

Renesas Synergy™ SPI マスタ機能の基本的な使い方

R30AN0303JJ0100 Rev.1.00 2017.06.15

要旨

本アプリケーションノートでは、Renesas SynergyTM Software Package(以下、SSP)の RSPI および SCI_SPI の SPI マスタ機能の基本的な使い方について説明します。Synergy MCU に搭載されている SPI 機能と SSP インタフェースを対応させて具体的な使い方の例を説明しています。Synergy Configuration の詳細な設定手順とコーディング例は、appendix に説明しています。

対象デバイス

• S124, S3A7, S5D9, S7G2

動作確認に使用した環境

- Synergy SK-S7G2 Starter Kit Board(v3.0)
- e2studio (Version5.3.1.002) and SSP 1.2.0



目次

1. はじめに	.3
1.1 概要	. 3
1.2 参考資料	. 3
2. ハードウェア構成	.3
2.1 Synergy MCU の SPI 機能	. 3
2.2 SPI 基本接続	. 3
3. ソフトウェア構成	.4
3.1 SSP モジュール	. 4
3.1.1 SPI 転送モード	.7
3.1.2 データ転送方式	. 9
3.1.3 チップセレクト	11
3.1.4 コールバック関数	12
3.1.5 スレーブモード	12
3.2 SPI フレームワークの使い方	13
3.2.1 SPI 転送モード(SPI フレームワーク)	14
3.2.2 共有バスの使い方	15
4. Appendix	17
4.1 SPI HAL ドライバの設定手順	17
4.2 SPI フレームワークの設定手順	27
4.3 SPI フレームワーク 共有バスの設定手順	30



1. はじめに

1.1 概要

Synergy MCU 用にアプリケーションプログラムを作成する為に、Renesas Synergy TM Software Package(以下、 SSP)を利用することができます。本アプリケーションノートでは、Synergy MCU に搭載されている SPI 機能 に対応させて SSP インタフェースでどのように使用しているかを説明します。SPI 機能には、SPI 転送モード とデータ転送方式などがあり、各機能の具体的な使用例(実行波形)を示します。SPI インタフェースを用いて SPI 関連のプログラムを効率的に開発することができます。

1.2 参考資料

- [1] S7G2 User's Manual: Microcontrollers (R01UM0001EU0120 Rev.1.20)
- [2] Renesas Synergy TM Software Package v1.2.0 User's Manual (R01US0171EU0100 Rev. 01.00)
- [3] Renesas Synergy TM Software Package(SSP) v1.2.0 Release Note (R11UT0004EU0110 Rev.1.10)

2. ハードウェア構成

Synergy MCU に内蔵されている SPI 機能と SPI の基本的な接続方法について説明します。

2.1 Synergy MCU の SPI 機能

Synergy MCU では、以下の SPI 機能が内蔵されています。

・SPI:シリアルペリフェラルインタフェース

Synergy MCU の専用 SPI 機能は、「SPI シリアルペリフェラルインタフェース」です。

・SCI_SPI:シリアルコミュニケーションインタフェースの簡易 SPI

シリアルコミュニケーションインタフェースのシリアル通信方式の1つに簡易 SPI 通信方式があります。

2.2 SPI 基本接続

図1に SPIの基本的な接続例を示します。



図 1 基本接続例(RSPI)

図1に示したように、クロック、入出力データ、チップセレクト(スレーブセレクト)端子を接続します。各 端子とその用途を表1に示します。端子名は、SPIとSCI_SPIで異なります。スレーブセレクト端子名も制 御する端子によって名称が変わります。

SPI 端子名	SCI_SPI 端子名	用途
RSPCK	SCK	クロック
MOSI	TXD_MOSI	マスタ出力スレーブ入力転送データ
MISO	RXD_MISO	マスタ入力スレーブ出力転送データ
SSL/GPIO	CTS_RTS_SS/GPIO	チップセレクト(スレーブセレクト制御信号)

表1 使用する端子とその用途



Renesas Synergy™

3. ソフトウェア構成

3.1 SSP モジュール

図 2 に SPIの SSP モジュールのブロック図を示します。

SSP モジュールには、SPIの SSP インタフェースが以下のように2つ提供されています。

・SPI HAL インタフェース

・SPI フレームワークインタフェース

共通化された SSP インタフェースを利用して、Synergy MCU の周辺機能を使用することができます。



図 2 SPI SSP モジュールのブロック図

SPI HAL インタフェースは、SPI HAL ドライバへの共通インタフェースとなり、RSPI または SCI_SPI のド ライバを呼び出すことができます。さらにデータ転送方式の選択ができます。

SPI フレームワークは、SPI HAL ドライバへ共通のインタフェースを介して、RSPI または SCI_SPI の HAL ドライバを呼び出すことができます。

SSPの RSPI および SCI_SPI の HAL ドライバは、Synergy MCU の SPI ハードウェア資源を制御する処理を 行います。SSP が使用する機能と Synergy MCU 機能との対応関係を表 2 と表 3 に示します。

Clock Phase と Clock Polarityの機能の組み合わせで、4 つの SPI 転送モードを選択することができます。

Data Length では、転送ビット長を選択できます。RSPI と SCI_SPI では、転送ビット長に違いがあります。 詳細は、「3.2.2 データ転送方式」を参照してください。Bit Order では、MSB/LSB ファーストのビット列の 選択ができます。

Operating Mode では、マスタモードとスレーブモードの選択ができます。ただし、SCI_SPIの HAL ドライ バはマスタモードのみ対応しています。本アプリケーションノートではマスタモードについて説明していま す。

Bit rate では、転送ビット数を bps 単位で指定できます。SPI HAL ドライバは、Bit rate の値を参照して対応 する Synergy MCU の Bit rate register に計算式から導かれる値を設定しています。



項目	SSP の提供機能	Synergy MCU(RSPI)機能
Clock Phase	[Synergy Configuration]	SPCMD.CPHA_b0
	Data sampling on odd edge, data variation on	0:立ち上がりエッジでデータサンプリング、
	even edge	立ち下がりエッジでデータ変化を選択
	Data sampling on even edge, data variation on	1: 立ち上がりエッジでデータ変化、立ち下が
	odd edge	りエッジでデータサンプリングを選択
Clock Polarity	[Synergy Configuration]	SPCMD.CPOL_b1
	Low when Idle	0 : アイドル時の RSPCK を Low に設定
	High when Idle	1 : アイドル時の RSPCK を High に設定
Data Length	[read/write API で指定]	SPCMD.SPB[3:0]_b11-b8
	8、16、32 ビットの指定が可能です。	b11 b8
	以下のマクロ定義が提供されています。	0100~0111:8 ビット
	SPI BIT WIDTH 8 BITS	1000:9ビット
	SPI_BIT_WIDTH_16_BITS	1001:10ビット
	SPI_BIT_WIDTH_32_BITS	1010:11ビット
	[Synergy Configuration]	1011:12ビット
	データ転送方式は、CPU 転送と DTC 転送の選	1100:13 ビット
	択ができます。	1101:14 ビット
	但し、DTC 転送の場合は、32 ビットのみ使用	1110:15 ビット
	可能です。	1111:16 ビット
		0000:20ビット
		0001:24ビット
		0010、0011:32 ビット
Bit Order	[Synergy Configuration]	SPCMD.LSBFF b12
	MSB First	 0:MSB ファースト
	LSB First	1 : LSB ファースト
Operating Mode	[Synergy Configuration]	SPCR.MSTR_b3
	Slave	0:スレーブモードを選択
	Master	1:マスタモードを選択
	本資料では Master の説明のみを行います。	
Bit rate	[Synergy Configuration]	SPBR
	Bit rate を bps 単位で指定します。	Bit rate の計算式で決まる値
Select SSL	[Synergy Configuration]	SPCMD.SSLA[2:0]_b6-4
(Slave Select)	SSL 信号アサート設定の選択ができます。	b6 b4
	SSL0	000:SSL0
	SSL1	001:SSL1
	SSL2	010:SSL2
	SSL3	0 1 1 : SSL3
		1 x x:設定禁止, x:Don't care
Slave Select	[Synergy Configuration]	SSLP.SSLnP_b3-0
Polarity	Active Low	0 : active-low
	Active High	1 : active-high

