

## Traveo™ファミリのマルチファンクションシリアルインタフェース使用方法

著者: Masahide Karino

関連製品ファミリ: Traveo ファミリ

S6J3110/3120/3200/3300/3350/3360/3370/3400 シリーズ

関連ドキュメント: [5 関連ドキュメント](#)

本アプリケーションノートは、サイプレスの Traveo ファミリ S6J3110/3120/3200/3300/3350/3360/3370/3400 シリーズのマルチファンクションシリアルインタフェース使用方法について記載しています。

### 目次

1	はじめに .....	1	3.4	I <sup>2</sup> C 通信の設定 .....	24
2	マルチファンクションシリアルインタフェースの概要 .....	1	4	要約 .....	31
3	マルチファンクションシリアルインタフェースの設定 .....	2	5	関連ドキュメント .....	31
3.1	UART 通信の設定 .....	2	6	改訂履歴 .....	32
3.2	CSIO 通信の設定 .....	9		セールス、ソリューションおよび法律情報 .....	33
3.3	LIN 通信の設定 .....	14			

## 1 はじめに

本アプリケーションノートは、Traveo ファミリ S6J3110/3120/3200/3300/3350/3360/3370/3400 シリーズのお客様を対象として、マルチファンクションシリアルインタフェースの使用法について説明しています。

## 2 マルチファンクションシリアルインタフェースの概要

マルチファンクションシリアルインタフェースは、UART (非同期シリアルインタフェース)、CSIO (SPI 対応、クロック同期シリアルインタフェース)、LIN インタフェース (LIN 通信制御インタフェース (V2.1))と I<sup>2</sup>C のシリアル通信機能を搭載しています。

UART は、外部デバイスと非同期通信をするための汎用のシリアルデータ通信インタフェースです。双方向通信機能 (ノーマルモード)、マスタ/スレーブ型通信機能 (マルチプロセッサモード: マスタ/スレーブ両方)をサポートしています。

CSIO は、外部デバイスと同期通信をするための汎用のシリアルデータ通信インタフェース (SPI に対応)です。双方向通信機能とチップセレクトコントロール機能をサポートしています。

LIN インタフェースは、LIN バスに対応するための機能をサポートしています。双方向通信において、マスタ/スレーブとして動作します。また、マニュアルモードとアシストモードをサポートしています。アシストモードでは LIN 通信におけるヘッダ部の自動送信、自動検出が可能です。

I<sup>2</sup>C インタフェースは、I<sup>2</sup>C バスをサポートし、I<sup>2</sup>C バス上のマスタ/スレーブデバイスとして動作します。I<sup>2</sup>C の通信モードとして、スタンダードモード (100 kbps)とファストモード (400 kbps)をサポートしています。[Table 1](#) に Traveo ファミリの通信モードの対応状況を示します。

Table 1. Traveo Family の I<sup>2</sup>C 通信モード

シリーズ	100 kbps	100 kbps / 400 kbps
S6J3110	全チャンネル	–
S6J3120	ch.0, ch.3, ch.4, ch.8～ch.11	–
S6J3200	ch.4, ch.10, ch.12	ch.16, ch.17
S6J3310/3320/3330/3340	ch.0, ch.1, ch.4, ch.8～ch.12	ch.16, ch.17
S6J3350	ch.0, ch.1, ch.4, ch.8～ch.12	ch.16, ch.17
S6J3360/3370	ch.0, ch.1, ch.4, ch.5, ch.8～ch.11	ch.6, ch.7
S6J3400	ch.0～ch.2, ch.4, ch.6～ch.13	ch.3, ch.5

### 3 マルチファンクションシリアルインタフェースの設定

S6J3110/3120/3200/3300/3350/3360/3370/3400 シリーズにおいて、マルチファンクションシリアルインタフェースを使うためには特定の設定が必要です。ここでは、UART, CSIO, LIN と I<sup>2</sup>C の使用方法を説明します。

#### 3.1 UART 通信の設定

Figure 1 に UART 通信の接続例を示します。

Figure 1. UART 通信の接続例

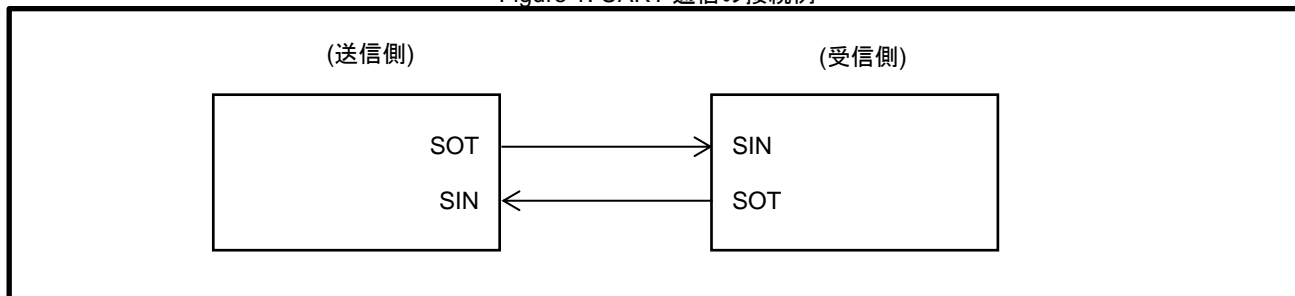


Figure 2 と Figure 3 の UART 通信のフローチャート例を示します。

Figure 2. UART 通信フローチャート例 (FIFO 未使用)

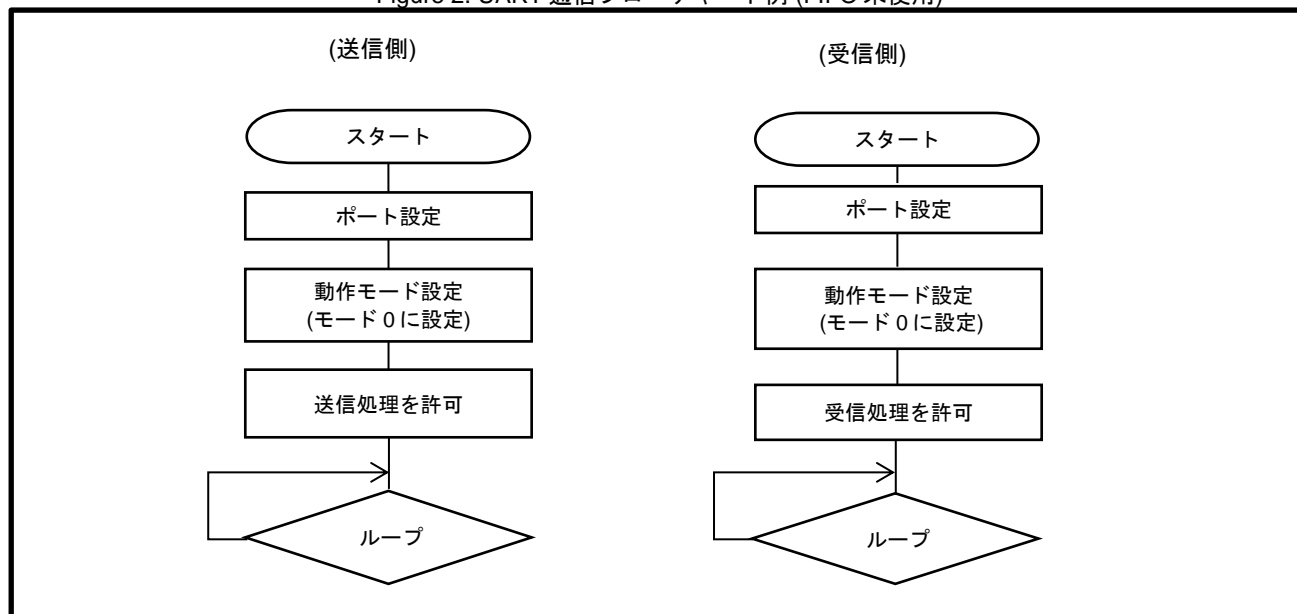
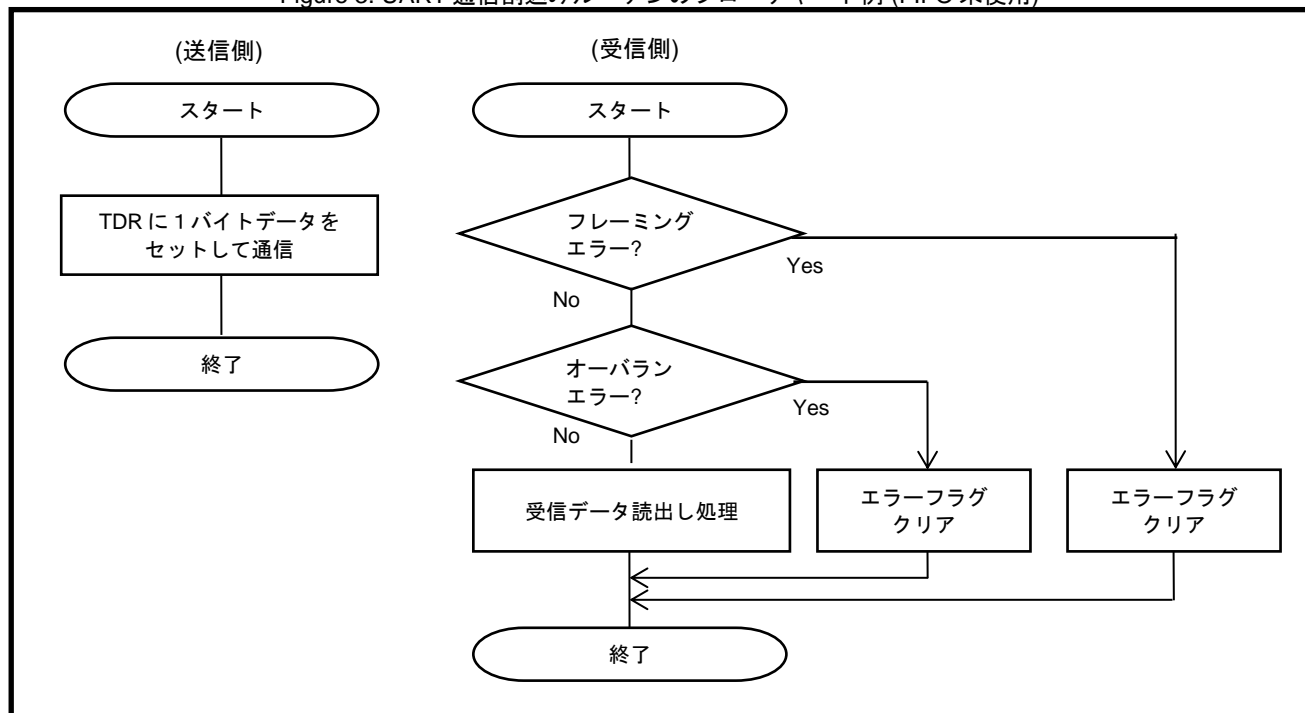


