY:
            /* Start UART */
            WRP_UART1_Init();
            WRP_UART1_Start();
            break;
        default:
            /* Do nothing */
            break;
    }
}
```

1.3.5 SerialTxEnd

Synopsis	Physical layer's callback of data transmit end through serial communication
Prototype	void **SerialTxEnd** (Unsigned8 channel_id,);
Description	Callback of data transmit end
Return value	None
Program example	Append this API to wrapper of UART driver for each channel (In \wrapper\user\wrp_user_uart.c)

```
#include "serial.h"

void WRP_UART0_SendEndCallback ()
{

        /* DLMS Transmit End */
        SerialTxEnd(CHANNEL_PRIMARY);
}
```

1.3.6 SerialRxEnd

Synopsis	Physical layer's callback of data receive end through serial communication
Prototype	void **SerialRxEnd** (Unsigned8 channel_id, Unsigned8 byte,);
Description	Callback of data Receive end
Return value	None
Program example	Append this API to wrapper of UART driver for each channel (In \wrapper\user\wrp_user_uart.c)

```
#include "serial.h"

void WRP_UART0_ReceiveEndCallback()
{

        /* DLMS Transmit End */
        SerialRxEnd(CHANNEL_PRIMARY, g_received_byte);

        /* Register to received next byte */
        WRP_UART0_ReceiveData(&g_received_byte, 1);
}
```

1.3.7 SerialTxBlock

Synopsis	Transmit block of data through serial communication
Prototype	void **SerialTxBlock** (Unsigned8 channel_id, Unsigned8 *BlockPtr // [Input] Pointer to block start address Integer16 Length // [Input] Length of the data in byte);
Description	Start serial transmit of block, complete until SerialTxEnd() callback is called
Return value	None
Program example	Implement this API to connect to UART driver for each channel (In \application\dlms\physical\serial.c)

```
#include "wrp_user_uart.h"

void SerialTxBlock(Unsigned8 channel, Unsigned8* BlockPtr, Integer16
Length)
{
        /* Start serial transmit */
        switch (channel)
        {
                case CHANNEL_PRIMARY:
                        WRP_UART0_SendData(BlockPtr, Length);
                        break;
                case CHANNEL_SECONDARY:
                        WRP_UART1_SendData(BlockPtr, Length);
                        break;
                default:
                        /* Do nothing */
                        break;
        }
}
```

1.3.8 SerialConfig

Synopsis	Reconfigure UART to adapt with new baud_rate,new protocol
Prototype	void **SerialTxBlock** (Unsigned8 channel_id, Unsigned8 new_baud_rate Unsigned8 new_protocol);
Description	Reconfigure UART to adapt with new baud_rate,new protocol
Return value	None
Program example	Implement this API to connect to UART driver (In \application\dlms\physical\serial.c)

```
#include "wrp_user_uart.h"

void SerialConfig(Unsigned8 channel, Unsigned8 new_baud_rate, Unsigned8
new_protocol)
{
        /* Set Baud rate of UART channel */
        if(new_baud_rate != BAUD_RATE_NOT_SPECIFIED)
        {
                        switch (channel)
                {
                        case CHANNEL_PRIMARY:
                                WRP_UART0_ChangeBaudRate(new_baud_rate);
                                break;
                        case CHANNEL_SECONDARY:
                                WRP_UART1_ChangeBaudRate(new_baud_rate);
                                break;
                        default:
                                /* Do nothing */
                                break;
                }
        }

        /* Reconfigure UART to adapt with new protocol */
        if(new_protocol == IEC_PROTOCOL)
        {
                switch (channel)
                {
                        case CHANNEL_PRIMARY:
                                WRP_UART0_ConfigIECProtocol();
                                break;
                        case CHANNEL_SECONDARY:
                                WRP_UART1_ConfigIECProtocol();
                                break;
                        default:
                                /* Do nothing */
                                break;
                }
        }
        else if(new_protocol == HDLC_PROTOCOL)
        {
                switch (channel)
                {
                        case CHANNEL_PRIMARY:
                                WRP_UART0_ConfigHDLCProtocol();
                                break;
                        case CHANNEL_SECONDARY:
                                WRP_UART1_ConfigHDLCProtocol();
                                break;
                        default:
                                /* Do nothing */
                                break;
                }
        }
        else
        {
                /* Do nothing */
        }
}
```

1.3.9 R_OBIS_GetRTCTime

Synopsis	Get RTC date time
Prototype	void **R_OBIS_GetRTCTime** (date_time_t * p_date_time // [Out] Buffer to store RTC date time);
Description	Get RTC date time This function is supported to get current date time of the meter
Return value	None
Program example	Implement this API to connect to RTC driver (In \application\dlms\meter_app\r_dlms_data_meter.c) `#include "r_cg_rtc.h"`

```
void R_OBIS_GetRTCTime (date_time_t *p_date_time)
{
        uint8_t  status;       /* Status of operation */
        uint16_t timeout;
       rtc_counter_value_t rtctime; /* RTC value for Driver IF */

        /* Check parameters */
        if (p_date_time == NULL)
        {
                return;
        }

        /* Set to by RTC driver IF */
        timeout = 0xFFFF;
        do
        {
                status = (uint8_t)R_RTC_Get_CounterValue(&rtctime);
        }
        while (status != MD_OK && timeout > 0);

        /* Check RTC driver error */
        if (status != MD_OK)
        {
                return;
        }

        /* Convert all to decimal */
        _BCD2DEC(rtctime.sec);
        _BCD2DEC(rtctime.min);
        _BCD2DEC(rtctime.hour);
        _BCD2DEC(rtctime.day);
        _BCD2DEC(rtctime.week);
        _BCD2DEC(rtctime.month);
        _BCD2DEC(rtctime.year);

        /* Convert to date_time_t structure */
       /* HIGH of Year */
        p_date_time->year_high      = (Unsigned8)(((Unsigned16)2000 +
rtctime.year) >> 8)    ;
       /* LOW  of Year */
        p_date_time->year_low       = (Unsigned8)((Unsigned16)2000 +
rtctime.year);
         p_date_time->month          = rtctime.month; /* Month */
        p_date_time->day_of_month   = rtctime.day;   /* Day */
        p_date_time->day_of_week    = g_DayOfWeek[rtctime.week];    /*
Friday */
        p_date_time->hour           = rtctime.hour;  /* Hour */
        p_date_time->minute         = rtctime.min;   /* Minute */
        p_date_time->second         = rtctime.sec;   /* Second */
       /* Not support */
        p_date_time->hundredths     = TIME_HUNDREDTHS_NOT_SPECIFIED;
       /* Initial at GMT +7 */
        p_date_time->deviation_high = (Unsigned8)(((Unsigned16)7 * 60) >>
8);
       p_date_time->deviation_low  = (Unsigned8)(((Unsigned16)7 * 60));
       /* Daylight saving active */
        p_date_time->clock_status   = CLOCK_STATUS_DAYLIGHT_SAVING_ACTIVE;
}
```

1.3.10 R_OBIS_SetRTCTime

Synopsis	Set RTC date time
Prototype	void **R_OBIS_SetRTCTime** (date_time_t date_time // [In] Date time value to set);
Description	Set RTC date time This function is supported to set current date time of the meter
Return value	None
Program example	Implement this API to connect to RTC driver (In \application\dlms\meter_app\r_dlms_data_meter.c)

