SPI Program Examples

8051 Microcontrollers

Application Note

1. Introduction

This Application Note provides to customers C and Assembler program examples for SPI.

These examples are developped for the different configuration modes of this feature.

1.1 References

- Atmel 8051 Microcontrollers Hardware Manual

Rev. 4348A–8051–06/04

2. C Examples

2.1 Master with Slaves Select

```
/**
 * @file $RCSfile: spi_master_ss.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use spi in master mode.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0 $ $Name:  $
 */
/* @section  I N C L U D E S */
#include "reg_c51.h"
char serial_data;
char data_example=0x55;
char data_save;
bit transmit_completed= 0;
/**
 * FUNCTION_PURPOSE: This file set up spi in master mode with
 * Fclk Periph/128 as baud rate and with slave select pin.
 * FUNCTION_INPUTS: P1.5(MISO) serial input
 * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
 */
void main(void)
{
SPCON |= 0x10;                /* Master mode */
P1_1=1;                       /* enable master */
SPCON |= 0x82;                /* Fclk Periph/128 */
SPCON &= ~0x08;               /* CPOL=0; transmit mode example */
SPCON |= 0x04;                /* CPHA=1; transmit mode example */
IEN1 |= 0x04;                 /* enable spi interrupt */
SPCON |= 0x40;                /* run spi */
EA=1;                         /* enable interrupts */
  while(1)   /* endless  */
   {
   SPDAT=data_example;        /* send an example data */
   while(!transmit_completed);/* wait end of transmition */
   transmit_completed = 0;    /* clear software transfert flag */
   SPDAT=0x00;                /* data is send to generate SCK signal */
   while(!transmit_completed);/* wait end of transmition */
   transmit_completed = 0;    /* clear software transfert flag */
   data_save = serial_data;   /* save receive data */
   }
}
```

```
/**
 * FUNCTION_PURPOSE:interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: transmit_complete is software transfert flag
 */
void it_SPI(void) interrupt 9 /* interrupt address is 0x004B */
{
  switch( SPSTA )         /* read and clear spi status register */
  {
    case 0x80:
         serial_data=SPDAT;   /* read receive data */
         transmit_completed=1;/* set software flag */
    break;

    case 0x10:
         /* put here for mode fault tasking */
    break;

    case 0x40:
         /* put here for overrun tasking */
    break;
  }
}
```

2.2 Slave with Slave Select

```
/**
 * @file $RCSfile: spi_slave_ss.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use spi in slave mode.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name:  $
 */

/* @section  I N C L U D E S */
#include "reg_c51.h"

bit transmit_completed;
char serial_data;
```

```
/**
 * FUNCTION_PURPOSE: This file set up spi in slave mode with
 * Fclk Periph/128 as baud rate and with slave select pin
 * FUNCTION_INPUTS: P1.5(MISO) serial input
 *                  P1.1(/SS)=0 slave selected
 * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
 */
void main(void)
{
SPCON &= ~0x10;               /* slave mode */
SPCON &= ~0x08;               /* CPOL=0; transmit mode example*/
SPCON |= 0x04;                /* CPHA=1; transmit mode example*/
IEN1 |= 0x04;                 /* enable spi interrupt */
SPCON |= 0x40;                /* spi run */
transmit_completed = 0;       /* clear software transfert flag */
EA=1;                         /* enable interrupts */

while(1)        /* endless  */
{
if(transmit_completed)
   {
   SPDAT = serial_data;       /* echo data to master */
   transmit_completed = 0;    /* clear software transfert flag */
   }
}
}

/**
 * FUNCTION_PURPOSE: spi interrupt, receive data to master
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void it_SPI(void) interrupt 9 /* interrupt address is 0x004B */
{
  switch( SPSTA )         /* read and clear spi status register */
  {
    case 0x80:
         serial_data=SPDAT;   /* read receive data */
         transmit_completed=1;/* set software flag */
    break;
    case 0x10:
         /* put here for mode fault tasking */
    break;
    case 0x40:
         /* put here for overrun tasking */
    break;
  }
   SPDAT=serial_data;        /* needed to complete clearing sequence */
}
```

