RL78 ファミリ

PMBus Master モジュール Software Integration System

要旨

本アプリケーションノートは PMBus Master モジュールについて説明します。

対象デバイス

RL78/G24

関連ドキュメント

・RL78/G24 ユーザーズマニュアル ハードウェア編 (R01UH0961J)

・PMBus Specification Rev. 1.4 Part I

・PMBus Specification Rev. 1.4 Part II

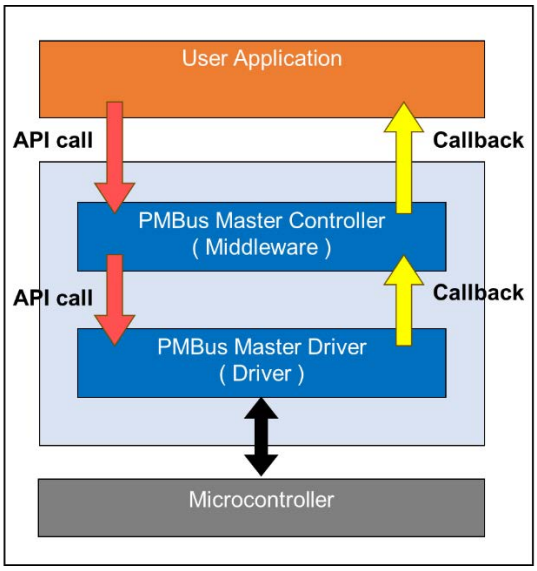
・System Management Bus (SMBus) Specification Version 3.2

目次

1. 概要

本モジュールはドライバ層(PMBus Master Driver)、ミドルウェア層(PMBus Master Controller)にて構成され、PMBus (Power Management Bus) 通信によるデータの送受信を行うためのインターフェースを提供します。

図 1-1　モジュール構成



1.1　PMBus Master Driver (PMBMDRV) 機能

PMBus Master Driver はミドルウェア層(PMBus Master Controller)からのアクセスを想定しています。

PMBus Master モジュールのドライバ層として、下記の機能を提供します。

1.1.1　バス通信機能

シリアル・インターフェース IICA を利用して PMBus のマスタ送受信動作を行います。

(1)　バス速度

Smart Configurator にて、100kHz、400kHz、1MHz の設定が可能です。

注意事項:

1MHz 設定時は外部回路にてレベルシフタを設ける必要があります。

(2)　シリアルバス通信

Microcontroller のシリアル・インターフェース IICA を利用し、下記の通信動作を行います。

- スタートコンディションの生成
- リピートスタートコンディションの生成
- データの送受信
- ストップコンディションの生成

(3) 上位レイヤへの通知

下記のタイミングで上位レイヤへの通知を行います。

- 送信完了
- 受信完了
- 通信エラー検出 (詳細については次章にて説明します)

1.1.2 通信エラー検出

PMBus Master モジュールのドライバ層では下記通信エラーの検出を行い、上位レイヤへの通知を行います。

(1) NACK 検出

スレーブへのデータ送信時に NACK を検出した場合、NACK 検出と判断します。

(2) 通信タイムアウト

通信トランザクション中に Byte データ受信タイミングの監視を行い、受信間隔が 25ms 以上となった場合に通信タイムアウトと判断します。Byte データ受信タイミングの監視は INTIICA0 割り込みの検出により実現します。これにより、SCLA0 ラインの Low 固着状態の検出が可能となります。

1.1.3 オプション端子機能

Smart Configurator にて下記オプション端子のポート割り当てを行うことができます。

(1) Control Signal (CONTROL)

CONTROL 端子は任意の入力端子です。PMBus コマンドと連動して、デバイスの ON/OFF を行うために使用されます。また、PMBus コマンドにより端子のアクティブレベルを設定する事ができます。

(2) SMBALERT#

SMBALERT#端子は任意の出力端子です。スレーブからマスタへの割り込みラインとして使用します。

注意事項:

これらの端子の電気特性は"RL78/G24 ユーザーズマニュアル ハードウェア編"に記載の電気特性に従います。SMBus 仕様に記載の電気特性に準拠するためには、外部回路による対応が必要となります。

1.2　PMBus Master Controller (PMBMCTL) 機能

PMBus Master Controller はユーザーアプリケーションからのアクセスを想定しています。

PMBus Master モジュールのミドルウェア層として、下記の機能を提供します。

1.2.1　通信フォーマット

ユーザーアプリケーションからのコマンドコード送信要求を受けて、スタートコンディション生成時はデータ送信要求、リピートスタートコンディション生成時はデータ受信要求を PMBMDRV に対して行います。以下に本モジュールでサポートする通信フォーマットとそのプロトコル図を記載します。

プロトコル図の凡例：

S : スタートコンディション

Sr : リピートスタートコンディション

Rd : リード(1)

Wr : ライト(0)

ACK : アクノリッジ

NACK : Not アクノリッジ

PEC : Packet Error Code　　※PEC のサポートはオプションです

P : ストップコンディション

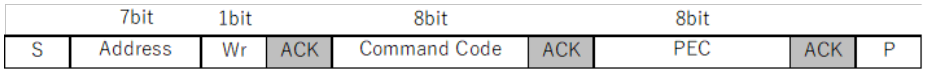
□ : マスタ → スレーブ

■ : スレーブ → マスタ

(1)　Send Byte

任意のターゲットアドレスに対し、コマンドコードを送信します。

図 1-2　通信フォーマット Send Byte

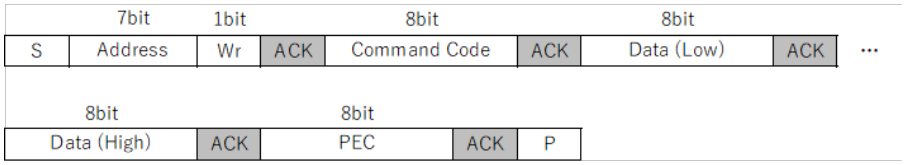


(2)　Write Byte/Word, Write 32 protocol, Write 64 protocol

任意のターゲットアドレスに対し、コマンドコードとそれに対応する書き込みデータを送信します。

図 1-3　通信フォーマット Write (Single byte)

図 1-4　通信フォーマット Write (Multi byte)



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

任意のターゲットアドレスに対し、コマンドコードを送信します。次にそのデバイスからコマンドコードに対応する応答データを読み出します。

図 1-5　通信フォーマット Read (Single byte)

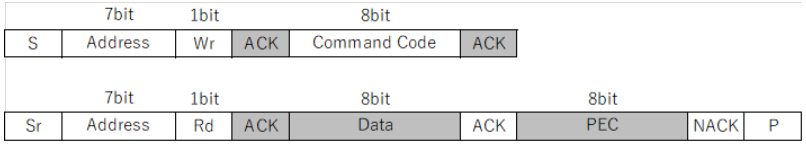
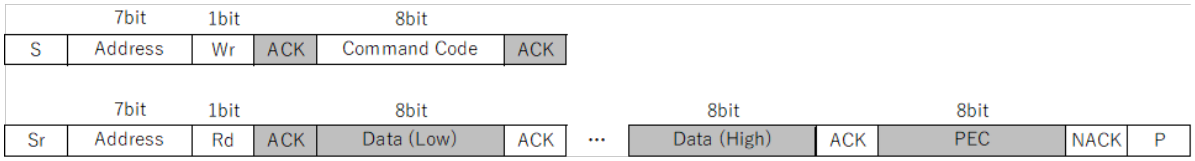


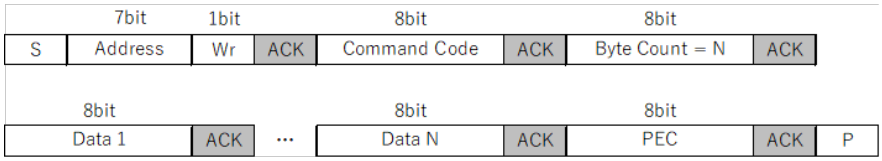
図 1-6　通信フォーマット Read (Multi byte)