表 2 SSP と Synergy MCU(RSPI)との対応

SPI Control Register(SPCR)

SPI Command Register(SPCMD)

SPI Bit Rate Register(SPBR)

Bit rate の計算式は、参考資料[1]を参照してください。

SPI Slave Select Polarity Register(SSLP)



百日	SSPの提供機能	Syperay MCU(SCL SPI)機能
CIUCK FIIdSE	Data sampling on odd odgo, data variation on	
	even edge	
	Data sampling on even edge, data variation on	「「シロック連延のリ
	odd edge	
Clock Polarity	[Synergy Configuration]	SPMR.CKPOL_b6
	Low when Idle	0:クロック極性反転なし
	High when Idle	1:クロック極性反転あり
Data Length	[read/write API で指定]	SMR.CHR_b6
	8 ビットの指定が可能です。	0:8ビットデータで送受信(初期値)
	SPI_BIT_WIDTH_8_BITS が定義されていま	SCMR.CHR1_b4
	す。	0:8 ビットデータで送受信(初期値)
	[Synergy Configuration]	
	データ転送方式はCPU転送とDTC転送の選択	
	ができます。	
Bit Order	[Synergy Configuration]	SCMR.SDIR_b3
	LSB First	0:LSB ファースト
	MSB First	1 : MSB ファースト
Operating Mode	[Synergy Configuration]	SPMR.MSS_b2
	Master のみ使用可能です。	0:マスタモードを選択
		1:スレーブモードを選択
Bit rate	[Synergy Configuration]	BRR
	Bit rate を bps 単位で指定します。	Bit rate の計算式で決まる値

表 3 SSP と Synergy MCU(SCI_SPI)との対応

Serial Mode Register(SMR) Smart Card Mode Register(SCMR) Bit Rate Register(BRR) SPI Mode Register(SPMR) Bit rate の計算式は、参考資料[1]を参照してください。

RSPIと SCI_SPI 機能の違いを以下の表にまとめました。

表	4	RSPI	と	SCI	SPI	機能の違い
---	---	------	---	-----	-----	-------

項目	RSPI	SCI_SPI
Data Length	8、16、32 ビットの指定が可能です。	8 ビットの指定が可能です。
	DTC 転送では、32 ビットのみ対応。	
Operating Mode	Master	Master
	Slave (本資料では説明していません。)	
Select SSL	SSL0-3 の選択が可能です。	なし
Slave Select Polarity	極性の指定が可能です。	

チップセレクト(スレーブセレクト)の詳細は、3.1.3を参照してください。



SPI HAL ドライバの使い方

最初に SPI HAL ドライバの使い方を以下の項目に従って具体的に説明します。

・SPI 転送モード

・データ転送方式

SPI 転送モードは、Clock Phase と Clock Polarity の組み合わせで 4 つのモードを選択できます。

データ転送方式は、CPU転送とDTCの機能を使用したDTC転送が選択できます。さらに、転送ビット長や転送バイト数をread/write APIの引数で指定できますので、いろいろな転送方法が実現できます。

また、SSP HAL インタフェースを使用する際に必要となる SPI HAL ドライバを制御する機能について説明 します。

・チップセレクト(スレーブセレクト)

・コールバック関数

3.1.1 SPI 転送モード

表 2 と表 3 に示したようにサンプリングのタイミングは、Clock phase(CPHA)と Clock polarity(CPOL)のパラメータで定義されます。

Clock Phase パラメータは、サンプリング位相をシフトすることができます。

Clock Phase=0 の時、データは odd(leading) clock edge でサンプリングされます。

Clock Phase=1 の時、データは even(trailing) clock edge でサンプリングされます。

Clock Polarity パラメータは、クロックの極性を設定できます。Idle 時の Clock の極性を「Low when idle」と「High when idle」から選択できます。

表 5のように Clock Phase と Clock Polarity のパラメータを組み合わせると、4 つの SPI 転送モードに分類されます。この 4 つのモードからデータのサンプリングタイミングを選択できます。

モード	Clock Polarity	Clock Phase
モード 0	Low when idle	Data sampling on odd edge, data variation on even edge
モード1	Low when idle	Data sampling on even edge, data variation on odd edge
モード2	High when idle	Data sampling on odd edge, data variation on even edge
モード 3	High when idle	Data sampling on even edge, data variation on odd edge

表 5 SPI 転送モード

4 つの SPI 転送モードで動作させた波形を以下に制御コードと共に示します。各図のクロックの矢印のタイ ミングでデータを取得しています。



同じ制御コードで各モードのパラメータを変更することでサンプリングタイミングを変更できます。 チップセレクトとコールバック関数については、後で説明します。



図 3-1 モード0の波形



図 3-2 モード1の波形



図 3-3 モード2の波形



図 3-4 モード3の波形

3.1.2 データ転送方式

データ転送方式には、CPU 転送と DTC 転送があります。Synergy MCU のデータトランスファコントロー ラモジュール(DTC)と接続することで、SPI HAL ドライバは DTC 機能を使用できるようになります。CPU へ 負荷を与えることなく DTC を経由して SPI データ転送を行うことができます。DTC 転送を使用した場合、 DTC の制限により 1 回あたりの転送サイズの上限値が 64Kbytes になります。

DTC 転送は RPSI と SCI_SPI ドライバの両方で使用できます。ただし、本 HAL ドライバでは表 6 に示すように転送可能なビット長に違いがあります。

ドライバ	DTC 転送	CPU 転送
RSPI	32 ビット	8、16、32 ビット
SCI_SPI	8ビット	8ビット

表 6 データ転送方式毎の転送ビット長

RSPIの HAL ドライバの DTC 転送では、32 ビットの転送ビット長のみ対応しています。CPU 転送では、8、 16、32 ビットの転送ビット長に対応できます。

SCI_SPIの HAL ドライバでは、8 ビット長のデータのみ対応しています。

次に、RSPIのHALドライバを使用していろいろな転送方式を実現してみます。writeRead API制御コードを使用して転送ビット長と転送数を変えた場合の波形を以下に示します。writeRead APIは、RSPIおよびSCI_SPIで共通のインタフェースになっています。



図 4-1 8ビット長、2データ転送(CPU)

このように8ビットのデータを複数バイト連続して転送することができます。



図 4-2 16 ビット長、1 データ転送(CPU)



図 4-3 32 ビット長、1 データ転送(CPU)



```
32 ビット長(DTC)、1 データ転送する場合

コードは、32 ビット長(CPU)と同じです。Synergy ConfigurationでDTC使用の設定を行います。

#define SSL_PIN IOPORT_PORT_01_PIN_03

#define TX_SIZE 1

volatile bool g_spi_done = false;

ssp_err_t err;

uint32_t wdata32[2] = {0x01234567, 0x89ABCD};

uint32_t rdata32[2];

err_code = g_spi0.p_api->open(g_spi0.p_ctrl, (spi_cfg_t *)g_spi0.p_cfg);

g_ioport_on_ioport.pinWrite(SSL_PIN, IOPORT_LEVEL_LOW);

err = g_spi0.p_api->writeRead(g_spi0.p_ctrl, &wdata32[0], &rdata32[0], TX_SIZE,

SPI_BIT_WIDTH_32_BITS);

while (g_spi_done == false) { } /* コールバック関数からの転送完了を待つ */

g_ioport_on_ioport.pinWrite(SSL_PIN, IOPORT_LEVEL_HIGH);

err = g_spi0.p_api->close(g_spi0.p_ctrl);
```



図 4-4 32 ビット長、1 データ転送(DTC)

