RL78 Family

PMBus Master Module Software Integration System

Introduction

This application note describes the PMBus Master module.

Target Device

RL78/G24

Related Documents

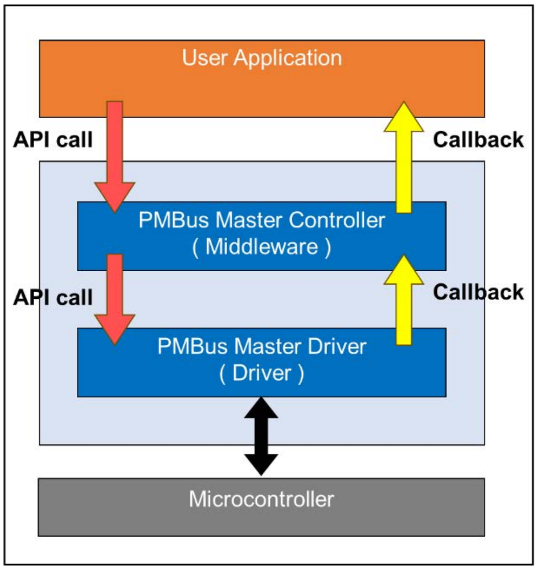
- RL78/G24 User's Manual: Hardware (R01UH0961J)
- PMBus Specification Rev. 1.4 Part I
- PMBus Specification Rev. 1.4 Part II
- System Management Bus (SMBus) Specification Version 3.2

Contents

1. Overview

This module consists of a driver layer (PMBus Master Driver) and a middleware layer (PMBus Master Controller) and provides an interface for sending and receiving data via PMBus (Power Management Bus) communication.

Figure 1-1 Module Configurations



1.1 PMBus Master Driver (PMBMDRV) Features

The PMBus Master Driver is intended to be accessed from the middleware layer (PMBus Master Controller).

The PMBus Master Driver provides the following features as the driver layer of the PMBus Master module.

1.1.1 Bus Communication Features

PMBus master send and receive operation is performed using the serial interface IICA.

(1) Bus speed

100 kHz, 400 kHz, and 1 MHz can be set with the Smart Configurator.

Note:

When 1MHz is set, a level shifter must be provided in an external circuit.

(2) Serial bus communication

The following communication operations are performed using Microcontroller's serial interface IICA.

- Generate Start condition
- Generate Repeat Start condition
- Send and receive data
- Generate Stop Condition

(3) Notification to upper layers

Notification is sent to upper layers at the following timing.

- Sending completed
- Receiving completed
- Communication error detection (Details are explained in the next chapter.)

1.1.2 Communication Error Detection

The PMBus Master module's driver layer detects the following communication errors and notifies the upper layers.

(1) NACK detection

When NACK is detected during data sending to a slave, it is judged as NACK detection.

(2) Communication timeout

Byte data reception timing is monitored during communication transactions, and when the reception interval is 25 ms or longer, communication timeout is considered to have occurred. Byte data receive timing monitoring is achieved by detecting INTIICA0 interrupts. This enables detection of a Low fixation state on the SCLA0 line.

1.1.3 Optional Pin Features

The following optional terminal ports can be assigned in the Smart Configurator.

(1) Control Signal (CONTROL)

The CONTROL pin is an optional input pin; it is used in conjunction with the PMBus command to turn the device on or off. The active level of the pin can also be set by the PMBus command.

(2) SMBALERT#

The SMBALERT# pin is an optional output pin. It is used as an interrupt line from the slave to the master.

Note:

The electrical characteristics of these pins follow those described in the "RL78/G24 User's Manual: Hardware". To comply with the electrical characteristics described in the SMBus specifications, it is necessary to use external circuits.

1.2 PMBus Master Controller (PMBMCTL) Features

The PMBus Master Controller is intended to be accessed by user applications.

As the middleware layer of the PMBus Master module, it provides the following features

1.2.1 Communication Formats

Upon receiving a command code transmission request from the user application, a data send request is made to PMBMDRV when a start condition is generated, and a data receive request is made to PMBMDRV when a repeat start condition is generated. The communication formats supported by this module and their protocol diagrams are described below.

Protocol diagram legend:

S : Start Condition

Sr : Repeat Start Condition

Rd : Read (1)

Wr : Write (0)

ACK : Acknowledge

NACK : Not Acknowledge

PEC : Packet Error Code *PEC support is optional

P : Stop Condition

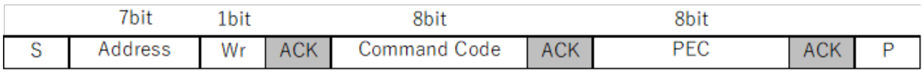
□ : Master → Slave

▨ : Slave → Master

(1) Send Byte

Command code is sent to any target address.

Figure 1-2 Communication Format Send Byte

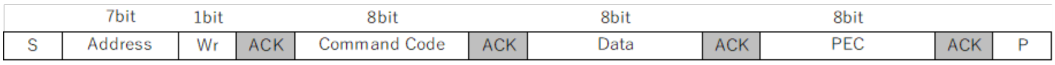


	7bit	1bit		8bit		8bit		
S	Address	Wr	ACK	Command Code	ACK	PEC	ACK	P

(2) Write Byte/Word, Write 32 protocol, Write 64 protocol

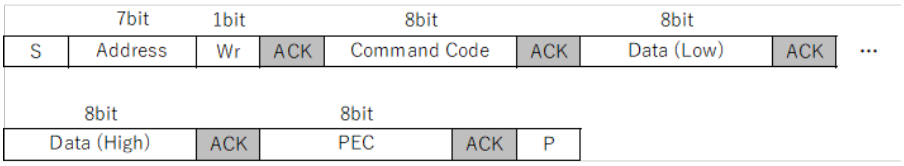
Command code and corresponding write data are sent to any target address.

Figure 1-3 Communication Format Write (Single byte)



	7bit	1bit		8bit		8bit		8bit		
S	Address	Wr	ACK	Command Code	ACK	Data	ACK	PEC	ACK	P

Figure 1-4 Communication Format Write (Multi byte)



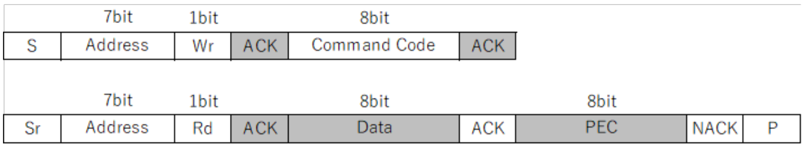
	7bit	1bit		8bit		8bit		
S	Address	Wr	ACK	Command Code	ACK	Data (Low)	ACK	…

8bit		8bit		
Data (High)	ACK	PEC	ACK	P

(3) Read Byte/Word, Read 32 protocol, Read 64 protocol

Command code is sent to any target address. Next, the response data corresponding to the command code is read from the device.

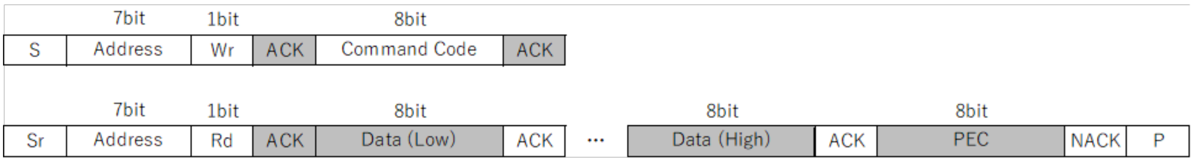
Figure 1-5 Communication Format Read (Single byte)



	7bit	1bit		8bit	
S	Address	Wr	ACK	Command Code	ACK

	7bit	1bit		8bit		8bit		
Sr	Address	Rd	ACK	Data	ACK	PEC	NACK	P

Figure 1-6 Communication Format Read (Multi byte)



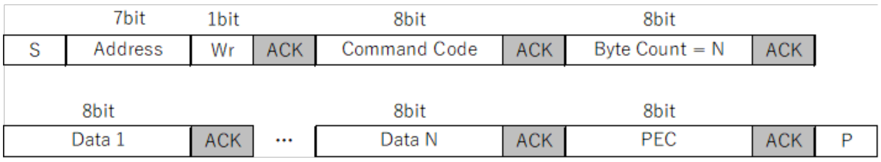
	7bit	1bit		8bit	
S	Address	Wr	ACK	Command Code	ACK

	7bit	1bit		8bit			8bit		8bit		
Sr	Address	Rd	ACK	Data (Low)	ACK	…	Data (High)	ACK	PEC	NACK	P

(4) Block Write

Command code and corresponding write data are sent to any target address. Before sending write data, data (Byte Count) indicating the data length is sent.

