R8C/10 グループ

汎用ポートによる I²C-BUS 制御プログラム

R01AN1272JJ0110
Rev.1.10
2012.06.01

1. 要約

ソフトウエアによる I²C-BUS 制御のプログラムと応用例です。EEPROM の制御などに応用できます。R8C/10 グループと同様の SFR(周辺装置制御レジスタ)を持つ他の R8C ファミリでも本プログラムを使用することができます。ただし、一部の機能を機能追加等で変更している場合がありますのでマニュアルで確認してください。このアプリケーションノートをご使用に際しては十分な評価を行ってください。

発振安定待ちに関する注意事項
Page 16 of 27 の init 関数において、高速オンチップオシレータの発振を開始させた後、発振安定時間を待ってから、高速オンチップオシレータを選択してください。

2. はじめに

ソフトウエアにより汎用ポートを操作することでシングルマスタの I²C-BUS 制御を行います。
P12(SDA)及び P13(SCL)端子は、外付けでプルアップ抵抗が必要になります。
表 1 に I²C-BUS インターフェース機能を示します。

表 1. シングルマスタ I²C-BUS インターフェース機能

項目	機能
通信モード	マスタ送信（シングルマスタ）
SCL クロック周波数	約 100kHz

注 1. CPU クロックが 16MHz で割り込み未使用時の値です。
CPU クロックが 16MHz 以外の場合で、この値にしたい場合は、調整が必要になります。

3. I²C-BUS

3.1 スタートコンディション／ストップコンディション

(1) スタートコンディション
SCL が”H”の状態で SDA を”H”から”L”へ立ち下げます。
後に、SCL を”L”に立ち下げます。
(2) ストップコンディション
SCL が”H”の状態で SDA を”L”から”H”へ立ち上げます。
後に、SCL を”L”に立ち下げます。

図 1 にスタートコンディション発生タイミング図、図 2 にストップコンディション発生タイミング図、表 2 にスタートコンディション／ストップコンディション発生タイミング表を示します。

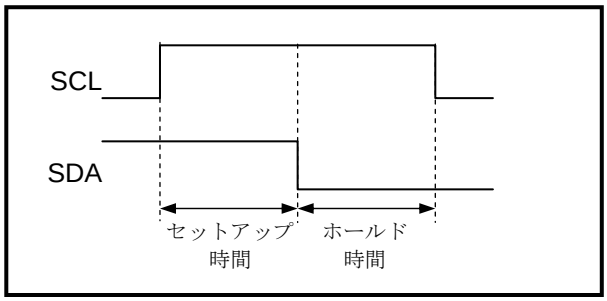


図 1. スタートコンディション発生タイミング図

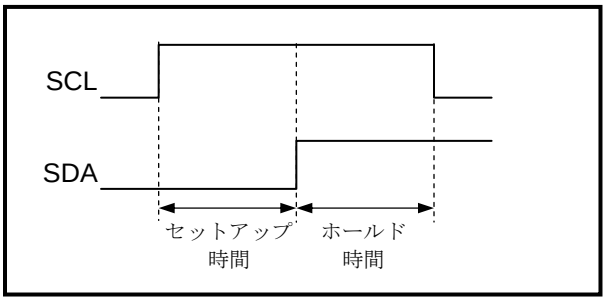


図 2. ストップコンディション発生タイミング図

表 2. スタートコンディション／ストップコンディション発生タイミング表

項目	スタートコンディション	ストップコンディション
セットアップ時間	約 2.0μs	約 1.6μs
ホールド時間	約 3.0μs	約 3.0μs

注 1. CPU クロックが 16MHz で割り込み未使用時の値です。
CPU クロックが 16MHz 以外の場合で、この値にしたい場合は、調整が必要になります。

3.2　データ入出力

(1) データ出力
　SDA へデータを出力し、データセットアップ時間経過後に SCL を出力します（”L”→”H”→”L”）。
(2) データ入力
　SCL を”H”に立ち上げた後、データ入力を行います。
　後に、SCL を”L”に立ち下げます。

図 3 にデータ入出力タイミング図、表 3 にデータ入出力タイミング表を示します。

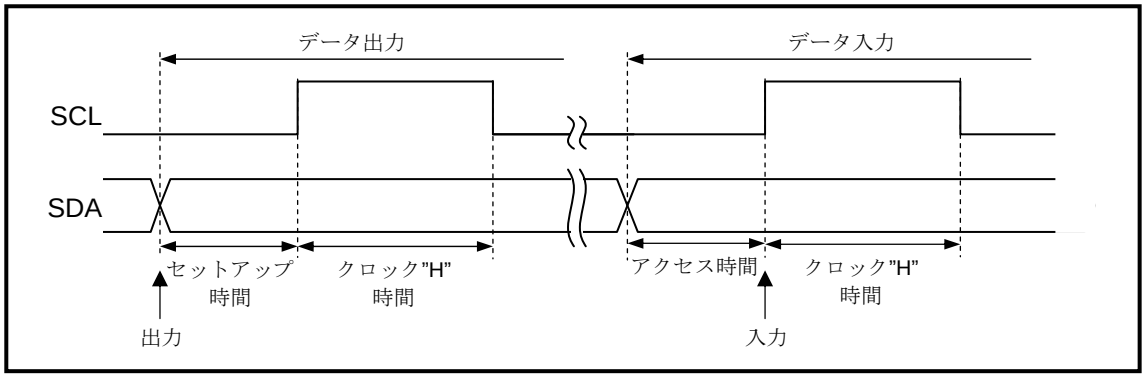


図 3.　データ入出力タイミング図

表 3.　データ入出力タイミング表

項目	データ出力	データ入力
セットアップ時間	約 3.3μs	－
アクセス時間	－	約 1μs 以上
クロック”H”時間	約 3.0μs	約 4.7μs

注 1.　CPU クロックが 16MHz で割り込み未使用時の値です。
　　CPU クロックが 16MHz 以外の場合で、この値にしたい場合は、調整が必要になります。

3.3　バイトフォーマット

1 バイトは、8 ビット長のデータと 1 ビット長の Acknowledge からなります。

Acknowledge は、データ転送が正しく行われたかどうかを示すための信号です。Acknowledge が”L”の場合はデータ転送が正しく行われたことを示し、”H”の場合はデータ転送が正しく行われなかったことを示します。

マスタがスレーブへデータを送信する場合は、9 クロック目でマスタが SDA ラインを解放し、スレーブが Acknowledge を出力します。マスタがスレーブからデータを受信する場合は、9 クロック目でスレーブが SDA を解放し、マスタが Acknowledge を出力します。