```
#include "r_cg_rtc.h"

void R_OBIS_SetRTCTime (date_time_t date_time)
{
    uint8_t  i;      /* Counter */
    uint8_t  status; /* Status of operation */
    uint16_t timeout; /* Timeout counter to ensure the system not crash */
    rtc_counter_value_t rtctime;     /* RTC value for Driver IF */

    /* Get the RTC value from date_time */
    rtctime.day     = date_time.day_of_month;
    rtctime.month   = date_time.month;
    timeout         = ((Unsigned16)date_time.year_high << 8);
    timeout        += (Unsigned16)date_time.year_low;
    rtctime.year    = (Unsigned8)(timeout & 0xFF);
    rtctime.hour    = date_time.hour;
    rtctime.min     = date_time.minute;
    rtctime.sec     = date_time.second;
    for (i = 0; i < 7; i++)
    {
        if (g_DayOfWeek[i] == date_time.day_of_week)
        {
            break;
        }
    }
    if (i == 7)
    {
        i = 0;
    }
    rtctime.week    = i;

    /* Convert to BCD */
    _DEC2BCD(rtctime.day);
    _DEC2BCD(rtctime.month);
    _DEC2BCD(rtctime.year);
    _DEC2BCD(rtctime.hour);
    _DEC2BCD(rtctime.min);
    _DEC2BCD(rtctime.sec);
    _DEC2BCD(rtctime.week);

    /* Set to by RTC driver IF */
    timeout = 0xFFFF;
    do
    {
        status = (uint8_t)R_RTC_Set_CounterValue(rtctime);
    }
    while (status != MD_OK && timeout > 0);
}
```

1.3.11 R_OBIS_WDT_Restart

Synopsis	Watchdog timer restart
Prototype	void **R_OBIS_WDT_Restart** (void);
Description	Watchdog timer restart to avoid reset by watchdog
Return value	None
Program example	Implement this API to connect to WDT driver (In \application\dlms\meter_app\r_dlms_data_meter.c)

```
#include "r_cg_wdt.h"

void R_OBIS_WDT_Restart (void)
{
        R_WDT_Restart();
}
```

1.4 How to link DLMS to EM SDK

All of related functions and macros are in folder [\dlms\meter_app\]. Implement these APIs to connect to EM SDK.

Table 2 Related file for linking to EM SDK

No.	File name	Description
1	r_dlms_data_meter.h	Reference header file. All related API prototypes
2	r_dlms_data_meter.c	APIs need to implement to link to EM SDK

Table 3 List of functions for linking to EM SDK

No.	Function name	Description
1	R_OBIS_GetEMData	Get EM data
2	R_OBIS_SetEMData	Set EM data
3	R_OBIS_Event_Report	Report event and update event code and profile

1.4.1 R_OBIS_GetEMData

Synopsis	Get EM data from EM
Prototype	Float32 **R_OBIS_GetEMData** (Unsigned16 em_data,);
Description	Get EM data from EM em_data: please refer to “ID For EM data” in \dlms\meter_app\r_dlms_data_meter.h
Return value	Float32
Program example	Implement this API to get EM data (In \dlms\meter_app\r_dlms_data_meter.c)

```
#include "em_operation.h"      /* EM Core Operation APIs */
#include "em_calibration.h"    /* EM Calibration APIs */
#include "em_measurement.h"    /* EM Measurement APIs */

Float32 R_OBIS_GetEMData(Unsigned16 em_data)
{
  Float32 result = 0;
  switch(em_data)
  {
    case EM_ACTIVE_POWER:
          result = EM_GetActivePower();
          break;
    case EM_REACTIVE_POWER:
          result = EM_GetReactivePower();
          break;
    case EM_APPARENT_POWER:
          result = EM_GetApparentPower();
          break;
    case EM_LINE_FREQ:
          result =  EM_GetLineFrequency ();
          break;
    case EM_POWER_FACTOR:
          result = EM_GetPowerFactor();
          break;
    case EM_DEMAND_INTEGRATION_PERIOD:
          /* Exchange from minutes to seconds*/
          result = (Float32)(EM_GetConfig().max_demand_period * 60);
          break;
    /* And more … */
    default:
         break;
  }

  return result;
}
```

1.4.2 R_OBIS_SetEMData

Synopsis	Set EM data to EM
Prototype	Integer8 **R_OBIS_GetEMData** (Unsigned16 em_data, Unsigned8 *p_em_value);
Description	Set EM data to EM em_data: please refer to “ID For EM data” in \dlms\meter_app\r_dlms_data_meter.h *p_em_value: pointer to setting value for em_data
Return value	Integer8: 0: OK, -1: not supported, -2: error
Program example	Implement this API to set EM data (In \dlms\meter_app\r_dlms_data_meter.c)

```
#include "em_operation.h"      /* EM Core Operation APIs */
#include "em_calibration.h"    /* EM Calibration APIs */
#include "em_measurement.h"    /* EM Measurement APIs */

Integer8 R_OBIS_SetEMData(Unsigned16 em_data, Unsigned8 *p_em_value)
{
  Integer8 result = -1;
        switch(em_data)
        {
                /*
                * TODO: Put your code HERE to set em_value to EM
                * base on em_data, see "ID For EM data" on
                * r_dlms_data_meter.h
                */
                default:
                        break;
        }

        return result;
}
```

1.4.3 R_OBIS_Event_Report

Synopsis	Report event and update event code and profile
Prototype	void **R_OBIS_Event_Report** (Unsigned16 event_id,);
Description	Report event and update event code and profile event_id: please refer to “Event ID” in \dlms\objects\r_dlms_obis.h
Return value	none
Program example	Call this API to report event related to EM

