RL78 Family

DLMS Object Layer User Manual

REJxxxxxxx-0100
Rev.1.00
July 17, 2013

Introduction

This document explains the usage of DLMS object layer.

Target Device

Energy Meter based on RL78 Family Device.

Contents

REFERENCES

- Blue_Book_10th_edition.pdf
- India COSEM specs.pdf
- Object_defs_v2.6_120912.xls

1. Overview

The software composition below shows DLMS object layer in relationship with other layers:

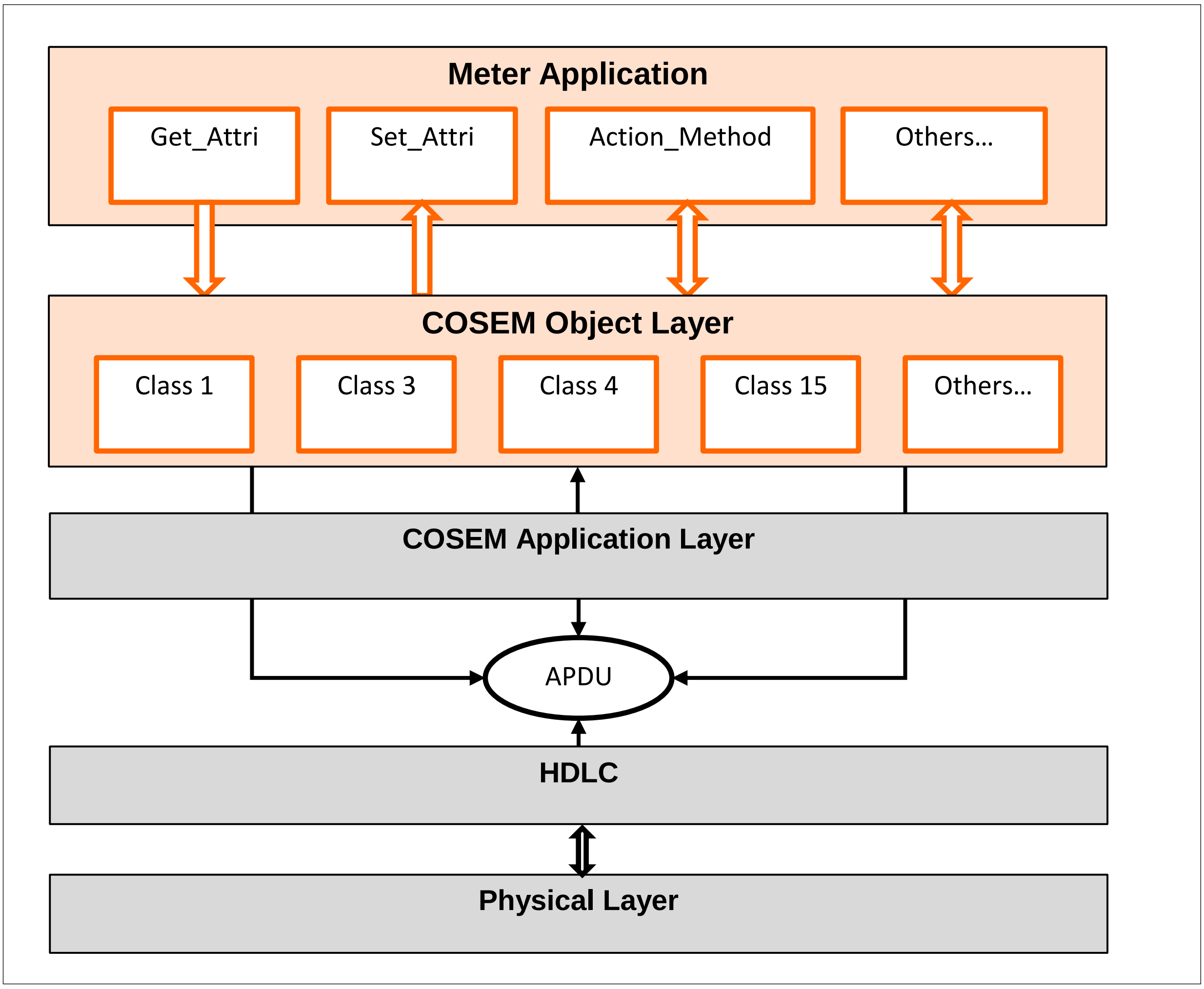


Figure 1 Software composition

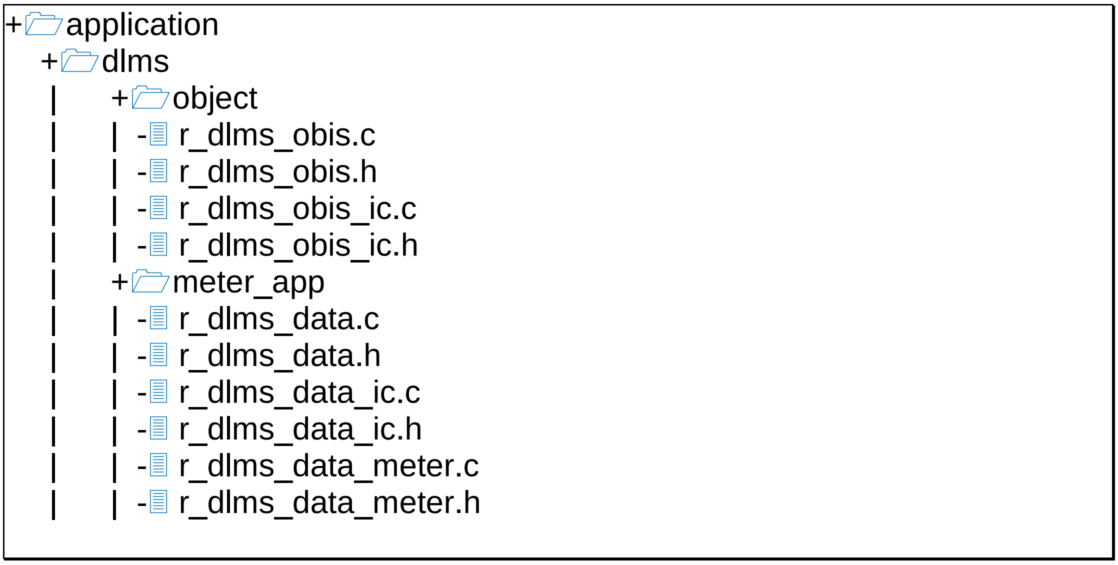
The implementation for DLMS object layer includes 2 parts:

- Meter Application: interface layer between object layer and meter data.
- COSEM Object Layer: implementation for all the classes in Blue Book.

2. Files/directories composition

The detail of DLMS object layer file structure is described as below:

- Meter Application: implemented as meter_app folder.
- COSEM Object Layer: implemented as object folder.



```
+application
   +dlms
    |     +object
    |     | -r_dlms_obis.c
    |     | -r_dlms_obis.h
    |     | -r_dlms_obis_ic.c
    |     | -r_dlms_obis_ic.h
    |     +meter_app
    |     | -r_dlms_data.c
    |     | -r_dlms_data.h
    |     | -r_dlms_data_ic.c
    |     | -r_dlms_data_ic.h
    |     | -r_dlms_data_meter.c
    |     | -r_dlms_data_meter.h
```