(4)　Block Write

任意のターゲットアドレスに対し、コマンドコードとそれに対応する書き込みデータを送信します。書き込みデータ送信前にデータ長を示すデータ(Byte Count)を送信します。

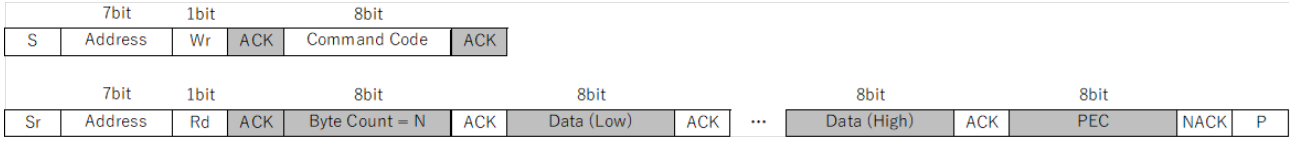
図 1-7　通信フォーマット Block Write



(5)　Block Read

任意のターゲットアドレスに対し、コマンドコードを送信します。次にそのデバイスからコマンドコードに対応する応答データを読み出します。デバイスは応答データの送信前にデータ長を示すデータ(Byte Count)を送信します。

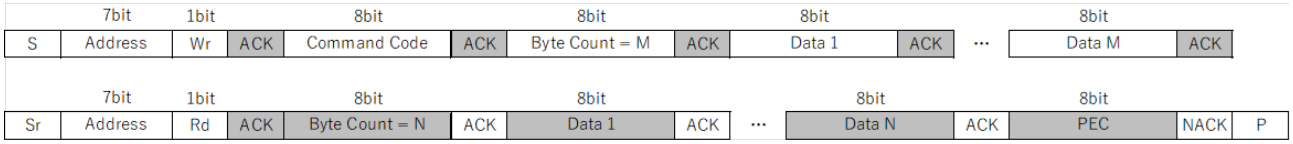
図 1-8　通信フォーマット Block Read



(6)　Block Write-Block Read Process Call

前述した Block Write と Block Read を同トランザクション内で実行します。書き込みデータ長(Byte Count M)と応答データ長(Byte Count N)は異なる場合があります。

図 1-9　通信フォーマット Block Write-Block Read Process Call



1.2.2 送受信完了通知

前述した通信フォーマットに従いコマンド送信が完了した際に、ユーザーへ送信完了を通知します。

(1) Send コマンド送信完了通知

Send Byte 通信フォーマットでの送信完了をユーザーへコールバックにて通知します。

(2) Write データ送信完了通知

Write Byte/Word, Write 32 protocol, Write 64 protocol 通信フォーマットまたは Block Write 通信フォーマットでの送信完了をユーザーへコールバックにて通知します。

(3) Read データ受信完了通知

Read Byte/Word, Read 32 protocol, Read 64 protocol 通信フォーマットまたは Block Read 通信フォーマットでの送信完了時に受信データをユーザーへコールバックにて通知します。

(4) Write/Read データ送信完了通知

Block Write-Block Read Process Call 通信フォーマットでの送信完了時に受信データをユーザーへコールバックにて通知します。

(5) AlertResponseAddress 送信完了通知

Device responds to an ARA 通信フォーマットでの送信完了時に受信したデバイスアドレスをユーザーへコールバックにて通知します。

(6) HostNotify 受信完了通知

Host Notify 通信フォーマットでの受信完了時に受信したデバイスアドレス、異常情報(STATUS_WORD)をユーザーへコールバックにて通知します。

1.2.3 Fault 検出とユーザー通知

データ通信中に発生した各障害の検出を行い、ユーザーへコールバックにて通知します。詳細については 4.1.2 pmbmctl_fault_t 章を参照ください。

1.2.4　コントローラへの報告

PMBus デバイスは異常検出時、警告や Fault 状態をコントローラ(マスタ)に通知する必要があります。本モジュールは通知に対して下記の 2 つの機能を提供します。PMBus 仕様ではこの 2 つの方法のうち、少なくとも 1 つをサポートしなければならないことが定められています。

(1)　SMBALERT#信号による通知

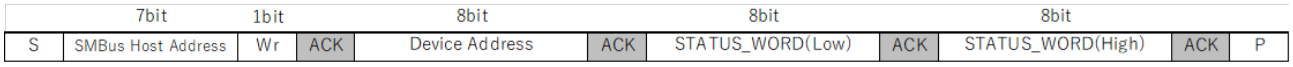
PMBus デバイスは、SMBALERT#端子よりアラート信号を発生させることができます。ユーザーは API 関数” RM_PMBMCTL_GetHWSignal”をコールすることにより、SMBALERT#端子の信号状態を取得することができます。また、通知元のデバイスを特定するために、API 関数” RM_PMBMCTL_SendAlertResponse”をコールすることにより、SMBus Alert Response Address への読み出し要求を行うことができます。この要求に対し、PMBus デバイスは自局アドレス値を応答します。

図 1-10　通信フォーマット Device responds to an ARA

(2)　SMBus Host Notify プロトコルによる通知

PMBus デバイスはデバイスアドレスと異常情報(STATUS_WORD)を SMBus Host Notify プロトコルにて通知することができます。ユーザーは前述した HostNotify 受信完了通知にてデバイスアドレス、異常情報(STATUS_WORD)を取得することができます。

図 1-11　通信フォーマット Host Notify



1.2.5　H/W シグナル情報取得

ユーザーは API 関数”RM_PMBMCTL_GetHWSignal”をコールすることにより下記オプション端子の信号レベルを取得することができます。

- Control Signal (CONTROL)
- SMBALERT#

2. API 情報

本モジュールの API 情報について説明します。

2.1 ハードウェアの要求

ご使用になる MCU が以下の端子をサポートしている必要があります。

- SCLA0 (P60)
- SDAA0 (P61)

対象製品：30,32,40,44,48,52,64 ピン製品

2.2 ソフトウェアの要求

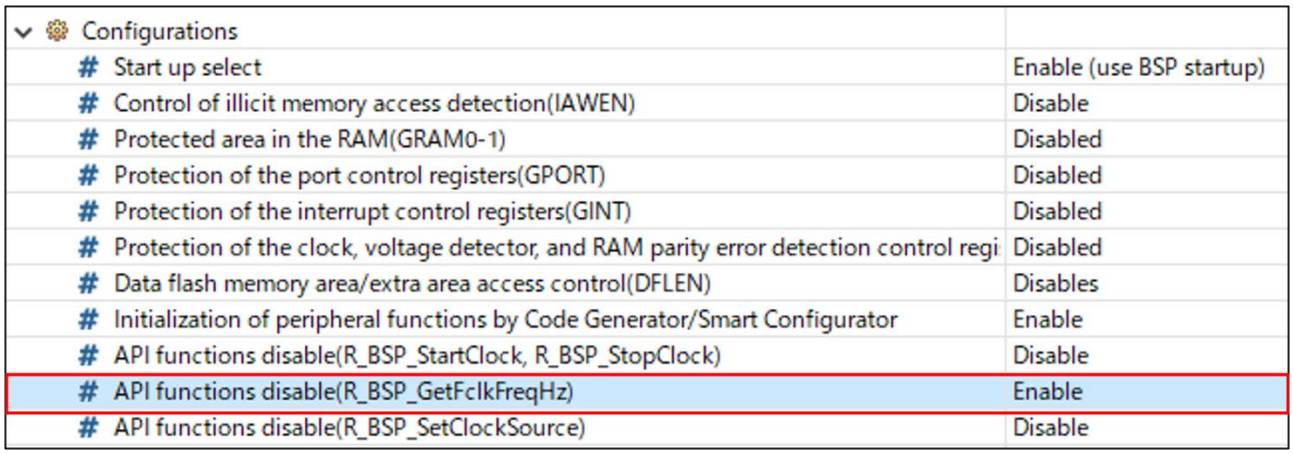
このドライバは以下のモジュールに依存しています。

- ボードサポートパッケージ (r_bsp) v1.61 以降

また、r_bsp の下記 API 関数を有効化する必要があります。Smart Configurator 上のソフトウェアコンポーネント設定画面より設定を行ってください。