2.3 Master without Slave Select

```
/**
 * @file $RCSfile: spi_master_no_ss.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use spi in master mode.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0 $ $Name:  $
 */
/* @section  I N C L U D E S */
#include "reg_c51.h"
char serial_data;
char data_example=0x55;
char data_save;
bit transmit_completed= 0;
/**
 * FUNCTION_PURPOSE: This file set up spi in master mode with
 * Fclk Periph/128 as baud rate  and without slave select pin
 * FUNCTION_INPUTS: P1.5(MISO) serial input
 * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
 *                    P1.1
 */
void main(void)
{
SPCON |= 0x10;                /* Master mode */
SPCON |= 0x82;                /* Fclk Periph/128 */
SPCON |= 0x20;                /* P1.1 is available as standard I/O pin */
SPCON &= ~0x08;               /* CPOL=0; transmit mode example */
SPCON |= 0x04;                /* CPHA=1; transmit mode example */
IEN1 |= 0x04;                 /* enable spi interrupt */
SPCON |= 0x40;                /* run spi */
EA=1;                         /* enable interrupts */
  while(1)   /* endless  */
   {
   P1_1=~P1_1;                /* P1.1 is available as standard I/O pin */
   SPDAT=data_example;        /* send an example data */
   while(!transmit_completed);/* wait end of transmition */
   transmit_completed = 0;    /* clear software transfert flag */
   SPDAT=0x00;                /* data is send to generate SCK signal */
   while(!transmit_completed);/* wait end of transmition */
   transmit_completed = 0;    /* clear software transfert flag */
   data_save = serial_data;   /* save receive data */
   }
}
```

```
/**
 * FUNCTION_PURPOSE:interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: transmit_complete is software transfert flag
 */
void it_SPI(void) interrupt 9 /* interrupt address is 0x004B */
{
  switch( SPSTA )         /* read and clear spi status register */
  {
    case 0x80:
         serial_data=SPDAT;   /* read receive data */
         transmit_completed=1;/* set software flag */
    break;
    case 0x10:
         /* put here for mode fault tasking */
    break;
    case 0x40:
         /* put here for overrun tasking */
    break;
  }
}
```

2.4 Slave without Slave Select

```
/**
 * @file $RCSfile: spi_slave_no_ss.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use spi in slave mode.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name:  $
 */

/* @section  I N C L U D E S */
#include "reg_c51.h"

bit transmit_completed;
char serial_data;
```

```
/**
 * FUNCTION_PURPOSE: This file set up spi in slave mode with
 * Fclk Periph/128 as baud rate and without slave select pin.
 * FUNCTION_INPUTS: P1.5(MISO) serial input
 * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
 *                  P1.1
 */
void main(void)
{
SPCON |= 0x20;                /* P1.1 is available as standard I/O pin */
/* SPCON.5(SSDIS) has no effect if CPHA=0 in slave mode then P1.1 is used to
slave select */
SPCON &= ~0x10;               /* slave mode */
SPCON &= ~0x08;               /* CPOL=0; transmit mode example*/
SPCON |= 0x04;                /* CPHA=1; transmit mode example*/
IEN1 |= 0x04;                 /* enable spi interrupt */
SPCON |= 0x40;                /* spi run */
transmit_completed = 0;       /* clear software transfert flag */
EA=1;                         /* enable interrupts */
while(1)        /* endless  */
{
P1_1=~P1_1;                   /* P1.1 is available as standard I/O pin */
if(transmit_completed)
   {
   SPDAT = serial_data;       /* echo data to master */
   transmit_completed = 0;    /* clear software transfert flag */
   }
}
}
/**
 * FUNCTION_PURPOSE: spi interrupt, receive data to master
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void it_SPI(void) interrupt 9 /* interrupt address is 0x004B */
{
  switch( SPSTA )         /* read and clear spi status register */
  {case 0x80:
         serial_data=SPDAT;   /* read receive data */
         transmit_completed=1;/* set software flag */
    break;
    case 0x10:
         /* put here for mode fault tasking */
    break;
    case 0x40:
         /* put here for overrun tasking */
    break;
  }
   SPDAT=serial_data;         /* needed to complete clearing sequence */
}
```