図 4 にバイトフォーマットを示します。

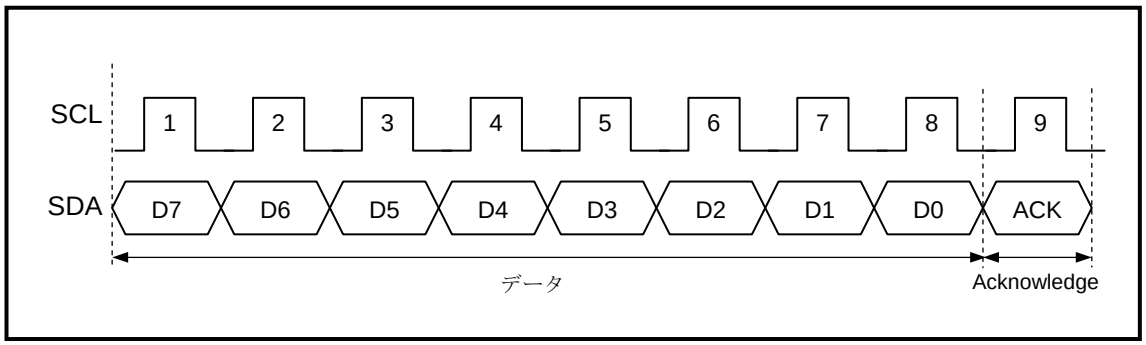


図 4.　バイトフォーマット

4. 応用例（EEPROM の制御）

2k ビットの EEPROM（HN58X2402SI）に、データの書き込み及び読み出しを行います。
7 ビットアドレッシングモードで、Device Address Word の下位 3 ビットで Device Address Code（A2,A1,A0）の指定を行います。
図 5 にマイコンと EEPROM（HN58X2402SI）の接続例を示します。

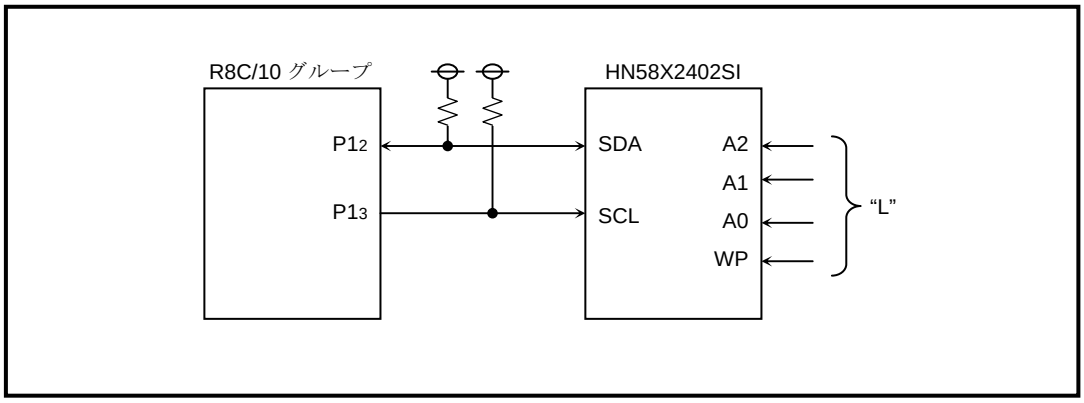


図 5. 接続例

4.1 Byte Write

Memory Address（W7～W0）で指定したアドレス(n)に Write Data データを書き込みます。
8 ビットの Write Data 出力後、Acknowledge を確認しストップコンディションを発行します。

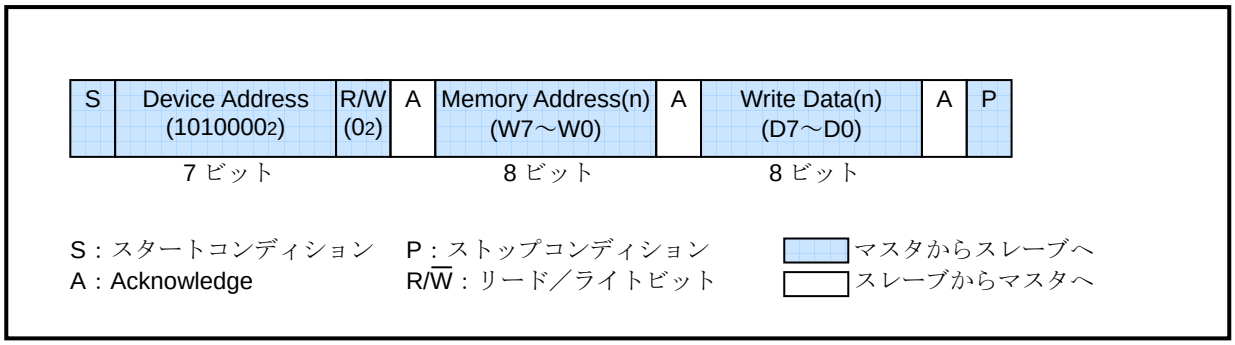


図 6. Byte Write

4.2 Page Write

Memory Address（W7～W0）で指定したアドレスから複数バイト(m+1)の Write Data データを書き込みます※。
指定バイトの Write Data 出力後、Acknowledge を確認しストップコンディションを発行します。

※ Page Write により最大 8 バイトのデータを連続して書き込めます。
詳細は、EEPROM(HN58X2402SI)のデータシートをご参照ください。

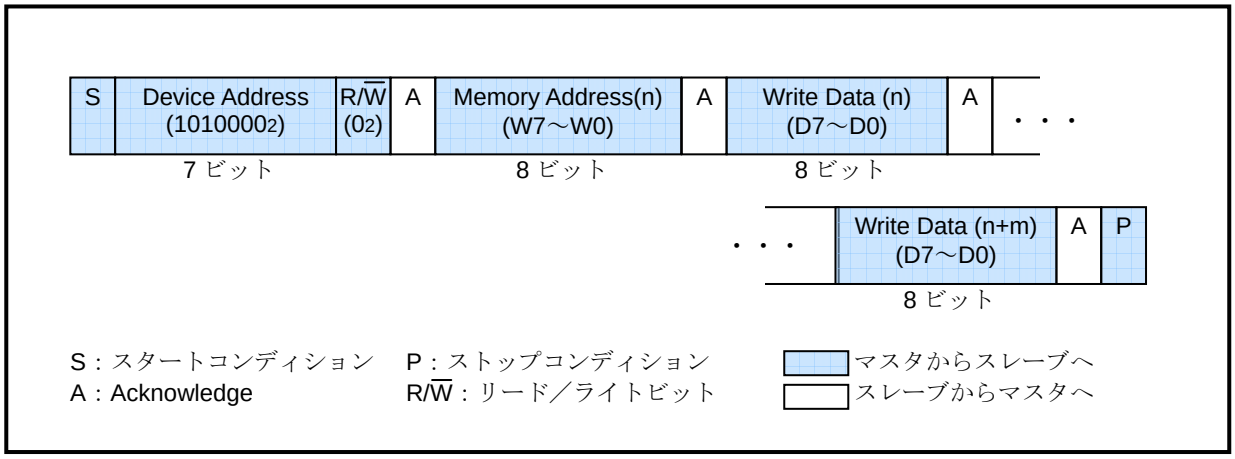


図 7.　Page Write

4.3 Sequential Read

Memory Address（W7～W0）で指定したアドレス(n)から Read Data を読み込みます。
Read Data 入力後、Acknowledge"0"を出力することで複数バイト(m+1)の Read Data を読み込むことができます。
指定バイトの Read Data を入力後、Acknowledge"1"を出力しストップコンディションを発行します。