- R_BSP_GetFclkFreqHz
  (BSP_CFG_GET_FREQ_API_FUNCTIONS_DISABLE = 0)

図 2-1 Smart Configurator BSP 設定



2.3 サポートされているツールチェーン

本モジュールは以下に示すツールチェーンで動作確認を行っています。

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

2.4 ヘッダファイル

API 呼び出しと使用される I/F 定義は”rm_pmbmctl_api.h”および”r_pmbmdrv_api.h”に記載されています。

2.5 整数型

このドライバは ANSI C99 を使用しています。これらの型は”stdint.h”で定義されています。

2.6 コードサイズ

ROM および RAM サイズは、Smart Configurator 上での設定値やコンパイラのオプション設定に応じて増減します。ここでは Smart Configurator 上の設定をデフォルト設定とし、CC-RL コンパイラでコンパイルオプションをデフォルト設定とした場合のサイズを参考値として記載します。

ROM : 3,378 [byte]

RAM : 1,377 [byte]

3. コンフィグレーション仕様

以下に Smart Configurator 上で設定可能なコンフィグレーション項目一覧を示します。

表 3.1 PMBus Master Driver 設定項目一覧

項目	取りうる値	説明
Bus Speed	100kHz, 400kHz、1MHz	バス速度を指定します。
IICA Input Mode	I2C, SMBus	IICA 入力モードを指定します。
SDAA and SCLA signal falling times (tF)[ns]	0～300	SDAA および SCLA 信号の立ち下がり時間を設定します。(注 1)
SDAA and SCLA signal rising times (tR)[ns]	0～1000	SDAA および SCLA 信号の立ち上がり時間を設定します。(注 1)
Digital filter	ON, OFF	デジタル・フィルタを有効にするかどうかを指定します。(注 2)
Interrupt level for INTIICA0	Level 0(Highest), Level 1, Level 2, Level 3(Lowest)	INTIICA0 割り込みの優先度を設定します。
Pin for Control Signal	Unused, P00～P147	Control Signal 端子を選択します。
Pin for SMBALERT#	Unused, P00～P147	SMBALERT#端子を選択します。
Timer resource for device timeout measurement	TAU0_0, TAU0_1, TAU0_2, TAU0_3	デバイスタイムアウトを計測するためのタイマリソースを選択します。
Host Notify Supported	Supported, Not Supported	SMBus Host Notify をサポートするかどうかを選択します。

注1. tF および tR はユーザーのハードウェア環境に応じて設定してください。

注2. f_{CLK} が 48 MHz の場合はデジタル・フィルタのフィルタ幅が 41.67 ns になります。50 ns 以上のフィルタ幅を確保する場合は f_{CLK} を 32 MHz 以下に設定するか、外付けでノイズ・フィルタを使用してください。

表 3.2 PMBus Master Controller 設定項目一覧

項目	取りうる値	説明
Support the SMBus Packet Error Checking (PEC)	Supported, Not Supported	PEC をサポートするかどうかを選択します。

4. API 仕様

4.1 API Typedef 定義 (PMBus Master Controller)

本モジュールのミドルウェア層が提供する Typedef 定義について説明します。

4.1.1 pmbmctl_ret_t

この Typedef はユーザーからのコマンドコード送信要求関数の戻り値を定義します。

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

Description

コマンドコード送信要求関数の戻り値として使用してください。

(a) PMBMCTL_RTN_OK

コマンドコード送信要求が受け付け可能である場合

(b) PMBMCTL_RTN_CMD_NOT_SUPPORTED

ユーザーから指定されたコマンドがサポート外コマンドの場合

(c) PMBMCTL_RTN_DATA_LENGTH_NG

ユーザーから指定されたデータ長が無効値であった場合

(d) PMBMCTL_RTN_SENDING

前のコマンドコード送信が送信処理中の場合

4.1.2 pmbmctl_fault_t

この Typedef はユーザーコールバック関数”FaultNotification”にて通知される Fault 情報を定義します。

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

Description

FaultNotification コールバック関数にて通知される Fault 情報として使用されます。

(a) PMBMCTL_FAULT_PEC

PEC エラーを検出したときに通知されます。

(b) PMBMCTL_FAULT_DATA_LENGTH

Block Read および Block Write-Block Read Process Call にて PMBus デバイスから受信したデータと Byte Count が不整合の場合に通知されます。

(c) PMBMCTL_FAULT_COMMUNICATION

PMBMDRV にて前述した通信エラーを検出したときに通知されます。

4.1.3 pmbmctl_polarity_t

この Typedef は Control 端子のアクティブレベルを定義します。

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

Description

API 関数” RM_PMBMCTL_SetPolarityControlPin”にて Control 端子のアクティブレベルを設定する際に使用してください。

4.1.1 pmbmctl_data_t

この Typedef は PMBus 通信によって送受信されるデータ構造体を定義します。

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

Description

ユーザーコールバック関数による受信データ通知および送信データの設定時に使用されます。

(a) data_length

データ長 [byte] を示します。

(b) p_data

受信データまたは送信データが格納された配列変数の先頭アドレスを示します。

4.1.1 pmbmctl_callback_t

この Typedef はユーザーコールバック関数構造体を定義します。

```
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

ユーザー関数ポインタをこちらの構造体に格納し、API 関数”RM_PMBMCTL_Open”コール時に引数として渡すことで、コールバック関数を登録してください。各イベントタイミングでコールバックによるユーザー通知を行います。

(a) SendCommandSended

Send Byte 通信フォーマットでの送信完了のタイミングで通知を行います。

(b) WriteDataSended

Write Byte/Word, Write 32 protocol, Write 64 protocol 通信フォーマットまたは Block Write 通信フォーマットでの送信完了のタイミングで通知を行います。

(c) ReadDataSended

Read Byte/Word, Read 32 protocol, Read 64 protocol 通信フォーマットまたは Block Read 通信フォーマットでの送信完了のタイミングで通知を行います。受信したデータが引数で渡されます。

(d) WriteReadDataSended

Block Write-Block Read Process Call 通信フォーマットでの送信完了のタイミングで通知を行います。受信したデータが引数で渡されます。

(e) AlertResponseSended

Device responds to an ARA 通信フォーマットでの送信完了のタイミングで通知を行います。受信したデバイスアドレスが引数で渡されます。

(f) HostNotifyReceived

Host Notify 通信フォーマットでの受信完了のタイミングで通知を行います。受信したデバイスアドレス、異常情報(STATUS_WORD)が引数で渡されます。

(g) FaultNotification

Fault 検出タイミングで通知を行います。検出した Fault 情報が引数で渡されます。Fault 情報の詳細については 4.1.2 pmbmctl_fault_t を参照してください。

4.1.2 pmbmctl_hw_signal_t

この Typedef は H/W シグナル構造体を定義します。

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

Description

API 関数”RM_PMBMCTL_GetHWSignal”の戻り値として使用されます。各 H/W シグナル情報が構造体のメンバとして定義されています。

(a) control

true : デバイス ON

false : デバイス OFF

(b) smbalert

true : SMBALERT#信号をアサート中

false : SMBALERT#信号をネゲート中

4.2 API 関数仕様 (PMBus Master Controller)

本モジュールのミドルウェア層が提供する API 関数仕様について説明します。

4.2.1 RM_PMBMCTL_Open

この関数はモジュールの初期化を行い、PMBus 通信機能を開始します。

Format

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

Parameters

`p_callback_set`

ユーザーコールバック関数構造体のポインタ

Return Values

なし

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

ドライバ層、ミドルウェア層の初期化を行い、PMBus 通信機能を開始します。また、引数で渡されたユーザーコールバック関数の登録を行います。

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

この関数はモジュールの停止処理を行い、PMBus 通信機能を終了します。

Format