2.5 SFR Register Definition

```
/*H***************************************************************************
**
* NAME: reg_c51.h
*----------------------------------------------------------------------------
-
* PURPOSE: SFR Description file for 8051 products
*      ON KEIL compiler
*****************************************************************************
*/

#define Sfr(x, y)  sfr x = y
#define Sbit(x, y, z)   sbit x = y^z
#define Sfr16(x,y)    sfr16 x = y

/*----------------------------------------*/
/* Include file for 8051 SFR Definitions  */
/*----------------------------------------*/

/*  BYTE Register  */
Sfr (P0 , 0x80);

Sbit (P0_7 , 0x80, 7);
Sbit (P0_6 , 0x80, 6);
Sbit (P0_5 , 0x80, 5);
Sbit (P0_4 , 0x80, 4);
Sbit (P0_3 , 0x80, 3);
Sbit (P0_2 , 0x80, 2);
Sbit (P0_1 , 0x80, 1);
Sbit (P0_0 , 0x80, 0);

Sfr (P1 , 0x90);

Sbit (P1_7 , 0x90, 7);
Sbit (P1_6 , 0x90, 6);
Sbit (P1_5 , 0x90, 5);
Sbit (P1_4 , 0x90, 4);
Sbit (P1_3 , 0x90, 3);
Sbit (P1_2 , 0x90, 2);
Sbit (P1_1 , 0x90, 1);
Sbit (P1_0 , 0x90, 0);



Sfr (P2 , 0xA0);
Sbit (P2_7 , 0xA0, 7);
Sbit (P2_6 , 0xA0, 6);
Sbit (P2_5 , 0xA0, 5);
Sbit (P2_4 , 0xA0, 4);
Sbit (P2_3 , 0xA0, 3);
Sbit (P2_2 , 0xA0, 2);
Sbit (P2_1 , 0xA0, 1);
Sbit (P2_0 , 0xA0, 0);


Sfr (P3 , 0xB0);

Sbit (P3_7 , 0xB0, 7);
Sbit (P3_6 , 0xB0, 6);
Sbit (P3_5 , 0xB0, 5);
Sbit (P3_4 , 0xB0, 4);
Sbit (P3_3 , 0xB0, 3);
Sbit (P3_2 , 0xB0, 2);
Sbit (P3_1 , 0xB0, 1);
Sbit (P3_0 , 0xB0, 0);

Sbit (RD , 0xB0, 7);
Sbit (WR , 0xB0, 6);
Sbit (T1 , 0xB0, 5);
Sbit (T0 , 0xB0, 4);
Sbit (INT1 , 0xB0, 3);
Sbit (INT0 , 0xB0, 2);
Sbit (TXD , 0xB0, 1);
Sbit (RXD , 0xB0, 0);

Sfr (P4 , 0xC0);
Sbit (P4_7 , 0xC0, 7);
Sbit (P4_6 , 0xC0, 6);
Sbit (P4_5 , 0xC0, 5);
Sbit (P4_4 , 0xC0, 4);
Sbit (P4_3 , 0xC0, 3);
Sbit (P4_2 , 0xC0, 2);
Sbit (P4_1 , 0xC0, 1);
Sbit (P4_0 , 0xC0, 0);

Sfr (P5 , 0xE8);
Sbit (P5_7 , 0xE8, 7);
Sbit (P5_6 , 0xE8, 6);
Sbit (P5_5 , 0xE8, 5);
Sbit (P5_4 , 0xE8, 4);
Sbit (P5_3 , 0xE8, 3);
Sbit (P5_2 , 0xE8, 2);
Sbit (P5_1 , 0xE8, 1);
Sbit (P5_0 , 0xE8, 0);


Sfr (PSW , 0xD0);

Sbit (CY  , 0xD0  , 7);
Sbit (AC  , 0xD0  , 6);
Sbit (F0  , 0xD0  , 5);
Sbit (RS1 , 0xD0  , 4);
Sbit (RS0 , 0xD0  , 3);
Sbit (OV  , 0xD0  , 2);
Sbit (UD  , 0xD0  , 1);
Sbit (P   , 0xD0  , 0);

Sfr (ACC , 0xE0);
Sfr (B , 0xF0);
Sfr (SP , 0x81);
Sfr (DPL , 0x82);
Sfr (DPH , 0x83);

Sfr (PCON , 0x87);
Sfr (CKCON0 , 0x8F);
Sfr (CKCON1 , 0xAF);

/*------------------ TIMERS registers ---------------------*/
Sfr (TCON , 0x88);
Sbit (TF1 , 0x88, 7);
Sbit (TR1 , 0x88, 6);
Sbit (TF0 , 0x88, 5);
Sbit (TR0 , 0x88, 4);
Sbit (IE1 , 0x88, 3);
Sbit (IT1 , 0x88, 2);
Sbit (IE0 , 0x88, 1);
Sbit (IT0 , 0x88, 0);

Sfr (TMOD , 0x89);

Sfr  (T2CON , 0xC8);
Sbit (TF2   , 0xC8, 7);