```
#include "r_dlms_data_meter.h"

void EVENT_PowerFail(void)
{
        g_event_flag.is_power_fail = 1;
        R_OBIS_Event_Report(EVENT_ID_POWER_FAIL_OCCUR);
}
```

Note:

Supported profile object for event as below:

Table 54 Capture Parameters for Events

Sl. No.	Parameter	A	B	C	D	E	F	IC
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
i)	Date and Time of event	0	0	1	0	0	255	8 (Clock)
ii)	Event Code	0	0	96	11	0	255	1 (Data)
iii)	Current	1	0	94	91	14	255	3 (Register)
iv)	Voltage	1	0	12	7	0	255	3 (Register)
v)	Power Factor	1	0	13	7	0	255	3 (Register)
vi)	Cumulative Energy – kWh	1	0	1	8	0	255	3 (Register)

2.How to install new object

2.1 List of change files

For new object installation, list of changed file is described as below:

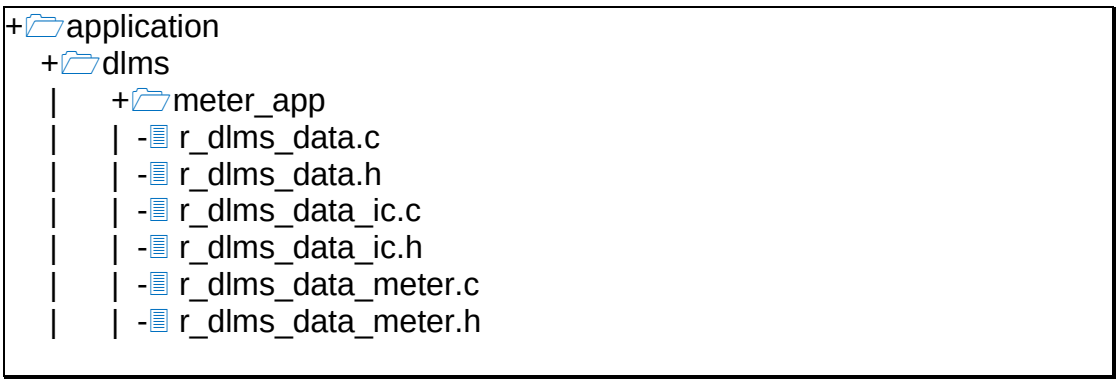
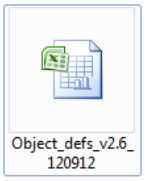


Figure 5 List of changed files

2.2 Preparation

Before adding new object for DLMS object layer, please specify the OBIS code of this object and its class.

The OBIS codes of each object are defined by DLMS UA. Please refer to below document for detail.



Link: http://dlms.com/documents/members/Object_defs_v2.6_120912.zip

2.3 Enable class usage

Before adding new object, please make sure its class is available and enable.

The list of supported classes is described in file r_dlms_data_ic.h.

For example: check the macro for class 07 is enabling:

```
/*
* The current version of DLMS support below classes
*
* Please specify which classes want to use
* 0 is no use, 1 is use
*/
#define USED_CLASS_01(1) /* Data */
#define USED_CLASS_03(1) /* Register */
#define USED_CLASS_04(1) /* Extended Register */
#define USED_CLASS_05(1) /* Demand Register */
#define USED_CLASS_06(0) /* Register activation */
#define USED_CLASS_07(1) /* Profile generic */
…
```

2.4 Add new object

To add new object for DLMS object layer, below files is related to:

Table 4 Related file for new object definition

No.	File name	Description
1	r_dlms_obis.h	Reference header file. All related structure definition for all OBIS classes.
2	r_dlms_obis_ic.h	Reference header file. All related type definition to construct the OBIS Object Layer.
3	r_dlms_data_ic.c	Append new OBIS objects definition.

For example: Add the object of class 07 in r_dlms_data_ic.c:

```
#if (defined(USED_CLASS_07) && USED_CLASS_07 == 1)
…
const class07_child_record_t g_ChildTableClass07[] =
{
/* Block Load Profile */
 {
  {1,0,99,1,0,255 },                    /* Logical name     */
  g_AccessRightTable[0],                /* Access right     */
  &g_class07_Obj0_Buf,                  /* Buffer           */
  g_Class07_Obj0_CaptureObjects,        /* Capture object list */
  &g_Class07_Obj0_CaptureObjectsLength, /* Capture object list length */
  &g_Class07_Obj0_CapturePeriod,        /* Capture period   */
  &g_Class07_Obj0_SortMethod,           /* Sort method      */
  &g_Class07_Obj0_SortObject,           /* Sort object      */
  &g_Class07_Obj0_EntriesInUse,         /* Entry in use     */
  &g_Class07_Obj0_ProfileEntries,       /* Profile entries */
 },
};
const Unsigned16 g_ChildTableLengthClass07 = sizeof(g_ChildTableClass07) /
sizeof(class07_child_record_t);
#endif
```

2.5 Encode/ decode attribute & execute action by User

All of modifiable functions are in r_dlms_data.c.

Table 5 List of functions for encode/ decode attribute by User

No.	Function name	Description
1	R_OBIS_GetObjectAttr	Get attribute for a specified object
2	R_OBIS_SetObjectAttr	Set attribute for a specified object
3	R_OBIS_ActionObjectMethod	Implement action for a specified object
4	R_OBIS_BufferScanByUserFunc	Scan to get info for filter
5	R_OBIS_BufferFilterByUserFunc	Filter a part of buffer

For example: Add the encode attribute process for object of class 01 (attribute 2, object 0) in R_OBIS_GetObjectAttr:

```
Unsigned8 R_OBIS_GetObjectAttr(
…
  else if (class_id == 1)
  {
    class01_child_record_t  *p_child_record;

    /* Get the child record */
    p_child_record = (class01_child_record_t *)(
       R_OBIS_FindChildRecordByIndex(class_id, child_id)
    );

    /* Attr 2 - value : CHOICE ? */
    if (attr_id == 2)
    {
       Unsigned16 u16;
       Integer16 size;
       void *buffer = NULL;

       /* Get the buffer pointer */
       switch (child_id)
       {
          case 0:     /* Logical Device Name */

             /*
              * TODO: Read logical device name from EEPROM here
              * Pass the pointer to buffer
              */
             buffer = "RES 5418 1";
             break;
…
  /* Encode & indicate as success when buffer is valid */
       if (buffer != NULL &&
          size != -1)
       {
       *p out len = R OBIS EncodeAttribute(
          p out buf,         /* Output buffer, pointed to g ServiceBuffer */
          OBIS SERVICE DATA BUFFER LENGTH, /* Max length of g_ServiceBuffer */
          p child record->value.choice.type, /* Type */
          (Unsigned8 *)buffer,              /* Buffer */
          size                            /* Length */
          );

          /* Success */
          response_result = DATA_ACCESS_RESULT_SUCCESS;
       }
…
```

3. How to append new channel

Appending 1 or more channels to physical layer is possible. Below is step-to-step guide line to append 2nd channel to DLMS physical layer.

3.1.1 Change max supported channel number

In \dlms\DLMS_User.h, change the max supported channel to 2 channels:

For example:

```
#define MAX_CONNMGR_CHANNEL_NUMBER          (2)
```

3.1.2 Specify channel ID

In \dlms\DLMS_User.h, specify the ID for new channel, make sure it has not same value with other channel ID or equal to 0xFF (CHANNEL_NOT_SPECIFIED):

For example: we define CHANNEL_SECONDARY as channel ID for new channel

```
/* ID of physical channel(s) */
#define CHANNEL_PRIMARY                     (0x00)
#define CHANNEL_SECONDARY                   (0x01)
```

3.1.3 Link driver APIs of new channel to physical layer

In \dlms\physical\serial.c, append driver API of new channel to below function to link new channel's driver with physical layer.

Table 6 List of driver API customized by User

No.	Function name	Description
1	InitSerial	Initialize for driver
2	SerialRxEnd	Receive end callback
3	SerialTxEnd	Transmit end callback
4	SerialTxBlock	Transmit function
5	SerialConfig	Configuration function

For example:

```
void SerialTxBlock()
{
    switch (channel)
    {
    case CHANNEL_PRIMARY:
        WRP_UART0_SendData(BlockPtr, Length);
        break;
    case CHANNEL_SECONDARY:
        WRP_UART1_SendData(BlockPtr, Length);
        break;
    default:
        /* Do nothing */
    break;
}
```

3.1.4 Add new channel to connection manager

Table 7 List of related functions for append channel to connection manager

No.	Function name	Description
1	CONNMGR_ChannelCount	Get registered channel number
2	CONNMGR_MaxChannelNumber	Get max supported channel number
3	CONNMGR_GetCurrentChannelID	Get current channel ID
4	CONNMGR_RegisterPriorityChannel	Register channel as priority channel based on channel ID
5	CONNMGR_AddChannel	Add 1 more channel for connection manager

CONNMGR_AddChannel () is supporting API used to add new channel for connection manager and must be called after calling DLMSInit().

The number of registered channels can checked by using CONNMGR_ChannelCount(), and maximum registered channels is the returned value of CONNMGR_MaxChannelNumber().

CONNMGR_RegisterPriorityChannel () is supporting API used to register priority channel that has priority to communicate with DLMS server than others.

Any further attempt to communicate while the priority channel is being used shall be returned as follow:

In NRM: switch state to NDM -> send DM
In NDM: always send DM

If this APIs not used, all channels have same priority, and follow FCFS rule (First-come, first-served).

For example: Add 2nd channel and register as priority channel for connection manager in DLMS_User.c:

```
void DLMS_Initialize(void)
{
        TimerRCinit();

        /* Initialize the stack library */
        DLMSInit(UserServerConfig);

        /* Add 1st channel */
        CONNMGR_AddChannel(
                CHANNEL_PRIMARY,                        /* Channel ID      */
                TxBuffer,                               /* Tx buffer start */
                MAX_TRANSMIT_BUFFER_SIZE,               /* Tx buffer size  */
                RxBufferMain,                           /* Rx buffer start */
```

```
                MAX_RECIEVE_BUFFER_SIZE                         /* Rx buffer size  */
        );

        if (CONNMGR_MaxChannelNumber() > 1)
        {
                /* Add 2nd channel */
                CONNMGR_AddChannel(
                        CHANNEL_SECONDARY,                      /* Channel ID      */
                        TxBuffer,                               /* Tx buffer start */
                        MAX_TRANSMIT_BUFFER_SIZE,               /* Tx buffer size  */
                        RxBufferSub,                            /* Rx buffer start */
                        MAX_RECIEVE_BUFFER_SIZE                 /* Rx buffer size  */
                );
        }
        /* Set priority channel */
        CONNMGR_RegisterPriorityChannel(CHANNEL_SECONDARY);
...
```

4. DLMS Object layer customize

4.1 Attribute get/set with selective methods

Current object layer implementation supports the attribute which can be accessed by many methods:

- Auto read from internal memory.
- Read through user-defined function (R_OBIS_GetObjectAttr function).
- Auto read from EEPROM.

By default, almost types are just support the access method of "auto read from internal memory". The type of value_t and status_t are used as CHOICE type described in Blue Book and support both method of "auto read from internal memory" and "read through user-defined function".

The type of dyn_value_t, dyn_status_t and dyn_date_time_t supports all of 3 access methods.

The summary of supported selective method of each type is described as follow.

Table 8 Supported selective method of each type:

No.	Type name	Auto read from internal memory	Read through user-defined function	Auto read from EEPROM
1	dyn_value_t	Supported	Supported	Supported
2	dyn_status_t	Supported	Supported	Supported
3	dyn_date_time_t	Supported	Supported	Supported
5	value_t	Supported	Supported	N/A
6	status_t	Supported	Supported	N/A
7	Others (*)	Supported	N/A	N/A

Note: (*) Exclude buffer_t type.

Example 1: Declare for attribute of dyn_value_t with "Auto read from internal memory" method.

```
Unsigned16  g_NumberPowerFailure_value = 8;
dyn_value_t g_NumberPowerFailure = DYN_VALUE(CHOICE_LONG_UNSIGNED, &g_
NumberPowerFailure_value,ATTR_ACCESS_MEMORY);
```

Example 2: Declare for attribute of dyn_value_t with "Read through user-defined function" method.

```
dyn_value_t g_NumberPowerFailure = DYN_VALUE(CHOICE_LONG_UNSIGNED, NULL,
ATTR_ACCESS_USERFUNC);
```

Example 3: Declare for attribute of dyn_value_t with "Auto read from EEPROM" method.