```
void RM_PMBMCTL_Close(void)
```

Parameters

なし

Return Values

なし

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

ドライバ層、ミドルウェア層の停止を行い、PMBus 通信機能を終了します。

Example

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

4.2.3 RM_PMBMCTL_GetHWSignal

この関数は H/W シグナル情報を取得します。

Format

```
pmbmctl_hw_signal_t RM_PMBMCTL_GetHWSignal(void)
```

Parameters

なし

Return Values

H/W シグナル情報

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

下記 H/W シグナル情報を取得します。

・Control Signal 入力状態

・SMBALERT#出力状態

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

この関数は Send Byte 通信フォーマットでのコマンドコード送信要求を行います。

Format

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

Parameters

address

送信先アドレス

command

コマンド

Return Values

送信要求受付結果

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

Send Byte 通信フォーマットでのコマンドコード送信を行います。

Example

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

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

4.2.1 RM_PMBMCTL_WriteData

この関数は Write Byte/Word, Write 32 protocol, Write 64 protocol 通信フォーマットでのコマンドコード送信要求を行います。

Format

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

Parameters

address

送信先アドレス

command

コマンド

wdata

送信データ

Return Values

送信要求受付結果

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

Write Byte/Word, Write 32 protocol, Write 64 protocol 通信フォーマットでのコマンドコード送信を行います。

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

この関数は Read Byte/Word, Read 32 protocol, Read 64 protocol 通信フォーマットでのコマンドコード送信要求を行います。

Format

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

Parameters

address

送信先アドレス

command

コマンド

rdata_length

受信データ長

Return Values

送信要求受付結果

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

Read Byte/Word, Read 32 protocol, Read 64 protocol 通信フォーマットでのコマンドコード送信を行います。

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

この関数は WriteReadDataSended 通信フォーマットでのコマンドコード送信要求を行います。

Format

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

Parameters

address

送信先アドレス

command

コマンド

wdata

送信データ

rdata_length

受信データ長

Return Values

送信要求受付結果

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

WriteReadDataSended 通信フォーマットでのコマンドコード送信を行います。

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

この関数は Device responds to an ARA　通信フォーマットでのコマンドコード送信要求を行います。

Format