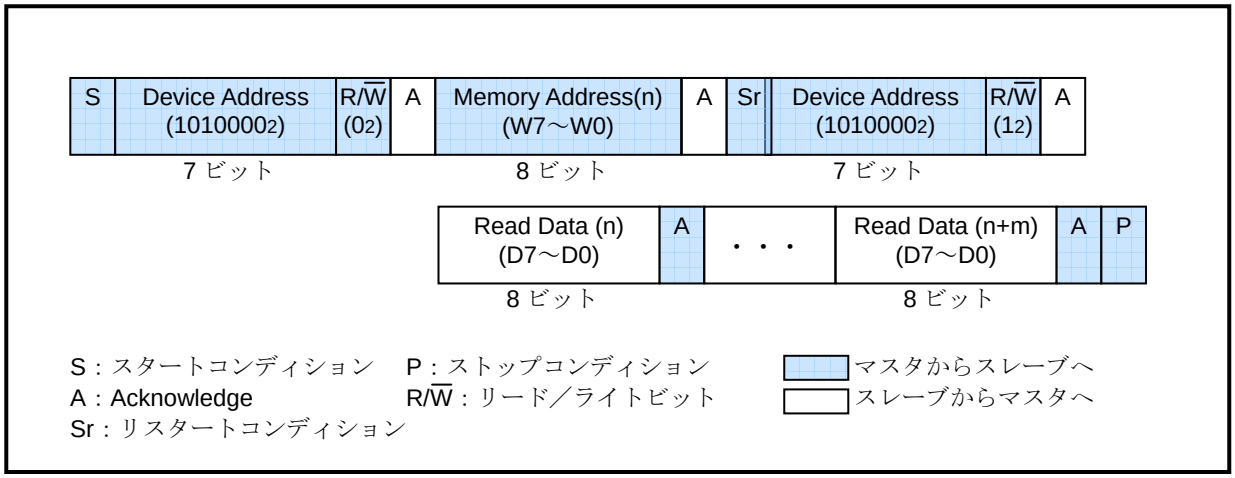
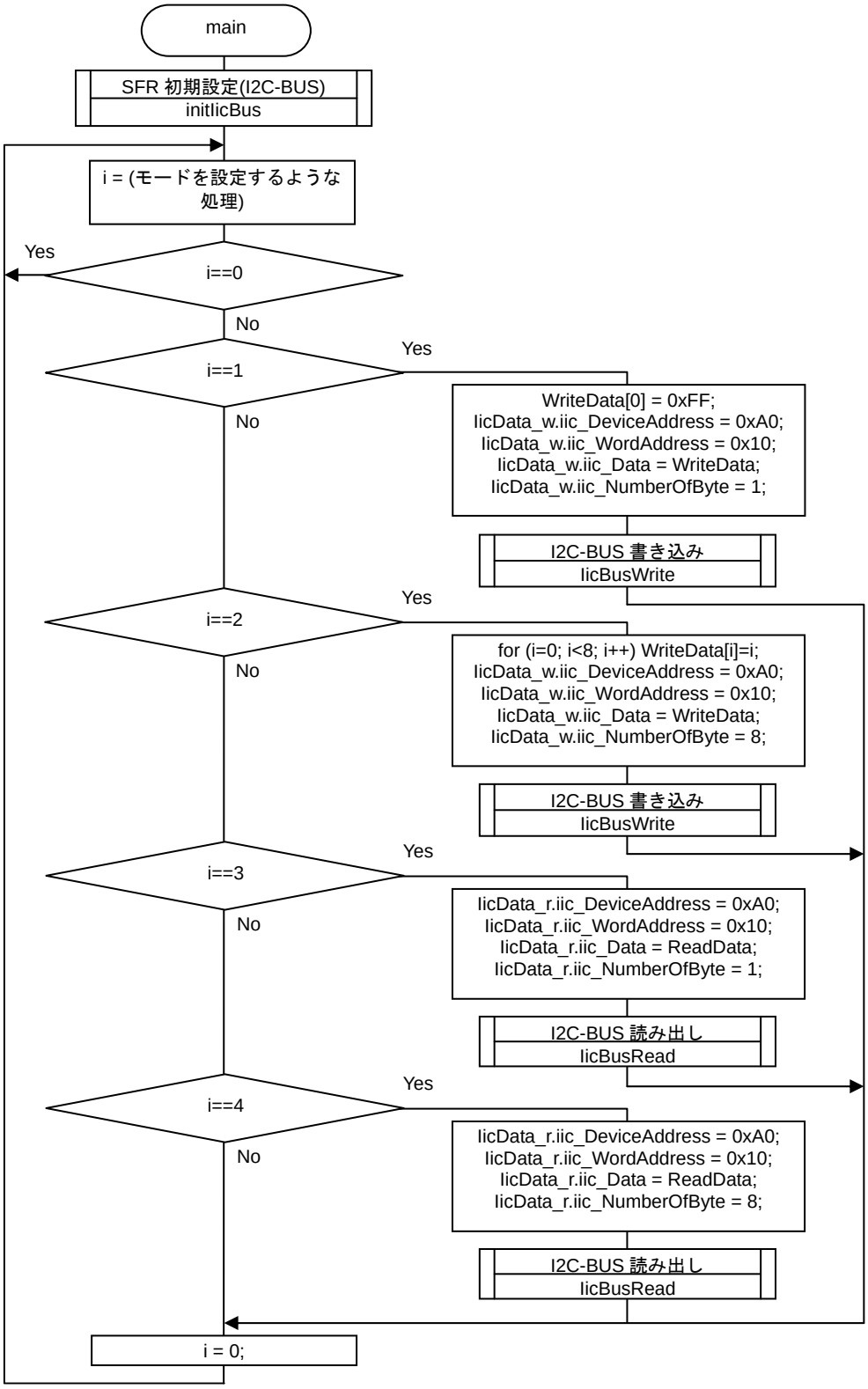