3.1.3 チップセレクト

チップセレクトは、SPI通信先のスレーブを選択する時に使用します。転送開始時にアサートされ、転送終 了時にアサート解除されます。

RSPIは、SPIのハードウェア機能にある SSL 端子を使用する方法と I/O PORT(GPIO)端子を使用する方法から選択できます。I/O PORT 端子を使用する場合は、ユーザのアプリケーションで制御を行います。

SCI_SPIは、マスタ動作用のチップセレクト端子は存在していません。制御が必要な場合は、ユーザのアプリケーションで I/O PORT 端子を使用して制御することになります。

RSPI の端子名は、使用する端子によって I/O PORT(GPIO)または SSL になります。SCI_SPI の端子名は、I/O PORT(GPIO)になります。

RSPIの SSL 端子を使用した構成例について説明します。図 5 に SSL 端子をチップセレクトとして使用した接続例を示します。この場合は、Synergy MCUの RSPI がアサートとアサート解除処理を行いますので、 ユーザのアプリケーションコードでチップセレクトの制御コードを記述する必要はありません。



図 5 SSL 端子をチップセレクトに使用した接続例



この構成で実行した制御コードと波形を以下に示します。チップセレクトの制御コードは不要になります。

チップセレクトの制御が I/O PORT 端子で制御した場合に比べて、短くなっているのが分かります。(図 3-1 と比較)



図 6 SSL 端子をチップセレクトに使用した時の波形

3.1.4 コールバック関数

SPI HAL ドライバは、コールバック関数のインタフェースをユーザに提供しています。ユーザはコールバック関数を使用するかしないかの選択ができます。

SPI HAL ドライバの ISR(Interrupt Service Routine)は、転送完了、転送中断、エラー検出などのイベント情報をセットして、コールバック関数が定義されていれば、コールバック関数を呼び出します。

コールバック関数は、ユーザのアプリケーションプログラムの中で定義します。コールバック関数を使用 することで、ISR(Interrupt Service Routine)からのイベント情報を受け取り、ユーザはそれに対応する処理を行 うことができます。

例えば、転送完了を確認したい場合、コールバック関数内で転送完了の事象をグローバル変数のソフトウェ アフラグ、semaphore あるいは event などで通知する処理を記述します。ユーザのアプリケーションでは、対応する転送完了の事象を待つ処理を記述します。制御コード例において、転送完了を待つ処理は、グローバル変数を使用したコードで実現しています。

コールバック関数の具体的な実装例は、appendix を参照してください。

3.1.5 スレーブモード

RSPIでは、表 2 で示した、Operating Mode を Slave にすることでスレーブモードの設定が可能です。本ア プリケーションノートでは、スレーブ機能およびその使い方の詳細については説明していません。



3.2 SPI フレームワークの使い方

SPI フレームワークは SPI 共有バス上で複数のスレーブデバイスを排他的に制御することができます。

共有バスは、Synergy MCU の SPI ハードウェア資源を1つ制御して、複数のスレーブデバイスにアクセス する場合に使用します。SPI フレームワークは、複数の RSPI または SCI_SPI の HAL ドライバを使用して、 それぞれのスレーブデバイスにアクセスします。この時、共有バスの機能を使用することで、それぞれのス レーブデバイスに割り当てられた複数の SPI HAL ドライバを排他的に制御することができます。

共有バスのような機能がない場合、ユーザは使用する SPI HAL ドライバを切り替えながら対応するスレー ブデバイスにアクセスする処理をアプリケーションプログラムに記述する必要があります。共有バスがある ことで、共通の SPI フレームワークインタフェースで切り替えプログラムを記述することなく、複数のスレー プデバイスにアクセスすることができます。同時に、チップセレクト端子のアサートとアサート解除処理を 使用するスレーブデバイスに対して行います。

SPI HAL ドライバと SPI フレームワークの使い方の違いを表 7 にまとめました。

コールバック関数とチップセレクト処理は、SPIフレームワークの中に組み込まれています。その為、ユー ザが制御コードを記述する必要はありません。

制御	SPI HAL ドライバ	SPI フレームワーク
共有バス	なし	SPI フレームワークは共有バスという構成を 取っています。SPI フレームワークが複数のス レープデバイスを使用する場合、この共有バス上 で SPI HAL ドライバを管理することができま す。使用しないドライバの close 処理と使用する ドライバの open 処理を行います。これにより、 各ドライバの read/write 処理を排他的に制御す ることができます。 [Synergy Configuration] 共有バスプロパティに使用する Synergy MCUの ハードウェア資源の情報を設定します。 [ユーザアプリケーション] SPI HAL ドライバを切り替える処理を記述する 必要がありません。
コールバッ ク関数	[Synergy Configuration] コールバック関数の使用の有無を設定 できます。使用する場合は、関数シンボ ルを設定します。 [ユーザアプリケーション] コールバック関数を使用する場合は、設 定した関数を定義し、転送完了などを待 つ処理を記述することができます。	[Synergy Configuration] ユーザはコールバック関数を使用できません。 [ユーザアプリケーション] SPI フレームワークの内部関数として、定義さ れていますので、ユーザは使用できません。
チップセレ クト(CS 制 御)	[Synergy Configuration] I/O Port(GPIO)と初期値を設定します。 [ユーザアプリケーション] ユーザが I/O Port の制御コードを記述す る必要があります。	[Synergy Configuration] I/O Port(GPIO)と初期値を設定します。 さらに、SPI フレームワークで I/O Ports 情報と active レベルを設定します。 [ユーザアプリケーション] SPI フレームワークの内部処理で行いますの で、ユーザが制御コードを記述する必要はありま せん。

表 7 SPI HAL ドライバと SPI フレームワークの対応表

次に SPI フレームワークの動作例を SPI 転送モードと共有バスの構成例を用いて説明します。



3.2.1 SPI 転送モード(SPI フレームワーク)

SPI フレームワークを使用した場合の SPI 転送モード 0 の動作波形を示します。SPI HAL ドライバと波形は ほぼ同じですが、チップセレクトのタイミングが若干異なっています。ユーザのアプリケーションが制御す る場合と SPI フレームワーク内で制御している違いが分かります。

制御コード例を以下に示します。SPI HAL ドライバの例から以下の関連するコードが不要になっているのが 分かります。

#define SSL_PIN IOPORT_PORT_01_PIN_03

volatile bool g_spi_done = false;



図 7 モード0の波形



3.2.2 共有バスの使い方

複数のスレーブ(スレーブデバイス)を制御する場合、共有バスを使用して制御します。図 9 に SPI マスタ をスレーブ1とスレーブ2 に接続する場合の接続例を示します。



図 8 複数のスレーブを制御する接続例

次に、Synergy Configuration の階層図を図 9 に示します。制御するスレーブ毎に対応する SPI フレームワークを作成します。最上位層が SPI フレームワークになります。それぞれの SPI フレームワークに SPI HAL ドライバと共有バス(青枠)が接続されています。

SSP インタフェースでは、スレーブ1と2に対応する SPI フレームワーク1と2を作成します。

チップセレクトの情報はこの枠のプロパティで設定します。

この情報を使用して、SPIフレームワークの read/write API がアクセスするスレーブに対してチップセレクト信号の制御を行います。

SPI フレームワーク 1 と 2 の共有バスのプロパティにおいて、使用する Synergy MCU の SPI ハードウェ ア資源の情報を設定します。

SPI フレームワークの read/write API は、共有バス上で使用している SPI HAL ドライバを確認して、必要な SPI HAL ドライバへの切り替えを排他的に行います。その後、のチップセレクト制御と共にスレーブへの read/write 処理が行われます。

プロパティ設定手順の詳細は appendix を参照してください。

g_sf_spi_device0 SPI Fi	ramework Device on sf_sp	ai -	g_sf_spi_device1 SPI Fr	amework Device on sf_sp	ท้
g_spi0 SPI Driver on r_	l rspi	g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi	g_spi1 SPI Driver on r_	rspi	g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi
Add DTC Driver for Transmission [Recommended but optional]	Add DTC Driver for Reception [Recommended but optional]		Add DTC Driver for Transmission [Recommended but optional]	Add DTC Driver for Reception [Recommended but optional]	