Figure 2 File structure of DLMS object layer

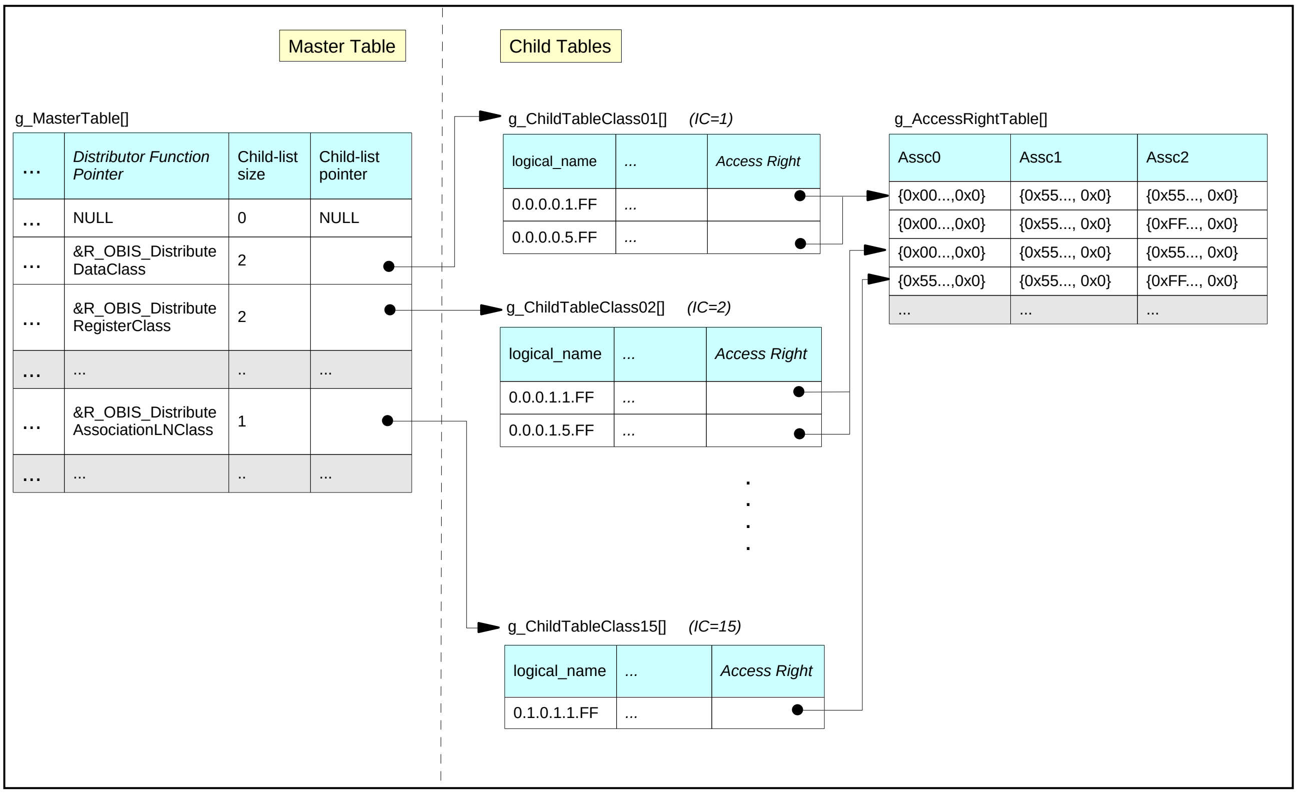
Table 1 File structure explanation

No.	File name	Description
1	r_dlms_obis.c	Master table source file. Implement master table for all OBIS classes.
2	r_dlms_obis.h	Master table definition header file. Declare all related structure definition for all OBIS classes.
3	r_dlms_obis_ic.c	Interface class source file. Implementation for all the classes in Blue Book.
4	r_dlms_obis_ic.h	Interface class header file. Declare all related type definition to the classes.
5	r_dlms_data.c	Meter data source file. Implemented by user/ customer to access the meter data.
6	r_dlms_data.h	Meter data header file. List all the interfaces in r_dlms_data.c.
7	r_dlms_data_ic.c	Interface class data source file. Object data definitions belong to classes.
8	r_dlms_data_ic.h	Interface class definition header file. Declare which classes want to use.
9	r_dlms_data_meter.c	Meter functions source file. Simulate parts of meter functions, easy for testing.
10	r_dlms_data_meter.h	Meter functions header file. List all the interfaces in r_dlms_data_ meter.c.

3. Implementation

3.1 Software structure

Based on structure of DLMS described in Blue Book, overview software structure of object layer implementation as following:


Master Table
Child Tables
g_MasterTable[]
g_ChildTableClass01[] (IC=1)
g_ChildTableClass02[] (IC=2)
g_ChildTableClass15[] (IC=15)
g_AccessRightTable[]

...	*Distributor Function Pointer*	Child-list size	Child-list pointer
...	NULL	0	NULL
...	&R_OBIS_Distribute DataClass	2	●
...	&R_OBIS_Distribute RegisterClass	2	●
...	...	..	...
...	&R_OBIS_Distribute AssociationLNClass	1	●
...	...	..	...

logical_name	...	*Access Right*
0.0.0.0.1.FF	...	●
0.0.0.0.5.FF	...	●

logical_name	...	*Access Right*
0.0.0.1.1.FF	...	●
0.0.0.1.5.FF	...	●

logical_name	...	*Access Right*
0.1.0.1.1.FF	...	●

Assc0	Assc1	Assc2
{0x00...,0x0}	{0x55..., 0x0}	{0x55..., 0x0}
{0x00...,0x0}	{0x55..., 0x0}	{0xFF..., 0x0}
{0x00...,0x0}	{0x55..., 0x0}	{0x55..., 0x0}
{0x55...,0x0}	{0x55..., 0x0}	{0xFF..., 0x0}
...	...	...

Figure 3 OBIS Objects software structure

3.2 Basic operation

The entry point of DLMS object layer is R_OBIS_DecodeObject function (on r_dlms_obis.c). This function called by DLMS application layer to access specific object’s attributes or execute object’s actions.

Each class has their own distributor function to response data to the request of GET/SET/ACTION service.

The basic operation for DLMS object layer is described as follow.

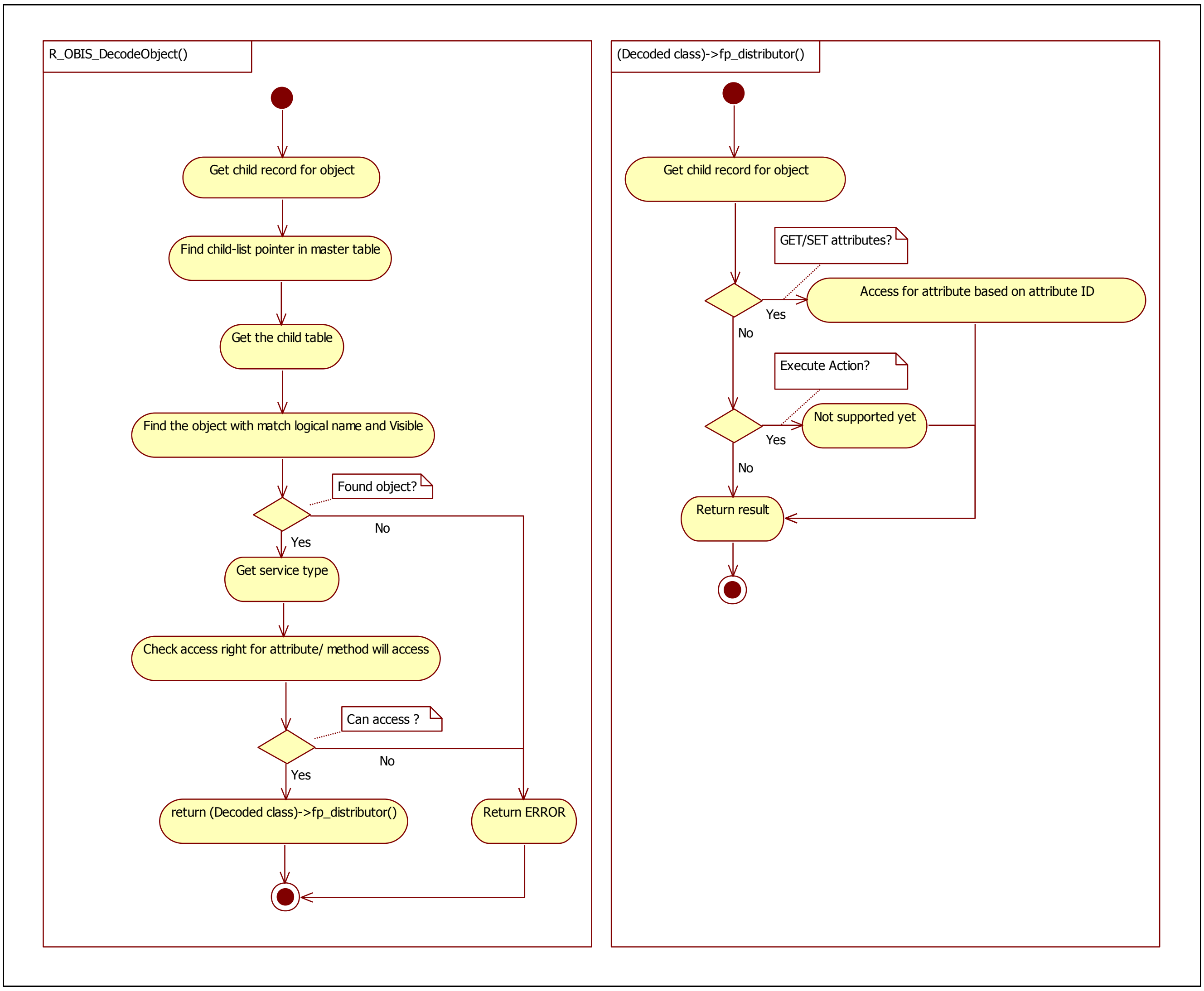

R_OBIS_DecodeObject()
Get child record for object
Find child-list pointer in master table
Get the child table
Find the object with match logical name and Visible
Found object?
No
Yes
Get service type
Check access right for attribute/ method will access
Can access ?
No
Yes
return (Decoded class)->fp_distributor()
Return ERROR
(Decoded class)->fp_distributor()
Get child record for object
GET/SET attributes?
Access for attribute based on attribute ID
Yes
No
Execute Action?
Not supported yet
Yes
No
Return result

Figure 4 OBIS Objects basic operation

3.3 Master table

The g_MasterTable holds the class information (class version, number of attributes, etc...) and a pointer to the child-list, which stores all the object information.

