# RL78 Family

## DLMS Physical Layer User Manual

REJxxxxxxx-0100
Rev.1.00
December 12, 2013

### Introduction

This document explains the usage of DLMS physical layer.

### Target Device

Energy Meter based on RL78 Family Device.

### Contents



**REFERENCES**

- Green_Book_7th_edition.pdf

# 1. Overview

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

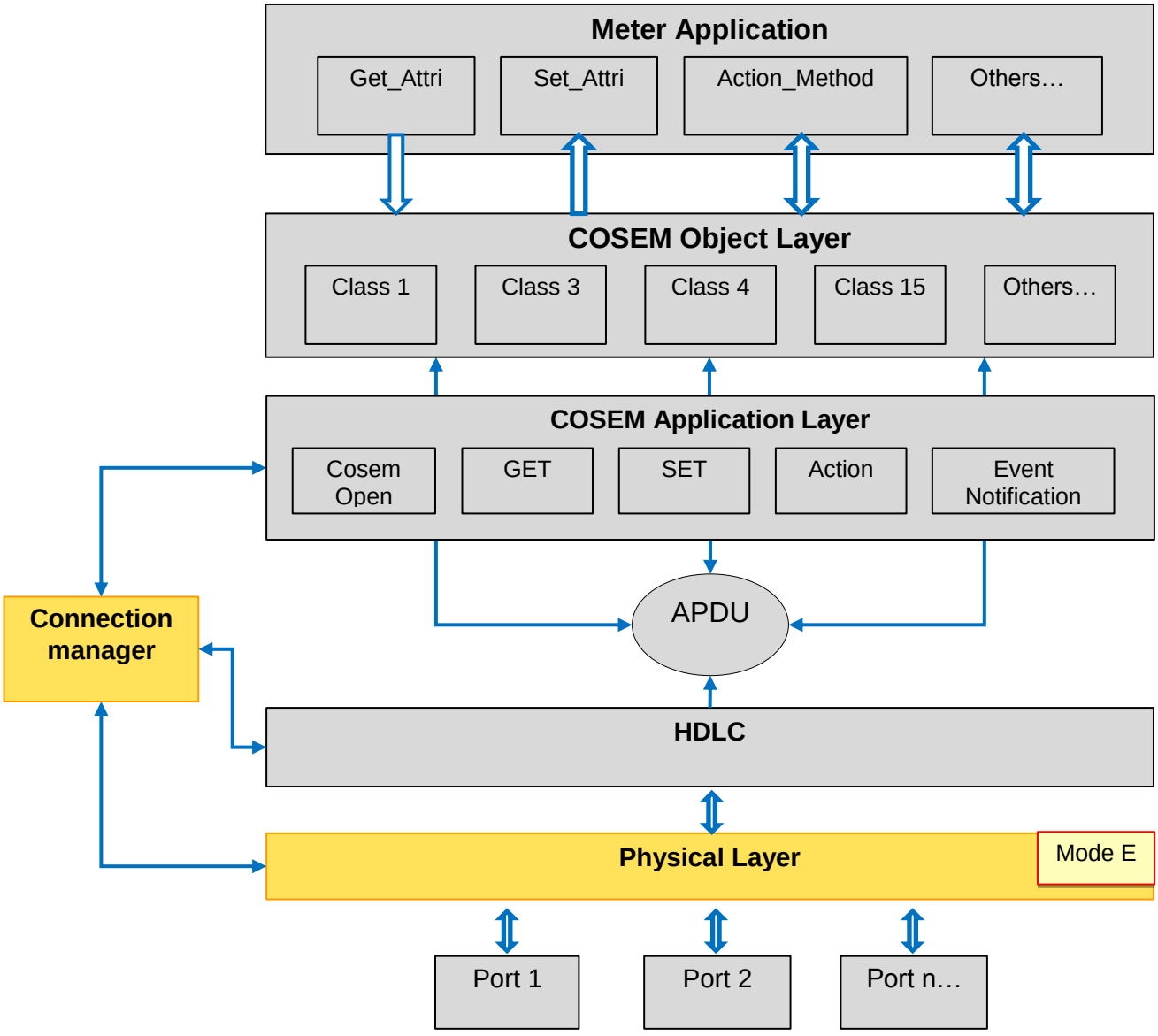


**Figure 1 Software composition**

The implementation for DLMS physical layer includes 2 parts:

- Connection manager: interface layer with others layer to manage all channels/ ports.
- Physical Layer: implementation for hardware accessing and mode E.

# 2. Files/directories composition

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

- Connection manager: implemented as connmgr folder.
- Physical Layer: consists of
  - Physical service implemented as physicalservice folder.
  - User-defined Physical Layer implemented as physical folder.