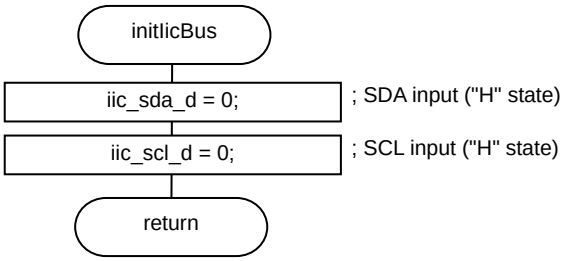
図 8.　シーケンシャルリードサイクル

5. フローチャート

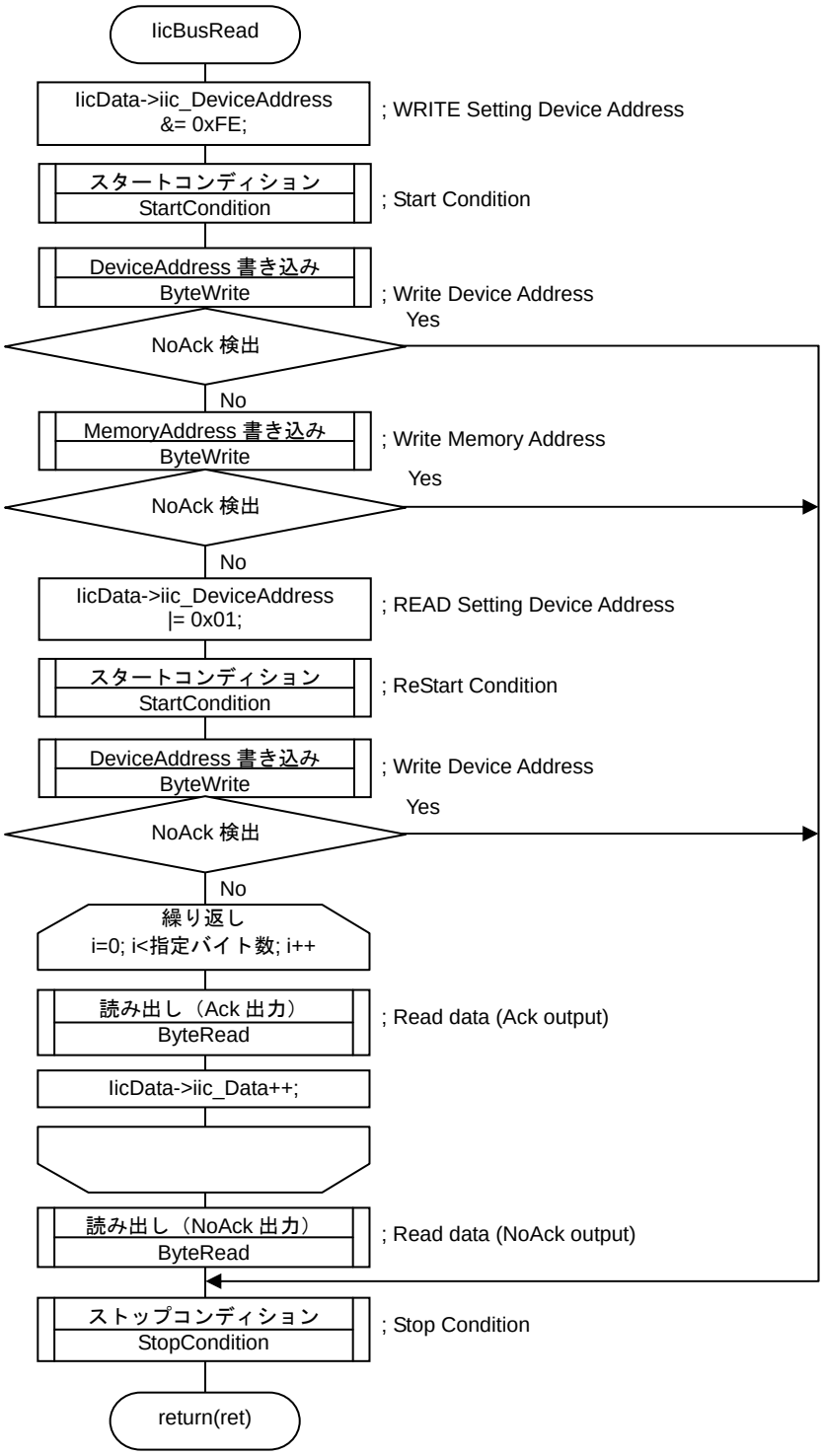
5.1 初期動作とメインループ



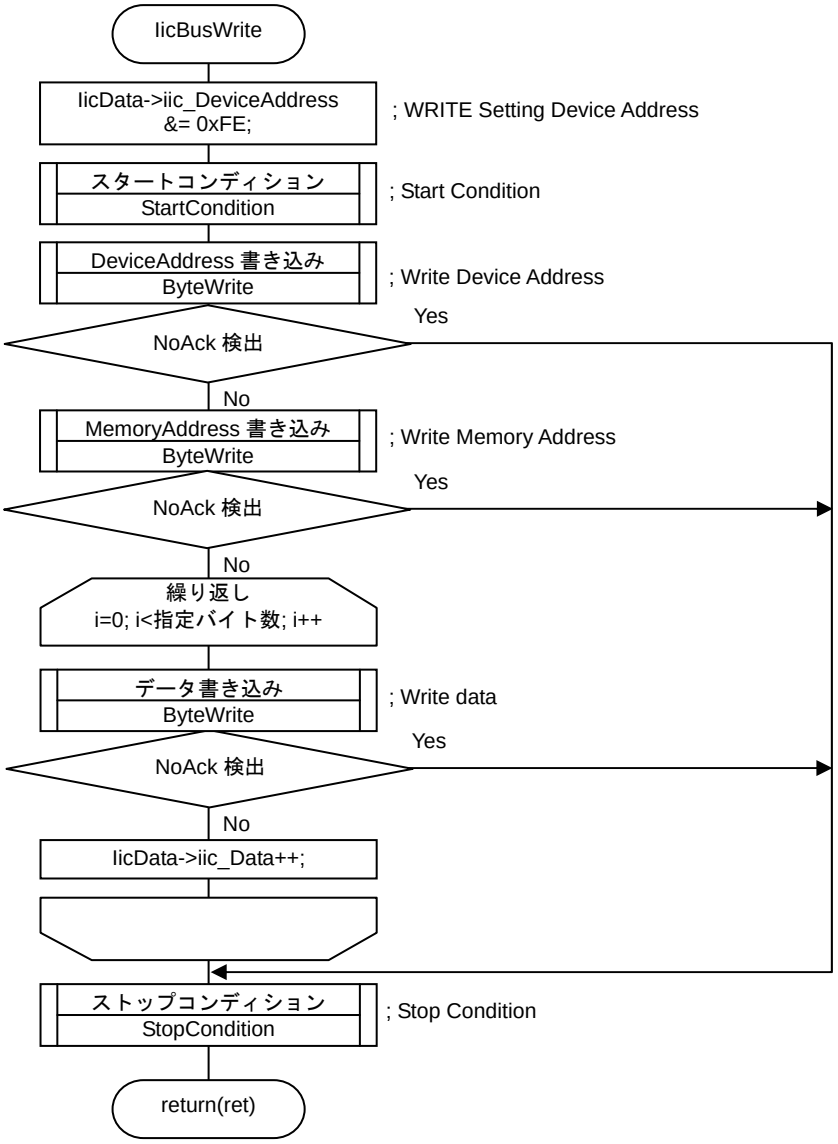
5.2　SFR 初期設定（I2C-BUS）



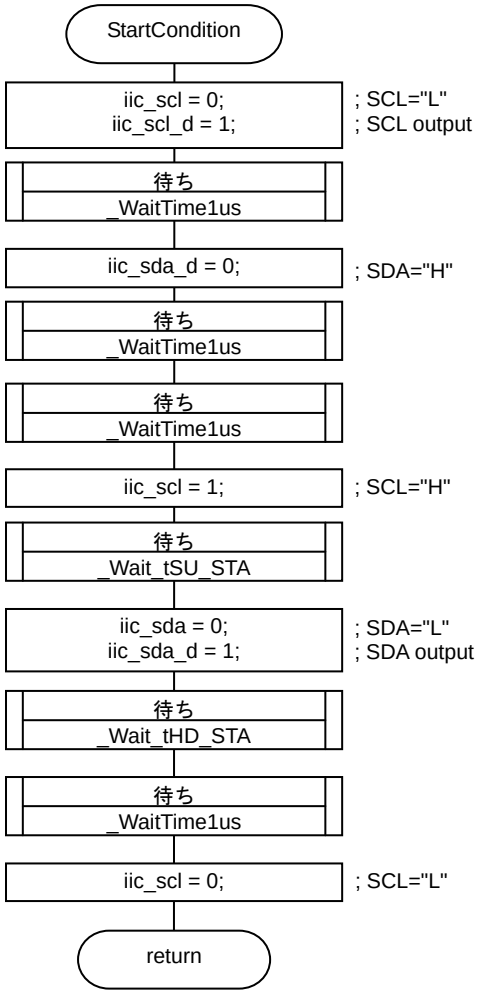
5.3　I2C-BUS 読み出し



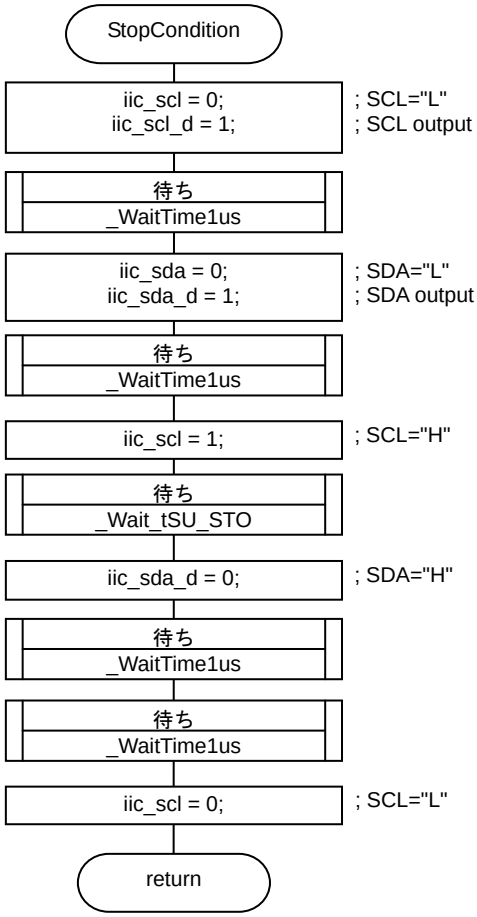
5.4　I2C-BUS 書き込み



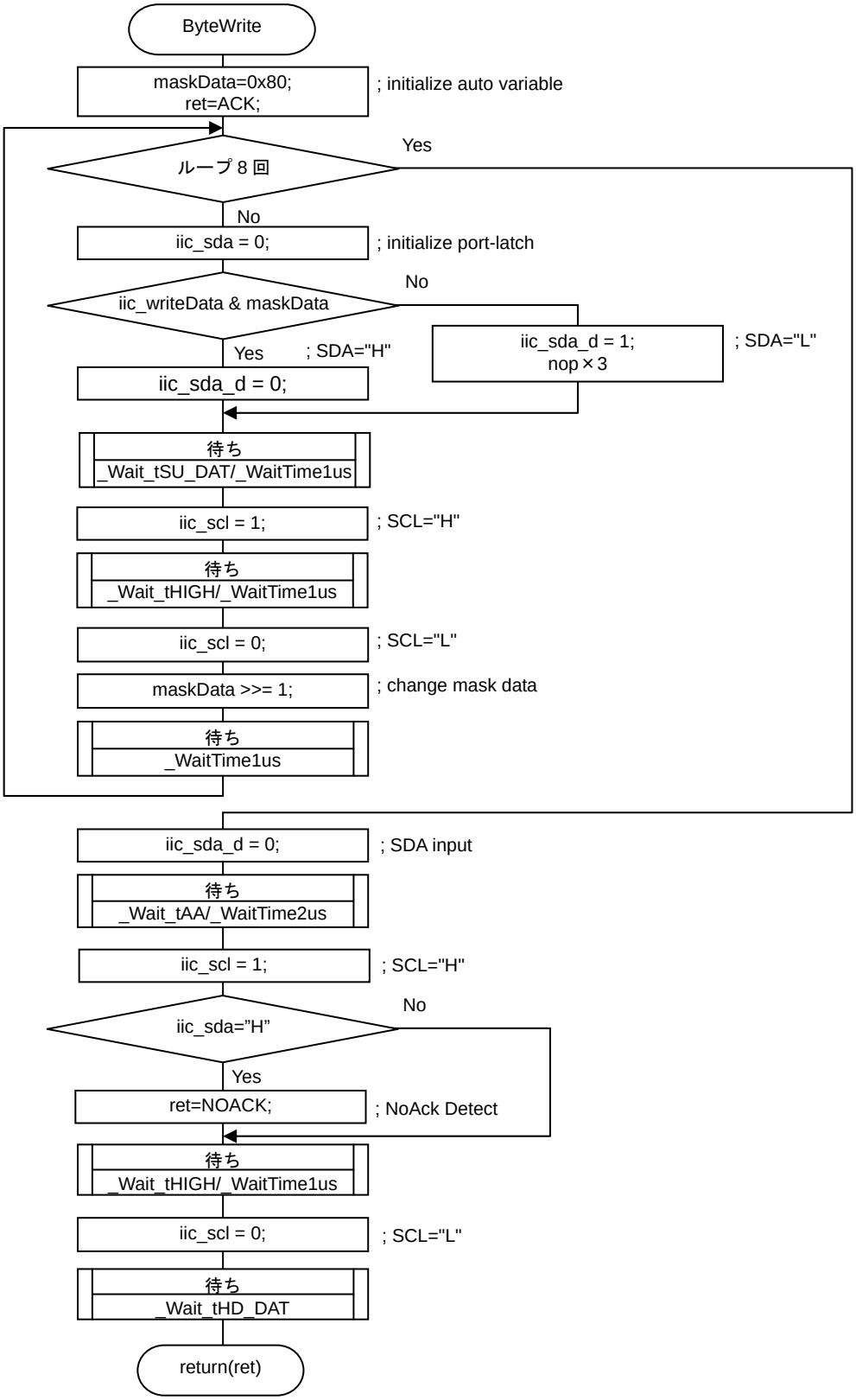
5.5　I2C-BUS スタートコンディション



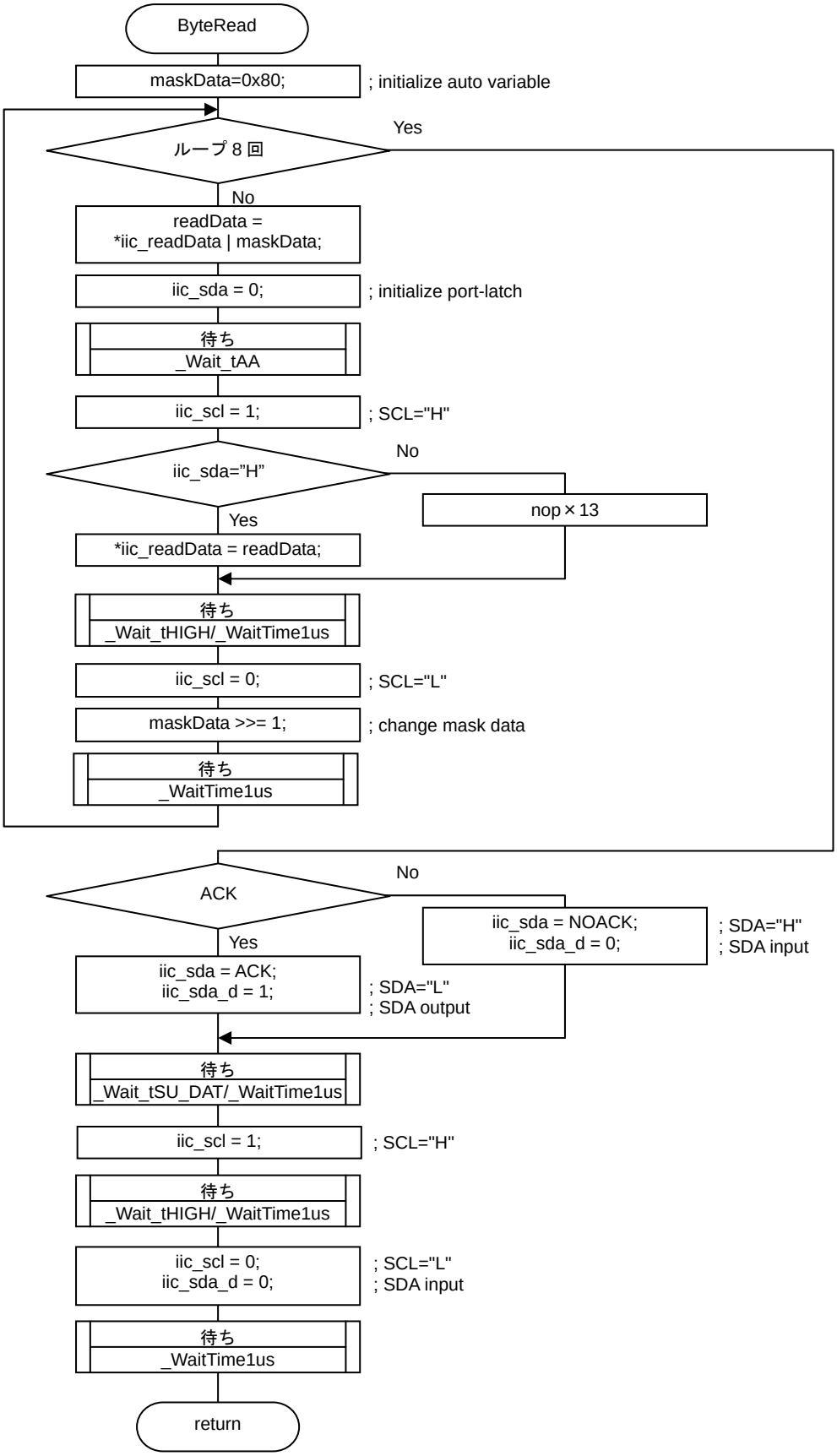
5.6　I2C-BUS ストップコンディション



5.7　I2C-BUS バイト書き込み



5.8　I²C-BUS バイト読み出し



6. プログラム

```
/************************************************************
 *                                                          *
 *      File Name       : main.c                            *
 *      Contents        : main file                         *
 *      Copyright       : RENESAS TECHNOLOGY CORPORATION    *
 *                        AND RENESAS SOLUTIONS CORPORATION *
 *      Version         : 1.0                               *
 *      note            :                                   *
 *                                                          *
************************************************************/

#include "sfrr8c10.h"
#include "Iic_Bus.h"

void main (void)
{
    static unsigned char i=0;
    static unsigned char WriteData[8];
    static unsigned char ReadData[8];
    IicPack IicData_w;
    IicPack IicData_r;

    p1_4 = 1;                   /* test port */
    pd1_4 = 1;                  /* test port */
    p1_1 = 1;                   /* test port */
    pd1_1 = 1;                  /* test port */

    while(1){
        while(i==0) {
            i = mode();                         /* Setting Access Mode */
        }
        p1_4 = 1;
        switch (i) {
        case 1:                                         /* Write data 1Byte */
            WriteData[0] = 0xAA;                        /* Setting write data */
            IicData_w.iic_DeviceAddress = 0xA0;
            IicData_w.iic_MemoryAddress = 0x10;
            IicData_w.iic_Data = WriteData;
            IicData_w.iic_NumberOfByte = 1;
            p1_4 = 0;
            if (IicBusWrite(&IicData_w) == ACK) {
                p1_4 = 1;
            };
            break;
        case 3:                                         /* Write data 8Bytes */
            for (i=0; i<8; i++) WriteData[i]=I*5;       /* Setting write data */
            IicData_w.iic_DeviceAddress = 0xA0;
            IicData_w.iic_MemoryAddress = 0x10;
            IicData_w.iic_Data = WriteData;
            IicData_w.iic_NumberOfByte = 8;
            p1_4 = 0;
```

```
            if (IicBusWrite(&IicData_w) == ACK) {
                p1_4 = 1;
            };
            break;
        case 2:                                         /* Read data 1Byte */
            IicData_r.iic_DeviceAddress = 0xA0;
            IicData_r.iic_MemoryAddress = 0x10;
            IicData_r.iic_Data = ReadData;
            IicData_r.iic_NumberOfByte = 1;
            p1_1 = 0;
            if(IicBusRead(&IicData_r) == ACK) {         /*  */
                p1_1 = 1;
            }
            break;
        case 4:                                         /* Read data 8Bytes */
            IicData_r.iic_DeviceAddress = 0xA0;
            IicData_r.iic_MemoryAddress = 0x10;
            IicData_r.iic_Data = ReadData;
            IicData_r.iic_NumberOfByte = 8;
            p1_1 = 0;
            if(IicBusRead(&IicData_r) == ACK) {         /*  */
                p1_1 = 1;
            }
            break;
        default:
            asm("nop");
            break;
        }
        p1_4 = 0;
        p1_1 = 0;
        i = 0;
    }