Figure 3. UART 通信割込みルーチンのフローチャート例 (FIFO 未使用)



### 3.1.1 ボーレートの設定

15 ビットリロードカウンタは、ボーレートジェネレータレジスタ(BGR0/1)で設定します。ボーレートの計算式を以下に示します。

- リロード値の計算式

$$V = (f/b) - 1$$

V: リロード値, f: バスクロック周波数, または外部クロック周波数 (Hz), b: ボーレート (bps)

- リロード値の計算例

バスクロック周波数 36MHz, ボーレートを 115200 bps に設定する場合のリロード値を以下に示します。

リロード値:

$$V = (36000000 / 115200) - 1 = 311$$

よって、ボーレートは、

$$b = (36000000 / (311+1)) = 115384 \text{ bps}$$

### 3.1.2 UART 送信/受信のポート設定プログラム例

Code1 に、UART 送信/受信の設定プログラム例を示します。

Code1. UART 設定プログラム例 (S6J3110 シリーズの場合)

ポート設定	<pre> int main(void) {     /* Keycode register setting for SOT0_0 */     PPC_KEYCDR=0x10000028;     PPC_KEYCDR=0x50000028;     PPC_KEYCDR=0x90000028;     PPC_KEYCDR=0xD0000028;     PPC_PCFGR020=0xC002;      /* Keycode register setting for SIN1_0 */     PPC_KEYCDR=0x10000038;     PPC_KEYCDR=0x50000038;     PPC_KEYCDR=0x90000038;     PPC_KEYCDR=0xD0000038;     PPC_PCFGR028=0x1000;      /* Keycode register setting for AN2 analog input disable */     ADER_KEYCDR=0x20000804;     ADER_KEYCDR=0x60000804;     ADER_KEYCDR=0xA0000804;     ADER_KEYCDR=0xE0000804;     ADER_ADER0=0xFFFFFFFF;      /* Keycode register setting for SOT0_0 output */     GPIO_KEYCDR=0x20000204;     GPIO_KEYCDR=0x60000204;     GPIO_KEYCDR=0xA0000204;     GPIO_KEYCDR=0xE0000204;     GPIO_DDR0=0x00100000;      /* Keycode register setting for port input enable */     GPIO_KEYCDR=0x20000400;     GPIO_KEYCDR=0x60000400;     GPIO_KEYCDR=0xA0000400;     GPIO_KEYCDR=0xE0000400;     GPIO_PORTEN=0x00000001;           </pre>	キーコードレジスタの設定  PPC_PCFGR レジスタで SOT0_0 を設定  PPC_PCFGR レジスタで SIN1_0 を設定  ADER_ADER0 レジスタで AN2 を設定
	<pre>     /* Keycode register setting for SOT0_0 output */     GPIO_KEYCDR=0x20000204;     GPIO_KEYCDR=0x60000204;     GPIO_KEYCDR=0xA0000204;     GPIO_KEYCDR=0xE0000204;     GPIO_DDR0=0x00100000;           </pre>	GPIO_DDR0 レジスタで SOT0_0 を出力設定
	<pre>     /* Keycode register setting for port input enable */     GPIO_KEYCDR=0x20000400;     GPIO_KEYCDR=0x60000400;     GPIO_KEYCDR=0xA0000400;     GPIO_KEYCDR=0xE0000400;     GPIO_PORTEN=0x00000001;           </pre>	GPIO_PORTEN0 レジスタ で端子の入力許可設定
	<pre>     /* Uart ch1 Rx */     Asynch_Uart1_RX();           </pre>	受信側初期設定, 受信処理開始
	<pre>     /* Uart ch0 Tx */     Asynch_Uart0_TX();           </pre>	送信側初期設定, 送信処理開始
	<pre>     /* Endless loop */     for(;;)     {         ClearWatchdog();     } }           </pre>	

動作モード / フォーマット 設定

### 3.1.3 UART 送信側設定プログラム例

Code2 と Code3 に、UART 送信側の設定プログラム例を示します。

Code2. UART 送信側設定プログラム例 (S6J3110 シリーズの場合)

```

/* Asynchronous Serial Interface (Transmission) */
void Asynch_Uart0_TX(void)
{
    /* UART0 status initialize */
    CPG_MFS00_UART_SCR_UPCL=1;

    /* Operation mode setting (Asynchronous normal mode)*/
    CPG_MFS00_UART_SMR=0x00;

    /* Enable serial data output */
    CPG_MFS00_UART_SMR_SOE=1;

    /* Baud rate setting (115.2kbps, 36MHz) */
    CPG_MFS00_UART_BGR=311;

    /* UART0 transmission setting initialize */
    CPG_MFS00_UART_SCR=0x00;

    /* Data format setting (8bit length) */
    CPG_MFS00_UART_ESCR=0x00;

    /* Enable transmission interrupt */
    CPG_MFS00_UART_SCR_TIE=1;

    /* Enable transmission operation */
    CPG_MFS00_UART_SCR_TXE=1;
}

```

UART 送信側初期設定

動作モードの初期設定

ボーレート設定

データフォーマット設定

送信側割込み処理許可

送信側動作開始

Code 3. UART 送信側割込みルーチン設定例(S6J3110 シリーズの場合)

```

/* UART0 Transmission interrupt routine */
FN_IRQ_DEFINE_BEGIN(Uart0_Tx_Interrupt, INTERRUPTS_IRQ_NUMBER_65)
{
    /* Transmission data register setting */
    CPG_MFS00_UART_TDR=tr_data[tr_num];

    /* Data count */
    tr_num++;
    if(tr_num == 10){
        tr_num = 0;
    }
}
FN_IRQ_DEFINE_END()

```

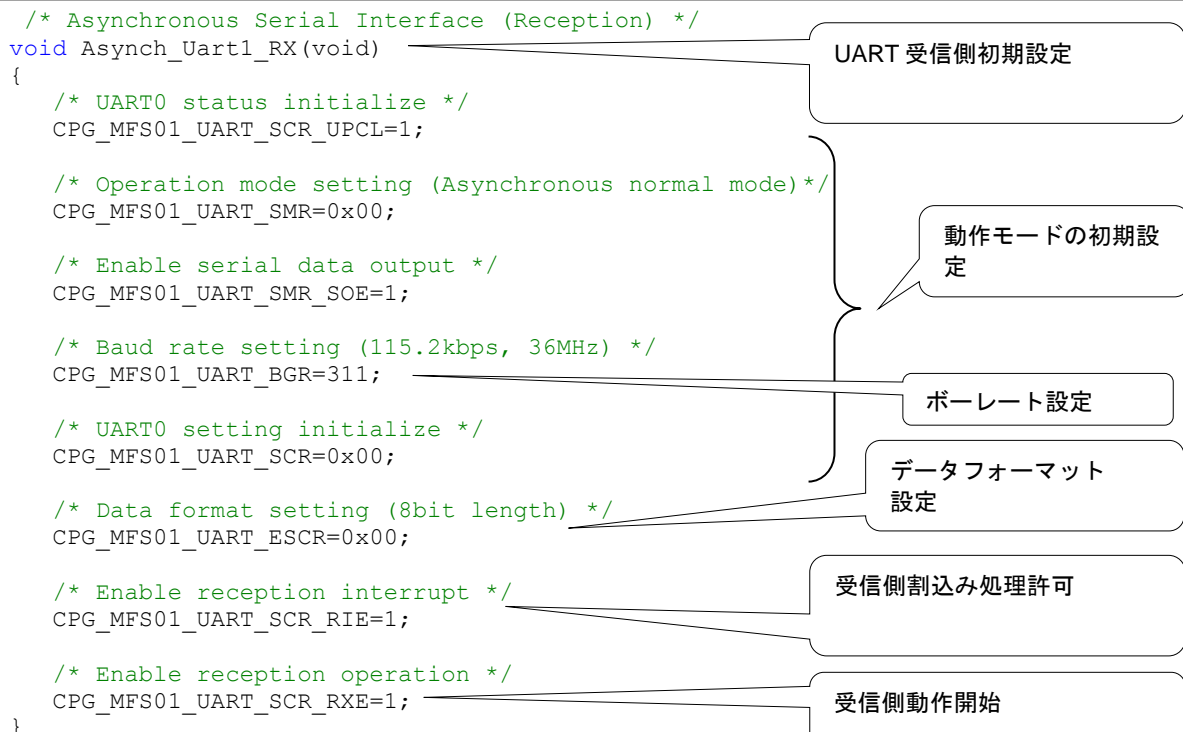
割込みルーチン

1バイトデータを TDR レジスタへ設定

### 3.1.4 UART 受信側設定プログラム例


Code 4 と Code 5 に UART 受信側の設定プログラム例を示します。

Code 4. UART 受信側設定プログラム例 (S6J3110 シリーズの場合)



Code 5. UART 受信側割込みルーチン設定例(S6J3110 シリーズの場合)

割込み  
ルーチン



```

/* UART1 Reception interrupt routine */
FN_IRQ_DEFINE_BEGIN(Uart1_Rx_Interrupt, INTERRUPTS_IRQ_NUMBER_66)
{
    /* Framing error */
    if(CPG_MFS01_UART_SSR_FRE==1){
        /* Reception error flag clear */
        CPG_MFS01_UART_SSR_REC=1;
    }
    /* Overrun error */
    else if(CPG_MFS01_UART_SSR_ORE==1){
        /* Reception error flag clear */
        CPG_MFS01_UART_SSR_REC=1;
    }
    else{
        /* Reception data register read */
        re_data[re_num]=CPG_MFS01_UART_RDR;

        /* Data count */
        re_num++;
        if(re_num == 10){
            re_num = 0;
        }
    }
}
FN_IRQ_DEFINE_END()

```

フレーミングエラー,  
オーバランエラー  
チェックとフラグ  
クリア

受信データ読出し



### 3.2 CSIO 通信の設定

Figure 4 に CSIO 通信の接続例を示します。

Figure 4. CSIO 通信の接続例

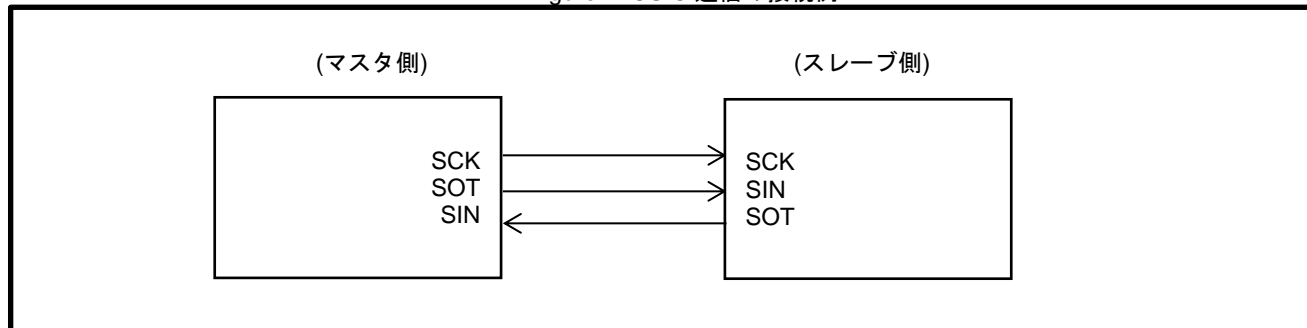


Figure 5 と Figure 6 に CSIO 通信のフローチャート例を示します。

Figure 5. CSIO 通信のフローチャート例 (FIFO 未使用)

