

# SPI Data Transmission in the HT56R2x and Relevant Programming

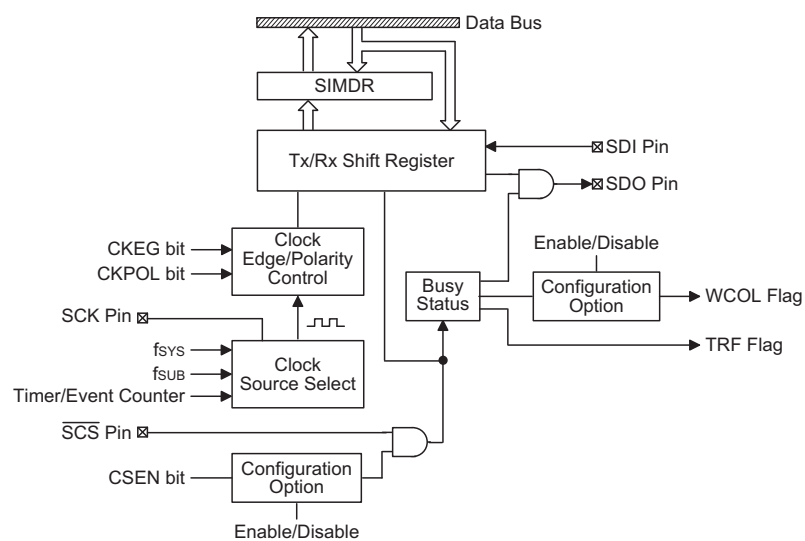
D/N : HA0213E

## Introduction

The HT56R2x includes an internal Serial Interface Function within which is included an SPI, I<sup>2</sup>C and SPI1 interface types. The SPI1 is mainly used to transmit voice communication signals. The communication and control methods for the SPI and SPI1 in the HT56R2x are similar, so the following description depicts a way of using the SPI for data transmission with the HT56R24 and any special considerations that need to be taken into account.

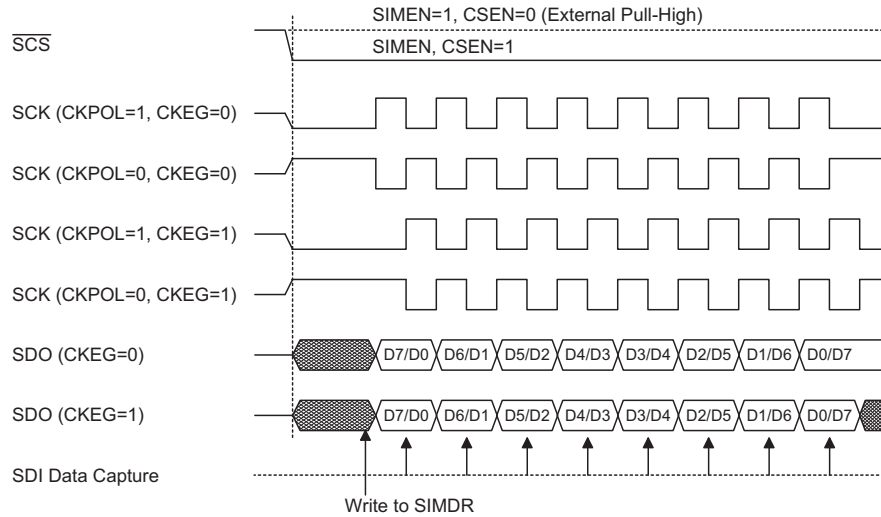
## SPI Interface

The SPI (Serial Peripheral Interface) is a full duplex synchronous serial data link which was originally developed by Motorola to allow communication with external devices. Using a master and slave technique, only the master can initiate a transmission. A simple four line SPI interface is used for all communication and all the SPI interface pins are pin-shared with normal I/O ports.