```
dyn_value_t g_NumberPowerFailure = DYN_VALUE(CHOICE_OCTET_STRING(-1) , NULL,
ATTR_ACCESS_E2PR(EM_SERIAL_START_ADDR, EM_SERIAL_SIZE_INBYTE));
```

4.2 Buffer asynchronous transfer process

Current object layer implementation supported the asynchronous transfer (or block transfer) for buffer attribute (buffer_t type). User just needs to modify the implementation in R_OBIS_BufferScanByUserFunc and R_OBIS_BufferFilterByUserFunc for attribute belongs to buffer_t type. For more information, please refer to r_dlms_data.c file.

R_OBIS_EncodeGenericBuffer function (in r_dlms_obis_ic.c) is used for handle this process and transparent to the user. The common process of this function is described as follow:

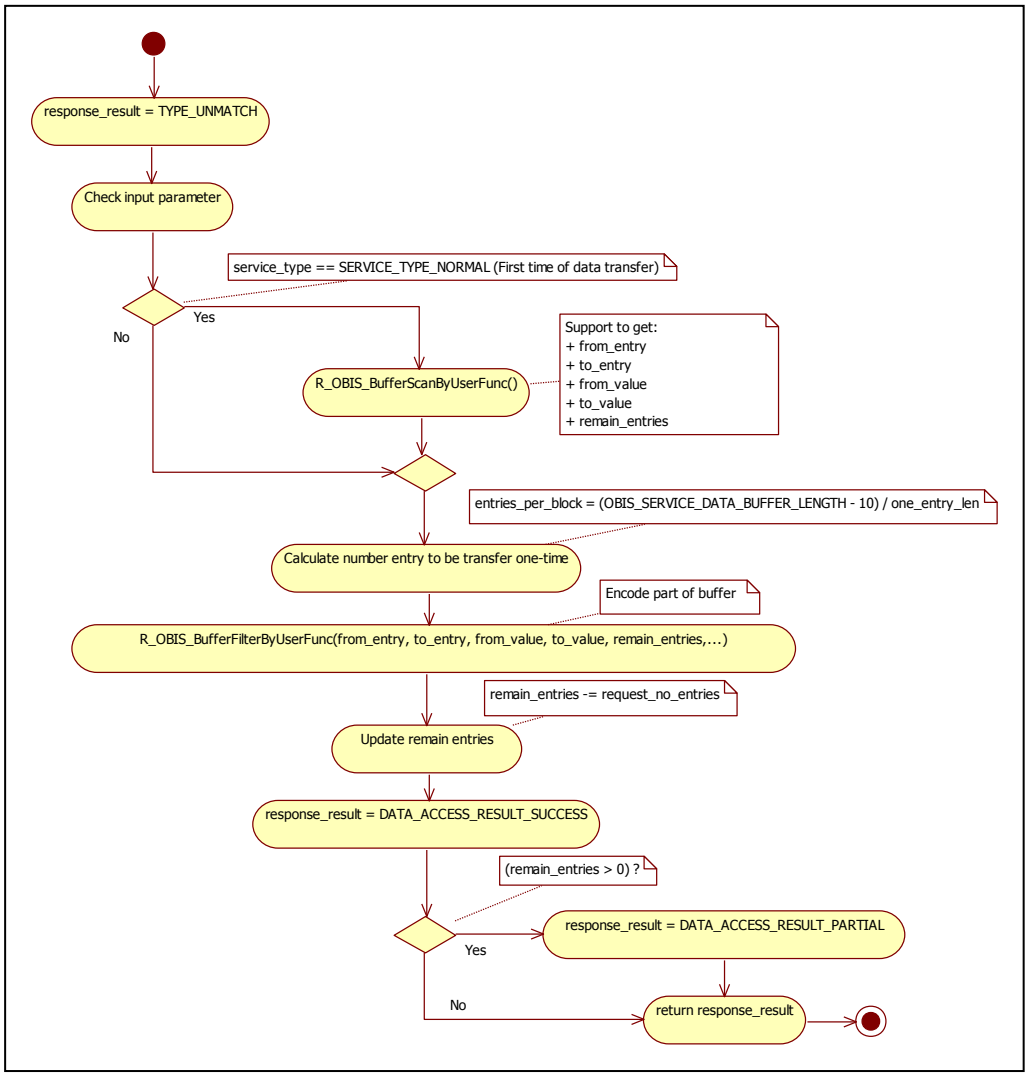


Figure 6 Processing of buffer asynchronous transfer

4.3 How to add profile object

Below is step-to-step guide line to append profile object to DLMS object layer.

Step 1:

Before adding new object, please make sure its class is available and enable. Please refer to 2.3 Enable class usage

Step 2:

In \dlms\meter_app\r_dlms_data.c, declare buffer definition

For example:

```
const scaler_unit_t g_Class07_DailyloadScalerBuffer[] =
{
        {0, PHY_UNIT_NONE              },      /* RTC time */
        {3, PHY_UNIT_WATT_HOUR         },      /* Cumulative Energy - kWh */
        {3, PHY_UNIT_VOLT_AMPERE_HOUR },      /* Cumulative Energy - kVAh */
};
```

Step 3:

In \dlms\meter_app\r_dlms_data_ic.c, declare all related structure for class07_child_record_t of profile object

For example:

```
#if (defined(USED_CLASS_07) && USED_CLASS_07 == 1)
...
/* Capture object list */
const class07_capture_object_t g_Class07_DailyloadCaptureObjects[] =
{
/*-------------------------------------------------------------------------------*/
/*        OBIS Code (A, B, C, D, E, F)       | Class ID     | Attr ID| Data ID */
/*-------------------------------------------------------------------------------*/
/* 00 */ {     {   0,   0,   1,   0,   0, 255 }, 8          , 2            , 0},   /* Clock */
/* 01 */ {     {   1,   0,   1,   8,   0, 255 }, 3          , 2            , 0},   /* Cumulative
Energy - kWh */
/* 02 */ {     {   1,   0,   9,   8,   0, 255 }, 3          , 2            , 0},   /* Cumulative
Energy - kVAh */
};
const Unsigned16 g_Class07_DailyloadCaptureObjectsLength = sizeof(g_Class07_DailyloadCaptureObjects)
/ sizeof(class07_capture_object_t);

buffer_t g_Class07_DailyloadScaler_Buf = {
        g_Class07_DailyloadScalerBuffer,      /* Buffer of value  */
        NULL,                                 /* Pointer to access method option */
        &g_Class07_FixCurrentEntry,           /* Pointer to index of current entries in buffer */
        0,                                    /* Number of entries per block to transfer */
        32,                                   /* Length of encode 1 entry after filter   */
        0,                                    /* Number of remain entries in run-time    */
        0,                                    /* First entry to retrieve in run-time     */
        0,                                    /* Last entry to retrieve in run-time      */
        0,                                    /* Index of first value to retrieve in run-time */
        0,                                    /* Index of last value to retrieve  in run-time */
};
```

Step 4:

In \dlms\meter_app\r_dlms_data_ic.c, add class07_child_record_t of profile object to g_ChildTableClass07

For example:

```
const class07_child_record_t g_ChildTableClass07[] =
{
…
        /* Daily Load Scaler Profile */
        {
        {   1,   0,   94,  91,   5, 255 }, /* Field 1. Logical name (OBIS code) of the object. */
         g_AccessRightTable[0],             /* Field 2. Access right definition for 1 object    */
        &g_Class07_DailyloadScaler_Buf,     /* Field 3. Buffer                                  */
        g_Class07_DailyloadCaptureObjects, /* Field 4. Capture object list                     */
        &g_Class07_DailyloadCaptureObjectsLength, /* Field 5. Capture object list length        */
        &g_Class07_NoCapturePeriod,            /* Field 6. Capture period                     */
        &g_Class07_SortMethod,                /* Field 7. Sort method                         */
        &g_Class07_SortObject,                /* Field 8. Sort object                         */
        &g_Class07_UniqueEntries,             /* Field 9. Entry in use                        */
        &g_Class07_UniqueEntries,             /* Field 10. Profile entries                    */
        },
```

Step 5:

In \dlms\meter_app\r_dlms_data.c, implement filter processing for buffer of profile object in function R_OBIS_Class07_BufferFilter()

For example:

```
Unsigned8 R_OBIS_Class07_BufferFilter (
…
while (total_entry > 0)
{
        switch(child_index)
        {
                        …
                case 4: /* Daily Load Scaler Profile */
                {
                length = R_OBIS_Class07_FilterOneDailyLoadEntry(buf, index, p_out_buf, p_out_len);
                }
                break;
…
}
```

Note: For implementation of R_OBIS_Class07_FilterOneDailyLoadEntry(), please refer to \dlms\meter_app\r_dlms_data_meter.c

5. How to customize for high level security

All of related functions and macros are in folder [\dlms\meter_app\]. Implement these APIs to connect to library of encryption/ decryption.

Table 9 Related file for linking to library of encryption/ decryption

No.	File name	Description
1	r_dlms_data.c.c	Implement R_OBIS_ActionObjectMethod() that process for reply_to_HLS_authentication() of Association LN class
2	r_dlms_data_ic.c	Define the Association LN object attributes
3	r_dlms_data_hls.c	APIs need to implement to link to EM SDK
4	r_dlms_data_hls.h	Reference header file.

Table 10 List of functions for linking to library of encryption/ decryption.

No.	Function name	Description
1	R_OBIS_Aes_Ecbenc	Wrapper for AES 128-bit Encryption Function (ECB Mode)
2	R_OBIS_ActionObjectMethod	Implement action for a specified object

5.1 Specify authentication mechanism, challenge StoC and shared key

In \dlms\meter_app\r_dlms_data_ic.c, configure security level and the challeng StoC, shared key

For example: Configure for Association 2

```
/* Class ID = 15, Association LN Child-table */
#if (defined(USED_CLASS_15) && USED_CLASS_15 == 1)
association_status_t g_ChildTableClass15_assc2_status = ASSC_STATUS_NON_ASSOCIATED;

/* for assc2 (HLS : Secret) */
authentication_status_t g_ChildTableClass15_assc2_authen_status = AUTHEN_STATUS_PENDING;

const class15_child_record_t g_ChildTableClass15[] =
{
…
/* Assc2 */
        {
                /* OBIS Code */
                {   0,   0,  40,   0,   3, 255 },
                /* Access right */
                g_AccessRightTable[3],
                /* Associated partners id                          */
                /*   client_SAP : unsigned                         */
                /*   server SAP : long-unsigned                    */
                {0x30, 0x0001},
                /* Context id of application context               */
                /* (Only select one of following)                  */
                CONTEXT_ID1_LN_NO_CIPHERING,
                /* Mechanism id of authentication mechanism name   */
                /* (Only select one of following)                  */
                MECHANISM_ID2_HIGH_SECURITY,
                /* Conformance block of association                */
                /* (Un-comment to enable features)                 */
                {
                        /* Byte 0 */
                        CONFORMANCE_BYTE0_NONE,

                        /* Byte 1 */
                        CONFORMANCE_BYTE1_LN_ATTR0_GET              |
                        //CONFORMANCE_BYTE1_LN_PRIORITY_MGMT          |
                        CONFORMANCE_BYTE1_LN_ATTR0_SET              |
                        CONFORMANCE_BYTE1_BLOCK_TRANSFER_GET        |
                        CONFORMANCE_BYTE1_BLOCK_TRANSFER_SET        |
                        //CONFORMANCE_BYTE1_LN_BLOCK_TRANSFER_ACTION  |
                        //CONFORMANCE_BYTE1_MULTI_REFERENCES          |
                        //CONFORMANCE_BYTE1_SN_INFORMATION_REPORT     |
```

```
                    CONFORMANCE_BYTE1_NONE,

                    /* Byte 2 */
                    //CONFORMANCE_BYTE2_SN_PARAMETERIZED_ACCESS    |
                    CONFORMANCE_BYTE2_LN_GET                      |
                    CONFORMANCE_BYTE2_LN_SET                      |
                    CONFORMANCE_BYTE2_LN_SELECTIVE_ACCESS         |
                    //CONFORMANCE_BYTE2_LN_EVENT_NOTIFICATION     |
                    CONFORMANCE_BYTE2_LN_ACTION                   |
                    CONFORMANCE_BYTE2_NONE,
                },

                /* DLMS version number                                */
                6,
                /* Secret key : LLS (password) or HLS (StoC)          */
                (Unsigned8 *)"P6wRJ21F",
                /* Shared key : HLS only                              */
                (Unsigned8 *)"ABCDEFGH",
                /* Association status                                 */
                &g_ChildTableClass15_assc2_status,

                /* Authentication status (required for HLS only)      */
                &g_ChildTableClass15_assc2_authen_status,