```
pmbmctl_ret_t RM_PMBMCTL_SendAlertResponse(void)
```

Parameters

なし

Return Values

送信要求受付結果

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

Device responds to an ARA 通信フォーマットでのコマンドコード送信を行います。

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

この関数は Control Signal 入力端子の極性を設定します。

Format

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

Parameters

`active_level`

Control Signal 端子アクティブレベル

Return Values

なし

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

Control Signal 入力端子の極性を設定します。PMBus 仕様の ON_OFF_CONFIG コマンドによる端子極性の設定に対応します。

Example

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

4.2.6 RM_PMBMCTL_SetControl

この関数は Control 信号レベルを設定します。

Format

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

Parameters

`control_level`

true : Control 信号をアクティブ

false : Control 信号を非アクティブ

Return Values

なし

Properties

rm_pmbmctl_api.h にプロトタイプ宣言されています。

Description

Control 信号レベルを設定します。

Example

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

4.3 API Typedef 定義 (PMBus Master Driver)

本モジュールのドライバ層が提供する Typedef 定義について説明します。ドライバ層は基本的に PMBus Master Controller (ミドルウェア層)よりアクセスされることを想定しているため、ユーザーは特に意識する必要はありませんが、参考情報として記載します。

表 4.1 PMBMDRV ポートレベル列挙型(pmbmdrv_port_lv_t)

マクロ	マクロ値	説明
PMBMDRV_PORT_LO	0	ポート Lo レベル
PMBMDRV_PORT_HI	1	ポート Hi レベル
PMBMDRV_PORT_NOTSUPPORTED	2	サポート外

表 4.2 PMBMDRV エラー通知列挙型(pmbmdrv_err_t)

マクロ	マクロ値	説明
PMBMDRV_ERR_NACK	0	NACK 検出
PMBMDRV_ERR_TIMEOUT	1	タイムアウト検出

表 4.3 PMBMDRV コールバック構造体(pmbmdrv_callback_t)

型	関数ポインタ名	引数	説明
void	ReceiveByteData	uint8_t data	データ受信通知
void	SendDataEnd	bool stp_cond	送信完了通知
void	ReceiveDataEnd	void	受信完了通知
void	ErrorNotification	pmbmdrv_err_t error	エラー検出通知

4.4 API 関数仕様 (PMBus Master Driver)

本モジュールのドライバ層が提供する API 関数仕様について説明します。ドライバ層は基本的に PMBus Master Controller (ミドルウェア層)よりアクセスされることを想定しているため、ユーザーは特に意識する必要はありませんが、参考情報として記載します。

表 4.4 R_PMBMDRV_Open

関数名	R_PMBMDRV_Open	I2C 機能の初期化を行い、PMBus 通信を開始します。
引数	const pmbmdrv_callback_t * p_callback_set	上位層モジュールへのコールバック関数を登録するためのポインタ
戻り値	-	-

表 4.5 R_PMBMDRV_Close

関数名	R_PMBMDRV_Close	I2C 機能の停止処理を行い、PMBus 通信を終了します。
引数	-	-
戻り値	-	-

表 4.6 R_PMBMDRV_SendData