Each row of the table stores all below information:

- class_version: The version number of class.
- number_attrs: Number of attributes of the class.
- number_methods: Number of methods of the class.
- fp_distributor: Distributor function pointer of all class's objects.
- child_list_size: Size of the child-list.
- child_list_pointer: Child-list pointer.
- child_item_size: Size of 1 child item.

For example:

```
/* Master table */
const master_record_t g_MasterTable[] =
{
  /*
   *----------------------------------------------
   * "IC Name"
   * {
   *    Class Version, Number of attrs, Number of methods,
   *    Class Distributor Function,
   *    Child-list size,
   *    Child-pointer
   * }
   *----------------------------------------------
   */

  /* 0 - (reserved or empty) */
  {0, 0, 0, NULL, NULL, NULL, 0},

  /* 1 - Data */
  {
   0, 2, 0,
#if (defined(USED_CLASS_01) && USED_CLASS_01 == 1)
   R_OBIS_DistributeDataClass,
   &g_ChildTableLengthClass01,
   &g_ChildTableClass01,
   sizeof(class01_child_record_t),
#else
   NULL, NULL, NULL, 0,
#endif
  },
…
}
```

Note: when a class is reserved for future use or not implementation, it will be marked as following,

```
/* 0 - (reserved or empty) */
  {0, 0, 0, NULL, NULL, NULL, 0},
```

The length of the table is depended to the maximum number of classes that be supported by DLMS. Currently, in the 10th Edition of Blue Book, the maximum number is 74. So, we will implement this table just have 75 rows (g_MasterTable[index] with index from 0 to 74).

3.4 Child table

Each child table will implement for all objects (instances) of a specified class:

- g_ChildTableClass01: declare for all objects of Data class.
- g_ChildTableClass03: declare for all objects of Register class.
- g_ChildTableClass04: declare for all objects of Extend Register class.
- Etc...

And each row of above tables store all related information to control the object, as following:

- logical_name: The expected logical_name value of the object wants to be installed.
- p_access_right: Access right array for all implemented associations.
- And other attributes…

Please refer to Blue Book for more detail information about all classes.
For example:

```
const class10_child_record_t g_ChildTableClass10[] =
{
 /* Abtract Schedule object*/
 {
 {0, 0, 12, 0, 0, 255 }, /* Field 1. Logical name (OBIS code) of the object. */
 g_AccessRightTable[0],  /* Field 2. Access right definition for 1 object    */
 g_Class10_schedule_lists, /* Field 3. Specifies the different entries      */
 &g_Class10_nr_entries, /* Field 4. Number of entries                       */
 },
};
```

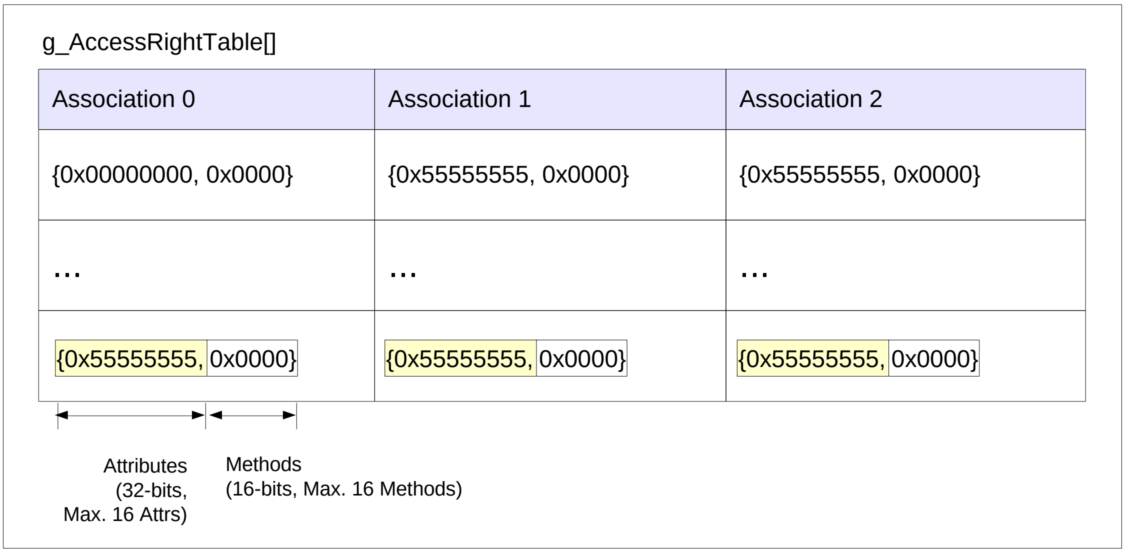
3.5 Access right table

Access right specifies the accessibility of all attributes & all methods in an object for a specified association. Two or more objects can have a same access right definition.

Visibility specify whether the object is visible or not for a specified association. If the object has no accessibility of all attributes & all methods, this object is not visible.

When we specify NULL for the access right pointer, it means the object is visible & full access (read, write for attribute, access for method) for all associations.

In current object layer, access right definition is implemented as below:



g_AccessRightTable[]

Association 0	Association 1	Association 2
{0x00000000, 0x0000}	{0x55555555, 0x0000}	{0x55555555, 0x0000}
…	…	…
{0x55555555, 0x0000}	{0x55555555, 0x0000}	{0x55555555, 0x0000}

Figure 5 Access right table

All attribute and method access rights are encoded in access_right_t type. Attributes encode from the least-significant bit, 2 bits for each attribute. Methods encode from the most-significant bit, 1 bit for each method.

The detail is as below:

Table 2 Data encoding of attribute access right

No.	Decimal	Binary	Meaning
1	0	00	No access
2	1	01	Read only
3	2	10	Write only
4	3	11	Read and write

Table 3 Data encoding of method access right

No.	Decimal	Binary	Meaning
1	0	0	No access
2	1	1	Access

For example, an extended register (IC 4) object has 5 attributes and 1 method.

— Attribute 1 (Logical Name)	: Read-only	(01)
— Attribute 2 (Value)	: Read-Write	(11)
— Attribute 3 (Scaler Unit)	: Read-only	(01)
— Attribute 4 (Status)	: Read-only	(01)
— Attribute 5 (Capture time)	: Read-only	(01)
— Method 1 (Reset)	: Execute	(1)

The above accesses are encoded as:

{0b00000000000000000000000101011101, 0b1000000000000000} = {0x0000015D, 0x8000}

3.6 Distributor Function

Distributor function sometime is called as service function. This is the class function to access the object and used to response data to the request of GET/SET/ACTION service from the protocol stack.

When the protocol stack finishes decode the request sent from client, it calls to the distributor function to access to the attribute or method of an object. The distributor functions then do some actions corresponding to the object, such as: read (get) / write (set) value for an attribute, execute a method (get, set, doing something), then response the encoded data back to the protocol.

The distributor function has below feature:

- Implement the GET/SET behavior (data access & storage) for an attribute of the current requested object.
- Implement the ACTION behavior for an ACTION request from Client.
- Response & encode (if any) the data as requirement of Green Book.

Each distributor function will implement for all objects (instances) of a specified class:

- R_OBIS_DistributeDataClass: implement for all objects of Data class.
- R_OBIS_DistributeRegisterClass: implement for all objects of Register class.
- R_OBIS_DistributeXRegisterClass: implement for all objects of Extend Register class.
- Etc…

3.7 Association LN

In this DLMS object layer, only Logical Name is supported.