Figure 1-7 Communication Format Block Write



	7bit	1bit		8bit		8bit	
S	Address	Wr	ACK	Command Code	ACK	Byte Count = N	ACK

8bit			8bit		8bit		
Data 1	ACK	…	Data N	ACK	PEC	ACK	P

(5) Block Read

Command code is sent to any target address. Next, the response data corresponding to the command code is read from the device. Before sending the response data, the device sends data (Byte Count) indicating the data length.

Figure 1-8 Communication Format Block Read

	7bit	1bit		8bit	
S	Address	Wr	ACK	Command Code	ACK

	7bit	1bit		8bit		8bit			8bit		8bit		
Sr	Address	Rd	ACK	Byte Count = N	ACK	Data (Low)	ACK	···	Data (High)	ACK	PEC	NACK	P

(6) Block Write-Block Read Process Call

The Block Write and Block Read described above are executed in the same transaction. Write data length (Byte Count M) and response data length (Byte Count N) may differ.

Figure 1-9 Communication Format Block Write-Block Read Process Call

	7bit	1bit		8bit		8bit		8bit			8bit	
S	Address	Wr	ACK	Command Code	ACK	Byte Count = M	ACK	Data 1	ACK	···	Data M	ACK

	7bit	1bit		8bit		8bit			8bit		8bit		
Sr	Address	Rd	ACK	Byte Count = N	ACK	Data 1	ACK	···	Data N	ACK	PEC	NACK	P

1.2.2 Send/Receive Completion Notification

Notifies the user of the completion of sending a command when it has been sent according to the communication format described above.

(1) Notification of completion of sending Send command

Notifies the user of the completion of sending in the Send Byte communication format by a callback.

(2) Notification of completion of sending Write data

Notifies the user of the completion of sending in Write Byte/Word, Write 32 protocol, Write 64 protocol communication format or Block Write communication format by a callback.

(3) Notification of completion of Read data reception

Notifies the user of received data via callback upon completion of sending in Read Byte/Word, Read 32 protocol, Read 64 protocol communication format or Block Read communication format.

(4) Notification of completion of sending Write/Read data

Notifies the user of received data via callback upon completion of sending in the Block Write-Block Read Process Call communication format.

(5) Notification of completion of sending AlertResponseAddress

Notifies the user of the device address received upon completion of sending in the Device responds to an ARA communication format by a callback.

(6) Notification of completion of HostNotify reception

Notifies the user of the received device address and error information (STATUS_WORD) via callback upon completion of reception in the Host Notify communication format.

1.2.3 Fault Detection and User Notification

This feature detects each failure that occurs during data communication and notifies the user via a callback. For details, see chapter 4.1.2 pmbmctl_fault_t.

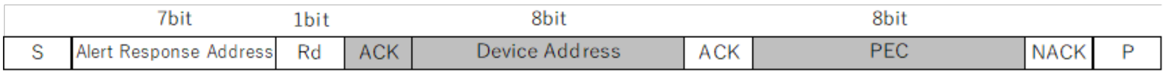
1.2.4 Reporting to the Controller

PMBus devices must notify the controller (master) of a warning or fault condition when an anomaly is detected. This module provides the following two features for notification; the PMBus specification specifies that at least one of these two methods must be supported.

(1) Notification by SMBALERT#signal

PMBus devices can generate an alert signal from the SMBALERT#pin. The user can call the API function "RM_PMBMCTL_GetHWSignal" to obtain the signal status of the SMBALERT#pin. To identify the device that is the source of the notification, the API function "RM_PMBMCTL_SendAlertResponse" can be called to request a read request to the SMBus Alert Response Address. This request is then sent to the PMBus device. The PMBus device will respond with its own station address value.

Figure 1-10 Communication format Device responds to an ARA

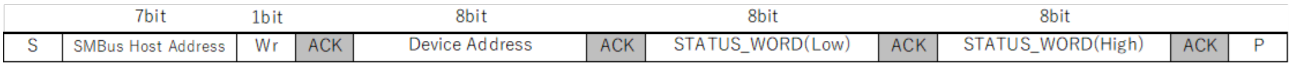


	7bit	1bit		8bit		8bit		
S	Alert Response Address	Rd	ACK	Device Address	ACK	PEC	NACK	P

(2) Notification by SMBus Host Notify Protocol

PMBus devices can notify device addresses and error information (STATUS_WORD) via the SMBus Host Notify protocol. The user can obtain the device address and error information (STATUS_WORD) through the HostNotify receipt completion notification described above.

Figure 1-11 Communication format Host Notify



	7bit	1bit		8bit		8bit		8bit		
S	SMBus Host Address	Wr	ACK	Device Address	ACK	STATUS_WORD(Low)	ACK	STATUS_WORD(High)	ACK	P

1.2.5 H/W Signal Information

The user can call the API function "RM_PMBMCTL_GetHWSignal" to obtain the signal levels of the following optional pins.

- Control Signal (CONTROL)
- SMBALERT#

2. API Information

This section describes the API information for this module.

2.1 Hardware Requirements

The MCU to be used must support the following pins

- SCLA0 (P60)
- SDAA0 (P61)

Target products: 30, 32, 40, 44, 48, 52, 64-pin products

2.2 Software Requirements

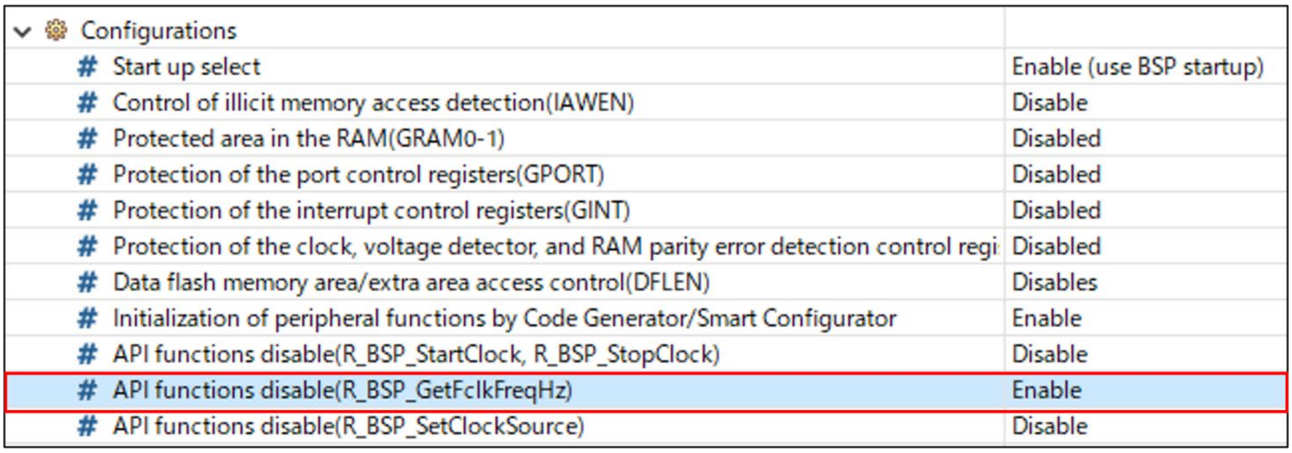
This driver depends on the following modules

- Board Support Package (r_bsp) v1.61 or later

In addition, the following API functions of r_bsp must be enabled, which can be configured from the Software Component Settings screen on the Smart Configurator.

- R_BSP_GetFclkFreqHz
  (BSP_CFG_GET_FREQ_API_FUNCTIONS_DISABLE = 0)

Figure 2-1 Smart Configurator BSP Settings



Configurations	
# Start up select	Enable (use BSP startup)
# Control of illicit memory access detection(IAWEN)	Disable
# Protected area in the RAM(GRAM0-1)	Disabled
# Protection of the port control registers(GPORT)	Disabled
# Protection of the interrupt control registers(GINT)	Disabled
# Protection of the clock, voltage detector, and RAM parity error detection control regi	Disabled
# Data flash memory area/extra area access control(DFLEN)	Disables
# Initialization of peripheral functions by Code Generator/Smart Configurator	Enable
# API functions disable(R_BSP_StartClock, R_BSP_StopClock)	Disable
# API functions disable(R_BSP_GetFclkFreqHz)	Enable
# API functions disable(R_BSP_SetClockSource)	Disable

2.3 Supported Tool Chains

This module has been tested with the following toolchains.

