PIC32MM CPU ソフトウェア　説明書

令和　2年　3月

初版暫定
スパイス

重要 必ずお読みください

本ソフトウェアを使用するには「ソフトウェア使用許諾契約書」に同意する必要があります。
本ソフトウェアを PC の HDD や DVD ドライブ等にセットして読み出しを行った段階で、「ソフトウェア使用許諾契約書」に同意したと見なされます。

ソフトウェア使用許諾契約書

屋号スパイスこと松岡伸（以下、「当社」といいます。）は、お客様に、ダウンロードその他の手段により提供され、インストールされたソフトウェア（以下、「本ソフトウェア」といいます。）を使用する権利を下記の条件で許諾します。

第１条（著作権） 本ソフトウェアに関する著作権等の知的財産権は、当社に帰属し又は第三者から正当なライセンスを得たものであり、本ソフトウェアは、日本およびその他の国の著作権法ならびに関連する条約によって保護されています。

第２条（権利の許諾） お客様は、本契約の条項にしたがって本ソフトウェアを使用する非独占的な権利を本契約に基づき取得します。お客様は、お客様のＰＣに搭載されたＨＤＤその他の記憶装置に本ソフトウェアを導入し、使用することができます。
２ お客様は、本ソフトウェアをバックアップまたは保存の目的において複製することができます。

第３条（制限事項） １ お客様は、いかなる方法によっても、本ソフトウェアの改変、リバースエンジニアリング、逆コンパイルまたは逆アセンブルをすることはできません。ただし、適法と認められる場合はこの限りではありません。
２ お客様は、本契約書に明示的に許諾されている場合を除いて、本ソフトウェアを全部または一部であるかを問わず、使用、複製することはできません。
３ お客様には本ソフトウェアを使用許諾する権利はなく、またお客様は本ソフトウェアを第三者に販売、貸与またはリースすることはできません。
４ 本ソフトウェアをハードウェアとセットで購入した場合、本ソフトウェアは同時購入のハードウェアでのみ使用できます。他ハードウェアで使用することは出来ません。

第４条（限定保証） 本ソフトウェアは、一切の保証なく、現状で提供されるものであり、当社はその商品性、特定用途への適合性をはじめ、明示的にも黙示的にも本ソフトウェアに関して一切保証しません。本ソフトウェアに関して発生するいかなる問題も、お客様の責任および費用負担により解決されるものとします。

第５条（責任の制限） 当社は、本契約その他いかなる場合においても、結果的、付随的あるいは懲罰的損害について、一切責任を負いません。お客様は、本ソフトウェアの使用に関連して第三者からお客様になされた請求に関連する損害、損失あるいは責任より当社を免責し、保証するものとします。

第６条（契約期間）　１　本契約は、お客様が本ソフトウェアをダウンロードし、またはお客様のハードウェアにインストールされた日をもって発効し、次によって終了されない限り有効に存続するものとします。
２　お客様が本契約のいずれかの条項に違反したときは、当社は、お客様に対し何らの通知、催告を行うことなく直ちに本契約を終了させることができます。その場合、当社は、お客様の違反によって被った損害をお客様に請求することができます。なお、本契約が終了したときには、お客様は直ちにお客様のハードウェアに保存されている本ソフトウェアを破棄するものとします。

第７条（輸出管理）　お客様は、本ソフトウェアあるいはそれに含まれる情報・技術を日本ならびにその他の関係国が出荷等を禁止ないし制限している国に出荷、移転または輸出しないことに同意します。

第８条（その他）　本契約は日本国法を準拠法とします。本契約に関連または起因する紛争は、高知地方裁判所を第一審の専属的合意管轄裁判所としてこれを解決するものとします。

以上

目次

はじめに

このたびは PIC32MM CPU ボードをお買い上げいただきまして誠にありがとうございます。

本製品が皆様のお役にたてば幸いです。

　本文書は、PIC32MM CPU ボードのドライバソフトウェアについての説明書です。ドライバソフトウェアはそれ自身で完成したものではなく、一部の関数はユーザーによって記述する必要があります。この説明書では、これらの記述方法をとドライバの使い方を説明しています。

ご注意

１．本書の内容および製品の仕様は、改良のため将来予告無しに変更することがありますので、ご了承願います。

２．本書の内容については万全を期して作成しておりますが、万一お気づきの点がございましたらご連絡いただければ幸いです。

１．fatalError()関数

fatalError()関数はドライバを含むプログラムの各所でプログラムの正常な継続実行が出来ない状態の時に呼び出すことを想定した関数です。このため fatalError()を実行した後、プログラム動作が fatalError()の呼び出し元に返ることは想定していません。必ず、fatalError()関数内で処理を終了させるか無限ループとします。

注記：プログラムとしてドライバのみを使用する場合は、fatalError()を呼び出さない使い方も出来ます。この場合でもダミーの fatalError()を作成する必要があります。

fatalError()関数の実体はマクロで、その定義は以下のようになっています。

```
#define fatalError(msg)          ¥
        ::library::fatalErrorFunc(__FILE__, __func__, msg)
```

::library::fatalErrorFunc()はユーザー自身で記述する必要があります。fatalErrorFunc()の最も単純化した処理は CPU にリセットをかけることです。これで少なくとも CPU が無意味な処理を継続することは無くなります。反面、何処でどのような問題が発生したかは分からないため、プログラムの修正は困難です。