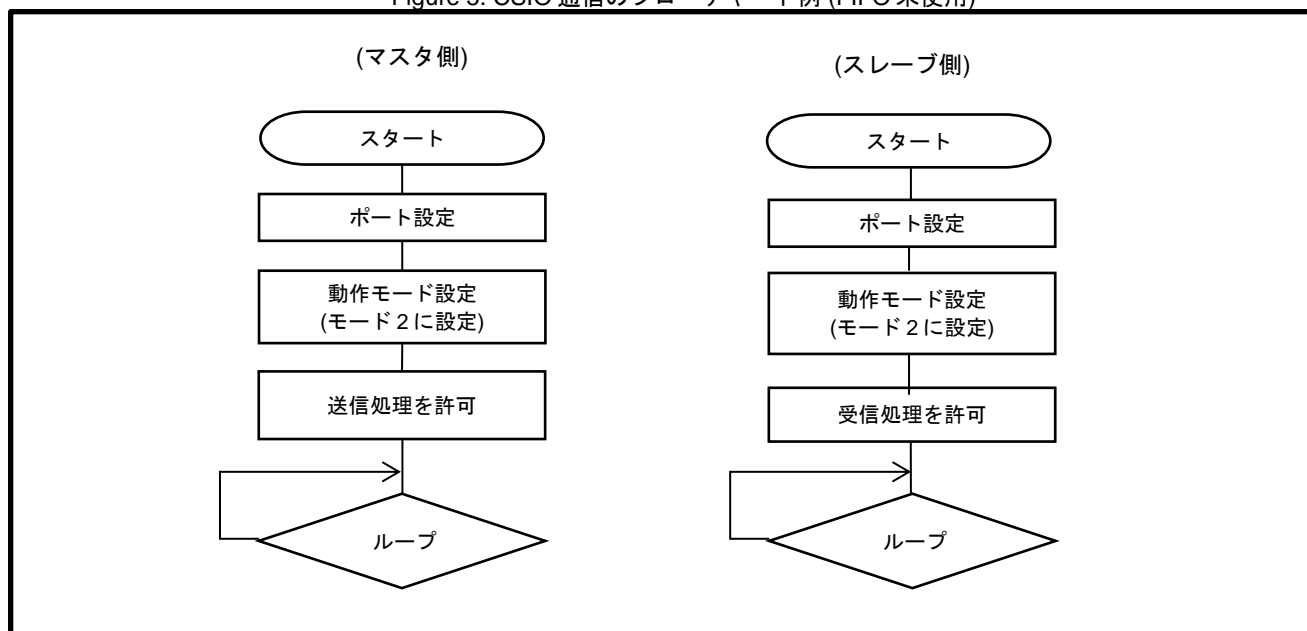
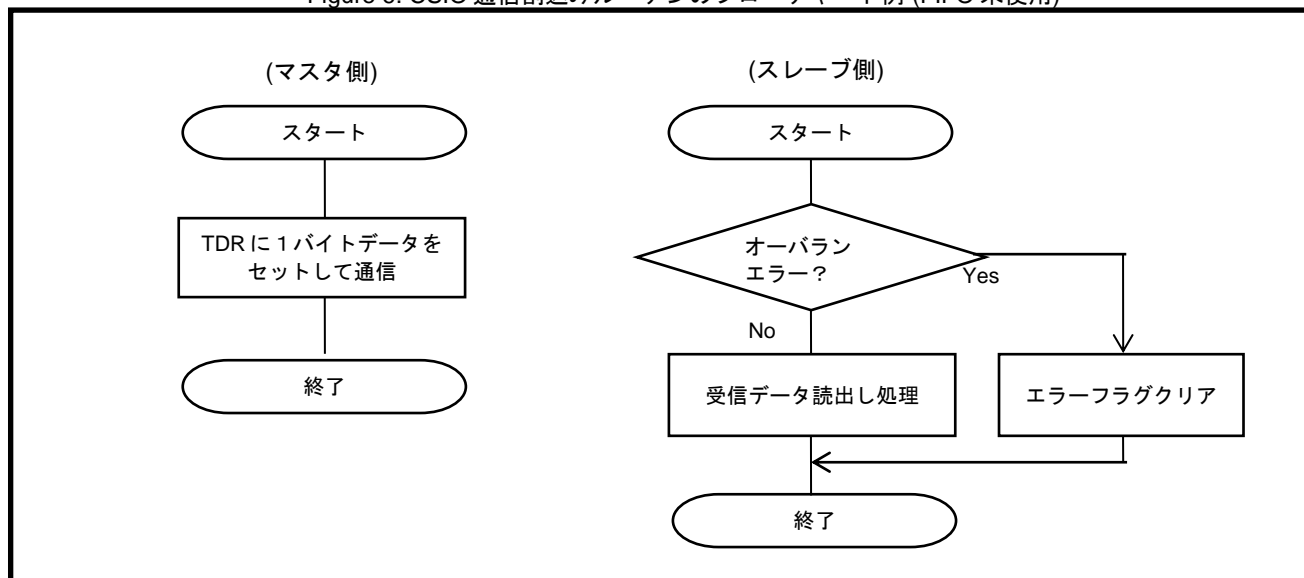


Figure 6. CSIO 通信割込みルーチンのフローチャート例 (FIFO 未使用)



### 3.2.1 ボーレートの設定

15 ビットリロードカウンタは、ボーレートジェネレータレジスタ(BGR0/1)で設定します。ボーレートの計算式を以下に示します。

#### ■ リロード値の計算式

$$V = (f/b) - 1$$

V: リロード値, f: バスクロック周波数, または外部クロック周波数(Hz), b: ボーレート(bps)

#### ■ リロード値の計算例

バスクロック周波数 36 MHz, ボーレートを 115200 bps に設定する場合のリロード値を以下に示します。

リロード値:

$$V = (36000000 / 115200) - 1 = 311$$

よって、ボーレートは、

$$b = (36000000 / (311+1)) = 115384 \text{ bps}$$

### 3.2.2 CSIO マスタ/スレーブ ポートの設定プログラム例

Code 6 に、CSIO マスタ/スレーブにおけるポートの設定プログラム例を示します。

Code 6. CSIO マスタ/スレーブのポート設定プログラム例 (S6J3110 シリーズの場合)

```

int main(void)
{
    /* Keycode register setting for SOT0_0 */
    PPC_KEYCDR=0x10000028;
    PPC_KEYCDR=0x50000028;
    PPC_KEYCDR=0x90000028;
    PPC_KEYCDR=0xD0000028;
    PPC_PCFGR020=0xC002;

    /* Keycode register setting for SIN1_0 */
    PPC_KEYCDR=0x10000038;
    PPC_KEYCDR=0x50000038;
    PPC_KEYCDR=0x90000038;
    PPC_KEYCDR=0xD0000038;
    PPC_PCFGR028=0x1000;

    /* Keycode register setting for SCK0_0 */
    PPC_KEYCDR=0x1000002A;
    PPC_KEYCDR=0x5000002A;
    PPC_KEYCDR=0x9000002A;
    PPC_KEYCDR=0xD000002A;
    PPC_PCFGR021=0x1002;

    /* Keycode register setting for AN2 analog input disable */
    ADER_KEYCDR=0x20000804;
    ADER_KEYCDR=0x60000804;
    ADER_KEYCDR=0xA0000804;
    ADER_KEYCDR=0xE0000804;
    ADER_ADER0=0xFFFFFFFFB;

    /* Keycode register setting for SCK1_0 */
    PPC_KEYCDR=0x10000040;
    PPC_KEYCDR=0x50000040;
    PPC_KEYCDR=0x90000040;
    PPC_KEYCDR=0xD0000040;
    PPC_PCFGR100=0x1002;

    /* Keycode register setting for SOT0_0 output */
    GPIO_KEYCDR=0x20000204;
    GPIO_KEYCDR=0x60000204;
    GPIO_KEYCDR=0xA0000204;
    GPIO_KEYCDR=0xE0000204;
    GPIO_DDR0=0x00100000;

    /* Keycode register setting for port input enable */
    GPIO_KEYCDR=0x20000400;
    GPIO_KEYCDR=0x60000400;
    GPIO_KEYCDR=0xA0000400;
    GPIO_KEYCDR=0xE0000400;
    GPIO_PORTEN=0x00000001;

    /* CSIO ch1 Rx */
    Synch_CSIO1_RX();

    /* CSIO ch0 Tx */
    Synch_CSIO0_TX();

    for(;;){ /* Endless loop */
        ClearWatchdog();
    }
}

```

PPC\_PCFGR, ADER\_ADER, GPIO\_DDR, GPIO\_PORTEN レジスタはキーコードレジスタの設定が必要

PPC\_PCFGR レジスタで SOT0\_0 を設定

PPC\_PCFGR レジスタで SIN1\_0 を設定

PPC\_PCFGR レジスタで SCK0\_0 を設定

ADER\_ADER0 レジスタで AN2 を設定

PPC\_PCFGR レジスタで SCK1\_0 を設定

GPIO\_DDR0 レジスタで SOT0\_0 を設定

GPIO\_PORTEN0 レジスタでポート入力許可

受信側初期設定, 受信処理開始

送信側初期設定, 送信処理開始

ポート設定

動作モード/フォーマット設定

### 3.2.3 CSIO マスタ側設定プログラム例

Code 7 と Code 8 に、CSIO マスタ側の設定プログラム例を示します。

Code 7. CSIO マスタ側設定プログラム例 (S6J3110 シリーズの場合)

```

/* Clock Synchronous Serial Interface (Transmission) */
void Synch_CSIO0_TX(void)
{
    /* CSIO0 status initialize */
    CPG_MFS00_CSIO_SCR_UPCL=1;

    /* Operation mode setting (Clock synchronous mode) */
    CPG_MFS00_CSIO_SMR=0x40;

    /* Enable serial clock output */
    CPG_MFS00_CSIO_SMR_SCKE=1;

    /* Enable serial data output */
    CPG_MFS00_CSIO_SMR_SOE=1;

    /* Baud rate setting (115.2kbps, 36MHz) */
    CPG_MFS00_CSIO_BGR=311;

    /* CSIO0 setting initialize (Master) */
    CPG_MFS00_CSIO_SCR=0x00;

    /* Data format setting (8bit length) */
    CPG_MFS00_CSIO_ESCR=0x00;

    /* Enable transmission interrupt */
    CPG_MFS00_CSIO_SCR_TIE=1;

    /* Enable transmission operation */
    CPG_MFS00_CSIO_SCR_TXE=1;
}

