SPI マスタ データシート SPIM V 1.1

リソース	API メモリ （バイト数）		ピン
	Flash	RAM	
CY7C639/638/633/602/601xx, CYRF69xx3			
MOSI/MISO/SCLK	38	0	3 - 4
SDIO/SCK	38	0	2 - 3

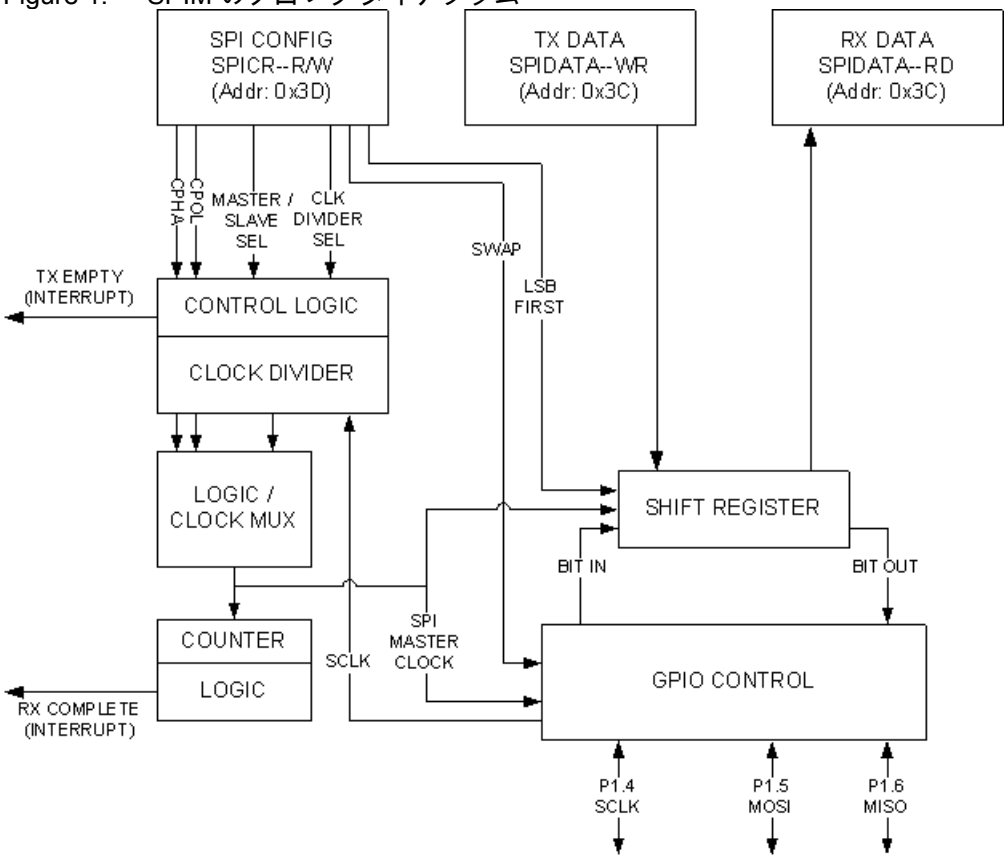
特性および概要

- シリアルペリフェラル相互接続（SPI）マスタ プロトコルをサポート
- SPI クロックモード 0、1、2 をサポート
- SPI 済み状態へのプログラム可能な割り込み
- SPI スレーブデバイスは個別に選択することができます。

SPIM ユーザ モジュールは、シリアルペリフェラル相互接続マスタです。全二重同期 8-bit データ転送を行います。SCLK フェーズ、SCLK 極性、LSB 優先を指定すると、殆どの SPI クロックモードに対応できます。ユーザ提供のソフトウェアによって制御されるスレーブ選択信号は、1 つまたは複数の SPI スレーブデバイスを制御するように構成できます。

SPI ユーザ モジュールは、SDIO/SCK を使用する光学マウス センサなどの 2 線式シリアルデバイスにも対応します。

Figure 1. SPIM のブロック ダイアグラム



パラメータおよびリソース

SPIPins

SPIPins ユーザ モジュールのパラメータは、SPI ブロックが使用するピンを選択します。このパラメータは、2 線式（SDIO/SCK）または 3 線式（MOSI/MISO/SCLK）の操作を可能にします。選択したデバイスが 5 ボルトあるいは 3.3 ボルトの動作に対応している場合、パラメータは電圧のオプションも提供します。

SPIPins パラメータは P14CR、P15CR、P16CR 構成レジスタをプログラミングします。

SPIPins の選択肢	電圧オプション	
	CY7C639xx CY7C638xx CY7C633xx	CY7C601xx CY7C602xx
MOSI(P1.5)/MISO(P1.6)/SCLK(P1.4)	5V または 3.3V	該当なし
SDIO(P1.5)/SCK(P1.4)	5V または 3.3V	該当なし
CY7C601xx および CY7C602xx デバイスは、SPI ピン用のデュアル電圧には対応しません。		

BitOrder

BitOrder ユーザ モジュールのパラメータは、SPI データストリーム用のビット順を選択します。選択肢は、LSBFirst または MSBFirst です。

BitOrder ユーザ モジュールのパラメータは SPI コンフィグレーション レジスタ（SPICR）の LSB 優先フィールドをプログラミングします。

CPOL

CPOL ユーザ モジュールのパラメータは、SPI クロック用のアイドル状態を選択します。

CPOL ユーザ モジュールのパラメータは SPI コンフィグレーション レジスタ（SPICR）の CPOL フィールドをプログラミングします。

CPHA

CPHA ユーザ モジュールのパラメータは、SPI データがサンプリングされる SPI クロックフェーズを選択します。

CPHA ユーザ モジュールのパラメータは SPI コンフィグレーション レジスタ（SPICR）の CPHA フィールドをプログラミングします。

ClockDivider

ClockDivider ユーザ モジュールのパラメータは、SPI クロックの ClockDivider を選択します。ClockDivider は CPUCLK に適用されて、SPI クロックを生成します。