Sbit (EXF2  , 0xC8, 6);
Sbit (RCLK  , 0xC8, 5);
Sbit (TCLK  , 0xC8, 4);
Sbit (EXEN2 , 0xC8, 3);
Sbit (TR2   , 0xC8, 2);
Sbit (C_T2  , 0xC8, 1);
Sbit (CP_RL2, 0xC8, 0);

Sfr (T2MOD , 0xC9);
Sfr (TL0 , 0x8A);
Sfr (TL1 , 0x8B);
Sfr (TL2 , 0xCC);
Sfr (TH0 , 0x8C);
Sfr (TH1 , 0x8D);
Sfr (TH2 , 0xCD);
Sfr (RCAP2L , 0xCA);
Sfr (RCAP2H , 0xCB);
Sfr (WDTRST , 0xA6);
Sfr (WDTPRG , 0xA7);


/*------------------- UART registers ------------------------*/
Sfr (SCON , 0x98);
Sbit (SM0  , 0x98, 7);
Sbit (FE   , 0x98, 7);
Sbit (SM1  , 0x98, 6);
Sbit (SM2  , 0x98, 5);
Sbit (REN  , 0x98, 4);
Sbit (TB8  , 0x98, 3);
Sbit (RB8  , 0x98, 2);
Sbit (TI   , 0x98, 1);
Sbit (RI   , 0x98, 0);

Sfr (SBUF , 0x99);
Sfr (SADEN , 0xB9);
Sfr (SADDR , 0xA9);

/*-------------------- Internal Baud Rate Generator --------*/
Sfr (BRL  , 0x9A);
Sfr (BDRCON , 0x9B);



/*-------------------- IT registers -----------------------*/
Sfr (IEN0  , 0xA8);
Sfr (IEN1  , 0xB1);
Sfr (IPH0 , 0xB7);
Sfr (IPH1 , 0xB3);
Sfr (IPL0 , 0xB8);
Sfr (IPL1 , 0xB2);



/*  IEN0  */
Sbit (EA   , 0xA8, 7);
Sbit (EC   , 0xA8, 6);
Sbit (ET2  , 0xA8, 5);
Sbit (ES   , 0xA8, 4);
Sbit (ET1  , 0xA8, 3);
Sbit (EX1  , 0xA8, 2);
Sbit (ET0  , 0xA8, 1);
Sbit (EX0  , 0xA8, 0);


/*--------------------- PCA registers -----------------------------*/
Sfr (CCON , 0xD8);
Sfr (CMOD , 0xD9);
Sfr (CH , 0xF9);
Sfr (CL , 0xE9);
Sfr (CCAP0H  , 0xFA);
Sfr (CCAP0L  , 0xEA);
Sfr (CCAPM0  , 0xDA);
Sfr (CCAP1H  , 0xFB);
Sfr (CCAP1L  , 0xEB);
Sfr (CCAPM1  , 0xDB);
Sfr (CCAP2H  , 0xFC);
Sfr (CCAP2L  , 0xEC);
Sfr (CCAPM2  , 0xDC);
Sfr (CCAP3H  , 0xFD);
Sfr (CCAP3L  , 0xED);
Sfr (CCAPM3  , 0xDD);
Sfr (CCAP4H  , 0xFE);
Sfr (CCAP4L  , 0xEE);
Sfr (CCAPM4  , 0xDE);
/* CCON */
Sbit (CF   , 0xD8, 7);
Sbit (CR   , 0xD8, 6);

Sbit (CCF4   , 0xD8, 4);
Sbit (CCF3  , 0xD8, 3);
Sbit (CCF2  , 0xD8, 2);
Sbit (CCF1  , 0xD8, 1);
Sbit (CCF0  , 0xD8, 0);


/*------------------ T W I registers ------------------------------*/
Sfr ( SSCON , 0x93);
Sfr ( SSCS , 0x94);
Sfr ( SSDAT , 0x95);
Sfr ( SSADR , 0x96);
Sfr ( PI2, 0xF8);
Sbit (PI2_1  , 0xF8, 1);
Sbit (PI2_0  , 0xF8, 0);

/*-------------------- OSC control registers ----------------------*/
Sfr ( CKSEL , 0x85 );
Sfr ( OSCCON , 0x86 );
Sfr ( CKRL , 0x97 );

/*-------------------- Keyboard control registers -----------------*/
Sfr ( KBLS , 0x9C );
Sfr ( KBE , 0x9D );
Sfr ( KBF , 0x9E );
/*-------------------- SPI --------------------- -----------------*/
Sfr ( SPCON, 0xC3 );
Sfr ( SPSTA, 0xC4 );
Sfr ( SPDAT, 0xC5 );
```

```
/*------ Misc ----------------------------------------------------*/
Sfr( AUXR , 0x8E);
Sfr ( AUXR1, 0xA2);
Sfr ( FCON, 0xD1);


/*------ E data --------------------------------------------------*/

Sfr ( EECON,  0xD2 );
```