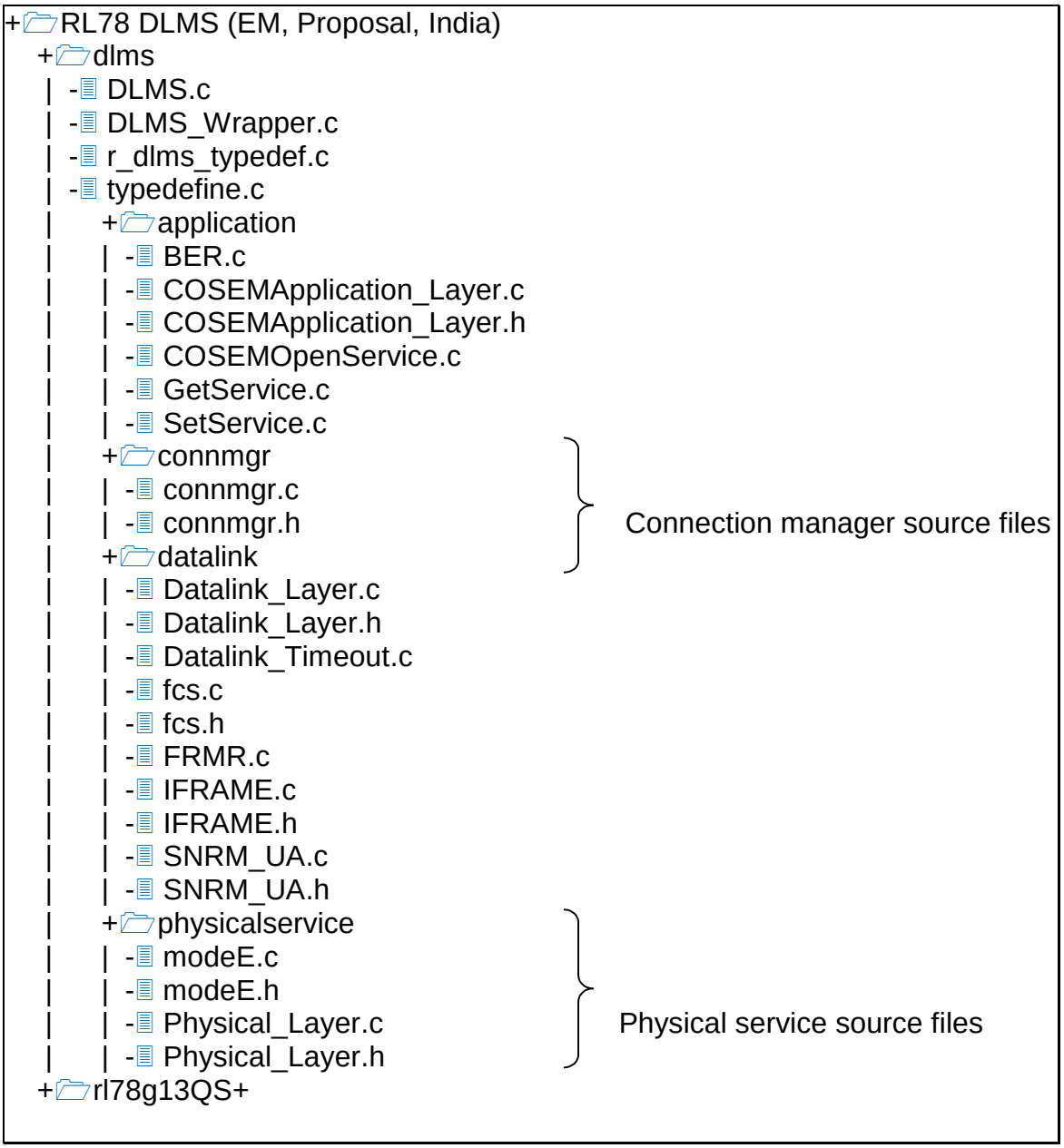


**Figure 2 File structure of DLMS physical layer**

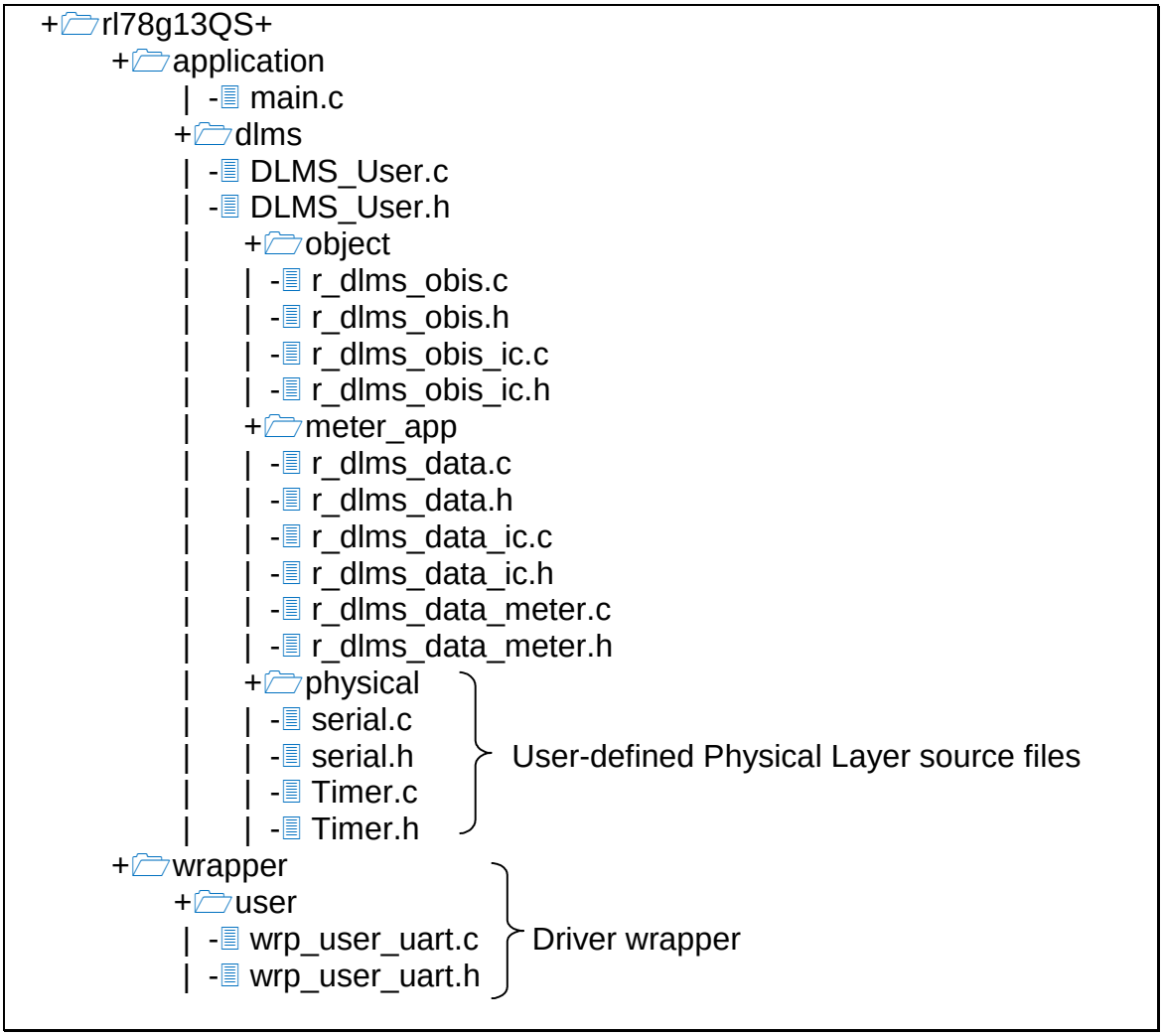


**Figure 3 File structure of DLMS physical layer**

**Table 1 File structure explanation**

| No. | File name | Description |
|---|---|---|
| 1 | connmgr.c | Connection manager source file. Implement for all connection manager APIs. |
| 2 | connmgr.h | Connection manager header file. Declare all related structure definition for connection manager. |
| 3 | mode_E.c | Mode E source file. Implementation for mode E protocol. |
| 4 | mode_E.h | Mode E header file. Declare all related structure definition for mode E. |
| 5 | Physical_Layer.c | Physical layer source file. Implement for all Physical layer service APIs. |
| 6 | Physical_Layer.h | Physical layer header file. Declare all related structure definition for Physical layer. |
| 7 | serial.c | Serial source file. Implemented by user/ customer to access the driver wrapper. |
| 8 | serial.h | Serial definition header file. |
| 9 | Timer.c | Timer source file. Implemented by user/ customer to initialize the timer. |
| 10 | Timer.h | Timer definition header file. Declare standards APIs. |
| 11 | wrp_user_uart.c | UART wrapper source file. Implemented by user/ customer to access the hardware driver of all channels. |
| 12 | wrp_user_uart.h | UART wrapper header file. |