図 9 複数のスレーブを制御するための Synergy Configuration の階層図



スレーブ1と2の使い方の例として、以下のようなパラメータで制御することを考えます。チップセレクト信号、サンプリングタイミング、Bit rate の値が違います。Synergy Configuration の具体的な設定手順については、appendix をご参照ください。

プロパティ	スレーブ1の値	スレーブ 2 の値
Chip Select Port	01	01
Chip Select Pin	03	14
Chip Select Active Level	Low	High

表 8 チップセレクトと Active Level

表 9 転送モードと Bit Rate

プロパティ	スレーブ 1 の値	スレーブ 2 の値
Channel	0	0
Operating Mode	Master	Master
Clock Phase	Data sampling on odd edge, data variation on even edge	Data sampling on even edge, data variation on odd edge
Clock Polarity	Low when idle	High when idle
Bit rate	500000	1000000

表 10 共有バスのハードウェア資源

プロパティ	値
SPI Implementation	SPI
Channel	0

図 10 にスレーブ1と2 にそれぞれに writeRead アクセスした場合の波形を示します。

最初にスレーブ1に writeRead アクセスし、次にスレーブ2に writeRead アクセスした処理の結果になりま す。制御コードは以下のようになります。チップセレクト信号、サンプリングタイミング、Bit rate の違いが 分かります。





図 10 2 つのスレーブを制御した場合の実行波形



4. Appendix

4.1 SPI HAL ドライバの設定手順

1 新しいプロジェクトを作成します。

Step 1.1 新規の Synergy C Project を作成します。

02 C - CS	C_127425	SPI_SCK_L	ow_Polarity_Frame	work_\$7G2/cd	onfiguratio	on.xn	nl - e2 studi	io			
ファイル(F)	福集(E)	7-2(S)	リファクタリング(T)	ナビゲート(N)	検索(A)	70	ジェクト(P)	Renesas Views	実行(R)	ウィンドウ(W)	ヘルプ(H)
新規	(N)			Alt+3	7++N>	6.1	C Project				
771	ルを開く(.).					0.1	C++ Proje	ect			
朝以	5(C)			54	Ctrl+W	C	Synergy C	C Project			
すべて	(閉じる(L)			Ctrl+シ	7ト+W	64	Synergy (C++ Project			

図 11-1 Synergy C Project の作成

Step 1-2 Project Name を入力します。

Project Name を入力し、「次へ[N]」を click します。ここでは"SPI_Sample_Code"を入力しています。

1. T.		Toolchains
Project name SPI	Sample_Code	GCC ARM Embedded
☑ デフォルト・ロケー	ションの使用(D)	IAR ARM Toolchain
07-932(U): G¥	Jsers#a5103170¥e2_studio#workspace_vS31¥SPI##IR]	
icense		
License file:		Change license file
C:¥Renesas¥e2_st	udio_V531_002¥internal¥projectgen¥arm¥Licenses¥SSP_Develo	pment_and_Production_License_20170124
License Details:		
License Details: CUSTOMER INFO Company: SYBD- UserName: Dale S Email: dale.sparlir LICENSE INFORM	RMATION: Renesas parling g@renesas.com ATION: 7	^
License Details: CUSTOMER INFO Company: SYBD- UserName: Dale S Email: dale.sparlir LICENSE INFORM Issued: 24/01/201	RMATION: Renexas parling g@renexas.com ATION: 7	

図 11-2 Project Name の入力

Step 1-3 Board を選択します。 「S7G2 SK」の Board を選択し、「次へ[N]」を click します。

Device Selection	n	
SSP version:	1.2.0 ~	
Board:	S7G2 SK ∽	
Device	R7FS7G27H3A01CFC	
ulant Taala		Surgister Test
Toolchain:	GCC ARM Embedded	GCC ARM Embedded
Toolchain ver	sion: 4.9.3.20150529	4,9.3.20150529
Debuggen	J-Link ARM	J-Link ARM
		V RTOS
		Smart Manual
		IO Registers Supported
		Software Manual Supported

図 11-3 Board の選択



Step 1-4 Project の template を選択します。

「Blinky with ThreadX」を選択し、「終了[F]」を click します。

e2 studio - e2 studio - ¹	- Project Configuration (Synergy C Pro Project Configuration (Synerg	iject) gy C Project)			-	×
Select the typ	pe of project you wish to create.				_	1
Project Terr	nplate Selection					
0	BSP Base Board Support Package for [Renesas.Synergy.1.2.0.pack]	the chosen Synergy fa	mily.			
0	Blinky Blinky project. [Renesas.Synergy.1.2.0.pack]					
•	Blinky with ThreadX Threaded version of Blinky project	t				
	[Renesas.Synergy.1.2.0.pack]					
Code Gener	ration Settings erov Code Formatter					
La six of its	- 57					
0		(第3(0)	The Araba	10 7 (0)	twich	

図 11-4 Blinky with ThreadX の選択

以下のメッセージが表示しれますので、「はい[Y]」を click します。



図 11-5 パースペクディブの関連付け



2 コンフィギュレータのセットアップ
 コンフィギュレータが表示されますので、以下の手順でセットアップ
 Step 2-1 「Thread」tabを click します。
 Step 2-2 「Threads」から「Blinky Thread」を click します。

incous	6 H
HAL/Common g_elc ELC Driver on r_elc g_fmi FMI Driver on r_fmi g_ioport I/O Port Driver on	n r_ioport
Blinky Thread	

図 11-6 Blinky Thread の click

Step 2-3 「Blinky Tread Stacks」から「SPI Driver on r_rspi」を選択します。

以下の画面のように「SPI Driver on r_rspi」を選択します。SCI_SPI を使用する場合は、r_sci_spi を選択します。



図 11-7 SPI Driver on r_rspiの選択

以下のような階層が表示されます。

inky Thread Stacks		<u>କ</u> ି 着
g_spi0 SPI Driver on r_	spi	
g_transfer1 Transfer Driver on r_dtc Software Activation 1	g_transfer0 Transfer Driver on r_dtc Event SPI0 RXI	

図 11-8 SPI Driver on r_rspi ブロック図

生成されたドライバは、デフォルトとして DTC 転送を利用するになっています。 非 DTC 転送(CPU 転送)にする場合は、Step 2-4 の操作を行います。



Step 2-4 ドライバを非 DTC 転送に設定する。

非 DTC 転送で動作確認します。

DTC 転送を止めたい場合は、各ドライバ層を選択し、右上にある「x」を click します。

nky Thread Stacks		
g_spi0 SPI Driver on r_r	ipi	
g_transfer1 Transfer Driver on r_dtc Software Activation	g_transfer0 Transfer Driver on r_dtc Event SPI0 RXI	
1		