3. Assembler 51 Examples

3.1 Master with Slave Select

```
$INCLUDE   (reg_c51.INC)
transmit_completed BIT 20H.1; software flag
serial_data DATA 08H
data_save DATA 09H
data_example DATA 0AH;

org 000h
ljmp begin

org 4Bh
ljmp it_SPI

;/**
; * FUNCTION_PURPOSE: This file set up spi in master mode with
; * Fclk Periph/128 as baud rate and with slave select pin.
; * FUNCTION_INPUTS: P1.5(MISO) serial input
; * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
; */
org 0100h
begin:

;init
MOV data_example,#55h;           /* data example */

ORL SPCON,#10h;                  /* Master mode */
SETB P1.1;                       /* enable master */
ORL SPCON,#82h;                  /* Fclk Periph/128 */
ANL SPCON,#0F7h;                 /* CPOL=0; transmit mode example */
ORL SPCON,#04h;                  /* CPHA=1; transmit mode example */
ORL IEN1,#04h;                   /* enable spi interrupt */
ORL SPCON,#40h;                  /* run spi */
CLR transmit_completed;          /* clear software transfert flag */
SETB EA;                         /* enable interrupts */

loop:                            /* endless */

   MOV SPDAT,data_example;       /* send an example data */
   JNB transmit_completed,$;     /* wait end of transmition */
   CLR transmit_completed;       /* clear software transfert flag */

   MOV SPDAT,#00h;               /* data is send to generate SCK signal */
   JNB transmit_completed,$;     /* wait end of transmition */
   CLR transmit_completed;       /* clear software transfert flag */
   MOV data_save,serial_data;    /* save receive data */

LJMP loop
```

```
;/**
; * FUNCTION_PURPOSE:interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: transmit_complete is software transfert flag
; */
it_SPI:;                         /* interrupt address is 0x004B */
MOV R7,SPSTA;
MOV ACC,R7
JNB ACC.7,break1;case 0x80:
    MOV serial_data,SPDAT;       /* read receive data */
    SETB transmit_completed;     /* set software flag */
break1:
JNB ACC.4,break2;case 0x10:
;         /* put here for mode fault tasking */
break2:;
JNB ACC.6,break3;case 0x40:
;         /* put here for overrun tasking */
break3:;
RETI

end
```