# 3. Basic operation

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

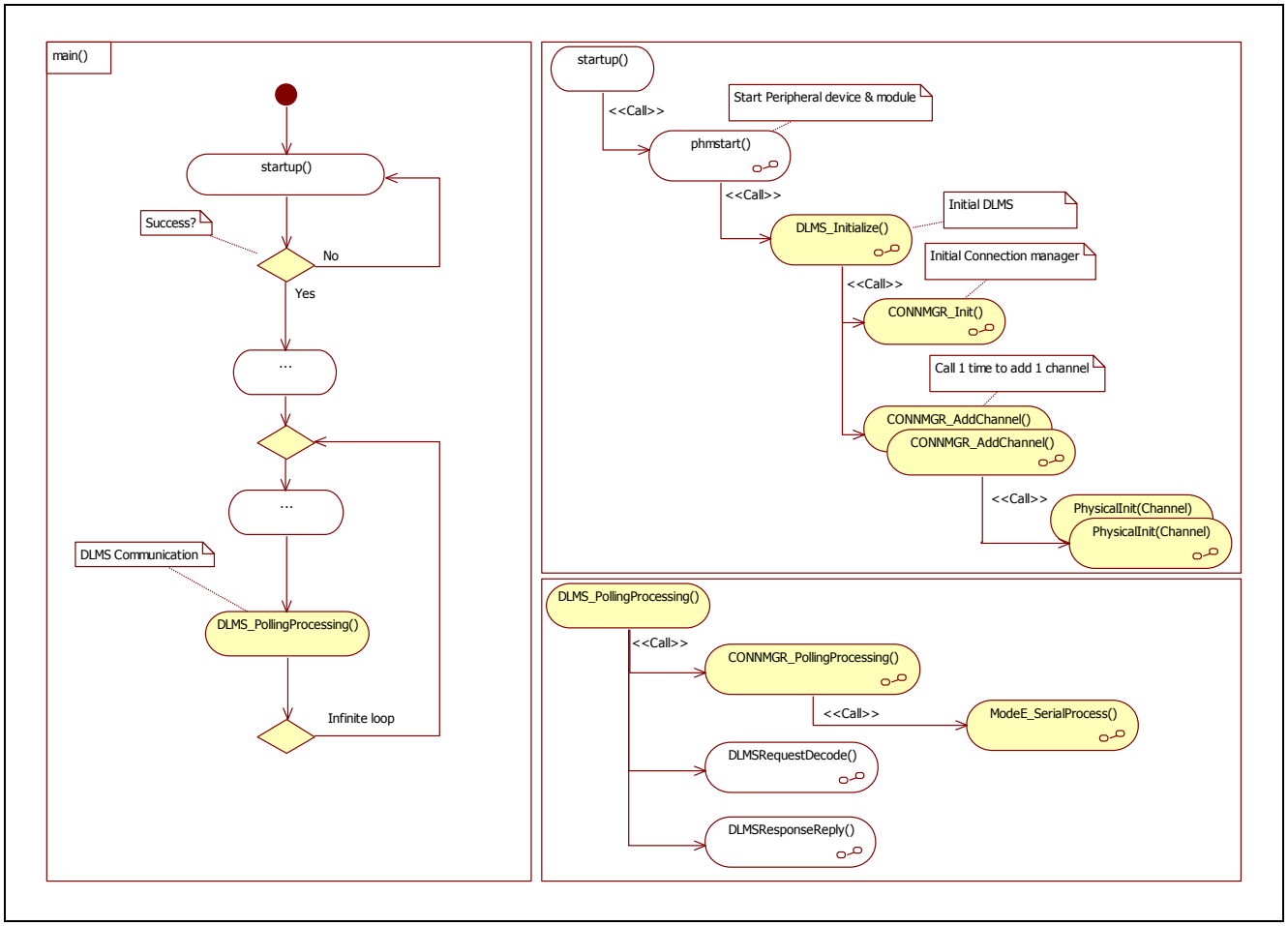


**Figure 4 Physical Layer basic operation**

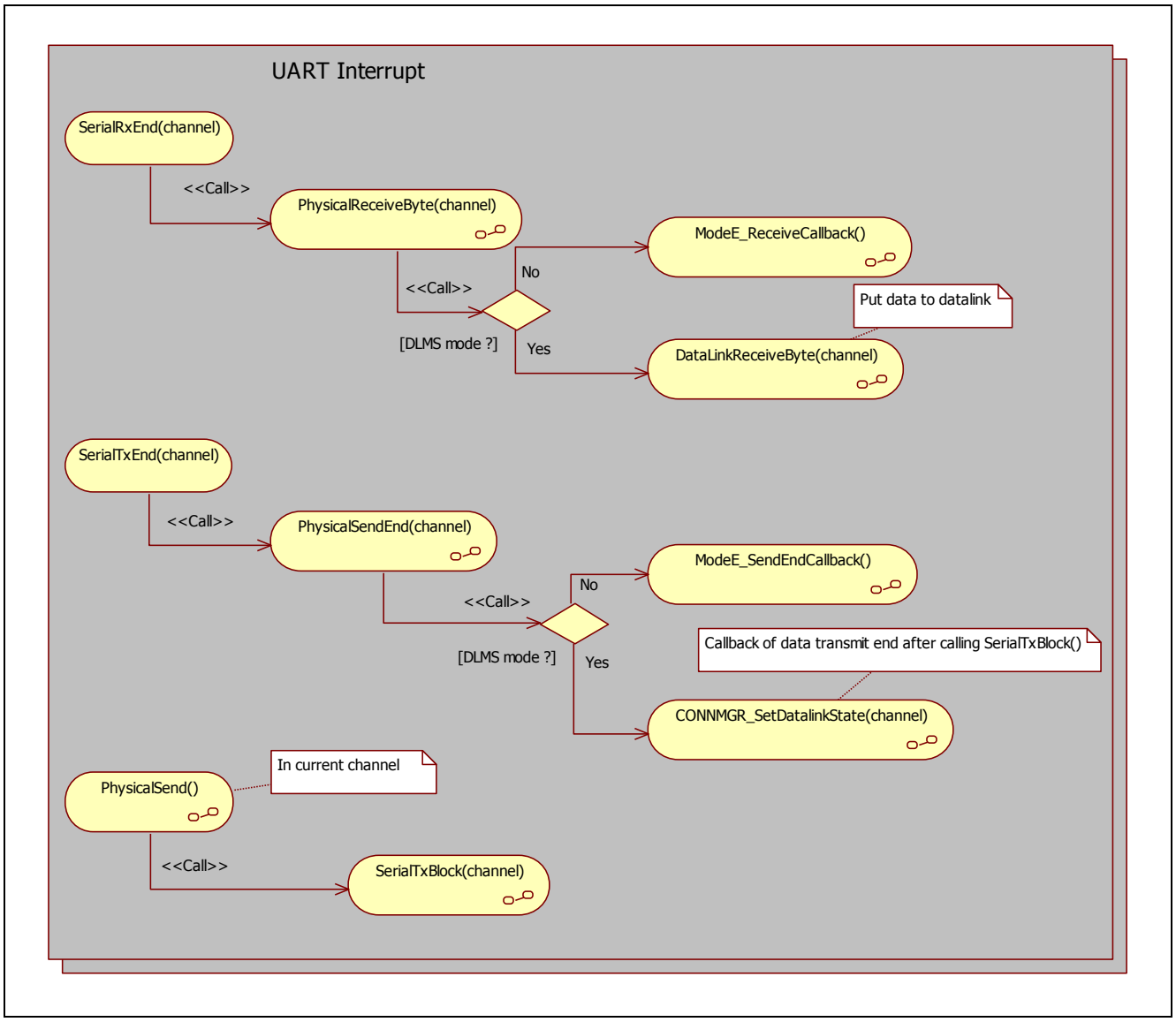


**Figure 5 Physical Layer basic interrupt operation**

# 4. Detail Implementation

## 4.1 Macro definitions

Below macros define the supported configuration in the current implementation. They are in \application\dlms\DLMS_User.h.

**Table 2 Common macros**

| No. | Macro name | Value | Description |
|---|---|---|---|
| 1 | ModeE_ENABLE | 1 | Using mode E for channel |
| 2 | ModeE_DISABLE | 0 | Not using mode E for channel |

## 4.2 Type definition

### 4.2.1 Primary type

Some common types for number as following:

**Table 3 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 Physical configuration formats

#### 4.2.2.1 et_PHY_PROTOCOL type

This is an enumeration type.

**Table 4 et_PHY_PROTOCOL type**

| No. | Unit name | Value | Description |
|---|---|---|---|
| 1 | IEC_PROTOCOL | 0 | IEC protocol |
| 2 | HDLC_PROTOCOL | 1 | HDLC protocol |
| 3 | PROTOCOL_NOT_SPECIFIED | 0xFF | Protocol is not specified |

#### 4.2.2.2 et_BAUD_RATE type

This is an enumeration type.

**Table 5 et_BAUD_RATE type**

| No. | Unit name | Value | Description |