エラーの起きたプログラムの実行位置を識別するために 3 つの引数__FILE__, __func__, msg が用意されています。__FILE__, __func__はコンパイラがファイル名と関数名に置き換えてくれます。msg は同一の関数内に複数の fatalError()がある場合の識別用です。意味のある文字列でも良いし、単純に”1”, ”2”, ”3”といった識別文字番号でも十分です。

fatalErrorFunc()では次の点に注意してプログラムを作成してください。

- 割り込みは可能な限り早い段階で禁止にします。
- 装置として安全な状態となるように入出力を再設定します。
- 可能であればエラー情報を外部または記録装置に出力します。
- 関数から呼び出し元へ戻るのは禁止です。無限ループとするかリセット処理します。

また、通信ポートのみで接続されており CPU の IO 経由で制御が出来ない装置ユニットがある場合は注意が必要です。FatalError()関数内で停止指示を送信できるなら問題は少ないです。それでも通信エラーの可能性は考慮する必要があります。そうでなければ装置ユニット側に fatalError()が実行されたことを伝える術がありません。通信間隔を計測してタイムアウト処理を行うなど別途、安全対策を施してください。

２．SpiMaster ドライバ

SpiMaster ドライバに関連するヘッダファイルは以下です。ヘッダファイルには使用する上での使い方や注意点などをコメントとして記述してあります。本章はそれらを補完するものです。

- library/mcu_def.hpp

・ mcu/spi/spi_reg.hpp
・ mcu/spi_spi_master.hpp
・ mcu/spi_control_io_base.hpp
・ mcu/spi_device_select_base.hpp

２．１　SpiMaster ドライバの概要

SpiMaster ドライバは PIC32MM CPU ボードの SPI バスをアクセスするためのドライバです。SPI バスは複数の SPI デバイスにアクセスが可能でオプションとしてハンドシェイク用の IO ポートを追加できます。

PIC32MM CPU ボードの実装では、デバイスとして最大 8 個の SPI デバイスを接続できる用 3 ビットのデバイス選択信号とデバイスイネーブル信号(DevEn)を用意してあります。DevEn を有効とするとデバイス選択信号で選択された SPI デバイスの CS 信号が有効になります。ハンドシェイク用の IO ポートは入力と出力を各 2 ビット用意してあり、この IO ポートの使い方は各 SPI デバイスに依存します。上記以外には SPI デバイスへのリセット信号出力が含まれています。PIC32MM CPU ボードでは、この出力は CPU の IO ポートから出力してあり、異常時には SPI デバイスをリセットしたまま CPU 動作の継続が可能です。PIC32MM_CPU ボードでの信号線入出力図を下記に示します。

DEV_SEL	3本	OUT
nDEV_EN		OUT
RESET		OUT
DO	2本	OUT
DI	2本	IN
SCK		OUT
SDO		OUT
SDI		IN

２．２　SpiMaster ドライバの初期化と引数

SpiMaster ドライバに対する引数は C++言語のコンストラクタで与えます。実際の初期化の大半は init()関数内で行われます。引数としては、PIC32CPU の SPI 番号(1～3)、先に説明したデバイスの選択信号および DevEn 信号を操作する CspiDeviceSelectBase クラスへの参照およびオプションのハンドシェイク用の IO ポートを操作する CspiControlIoBase クラスへのポインタを渡します。

```
CSpiMaster(uint8_t ch, CSpiDeviceSelectBase& select,
             CSpiControlIoBase* io = nullptr);

bool init();
//SPI バスのリセット解除は SPI デバイスを読み書きするまでに行う。
```

init()関数は、以下のチェックを行ない問題が無ければ true を返します。

・ 引数チェックでエラーがない
・ 引数で渡されたクラスの init()関数を実行し、true が返された
・ このインスタンスが使用するメモリの確保が成功

init()関数が true を返した後、SPI バスへのアクセスが可能になります。
SPI バスのリセット解除は init()関数の前後のどこかで実施しておきます。

２．３　SPI デバイスへのアクセス手順

SPI デバイスへアクセスするには、最初にデバイスを選択します。

```
typedef struct {
	uint8_t		mode:2;		//spi mode
	uint8_t		brg;
} SpiDevDef_t;

void selectDevice(uint8_t no, const SpiDevDef_t& def);
```

no は選択する SPI デバイスの番号、SpiDevDef_t はそのデバイスの動作モード(0～3)と SPIxBRG に設定する分周比をまとめたものです。

次にフラッシュ ROM などの SPI IF をもつデバイスの読み書きでは専用の関数が用意されています。

```
struct SpiIoBuf_t {
	uint8_t*	buf;
	uint16_t	req_count;
};

void readDevice(SpiIoBuf_t& cmd, SpiIoBuf_t& data);
void writeDevice(SpiIoBuf_t& cmd, SpiIoBuf_t& data);
```

SPI デバイスの多くは SPI 経由でコマンドを受け取り、その結果を返します。
つまり、読み出しでは先に cmd の書き込み動作があり、次に結果を返すための読み出し動作が続きます。cmd はコマンドとそのオプションを設定します。data は読み出したデータを保管します。
書き込み動作では、書き込みコマンドとそのオプションを cmd に設定し、実際の書き込むデータは data に設定します。

少し特殊な動作として送信データと受信データが共に有効な送受信では下記を使用します。

```
void readWriteDevice(uint8_t* write, uint8_t* read, uint16_t count);
```

SPI_DAx2, SPI_AD などの SPI バスに接続する当社基板には、それらを読み書きするためのドライバが用意されています。

ご自身でドライバを作成する必要のある方は、上述のヘッダファイルを詳しく読んでください。各ヘッダファイルには使い方に関する情報をコメントとして記述してあります。それらを参考にしてください。

３．サポート

本製品に関する修理・サポートはメールにてご連絡ください。
メールアドレス　：　info@spice-elec.com

なお、本製品に関する最新の情報は HP に記載します。
http://www.spice-elec.com