- Renesas CC-RL Toolchain v1.12.01
- IAR Embedded Workbench for Renesas RL78 v5.10.3

2.4 Header files

API calls and I/F definitions used are described in "rm_pmbmctl_api.h" and "rm_pmbmdrv_api.h".

2.5 Integer Type

This driver uses ANSI C99. These types are defined in "stdint.h".

2.6 Code Size

ROM and RAM sizes increase or decrease depending on the settings on the Smart Configurator and compiler option settings. Here, the sizes are given for reference when the settings on the Smart Configurator are the default settings and the compile options on the CC-RL compiler are set to the default settings.

ROM : 4,720 [byte]

RAM : 1,377 [byte]

3. Configuration Specifications

A list of configuration items that can be set in the Smart Configurator is shown below.

Table 3.1 PMBus Master Driver setting items list

Item	Possible values	Description
Bus Speed	100kHz, 400kHz、1MHz	Specify the bus speed.
IICA Input Mode	I2C, SMBus	Specify the IICA input mode.
SDAA and SCLA signal falling times (tF)[ns]	0 to 300	Set the fall time of the SDAA and SCLA signals. (Note1)
SDAA and SCLA signal rising times (tR)[ns]	0 to 1000	Set the rise time of the SDAA and SCLA signals. (Note1)
Digital filter	ON, OFF	Specify whether the digital filter is enabled or not. (Note2)
Interrupt level for INTIICA0	Level 0(Highest), Level 1, Level 2, Level 3(Lowest)	Set the INTIICA0 interrupt priority.
Pin for Control Signal	Unused, P00 to P147	Select the Control Signal pin.
Pin for SMBALERT#	Unused, P00 to P147	Select the SMBALERT#pin.
Timer resource for device timeout measurement	TAU0_0, TAU0_1, TAU0_2, TAU0_3	Select the Timed Source for measuring device timeout.
Host Notify Supported	Supported, Not Supported	Select whether SMBus Host Notify is supported.

Note1. Set tF and tR according to the user's hardware environment.

Note2. When f_{CLK} is 48 MHz, the filter width of the digital filter is 41.67 ns. To obtain a filter width of 50 ns or more, set f_{CLK} to 32 MHz or lower or use an external noise filter.

Table 3.2 PMBus Master Controller setting items list

Item	Possible values	Description
Support the SMBus Packet Error Checking (PEC)	Supported, Not Supported	Select whether to support PEC.

4. API Specification

4.1 API Typedef Definitions (PMBus Master Controller)

This section describes the Typedef definition provided by the middleware layer of this module.

4.1.1 pmbmctl_ret_t

This typedef defines the return value of the function requesting the sending of the command code from the user.

```
typedef enum
{
    PMBMCTL_RTN_OK,
    PMBMCTL_RTN_CMD_NOT_SUPPORTED,
    PMBMCTL_RTN_DATA_LENGTH_NG
    PMBMCTL_RTN_SENDING
} pmbmctl_ret_t;
```

Description

Use as the return value of the command code sending request function.

(a) PMBMCTL_RTN_OK

When a request for sending a command code can be accepted.

(b) PMBMCTL_RTN_CMD_NOT_SUPPORTED

When a command specified by the user is an unsupported command

(c) PMBMCTL_RTN_DATA_LENGTH_NG

When a data length specified by the user is an invalid value

(d) PMBMCTL_RTN_SENDING

When a previous command code sending is in progress.

4.1.2 pmbmctl_fault_t

This typedef defines the fault information to be notified by the user callback function "FaultNotification".

```
typedef enum
{
    PMBMCTL_FAULT_PEC,
    PMBMCTL_FAULT_DATA_LENGTH,
    PMBMCTL_FAULT_COMMUNICATION
} pmbmctl_fault_t;
```

Description

Used as fault information to be notified by the FaultNotification callback function.

(a) PMBMCTL_FAULT_PEC

Notified when a PEC error is detected.

(b) PMBMCTL_FAULT_DATA_LENGTH

Notified when there is a mismatch between the data received from the PMBus device and the Byte Count in the Block Read and Block Write-Block Read Process Call.

(c) PMBMCTL_FAULT_COMMUNICATION

Notified when PMBMDRV detects a communication error as described above.

4.1.3 pmbmctl_polarity_t

This typedef defines the active level of the Control pin.

```
typedef enum
{
   PMBMCTL_POLARITY_ACTIVE_LO,
   PMBMCTL_POLARITY_ACTIVE_HI,
} pmbmctl_polarity_t;
```

Description

Use the API function "RM_PMBMCTL_SetPolarityControlPin" to set the active level of the Control pin.

4.1.1 pmbmctl_data_t

This typedef defines the data structures that are sent and received via PMBus communication.

```
typedef struct
{
   uint8_t   data_length;
   uint8_t * p_data;
} pmbmctl_data_t;
```

Description

Used for notification of incoming data and setting of sending data by the user callback function.

(a) data_length

Indicates data length [bytes].

(b) p_data

Indicates the initial address of the array variable in which the receiving or sending data is stored.

4.1.1 pmbmctl_callback_t

This typedef defines a user callback function structure.

```
typedef struct
{
   void (* SendCommandSended)(void);
   void (* WriteDataSended)(void);
   void (* ReadDataSended)(pmbmctl_data_t rdata);
   void (* WriteReadDataSended)(pmbmctl_data_t rdata);
   void (* AlertResponseSended)(uint8_t device_address);
   void (* HostNotifyReceived)(uint8_t device_address, uint16_t status_word);
   void (* FaultNotification)(pmbmctl_fault_t fault);
} pmbmctl_callback_t;
```

Description

Register the callback function by storing the user function pointer in this structure and passing it as an argument when calling the API function "RM_PMBMCTL_Open". The callback function will notify the user at each event timing.

(a) SendCommandSended

Notifies at the timing of completion of sending in Send Byte communication format.

(b) WriteDataSended

Notifies at the timing of the completion of sending in Write Byte/Word, Write 32 protocol, Write 64 protocol communication format or Block Write communication format.

(c) ReadDataSended

Notifies when sending is completed in Read Byte/Word, Read 32 protocol, Read 64 protocol communication format or Block Read communication format. The received data is passed as an argument.

(d) WriteReadDataSended

Notifies at the completion of sending in Block Write-Block Read Process Call communication format. The received data is passed as an argument.

(e) AlertResponseSended

Notifies at the timing of the completion of sending in the Device responds to an ARA communication format. The received device address is passed as an argument.

(f) HostNotifyReceived

Notifies at the timing of the completion of reception in Host Notify communication format. The received device address and error information (STATUS_WORD) are passed as arguments.

(g) FaultNotification

Notifies at the timing of Fault detection. The detected fault information is passed as an argument; see 4.1.2 pmbmctl_fault_t for details on fault information.

4.1.2 pmbmctl_hw_signal_t

This typedef defines the H/W signal structure.

```
typedef struct
{
   bool control;
   bool smbalert;
} pmbmctl_hw_signal_t;
```

Description

It is used as the return value of the API function "RM_PMBMCTL_GetHWSignal". Each H/W signal information is defined as a member of the structure.

(a) control

true : Device ON

false : Device OFF

(b) smbalert

true : SMBALERT#signal is being asserted

false : SMBALERT#signal is being negated

4.2 API Function Specifications (PMBus Master Controller)

This section describes the API function specifications provided by the middleware layer of this module.

4.2.1 RM_PMBMCTL_Open

This function initializes the module and starts the PMBus communication feature.

Format

```
void RM_PMBMCTL_Open(const pmbmctl_callback_t * p_callback_set)
```

Parameters

`p_callback_set`

Pointer to user callback function structure

Return Values

None

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Initializes the driver and middleware layers and starts the PMBus communication feature. It also registers the user callback function passed as an argument.