図 11-9 CPU 転送の選択

非 DTC 転送になると、以下のようにドライバの枠にパープルラインが表示されます。

g_spi0 SPI Driver on r_	rspî
A CONTRACTOR OF	





Step 2-5 「SPI Driver on r_rspi」のプロパティを変更します。

Settings	プロパティ	值	
nformation	V Common		
	Parameter Checking	Default (BSP)	
	 Module g_spi0 SPI Driver on r_rspi 		
	Name	g_spi0	
	Channel	0	
	Operating Mode	Master	
	Clock Phase	Data sampling on odd edge, data variation on even edge	
	Clock Polarity	Low when idle	
	Mode Fault Error	Disable	
	Bit Order	MSB First	
	Bitrate	500000	
	Callback	r_spi0_callback	
	SPI Mode	SPI Operation	
	SPI Communication Mode	Full Duplex	
	Slave Select Polarity(SSL0)	Active Low	
	Slave Select Polarity(SSL1)	Active Low	
	Slave Select Polarity(SSL2)	Active Low	
	Slave Select Polarity(SSL3)	Active Low	
	Select Loopback1	Normal	
	Select Loopback2	Normal	
	Enable MOSI Idle	Disable	
	MOSI Idle State	MOSI Low	
	Enable Parity	Disable	
	Parity Mode	Parity Odd	
	Select SSL(Slave Select)	SSLO	
	Select SSL Level After Transfer	SSL Level Do Not Keep	
	Clock Delay Enable	Clock Delay Disable	
	Clock Delay Count	Clock Delay 1 RSPCK	
	SSL Negation Delay Enable	Negation Delay Disable	
	Negation Delay Count	Negation Delay 1 RSPCK	
	Next Access Delay Enable	Next Access Delay Disable	
	Next Access Delay Count	Next Access Delay 1 RSPCK	
	Receive Interrupt Priority	Priority 2	
	Transmit Interrupt Priority	Priority 2	
	Error Interrupt Priority	Priority 2	

図 11-11 SPI Drive on r_rspiのプロパティ

以下の表のように設定します。必要に応じて修正してください。

プロパティ	值
Channel	0
Operating Mode	Master
Clock Phase	Data sampling on odd edge, data variation on even edge
Clock Polarity	Low when idle
Bitrate	500000
callback	r_spi0_callback
Receive Interrupt Priority	Priority2
Transmit Interrupt Priority	Priority2
Error Interrupt Priority	Priority2

表 11	^r SPI Driver on r_	_rspi」のブ	゚ロパティ	の設定例
------	-------------------------------	----------	-------	------



Step 2-6 Pins Peripherals のプロパティを設定します。

「Pins」 「Peripherals」 「ConnectivitySPI」 「SPI0」のプロパティを設定します。 CS 制御を I/O Port(GPIO)で行いますので、「SSL0」を None(P103)に設定します。

Pin Sel	ecti	on		Pin Configuration			
フィルタ	λ,	n 🖉 🕅	8				
~	~	Connectivity:SPI SPI0 SPI1	^	Module name: Usage:		SPIO For SPI, same Pin G	roup recommended
>		Connectivity:SSI		Pin Group Selection:		_A only	Ý
>	*	Connectivity:US8FS		Operation Moder			0
>	4	Connectivity:US8HS		operation mode.			
2	~	InputCISU	11	Input/Output			
1		inputKINT		MISO:	~	P100	~
5	4	Graphics:GLCDC		MOSE	-	P101	~
>		Graphics:PDC				erster Former	
>	4	Storage:QSPI		RSPCK:	~	P102	~
>		Storage:SDHI		SSL0:		None	~
>		System:BUS					
>	~	System:CGC		SSL1:		None	×
>	4	System:DEBUG		SSI 2:		None	~
>		Timer:AGT					122
>		Timer:GPT		SSL3:		None	~
>		Timer:OPS					

図 11-12 Pins Peripherals のプロパティ

Step 2-7 Pins Ports のプロパティを設定します。 「Pins」 「Ports」 「P1」 「P103」のプロパティを設定します。 Mode を「Output mode(Initial High)」に設定します。

n Selection	Pin Configuratio	n
1119入力 🔐	⊞ ⊟	
 P1 P100 P101 P102 P103 	Module name: Symbolic Name Comment:	P 103
 P104 P105 P106 P107 P108 P109 	Port Capabilitie	S BUSO: D3_DQ3 GLCDC0: LCD_TCON1 GPT2: GTIOCA KINTO: KRM3 OPS0: GTOWUP SCI0: CTS_RTS_SS SPI0: SSL0
 P110 P111 	P103 Configurat	lion
→ P112	Mode:	Output mode (Initial High) 🛛 🗸
 P113 P114 	Pull up:	None 🗸
P115	Drive Capacity:	Low 🗸
> v P2 > v P3	Output type:	CMOS ~
> V P4 > V P5	Chip input/out; P103:	GPIO GPIO

図 11-13 Pins Ports のプロパティ

以上で設定は完了です。

Step 2-10 Project Content code を生成します。

「Generate Project Content」を click します。



🗵 11-14 Generate Project Content

これでコードが生成されました。(src/synergy_gen、synergy および synergy_cfg フォルダに関連ファイルが 出力されます。)



3 アプリケーションコードの作成と動作確認

Step 3-1 アプリケーションコードを作成します。

blinky_thread_entry.c に SPI 制御コードと callback 関数のコードを記述します。callback 関数からの実行完了 通知は、グローバル変数 g_spi_done を用いて実現しています。

```
#define SSL_PIN IOPORT_PORT_01_PIN_03
#define TX_SIZE 2
volatile bool g_spi_done = false;
void blinky_thread_entry(void)
{
 ssp_err_t err;
 uint8_t wdata8[2] = \{0x01, 0x23\};
 uint8_t rdata8[2];
 err = g_spi0.p_api->open(g_spi0.p_ctrl, (spi_cfg_t *)g_spi0.p_cfg);
 g_ioport_on_ioport.pinWrite(SSL_PIN, IOPORT_LEVEL_LOW);
 err = g_spi0.p_api->writeRead(g_spi0.p_ctrl, wdata8, rdata8, TX_SIZE,
                                                     SPI_BIT_WIDTH_8_BITS);
 while (g_spi_done == false)
 g_spi_done = false;
 g_ioport_on_ioport.pinWrite(SSL_PIN, IOPORT_LEVEL_HIGH);
 err = g_spi0.p_api->close(g_spi0.p_ctrl);
}
/*** callbak 関数 ***/
void r_spi0_callback(spi_callback_args_t * p_args)
{
   if (p_args->event == SPI_EVENT_TRANSFER_COMPLETE)
   {
      g_spi_done = true;
   }
}
```

Step 3-2 build を行います。

Step 3-3 debugger を起動して、動作確認を行います。



4 補足情報

DTC のプロパティ

HAL ドライバ(RSPI)を DTC 転送ありとした場合のブロック図とそのプロパティを以下に示します。 DTC が有効になると、r_dtc の記述された階層が表示されます。

各プロパティの「Transfer Size」が4 Bytes 固定になっていることが分かります。

g_spi0 SPI Driver on r_	rspi
g_transfer0 Transfer	g_transfer1 Transfer
Driver on r_dtc	Driver on r_dtc Event



g_transfer1	Transfer Driver on r_dtc Software Activation 1		g_transfer0 Transfer Driver on r_dtc Event SPI0 RXI				
Settings	プロパティ	đ.	Cattlinger	プロパティ	đ		
Information	✓ Common		secondo	v Common			
	Parameter Checking	Default (BSP)	Information	Parameter Charking	Default (RSD)		
	Software Start	Disabled		Codeurara Start	Disabled		
	Linker section to keep DTC vector table	.ssp_dtc_vector_table		Linker restion to keep DTC verter table	cra dte verter table		
	✓ Module g_transfer1 Transfer Driver on r_dtc Software Activ	ation 1		Linker section to keep Dic Vector table	.ssp_dic_vector_table		
	Name	g_transfer1		 Module g_transfero fransfer briver on r_dit event selo rol 	a household		
	Mode	Normal		Name	G Namel		
	Transfer Size	a 4 Bytes		Mode T			
	Destination Address Mode	G Fixed		transfer size	u 4 sytes		
	Source Address Mode	(a) Incremented		Destination Address Mode	(incremented		
	Repeat Area (Unused in Normal Mode)	A Source		Source Address Mode	Fixed		
	Internet Frequency	After all transfers have completed		Repeat Area (Unused in Normal Mode)	Destination		
	Destination Pointer			Interrupt Frequency	After all transfers have completed		
	Course Deleter			Destination Pointer	M NULL		
	Number of Transform	0.0		Source Pointer	MULL .		
	Number of Blacks Official sets in Black Made)	10 0		Number of Transfers	a o		
	Number of blocks (valid only in block mode)	0.0		Number of Blocks (Valid only in Block Mode)	🙆 O		
	Activation Source (Must enable INQ)	Dottware Activation 1		Activation Source (Must enable IRQ)	Event SPI0 RXI		
	Auto Enable	a raise		Auto Enable	G False		
	Callback (Only valid with Software start)	I NULL		Callback (Only valid with Software start)	MULL .		
	ELC Software Event Interrupt Priority	Disabled		ELC Software Event Interrupt Priority	Disabled		