Below child table implements 4 associations:

- g_ChildTableClass15

Current object layer implementation supports the following associations for India COSEM specs:

Table 4 Supported association objects:

No.	OBIS code	Interface Class	Description
1	0. 0. 40.0.0.255	Association LN (class id = 15)	Current Association
2	0. 0. 40.0.1.255	Association LN (class id = 15)	Public Client Association
3	0. 0. 40.0.2.255	Association LN (class id = 15)	Meter Reader Association
4	0. 0. 40.0.3.255	Association LN (class id = 15)	Utility Setting Association

The distribute function for these association objects is implemented by object layer, new object can be removed or added without any modification for this function of this class.

Note that the max number of association (exclude Current Association) is 3; this value is defined in r_dlms_data_ic.h as OBIS_NUMBER_ASSOCIATION and must not be changed.

For example: remove 1 object from g_ChildTableClass15 child table in r_dlms_data_ic.c:

```
const class15_child_record_t g_ChildTableClass15[] =

{

/*----------------------------------------------------------------------*/

/*          OBIS Code (A, B, C, D, E, F)     | Access Right            */

/*----------------------------------------------------------------------*/

/* 00 */ {{0,0,40,0,0,255}, g_AccessRightTable[4]},   /* Current Association */

/* 01 */ {{0,0,40,0,1,255}, g_AccessRightTable[4]},   /* Assc0 - Public Client
Association */

/* 02 */ {{0,0,40,0,2,255}, g_AccessRightTable[4]},   /* Assc1 - Meter Reader
Association */

/* 03 */ ~~{{0,0,40,0,3,255}, g_AccessRightTable[4]},~~   /* Assc2 - Utility Setting
Association */

};
```

3.8 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 5 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. Detail Implementation

4.1 Macro definitions

4.1.1 Common macros

Below macros define the supported classes in the current implementation. They are in r_dlms_data_ic.h.

Table 6 Common macros

No.	Macro name	Value	Description
1	USED_CLASS_01	0/1	Using Data class
2	USED_CLASS_03	0/1	Using Register class
3	USED_CLASS_04	0/1	Using Extended Register class
4	USED_CLASS_05	0/1	Using Demand Register class
5	USED_CLASS_06	0/1	Using Register activation class
6	USED_CLASS_07	0/1	Using Profile generic class
7	USED_CLASS_08	0/1	Using Clock class
8	USED_CLASS_09	0/1	Using Script table class
9	USED_CLASS_10	0/1	Using Schedule class
10	USED_CLASS_11	0/1	Using Special days table class
11	USED_CLASS_15	0/1	Using Association LN class
12	USED_CLASS_17	0/1	Using SAP assignment class
13	USED_CLASS_18	0/1	Using Image transfer class
14	USED_CLASS_19	0/1	Using IEC local port setup class
15	USED_CLASS_20	0/1	Using Activity calendar class
16	USED_CLASS_21	0/1	Using Register monitor class
17	USED_CLASS_22	0/1	Using Single action schedule class
18	USED_CLASS_23	0/1	Using IEC HDLC setup class
19	USED_CLASS_24	0/1	Using IEC twisted pair class
20	USED_CLASS_26	0/1	Using Utility tables class
21	USED_CLASS_27	0/1	Using Modem configuration class
22	USED_CLASS_28	0/1	Using Auto answer class
23	USED_CLASS_29	0/1	Using Auto connect class
24	USED_CLASS_61	0/1	Using Register table class
25	USED_CLASS_63	0/1	Using Status mapping class
26	USED_CLASS_64	0/1	Using Security setup class
27	USED_CLASS_70	0/1	Using Disconnect control class
28	USED_CLASS_71	0/1	Using Limiter class

Note: Supported classes in current implementation. Specify which classes want to use: 0 is no use, 1 is use.

4.1.2 Macros for date_t type

Table 7 Macros for date_t type

No.	Macro name	Value	Description
1	DATE_YEAR_NOT_SPECIFIED	0xFFFF	The year is not specified
2	DATE_MONTH_DAYLIGHT_SAVINGS_END	0xFD	Month is daylight savings end
3	DATE_MONTH_DAYLIGHT_SAVINGS_BEGIN	0xFE	Month is daylight savings begin
4	DATE_MONTH_NOT_SPECIFIED	0xFF	Month is not specified
5	DATE_DAY_2ND_LAST_OF_MONTH	0xFD	2nd last day of month
6	DATE_DAY_LAST_OF_MONTH	0xFE	Last day of month
7	DATE_DAY_NOT_SPECIFIED	0xFF	Day is not specified
8	DATE_DAY_RESERVED_START	0xE0	Reserved (start – end)
9	DATE_DAY_RESERVED_END	0xFC	from 0xE0 to 0xFC
10	DATE_WEEK_MON	0x01	Week date is Monday
11	DATE_WEEK_TUE	0x02	Week date is Tuesday
12	DATE_WEEK_WED	0x03	Week date is Wednesday
13	DATE_WEEK_THU	0x04	Week date is Thursday
14	DATE_WEEK_FRI	0x05	Week date is Friday
15	DATE_WEEK_SAT	0x06	Week date is Saturday
16	DATE_WEEK_SUN	0x07	Week date is Sunday
17	DATE_WEEK_NOT_SPECIFIED	0xFF	Week date is not specified

4.1.3 Macros for time_t type

Table 8 Macros for time_t type

No.	Macro name	Value	Description
1	TIME_HOUR_NOT_SPECIFIED	0xFF	Hour is not specified.
2	TIME_MINUTE_NOT_SPECIFIED	0xFF	Minute is not specified.
3	TIME_SECOND_NOT_SPECIFIED	0xFF	Second is not specified.
4	TIME_HUNDREDTHS_NOT_SPECIFIED	0xFF	Hundredths is not specified.

4.1.4 Macros for deviation element

Table 9 Macros for deviation element

No.	Macro name	Value	Description
1	DEVIATION_NOT_SPECIFIED	0x8000	Deviation is not specified.

4.1.5 Macros for clock_status_t type

Table 10 Macros for clock_status_t type

No.	Macro name	Value	Description
1	CLOCK_STATUS_INVALID_VALUE	0x01	Clock status set bit invalid_value
2	CLOCK_STATUS_DOUBTFUL_VALUE	0x02	Clock status set bit doubtful_value
3	CLOCK_STATUS_DIFFERENT_CLOCK_BASE	0x04	Clock status set bit different_clock_base
4	CLOCK_STATUS_INVALID_CLOCK_STATUS	0x08	Clock status set bit invalid_clock_status
5	CLOCK_STATUS_DAYLIGHT_SAVING_ACTIVE	0x80	Clock status set bit daylight_saving_active

4.2 Type definition

4.2.1 Primary type

Some common types for number as following:

Table 11 Common number's type

No.	Type name	GSCE Type	Definition	Size (bytes)	Description
1	Integer8	int8_t	signed char	1	Signed 1-byte character, -128...127
2	Integer16	int16_t	signed int	2	Signed 2-bytes integer, -32768…32767
3	Integer32	int32_t	signed long	4	Signed 4-bytes integer, -2147483648… 2147483647
5	Unsigned8	uint8_t	unsigned char	1	Unsigned 1-byte character, 0...255
6	Unsigned16	uint16_t	unsigned int	2	Unsigned 2-bytes integer, 0…65 535
7	Unsigned32	uint32_t	unsigned long	4	Unsigned 4-bytes integer, 0…4294967295
9	Float32	float32_t	float	4	IEEE 754 Floating point format, single-precision

4.2.2 Date and time formats

Date and time information may be represented with data type octet-string, or using the data types below:

4.2.2.1 date_t type

This is a structure type, little-endian. Total size is 5 bytes.

Table 12 date_t type

No.	Type name (Element)	Type	Description
1	year_high	Unsigned8	Interpreted as long-unsigned, Range is 0…0xFFFF-1. 0xFFFF mean not specified.
2	year_low	Unsigned8	
3	month	Unsigned8	Interpreted as unsigned. Normal range 1…12. 0xFF mean not specified.
4	day_of_month	Unsigned8	Interpreted as unsigned. Normal range 1…31. 0xFF mean not specified.
5	day_of_week	Unsigned8	Interpreted as unsigned. Normal range 1…7. 0xFF mean not specified.

4.2.2.2 time_t type

This is a structure type, little-endian. Total size is 4 bytes.

Table 13 time_t type

No.	Type name (Element)	Type	Description
1	hour	Unsigned8	Interpreted as unsigned, Range 0…23. 0xFF means not specified.
2	minute	Unsigned8	Interpreted as unsigned, Range 0…59. 0xFF means not specified.
3	second	Unsigned8	Interpreted as unsigned, Range 0…59. 0xFF means not specified.
4	hundredths	Unsigned8	Interpreted as unsigned, Range 0…99. 0xFF means not specified.