Example

```
/** User function */
static void r_cbk_send_command_sended(void);
static void r_cbk_write_data_sended(void);
static void r_cbk_read_data_sended(pmbmctl_data_t rdata);
static void r_cbk_write_read_data_sended(pmbmctl_data_t rdata);
static void r_cbk_alert_response_sended(uint8_t device_address);
static void r_cbk_host_notify_received(uint8_t device_address, uint16_t status_word);
static void r_cbk_fault_notification(pmbmctl_fault_t fault);
/** Callback function set */
static pmbmctl_callback_t gs_pmbmctl_cbk;
  . . .

/** User Init */
gs_pmbmctl_cbk.SendCommandSended = r_cbk_send_command_sended;
gs_pmbmctl_cbk.WriteDataSended = r_cbk_write_data_sended;
gs_pmbmctl_cbk.ReadDataSended = r_cbk_read_data_sended;
gs_pmbmctl_cbk.WriteReadDataSended = r_cbk_write_read_data_sended;
gs_pmbmctl_cbk.AlertResponseSended = r_cbk_alert_response_sended;
gs_pmbmctl_cbk.HostNotifyReceived = r_cbk_host_notify_received;
gs_pmbmctl_cbk.FaultNotification = r_cbk_fault_notification;
RM_PMBMCTL_Open(gs_pmbmctl_cbk);
```

4.2.2 RM_PMBMCTL_Close

This function performs the module shutdown process and terminates the PMBus communication feature.

Format

```
void RM_PMBMCTL_Close(void)
```

Parameters

None

Return Values

None

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Stops the driver layer and middleware layer and terminates the PMBus communication feature.

Example

```
/** Terminate PMBus communication */
RM_PMBMCTL_Close();
```

4.2.3 RM_PMBMCTL_GetHWSignal

This function obtains H/W signal information.

Format

```
pmbmctl_hw_signal_t RM_PMBMCTL_GetHWSignal(void)
```

Parameters

None

Return Values

H/W Signal Information

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Obtain the following H/W signal information.

・Control Signal input status

・SMBALERT#output status

Example

```
static pmbmctl_hw_signal_t gs_hw_signal;

/* get H/W signal */
gs_hw_signal = RM_PMBMCTL_GetHWSignal();

if (gs_hw_signal.control == true)
{
  . . .
}
```

4.2.4 RM_PMBMCTL_SendCommand

This function requests sending the command code in Send Byte communication format.

Format

```
pmbmctl_ret_t RM_PMBMCTL_SendCommand(uint8_t address, uint8_t command)
```

Parameters

`address`

Destination address

`command`

command

Return Values

Result of accepting a request for sending

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Send command code in the Send Byte communication format.

Example

```
/** Set address */
address = . . .
/** Set command */
command = . . .

/** SendCommand Request */
rtn_value = RM_PMBMCTL_SendCommand(address, command);
```

4.2.1 RM_PMBMCTL_WriteData

This function requests sending the command code in Write Byte/Word, Write 32 protocol, or Write 64 protocol communication format.

Format

```
pmbmctl_ret_t RM_PMBMCTL_WriteData(uint8_t address, uint8_t command,
                                   pmbmctl_data_t wdata)
```

Parameters

`address`

Destination address

`command`

command

`wdata`

sending data

Return Values

Result of accepting a request for sending

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Send command codes in Write Byte/Word, Write 32 protocol, and Write 64 protocol communication formats.

Example

```
/** Set address */
address = . . .
/** Set command */
command = . . .
/** Set send data */
wdata.data_length = . . .
wdata.p_data = . . .

/** WriteData Request */
rtn_value = RM_PMBMCTL_WriteData(address, command, wdata);
```

4.2.2 RM_PMBMCTL_ReadData

This function requests sending the command code in Read Byte/Word, Read 32 protocol, or Read 64 protocol communication format.

Format

```
pmbmctl_ret_t RM_PMBMCTL_ReadData(uint8_t address, uint8_t command,
                                  uint8_t rdata_length)
```

Parameters

`address`

Destination address

`command`

command

`rdata_length`

Receive data length

Return Values

Result of accepting a request for sending

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Send command codes in Read Byte/Word, Read 32 protocol, and Read 64 protocol communication formats.

Example

```
/** Set address */
address = . . .
/** Set command */
command = . . .
/** Set data length */
rdata_length = . . .

/** ReadData Request */
rtn_value = RM_PMBMCTL_ReadData(address, command, rdata_length);
```

4.2.3 RM_PMBMCTL_WriteReadData

This function requests sending the command code in the WriteReadDataSended communication format.

Format

```
pmbmctl_ret_t RM_PMBMCTL_WriteReadData(uint8_t address, uint8_t command,
                                        pmbmctl_data_t wdata,
                                        uint8_t rdata_length)
```

Parameters

`address`

Destination address

`command`

command

`wdata`

sending data

`rdata_length`

Receive data length

Return Values

Result of accepting a request for sending

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Send command codes in WriteReadDataSended communication format.

Example

```
/** Set address */
address = . . .
/** Set command */
command = . . .
/** Set send data */
wdata.data_length = . . .
wdata.p_data = . . .
/** Set data length */
rdata_length = . . .

/** WriteReadData Request */
rtn_value = RM_PMBMCTL_WriteReadData(address, command, wdata, rdata_length);
```

4.2.4 RM_PMBMCTL_SendAlertResponse

This function requests sending the command code in Device responds to an ARA communication format.

Format

```
pmbmctl_ret_t RM_PMBMCTL_SendAlertResponse(void)
```

Parameters

None

Return Values

Result of accepting a request for sending

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Send a command code in the Device responds to an ARA communication format.

Example

```
static pmbmctl_hw_signal_t gs_hw_signal;

/* get H/W signal */
gs_hw_signal = RM_PMBMCTL_GetHWSignal();

if (gs_hw_signal. smbalert == true)
{
  /** Alert Response Request */
  rtn_value = RM_PMBMCTL_SendAlertResponse(void);

}
```

4.2.5 RM_PMBMCTL_SetPolarityControlPin

This function sets the polarity of the Control Signal input pin.

Format

```
void RM_PMBMCTL_SetPolarityControlPin (pmbmctl_polarity_t active_level)
```

Parameters

`active_level`

Control Signal pin active level

Return Values

None

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Set the polarity of the Control Signal input terminal, corresponding to the terminal polarity setting by the ON_OFF_CONFIG command of the PMBus specification.

Example

```
/** Set active level of the Control Signal pin */
RM_PMBMCTL_SetPolarityControlPin(PMBMCTL_POLARITY_ACTIVE_HI);
```

4.2.6 RM_PMBMCTL_SetControl

This function sets the Control signal level.

Format

```
void RM_PMBMCTL_SetControl(bool control_level)
```

Parameters

`control_level`

true :	Activate Control signal
false :	Deactivate Control signal

Return Values

None

Properties

Prototype declared in rm_pmbmctl_api.h.

Description

Set the Control signal level.

Example

```
/** Set Control Signal pin level */
RM_PMBMCTL_SetControl(true);
```

4.3 API Typedef Definitions (PMBus Master Driver)

This section describes the Typedef definition provided by the driver layer of this module. Since the driver layer is basically assumed to be accessed from PMBus Master Controller (middleware layer), users do not need to be particularly aware of it, but it is provided here for reference.

Table 4.1 PMBMDRV port-level enumerated type (pmbmdrv_port_lv_t)

Macro	Macro Value	Description
PMBMDRV_PORT_LO	0	Port Lo level
PMBMDRV_PORT_HI	1	Port Hi level
PMBMDRV_PORT_NOTSUPPORTED	2	Not supported

Table 4.2 PMBMDRVerror-notification enumeration type (pmbmdrv_err_t)

Macro	Macro Value	Description
PMBMDRV_ERR_NACK	0	NACK detection
PMBMDRV_ERR_TIMEOUT	1	Timeout detection

Table 4.3 PMBMDRVcall-back structure (pmbmdrv_callback_t)

Type	Function Pointer Name	Arguments	Description
void	ReceiveByteData	uint8_t data	Data Receiving Notification
void	SendDataEnd	bool stp_cond	Sending Completion Notification
void	ReceiveDataEnd	void	Receiving Completion Notification
void	ErrorNotification	pmbmdrv_err_t error	Error Detection Notification

4.4 API Function Specifications (PMBus Master Driver)

This section describes the API function specifications provided by the driver layer of this module. Since the driver layer is basically assumed to be accessed from PMBus Master Controller (middleware layer), users do not need to be particularly aware of it, but it is provided here as reference information.

