文档编号：AN1003

上海东软载波微电子有限公司

应用笔记

HW2000 Programming Guide

版权所有©上海东软载波微电子有限公司

修订历史

版本	修订日期	修改概要
V1.0	2018-5-8	初版公开发布

地　址：中国上海市龙漕路 299 号天华信息科技园 2A 楼 5 层
邮　编：200235
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com

版权所有©

上海东软载波微电子有限公司

本资料内容为上海东软载波微电子有限公司在现有数据资料基础上慎重且力求准确无误编制而成，本资料中所记载的实例以正确的使用方法和标准操作为前提，使用方在应用该等实例时请充分考虑外部诸条件，上海东软载波微电子有限公司不担保或确认该等实例在使用方的适用性、适当性或完整性，上海东软载波微电子有限公司亦不对使用方因使用本资料所有内容而可能或已经带来的风险或后果承担任何法律责任。基于使本资料的内容更加完善等原因，上海东软载波微电子有限公司保留未经预告的修改权。使用方如需获得最新的产品信息，请随时用上述联系方式与上海东软载波微电子有限公司联系。

目 录

内容目录

第 1 章 SPI 接口 4
1.1 SPI 帧格式和通讯时序 4
1.2 SPI 读写函数原型 4
1.3 HR7P 系列 MCU SPI 汇编驱动 5
第 2 章 程序设计示例 12
2.1 HW2000 CE、PD_CTRL、SFT_RST 区别 12
2.2 HW2000 默认寄存器初始化 12
2.3 数据载荷说明 13
2.4 HW2000 TX CW 13
2.5 HW2000 TX NOACK 14
2.6 HW2000 RX NOACK 15
2.7 HW2000 RX with RSSI 15
2.8 HW2000 TX ACK NOPAYLOAD 16
2.9 HW2000 RX ACK NOPAYLOAD 17
2.10 HW2000 TX ACK PAYLOAD 18
2.11 HW2000 RX ACK PAYLOAD 19
2.12 HW2000 ACK PAYLOAD 模式特别说明 20
2.13 状态查询模式特别说明 20

版权所有©上海东软载波微电子有限公司 http://www.essemi.com

第 1 章 SPI接口

1.1 SPI帧格式和通讯时序

SPI 通讯帧格式和时序，参见 HW2000 芯片数据手册的“SPI 通信接口”章节。

1.2 SPI读写函数原型

通用 MCU 端需要根据 SPI 读写时序，实现寄存器读写和 FIFO 读写功能函数，函数原型如下：

```
/*****************************************************************************
* 函数名称：hw2000_write_reg
* 功能描述：写 HW2000 寄存器
* 输入参数：addr    寄存器地址
            value   寄存器值
* 返回参数：无
*****************************************************************************/
void hw2000_write_reg(uint8_t addr, uint16_t value)

/*****************************************************************************
* 函数名称：hw2000_read_reg
* 功能描述：读 HW2000 寄存器
* 输入参数：addr    寄存器地址
* 返回参数：value   寄存器值
*****************************************************************************/
uint16_t hw2000_read_reg(uint8_t addr)

/*****************************************************************************
* 函数名称：hw2000_write_fifo
* 功能描述：写 HW2000 FIFO
* 输入参数：addr     FIFO 地址
            data     数据地址
            length   数据长度
*****************************************************************************/
void hw2000_write_fifo(uint8_t addr, uint8_t *data, uint8_t length)
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
/*****************************************************************************
* 函数名称：hw2000_read_fifo
* 功能描述：写 HW2000 FIFO
* 输入参数：addr     FIFO 地址
            data     数据地址
            length   数据长度
* 返回参数：无
*****************************************************************************/
void hw2000_read_fifo(uint8_t addr, uint8_t *data, uint8_t length)
```