4.2.2.3 date_time_t type

This is a structure type, little-endian. Total size is 12 bytes.

Table 14 date_t type

No.	Type name (Element)	Type	Description
1	year_high	Unsigned8	Interpreted as long-unsigned, Range is 0…0xFFFF-1. 0xFFFF mean not specified.
2	year_low	Unsigned8	
3	month	Unsigned8	Interpreted as unsigned. Normal range 1…12. 0xFF mean not specified.
4	day_of_month	Unsigned8	Interpreted as unsigned. Normal range 1…31. 0xFF mean not specified.
5	day_of_week	Unsigned8	Interpreted as unsigned. Normal range 1…7. 0xFF mean not specified.
6	hour	Unsigned8	Interpreted as unsigned, Range 0…23. 0xFF means not specified.
7	minute	Unsigned8	Interpreted as unsigned, Range 0…59. 0xFF means not specified.
8	second	Unsigned8	Interpreted as unsigned, Range 0…59. 0xFF means not specified.
9	hundredths	Unsigned8	Interpreted as unsigned, Range 0…99. 0xFF means not specified.
10	deviation_high	Unsigned8	Range -720…720, in minutes of local time to GMT
11	deviation_low	Unsigned8	
12	clock_status	Unsigned8	The clock status for date time value

4.2.3 Common auxiliary data type

4.2.3.1 clock_status_t type

This is a structure type, little-endian. Total size is 1 byte.

Table 15 clock_status_t type

No.	Type name (Element)	Type	Description
1	invalid_value	Unsigned8 : 1	Indicate the time could NOT be recovered after an incident.
2	doubtful_value	Unsigned8 : 1	Indicate the time could be recovered after an incident but the value cannot be guaranteed.
3	different_clock_base	Unsigned8 : 1	Bit is set if the basic timing information for the clock at the actual moment is taken from a timing source different from the source specified in clock_base
4	invalid_clock_status	Unsigned8 : 1	This bit indicates that at least one bit of the clock status is invalid. Some bits may be correct.
5	reserved0	Unsigned8 : 1	Reserved (no-use)
6	reserved1	Unsigned8 : 1	Reserved (no-use)
7	reserved2	Unsigned8 : 1	Reserved (no-use)
8	daylight_saving_active	Unsigned8 : 1	Flag set to true: the transmitted time contains the daylight saving deviation (summer time). Flag set to false: the transmitted time does not contain daylight saving deviation (normal time).

4.2.3.2 physical_unit_t type

This is an enumeration type.

Table 16 physical_unit_t type

No.	Unit name	Value	Description
1	PHY_UNIT_YEAR	1	Year
2	PHY_UNIT_MONTH	2	Month
3	PHY_UNIT_WEEK	3	Week. 7*24*60*60 s
4	PHY_UNIT_DAY	4	Day. 24*60*60 s
5	PHY_UNIT_HOUR	5	Hour. 60*60 s
6	PHY_UNIT_MIN	6	Minute. 60 s
7	PHY_UNIT_SECOND	7	Second
8	PHY_UNIT_DEGREE	8	Phase angle in degree
9	PHY_UNIT_DEGREE_CELSIUS	9	Temperature (T) in °C = K - 273.15
10	PHY_UNIT_CURRENCY	10	(local) Currency
11	PHY_UNIT_METRE	11	Metre (m)
12	PHY_UNIT_METRE_PER_SEC	12	Metre per second (m/s)
13	PHY_UNIT_CUBIC_METRE_V	13	Cubic metre (m^3) for volume (V). rV , meter constant or pulse value (volume)
14	PHY_UNIT_CUBIC_METRE_C	14	Cubic metre (m^3). Corrected volume
15	PHY_UNIT_CUBIC_METRE_PER_HOUR_V	15	Cubic metre (m^3) per hour. (m^3 / h) m^3/(60*60s)
16	PHY_UNIT_CUBIC_METRE_PER_HOUR_C	16	Corrected cubic metre (m^3) per hour. (m^3 / h). m^3/(60*60s)
17	PHY_UNIT_CUBIC_METRE_PER_DAY_V	17	Cubic metre (m^3) per day. (m^3 / d) m^3/(24*60*60s)
18	PHY_UNIT_CUBIC_METRE_PER_DAY_C	18	Corrected cubic metre (m^3) per day. (m^3 / d). m^3/(24*60*60s)
19	PHY_UNIT_LITRE	19	Litre (l). 10^-3 m^3
20	PHY_UNIT_KILOGRAM	20	Kilogram (kg).
21	PHY_UNIT_NEWTON	21	Newton (N). Unit of force (F)
22	PHY_UNIT_NEWTON_METER	22	Newton meter (Nm). J = Nm = Ws
23	PHY_UNIT_PASCAL	23	Pascal (Pa). N/m^2
24	PHY_UNIT_BAR	24	Bar. 10^5 (N/m^2)
25	PHY_UNIT_JOULE	25	Joule. J = Nm = Ws
26	PHY_UNIT_JOULE_PER_HOUR	26	Joule per hour. J/(60*60s)
27	PHY_UNIT_WATT	27	Watt (W). W = J/s
28	PHY_UNIT_VOLT_AMPERE	28	Volt-Ampere (VA).
29	PHY_UNIT_VAR	29	Var (of reactive power)
30	PHY_UNIT_WATT_HOUR	30	Watt-hour (Wh). W*(60*60s)
31	PHY_UNIT_VOLT_AMPERE_HOUR	31	Volt-ampere-hour (VAh). VA*(60*60s)
32	PHY_UNIT_VAR_HOUR	32	VAr-hour (VArh). VAr*(60*60s)

No.	Unit name	Value	Description
33	PHY_UNIT_AMPERE	33	Ampere (A)
34	PHY_UNIT_COULOMB	34	Coulomb. C = As
35	PHY_UNIT_VOLT	35	Volt (V)
36	PHY_UNIT_VOLT_PER_METRE	36	Volt per metre (V/m).
37	PHY_UNIT_FARAD	37	Farad (F). C/V = As/V
38	PHY_UNIT_OHM	38	Ohm. Ω = V/A
39	PHY_UNIT_OHM_METRE	39	Ohm metre (Ωm)
40	PHY_UNIT_WEBER	40	Weber. Wb = Vs
41	PHY_UNIT_TESLA	41	Tesla (T). Wb/m^2
42	PHY_UNIT_AMPERE_PER_METRE	42	Ampere per metre. A/m
43	PHY_UNIT_HENRY	43	Henry. H = Wb/A
44	PHY_UNIT_HERTZ	44	Hertz (Hz).
45	PHY_UNIT_PULSE_PER_WATT_HOUR	45	1Pulse/Wh
46	PHY_UNIT_PULSE_PER_VAR_HOUR	46	1Pulse/VArh
47	PHY_UNIT_PULSE_PER_VA_HOUR	47	1Pulse/VAh
48	PHY_UNIT_VOLT_SQUARED_HOURS	48	Volt-squared-hours. V^2(60*60s)
49	PHY_UNIT_AMPERE_SQUARED_HOURS	49	Ampere-squared-hours. A^2(60*60s)
50	PHY_UNIT_KILOGRAM_PER_SECOND	50	Kilogram per second. Kg/s
51	PHY_UNIT_SIEMENS	51	Siemens (S). 1/Ω
52	PHY_UNIT_KELVIN	52	Kelvin (K)
53	PHY_UNIT_PULSE_PER_VOLT_SQUARED_HOUR	53	1/(V^2h)
54	PHY_UNIT_PULSE_PER_AMPERE_SQUARED_HOUR	54	1/(A^2h)
55	PHY_UNIT_PULSE_PER_VOLUME	55	1/m^3
56	PHY_UNIT_PERCENTAGE	56	%
57	PHY_UNIT_AMPERE_HOUR	57	Ah
58	PHY_UNIT_WATT_HOUR_PER_VOLUME	60	Wh/m^3. 3,6*103 J/m3
59	PHY_UNIT_JOULE_PER_VOLUME	61	J/m^3
60	PHY_UNIT_MOLE_PERCENT	62	Mole percent
61	PHY_UNIT_G_PER_VOLUME	63	g/m^3
62	PHY_UNIT_PASCAL_SECOND	64	Pa s
63	PHY_UNIT_JOULE_PER_KILOGRAM	65	J/kg
64	PHY_UNIT_DECIBELS_METRE	70	dBm
65	PHY_UNIT_RESERVED	253	Reserved
66	PHY_UNIT_OTHER_UNIT	254	Other unit
67	PHY_UNIT_NONE	255	No unit

4.2.3.3 attr_type_t type

This is an enumeration type.