|---|---|---|---|
| 1 | BAUD_RATE_300 | '0' | |
| 2 | BAUD_RATE_600 | '1' | |
| 3 | BAUD_RATE_1200 | '2' | |
| 4 | BAUD_RATE_2400 | '3' | |
| 5 | BAUD_RATE_4800 | '4' | |
| 6 | BAUD_RATE_9600 | '5' | |
| 7 | BAUD_RATE_19200 | '6' | |
| 8 | BAUD_RATE _NOT_SPECIFIED | 0xFF | |

#### 4.2.2.3 st_ServerConfig type

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

**Table 6 st_ServerConfig type**

| No. | Type name (Element) | Type | Description |
|---|---|---|---|
| 1 | SerialInit_FuncPtr | fn_SerialInit | Function pointer |
| 2 | SerialConfig_FuncPtr | fn_SerialConfig | Function pointer |
| 3 | SerialTx_FuncPtr | fn_SerialTx | Function pointer |
| 4 | *Server_Buffer | Unsigned8 | |
| 5 | Server_BufferSize | Unsigned16 | |
| 6 | AdressByteMode | Unsigned8 | |
| 7 | *ServerPhysicalAddress | Unsigned8 | |
| 8 | *DeviceAddress | Unsigned8 | |
| 9 | *ManufacturerID | Unsigned8 | |
| 10 | *ID | Unsigned8 | |
| 11 | IEC_BaudRate | Unsigned8 | |
| 12 | HDLC_BaudRate | Unsigned8 | |
| 13 | Response_Timeout | Unsigned16 | |
| 14 | Inactivity_Timeout | Unsigned16 | |
| 15 | Interframe_Timeout | Unsigned16 | |

## 4.3 Function reference

Below functions are supported APIs for establish and configure channels in DLMS physical layer.