1.3 HR7P系列MCU SPI汇编驱动

用户若采用上海东软载波微电子有限公司 HR7P 系列 MCU 作为主控，推荐采用以下 SPI 汇编驱动程序。

```
#define  PCSN    PB     //MCU 端口选择，用户需要初始化输入/输出属性
#define  PSCK    PA
#define  PMOSI   PB
#define  PMISO   PA

#define CSN      0x00   //MCU 端口比特位选择，用户需要初始化输入/输出属性
#define SCK      0x02
#define MOSI     0x01
#define MISO     0x03

void hw2000_write_reg(uint8_t addr, uint16_t value)
{
uint8_t i;

__asm
{
    MOVI    0x80
    IOR     (&addr&) % 0x80, 1

    BCC     PCSN, CSN
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
CLR     (&i&)   % 0x80

JBC     (&addr&)   % 0x80, 7 ;write     addr
BSS     PMOSI, MOSI
JBS     (&addr&)   % 0x80, 7
BCC     PMOSI, MOSI
RL      (&addr&)   % 0x80
BSS     PSCK, SCK
INC     (&i&)   % 0x80, 1
BCC     PSCK, SCK
MOVI    0x08
XOR     (&i&)   % 0x80, 0
JBS     PSW, Z
GOTO    $-0x0B;

CLR     (&i&)   % 0x80

JBC     (&value&)   % 0x80 + 1, 7 ;write     value_h
BSS     PMOSI, MOSI
JBS     (&value&)   % 0x80 + 1, 7
BCC     PMOSI, MOSI
RL      (&value&)   % 0x80 + 1
BSS     PSCK, SCK
INC     (&i&)   % 0x80, 1
BCC     PSCK, SCK
MOVI    0x08
XOR     (&i&)   % 0x80, 0
JBS     PSW, Z
GOTO    $-0x0B

CLR     (&i&)   % 0x80

JBC     (&value&)   % 0x80, 7;write     value_l
BSS     PMOSI, MOSI
JBS     (&value&)   % 0x80, 7
BCC     PMOSI, MOSI
RL      (&value&)   % 0x80
BSS     PSCK, SCK
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        INC     (&i&)   % 0x80, 1
        BCC     PSCK, SCK
        MOVI    0x08
        XOR     (&i&)   % 0x80, 0
        JBS     PSW, Z
        GOTO    $-0x0B

        BSS     PCSN, CSN
    }
    }

    uint16_t hw2000_read_reg(uint8_t addr)
    {
        uint8_t i;
        uint16_t value;

    __asm
    {
        BCC     PCSN, CSN

        CLR     (&i&)   % 0x80

        JBC     (&addr&)   % 0x80, 7 ;write     addr
        BSS     PMOSI, MOSI
        JBS     (&addr&)   % 0x80, 7
        BCC     PMOSI, MOSI
        RL      (&addr&)   % 0x80
        BSS     PSCK, SCK
        INC     (&i&)   % 0x80, 1
        BCC     PSCK, SCK
        MOVI    0x08
        XOR     (&i&)   % 0x80, 0
        JBS     PSW, Z
        GOTO    $-0x0B;

        CLR     (&i&)   % 0x80

        BSS     PSCK, SCK                   ;read value
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        BCC     PSW, C
        RL      (&value&)   % 0x80
        RL      (&value&)   % 0x80 + 1
        BCC     PSCK, SCK
        JBC     PMISO, MISO
        BSS     (&value&)   % 0x80, 0
        INC     (&i&)   % 0x80, 1
        MOVI    0x10
        XOR     (&i&)   % 0x80, 0
        JBS     PSW, Z
        GOTO    $-0x0B;

        BSS     PCSN, CSN
    }

        return value;
    }

    void hw2000_write_fifo(uint8_t addr, uint8_t *data, uint8_t length)
    {
    uint8_t i, j;
    uint8_t   value;

    __asm
    {
        MOVI    0x80
        IOR     (&addr&) % 0x80, 1

        BCC     PCSN, CSN

        CLR     (&i&)   % 0x80

        JBC     (&addr&)   % 0x80, 7 ;write     addr
        BSS     PMOSI, MOSI
        JBS     (&addr&)   % 0x80, 7
        BCC     PMOSI, MOSI
        RL      (&addr&)   % 0x80
        BSS     PSCK, SCK
```