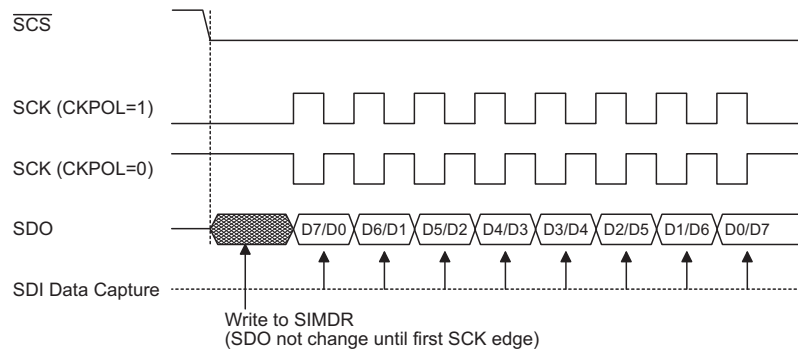


SPI Block Diagram

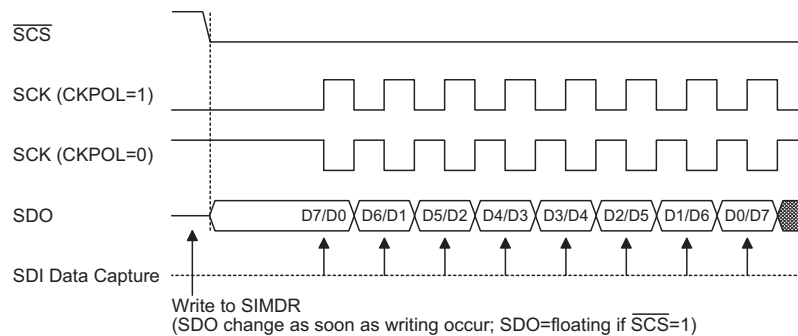
The SPI interface is a full duplex synchronous serial data link, a four line interface with pin names SDI (serial data input), SDO (serial data output), SCK (serial clock) and SCS (slave select). It should be noted that the Slave Select line is determined by the CSEN bit in the SIMCTL2 register. If the CSEN bit is set, then SCS will be active, otherwise it will be floating when the CSEN bit is cleared to 0. The following timing figure describes the timing protocol under the master/slave mode of the SPI bus.



**SPI Master Mode Timing**



**SPI Slave Mode Timing – CKEG=0**

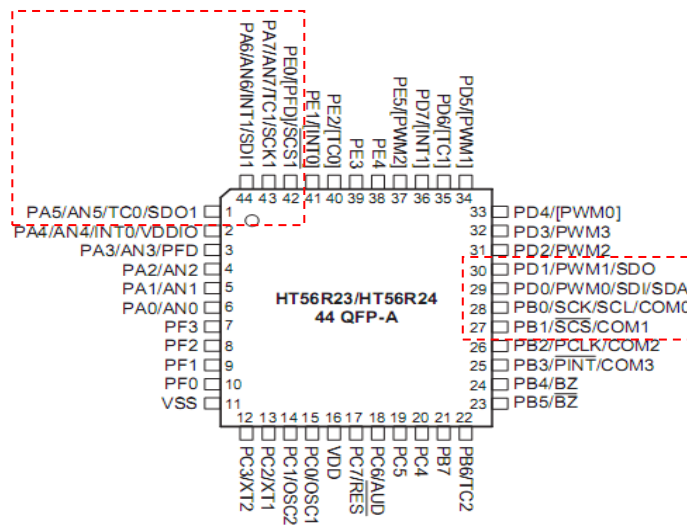


Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignore the SCS level.

**SPI Slave Mode Timing – CKEG=1**

## Control Description

As the figure below shows, the SPI interface pins are pin-shared with I/O, I<sup>2</sup>C, COM channels and PWM outputs. The SPI interface must be enabled by enabling the SIM, WCOL and CSEN configuration options and setting the correct bit in the SIMCTL0 and SIMCTL2 registers. To enable the SPI function in the Idle Mode, set the CLKMOD bit 4, SIMIDLE, to 1, which is only used in the Idle Mode to provide the SPI interface clock. (The SPI1, for operation in the idle mode, is also determined by the CLKMOD bit 4, SIMIDLE, by setting it high to enable or 0 to disable.)



There are three registers in the HT56R24 associated with the SPI function, these are SIMDR, SIMCTL0, and SIMCTL2. They are also used by the I<sup>2</sup>C functions in the HT56R24 so either the SPI or I<sup>2</sup>C must be first selected. To write data into the SPI module, it should first be placed in the SIMDR register before transmission. After the data is received from the SPI bus, the microcontroller can read it from the SIMDR register. Any transmission or reception of data from the SPI bus must be made via the SIMDR register.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>SIMDR</b>	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
<b>R/W</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**SIMDR Register**

### Data Transmission Operating Mode

The SIMCTL0 register is also used by both the SPI and I<sup>2</sup>C interface. It is used to control the enable/disable function and to set the data transmission clock frequency.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Label	SIM2	SIM1	SIM0	PCKEN	PCKPSC1	PCKPSC0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

- **SIMEN**  
This bit is the overall on/off control for the SPI interface. When the SIMEN bit is cleared to zero to disable the SPI interface, the SDI, SDO, SCK and SCS lines will be in a floating condition and the SPI operating current will be reduced to a minimum value. When the bit is 1, the SPI interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. Note that when the SIMEN bit changes from 0 to 1, the contents of the SPI control registers will be in an unknown condition and should therefore be first initialized by the application program.