3.2 Slave with Slave Select

```
$INCLUDE   (reg_c51.INC)
transmit_completed BIT 20H.1; software flag
serial_data DATA 08H

org 000h
ljmp begin

org 4Bh
ljmp it_SPI

;/**
; * FUNCTION_PURPOSE: This file set up spi in slave mode with
; * Fclk Periph/128 as baud rate and with slave select pin
; * FUNCTION_INPUTS: P1.5(MISO) serial input
; *                    P1.1(/SS)=0 slave selected
; * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
; */
org 0100h
begin:

ANL SPCON,#0EFh;                 /* slave mode */
ANL SPCON,#0F7h;                 /* CPOL=0; transmit mode example */
ORL SPCON,#04h;                  /* CPHA=1; transmit mode example */
ORL IEN1,#04h;                   /* enable spi interrupt */
ORL SPCON,#40h;                  /* run spi */
CLR transmit_completed;          /* clear software transfert flag */
SETB EA;                         /* enable interrupts */
```

```
loop:                /* endless  */
JNB transmit_completed,end_if

   MOV SPDAT,serial_data;        /* echo data to master */
   CLR transmit_completed;       /* clear software transfert flag */

end_if:
LJMP loop



;/**
; * FUNCTION_PURPOSE: spi interrupt, receive data to master
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
it_SPI:;                        /* interrupt address is 0x004B */

MOV R7,SPSTA;
MOV ACC,R7
JNB ACC.7,break1;case 0x80:
    MOV serial_data,SPDAT;       /* read receive data */
    SETB transmit_completed;     /* set software flag */
break1:

JNB ACC.4,break2;case 0x10:
;         /* put here for mode fault tasking */
break2:;

JNB ACC.6,break3;case 0x40:
;         /* put here for overrun tasking */
break3:;

MOV SPDAT,serial_data;        /* needed to complete clearing sequence */

RETI

end
```

3.3 Master without Slave Select

```
$INCLUDE   (reg_c51.INC)
transmit_completed BIT 20H.1; software flag
serial_data DATA 08H
data_save DATA 09H
data_example DATA 0AH;

org 000h
ljmp begin

org 4Bh
ljmp it_SPI

;/**
; * FUNCTION_PURPOSE: This file set up spi in master mode with
; * Fclk Periph/128 as baud rate  and without slave select pin
; * FUNCTION_INPUTS: P1.5(MISO) serial input
; * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
; *                   P1.1
; */
org 0100h
begin:

;init
MOV data_example,#55h;           /* data example */

ORL SPCON,#10h;                  /* Master mode */
ORL SPCON,#20h;                  /* P1.1 is available as standard I/O pin */
ORL SPCON,#82h;                  /* Fclk Periph/128 */
ANL SPCON,#0F7h;                 /* CPOL=0; transmit mode example */
ORL SPCON,#04h;                  /* CPHA=1; transmit mode example */
ORL IEN1,#04h;                   /* enable spi interrupt */
ORL SPCON,#40h;                  /* run spi */
CLR transmit_completed;          /* clear software transfert flag */
SETB EA;                         /* enable interrupts */

loop:                            /* endless */

   CPL P1.1;                     /* P1.1 is available as standard I/O pin */
   MOV SPDAT,data_example;       /* send an example data */
   JNB transmit_completed,$;     /* wait end of transmition */
   CLR transmit_completed;       /* clear software transfert flag */

   MOV SPDAT,#00h;               /* data is send to generate SCK signal */
   JNB transmit_completed,$;     /* wait end of transmition */
   CLR transmit_completed;       /* clear software transfert flag */
   MOV data_save,serial_data;    /* save receive data */

LJMP loop
```

```
;/**
; * FUNCTION_PURPOSE:interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: transmit_complete is software transfert flag
; */
it_SPI:;                        /* interrupt address is 0x004B */
MOV R7,SPSTA;
MOV ACC,R7
JNB ACC.7,break1;case 0x80:
    MOV serial_data,SPDAT;      /* read receive data */
    SETB transmit_completed;    /* set software flag */
break1:
JNB ACC.4,break2;case 0x10:
;         /* put here for mode fault tasking */
break2:;
JNB ACC.6,break3;case 0x40:
;         /* put here for overrun tasking */
break3:;
RETI

end
```