図 11-16 SPI Driver DTC のプロパティ

SCI_SPI のプロパティ

Step2-5 で SCI_SPI を選択した場合の SCI_SPI のプロパティは以下のようになります。

ettings	プロパティ	值
nformation	V Common	
	Parameter Checking	Default (BSP)
	 Module g_spi0 SPI Driver on r_sci_spi 	
	Name	g_spi0
	Channel	0
	Operating Mode	Master
	Clock Phase	Data sampling on odd edge, data variation on even edge
	Clock Polarity	Low when idle
	Mode Fault Error	Disable
	Bit Order	MSB First
	Bitrate	500000
	Bit Rate Modulation Enable	Enable
	Callback	r_spi0_callback
	Receive Interrupt Priority	Priority 2
	Transmit Interrupt Priority	Priority 2
	Transmit End Interrupt Priority	Priority 2
	Error Interrupt Priority	Priority 2

図 11-17 SPI Drive on r_sci_spi のプロパティ



チップセレクトを SSL にした場合の Pins Configuration

チップセレクト制御を SSL で行う場合の pins の configuration のプロパティを以下に示します。

Pin Sel	ecti	on		Pin Configuration			
フィルタ	λ,†	2 🖉 🕅	E				
~	>	Connectivity:SPI	^	Module name: Usage:		SPIO For SPI, same Pin G	roup recommended
>		Connectivity:SSI		Pin Group Selection:		_A only	~
>	*	Connectivity:USBFS		Operation Moder		1	
>	~	Connectivity:USBHS		operation mode.			1.1
>	~	Input:CTSU	11	Input/Output			
12		Inputiku		MISO:	4	P100	~
5	~	Graphics:GLCDC		MOSI:	-	P101	~
3	,	Graphics:PDC Storage:OSPI		RSPCK:	-	P102	¥
- 2		Storage:SDHI	1	SSL0:	~	P103	~
3	~	System:BUS System:CGC	12	SSL1:		None	¥
>	*	System:DEBUG		SSL2:		None	~
5		Timer:GPT		SSL3:		None	~
		Timer:OPS					

図 11-18 SSL 端子として使用する場合 SPI プロパティ

P103の Configuration は以下のようになります。

~
v
~
~

図 11-19 SSL 端子として使用する場合の P103 Configuration



コールバック関数内で semaphore を使用した場合の設定例 semaphore を使用したコールバック関数の例を以下に示します。 Step 1 callback 関数で使用する semaphore を定義します。 「Blinky Thread Objects」から semaphore を click します。



図 11-20 Blinky Thread Objects の semaphore を click

```
Step 2 semaphore のプロパティを設定します。
Symbol 名を「g_spi0_semaphore」にします。
g_spi0_semaphore Semaphore
Settings
プロパティ 値
Name R SPI Semaphore
Symbol g_spi0_semaphore
Initial count 0
```

図 11-21 semaphore のプロパティ

```
/*** コールバック関数からの転送完了を待つ ***/
tx_semaphore_get(&g_spi0_semaphore, TX_WAIT_FOREVER);
/*** コールバック関数 ***/
void g_spi0_callback(spi_callback_args_t * p_args)
{
    if (p_args->event == SPI_EVENT_TRANSFER_COMPLETE)
    {
        tx_semaphore_ceiling_put(&g_spi0_semaphore, 1);
    }
}
```

SSP 未対応の Data length について

R_SPIのHALドライバでは、8、16、32ビットの転送ビット長のみサポートしていますが、SynergyMCUの機能としては、転送ビット長を8、9、10、11、12、13、14、15、16、20、24、32ビットから選択可能になっています。SSPモジュールでサポートしていない転送ビット長を使用したい場合には、SPIHALドライバのコードが開示されていますので、ユーザご自身でカスタマイズすることができます。

read/write 用のデータバッファのアライメントについて

転送ビット長が16ビットや32ビットである場合、その転送ビット長でデータバッファにアクセスします。 データバッファの配列は転送ビット長に合わせて宣言してください。

4.2 SPI フレームワークの設定手順

1 新しいプロジェクトを作成します。

- この部分は、HALドライバの設定手順と同じです。
- 2 コンフィギュレータのセットアップ
 - コンフィギュレータが表示されますので、以下の手順でセットアップ
 - Step 2-1 「Thread」 tab を click します。
 - Step 2-2 「Threads」から「Blinky Thread」を click します。



図 12-1 Blinky Thread の click

Step 2-3 「Blinky Tread Stacks」から「SPI Framework Device on sf_spi」を選択します。 以下の画面のように「SPI Framework Device on sf_spi」を選択します。

leau	STACKS		•	Driver	>
		Analog	>	Framework	>
		Audio	>	X-Ware	>
\$	Communications Framework on sf_el_nx_comms	Connectivity	>		P400
	Communications Framework on sf_el_ux_comms	File System	>		P401
	Communications Framework on sf_uart_comms	Graphics	>		P403
•	I2C Framework Device on sf_i2c	Input	>		P405
	I2C Framework Shared Bus on sf_i2c	Networking	>		P406
*	SPI Framework Device on sf_spi	Services	>		P701 -
	SPI Framework Shared Bus on sf_spi	USB	>		P703
	[DEPRECATED] Communications Framework on sf_el_ux_comms				P705
_					-nu 3

図 12-2 SPI Framework device on sf_spiの選択

以下のような階層が表示されます。

g_sf_spi_device0 SPI Framework Device of sf_spi					
Add SPI Driver	g_sf_spi_bus0 SPI Framework Shared				

図 12-3 SPI Framework Device on sf_spi ブロック図

Step 2-4 「Blinky Tread Stacks」から「SPI Driver on r_rspi」を選択します。

この階層から「Add SPI Driver」を click し、使用する SPI を選択します。「SPI Driver on r_rspi」を選択します。SCI_SPI を使用する場合は、「SPI Driver on r_sci_spi」を選択します。



g_sf_spi_device0 SPI F sf_spi	ramework	Device	on	
		-	6	1
l Add SPI Driver	g_sf_sp Frame Bus or	bi_bus0 work Sh i sf_spi	SPI ared	
Add SPI Driver	g_sf_sp Frame Bus or	bi_bus0 work Sh sf_spi	SPI ared PI Drive	r on r_rspi

図 12-4 使用する SPI Driver の選択

Step 2-5 ドライバを非 DTC 転送に設定する。

SPI Driver の選択が完了すると DTC を使用する Driver の階層が表示されます。

DTC を使用しない場合は、DTC の枠を選択し、右上の「x」を選択して下さい。DTC を使用しない枠がパープル色で表示されます。

C			
_st_spi_device0 SPI Fra	mework Device on sf_sp	ū	
_spi0 SPI Driver on r_rs	spi	g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi	
transfer() Transfer	a transfer1 Transfer		
oftware Activation	Driver on r_dtc Event SPI0 RXI		
nky Thread Stacks			
nky Thread Stacks g_sf_spi_device0 SPI Fr	ramework Device on sf_s	pi	
nky Thread Stacks g_sf_spi_device0 SPI Fr	amework Device on sf_s	pi	
nky Thread Stacks g_sf_spi_device0 SPI Fr g_spi0 SPI Driver on r_	ramework Device on sf_s	pi g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi	
nky Thread Stacks g_sf_spi_device0 SPI Fr g_spi0 SPI Driver on r_	ramework Device on sf_s	pi g_sf_spi_bus0 SP1 Framework Shared Bus on sf_spi	