Table 17 attr_type_t type

No.	Unit name	Value	Description
Simple Data Types			
1	ATTR_TYPE_NULL_DATA	0	Null data (no data)
2	ATTR_TYPE_BOOLEAN	3	Boolean
3	ATTR_TYPE_BIT_STRING	4	An ordered sequence of boolean values
4	ATTR_TYPE_DOUBLE_LONG	5	Integer32
5	ATTR_TYPE_DOUBLE_LONG_UNSIGNED	6	Unsigned32
6	ATTR_TYPE_OCTET_STRING	9	An ordered sequence of octets (8 bit bytes)
7	ATTR_TYPE_VISIBLE_STRING	10	An ordered sequence of ASCII characters
8	ATTR_TYPE_UTF8_STRING	12	An ordered sequence of characters encoded as UTF-8
9	ATTR_TYPE_BCD	13	Binary coded decimal
10	ATTR_TYPE_INTEGER	15	Integer8
11	ATTR_TYPE_LONG	16	Integer16
12	ATTR_TYPE_UNSIGNED	17	Unsigned8
13	ATTR_TYPE_LONG_UNSIGNED	18	Unsigned16
14	ATTR_TYPE_LONG64	20	Integer64
15	ATTR_TYPE_LONG64_UNSIGNED	21	Unsigned64
16	ATTR_TYPE_ENUM	22	The elements of the enumeration type (*)
17	ATTR_TYPE_FLOAT32	23	OCTET STRING (SIZE(4))
18	ATTR_TYPE_FLOAT64	24	OCTET STRING (SIZE(8))
19	ATTR_TYPE_DATE_TIME	25	OCTET STRING SIZE(12))
20	ATTR_TYPE_DATE	26	OCTET STRING (SIZE(5))
21	ATTR_TYPE_TIME	27	OCTET STRING (SIZE(4))
Complex Data Types			
22	ATTR_TYPE_ARRAY	1	The elements of the array (*)
23	ATTR_TYPE_STRUCTURE	2	The elements of the structure (*)
24	ATTR_TYPE_COMPACT_ARRAY	19	The elements of the compact array (*)

Note: (*) defined in the “Attribute description” section of a COSEM IC specification.

4.2.3.4 choice_t type

This is a structure type, little-endian. Total size is 2 bytes.

Table 18 choice_t type

No.	Type name (Element)	Type	Description
1	type	attr_type_t	Type of the choice.
2	size	integer8	Size of chosen type.

4.2.3.5 attr_access_method_t type

This is an enumeration type.

Table 19 attr_access_t type

No.	Unit name	Value	Description
1	ACCESS_MTD_MEM	0	Auto read from internal memory
2	ACCESS_MTD_USERFUNC	1	Read through user-defined function (R_OBIS_GetObjectAttr function)
3	ACCESS_MTD_E2PR	2	Auto read from EEPROM.

4.2.3.6 attr_access_t type

This is a structure type, little-endian. Total size is 5 bytes.

Table 20 attr_access_t type

No.	Type name (Element)	Type	Description
1	method	attr_access_method_t	Access method.
2	addr	Unsigned16	Start address in EEPROM.
3	size	Unsigned16	Size in EEPROM.

4.2.4 Common data type for OBIS Object definition

4.2.4.1 scaler_unit_t type

This is a structure type, little-endian. Total size is 2 bytes.

Table 21 scaler_unit_t type

No.	Type name (Element)	Type	Description
1	scaler	Integer8	This is the exponent (to the base of 10) of the multiplication factor.
2	unit	physical_unit_t	Enumeration defining the physical unit.

4.2.4.2 value_t, status_t type

This is a structure type, little-endian. These types are used as CHOICE data type in Blue Book.

Table 22 scaler_unit_t type

No.	Type name (Element)	Type	Description
1	choice	choice_t	Choice of type.
2	buffer	void*	Pointer to buffer of value.

4.2.4.3 dyn_value_t, dyn_status_t, dyn_date_time_t type

This is a structure type, little-endian. These types are used as CHOICE data type in Blue Book with supported read/write with selective methods defined in access element.

Table 23 scaler_unit_t type

No.	Type name (Element)	Type	Description
1	Choice	choice_t	Choice of type.
2	Buffer	void*	Pointer to buffer of value.
3	Access	attr_access_t	Access option of value.

4.2.4.4 buffer_t type

This is a structure type, little-endian. This type is used for buffer attribute (compact-array or array data type) in Blue Book supporting read/write with selective methods defined in access element.

Table 24 buffer_t type

No.	Type name (Element)	Type	Description
1	p_buffer	void*	Buffer of value.
2	p_access	attr_access_t*	Pointer to access method option.
3	p_current_buf_index	Unsigned32	Pointer to index of current entries in buffer.
4	async_entries_per_block	Unsigned16	Number of entries per block to transfer.
5	one_entry_len	Unsigned16	Length of encode 1 entry after filter.
6	async_remain_entries	Unsigned16	Number of remain entries in run-time.
7	from_entry	Unsigned32	First entry to retrieve in run-time.
8	to_entry	Unsigned32	Last entry to retrieve in run-time
9	from_value	Unsigned16	Index of first value to retrieve in run-time
10	to_value	Unsigned16	Index of last value to retrieve in run-time.

4.2.5 Return value

4.2.5.1 et_DATA_ACCESS_RESULT type

This is an enumeration type.

Table 25 et_DATA_ACCESS_RESULT type

No.	Unit name	Value	Description
1	DATA_ACCESS_RESULT_SUCCESS	0	Data access result successfully
2	HARDWARE_FAULT	1	Hardware fault
3	TEMP_FAIL	2	Temporal fail
4	RD_WRT_DENIED	3	Read/write denied
5	OBJ_UNDEFINE	4	Object undefined
6	OBJ_CLS_INCONSISTENT	9	Object class inconsistent
7	OBJ_UNAVAILABLE	11	Object unavailable
8	TYPE_UNMATCH	12	Type unmatched
9	SCOPE_ACCESS_VIOLATED	9	Scope access violated
10	DATA_BLK_UNAVAILABLE	10	Data block unavailable
11	LONG_GET_ABORT	11	Long get abort
12	NO_LONG_GET	12	No long get
13	LONG_SET_ABORT	13	Long set abort
14	NO_LONG_SET	14	No long set
15	DATA_ACCESS_RESULT_OTHER	250	Data access result other
16	DATA_ACCESS_RESULT_PARTIAL	251	Data access result partial

4.3 Attribute GET/SET methods

4.3.1.1 Common process of GET/SET attribute

Generally, R_OBIS_DistributeAttrBuffer function (in r_dlms_obis_ic.c) is used for handle attribute accessing (For buffer_t , please refer to 4.3.1.2 Buffer asynchronous transfer process).

This function support get/set attributes with selective methods and used inside distributor functions of class.

For more on the usage, please refer to r_dlms_obis_ic.c file.

The common process of this function is described as follow.