3.4 Slave without Slave Select

```
$INCLUDE   (reg_c51.INC)
transmit_completed BIT 20H.1; software flag
serial_data DATA 08H

org 000h
ljmp begin

org 4Bh
ljmp it_SPI

;/**
; * FUNCTION_PURPOSE: This file set up spi in slave mode with
; * Fclk Periph/128 as baud rate and without slave select pin.
; * FUNCTION_INPUTS: P1.5(MISO) serial input
; * FUNCTION_OUTPUTS: P1.7(MOSI) serial output
; *                   P1.1
; */
org 0100h
begin:

ORL SPCON,#20h;                 /* P1.1 is available as standard I/O pin */
;/* SPCON.5(SSDIS) has no effect if CPHA=0 in slave mode then P1.1 is used to
slave select */
ANL SPCON,#0EFh;                /* slave mode */
ANL SPCON,#0F7h;                /* CPOL=0; transmit mode example */
ORL SPCON,#04h;                 /* CPHA=1; transmit mode example */
ORL IEN1,#04h;                   /* enable spi interrupt */
ORL SPCON,#40h;                  /* run spi */
CLR transmit_completed;          /* clear software transfert flag */
SETB EA;                         /* enable interrupts */


loop:                /* endless  */
CPL P1.1;                        /* P1.1 is available as standard I/O pin */
JNB transmit_completed,end_if

   MOV SPDAT,serial_data;        /* echo data to master */
   CLR transmit_completed;       /* clear software transfert flag */

end_if:
LJMP loop


;/**
; * FUNCTION_PURPOSE: spi interrupt, receive data to master
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
it_SPI:;                         /* interrupt address is 0x004B */

MOV R7,SPSTA;
MOV ACC,R7
JNB ACC.7,break1;case 0x80:
    MOV serial_data,SPDAT;       /* read receive data */
    SETB transmit_completed;     /* set software flag */
break1:

JNB ACC.4,break2;case 0x10:
;         /* put here for mode fault tasking */
break2:;

JNB ACC.6,break3;case 0x40:
;         /* put here for overrun tasking */
break3:;

MOV SPDAT,serial_data;        /* needed to complete clearing sequence */

RETI

end
```