図 12-5 CPU 転送を選択した場合のプロック図

この構成図が SPI フレームワークの基本構成になります。複数のスレーブを制御する場合は、この基本構成を複数生成する必要があります。以下で各階層のプロパティを設定します。

- ・「SPI Framework Device on sf_spi」のプロパティの設定
- ・「SPI Driver on r_rspi」のプロパティの設定
 - 「SPI Driver on r_rspi」のプロパティの設定手順は 4.1 のドライバの Configuration と同じになります。
- ・「SPI Framework Shared Bus on sf_spi」プロパティの設定



Step 2-6 「SPI Framework Device on sf_spi」のプロパティを設定します。

チップセレクト PORT の設定になります。使用する I/O PORT の番号を設定します。Chip Select Active Level については、使用するスレーブデバイスの仕様に合わせてください。

Settings	70パティ	値
Information	Common	D. (
	Parameter Checking	Default (BSP)
	 Module g_sf_spi_device0 SPI Framework Device on sf_spi 	
	Name	q sf spi device
	Chip Select Port	01
	Chip Select Pin	03
	Chip Select Active Level	Low

図 12-6 チップセレクトを設定するプロパティ

Step 2-7 「SPI Framework Shared Bus on sf_spi」プロパティの設定を行います。

「SPI Implementation」は、使用する SPI の機能を選択します。RSPI を選択する場合は、SPI になり、SCI_SPI を選択する場合は、SCI_SPI になります。

「Channel」は、Synergy MCU で使用できるチャネル数がありますので、使用するチャネル番号を設定します。

Settings	プロパティ	値
Information	✓ Common	
	Parameter Checking	Default (BSI
	 Module g_sf_spi_bus0 SPI Framework Share 	ed Bus on sf_spi
	Name	<u>q_sf_spi_bu</u>
	SPI Implementation	SPI
	Channel	0

図 12-7 共有バスのプロパティ

スレーブが1つの場合は、SPIフレームワークのプロパティの設定はこれで完了です。複数のスレーブを使用する場合の設定手順は、4.3 を参照してください。

HAL Driver の設定が完了すれば、Synergy のコード生成を行い、build、実行することができます。

制御コードは以下のようになります。



4.3 SPI フレームワーク 共有バスの設定手順

2つのスレーブデバイスを制御する場合、SPIフレームワークを2つ作成する必要があります。

2つ目の SPI フレームワークを生成した時の階層図を以下に示します。

g_sf_spi_device0 SPI Framework Device on sf_spi			g_sf_spi_device1 SP sf_spi	I Framework Device on
g_spi0 SPI Driver on r_i	I rspi	g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi	Add SPI Driver	Add SPI Framework Shared Bus
Add DTC Driver for Transmission [Recommended but optional]	Add DTC Driver for Reception [Recommended but optional]			

図 13-1 2つ目の SPI フレームワークを生成直後の階層図

次に「Add SPI Driver」を click し、使用する SPI HAL ドライバを選択します。HAL Driver の DTC 機能を使用しない設定にします。

g_sf_spi_device0 SPI Framework Device on sf_spi		g_sf_spi_device1 SPI Framework Device on sf_spi			
g_spi0 SPI Driver on r_	1 rspi	g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi	g_spi1 SPI Driver on r_t	I rspi	Add SPI Framework Shared Bus
Add DTC Driver for Transmission [Recommended but optional]	Add DTC Driver for Reception [Recommended but optional]		Add DTC Driver for Transmission [Recommended but optional]	Add DTC Driver for Reception [Recommended but optional]	

図 13-2 2 つの SPI フレームワークの HAL ドライバを設定した階層図

次に 2 つの SPI フレームワークで使用する共有バスのプロパティを同じ設定にします。「Use」-> 「g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi」を選択します。

これにより、g_sf_spi_device0 と g_sf_spi_device1 は、同じ共有バスを参照し、使用する SPI Driver を排他 的に制御することができます。

sf_spi_device0 SPI Fi	amework Device on sf_sp	i -	g_sf_spi_device1 SPI Fra	amework Device on sf_sp	Di l	
spi0 SPI Driver on r_	1 rspi	g_sf_spi_bus0 SPI Framework Shared Bus on sf spi	g_spi1 SPI Driver on r_r	spi	Add SPI Frame Shared Bus	work
			g_sf_spi_bus0 SPI	Framework Shared Bus o	on sf_spi	New Use
a DIC Driver for insmission ecommended but itional]	Reception [Recommended but optional]		Transmission [Recommended but optional]	Reception [Recommended but optional]		
ky Thread Stacks	Framework Device on sf	spi	g_sf_spi_device1 SPI	Framework Device on sf	_spi	
		7				
	r rspi	g_st_spi_bus0 SPI	g_spi1 SPI Driver on r	r_rspi	g_st_spi_bu Framework	s0 SPI Shared
g_spi0 SPI Driver on		Framework Shared Bus on sf_spi			Bus on sf_sp	pi

図 13-3 2つの共有バスを同じ設定にする手順

以上で SPI フレームワークの基本構成が2つ生成されたことになります。以下で各階層のプロパティの設定を行っていきます。

- ・「SPI Framework Device on sf_spi」のプロパティの設定
- ・「SPI Driver on r_rspi」のプロパティの設定
- ・「SPI Framework Shared Bus on sf_spi」プロパティの設定

以下の仕様で2つのスレーブを制御するようにします。

「SPI Framework Device on sf_spi」のプロパティの設定は以下のようになります。

表 12 「SPI Framework Device on sf_spi」のプロパティ値

プロパティ	スレーブ 1 の値	スレーブ 2 の値
Chip Select Port	01	01
Chip Select Pin	03	14
Chip Select Active Level	Low	High



設定したプロパティの情報

g_sf_spi_device0 SPI Framework Device on sf_spi

S <mark>etting</mark> s	プロパティ	値	
Information	✓ Common		
	Parameter Checking	Default (BS	P)
	 Module g_sf_spi_device0 SPI Framework Device on sf_spi 		
	Name	g_sf_spi_de	evice0
	Chip Select Port	01	
	Chip Select Pin	03	
	Chip Select Active Level	Low	

Settings	לם (לקי ביו לי ביו ל ביו לי ביו לי		
Information	Common Darameter Checking	Default (B	(D2
	 Module g_sf_spi_device1 SPI Framework Device on sf_spi 	Delaute (D	GF)
	Name	q_sf_spi_d	levice1
	Chip Select Port	01	
	Chip Select Pin	14	
	Chip Select Active Level	High	

図 13-4 SPI Framework Device on sf_spiのプロパティ

「SPI Driver on r_rspi」のプロパティの設定は以下のようになります。

プロパティ	スレーブ 1 の値	スレーブ 2 の値
Channel	0	0
Operating Mode	Master	Master
Clock Phase	Data sampling on odd edge, data variation on even edge	Data sampling on even edge, data variation on odd edge
Clock Polarity	Low when idle	High when idle
Bitrate	500000	1000000