版权所有©上海东软载波微电子有限公司

```
        INC     (&i&)   % 0x80, 1
        BCC     PSCK, SCK
        MOVI    0x08
        XOR     (&i&)   % 0x80, 0
        JBS     PSW, Z
        GOTO    $-0x0B;
    }

    for (j = 0; j < length; j++) {
        value = data[j];

        __asm
        {
            CLR     (&i&)   % 0x80

            JBC     (&value&)   % 0x80, 7;write data[j]
            BSS     PMOSI, MOSI
            JBS     (&value&)   % 0x80, 7
            BCC     PMOSI, MOSI
            RL      (&value&)   % 0x80
            BSS     PSCK, SCK
            INC     (&i&)   % 0x80, 1
            BCC     PSCK, SCK
            MOVI    0x08
            XOR     (&i&)   % 0x80, 0
            JBS     PSW, Z
            GOTO    $-0x0B
        }
    }

    __asm
    {
        BSS     PCSN, CSN
    }
    }

    void hw2000_read_fifo(uint8_t addr, uint8_t *data, uint8_t length)
    {
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
uint8_t i, j;
uint8_t   value;

__asm
{
    BCC     PCSN, CSN

    CLR     (&i&)   % 0x80

    JBC     (&addr&)   % 0x80, 7 ;write     addr
    BSS     PMOSI, MOSI
    JBS     (&addr&)   % 0x80, 7
    BCC     PMOSI, MOSI
    RL      (&addr&)   % 0x80
    BSS     PSCK, SCK
    INC     (&i&)   % 0x80, 1
    BCC     PSCK,    SCK
    MOVI    0x08
    XOR     (&i&)   % 0x80, 0
    JBS     PSW, Z
    GOTO    $-0x0B;
}

for (j = 0; j < length; j++) {
    __asm
    {
        CLR     (&i&)   % 0x80

        BSS     PSCK, SCK                   ;read data
        BCC     PSW, C
        RL      (&value&)   % 0x80
        BCC     PSCK, SCK
        JBC     PMISO, MISO
        BSS     (&value&)   % 0x80, 0
        INC     (&i&)   % 0x80, 1
        MOVI    0x08
        XOR     (&i&)   % 0x80, 0
        JBS     PSW, Z
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
            GOTO    $-0x0A;
        }
        data[j] = value;
    }

    __asm
    {
        BSS     PCSN, CSN
    }
}
```


版权所有©上海东软载波微电子有限公司

第 2 章　　程序设计示例

以下为 HW2000 芯片常用收发程序示例。

2.1　HW2000 CE、PD_CTRL、SFT_RST区别

正常工作时，请将 CE 引脚拉高，使能 HW2000 芯片。若拉低 CE 引脚将全局复位 HW2000 芯片，包括复位芯片内部各状态信号与寄存器。

PD_CTR(0x23[15])为芯片进入 POWER DOWN 模式使能信号，仅控制芯片进入掉电模式，寄存器状态保持并可读写。

SFT_RST(0x23[14])为软件复位使能信号，仅复位芯片内部各状态信号，并不复位内部寄存器。也即 INT 中断信号等状态变量将会被复位， 同时 SFT_RST 使能后会将写保护关闭（0x4c 寄存器的值变为 0xffff），导致功率寄存器（0x12 0x13 0x14 0x15）等未开放寄存器不能修改。

2.2　HW2000 默认寄存器初始化

HW2000 上电后需要初始化以下寄存器，1Mbps 和 250Kbps 初始化函数如下。

```
void hw2000_init_250K(void)
{
    uint8_t i;
    uint16_t agcTab[18] = { 0x0012, 0x0012, 0x0012, 0x0012, 0x0012, 0x0012,
                            0x0052, 0x0249, 0x0251, 0x0252, 0x0451, 0x0491,
                            0x04D1, 0x04D9, 0x06D8, 0x06E0, 0x06E1, 0x06E2
                           };

    hw2000_write_reg(0x4C, 0x55AA);
    for (i = 0; i < 18; i++) {
        hw2000_write_reg(0x50 + i, agcTab[i]);
    }
    hw2000_write_reg(0x01, 0x4D55);
    hw2000_write_reg(0x02, 0x44CC);
    hw2000_write_reg(0x08, 0x7BC8);
    hw2000_write_reg(0x09, 0xCC00);
    hw2000_write_reg(0x28, 0x8402);
    hw2000_write_reg(0x2C, 0x918B);
    hw2000_write_reg(0x2A, 0x40B4);
    hw2000_write_reg(0x1A, 0x0D31);
    hw2000_write_reg(0x19, 0x0884);
    hw2000_write_reg(0x20, 0xF000);
```