```

CSIO 送信側初期設定

動作モードの初期設定

ボーレート設定

データフォーマット設定

マスタ側割込み処理許可

マスタ側送信動作開始

Code 8. CSIO マスタ側割込みルーチン設定例 (S6J3110 シリーズの場合)

```

/* CSIO0 Transmission interrupt routine */
FN_IRQ_DEFINE_BEGIN(Uart0_Tx_Interrupt, INTERRUPTS_IRQ_NUMBER_65)
{
    /* Transmission data register setting */
    CPG_MFS00_CSIO_TDR=tr_data[tr_num];

    tr_num++;
    if(tr_num == 10){
        tr_num = 0;
    }
}
FN_IRQ_DEFINE_END()

```

割込みルーチン

1 バイトデータを TDR レジスタへ設定

### 3.2.4 CSIO スレーブ側設定プログラム例

Code 9 と Code 10 に、CSIO スレーブ側の設定プログラム例を示します。

Code 9. CSIO スレーブ側設定プログラム例(S6J3110 シリーズの場合)

```

/* Clock Synchronous Serial Interface (Reception) */
void Synch_CSIO1_RX(void)
{
    /* CSIO1 status initialize */
    CPG_MFS01_CSIO_SCR_UPCL=1;

    /* Operation mode setting (Clock synchronous mode) */
    CPG_MFS01_CSIO_SMR=0x40;

    /* Enable serial data output */
    CPG_MFS01_CSIO_SMR_SOE=1;

    /* CSIO1 setting initialize (Slave) */
    CPG_MFS01_CSIO_SCR=0x40;

    /* Data format setting (8bit length) */
    CPG_MFS01_CSIO_ESCR=0x00;

    /* Enable reception interrupt */
    CPG_MFS01_CSIO_SCR_RIE=1;

    /* Enable reception operation */
    CPG_MFS01_CSIO_SCR_RXE=1;
}

```

CSIO 受信側初期設定

動作モードの初期設定

データフォーマット設定

受信側割り込み処理許可

受信側動作開始

Code 10. CSIO スレーブ側割り込みルーチン設定例 (S6J3110 シリーズの場合)

{

割り込み  
ルーチン

```

/* CSIO1 Rx interrupt routine */
FN_IRQ_DEFINE_BEGIN(Uart1_Rx_Interrupt, INTERRUPTS_IRQ_NUMBER_66)
{
    /* Overrun error */
    if(CPG_MFS01_CSIO_SSR_ORE==1) {

        /* Reception error flag clear */
        CPG_MFS01_CSIO_SSR_REC=1;

    }
    else{
        /* Receive data register read */
        re_data[re_num]=CPG_MFS01_CSIO_RDR;

        /* Data count */
        re_num++;
        if(re_num == 10){
            re_num = 0;
        }
    }
}
FN_IRQ_DEFINE_END()

```

オーバラン  
エラーチェックと  
フラグクリア

受信データ読出し

### 3.3 LIN 通信の設定

Figure 7 に LIN 通信の接続例を示します。

Figure 7. LIN 通信の接続例

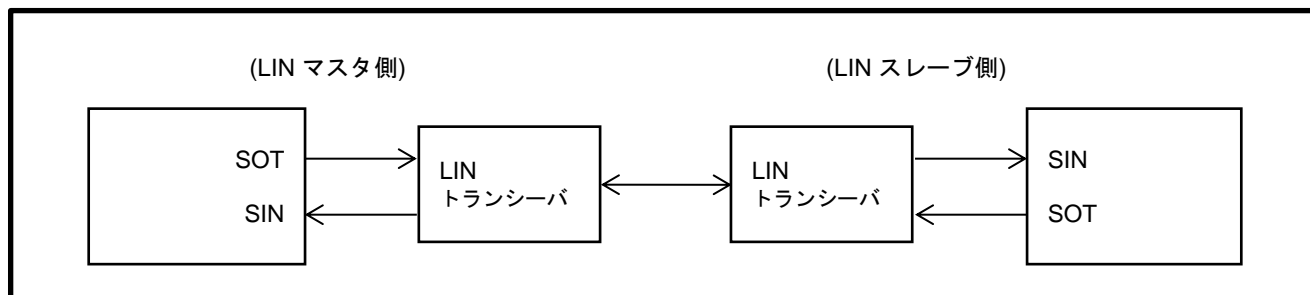
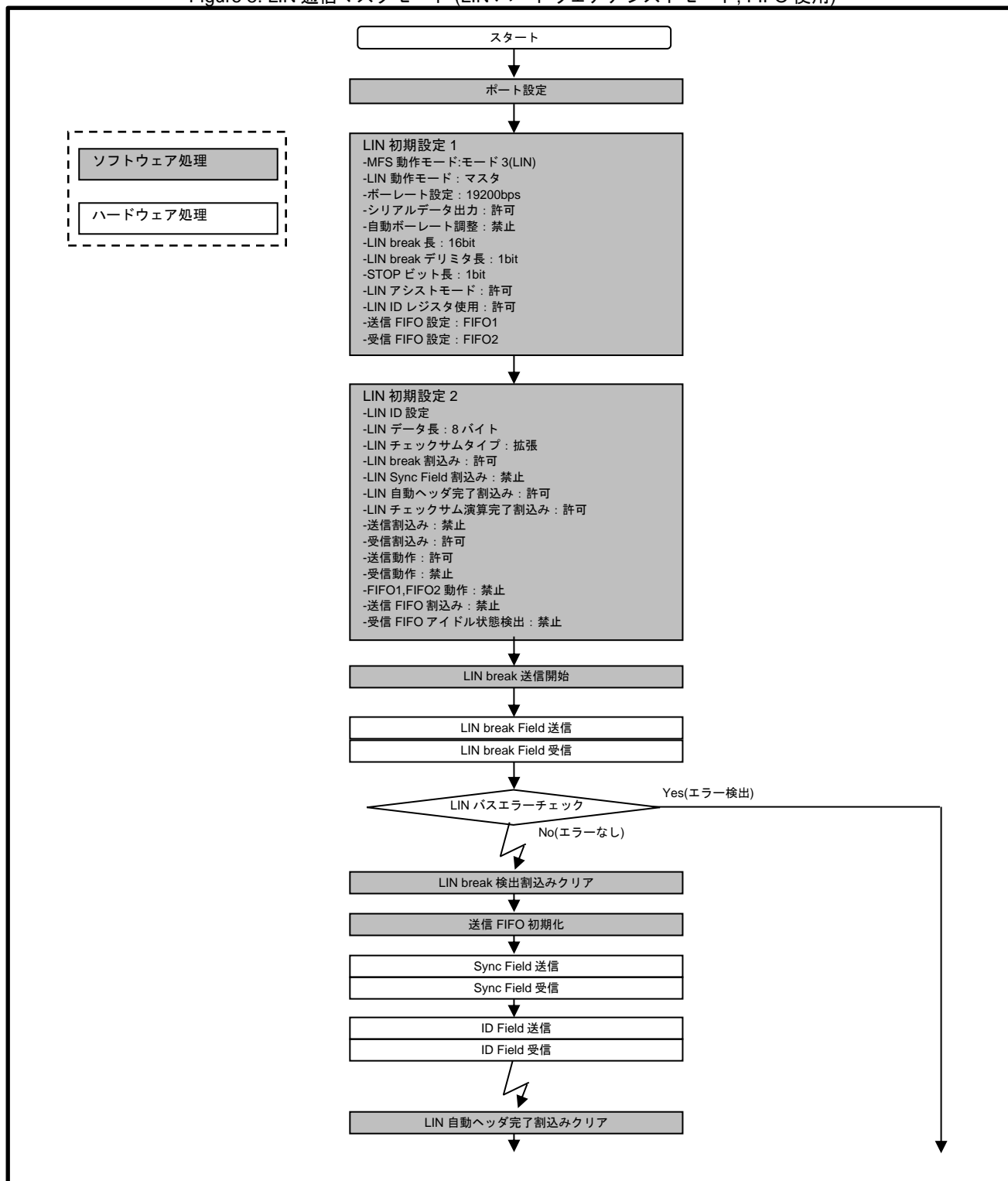


Figure 8 と Figure 9 に LIN 通信のフローチャート例を示します。

Figure 8. LIN 通信マスタモード (LIN ハードウェアアシストモード, FIFO 使用)