関数名	R_PMBMDRV_SendData	スレーブにデータ送信をします。
引数	uint8_t address	送信先アドレス
引数	uint8_t *tx_buf	送信データのポインタ
引数	uint8_t tx_num	送信データ長
引数	bool sp_flg	ストップコンディション要求フラグ
戻り値	-	-

表 4.7 R_PMBMDRV_ReceiveData

関数名	R_PMBMDRV_ReceiveData	スレーブからデータ受信をします。
引数	uint8_t address	送信先アドレス
引数	uint8_t *rx_buf	受信データのポインタ
引数	uint8_t rx_num	受信データ長
戻り値	-	-

表 4.8 R_PMBMDRV_GetControlLevel

関数名	R_PMBMDRV_GetControlLevel	Control Pin の入力レベルを取得します。
引数	-	-
戻り値	pmbmdrv_port_lv_t level	ポートレベル

表 4.9 R_PMBMDRV_SetControlLevel

関数名	R_PMBMDRV_SetControlLevel	Control Pin の出力レベルを設定します。
引数	pmbmdrv_port_lv_t level	ポートレベル
戻り値	-	-

表 4.10 R_PMBMDRV_SetSMBALertLevel

関数名	R_PMBMDRV_SetSMBALertLevel	SMBALERT#端子の出力レベルを設定します。
引数	PMBMDRV_port_lv_t level	ポートレベル
戻り値	-	-

表 4.11 R_PMBMDRV_SetResponseData

関数名	R_PMBMDRV_SetResponseData	マスタへの応答データを設定します。
引数	uint8_t data_length	送信データ長
引数	uint8_t *p_data	送信データのポインタ
戻り値	-	-

表 4.12 R_PMBMDRV_SendData

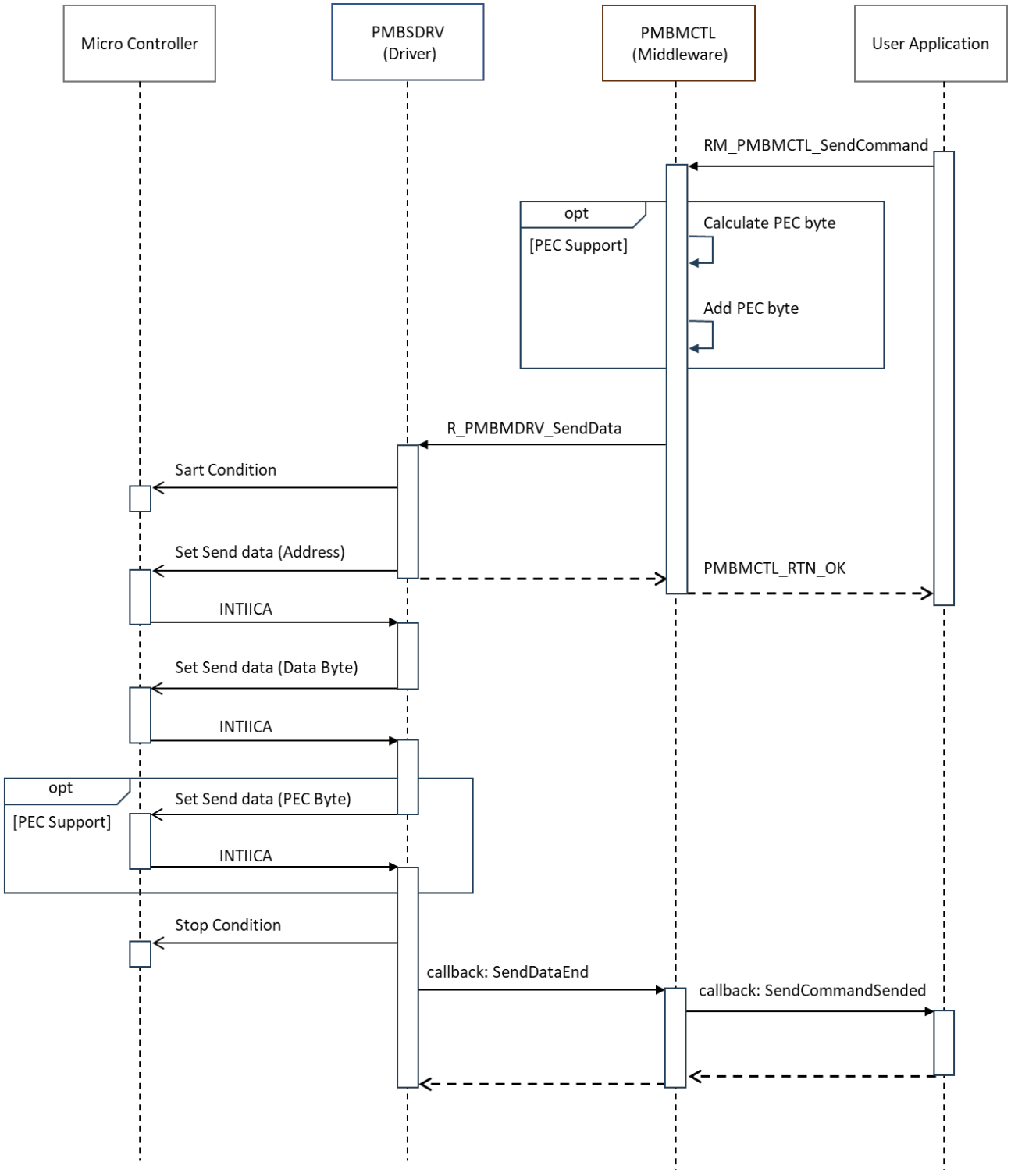
関数名	R_PMBMDRV_SendData	任意のデータを送信します。(マスタ送信)
引数	uint8_t dest	送信先アドレス
引数	uint8_t data_length	送信データ長