表 13 「SPI Driver on r_rspi」のプロパティ値

Settings	ี้ วื่อที่ร่า	値	
Information	✓ Common		
inomation	Parameter Checking	Default (BSP)	
	 Module g_spi0 SPI Driver on r_rspi 		
	Name	g_spi0	
	Channel	0	
	Operating Mode	Master	
	Clock Phase	Data sampling on odd edge, data variation on even edge	
	Clock Polarity	Low when idle	
	Mode Fault Error	Disable	
	Bit Order	MSB First	
	Bitrate	500000	
	Callback	NULL	
	SPI Mode	SPI Operation	
	SPI Communication Mode	Full Duplex	
	Slave Select Polarity(SSL0)	Active Low	
	Slave Select Polarity(SSL1)	Active Low	
	Slave Select Polarity(SSL2)	Active Low	
	Slave Select Polarity(SSL3)	Active Low	
	Select Loopback1	Normal	
	Select Loopback2	Normal	
	Enable MOSI Idle	Disable	
	MOSI Idle State	MOSI Low	
	Enable Parity	Disable	
	Parity Mode	Parity Odd	
	Select SSL(Slave Select)	SSLO	
	Select SSL Level After Transfer	SSL Level Do Not Keep	
	Clock Delay Enable	Clock Delay Disable	
	Clock Delay Count	Clock Delay 1 RSPCK	
	SSL Negation Delay Enable	Negation Delay Disable	
	Negation Delay Count	Negation Delay 1 RSPCK	
	Next Access Delay Enable	Next Access Delay Disable	
	Next Access Delay Count	Next Access Delay 1 RSPCK	
	Receive Interrupt Priority	Priority 2	
	Transmit Interrupt Priority	Priority 2	
	Error Interrupt Priority	Priority 2	

g_spi1 SPI Driver on r_rspi

Settings	プロパティ	值
Information	V Common	
ntormation	Parameter Checking	Default (BSP)
	 Module g_spi1 SPI Driver on r_rspi 	
	Name	g_spi1
	Channel	0
	Operating Mode	Master
	Clock Phase	Data sampling on even edge, data variation on odd edge
	Clock Polarity	High when idle
	Mode Fault Error	Disable
	Bit Order	MSB First
	Bitrate	1000000
	Callback	NULL
	SPI Mode	SPI Operation
	SPI Communication Mode	Full Duplex
	Slave Select Polarity(SSL0)	Active Low
	Slave Select Polarity(SSL1)	Active Low
	Slave Select Polarity(SSL2)	Active Low
	Slave Select Polarity(SSL3)	Active Low
	Select Loopback1	Normal
	Select Loopback2	Normal
	Enable MOSI Idle	Disable
	MOSI Idle State	MOSI Low
	Enable Parity	Disable
	Parity Mode	Parity Odd
	Select SSL(Slave Select)	SSL0
	Select SSL Level After Transfer	SSL Level Do Not Keep
	Clock Delay Enable	Clock Delay Disable
	Clock Delay Count	Clock Delay 1 RSPCK
	SSL Negation Delay Enable	Negation Delay Disable
	Negation Delay Count	Negation Delay 1 RSPCK
	Next Access Delay Enable	Next Access Delay Disable
	Next Access Delay Count	Next Access Delay 1 RSPCK
	Receive Interrupt Priority	Priority 2
	Transmit Interrupt Priority	Priority 2
	Error Interrupt Priority	Priority 2

図 13-5 SPI Driver on r_rspiのプロパティ



「SPI Framework Shared Bus on sf_spi」プロパティの設定は以下のようになります。

表 14	^r SPI Framework Shared Bus on sf_	_spi」のプロパティ値
------	--	--------------

プロパティ	值
SPI Implementation	SPI
Channel	0

Settings	プロパティ	値	
Information	✓ Common		
	Parameter Checking	Defaul	t (BSP)
	 Module g_sf_spi_bus0 SPI Framework Shared Bus on sf_spi 		
	Name	g sf spi busl	
	SPI Implementation	SPI	
	Channel	0	

図 13-6 SPI Framework Bus on sf_spiのプロパティ

以上で共有バスの設定は完了です。

コード生成します。

Build 時のエラーを回避するためにプロジェクトのプリプロセッサに「SSP_SUPPRESS_ISR_g_spi1」のマクロ定義の設定を行います。

プロジェクトのプロパティを選択します。

「C/C++ ビルド」 「Settings」 「Cross ARM C Compiler」 「Preprocessor」を選択します。

Defined symbols (-D)で「SSP_SUPPRESS_ISR_g_spi1」を追加します。

「適用(L)」 「OK」を click で完了です。



図 13-7 「SSP_SUPPRESS_ISR_g_spi1」マクロ定義の設定

Build、実行します。



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/

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。



改訂記録

			改訂内容	
Rev.	発行日	ページ	ポイント	
1.00	2017.06.15	-	初版	

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。

٦

1.	未使用端子の処理
	【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用
	端子の処理」で説明する指示に従い処理してください。
2.	電源投入時の処置
	【注意】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.	リザーブアドレス(予約領域)のアクセス禁止
	【注意】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、将来の機能拡張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。
4.	クロックについて
	【注意】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
	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たさい。また、フロク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
_	しから切り省えてくたさい。
5.	彩品間の相遅に しい (【注金】 副なる思わる制具に変更する場合は、制具副なず にに シュニノ 評価試験を実施し てくざき
	【注息】型名の異なる製品に変更9る場合は、製品型名ことにシステム評価試験を美施してください。
	vi。 同じグループのフィコンでも刑夕が造こと。内部DOM レイマウトパターンの知道かどにとい。電
	回しフルーフのミキョフにも空石が建つて、内部KOW、アキアファハターブの相逢なとにより、電気的特徴の範囲で、特性値、動作フェジン、ノイブ耐旱、ノイブ転射旱たドが思たて担合がもりま
	式的特性の範囲で、特性値、動作マーンノ、ノイス脳重、ノイス輻射重なとか共なる場合がのりま

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。

 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。お客様の機器・システムの設計におい て、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には、お客様の責任において行ってください。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。 当社製品、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。 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。 当社製品を、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製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。 当社は、当社製品の品質水準を「標準水準」あよび「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、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、海底中総器、原子力制御システム、航空機制御システム、ブ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、これらの印絵ではするたとてきません、たとき、音照したい田絵ではかり開きた使用したことにに増売数件にする、単計は一切るの手はを白い
 し、回路、 シブドウェアあよびこれらに制建する情報を使用する場合には、あ各核の負任において行うてください。これらの使用に起因して生むた損害(あ各核 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、本資料に記載された製品データ、図、表、ブログラム、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、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製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。 5. 当社は、当社製品の品質水準を「標準水準」あよび「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、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、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、こわらの用途に使用することはできません。たとう、意図したい用途に当社制品を使用したことにに加速すべたいです。
 2. 当社製品、準算料に起動とれた製品) 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、
 3. 当在は、本資料に基しざ当在または第二者の特許権、著作権その他の知道的解産権を何ら許諾するものではありません。 4. 当社製品を、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製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。 5. 当社は、当社製品の品質水準を「標準水準」あよび「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、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、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 こわらの用途に使用することはできません、たとう 意図したい用途に当社製品を使用したことにとい場害がたいです。当社は一切るの表示がたいです。
は、一切その責任を負いません。 5. 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、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、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できません、たとう、音図したい用途に当社製品を使用したことにに知識率がたいても、当社ました。
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、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、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 こわらの田珍に使用することはできません。たとう 音図したい田珍に当社製品を使用したことにに知道をがたいても 当社は一切るの表示
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、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、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できません。たとう、意図したい用途に当社制品を使用したことにに10歳実がたいても、当社は一切るの表でを含い
当社製品は、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、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、こわらの田珍に使用することはできません。たとう、音図したい田珍に当社製品を使用したことにとい場害がたいです。当社は一切るの表示を含い
してのション、これシシの内陸に使用することはてきよどれ。にこれ、忌凶しない内陸に当社装配を使用したことにより損害が生しても、当社は一切での員社を負い
ません。 6. 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、信頼性ハンドプックに記載の「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。
7. 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。
8. 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。
9. 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。また、当社製品および技術 を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル(無人航空機を含みます。)の開発、設計、製造、使用もし くは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、か つ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
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。
10. お客様の転売、貸与等により、本書(本ご注意書きを含みます。)記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その 責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
11. 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。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。
注1. 今資料において使用されている「当住」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その総株主の議決権の過半数を 直接または間接に保有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.3.0-1 2016.11)

そこれにあった。 ルネサスエレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24(豊洲フォレシア)

■技術的なお問合せおよび資料のご請求は下記へどうぞ。 総合お問合せ窓口:https://www.renesas.com/contact/

@ 2017 Renesas Electronics Corporation. All rights reserved. Colophon 5.0