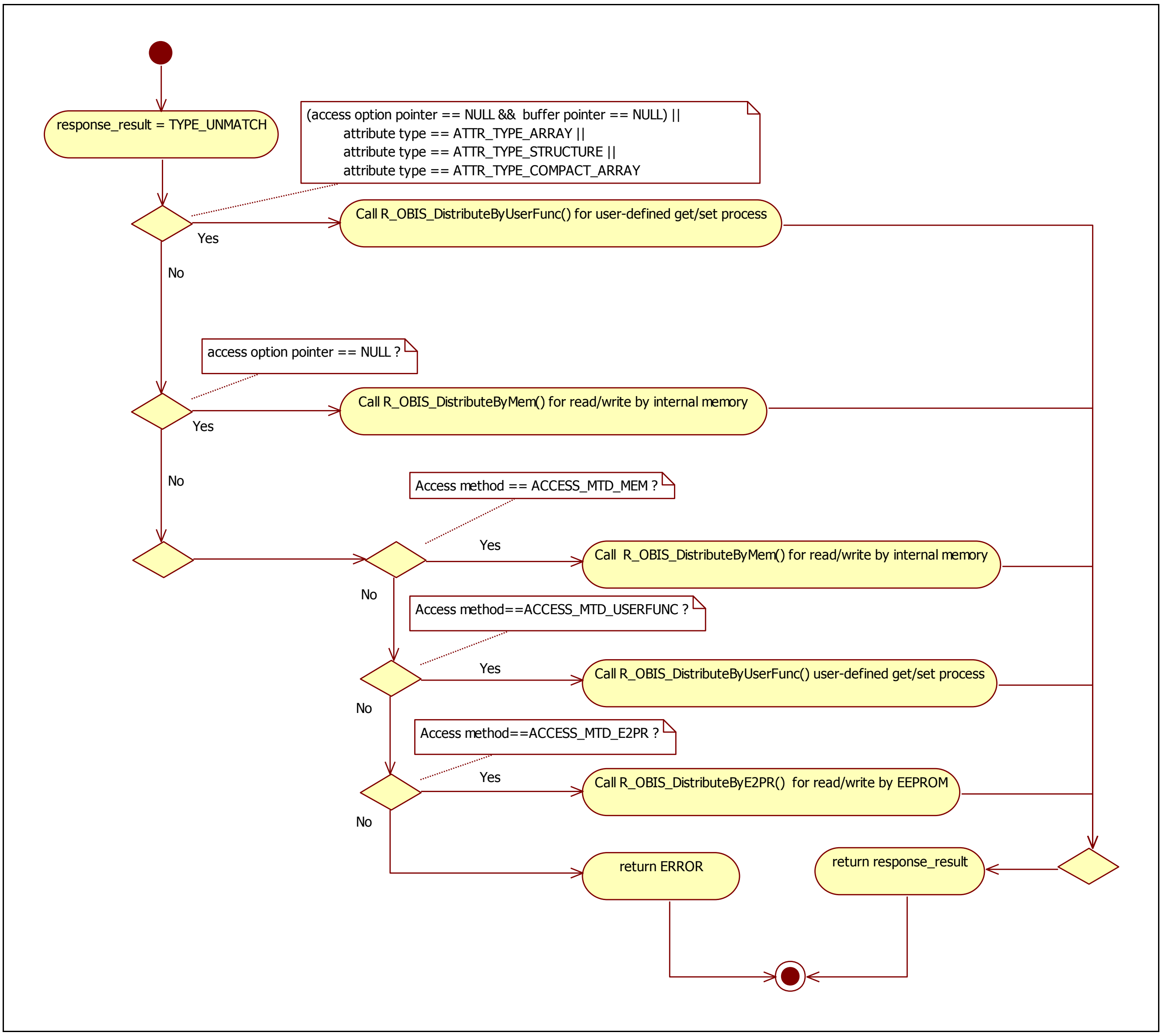

response_result = TYPE_UNMATCH
(access option pointer == NULL && buffer pointer == NULL) ||
attribute type == ATTR_TYPE_ARRAY ||
attribute type == ATTR_TYPE_STRUCTURE ||
attribute type == ATTR_TYPE_COMPACT_ARRAY
Yes
Call R_OBIS_DistributeByUserFunc() for user-defined get/set process
No
access option pointer == NULL ?
Yes
Call R_OBIS_DistributeByMem() for read/write by internal memory
No
Access method == ACCESS_MTD_MEM ?
Yes
Call R_OBIS_DistributeByMem() for read/write by internal memory
No
Access method==ACCESS_MTD_USERFUNC ?
Yes
Call R_OBIS_DistributeByUserFunc() user-defined get/set process
No
Access method==ACCESS_MTD_E2PR ?
Yes
Call R_OBIS_DistributeByE2PR() for read/write by EEPROM
No
return ERROR
return response_result

Figure 6 Processing of read/write attribute with 3 selective methods

4.3.1.2 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:

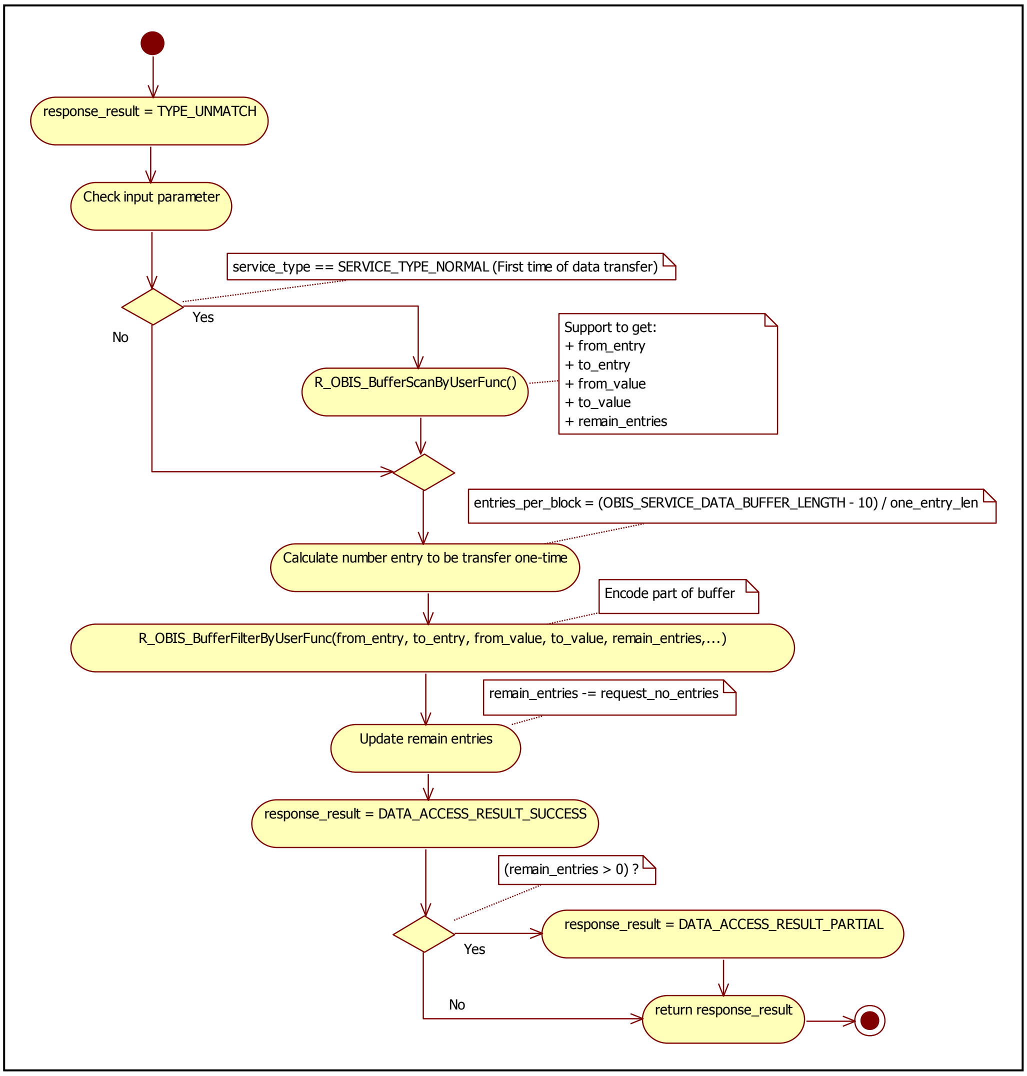

response_result = TYPE_UNMATCH
Check input parameter
service_type == SERVICE_TYPE_NORMAL (First time of data transfer)
Yes
No
R_OBIS_BufferScanByUserFunc()
Support to get:
+ from_entry
+ to_entry
+ from_value
+ to_value
+ remain_entries
entries_per_block = (OBIS_SERVICE_DATA_BUFFER_LENGTH - 10) / one_entry_len
Calculate number entry to be transfer one-time
Encode part of buffer
R_OBIS_BufferFilterByUserFunc(from_entry, to_entry, from_value, to_value, remain_entries,...)
remain_entries -= request_no_entries
Update remain entries
response_result = DATA_ACCESS_RESULT_SUCCESS
(remain_entries > 0) ?
response_result = DATA_ACCESS_RESULT_PARTIAL
Yes
No
return response_result

Figure 7 Processing of buffer asynchronous transfer

4.4 Function reference

Below functions are supported APIs for distributing attributes.

Note that using the R_OBIS_EncodeAttribute function and R_OBIS_DecodeAttribute function is not the most efficient way to encode/decode the data. User can encode/decode attribute by other ways or customize these APIs.

4.4.1 R_OBIS_EncodeAttribute

Synopsis

Encode data buffer for a specified attribute type

Prototype

```
Unsigned8 R_OBIS_EncodeAttribute (
    Unsigned8  *buf                  // [Out] Output buffer to store the encoded data
    Unsigned16 buf_len               // [In] The length of output buffer
    attr_type_t attr_type            // [In] Specify attribute type want to encode
    Unsigned8 *value_buf             // [In] Buffer that store value need to encode
    Unsigned16 value_buf_len         // [In] Specify the length of buffer of value
);
```

Description

Encode data buffer for a specified attribute type

This function can be used by User to encode the simple data type.