                /* Security setup reference                           */
                NULL,
        },
};
```

5.2 How to append AES library

Following is the example to append AES to project to work with DLMS

Step 1: Copy AES folder to \dlms\meter_app folder like below figure.

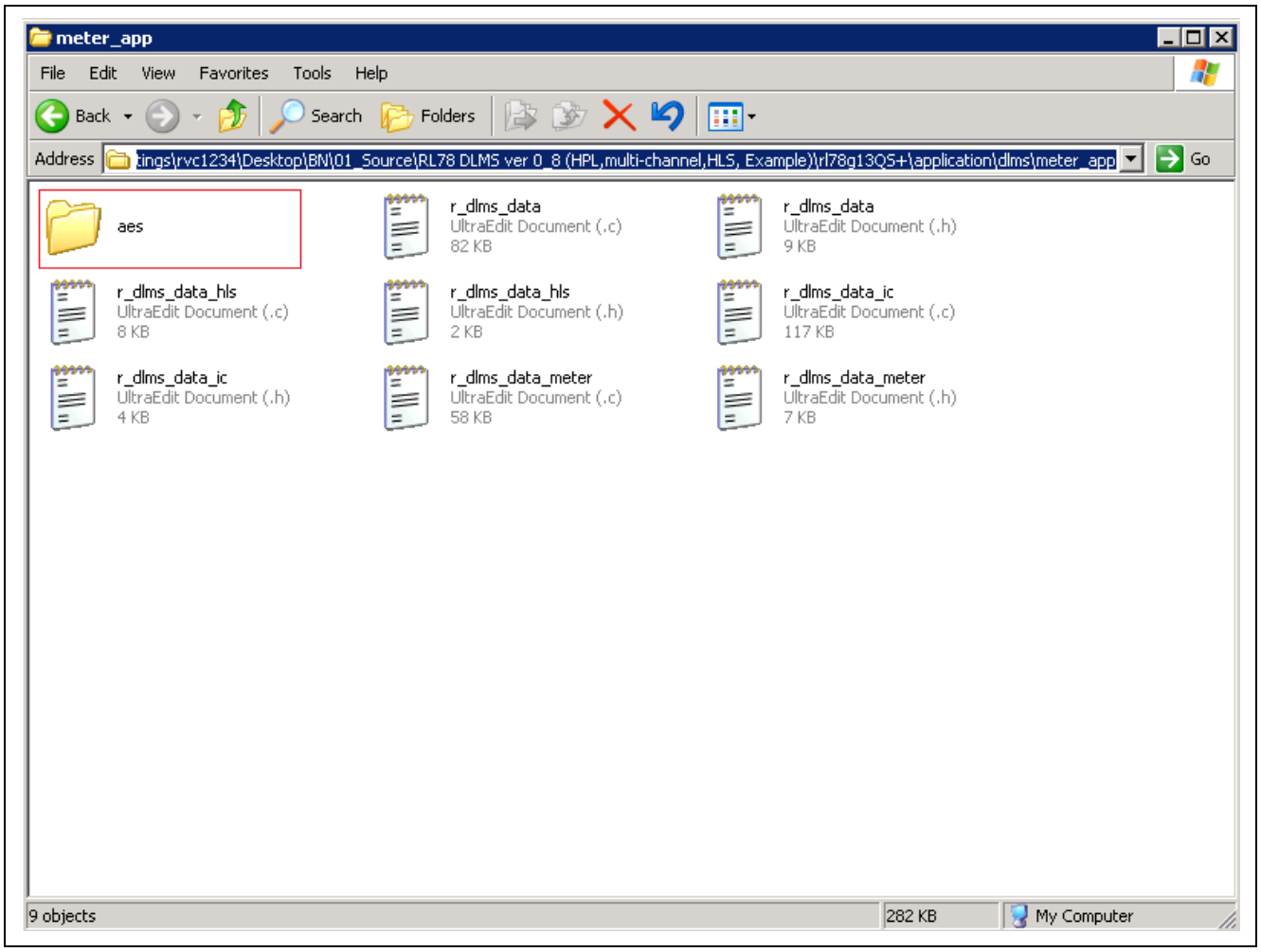


Figure 7 Copy aes folder

Step 2: Use CubeSuite+ to open the project file. Then add all files and library in aes folder to project like below figure.

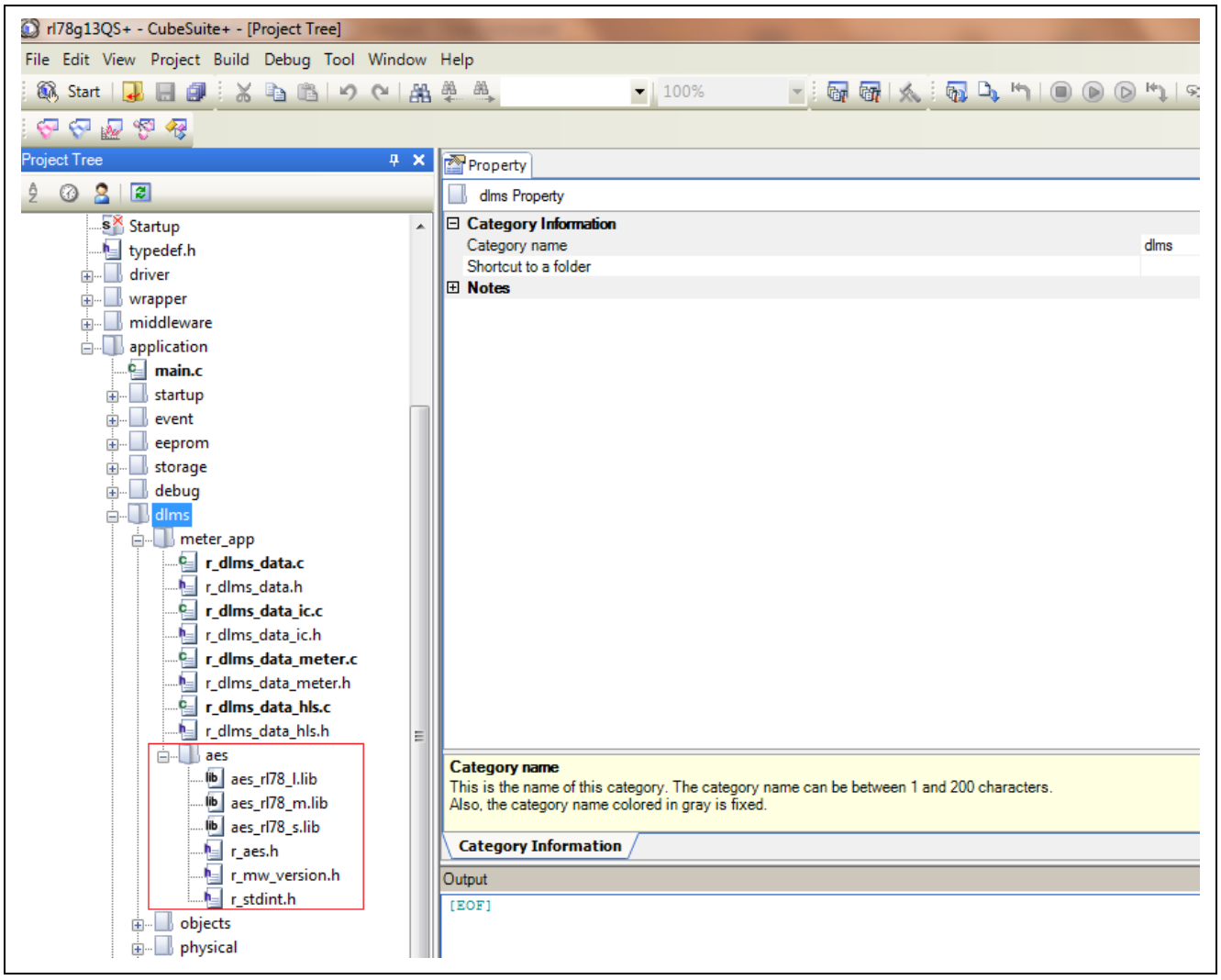


Figure 8 Add all files in aes folder to project

5.3 R_OBIS_Aes_Ecbenc

Synopsis	Wrapper for AES 128-bit Encryption Function (ECB Mode)
Prototype	Unsigned16 **R_OBIS_GetEMData** (Unsigned8 *in_ptext, Unsigned8 *out_ctext, Unsigned8 * in_key,);
Description	Wrapper for AES 128-bit Encryption Function (ECB Mode) Unsigned8 *in_ptext: plain text area Unsigned8 *out_ctext: cipher text area Unsigned8 *in_key : expanded key area
Return value	Unsigned16
Program example	Implement this API to connect to AES 128-bit Encryption Function (ECB Mode)