}

void init(void)
{
    asm("fclr   i");
    prcr = 0x01;
    cm0 = 0x08;
    cm1 = 0x28;
    ocrd = 0x00;
    prcr = 0x00;
}

unsigned char mode(void)
{
    unsigned int loop;
    static unsigned char mode=0;

    for (loop=1; 0!=loop; loop++) {}                    /* about 82ms at 16MHz/1 */
    if (++mode > 4) mode=0;                             /* change mode */
    return(mode);
}
```

```
/************************************************************
 *                                                          *
 *      File Name       : Iic_bus.h                         *
 *      Contents        : IIC Bus Definition file           *
 *      Copyright       : RENESAS TECHNOLOGY CORPORATION    *
 *                        AND RENESAS SOLUTIONS CORPORATION *
 *      Version         : 1.0                               *
 *      note            :                                   *
 *                                                          *
************************************************************/

#define ACK     0
#define NOACK   1

#define WRITE_MODE      0
#define READ_MODE       1

typedef unsigned char uchar;
typedef struct {
    unsigned char iic_DeviceAddress;
    unsigned char iic_MemoryAddress;
    unsigned char *iic_Data;
    unsigned char iic_NumberOfByte;
}IicPack;

void initIicBus(void);
unsigned char IicBusRead(IicPack *);
unsigned char IicBusWrite(IicPack *);
void StartCondition(void);
void StopCondition(void);
unsigned char ByteWrite(unsigned char);
void ByteRead (unsigned char *, unsigned char);
```

```
/************************************************************
 *                                                          *
 *      File Name       : Iic_bus.c                         *
 *      Contents        : IIC Bus file                      *
 *      Copyright       : RENESAS TECHNOLOGY CORPORATION    *
 *                        AND RENESAS SOLUTIONS CORPORATION *
 *      Version         : 1.0                               *
 *      note            :                                   *
 *                                                          *
 ************************************************************/

#include "sfr8c10.h"
#include "Iic_Bus.h"

#define iic_sda_d       pd1_2
#define iic_sda         p1_2
#define iic_scl_d       pd1_3
#define iic_scl         p1_3

void _WaitTime0us(void);
void _WaitTime1us(void);
void _WaitTime2us(void);

#define _Wait_tHIGH     _WaitTime1us()  /* Clock pulse width high */
#define _Wait_tLOW      _WaitTime2us()  /* Clock pulse width low */
#define _Wait_tHD_STA   _WaitTime1us()  /* Start hold time */
#define _Wait_tSU_STA   _WaitTime1us()  /* Start setup time */
#define _Wait_tHD_DAT   _WaitTime0us()  /* Data in hold time */
#define _Wait_tSU_DAT   _WaitTime1us()  /* Data in setup time */
#define _Wait_tAA       _WaitTime1us()  /* Access time */
#define _Wait_tSU_STO   _WaitTime1us()  /* Stop setup time */
#define _Wait_tBUF      _WaitTime2us()  /* Bus free time for next mode */


/******************************************************
 Name           : initIicBus
 Parameters     : None
 Returns        : None
 Description    : initialize I2C-BUS port
******************************************************/
void initIicBus(void)
{
    iic_sda_d = 0;              /* SDA input ("H" state) */
    iic_scl_d = 0;              /* SCL input ("H" state) */
}
```

```
/******************************************************
 Name           : IicBusRead
 Parameters     : structure IicPack pointer
 Returns        : Acknowledge
 Description    : Sequential Ramdom Read Cycle (I2C-BUS)