Return value

Unsigned16
0 mean error
> 0 is success, its value is actual used length of encode data.

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.2 R_OBIS_DecodeAttribute

Synopsis

Decode data buffer for a specified attribute type

Prototype

```
Unsigned8 R_OBIS_DecodeAttribute (
    Unsigned8 *value_buf             // [Out] Buffer that store result value
    Unsigned16 value_buf_len         // [In] Specify the length of buffer of value
    attr_type_t attr_type            // [In] Specify attribute type want to encode
    Unsigned8  *buf                  // [In] Encoded data that send by client
    Unsigned16 buf_len               // [In] The actual length of encoded buffer
);
```

Description

Decode data buffer of a specified attribute type to value type

This function can be used by User to decode the simple data type.

Return value

Unsigned16
0 mean error
> 0 is success, its value is actual used length of decode data.

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.3 R_OBIS_GetObjectAttr

Synopsis

Get attribute for a specified object

Prototype

```
Unsigned8 R_OBIS_GetObjectAttr (
    st_Cosem_Attr_Desc *cosem_attr_desc    // [In] COSEM Obj Descriptor
    service_type_t service_type            // [In] Service type
    Unsigned16 child_id                    // [In] Child ID in the child object table
    Unsigned8  *p_out_buf                  // [Out] Encoded output data
    Unsigned16 *p_out_len                  // [Out] Encoded length
    Unsigned8 *block_transfer              // [Out] Indicate block transfer is used or not
    Unsigned8 *p_data                      // [In] Data from client
);
```

Description

Get attribute for a specified object by User implementation

This function is customized by User to get the User-defined value for attribute

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.4 R_OBIS_SetObjectAttr

Synopsis

Set attribute for a specified object

Prototype

```
Unsigned8 R_OBIS_SetObjectAttr (
    st_Cosem_Attr_Desc *cosem_attr_desc    // [In] COSEM Obj Descriptor
    service_type_t service_type            // [In] Service type
    Unsigned16 child_id                    // [In] Child ID in the child object table
    Unsigned8 *p_data                      // [In] Data from client
    Unsigned16 data_len                    // [In] Data length
    Unsigned8 block_transfer               // [In] Indicate block transfer is used or not
);
```

Description

Set attribute for a specified object by User implementation

This function is customized by User to set the User-defined value for attribute

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.5 R_OBIS_E2PRGetData

Synopsis

Get attribute on EEPROM for a specified object

Prototype

```
Unsigned8 R_OBIS_E2PRGetData (
    attr_access_t *p_attr_access            // [In] Access method option
    attr_type_t attr_type                   // [In] Choice selection
    Integer16 attr_size                     // [In] Buffer of value
    Unsigned8 *p_out_buf                    // [Out] Data pointer for get
    Unsigned16 *p_out_len                   // [Out] Data length pointer for get
);
```

Description

Get attribute on EEPROM for a specified object

NOTE: Please take care on parameter passing,
- p_attr_access must not NULL.
- p_attr_access->method = ACCESS_MTD_E2PR

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.6 R_OBIS_E2PRSetData

Synopsis

Set attribute on EEPROM for a specified object

Prototype

```
Unsigned8 R_OBIS_E2PRSetData (
    attr_access_t *p_attr_access            // [In] Access method option
    attr_type_t attr_type                   // [In] Choice selection
    Integer16 attr_size                     // [In] Buffer of value
    Unsigned8 *pdata                        // [In]Data from COSEM Appl layer
);
```

Description

Set attribute on EEPROM for a specified object

Please take care on parameter passing,
- p_attr_access must not NULL.
- p_attr_access->method = ACCESS_MTD_E2PR

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.7 R_OBIS_BufferScanByUserFunc

Synopsis

Scan the buffer to get information before filter buffer

Prototype

```
Unsigned8 R_OBIS_BufferScanByUserFunc (
    buffer_t *buf                       // [In] Pointer to buffer_t
    Unsigned8 selective_access          // [In] Selective access
    Unsigned16 master_index             // [In] Id of the row in master list
    Unsigned16 child_index              // [In] Id of the row in child list
    Unsigned8 *p_selector_buffer        // [In] Pointer to the selector (from Client)
);
```

Description

Scan the buffer to get information before filter buffer

This function is customized by User to implement process of getting information about the part of buffer need to encoded for both cases of normal access and selective access

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.8 R_OBIS_BufferFilterByUserFunc

Synopsis

Encoded a part of buffer

Prototype

```
Unsigned8 R_OBIS_BufferFilterByUserFunc (
    buffer_t *buf                       // [In] Pointer to buffer_t
    Unsigned16 master_index             // [In] Id of the row in master list
    Unsigned16 child_index              // [In] Id of the row in child list
    Unsigned16 request_entries_nr       // [In] Number of entries need to filter
    Unsigned8 *p_out_buf                // [Out] Data pointer for get
    Unsigned16 *p_out_len               // [Out] Data length pointer for get
);
```

Description

Encode a part of buffer

This function is customized by user to encode the buffer based on scan information

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.9 R_OBIS_BufferBlockGet

Synopsis

Get 1 block data of buffer

Prototype

```
Unsigned8 R_OBIS_BufferBlockGet (
    void *buffer                         // [In] pointer to buffer list
    attr_access_t *p_attr_access         // [In] Access method option
    Unsigned16 block_index               // [In] Block index in buffer
    Unsigned16 block_size                // [In] Block size in byte
    Unsigned8 *p_out_buf                 // [Out] Data pointer for get
    Unsigned16 *p_out_len                // [Out] Data length pointer for get
);
```

Description

Get 1 block data of buffer

This function is supported to get block data on both internal memory and EEPROM.
The output data are not encoded

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.10 R_OBIS_BufferBlockSet

Synopsis

Set 1 block data of buffer

Prototype

```
Unsigned8 R_OBIS_BufferBlockSet (
    void *buffer                         // [In] pointer to buffer list
    attr_access_t *p_attr_access         // [In] Access method option
    Unsigned16 block_index               // [In] Block index in buffer
    Unsigned16 block_size                // [In] Block size in byte
    Unsigned8 *p_in_buf                  // [In] Data pointer for set
    Unsigned16 *p_in_len                 // [In] Data pointer for set
);
```

Description

Set 1 block data of buffer

This function is supported to set block data on both internal memory and EEPROM
The input data must not encoded

Return value

Data access result,
(Please refer to et_DATA_ACCESS_RESULT type)

Program example

For more on the usage, please refer to r_dlms_data.c file.

4.4.11 R_OBIS_RTC_ConstInterruptCallback

Synopsis

RTC Callback for internal processing

Prototype

```
void R_OBIS_RTC_ConstInterruptCallback (
);
```

Description

RTC Callback for internal processing

This function is call-back function, called by the meter for every interval period of RTC interrupt

Return value

None

Program example

For more on the usage, please refer to r_dlms_data_meter.c file.

4.4.12 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

For more on the usage, please refer to r_dlms_data_meter.c file.

4.4.13 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

For more on the usage, please refer to r_dlms_data_meter.c file.

4.4.14 R_OBIS_CompareDateTime

Synopsis

Compare RTC date time

Prototype

```
void R_OBIS_CompareDateTime (
    date_time_t *p_src_date_time            // [In] Source date time value
    date_time_t *p_des_date_time            // [In] Destination date time value
);
```

Description

Compare RTC date time

This function is supported to compare 2 date time values

Return value

Integer8
-1: src < des, 1: src > des, 0: src = des

Program example

For more on the usage, please refer to r_dlms_data_meter.c file.

5. Object installation

5.1 List of change files

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

```
+application
   +dlms
    |      +meter_app
    |      | -r_dlms_data.c
    |      | -r_dlms_data.h
    |      | -r_dlms_data_ic.c
    |      | -r_dlms_data_ic.h
    |      | -r_dlms_data_meter.c
    |      | -r_dlms_data_meter.h
```

Figure 8 List of changed files

5.2 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.

Object_defs_v2.6_
120912

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

5.3 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 */
…
```

5.4 Add new object

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

Table 26 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
```

5.5 Encode/ decode attribute by User

After adding new object definition, encode /decode processing for attributes maybe needed because DLMS object layer doesn’t support encode /decode for complex type attributes (structure, array or compact array and buffer).

All of modifiable functions are in r_dlms_data.c.

Table 27 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_BufferScanByUserFunc	Scan to get info for filter
4	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;
      }
…
```

5.6 Execute action method by User

Current object layer implementation doesn’t support this feature.

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	July.17.13	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