- **PCKEN, PCKPSC1, PCKPSC0**  
The PCKEN, PCKPSC1, PCKPSC0 registers are used to control the peripheral clock outputs. This peripheral clock output pin PCLK is pin-shared with the I/O (PB.2) and it is derived from a divide by two division of the Timer/counter 2 or a division ratio of the internal system clock.

PCKEN: This bit is used to control the overall on/off of the peripheral clock outputs.

PCKPSC0, PCKPSC1: these two bits select the clock source of the peripheral clock outputs, such as:

PCKPSC [1:0]: Peripheral Clock Output Clock Source
00: $f_{SYS}$
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: Timer 2 output/2

- **SIM0~SIM2**  
These bits setup the overall operating mode of the SIM function. As well as selecting the I<sup>2</sup>C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master Clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the Timer/Event Counter. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device. Detailed information is shown as below:

**SIM2, SIM1, SIM0:** SIM Operating Mode Control  
 000: SPI master mode; SPI clock is  $f_{SYS}/4$   
 001: SPI master mode; SPI clock is  $f_{SYS}/16$   
 010: SPI master mode; SPI clock is  $f_{SYS}/64$   
 011: SPI master mode; SPI clock is  $f_{TBC}$   
 100: SPI master mode; SPI clock is TM0 CCRP match frequency/2  
 101: SPI slave mode  
 110: I<sup>2</sup>C slave mode  
 111: Unused mode

The SIMCTL2 register is used only by the SPI function

**SIMCTL2 Register**

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Label	—	—	CKPOL	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

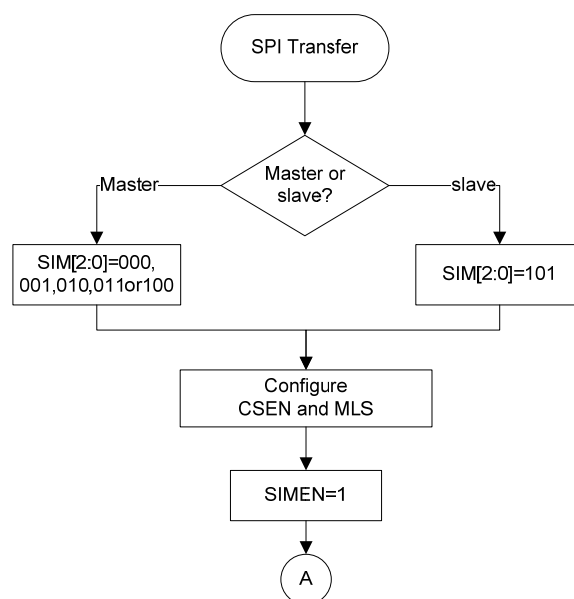
- **TRF**  
The TRF bit is the Transmit/Receive Complete flag and is set to 1 automatically when an SPI data transmission is completed, but must be cleared by the application program.
- **WCOL**  
The WCOL bit is used when data has been attempted to be written to the SIMDR register during a data transfer operation. This writing operation will be ignored if data is being transferred. The WCOL bit can be disabled or enabled by a configuration option and be cleared by the application program.