******************************************************/
unsigned char IicBusRead(IicPack *IicData)
{
    unsigned char i,ret;

    /* Ramdom Read Cycle / Sequential Ramdom Read Cycle */
    IicData->iic_DeviceAddress &= 0xFE;               /* WRITE Setting DeviceAddress */
    StartCondition();                                 /* Start Condition */
    while (1) {
        if ((ret=ByteWrite(IicData->iic_DeviceAddress)) == NOACK)
                                                      /* WRITE DeviceAddress */
            break;                                    /* NoAck Detect */
        if ((ret=ByteWrite(IicData->iic_MemoryAddress)) == NOACK)
                                                      /* WRITE MemoryAddress */
            break;                                    /* NoAck Detect */
        IicData->iic_DeviceAddress |= 0x01;           /* READ Setting DeviceAddress */
        StartCondition();                             /* ReStart Condition */
        if ((ret=ByteWrite(IicData->iic_DeviceAddress)) == NOACK)
                                                      /* DeviceAddress WRITE */
            break;                                    /* NoAck Detect */
        for (i=1; i<IicData->iic_NumberOfByte; i++) { /* specified bytes as loop */
            ByteRead (IicData->iic_Data, ACK);        /* Read data (Ack output) */
            IicData->iic_Data++;                      /*  */
        }
        ByteRead (IicData->iic_Data, NOACK);          /* Read data (NoAck output) */
        break;
    }
    StopCondition();                                  /* Stop Condition */
    return(ret);
}
```

```
/******************************************************
 Name          : IicBusWrite
 Parameters    : structure IicPack pointer
 Returns       : Acknowledge
 Description   : Byte Write or Page Write Cycle (I2C-BUS)
******************************************************/
unsigned char IicBusWrite(IicPack *IicData)
{
    unsigned char i,ret;

    /* Byte Write / Page Write */
    IicData->iic_DeviceAddress &= 0xFE;              /* WRITE Setting DeviceAddress */
    StartCondition();                                /* Start Condition */
    while (1) {
        if ((ret=ByteWrite(IicData->iic_DeviceAddress)) == NOACK)
                                                     /* WRITE DeviceAddress */
            break;                                   /* NoAck Detect */
        if ((ret=ByteWrite(IicData->iic_MemoryAddress)) == NOACK)
                                                     /* WRITE MemoryAddress */
            break;                                   /* NoAck Detect */
        for (i=0; i<IicData->iic_NumberOfByte; i++) {  /* specified bytes as loop */
            if ((ret=ByteWrite(*(IicData->iic_Data))) == NOACK)       /* Write Data */
                break;                               /* NoAck Detect */
            IicData->iic_Data++;                     /*  */
        }
        break;
    }
    StopCondition();                                 /* Stop Condition */
    return(ret);
}
```

```
/******************************************************
 Name           : StartCondition
 Parameters     : None
 Returns        : None
 Description    : Output Start Condition (I2C-BUS)
 Note           : *1 adjust a wait time
******************************************************/
void StartCondition(void)
{
    iic_scl = 0;                            /* SCL="L" */
    iic_scl_d = 1;                          /* SCL output */
    _WaitTime1us();                         /* wait *1 */
    iic_sda_d = 0;                          /* SDA="H" */