```
#include "r_aes.h"     /* AES 128-bit (ECB Mode) */

Unsigned16 R_OBIS_Aes_Ecbenc(Unsigned8 *in_ptext, Unsigned8 *out_ctext,
Unsigned8 *in_key)
{
        Unsigned16     length = 0;
        /*********************************************************************
         * Generate cipher key from in_key
        *********************************************************************/
        R_OBIS_Aes_Keysch(in_key);

        /*********************************************************************
         * Generate challenge from in_ptext
        *********************************************************************/
        length = R_OBIS_Aes_Challenge_generate(in_ptext);

        /*********************************************************************
         * TODO: put the custom method to encrypt data HERE
        *********************************************************************/
        R_Aes_128_Ecbenc((uint8_t near*)pdat128, (uint8_t near*)out_ctext,
ekey128, (length/AES_BLOCK_SIZE));

        return length;
}
```

6. Memory size

The memory usage varies with different configurations. The memory usage below is specified for DLMS components.

Table 11 Memory usage for DLMS components

No.	Module	ROM size (bytes)	RAM size (bytes)	Description
1	DLMS Server Protocol Stack Library	29368	1988	Include all layers. (*) Include 1 channel, class 15 with 3 objects as India spec.
2	Append 1 channel	0	120	Not include driver implementation for appended channel.
3	Data class	416	0	Include no object
4	Data object	17	0	Minimal value
5	Register class	261	0	Include no object
6	Register object	19	0	Minimal value
7	Extended Register class	429	0	Include no object
8	Extended Register object	37	0	Minimal value
9	Demand Register class	665	0	Include no object
10	Demand Register object	59	0	Minimal value
11	Register activation class	689	0	Include no object
12	Register activation object	23	0	Minimal value
13	Profile generic class	3645	0	Include no object
14	Profile generic object	24	0	Minimal value
15	Clock class	602	0	Include no object
16	Clock object	38	0	Minimal value
17	Script table class	868	0	Include no object
18	Script table object	11	0	Minimal value
19	Schedule class	858	0	Include no object
20	Schedule object	12	0	Minimal value
21	Special days table class	395	0	Include no object
22	Special days table object	12	0	Minimal value
23	Association LN class	2142	0	Include 4 association objects for India COSEM specs.
24	Association LN object	8	0	Minimal value
25	SAP assignment class	352	0	Include no object
26	SAP assignment object	11	0	Minimal value
27	Image transfer class	846	0	Include no object
28	Image transfer object	43	0	Minimal value
29	IEC local port setup class	426	0	Include no object

30	IEC local port setup object	32	0	Minimal value
31	Activity calendar class	1725	0	Include no object
32	Activity calendar object	36	0	Minimal value
33	Register monitor class	1181	0	Include no object
34	Register monitor object	16	0	Minimal value
35	Single action schedule class	927	0	Include no object
36	Single action schedule object	15	0	Minimal value
37	IEC HDLC setup class	415	0	Include no object
38	IEC HDLC setup object	24	0	Minimal value
39	IEC twisted pair class	500	0	Include no object
40	IEC twisted pair object	31	0	Minimal value
41	Utility tables class	1125	0	Include no object
42	Utility tables object	14	0	Minimal value
43	Modem configuration class	634	0	Include no object
44	Modem configuration object	18	0	Minimal value
45	Auto answer class	692	0	Include no object
46	Auto answer object	27	0	Minimal value
47	Auto connect class	683	0	Include no object
48	Auto connect object	22	0	Minimal value
49	Register table class	633	0	Include no object
50	Register table object	10	0	Minimal value
51	Status mapping class	470	0	Include no object
52	Status mapping object	13	0	Minimal value
53	Security setup class	362	0	Include no object
54	Security setup object	19	0	Minimal value
55	Disconnect control class	319	0	Include no object
56	Disconnect control object	28	0	Minimal value
57	Limiter class	1633	0	Include no object
58	Limiter object	58	0	Minimal value

(*) Note: In DLMS Server Protocol Stack Library, memory size for application side as below:

No.	Module	ROM size (bytes)	RAM size (bytes)	Description
1	Open service	4001	120	
2	Get service	2656	204	
3	Set service	4379	244	
4	Action service	626	8	

7. Time characteristic

Table 12 Fetching time

No.	Processing time	Min	Typical	Max	Unit	Description
1	RAM access (1byte)	1.44	1.46	157.265	us	Using memcpy()
2	RAM access (128 bytes)	48.82	48.85	204.625	us	Using memcpy()
3	EEPROM read (1page)	960	965	1566.8	ns	Using EPR_Read()
4	EEPROM read (128 bytes)	960	965	1575.75	ns	Using EPR_Read()
5	EEPROM write (1page)	960	965	1567.4	ns	Using EPR_Write()
6	EEPROM write (128 bytes)	960	965	1612	ns	Using EPR_Write()

Note: 1 EEPROM page = 32 bytes.

Table 13 UART transfer data time

No.	Processing time	Min	Typical	Max	Unit	Description
1	Transmit data (1 byte)	1.038	1.042	1.046	ms	Begin of SerialTxBlock() -> end of SerialTxEnd()
2	Receive data (1 byte)	1.039	1.041	1.045	ms	Begin of SerialRxEnd() -> begin of SerialRxEnd()

Note: At baud rate: 9600 bps, 8N1, LSB transmits first, without EM metrology.

Table 14 Decode data time

No.	Processing time	Typical	Unit	Description
1	Decode NULL data	2	CPU cycles	Not include checking parameter
2	Decode 1 byte data	23	CPU cycles	Not include checking parameter
3	Decode 2 bytes data	39	CPU cycles	Not include checking parameter
4	Decode 4 bytes data	69	CPU cycles	Not include checking parameter
5	Decode 8 bytes data	129	CPU cycles	Not include checking parameter
6	Decode variant size data (*)	108+27*N	CPU cycles	Not include checking parameter

Note: (*) Variant size data: octet string, bit string, time type, etc. with N is size in byte.

Table 15 Encode data time

No.	Processing time	Typical	Unit	Description
1	Encode NULL data	8	CPU cycles	Not include checking parameter
2	Encode 1 byte data	23	CPU cycles	Not include checking parameter
3	Encode 2 bytes data	39	CPU cycles	Not include checking parameter
4	Encode 4 bytes data	70	CPU cycles	Not include checking parameter
5	Encode 8 bytes data	134	CPU cycles	Not include checking parameter
6	Encode variant size data (*)	119+27*N	CPU cycles	Not include checking parameter

Note: (*) Variant size data: octet string, bit string, time type, etc. with N is size in byte.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Mar.13.14	All	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.2