Table 4.4 R_PMBMDRV_Open

Function name	R_PMBMDRV_Open	Initialize I2C features and start PMBus communication.
Arguments	const pmbmdrv_callback_t * p_callback_set	Pointer to register callback function to upper layer module
Return value	-	-

Table 4.5 R_PMBMDRV_Close

Function name	R_PMBMDRV_Close	Stop the I2C feature and terminate PMBus communication.
Arguments	-	-
Return value	-	-

Table 4.6 R_PMBMDRV_SendData

Function name	R_PMBMDRV_SendData	Send data to the slave.
Arguments	uint8_t address	Destination address
Arguments	uint8_t *tx_buf	Pointer to sending data
Arguments	uint8_t tx_num	Length of sent data
Arguments	bool sp_flg	Stop Condition Request Flag
Return value	-	-

Table 4.7 R_PMBMDRV_ReceiveData

Function name	R_PMBMDRV_ReceiveData	Receive data from the slave.
Arguments	uint8_t address	Destination address
Arguments	uint8_t *rx_buf	Pointer to received data
Arguments	uint8_t rx_num	Length of received data
Arguments	-	-
Return value		

Table 4.8 R_PMBMDRV_GetControlLevel

Function name	R_PMBMDRV_GetControlLevel	Get the input level of the Control Pin.

Arguments	-	-
Return value	pmbmdrv_port_lv_t level	Port level

Table 4.9 R_PMBMDRV_SetControlLevel

Function name	R_PMBMDRV_SetControlLevel	Set the output level of the Control Pin.
Arguments	pmbmdrv_port_lv_t level	Port level
Return value	-	-

Table 4.10 R_PMBMDRV_SetSMBALertLevel

Function name	R_PMBMDRV_SetSMBALertLevel	Set the output level of the SMBALERT#pin.
Arguments	PMBMDRV_port_lv_t level	Port level
Return value	-	-

Table 4.11 R_PMBMDRV_SetResponseData

Function name	R_PMBMDRV_SetResponseData	Set the response data to the master.
Arguments	uint8_t data_length	Length of sending data
Arguments	uint8_t *p_data	Pointer to sending data
Return value	-	-

Table 4.12 R_PMBMDRV_SendData

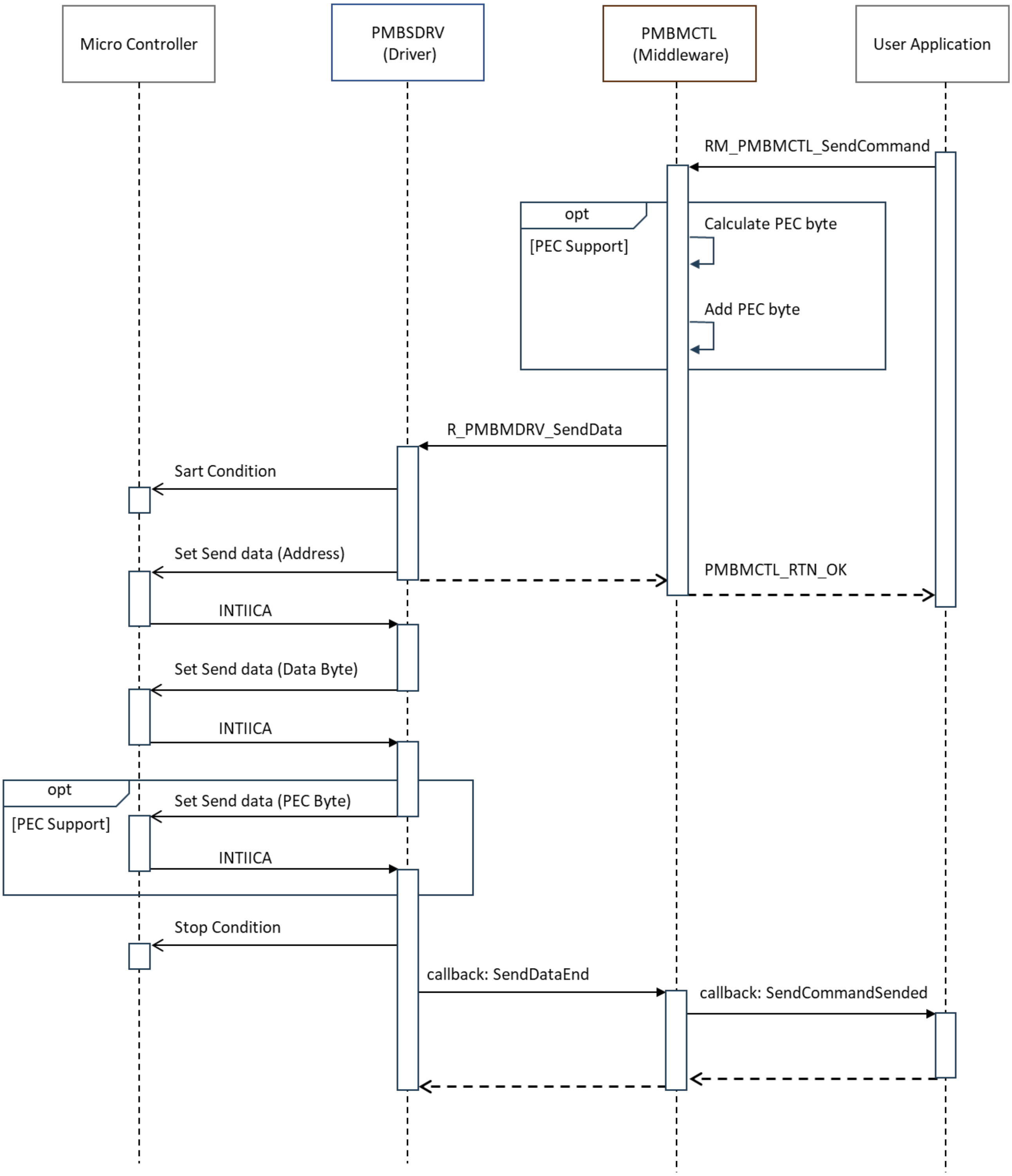
Function name	R_PMBMDRV_SendData	Send any data. (Master sending)
Arguments	uint8_t dest	Destination address
Arguments	uint8_t data_length	Length of sending data
Arguments	uint8_t *p_data	Pointer to sending data
Return value	-	-

5. Operation Sequence

Sequence diagrams for sending command codes and notifying the user for each communication format, as well as for requesting a read to the SMBus Alert Response Address, are shown below.