### 4.3.1 DLMSInit

| | |
|---|---|
| **Synopsis** | DLMS Init |
| **Prototype** | Integer8 **DLMSInit** (<br>st_ServerConfig ServerConfig<br>); |
| **Description** | DLMS Init |
| **Return value** | Integer8 |
| **Program example** | For more on the usage, please refer to [\application\dlms\DLMS_User.c] file. |

```
#include "DLMS_User.h"

void DLMS_Initialize(void)
{
…

        /* Initialize the stack library */
        DLMSInit(UserServerConfig);
…
}
```

### 4.3.2 CONNMGR_ChannelCount

| | |
|---|---|
| **Synopsis** | Get registered channel number |
| **Prototype** | Unsigned8 **CONNMGR_ChannelCount**(void); |
| **Description** | Get registered channel number<br>At least 0, maximum is the returned value of CONNMGR_MaxChannelNumber() |
| **Return value** | Unsigned8<br>Number of registered channels |
| **Program example** | |

```
#include "DLMS_User.h"

Unsigned8 channel_nr = CONNMGR_ChannelCount();
```

### 4.3.3 CONNMGR_MaxChannelNumber

| | |
|---|---|
| **Synopsis** | Get max supported channel number |
| **Prototype** | Unsigned8 **CONNMGR_MaxChannelNumber** (void); |
| **Description** | Get max supported channel number |
| **Return value** | Unsigned8<br>Number of supported channels |

**Program example**

```
#include "DLMS_User.h"

Unsigned8 max_channel_nr = MAX_CONNMGR_CHANNEL_NUMBER();
```

### 4.3.4 CONNMGR_GetCurrentChannelID

**Synopsis**

Get current channel ID

**Prototype**

Unsigned8 **CONNMGR_GetCurrentChannelID** (void);

**Description**

Get current channel ID, assigned by CONNMGR_AddChannel()

**Return value**

Unsigned8
Channel ID of current channel

**Program example**

```
#include "DLMS_User.h"

Unsigned8 channel_id = CONNMGR_GetCurrentChannelID();
```

### 4.3.5 CONNMGR_AddChannel

**Synopsis**

Add 1 more channel for connection manager

**Prototype**

```
Unsigned8 CONNMGR_AddChannel (
    Unsigned8       channel_id,         /* Channel ID      */
    Unsigned8       modeE_enable,       /* Mode E support  */
    Unsigned8       *Tx_Buffer,         /* Tx buffer start */
    Unsigned16       Tx_Buffer_Size,    /* Tx buffer size  */
    Unsigned8       *Rx_Buffer,         /* Rx buffer start */
    Unsigned16       Rx_Buffer_Size     /* Rx buffer size  */
);
```

**Description**

Add 1 more channel for connection manager

This function should be called after DLMSInit() calling

**Return value**

Unsigned8
1: success, 0: fail

**Program example**

For more on the usage, please refer to [application\dlms\DLMS_User.c] file.

### 4.3.6 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 */
}
```

### 4.3.7 PhysicalReceiveCallback

| | |
|---|---|
| **Synopsis** | Callback when receive 1 byte |
| **Prototype** | void **PhysicalReceiveCallback** (<br>Unsigned8 channel_id,<br>Unsigned8 byte,<br>); |
| **Description** | Callback when receive 1 byte |
| **Return value** | None |
| **Program example** | For more on the usage, please refer to [\application\dlms\physical\serial.c] file. |

```
#include "DLMS_User.h"

void SerialRxEnd(Unsigned8 channel, Unsigned8 byte)
{
        /* Put data to physical layer */
        PhysicalReceiveCallback(channel, byte);
}
```

### 4.3.8 PhysicalSendEndCallback

| | |
|---|---|
| **Synopsis** | Send End callback |
| **Prototype** | void **PhysicalSendEndCallback** (<br>Unsigned8 channel_id,<br>); |
| **Description** | Callback of Send End |
| **Return value** | None |

| Program example | For more on the usage, please refer to [\application\dlms\physical\serial.c] file. |
|---|---|

```
#include "DLMS_User.h"

void SerialTxEnd ()
{
        /* Notify to physical layer */
        PhysicalSendEndCallback(channel);
}
```

### 4.3.9 InitSerial

| Synopsis | Initialization of serial |
|---|---|
| Prototype | void **InitSerial** (<br>Unsigned8 channel_id,<br>); |
| Description | Initialization of UART unit to enable<br>serial receive/transmit operations |
| Return value | None |
| Program example | For more on the usage, please refer to [application\dlms\DLMS_User.c] file. |

### 4.3.10 SerialTxEnd

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