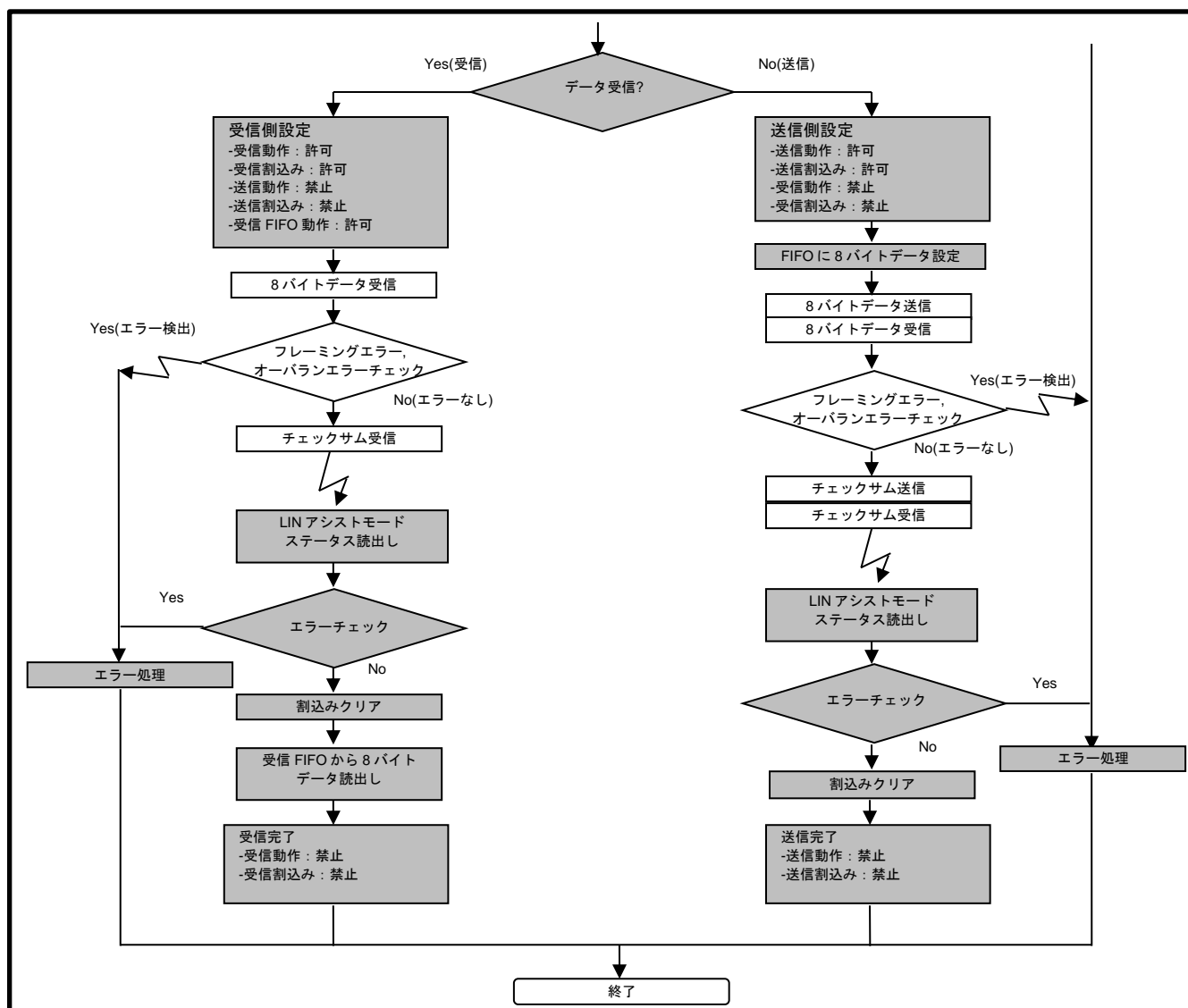
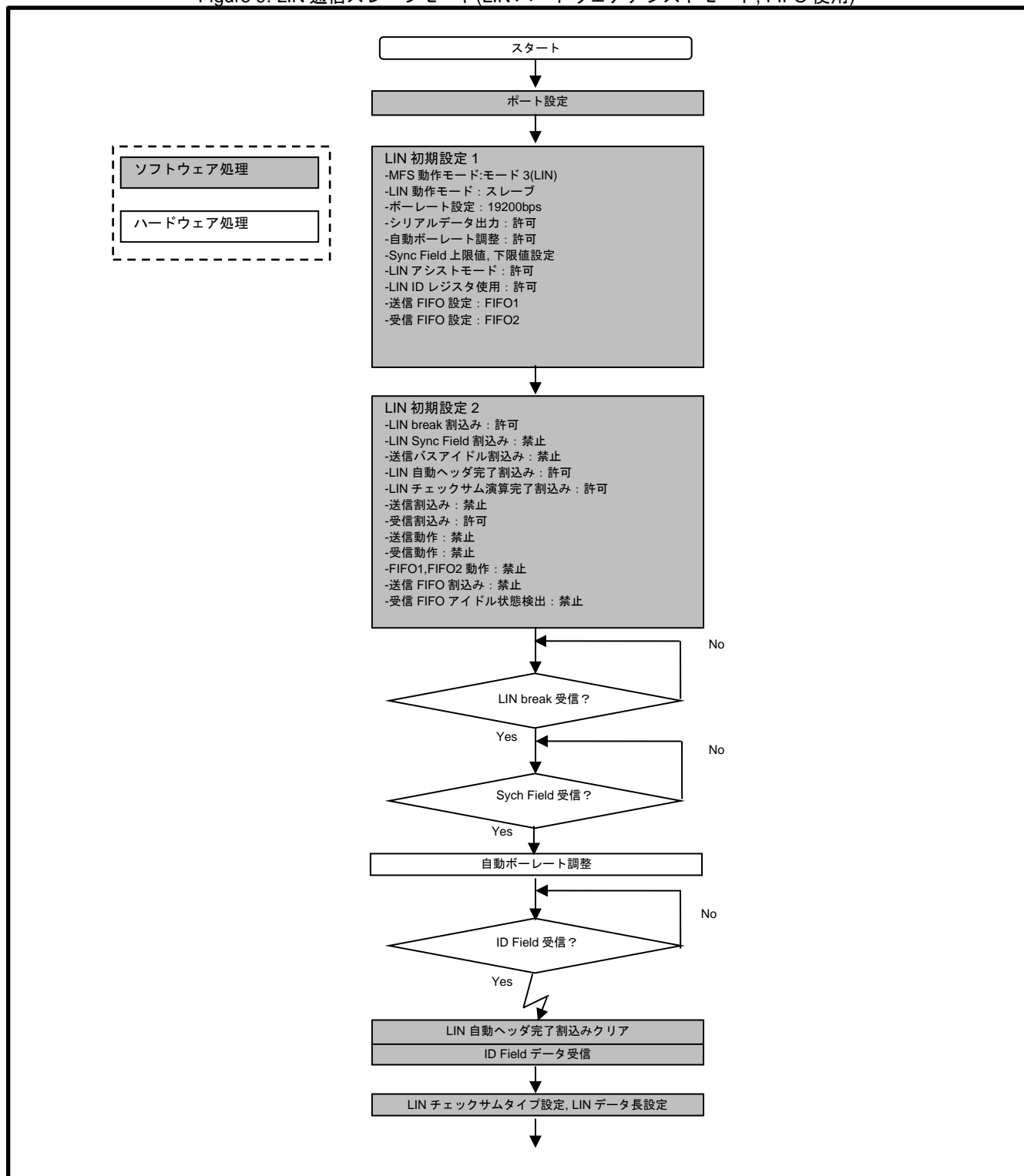
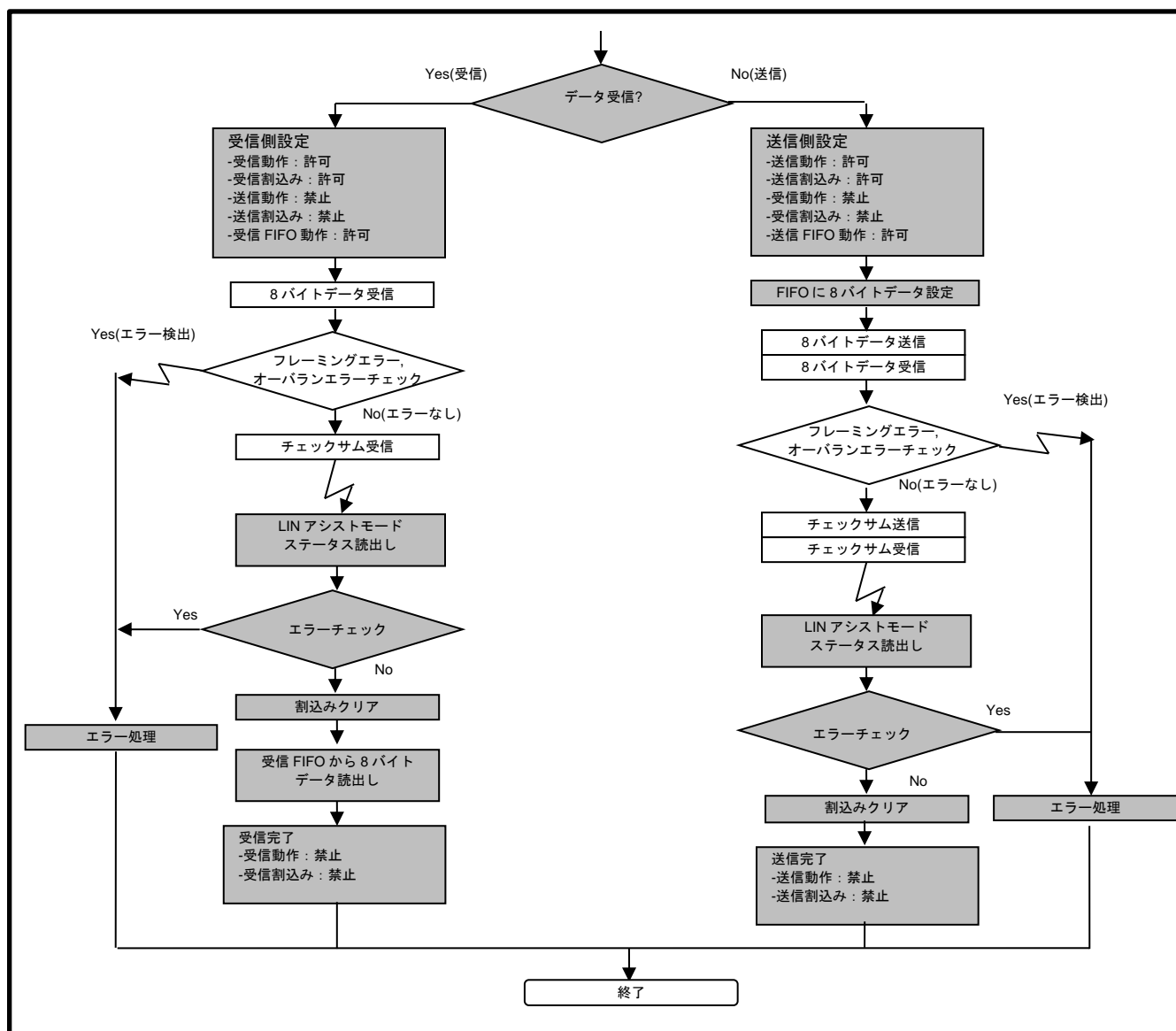




Figure 9. LIN 通信スレープモード(LIN ハードウェアアシストモード, FIFO 使用)





### 3.3.1 ボーレートの設定

15 ビットリロードカウンタは、ボーレートジェネレータレジスタ (BGR0/1) で設定します。ボーレートの計算式を以下に示します。

#### ■ リロード値の計算式

$$V = (f/b) - 1$$

V: リロード値, f: バスクロック周波数, または外部クロック周波数 (Hz), b: ボーレート (bps)