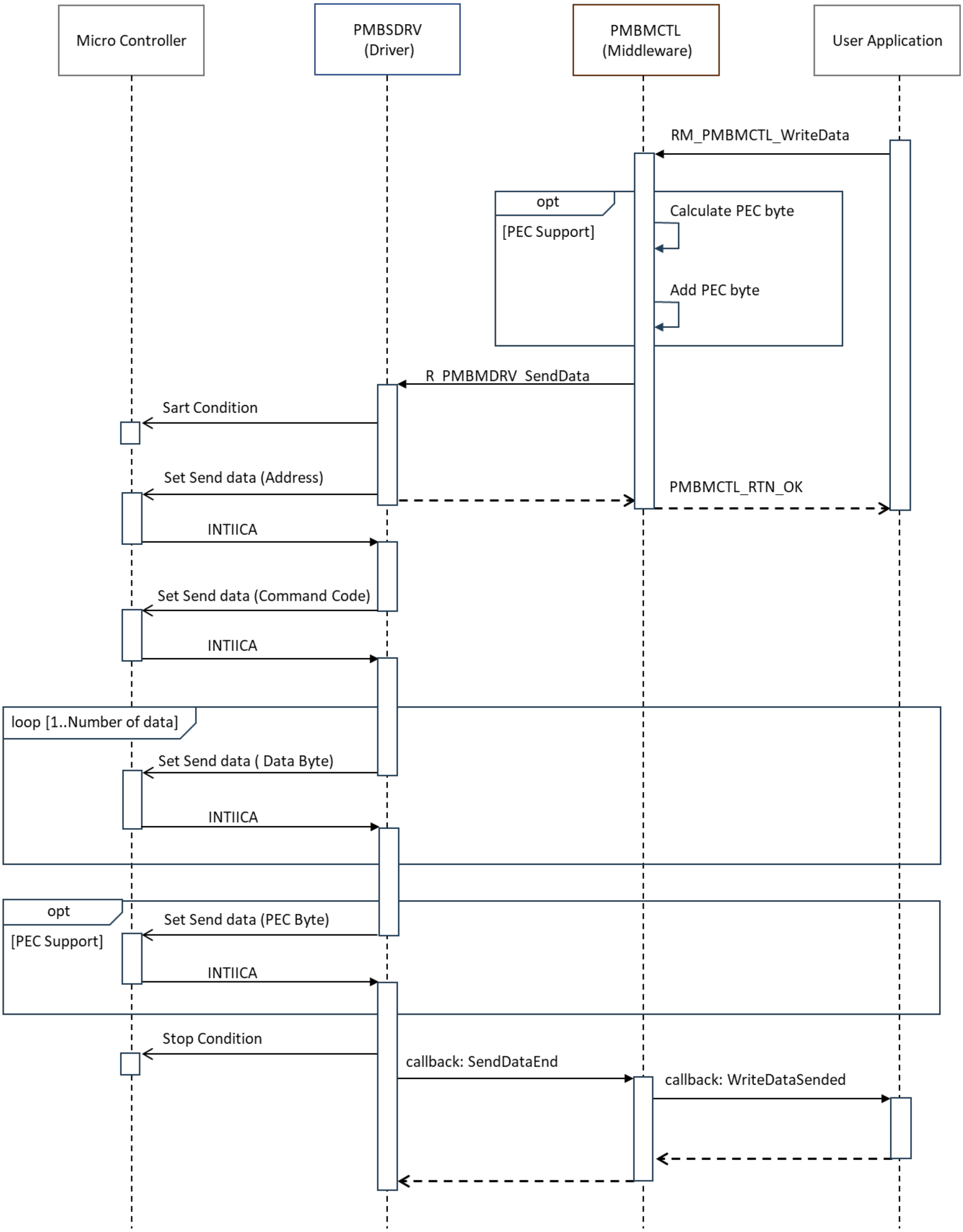
5.1.1 Send Byte

Figure 5-1 Send Byte sequence



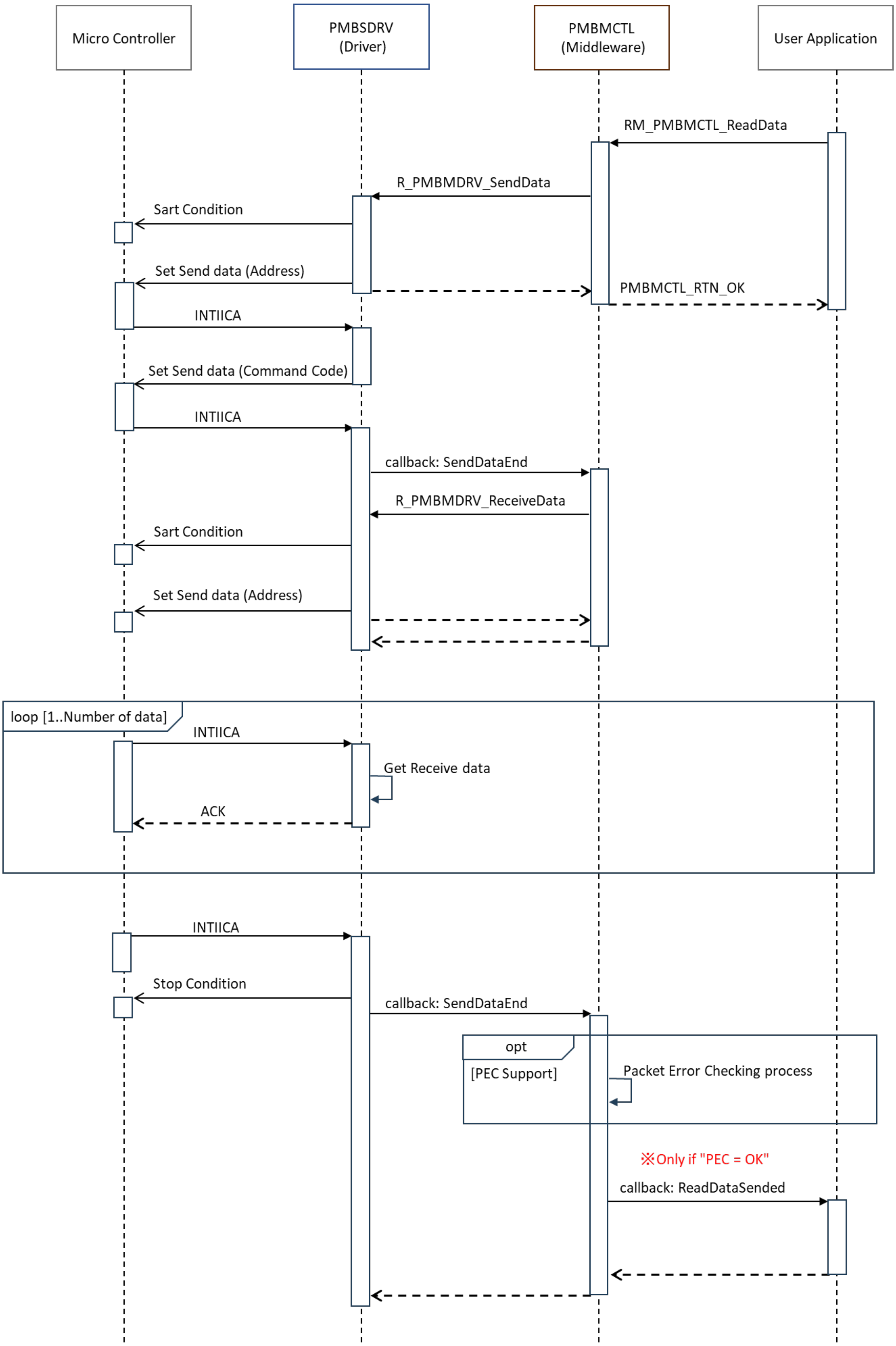
5.1.2 Write Byte/Word, Write 32 protocol, Write 64 protocol

Figure 5-2 Write sequence



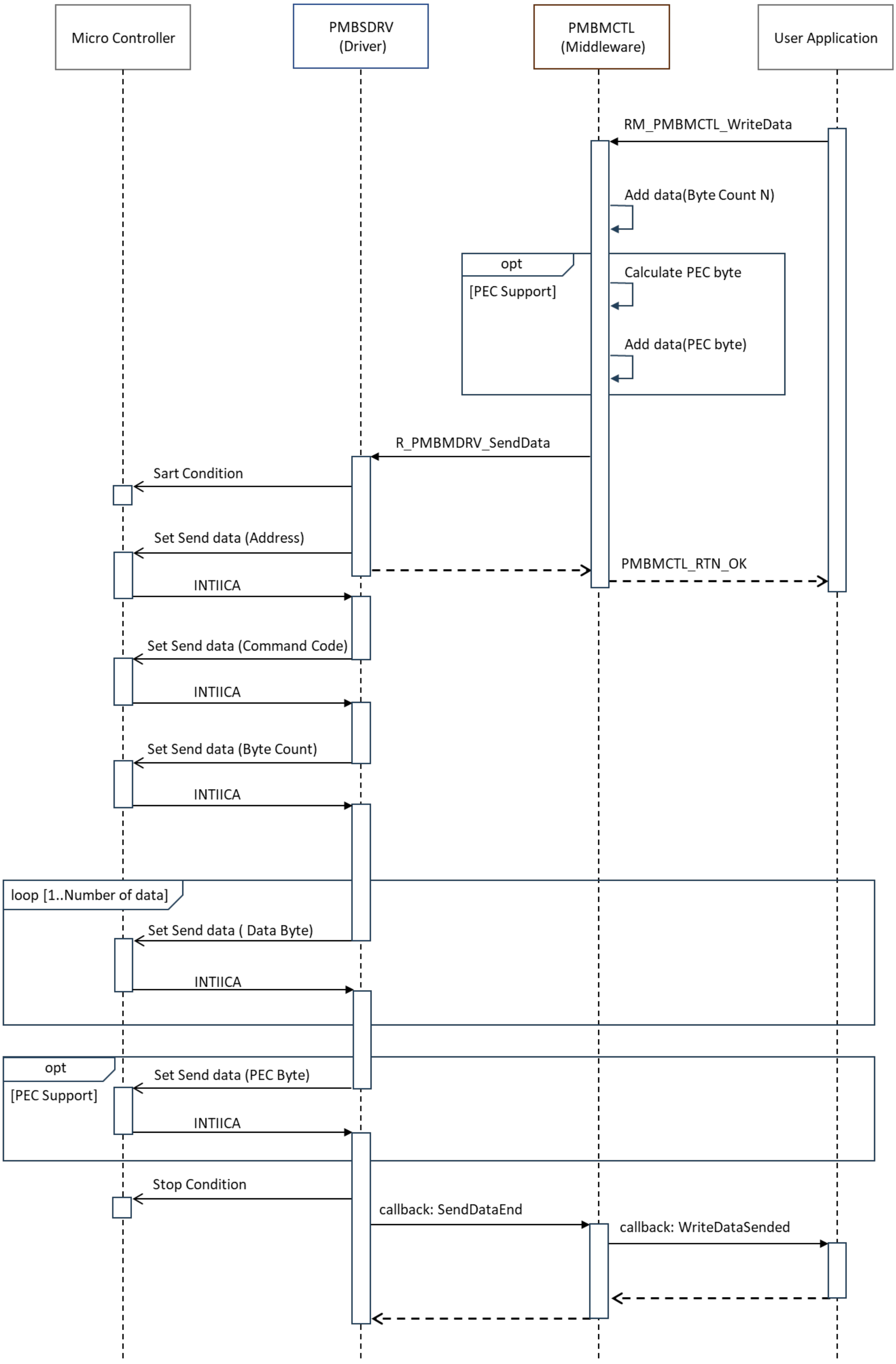
5.1.3 Read Byte/Word, Read 32 protocol, Read 64 protocol