```
#include "serial.h"

void WRP_UART0_SendEndCallback ()
{

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

### 4.3.11 SerialRxEnd

| Synopsis | Physical layer's callback of data receive end through serial communication |
|---|---|
| Prototype | void **SerialRxEnd** (<br>Unsigned8 channel_id,<br>Unsigned8 byte,<br>); |
| Description | Callback of data Receive end |
| Return value | None |

**Program example**

Append this API to wrapper of UART driver (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);
}
```

### 4.3.12 SerialTxBlock

| | |
|---|---|
| **Synopsis** | Transmit block of data through serial communication |
| **Prototype** | void **SerialTxBlock** (<br>Unsigned8 channel_id,<br>Unsigned8 *BlockPtr // [Input] Pointer to block start address<br>Integer16 Length // [Input] Length of the data in byte<br>); |
| **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 (In \application\dlms\physical\serial.c) |

```
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;
        }
}
```

### 4.3.13 SerialConfig

| | |
|---|---|
| **Synopsis** | Reconfigure UART to adapt with new baud_rate,new protocol |
| **Prototype** | void **SerialTxBlock** (<br>Unsigned8 channel_id,<br>Unsigned8 new_baud_rate<br>Unsigned8 new_protocol<br>); |
| **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)

```
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 */
        }
}
```

### 4.3.14 TimerRCinit

**Synopsis** Initialises RC timer

**Prototype** Unsigned8 **TimerRCinit** (void);

**Description** Initialises RC timer

**Return value** None

**Program example**

Implement this API to connect to TImer driver (In \application\dlms\physical\Timer.c)

```
#include "Timer.h"

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

# 5. Sample workspace

## 5.1 List of change files

For physical layer porting and customize, list of changed file is described as below:

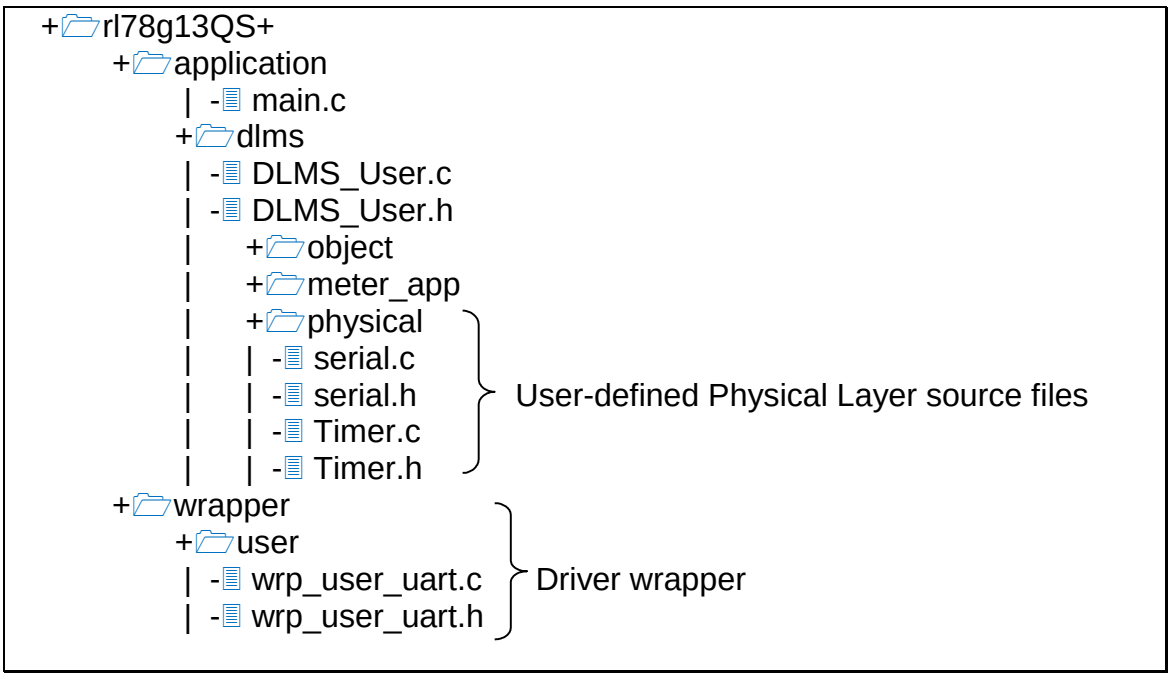


**Figure 6 List of changed files**

## 5.2 Register API callbacks in driver

For driver adaptation, below is list of API callbacks customized by User.