#### ■ リロード値の計算例

バスクロック周波数 36 MHz, ボーレートを 19200 bps に設定する場合のリロード値を以下に示します。

リロード値:

$$V = (36000000 / 19200) - 1 = 1874$$

よって、ボーレートは、

$$b = (36000000 / (1874 + 1)) = 19200 \text{ bps}$$

### 3.3.2 LIN マスタ/スレーブのポート設定プログラム例

Code 11 に、LIN マスタ/スレーブにおけるポート設定のプログラム例を示します。

Code 11. LIN マスタ/スレーブのポート設定プログラム例(S6J3110 シリーズの場合)

```

/* Initialize port setting */
static void SampleLinPortInit(void)
{
    stc_port_pin_config_t stcPinConf = { PortOutputResourceGPIO };

    /* SIN3 setting for master side */
    stcPinConf.bInputEnable = TRUE; /* Enable input */
    stcPinConf.bNoiseFilterEnable = FALSE;
    stcPinConf.enGpioDirection = PortGpioInput;
    stcPinConf.enGpioInitOutputLevel = PortGpioHigh;
    stcPinConf.enInputLevel = PortInputLevelCmosA;
    stcPinConf.enOutputDrive = PortOutputDriveA;
    stcPinConf.enOutputFunction = PortOutputResourceGPIO;
    stcPinConf.enPullResistor = PortPullResistorNone;
    (void)Port_SetPinConfig(0, 5, &stcPinConf);

    /* SIN2_1 setting for slave side */
    (void)Port_SelectInputPort(6, PortInputPortB); /*Select input port SIN2 1*/
    (void)Port_SetPinConfig(4, 21, &stcPinConf);

    /* SOT3 setting for master side */
    stcPinConf.bInputEnable = FALSE;
    stcPinConf.enGpioDirection = PortGpioOutput;
    stcPinConf.enOutputFunction = PortOutputResourceC;
    (void)Port_SetPinConfig(0, 6, &stcPinConf);

    /* SOT2_1 setting for slave side */
    stcPinConf.enOutputFunction=PortOutputResourceD; /*Select output port SOT2 1*/
    (void)Port_SetPinConfig(0, 0, &stcPinConf);

    /* For LIN transceiver */
    /* P007 = LIN NSLP1 */
    stcPinConf.bInputEnable = FALSE;
    stcPinConf.enGpioDirection = PortGpioOutput;
    stcPinConf.enOutputFunction = PortOutputResourceGPIO;
    (void)Port_SetPinConfig(0, 7, &stcPinConf);
    /* P001 = LIN NSLP2 */
    (void)Port_SetPinConfig(0, 1, &stcPinConf);

    (void)Port_EnableInput();
}

```

PPC\_PCFGR レジスタで SIN3\_0 を設定

PPC\_PCFGR レジスタで SIN2\_1 を設定

PPC\_PCFGR レジスタで SOT3\_0 を設定

PPC\_PCFGR レジスタで SOT2\_1 を設定

LIN トランシーバ NSLP1 のために P007 を設定

LIN トランシーバ NSLP2 のために P001 を設定

GPIO\_PORTEN0 レジスタでポート入力許可設定

### 3.3.3 LIN マスタ側の初期化設定プログラム例

Code 12 に、LIN マスタ側における初期化設定プログラム例を示します。

Code 12. LIN マスタ側の初期化設定プログラム例(S6J3110 シリーズの場合)

```

/* Initialize LIN setting */
static en_result_t SampleLinInit(void)
{
    /* Initialize LIN ch 3 (Master) */
    stc_lin_config_t stcLinConfig;
    /* Master mode */
    stcLinConfig.bMasterMode = TRUE;
    /* Break length=16bits*/
    stcLinConfig.enBreakFieldLength
        = LinBreakFieldLength16bits;
    /*Break delimiter length = 1bit */
    stcLinConfig.enBreakDelimiterLength
        = LinBreakDelimiterLength1bits;
    /* Stop bit = 1bit */
    stcLinConfig.enStopBit = LinOneStopBit;
    /* Baud rate = 19200bps */
    stcLinConfig.u32DataRate = 19200;
    /* Status callback */
    stcLinConfig.pfnStatusCb = SampleStatusCallBack_ch3;
    enResult = Lin_Init(&CPG_MFS03_LIN, &stcLinConfig);
}

```

LIN channel 3 をマスタ側に設定

LIN sync break 長の設定

LIN stop ビット長の設定

ボーレート設定

動作モード, LIN アシストモード  
設定

割込み許可設定 (受信エラー,  
自動ヘッダ, チェックサム完了,  
LIN break 検出)

### 3.3.4 LIN スレーブ側の初期化設定プログラム例

Code 13 に、LIN スレーブ側の初期化設定プログラム例を示します。

Code 13. LIN スレーブ側の初期化設定プログラム例 (S6J3110 シリーズの場合)

```
/* Initialize LIN setting */
static en_result_t SampleLinInit(void)
{
    /* Initialize LIN ch 2 (Slave) */
    stcLinConfig.bMasterMode = FALSE; /* Slave mode */
    /* Auto adjustment upper limit (maximum value) */
    stcLinConfig.ul6BgrUpperLimit = 0x7FFF;
    /* Auto adjustment lower limit (minimum value) */
    stcLinConfig.ul6BgrLowerLimit = 0x3;
    /* Status callback */
    stcLinConfig.pfnStatusCb = SampleStatusCallBack_ch2;
    /* Baud rate = 19200bps */
    stcLinConfig.u32DataRate = 19200;
    enResult = Lin_Init(&CPG_MFS02_LIN, &stcLinConfig);
}
```

LIN channel 2 をスレーブ側に設定  
(LIN sync break 長, stop ビット, ボーレート設定はマスタ側と同じ設定)

動作モード, LIN アシストモード, 割込み設定はマスタ側と同じ

### 3.3.5 LIN マスタ/スレーブ動作の設定プログラム例

Code 14 に、LIN マスタ/スレーブ動作の設定プログラム例を示します。

Code 14. LIN マスタ/スレーブ動作の設定プログラム例(S6J3110 シリーズの場合)

```

int main(void)
{
    /* Initialize port setting */
    (void) SampleLinPortInit();

    /* Initialize LIN */
    enResult = SampleLinInit();
    if (enResult == Ok)
    {
        SampleLinCh2Status = NoOperation;
        SampleLinCh3Status = NoOperation;

        /* Send Master to Slave */
        /* Start auto header */
        (void) Lin_SetAutoHeader(&CPG_MFS03_LIN, SAMPLE_ID_MASTER_TO_SLAVE,
                                SAMPLE_DATA_LENGTH, LinChecksumTypeExtended);

        /* Wait until auto header complete */
        while (SampleLinCh2Status != CompleteAutoHeader);

        /* Get ID */
        (void) Lin_GetReceivedId(&CPG_MFS02_LIN, &u8LinCh2ReceivedId,
                                &u8LinCh2ReceivedParity);

        /* Enable reception */
        (void) Lin_EnableRx(&CPG_MFS02_LIN, SAMPLE_DATA_LENGTH,
                            LinChecksumTypeExtended);

        /* Write Data Master to slave */
        (void) Lin_WriteData(&CPG_MFS03_LIN, au8LinCh3TxData, SAMPLE_DATA_LENGTH,
                            LinChecksumTypeExtended);

        /* Wait until checksum calculation complete */
        while (SampleLinCh2Status != CompleteChecksum);

        /* Read response data */
        (void) Lin_ReadData(&CPG_MFS02_LIN, au8LinCh2RxBuf,
                            &u8LinCh2ReceivedLength);

        /* Finally disable Rx */
        (void) Lin_DisableRx(&CPG_MFS02_LIN);

        /* Finally disable Tx */
        (void) Lin_DisableTx(&CPG_MFS03_LIN);

        /* Wait time for next frame (usually done by scheduler) */
        for (uint32_t u32Cnt = 0; u32Cnt < 500000; u32Cnt++)
        {
            NOP();
        }

        /* Endless loop */
        for (;;)
        {
            ClearWatchdog();
        }
    }
}

```

マスタ側ポート設定 (SIN3\_0, SOT3\_0)とスレーブ側  
ポート設定 (SIN2\_1, SOT2\_1)

LIN ヘッダとボーレート設定

マスタ側 LIN ID, データ長,  
チェックサムタイプ設定

スレーブ側  
自動ヘッダ受信待ち

スレーブ側 LIN ID,  
パリティビット受信

スレーブ側受信許可,  
FIFO 許可

マスタ側送信データ設定,  
送信許可

スレーブ側チェックサム演算完了  
チェック

スレーブ側データ受信

スレーブ側受信禁止

マスタ側送信禁止

次のフレーム待ち

### 3.3.6 LIN マスタ/スレーブ割込み処理の設定プログラム例

Code 15 と Code 16 に、LIN 割込み処理の設定プログラム例を示します。

Code 15. LIN マスタ側の割込み処理設定プログラム例(S6J3110 シリーズの場合)

```

/* Status callback function for LIN ch3 (master) */
static void SampleStatusCallBack_ch3(un_lin_interrupt_trigger_t unIntTrigger,
                                     un_lin_detected_error_t   unDetectedError)
{
    /* Detected Error */
    if (unIntTrigger.stcBits.DetectError != 0)
    {
        /* Implement error handling code here */
        /* Save error flag */
        unLinCh3Error.u8Byte = unDetectedError.u8Byte;
        SampleLinCh2Status = DetectError;
    }

    /* Checksum complete */
    else if(unIntTrigger.stcBits.CompleteChecksum != 0)
    {
        SampleLinCh2Status = CompleteChecksum;
    }

    /* Header complete */
    else if(unIntTrigger.stcBits.CompleteAutoHeader != 0)
    {
        SampleLinCh2Status = CompleteAutoHeader;
    }

    /* Break is detected */
    else if(unIntTrigger.stcBits.DetectBreak != 0)
    {
        SampleLinCh2Status = DetectBreak;
    }
    else
    {
        /* do nothing */
    }
}

```

マスタ側割込み処理コールバック関数

フレーミングエラー, オーバーランエラー割込みの場合

チェックサム演算完了割込みの場合

自動ヘッダ完了割込みの場合

LIN sync break 検出割込みの場合

Code 16. LIN スレーブ側の割込み処理の設定プログラム例(S6J3110 シリーズの場合)

```

/* Status callback function for LIN ch2 (slave) */
static void SampleStatusCallBack_ch2(un_lin_interrupt_trigger_t unIntTrigger,
                                     un_lin_detected_error_t   unDetectedError)
{
    /* Detected Error */
    if (unIntTrigger.stcBits.DetectError != 0)
    {
        /* Implement error handling code here */
        /* Save error flag */
        unLinCh2Error.u8Byte = unDetectedError.u8Byte;
        SampleLinCh2Status = DetectError;
    }
    /* Checksum complete */
    else if(unIntTrigger.stcBits.CompleteChecksum != 0)
    {
        SampleLinCh2Status = CompleteChecksum;
    }
    /* Header complete */
    else if(unIntTrigger.stcBits.CompleteAutoHeader != 0)
    {
        SampleLinCh2Status = CompleteAutoHeader;
    }
    /* Break is detected */
    else if(unIntTrigger.stcBits.DetectBreak != 0)
    {
        SampleLinCh2Status = DetectBreak;
    }
    else
    {
        /* do nothing */
    }
}

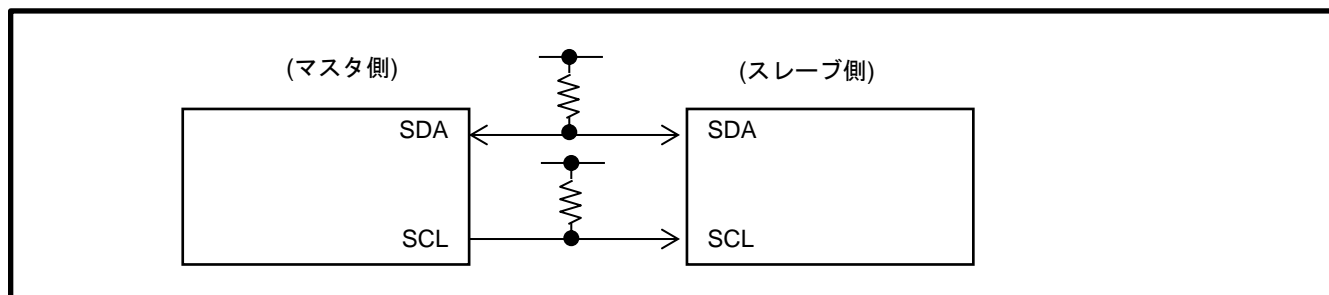
```