Figure 5-3 Read sequence



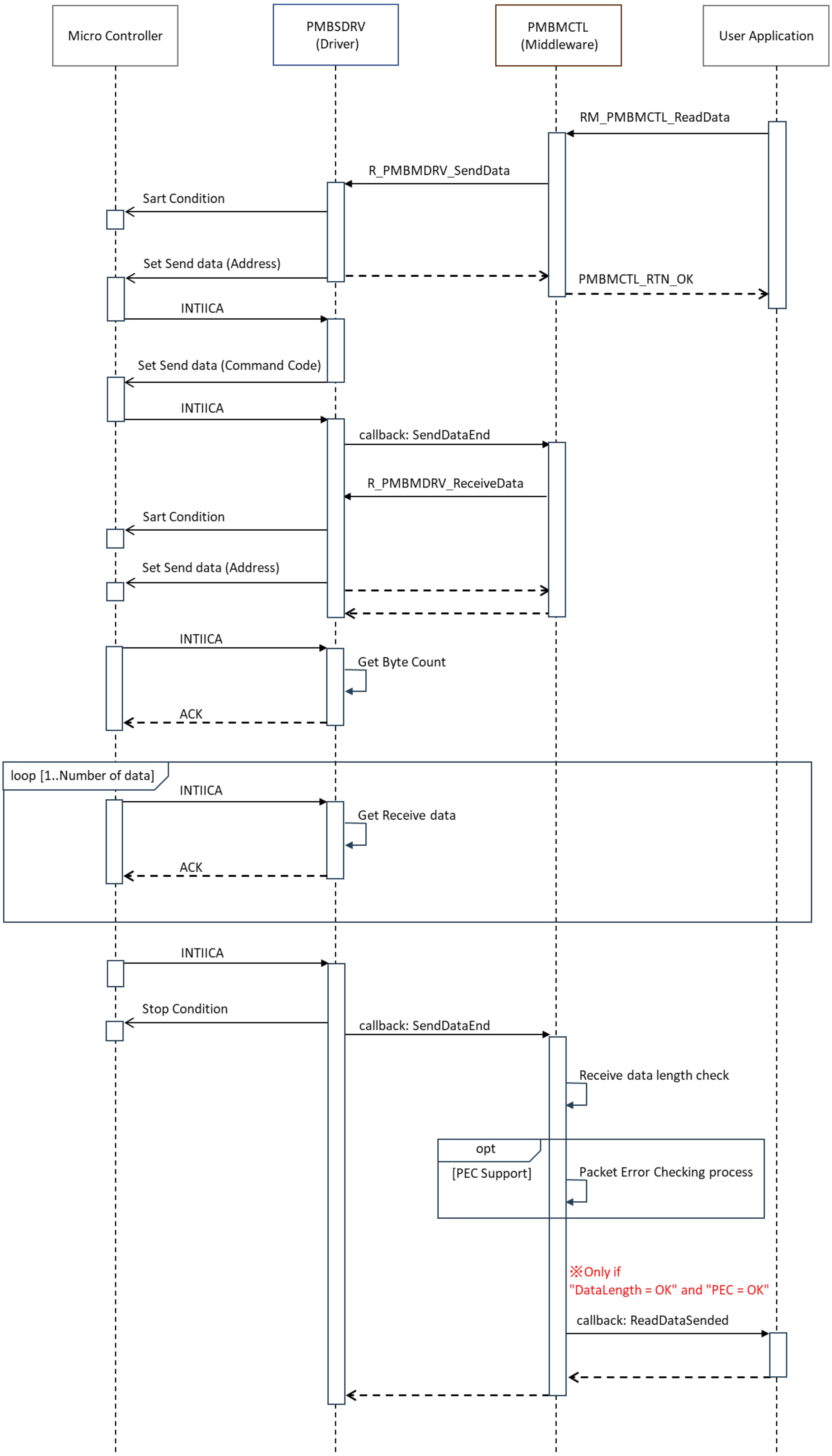
5.1.4 Block Write

Figure 5-4 Block Write sequence



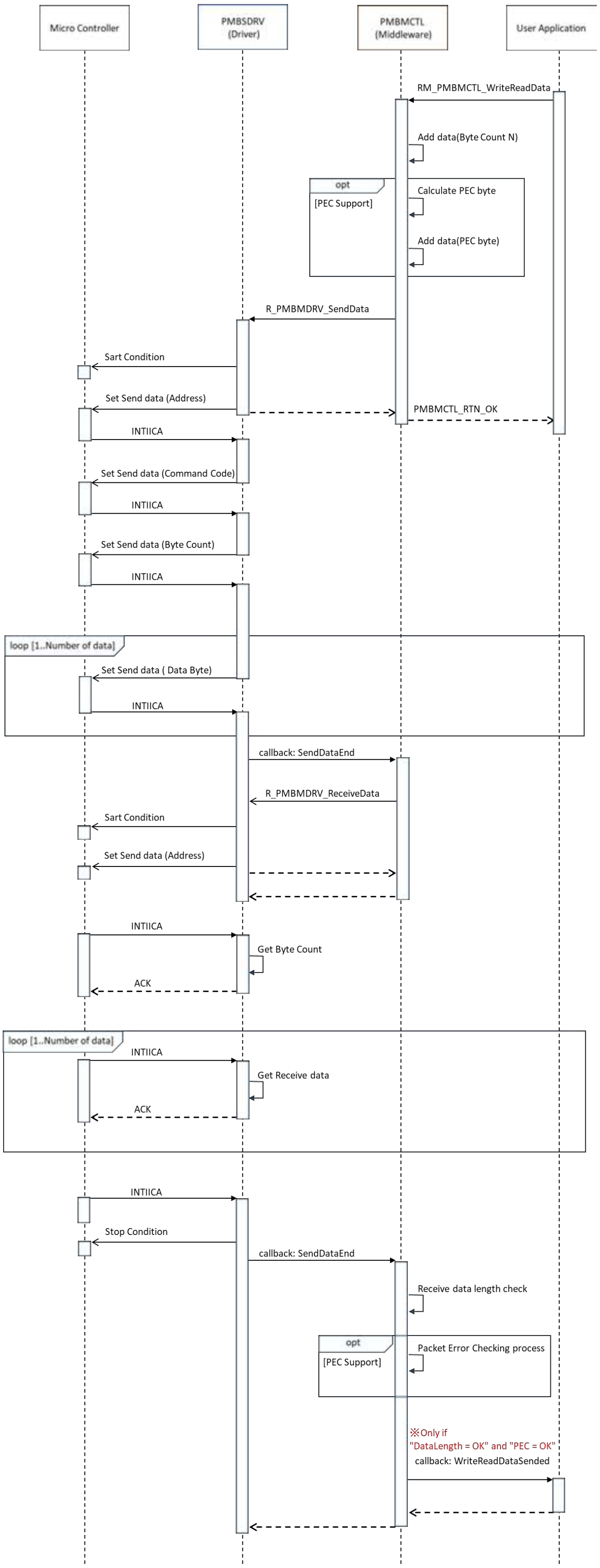
5.1.5 Block Read

Figure 5-5 Block Read sequence



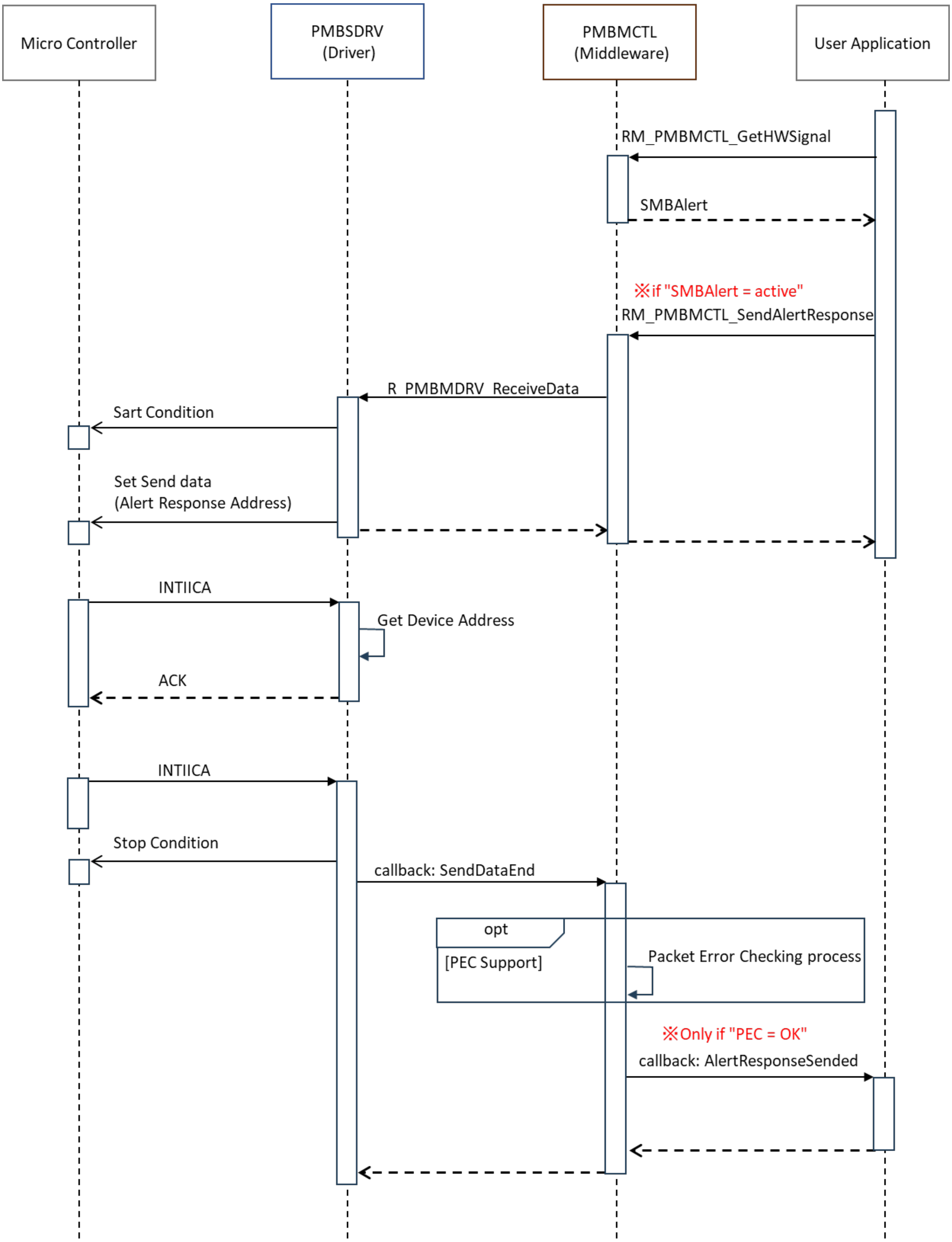
5.1.6 Block Write-Block Read Process Call

Figure 5-6 Block Write-Block Read Process sequence