3.5 SFR Register Definition

```
$SAVE
$NOLIST


P0      DATA    80H
TCONDATA88H
;---  TCON Bits ---
TF1     BIT     8FH
TR1     BIT     8EH
TF0     BIT     8DH
TR0     BIT     8CH
IE1     BIT     8BH
IT1     BIT     8AH
IE0     BIT     89H
IT0     BIT     88H

P1      DATA    90H

SCON    DATA    98H
;--- SCON Bits ----
SM0     BIT     9FH
SM1     BIT     9EH
SM2     BIT     9DH
REN     BIT     9CH
TB8     BIT     9BH
RB8     BIT     9AH
TI      BIT     99H
RI      BIT     98H


P2      DATA    0A0H
IEN0    DATA    0A8H
;--- IEN0 Bits -----
EA   BIT0AFH
EC   BIT0AEH
ET2  BIT0ADH
ES  BIT0ACH
ET1 BIT0ABH
EX1  BIT0AAH
ET0 BIT0A9H
EX0BIT0A8H

P3      DATA    0B0H
;--- P3 Bits -------
RD      BIT     0B7H
WR      BIT     0B6H
T1      BIT     0B5H
T0      BIT     0B4H
INT1    BIT     0B3H
```

```
INT0    BIT     0B2H
TXD     BIT     0B1H
RXD     BIT     0B0H


P4      DATA    0C0H
P5      DATA    0E8H

IPL0DATA0B8H
;--- IPL0 Bits -----
PPCL   BIT0BEH
PT2L  BIT0BDH
PSL  BIT0BCH
PT1L BIT0BBH
PX1L  BIT0BAH
PT0L BIT0B9H
PX0LBIT0B8H


T2CON   DATA    0C8H
;--- T2CON bits ----
TF2     BIT     0CFH
EXF2    BIT     0CEH
RCLK    BIT     0CDH
TCLK    BIT     0CCH
EXEN2   BIT     0CBH
TR2     BIT     0CAH
C_T2    BIT     0C9H
CP_RL2  BIT     0C8H

PSW     DATA    0D0H
;--- PSW bits ------
CY      BIT     0D7H
AC      BIT     0D6H
F0      BIT     0D5H
RS1     BIT     0D4H
RS0     BIT     0D3H
OV      BIT     0D2H
P       BIT     0D0H

CCONDATA0D8H
;--- CCON bits -----
CF   BIT     0DFH
CR   BIT     0DEH
CCF4  BIT     0DCH
CCF3  BIT     0DBH
CCF2  BIT     0DAH
CCF1  BIT     0D9H
CCF0  BIT     0D8H
```

```
ACC     DATA    0E0H
B       DATA    0F0H


SP      DATA    81H
DPL     DATA    82H
DPH     DATA    83H
PCON    DATA    87H


TMOD    DATA    89H
TL0     DATA    8AH
TL1     DATA    8BH
TH0     DATA    8CH
TH1     DATA    8DH
AUXRDATA08EH
CKCON0DATA08Fh




SBUF    DATA    99H
;-- Baud Rate generator
BRL  DATA09AH
BDRCON  DATA 09BH
;--- Keyboard
KBLSDATA09CH
KBEDATA09DH
KBFDATA09EH





;--- Watchdog timer
WDTRSTDATA0A6H
WDTPRG DATA0A7H

SADDRDATA0A9H
CKCON1DATA0AFH


IEN1DATA0B1H
IPL1DATA0B2H
IPH1DATA0B3H
IPH0DATA0B7H


SADENDATA0B9H
```

```
T2MODDATA    0C9h
RCAP2L  DATA    0CAH
RCAP2H  DATA    0CBH
TL2     DATA    0CCH
TH2     DATA    0CDH

CMODDATA0D9H
CCAPM0DATA0DAH
CCAPM1DATA0DBH
CCAPM2DATA0DCH
CCAPM3DATA0DDH
CCAPM4DATA0DEH


CHDATA0F9H
CCAP0HDATA0FAH
CCAP1HDATA0FBH
CCAP2HDATA0FCH
CCAP3HDATA0FDH
CCAP4HDATA0FEH

CLDATA0E9H
CCAP0LDATA0EAH
CCAP1LDATA0EBH
CCAP2LDATA0ECH
CCAP3LDATA0EDH
CCAP4LDATA0EEH

; SPI
SPCON    DATA    0C3H
SPSTA    DATA    0C4H
SPDAT    DATA    0C5H


; TWI
PI2DATA    0F8h
SSCONDATA093H
SSCSDATA094H
SSDATDATA095H
SSADRDATA096H
PI2_OBIT0F8H
PI2_1BIT0F9H

; Clock Control
OSCCONDATA086H
CKSELDATA085H
CKRLDATA097H

;MISC
AUXR1DATA0A2H
```

```
; Flash control
FCON   DATA    0D1H

;EEData
EECONDATA0D2H



$RESTORE
```

Table of Contents

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com