引数	uint8_t *p_data	送信データのポインタ
戻り値	-	-

5. 動作シーケンス

以下に各通信フォーマットのコマンドコード送信時動作およびユーザーへの通知動作について、また SMBus Alert Response Address への読み出し要求のシーケンス図を記載します。

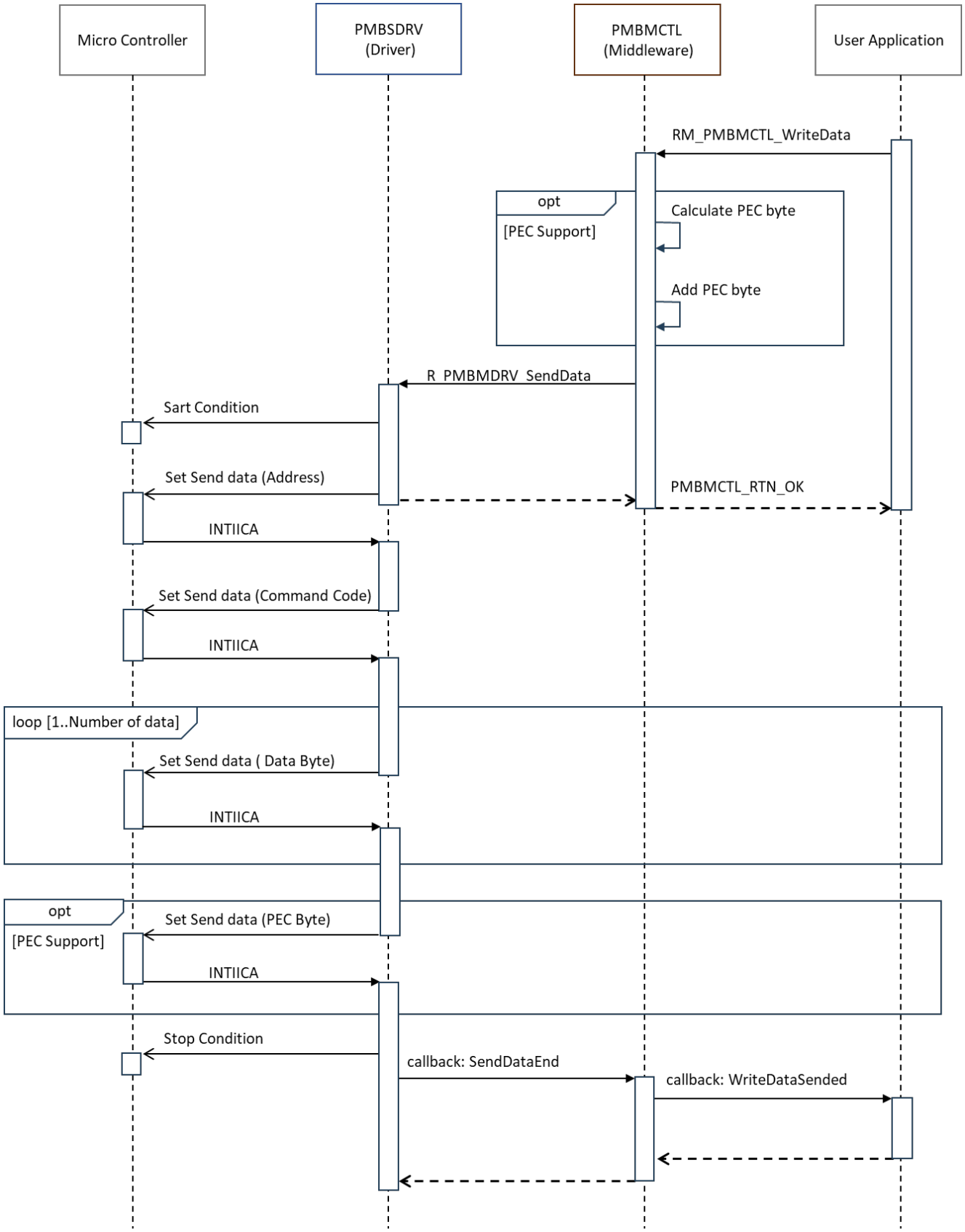
5.1.1 Send Byte

図 5-1 Send Byte sequence



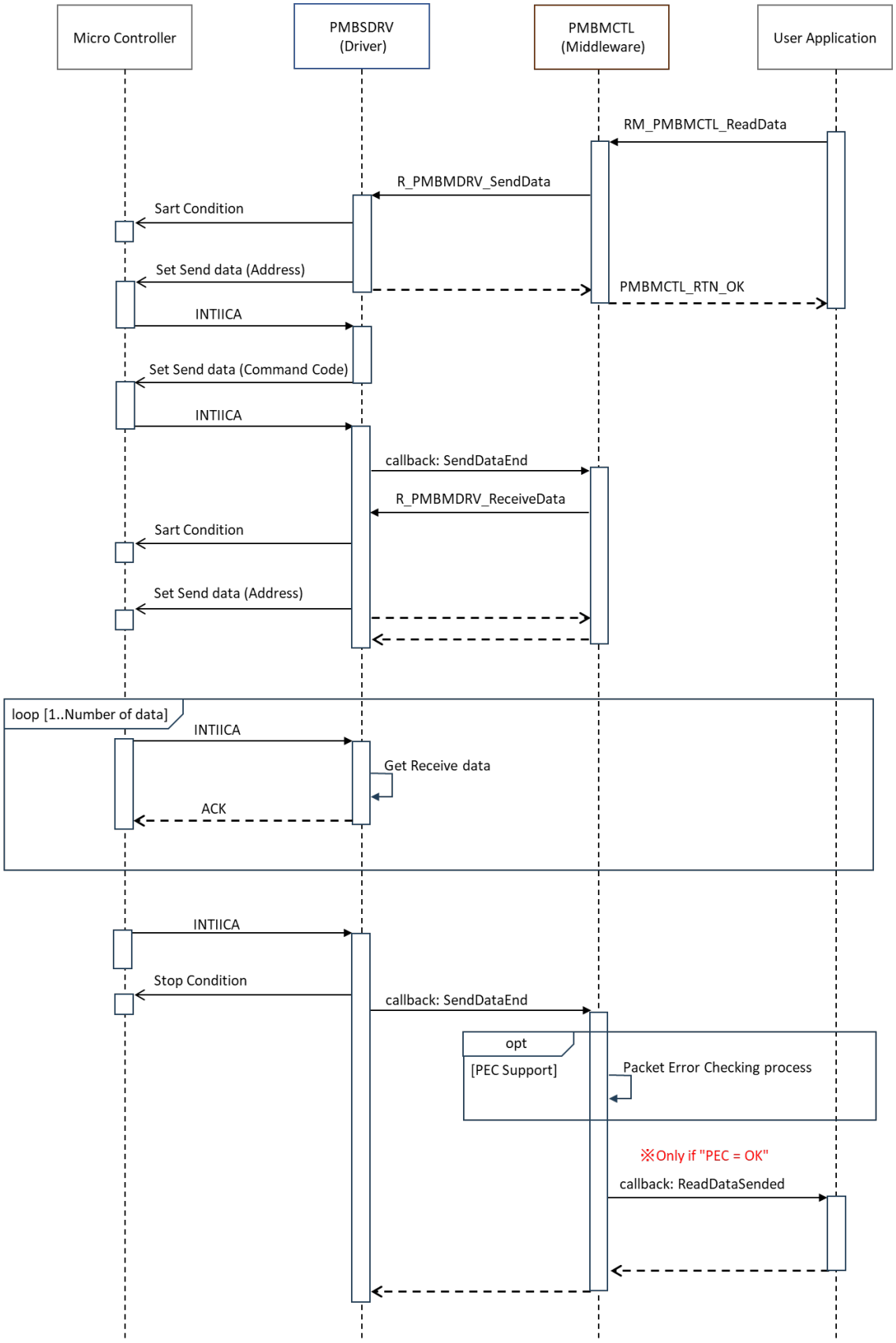
5.1.2 Write Byte/Word, Write 32 protocol, Write 64 protocol

図 5-2 Write sequence



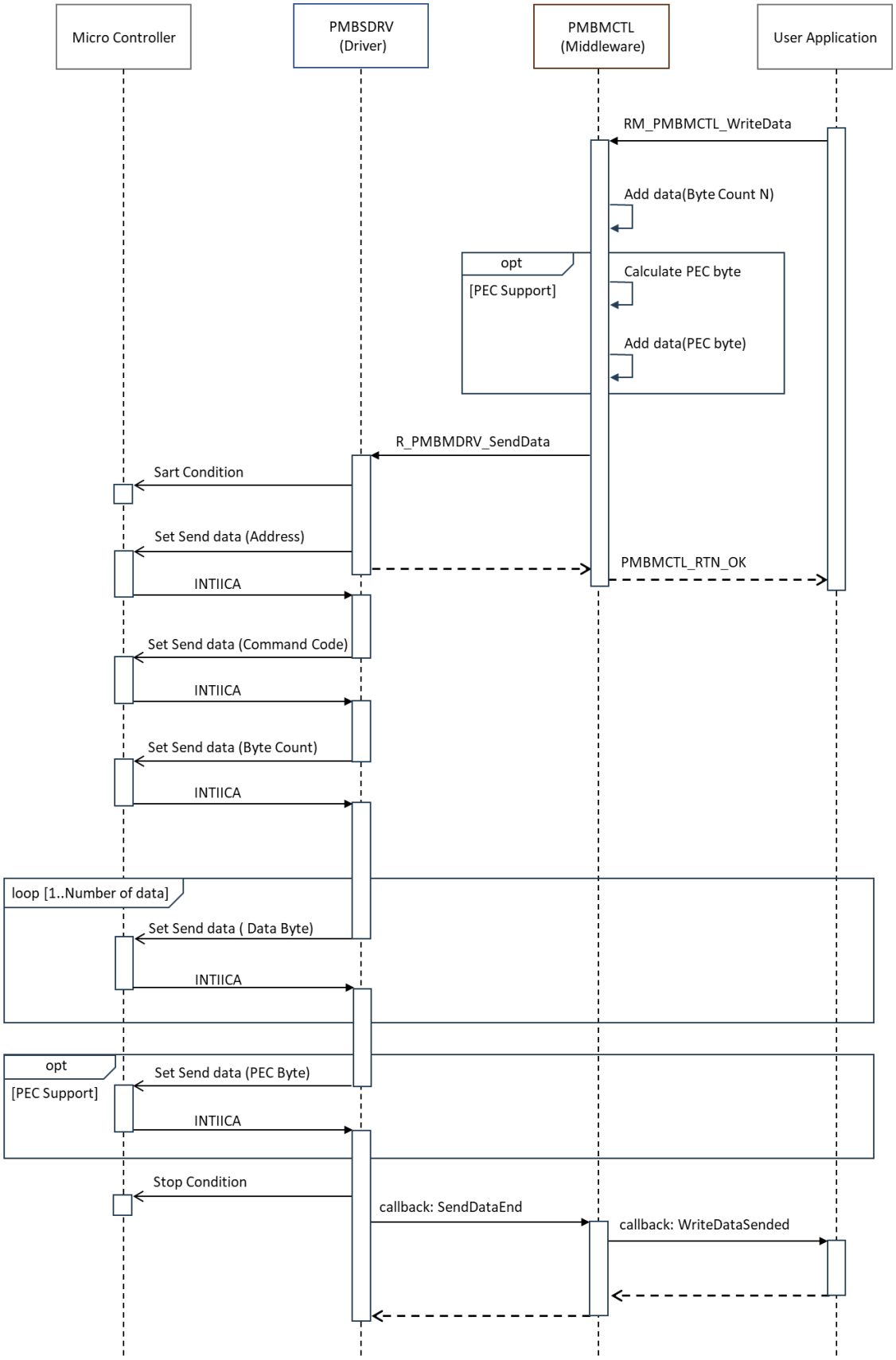
5.1.3 Read Byte/Word, Read 32 protocol, Read 64 protocol

図 5-3 Read sequence



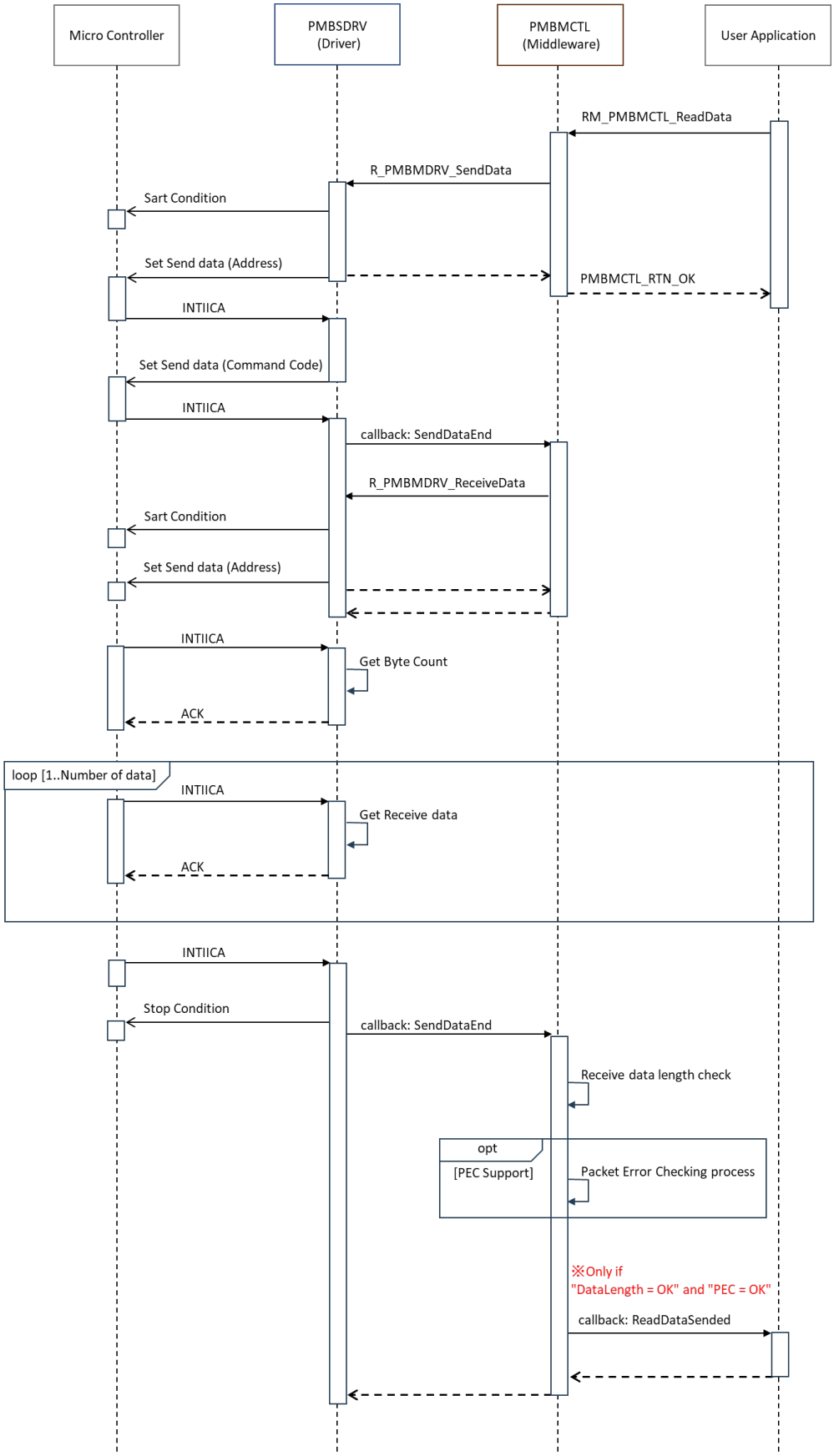
5.1.4 Block Write

図 5-4 Block Write sequence



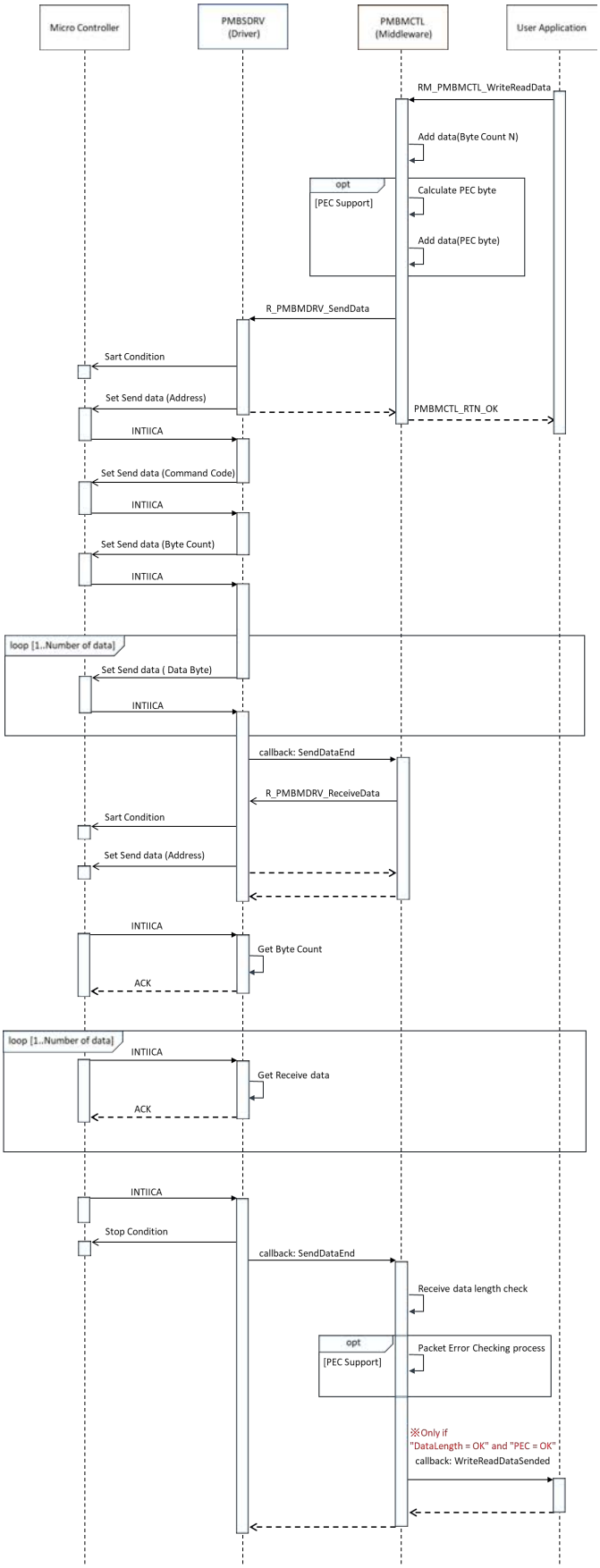
5.1.5 Block Read

図 5-5 Block Read sequence



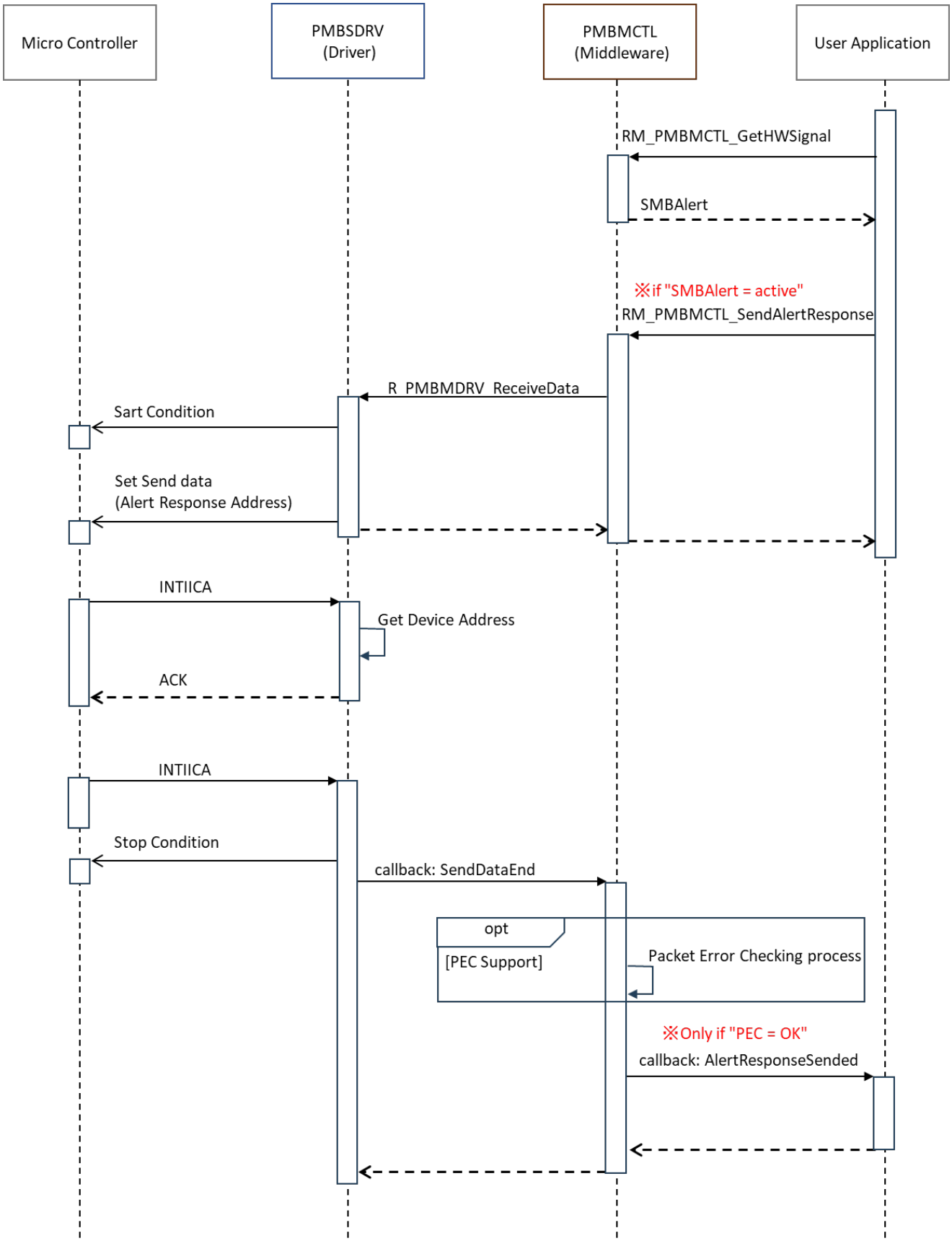
5.1.6 Block Write-Block Read Process Call

図 5-6 Block Write-Block Read Process sequence



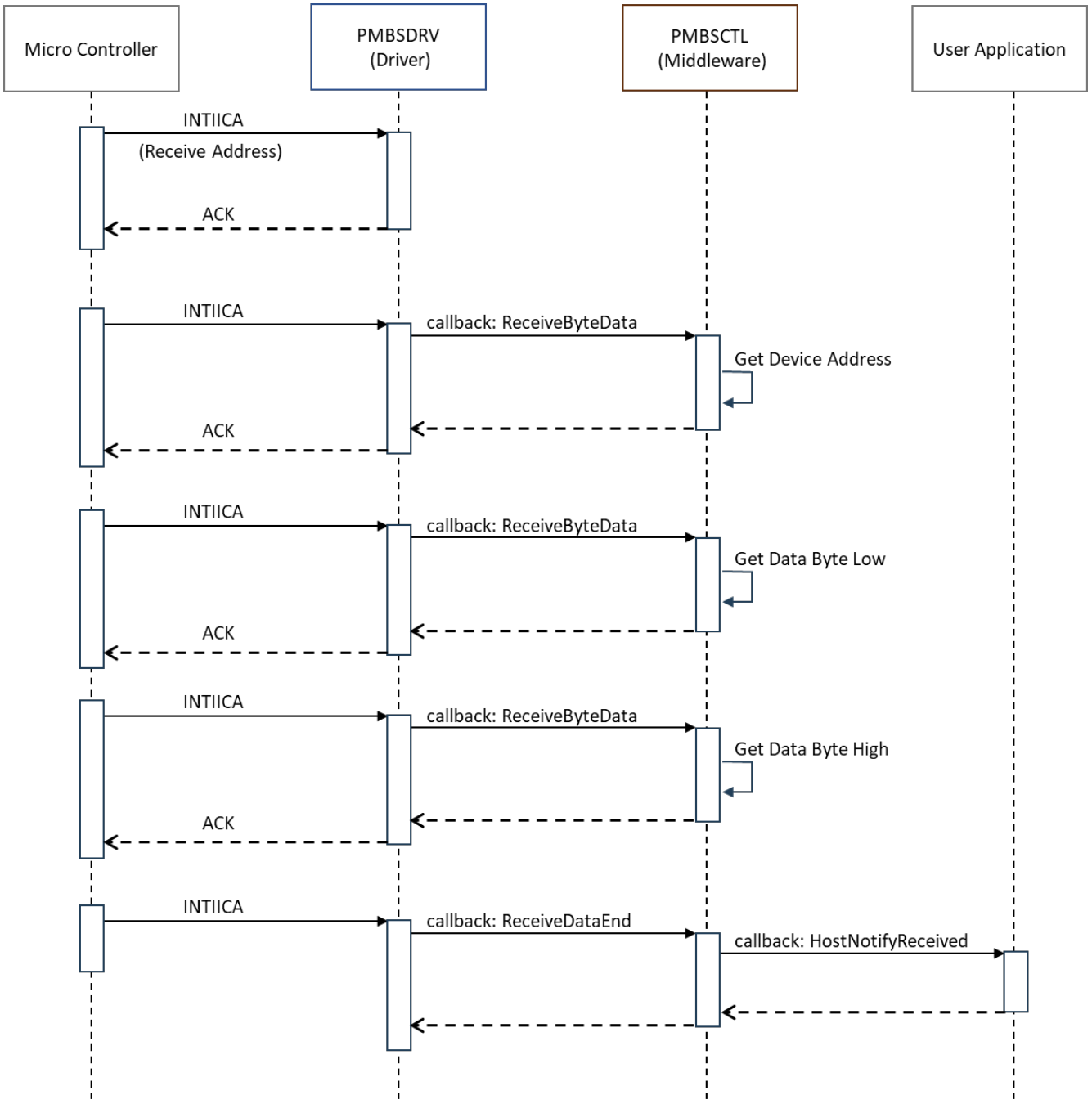
5.1.7 SMBALERT#

図 5-7 SMBALERT# sequence



5.1.8 SMBus Host Notify

図 5-8 SMBus Host Notify sequence



6. ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2024.4.18	-	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。