スレーブ側割込み処理コールバック関数

割込み処理の振分けは、マスタ側と同じ

### 3.4 I<sup>2</sup>C 通信の設定

Figure 10 に I<sup>2</sup>C 通信の接続例を示します。

Figure 10. I<sup>2</sup>C 通信接続例

Figure 11, Figure 12 と Figure 13 に、I<sup>2</sup>C 通信のフローチャート例を示します。

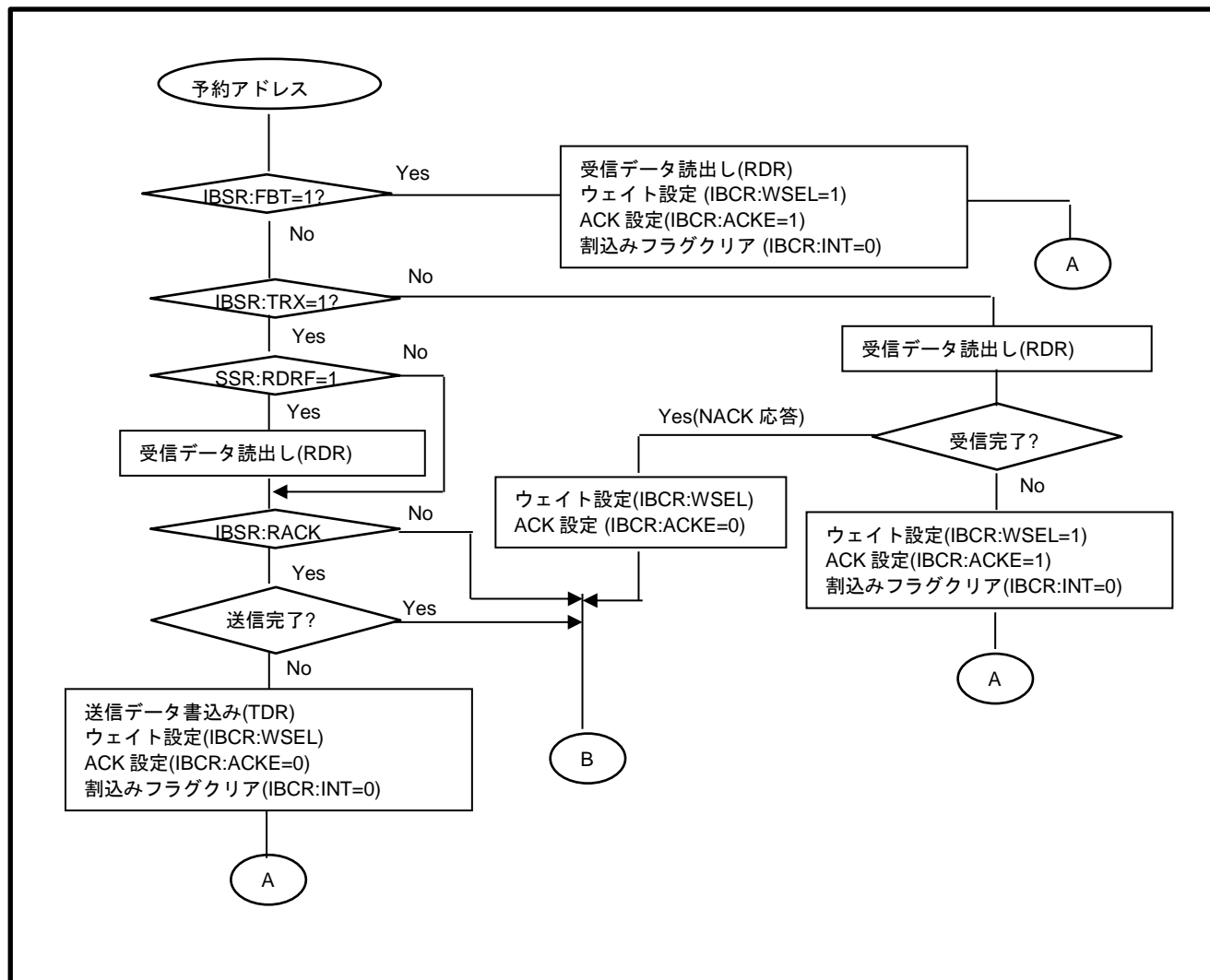


```

graph TD
    Start([スタート]) --> Init[<初期設定>  
ポート設定  
ボーレート設定 (BGR)  
スレーブアドレス設定 (ISBA)  
スレーブマスク設定 (ISMK)  
I²C 許可 (ISMK:EN=1)]
    Init --> Master{マスタ?}
    Master -- No --> BusError[バスエラー処理]
    BusError --> End1([終了])
    Master -- Yes --> TDR1[送信データ書込み (TDR)  
マスタ設定 (IBCR:MSS=1)]
    TDR1 --> Int1{IBCR:INT=1?}
    Int1 -- No --> A((A))
    Int1 -- Yes --> BER{IBCR:BER=0?}
    BER -- No --> A
    BER -- Yes --> AL{IBSR:AL=0?}
    AL -- No --> A
    AL -- Yes --> MSS{IBCR:MSS=1?}
    MSS -- No --> Sleep([スレープ])
    MSS -- Yes --> RSA{IBSR:RSA=0?}
    RSA -- No --> Sleep
    RSA -- Yes --> RACK{IBSR:RACK=0?}
    RACK -- No --> B((B))
    RACK -- Yes --> TRX{IBSR:TRX=1?}
    TRX -- No --> B
    TRX -- Yes --> Done1{送信完了?}
    Done1 -- No --> TDR2[送信データ書込み (TDR)  
ウェイト設定 (IBCR:WSEL)  
ACK 設定 (IBCR:ACKE)  
割込みフラグクリア (IBCR:INT=0)]
    Done1 -- Yes --> B
    TDR2 --> Repeat{反復スタート?}
    Repeat -- No --> Stop[ストップ設定 (IBCR:MSS=0)  
ACK 設定 (IBCR:ACKE)  
割込みフラグクリア (IBCR:INT=0)]
    Stop --> End2([終了])
    Repeat -- Yes --> TDR3[送信データ書込み (TDR)  
反復スタート設定 (IBCR:MSS=SCC=1)  
ACK 設定 (IBCR:ACKE)  
割込みフラグクリア (IBCR:INT=0)]
    TDR3 --> Int1
    A --> Int1
    B --> FBT{IBSR:FBT=0?}
    FBT -- No --> RDR[受信データ読出し (RDR)]
    RDR --> Done2{受信完了?}
    Done2 -- No --> FBT
    Done2 -- Yes --> NACK[Yes (NACK 応答)]
    NACK --> TDR2
    FBT -- Yes --> Done2
  
```

```

graph TD
    Start([スレーブ]) --> D1{IBSR:RSA=0?}
    D1 -- No --> D2{IBSR:TRX=0?}
    D1 -- Yes --> D2
    D2 -- No --> D3{IBSR:RACK=0?}
    D2 -- Yes --> D3
    D3 -- No --> D4{IBSR:INT=0?}
    D3 -- Yes --> P1[送信データ書込み (TDR)  
ウェイト設定 (IBCR:WSEL)  
割込みフラグクリア (IBCR:INT=0)]
    D4 -- No --> End1([終了])
    D4 -- Yes --> P1
    P1 --> A1((A))
    A1 --> D5{IBSR:FBT=0?}
    D5 -- No --> D6{IBSR:TRX=1?}
    D5 -- Yes --> P2[受信データ読出し (RDR)]
    P2 --> D6
    D6 -- No --> P3[ACK 設定 (IBCR:ACKE=0)  
割込みフラグクリア (IBCR:INT=0)]
    D6 -- Yes --> P4[送信データ書込み (TDR)  
ウェイト設定 (IBCR:WSEL)  
ACK 設定 (IBCR:ACKE=0)  
割込みフラグクリア (IBCR:INT=0)]
    P3 --> End2([終了])
    P4 --> A2((A))
    A2 --> D7{IBSR:FBT=1?}
    D7 -- No --> P5[受信データ読出し (RDR)]
    D7 -- Yes --> P6[ウェイト設定 (IBCR:WSEL)  
ACK 設定 (IBCR:ACKE=1)  
割込みフラグクリア (IBCR:INT=0)]
    P5 --> P6
    P6 --> A2
  
```

Figure 13. I<sup>2</sup>C 通信フローチャート例(DMA, FIFO 未使用) (3 / 3)


### 3.4.1 ボーレート設定

15 ビットリロードカウンタは、ボーレートジェネレータレジスタ(BGR0/1)で設定します。ボーレートの計算式を以下に示します。

#### ■ リロード値の計算式

$$V = (f/b) - 1$$

V: リロード値, f: バスクロック周波数, または外部クロック周波数(Hz), b: ボーレート(bps)

#### ■ リロード値の計算例

バスクロック 16 MHz, ボーレートを 100 kbps に設定する場合のリロード値を以下に示します。

リロード値:

$$V = (16000000 / 100000) - 1 = 159$$

よって、ボーレートは、

$$b = (16000000 / (159+1)) = 100 \text{ kbps}$$

### 3.4.2 I<sup>2</sup>C ポートの設定プログラム例

Code 17 は、I<sup>2</sup>C ポートの設定プログラム例を示します。

Code 17. I<sup>2</sup>C ポート設定プログラム例(S6J3110 シリーズの場合)

```

/* Port settings */
PpcConf.bInputEnable = TRUE;
PpcConf.bNoiseFilterEnable = FALSE;
PpcConf.enGpioDirection = PortGpioOutput;
PpcConf.enInputLevel = PortInputLevelCmosA;
PpcConf.enOutputDrive = PortOutputDriveA;
PpcConf.enOutputFunction = PortOutputResourceG;
PpcConf.enPullResistor = PortPullResistorNone;
PpcConf.enGpioInitOutputLevel = PortGpioLow;

/* SDA0_0 (P020) */
Port_SetPinConfig(0, 20, &PpcConf);
/* SCL0_0 (P021) */
Port_SetPinConfig(0, 21, &PpcConf);