    _WaitTime1us();                         /* wait */
    _WaitTime1us();                         /* wait *! */
    iic_scl = 1;                            /* SCL="H" */
    _Wait_tSU_STA;                          /* wait */
    iic_sda = 0;                            /* SDA="L" */
    iic_sda_d = 1;                          /* SDA output */
    _Wait_tHD_STA;                          /* wait */
    _WaitTime1us();                         /* wait *1 */
    iic_scl = 0;                            /* SCL="L" */
}

/******************************************************
 Name           : StopCondition
 Parameters     : None
 Returns        : None
 Description    : Output Stop Condition (I2C-BUS)
 Note           : *1 adjust a wait time
******************************************************/
void StopCondition(void)
{
    iic_scl = 0;                            /* SCL="L" */
    iic_scl_d = 1;                          /* SCL output */
    _WaitTime1us();                         /* wait *1 */
    iic_sda = 0;                            /* SDA="L" */
    iic_sda_d = 1;                          /* SDA output */
    _WaitTime1us();                         /* wait *1 */
    iic_scl = 1;                            /* SCL="H" */
    _Wait_tSU_STO;                          /* wait */
    iic_sda_d = 0;                          /* SDA="H" */
    _WaitTime1us();                         /* wait */
    _WaitTime1us();                         /* wait *1 */
    iic_scl = 0;                            /* SCL="L" */
}
```

```
/******************************************************
 Name           : ByteWrite
 Parameters     : Write data
 Returns        : Acknowledge
 Description    : byte data Output (I2C-BUS)
 Note           : *1 adjust a wait time
******************************************************/
unsigned char ByteWrite(unsigned char iic_writeData)
{
    unsigned char maskData=0x80;              /* MSB first */
    unsigned char ret=ACK;                    /* Ack/NoAck */

    while (maskData) {                        /* 8times as loop */
        iic_sda = 0;                          /* initialize port-latch */
        if (iic_writeData & maskData) {       /* "H" output ? */
            iic_sda_d = 0;                    /* Yes SDA="H" */
        }else{
            iic_sda_d = 1;                    /* No  SDA="L" */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
        }
        _Wait_tSU_DAT;                        /* wait */
        _WaitTime1us();                       /* wait *1 */
        iic_scl = 1;                          /* SCL="H" */
        _Wait_tHIGH;                          /* wait */
        _WaitTime1us();                       /* wait *1 */
        iic_scl = 0;                          /* SCL="L" */
        maskData >>= 1;                       /* change mask data */
        _WaitTime1us();                       /* wait *1 */
    }
    iic_sda_d = 0;                            /* SDA input */
    _Wait_tAA;                                /* wait */
    _WaitTime2us();                           /* wait *1 */
    iic_scl = 1;                              /* SCL="H" */
    if (iic_sda) ret=NOACK;                   /* NoAck Detect */
    _Wait_tHIGH;                              /* wait */
    _WaitTime1us();                           /* wait *1 */
    iic_scl = 0;                              /* SCL="L" */
    _Wait_tHD_DAT;                            /* wait */
    return(ret);
}
```

```
/******************************************************
 Name           : ByteRead
 Parameters     : Read data strage location pointer, Select Ack/NoAck