ClockDivider	SCLK 選択	SCLK 周波数 （CPUCLK が以下の場合）	
		12 MHz	24 MHz
6	00	2 MHz	4 MHz
12	01	1 MHz	2 MHz
48	10	250 kHz	500 kHz
96	11	125 kHz	250 kHz

ClockDivider ユーザ モジュールのパラメータは SPI コンフィグレーション レジスタ（SPICR）の SCLK 選択フィールドをプログラミングします。

アプリケーション プログラミング インタフェース

アプリケーション プログラミング インタフェース (API) ルーチンは設計者がより高度なレベルでモジュールを処理できるようにユーザ モジュールの一部として提供されます。このセクションでは、各関数に対するインタフェースを include ファイルによって提供される関連定数とともに示します。

SPIM_Start

説明：

SPIM ユーザ モジュールを開始します。この関数は、API 整合性を保つためのものです。

C プロトタイプ：

```
void  SPIM_Start(void)
```

アセンブラ：

```
lcall  SPIM_Start
```

パラメータ：

なし

戻り値：

なし

副作用：

この関数によって、A および X レジスタが変更される場合があります。

SPIM_Stop

説明

SPIM モジュールを無効にします。この関数は、API 整合性を保つためのものです。

C プロトタイプ：

```
void  SPIM_Stop(void)
```

アセンブリ：

```
lcall  SPIM_Stop
```

パラメータ：

なし

戻り値：

なし

副作用：

この関数によって、A および X レジスタが変更される場合があります。

SPIM_EnableInt

説明

SPIM 送信バッファフルおよび受信完了割り込みを有効にします。

注：プロジェクト lib ディレクトリの SPIMINT.asm ファイルにある ISR に、カスタムの ISR コードを加えることができます。

C プロトタイプ：

```
void  SPIM_EnableInt(void)
```

アセンブリ：

```
lcall  SPIM_EnableInt
```

パラメータ：

なし

戻り値：

なし

副作用：

この関数によって、A および X レジスタが変更される場合があります。

SPIM_DisableInt

説明

SPIM 送信バッファフルおよび受信完了割り込みを無効にします。

C プロトタイプ：

```
void  SPIM_DisableInt(void)
```

アセンブリ：

```
lcall  SPIM_DisableInt
```

パラメータ：

なし

戻り値：

なし

副作用：

この関数によって、A および X レジスタが変更される場合があります。

SPIM_SetMOSI

説明

マスターアウト、スレーブインの出力ピンを設定します。

C プロトタイプ：

```
void  SPIM_SetMOSI(BYTE bPin)
```

アセンブリ：

```
mov   A, bPin
```

```
lcall  SPIM_SetMOSI
```

パラメータ：

BYTE bPin: 0x00--SPIM_MOSI_P15, 0x01--SPIM_MOSI_P16. アキュムレーターを通過します。

戻り値：

なし

副作用：

この関数によって、A および X レジスタが変更される場合があります。

SPIM_SetMISO

説明

マスターイン、スレーブアウトの出力ピンを設定します。

C プロトタイプ：

```
void  SPIM_SetMISO(BYTE bPin)
```

アセンブリ：

```
mov   A, bPin
lcall  SPIM_SetMISO
```

パラメータ：

BYTE bPin: 0x00--SPIM_MISO_P16, 0x01--SPIM_MISO_P15. 累算器を通過します。

戻り値：

なし

副作用：

この関数によって、A および X レジスタが変更される場合があります。

SPIM_bIO

説明

単一データバイトを送信し、スレーブデバイスからの受信データを返します。

C プロトタイプ：

```
BYTE  SPIM_bIO(BYTE bTXData)
```

アセンブリ：

```
mov   A, bRxData
lcall   SPIM_bIO
```

パラメータ：

bTxData - 送信するデータ

戻り値：

スレーブ SPI から受信したデータが累算器を通して戻されます。

副作用：

この関数が呼び出されると、ステータスビットはクリアされます。この関数によって、A および X レジスタが変更される場合があります。

ファームウェア ソースコードの例

C 言語では、開始 API を使用して動作を開始し、終了時には停止 API を呼び出します。

```
#include "SPIM.h"

SPIM_Start();
...                             // (application processing)
SPIM_Stop();
```

次の C コードの部分には、MOSI/MISO/SCLK 選択用の SPI EEPROM からの読み値が示されています。

```
SPIM_Start();
SPIM_SetMOSI(SPIM_MOSI_P16);  // Set MOSI to P1.6/MISO P1.5
EEPROM_Reset();               // Reset the EEPROM (user provided function)
SPI_SELECT();                 // Assert slave select (user provided function)
SPIM_bIO(EEP_READ);           // Send a read command
SPIM_bIO(0x00);               // Address 0:0
SPIM_bIO(0x00);

for (i = 0; i < 32; ++i)
{
   aRD[i] = SPIM_bIO(0);      // Read some data
}
SPI_UNSELECT();               // Deassert the slave select (user provided function)
```

次の C コードの部分には、SDIO/SCK 選択を使用して SDIO 光センサからの読み取りレジスタ 0x01 が示されています。

```
SPIM_Start();
SPIM_SetMOSI(SPIM_MOSI_P15);  // Set MOSI to P1.5
SPIM_bIO((0x01);              // Write the register number
timer_delay_100_usec();       // delay (user provided function)
SPIM_SetMISO(SPIM_MIS0_P15);  // Reverse the SDIO line
data = SPIM_bIO(0);           // Read the register data
```