版权所有©上海东软载波微电子有限公司

```
}

void hw2000_init_1M(void)
{
    uint8_t i;
    uint16_t agcTab[18] = { 0x0012, 0x0012, 0x0012, 0x0012, 0x0012, 0x0012,
                            0x0052, 0x0249, 0x0251, 0x0252, 0x0451, 0x0491,
                            0x04D1, 0x04D9, 0x06D8, 0x06E0, 0x06E1, 0x06E2
                            };
    hw2000_write_reg(0x4C, 0x55AA);
    for (i = 0; i < 18; i++) {
        hw2000_write_reg(0x50 + i, agcTab[i]);
    }
    hw2000_write_reg(0x01, 0x4D55);
    hw2000_write_reg(0x02, 0x44CC);
    hw2000_write_reg(0x08, 0x7BC8);
    hw2000_write_reg(0x09, 0xCC00);
    hw2000_write_reg(0x28, 0x8402);
    hw2000_write_reg(0x2C, 0x918B);
    hw2000_write_reg(0x1B, 0xE754);
    hw2000_write_reg(0x06, 0xB000);
    hw2000_write_reg(0x07, 0x54E0);
    hw2000_write_reg(0x1C, 0x51A0);
    hw2000_write_reg(0x19, 0x2084);
    hw2000_write_reg(0x20, 0xF000);
    hw2000_write_reg(0x2A, 0xC0E4);
}
```

2.3 数据载荷说明

HW2000 收发数据 FIFO 深度为 64，若 0x29[12]设置为‘1’，则 FIFO 第一个字节代表收发数据长度，用户数据需限制为 63 字节，注意 FIFO 第一字节所代表数据长度并未包含该字节本身。

2.4 HW2000 TX CW

HW2000 提供单载波发送模式，以方便频点测试与发送功率测试。

```
void power_test(void) //使能单载波发送
{
    _reg = hw2000_read_reg(0x1C); //读出寄存器 0x1C 的原始值，关闭单载波发送时，
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
                                        //0x1C 寄存器的值需还原为原始值
        hw2000_write_reg(0x1C, _reg&0xFE7F);
        hw2000_write_reg(0x29, 0x0000);
        hw2000_write_reg(0x21, 0x0100);
        hw2000_write_reg(0x36, 0x0081);
    }

    void power_test_cancel(void) //关闭单载波发送
    {
        hw2000_write_reg(0x36, 0x0080);
        hw2000_write_reg(0x21, 0x0000);
        hw2000_write_reg(0x29, 0x1800);
        hw2000_write_reg(0x3D, 0x0008);
        hw2000_write_reg(0x1C,   _reg); //还原 0x1C 寄存器的原始值
    }
```

2.5 HW2000 TX NOACK

以下配置实现 HW2000 最基本的 FIFO 发射功能，ACK 功能不使能。用户需参考 SPI 时序实现相关 SPI 操作函数。

```
    _data[0] = 15;    //数据格式_data[]={15, 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}
    for (i = 1; i < 16; i++)
        _data[i] = i;
    while (1) {
        hw2000_write_reg(0x21, 0x0100);    // TX enable
        delay_us(5); //延时>=5us
        hw2000_write_reg(0x3B, 0x8000);    //清发射 FIFO
        hw2000_write_fifo(0x32, _data, _data[0]+1);
        hw2000_write_reg(0x36, 0x0081);
        hw2000_write_reg(0x37, 0x0000);

        delay_ms(5); //延时 5ms
        _reg = hw2000_read_reg(0x3D);    //发射状态判断，也可由 IRQ 中断实现
        while (!(_reg & 0x0001)) {
            delay_ms(5); //延时 5ms
            __reg = hw2000_read_reg(0x3D);
        }
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        hw2000_write_reg(0x36, 0x0080); // FIFO occupy disable
        hw2000_write_reg(0x3D, 0x0008); // clear INT0
        hw2000_write_reg(0x21, 0x0000); // TX disable
    }
```