```

PPC\_PCFGR レジスタで  
SDA0\_0, SCL0\_0 を設定

### 3.4.3 I<sup>2</sup>C マスタ側の初期化設定プログラム例

Code 18 に、I<sup>2</sup>C マスタ側の初期化設定プログラム例を示します。

Code 18. I<sup>2</sup>C マスタ側初期化設定プログラム例(S6J3110 シリーズの場合)

```

/* I2C settings */
stc_i2c_config_t stcI2cConfig = {
    .u32DataRate          = 100000,
    .u8SlaveAddress       = I2C_SLAVE_ADDR_STD,
    .bSlaveAddressEnable  = TRUE,
    .bAcknowledgeEnable   = FALSE,
    .bWaitSelect          = FALSE,
    .bInterruptEnable     = FALSE,
    .bDmaModeEnable       = FALSE,
};

```

I<sup>2</sup>C ボーレート設定、および  
スレーブアドレス設定

```

/* I2C initialize */
I2c_Init(&CPG_MFS00_I2C, &stcI2cConfig);

```

```

/* write transmission buffer 'write_buf[]' */
writeBuf[0] = 0x11;
writeBuf[1] = 0x22;
writeBuf[2] = 0x33;
writeBuf[3] = 0x44;

```

送信データ書込み

```

/* Write I2C */
I2c_WriteSequence(&CPG_MFS00_I2C, I2C_SLAVE_ADDR_STD, 0xAA, writeBuf, 4);

```

### 3.4.4 I<sup>2</sup>C スレーブ側の初期化設定プログラム例

Code 19 に、I<sup>2</sup>C スレーブ側の初期化設定プログラム例を示します。

Code 19. I<sup>2</sup>C スレーブ側初期化設定プログラム例(S6J3110 シリーズの場合)

```

/* I2C settings */
stc_i2c_config_t stcI2cConfig = {
    .u32DataRate          = 100000,
    .u8SlaveAddress       = I2C_SLAVE_ADDR_STD,
    .bSlaveAddressEnable  = TRUE,
    .bAcknowledgeEnable   = TRUE,
    .bWaitSelect          = FALSE,
    .bInterruptEnable     = FALSE,
    .bDmaModeEnable       = FALSE,
};

/* I2C initialize */
I2c_Init(&CPG_MFS00_I2C, &stcI2cConfig);

/* Slave address */
CPG_MFS00_I2C_ISBA_SA = I2C_SLAVE_ADDR_STD >> 1;

/* Enable I2C */
CPG_MFS00_I2C_ISMK_EN = 1;
  
```

I<sup>2</sup>C ボーレート,  
スレーブアドレス, ACK 設定

I<sup>2</sup>C 通信を許可

### 3.4.5 I<sup>2</sup>C マスタ/スレーブ動作の設定プログラム例

Code 20 に、I<sup>2</sup>C マスタ/スレーブ動作の設定プログラム例を示します。

Code 20. I<sup>2</sup>C マスタ/スレーブ動作設定プログラム例(S6J3110 シリーズの場合)

```
int main(void)
{
    /* SDA0_0, SCL0_0 Port settings */

    /* I2C initial settings (Master or Slave) */

    /* Endless loop */
    for (;;)
    {
        #if M_S_SELECTION == SLAVE
            if (CPG_MFS00_I2C_IBCR_INT == 1)
            {
                /* Check for Reserved start address is detected */
                if ((CPG_MFS00_I2C_IBSR_RSA == 1) ||
                    (CPG_MFS00_I2C_IBSR_TRX == 1))
                {
                    /* clear transfer end interrupt flag */
                    CPG_MFS00_I2C_IBCRC_INTC = 1;
                }
                else
                {
                    /* Read I2C */
                    if (readCnt < READ_COUNT_MAX)
                    {
                        I2c_Read(&CPG_MFS00_I2C, &readValue);

                        readBuf[readCnt] = readValue;
                        readCnt++;
                    }
                }
            }
        #endif
        ClearWatchdog();
    }
}
```

3.4.2 を参照

3.4.3, 3.4.4 を参照

スレーブ側のみ設定

データ送信又はデータアドレス検出

データ受信

## 4 要約

マルチファンクションシリアルインタフェースは、自動車アプリケーションで主に使用されるシリアル通信機能をサポートしています。この機能によって、デバイスとフレキシブルなネットワーク間での双方向通信が実現できます。サイプレスの S6J3110/3120/3200/3300/3350/3360/3370/3400 シリーズは、マルチファンクションシリアルインタフェースを搭載しています。このアプリケーションノートは、S6J3110/3120/3200/3300/3350/3360/3370/3400 シリーズのマルチファンクションシリアルインタフェースのスムーズな実装を支援します。

## 5 関連ドキュメント

Traveo ファミリのデータシートとハードウェアマニュアル:

- [S6J311E/D/C/B Series Datasheet \(Doc. No.002-05681\)](#)
- [S6J311A/9/8 Series Datasheet \(Doc. No.002-04632\)](#)
- [S6J3110 Series Hardware Manual \(Doc.No.002-10667\)](#)
- [S6J3120 Series Datasheet \(Doc.No.002-04863\)](#)
- [S6J3120 Series Hardware Manual \(Doc.No.002-04855\)](#)
- [S6J3200 Series Datasheet \(Doc.No.002-05682\)](#)
- [S6J3200 Series Hardware Manual \(Doc.No.002-04852\)](#)
- [S6J32E/F/G Series Datasheet \(Doc.No.002-10689\)](#)
- [S6J32E/F/G Series Hardware Manual \(Doc.No.002-12500\)](#)
- [Traveo Family Hardware Manual Platform Part for S6J3200 Series \(Doc.No.002-04854\)](#)
- [S6J3310/20/30/40 Series Datasheet \(Doc.No.002-10635\)](#)
- [S6J3350 Series Datasheet \(Doc.No.002-10634\)](#)
- [S6J3310/20/30/40/50 Series Hardware Manual \(Doc.No.002-10185\)](#)
- [TraveoFamily HardwareManual Platform Part for S6J3310/3320/3330/3340/3350 Series \(Doc.No.002-07884\)](#)
- [S6J3360/70 Series Datasheet \(Doc.No.002-03359\)](#)
- [S6J3360/70 Series Hardware Manual \(Doc.No.002-18302\)](#)
- [Traveo Family HardwareManual Platform Part for S6J3360/3370 Series \(Doc.No.002-07884\)](#)
- [S6J3400 Series Datasheet \(Doc.No.001-97829\)](#)
- [S6J3400 Series Hardware Manual \(Doc.No.002-09919\)](#)
- [Traveo Family Hardware Manual Platform Part for S6J3400 Series \(Doc.No.002-07884\)](#)

## 6 改訂履歴

文書名: AN202495 - Traveo™ファミリのマルチファンクションシリアルインタフェース使用方法

文書番号: 002-15746

Revision	ECN	変更者	発行日	変更内容
**	5438833	MKAR	09/21/2016	Initial Release これは英語版 002-02495 Rev.*B を翻訳した日本語版 002-15746 Rev.**です。
*A	5897930	MKAR	09/27/2017	これは英語版 002-02495 Rev.*C を翻訳した日本語版 002-15746 Rev.*A です。



## セールス、ソリューションおよび法律情報

### ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、[サイプレスのロケーション ページ](#)をご覧ください。

#### 製品

ARM® Cortex® Microcontrollers	<a href="http://cypress.com/arm">cypress.com/arm</a>
車載用	<a href="http://cypress.com/automotive">cypress.com/automotive</a>
クロック&バッファ	<a href="http://cypress.com/clocks">cypress.com/clocks</a>
インターフェース	<a href="http://cypress.com/interface">cypress.com/interface</a>
IoT (モノのインターネット)	<a href="http://cypress.com/iot">cypress.com/iot</a>
メモリ	<a href="http://cypress.com/memory">cypress.com/memory</a>
マイクロコントローラ	<a href="http://cypress.com/mcu">cypress.com/mcu</a>
PSoC	<a href="http://cypress.com/psoc">cypress.com/psoc</a>
電源用 IC	<a href="http://cypress.com/pmic">cypress.com/pmic</a>
タッチ センシング	<a href="http://cypress.com/touch">cypress.com/touch</a>
USB コントローラー	<a href="http://cypress.com/usb">cypress.com/usb</a>
ワイヤレス/RF	<a href="http://cypress.com/wireless">cypress.com/wireless</a>

#### PSoC® ソリューション

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6](#)

#### サイプレス開発者コミュニティ

[フォーラム](#) | [WICED IOT Forums](#) | [Projects](#) | [ビデオ](#) | [ブログ](#) | [トレーニング](#) | [Components](#)

#### テクニカルサポート

[cypress.com/support](http://cypress.com/support)

ARM and Cortex are the registered trademarks of ARM Limited in the EU and other countries.



Cypress Semiconductor  
 198 Champion Court  
 San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2015-2017. 本書面は、Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。) を含む) は、アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し、本段落で特に記載されているものを除き、その特許権、著作権、商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず、かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合、Cypress は、(1) 本ソフトウェアの著作権に基づき、(a) ソースコード形式で提供されている本ソフトウェアについて、Cypress ハードウェア製品と共に用いるためにのみ、かつ組織内部でのみ、本ソフトウェアの修正及び複製を行うこと、並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ、(直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリコード形式で外部エンドユーザーに配布すること、並びに (2) 本ソフトウェア (Cypress (Cypress により提供され、修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき、Cypress ハードウェア製品と共に用いるためにのみ、本ソフトウェアの作成、利用、配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用、複製、修正、変換又はコンパイルを禁止する。

**適用される法律により許される範囲内で、Cypress は、本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても、明示又は黙示を問わず、いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。**適用される法律により許される範囲内で、Cypress は、別途通知することなく、本書面を変更する権利を留保する。Cypress は、本書面に記載のある、いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は、参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計、プログラム、かつテストすることは、本書面のユーザーの責任において行われるものとする。Cypress 製品は、兵器、兵器システム、原子力施設、生命維持装置若しくは生命維持システム、蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム、汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用、又は装置若しくはシステムの不具合が人身傷害、死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。) のためには設計、意図又は承認されていない。重要な構成部分とは、その不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分を含む。Cypress 製品のあらゆる本目的外使用から生じ、若しくは本目的外使用に関連するいかなる請求、損害又はその他の責任についても、Cypress はその全部又は一部を問わず一切の責任を負わず、かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の目的外使用から生じ又は本目的外使用に関連するあらゆる請求、費用、損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, Capsense, EZ-USB, F-RAM, 及び Traveo は、米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは、[cypress.com](http://cypress.com) を参照すること。その他の名称及びブランドは、それぞれの権利者の財産として権利主張がなされている可能性がある。