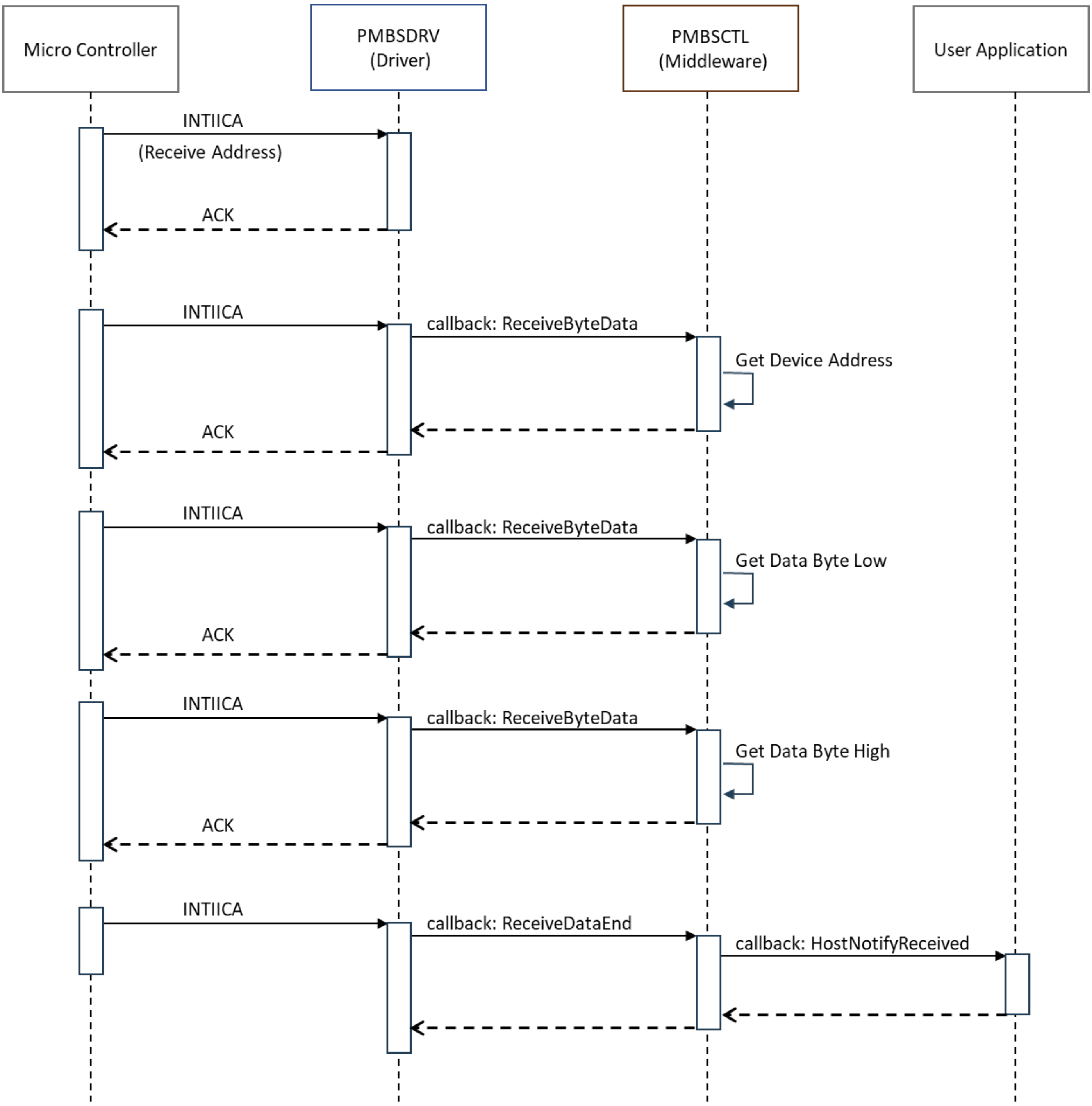
5.1.7 SMBALERT#

Figure 5-7 SMBALERT# sequence



5.1.8 SMBus Host Notify

Figure 5-8 SMBus Host Notify sequence



6. Website and Support

Renesas Electronics Website

http://www.renesas.com/

Contact information

http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.18, 2024	-	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.