2.6 HW2000 RX NOACK

以下配置实现 HW2000 最基本的 FIFO 接收功能，ACK 功能不使能。

```
    while (1) {
        hw2000_write_reg(0x36, 0x0080);   //接收 FIFO 使能
        hw2000_write_reg(0x37, 0x0000);
        hw2000_write_reg(0x21, 0x0080);   //RX enable

        _reg = hw2000_read_reg(0x3D);    //接收状态判断，也可通过 IRQ 中断实现
        while (!(_reg & 0x0001)) {
            delay_ms(5); //延时 5ms
            _reg = hw2000_read_reg(0x3D);
        }

        _reg = hw2000_read_reg(0x36);
        if (!(_reg & 0x2000)) {  //CRC 校验
            hw2000_read_fifo(0x32, _data, 1);
            hw2000_read_fifo(0x32, &_data[1], _data[0]);
        }
        hw2000_write_reg(0x3D, 0x0008);   //clear INT0
        hw2000_write_reg(0x21, 0x0000);   //RX disable
    }
```

2.7 HW2000 RX with RSSI

以下配置方式实现 HW2000 在接收数据的同时，检测当前接收数据的能量大小（RSSI 值）。请注意，在该模式下 IRQ 中断功能将被修改为 Sync word 中断检测功能，原有 IRQ 引脚功能将失效。

```
    while (1) {
        hw2000_write_reg(0x36, 0x0080);   //接收 FIFO 使能
        hw2000_write_reg(0x37, 0x0000);
        hw2000_write_reg(0x21, 0x0080);   //RX enable

        //hw2000_write_reg(0x4C, 0x55AA); //关闭内部寄存器写保护
        hw2000_write_reg(0x4F,0x 0131); //将 IRQ 中断配置为 Sync word 检测中断
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        while (IRQ 引脚产生中断);
        rssi = hw2000_read_reg(0x2D) && 0x00FF; //读取 RSSI 值

        hw2000_read_fifo(0x32, _data, 1); //先读取数据，再校验数据是否争取
        hw2000_read_fifo(0x32, &_data[1], _data[0]);

        _reg = hw2000_read_reg(0x36);
        if (!(_reg & 0x2000)) {  //CRC 校验
            判断接收数据 CRC 校验是否正确
        }
        hw2000_write_reg(0x3D, 0x0008);   //clear INT0
        hw2000_write_reg(0x21, 0x0000);   //RX disable

        hw2000_write_reg(0x4F,0x 0000); //还原 IRQ 引脚默认功能
    }
```

2.8 HW2000 TX ACK NOPAYLOAD

以下配置实现 HW2000 TX ACK 接收功能，其中 ACK PAYLOAD 不使能

```
    hw2000_write_reg(0x23, 0x0300 ); // ack times = 3
    hw2000_write_reg(0x3C, 0x1000 | 0x0001); // pipe 0 ack enable
    while (1) {
        for (i = 1; i < 16; i++) {     //TX example data
            _txdata[i] = i;
        }
        _txdata[0] = 15;

        hw2000_write_reg(0x21, 0x0100);   // TX enable
        delay_us(5); //延时>=5us
        hw2000_write_reg(0x3B, 0x8000); // FIFO write pointer clear
        hw2000_write_fifo(0x32, _txdata, _txdata[0]+1); // write data
        hw2000_write_reg(0x36, 0x0081); // fifo0, pipe0 enable , fifo0 ocupy
        //若 0x36 写入 0x0091 表示当前发送数据不需要接收端 ack 回复
        //hw2000_write_reg(0x36, 0x0091); // fifo0, pipe0 enable , fifo0 ocupy, no ack
        hw2000_write_reg(0x37, 0x0000);

        delay_ms(5); //延时 5ms
        _reg = hw2000_read_reg(0x3D);     // waiting for tx finish，也可用 IRQ 中断实现
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        while (!(_reg & 0x0001)) {
            delay_ms(5); //延时 5ms
            _reg = hw2000_read_reg(0x3D);
        }

        _reg = hw2000_read_reg(0x36);
        if (_reg & 0x8000) {
            // 重发超时，发射端未收到 ack 回复，认为发送失败
         }

        hw2000_write_reg(0x36, 0x0080); // disable all
        hw2000_write_reg(0x3D, 0x0008); // clear INT0
        hw2000_write_reg(0x21, 0x0000);   // TX disable
    }
```