 Returns        : None
 Description    : byte data input with Ack output (I2C-BUS)
 Note           : *1 adjust a wait time
******************************************************/
void ByteRead(unsigned char *iic_readData, unsigned char ackData)
{
    unsigned char maskData=0x80;              /* MSB first */
    unsigned char readData;

    *iic_readData = 0;                        /*  */
    while (maskData) {                        /* 8times as loop */
        readData = *iic_readData | maskData;  /*  */
        iic_sda_d = 0;                        /* initialize port-latch */
        _Wait_tAA;                            /* wait */
        iic_scl = 1;                          /* SCL="H" */
        if (iic_sda) {                        /* SDA="H" ? */
            *iic_readData = readData;         /* Yes  */
        }else{
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
            asm("nop");                       /* wait *1 */
        }
        _Wait_tHIGH;                          /* wait */
        _WaitTime1us();                       /* wait *1 */
        iic_scl = 0;                          /* SCL="L" */
        maskData >>= 1;                       /* Change mask data */
        _WaitTime1us();                       /* wait *1 */
    }
    if (!ackData) {                           /* Ack output ? */
    /* Ack output */
        iic_sda = ACK;                        /* Yes SDA="L" */
        iic_sda_d = 1;                        /* SDA output */
    }else{
    /* NoAck output */
        iic_sda = NOACK;                      /* No  SDA="H" */
        iic_sda_d = 0;                        /* SDA input */
    }
    _Wait_tSU_DAT;                            /* wait */
    _WaitTime1us();                           /* wait *1 */
    iic_scl = 1;                              /* SCL="H" */
```

```
    _Wait_tHIGH;                                /* wait */
    _WaitTime1us();                             /* wait *1 */
    iic_scl = 0;                                /* SCL="L" */
    iic_sda_d = 0;                              /* SDA input */
    _WaitTime1us();                             /* wait *1 */
}


/*******************************************************
 Name           : _WaitTime0us
 Parameters     : None
 Returns        : None
 Description    : a 0us wait
*******************************************************/
void _WaitTime0us(void)
{
}


/*******************************************************
 Name           : _WaitTime1us
 Parameters     : None
 Returns        : None
 Description    : a 1us wait
*******************************************************/
void _WaitTime1us(void)
{
    /* +14cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle = 16cycle */
}


/*******************************************************
 Name           : _WaitTime2us
 Parameters     : None
 Returns        : None
 Description    : a 2us wait
*******************************************************/
void _WaitTime2us(void)
{
    /* +14cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
```

```
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle */
    asm("nop");         /* +1cycle = 32cycle */
}
```

7. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
R8C/10 グループ　ハードウエアマニュアル
（最新版をルネサスエレクトロニクスホームページから入手してください。）

テクニカルアップデート/テクニカルニュース
（最新版をルネサスエレクトロニクスホームページから入手してください。）

8. 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

お問い合わせ先
http://japan.renesas.com/contact/

改訂記録	R8C/10 グループ 汎用ポートによる I²C-BUS 制御プログラムアプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.09.19	-	初版発行
1.10	2012.06.01	1	発振安定待ちに関する注意事項を追加
		-	旧ドキュメント番号：RJJ05B0306