**Table 7 List of API callbacks customized by User**

| No. | Function name | Description |
|---|---|---|
| 1 | PhysicalTimeoutCount | 1ms timer callback |
| 2 | SerialRxEnd | Call PhysicalSendEndCallback |
| 3 | SerialTxEnd | Call PhysicalReceiveCallback |

## 5.3 Configure DLMS physical parameters

To configure DLMS physical layer by DLMSInit() function, below files is related to:

**Table 8 Related file for new object definition**

| No. | File name | Description |
|---|---|---|
| 1 | DLMS_User.h | Reference header file. All related structure definition for DLMS physical layer. |
| 2 | DLMS_User.c | Configure DLMS parametrs. |

For example:

```
Unsigned8       RxBufferMain[MAX_RECIEVE_BUFFER_SIZE];
Unsigned8       RxBufferSub[MAX_RECIEVE_BUFFER_SIZE];
Unsigned8       TxBuffer[MAX_TRANSMIT_BUFFER_SIZE];
Unsigned8       ServerBuffer[MAX_SERVER_BUFFER_SIZE];

const Unsigned8        ServerPhysicalAddress[] = {0x03, 0x23, 0x00, 0x23};
const Unsigned8        DeviceAddress[] = "RES001";                  /* max 32 character  */
const Unsigned8        ManufacturerID[] = "REN";                    /* 3 character  */
const Unsigned8        ID[] = "SPEM";                               /* max 16 character  */

/* Config parameter for DLMS library here */
const st_ServerConfig UserServerConfig =
{
        InitSerial,                                          /* SerialInit_FuncPtr     */
        SerialConfig,                                        /* SerialConfig_FuncPtr   */
        SerialTxBlock,                                       /* SerialTx_FuncPtr       */
        ServerBuffer,                                        /* *Server_Buffer         */
        MAX_SERVER_BUFFER_SIZE,                              /* Server_BufferSize      */
        PHYADD_1BYTE_SUPPORTED,                              /* AdressByteMode         */
        ServerPhysicalAddress,                               /* *ServerPhysicalAddress */
        DeviceAddress,                                       /* *DeviceAddress         */
        ManufacturerID,                                      /* *ManufacturerID        */
        ID,                                                  /* *ID                    */
        BAUD_RATE_300,                                       /* IEC_BaudRate           */
        BAUD_RATE_9600,                                      /* HDLC_BaudRate          */
        5000,                                                /* Response_Timeout       */
        5000,                                                /* Inactivity_Timeout     */
        100,                                                 /* Interframe_Timeout     */
};
```

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

        /* Initialize the stack library */
        DLMSInit(UserServerConfig);
…
}
```

## 5.4 Append new channel

Appending 1 or more channels to physical layer is possible. Below is step-to-step guide line to append 3rd channel to DLMS physical layer.

### 5.4.1 Change max supported channel number

In \rl78g13QS+\application\dlms\DLMS_User.h, change the max supported channel to 3 channels:

For example:

```
#define MAX_CONNMGR_CHANNEL_NUMBER          (3)
```

### 5.4.2 Specify channel ID

In \rl78g13QS+\application\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_NEW as channel ID for new channel

```
/* ID of physical channel(s) */
#define CHANNEL_PRIMARY                     (0x00)
#define CHANNEL_SECONDARY                   (0x01)
#define CHANNEL_NEW                         (0x02)
```

### 5.4.3 Link driver APIs of new channel to physical layer

In \rl78g13QS+\application\dlms\physical\serial.c, append driver API of new channel to below function to link new channel's driver with physical layer.

**Table 9 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;
    case CHANNEL_NEW:
        WRP_UART2_SendData(BlockPtr, Length);
        break;
    default:
        /* Do nothing */
    break;
}
```

### 5.4.4 Add new channel to 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().

For example: Add 3rd 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      */

                ModeE_ENABLE,                           /* Mode E support  */

                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() >= 2)

        {

                /* Add 2nd channel */

                CONNMGR_AddChannel(

                        CHANNEL_SECONDARY,              /* Channel ID      */

                        ModeE_ENABLE,                   /* Mode E support  */

                        TxBuffer,                       /* Tx buffer start */

                        MAX_TRANSMIT_BUFFER_SIZE,       /* Tx buffer size  */
```

```
                    RxBufferSub,                    /* Rx buffer start */
                    MAX_RECIEVE_BUFFER_SIZE         /* Rx buffer size  */
            );
    }
    if (CONNMGR_MaxChannelNumber() >= 3)
    {
            /* Add 3rd channel */
            CONNMGR_AddChannel(
                    CHANNEL_NEW,                    /* Channel ID      */
                    ModeE_ENABLE,                   /* Mode E support  */
                    TxBuffer,                       /* Tx buffer start */
                    MAX_TRANSMIT_BUFFER_SIZE,       /* Tx buffer size  */
                    RxBufferSub3,                   /* Rx buffer start */
                    MAX_RECIEVE_BUFFER_SIZE         /* Rx buffer size  */
            );
    }
...
```

## 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 | December.12.13 | 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