2.9 HW2000 RX ACK NOPAYLOAD

以下配置实现 HW2000 RX ACK 发射功能，其中 ACK PAYLOAD 不使能

```
    hw2000_write_reg(0x23, 0x0300 ); // ack times = 3
    hw2000_write_reg(0x3C, 0x1000 | 0x0001); // pipe 0 ack enable
    while (1) {
        for (i = 0; i < 64; i++) { //clear rx buffer
            _rxdata[i] = 0;
        }

        hw2000_write_reg(0x36, 0x0080); //FIFO0 config
        hw2000_write_reg(0x37, 0x0000); //FIFO1 bypass
        hw2000_write_reg(0x21, 0x0080); //RX enable

        _reg = hw2000_read_reg(0x3D);     //Waiting for rx finish，也可用 IRQ 中断实现
        while (!(_reg & 0x0001)) {
            delay_ms(5); //延时 5ms
            _reg = hw2000_read_reg(0x3D);
        }
        _reg = hw2000_read_reg(0x36);
        if (!(_reg & 0x2000)) {  //CRC 校验
            hw2000_read_fifo(0x32, _data, 1);
             hw2000_read_fifo(0x32, &_data[1], _data[0]);
        }
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        hw2000_write_reg(0x3D, 0x0008); //Clear INT0
        hw2000_write_reg(0x21, 0x0000); //RX disable
    }
```

2.10 HW2000 TX ACK PAYLOAD

以下配置实现 HW2000 TX ACK 接收功能，并使能 ACK PAYLOAD

```
    hw2000_write_reg(0x23, 0x0300 ); // ack times = 3
    hw2000_write_reg(0x3C, 0x1000 | 0x0011); // pipe 0 ack, ackpayload enable

    while (1) {
        for (i = 1; i < 16; i++) { //tx   example data
            _txdata[i] = i;
        }
        _txdata[0] = 15;

        for (i = 0; i < 32; i++) { // clear rxack buffer
            _rxackdata[i] = 0;
        }

        hw2000_write_reg(0x21, 0x0100);   // TX enable
        delay_us(5); //延时>=5us
        hw2000_write_reg(0x3B, 0x8000); // FIFO write pointer clear
        hw2000_write_fifo(0x32, _txdata, _txdata[0]+1); // write data

        hw2000_write_reg(0x36, 0x0081); // fifo0, pipe0 enable , fifo0 ocupy
        //若 0x36 写入 0x0091 表示当前发送数据不需要接收端 ack 回复
        //hw2000_write_reg(0x36, 0x0091); // fifo0, pipe0 enable , fifo0 ocupy, no ack
        hw2000_write_reg(0x37, 0x0000);
        hw2000_write_reg(0x38, 0x0080); // ACKFIFO0 config
        hw2000_write_reg(0x39, 0x0000); // ACKFIFO1 disable

        delay_ms(5); //延时 5ms
        _reg = hw2000_read_reg(0x3D);// waiting for tx finish，也可用 IRQ 中断方式实现
        while (!(_reg & 0x0001)) {
            delay_ms(5); //延时 5ms
            _reg = hw2000_read_reg(0x3D);
        }
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        if (hw2000_read_reg(0x36) & 0x8000) {
            //重发超时异常，用户自行处理
        } else if ((_reg & 0x0102) == 0x0102) { // ackint0 and ack with payload
            hw2000_read_fifo(0x34, _rxackdata, 1);
            hw2000_read_fifo(0x34, &_rxackdata[1], _rxackdata[0]);
            //ACK payload 接收数据
        }

        hw2000_write_reg(0x36, 0x0080); // disable all
        hw2000_write_reg(0x3D, 0x0808); // clear INT0 and ACKINT0
        hw2000_write_reg(0x21, 0x0000);   // TX disable
    }
```

2.11 HW2000 RX ACK PAYLOAD

以下配置实现 HW2000 RX ACK 发射功能，并使能 ACK PAYLOAD

```
    hw2000_write_reg(0x23, 0x0300); // ack times = 3
    hw2000_write_reg(0x3C, 0x1000 | 0x0011); // pipe 0 ack, ackpayload enable
    while (1) {
        for (i = 0; i < 64; i++) { // clear buffer
            _rxdata[i] = 0;
        }
        for (i = 0; i < 32; i++) { //ack payload example data
            _txackdata[i] = 32-i;
        }
        _txackdata[0] = 31;

        hw2000_write_reg(0x21, 0x0080); //RX enable，若需写入 FIFO 数据请先打开接收

        //提前写入 ack payload 然后再通过使能接收 FIFO 开始接收
        hw2000_write_reg(0x36, 0x0000); //FIFO0 bypass
        hw2000_write_reg(0x37, 0x0000); //FIFO1 bypass
        hw2000_write_reg(0x3B, 0x4000); //ACK FIFO clear
        hw2000_write_fifo(0x34, _txackdata, _txackdata[0]+1); // write ack data
        hw2000_write_reg(0x38, 0x0180); //ACKFIFO0 config
        hw2000_write_fifo(0x35, _txackdata, _txackdata[0]+1); // write ack data
        hw2000_write_reg(0x39, 0x0180); //ACKFIFO1 config
        //请往 ACKFIFO0 和 ACKFIFO1 写入同样数据，由状态机自动选取发送数据
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
        hw2000_write_reg(0x36, 0x0080); //FIFO0 enable

        delay_ms(5); //延时 5ms
        _reg = hw2000_read_reg(0x3D);     //Waiting for rx end，也可用 IRQ 中断方式实现
        while (!(_reg & 0x0001)) {
            delay_ms(5); //延时 5ms
            _reg = hw2000_read_reg(0x3D);
        }

        _reg = hw2000_read_reg(0x36);
        if (!(_reg & 0x2000)) {  //CRC 校验
            hw2000_read_fifo(0x32, _data, 1);
             hw2000_read_fifo(0x32, &_data[1], _data[0]);
        }

        hw2000_write_reg(0x38, 0x0080); //clear ackfifo0_ocupy
        hw2000_write_reg(0x39, 0x0080); //clear ackfifo1_ocupy
        hw2000_write_reg(0x3D, 0x8808); //Clear INT0, ACKINT0，ACKINT1
        hw2000_write_reg(0x21, 0x0000); //RX disable
    }
```

2.12 HW2000 ACK PAYLOAD模式特别说明

在 ACK PAYLOAD 功能使能模式下，因 ACKINTn(0x3D 寄存器)表征的是前一次接收数据时回复之 ACK PAYLOAD 是否成功，当接收到 PID 信号时对应 ACKINTn 产生，而 INTn 中断需要接收完整数据帧后才能产生。因此，在使能 ACK PAYLOAD 功能时，每次回复时需要同时使能 ACKFIFO0 和 ACKFIFO1，并写入相同返回数据，由状态机自动选取满足条件 ACKFIFO 发送数据。

2.13 状态查询模式特别说明

当通过 SPI 轮询方式检测 HW2000 收发状态是否完成时，在轮询相关状态寄存器时请加入适当延时，以避免 SPI 通信所产生的高次谐波可能对收发过程产生影响。

采用 IRQ 中断方式实现收发状态检测则无此问题产生。

版权所有©上海东软载波微电子有限公司 http://www.essemi.com