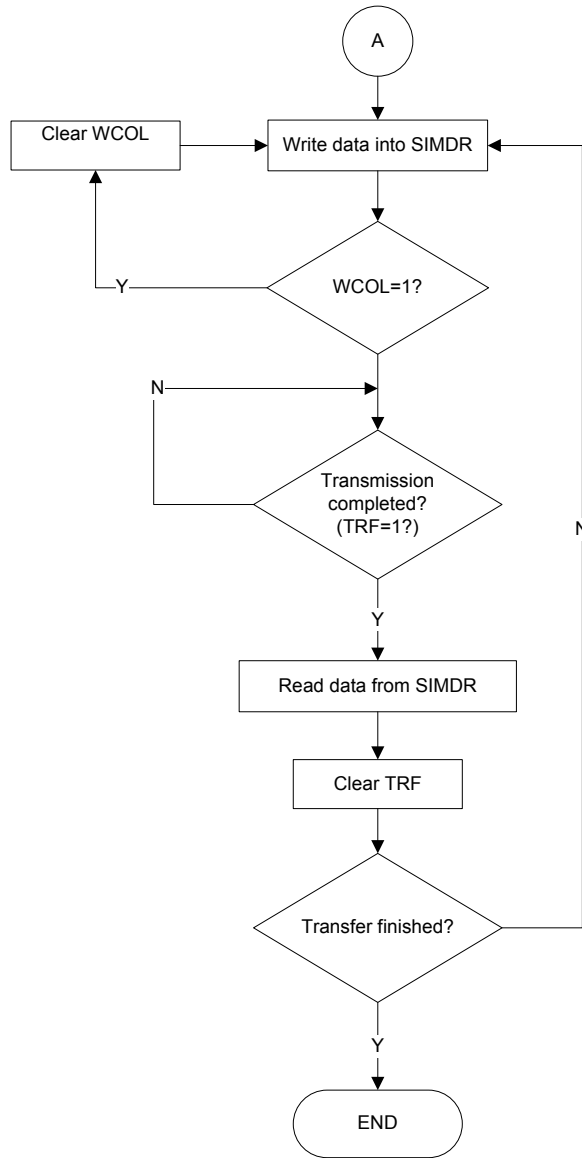
- **CSEN**  
The CSEN bit is used as an On/Off control for the SCS pin. If the bit is 1, then the SCS will be effective. In the Master Mode, an SCS pin signal will be sent out before an SCK clock signal is generated. In the Slave Mode, data transfers will be disabled/enabled before/after an SCS signal is received. If CSEN=0, then SCS will be placed into a floating state and the function disabled. A pull-high resistor should be connected to the SCS pin to implement this function. The CSEN bit can be enabled or disabled via a configuration option.
- **MLS**  
This selects either MSB or LSB.
- **CKEG and CKPOL**  
These two bits must be configured before any data transfers are executed otherwise an erroneous clock edge may be generated. The CKPOL bit determines the base condition of the clock line, if the bit is high then the SCK line will be low when the clock is inactive, the opposite is true if the bit is low. The CKEG bit determines the active clock edge type which depends on the condition of CKPOL. The following shows the condition of the combination of the two bits.

CKPOL	CKEG	SCKClock Signal
0	0	High Base Level Active Rising Edge
0	1	High Base Level Active Falling Edge
1	0	Low Base Level Active Falling Edge
1	1	Low Base Level Active Rising Edge

**S/W Flowchart**

Take the SPI data transmission as an example, the SPI (SIM) procedure flowchart (similar with the SPI1 transmission flowchart) is provided below:





SPI Transfer Control Flowchart

## **SPI Operation**

In the Master/Slave Mode, all communication can be implemented by the SPI function. The timing figure depicts the basic operation of the SPI function. During the SPI transfer, the Master device, before sending an SCK signal, will select a Slave device by sending an SCS signal. When CSEN=0, an external pull-high resistor should be added to the SCS line as shown in the SPI Master Mode Timing figure. In the Slave Mode when SCS=1, SDO will be in a floating condition and when SCS=0, SDO will be effective. When CSEN=0, whatever condition the SCS is in (high level or low level) as long as SIMEN=1, the SPI will be effective. Setting the SIMEN bit in the SIMCTL0 register high will force the SDI pin to a floating condition and the SDO pin high. The SCK pin will be in a floating condition in the Slave Mode. If the SIMEN bit is cleared to 0, then the SPI bus will be disabled, and the SCS, SDI, SDO and SCK pins will all be placed into a floating condition. In the Master Mode, the clock signal is always generated by the master device. Clock and data transfers will be activated after data is written to the SIMDR register. In the Slave Mode, data transmission/reception will be enabled by the clock signal of the external master device. The following steps show the data transfer order in the Master/Slave Mode.

### **Master Mode**

- Step 1  
Initialize the program. Set the SIM0~SIM2 bits in the SIMCTL0 register to select the Master Mode and the required serial transmission rate.
- Step 2  
Set CSEN and use MLS bits to decide whether the data starts from the low or high bit. The Slave must correspond with the master device.
- Step 3  
Set the SIMEN bit in the SIMCTL0 register to enable the SPI interface.
- Step 4  
Write data to the SIMDR and check WCOL. If WCOL=1 then a collision has occurred and jump to Step 4. If WCOL=0 then jump to Step 5.
- Step 5  
Check TRF or wait for SPI serial function interrupt.
- Step 6  
Read data from the SIMDR register and place it in the temp3 data register.
- Step 7  
Clear TRF, return to step 4.

**Slave Mode**

- Step 1  
Set the SIM0~SIM2 bit to 101 to select the Slave Mode.
- Step 2  
Set CSEN and use MLS to decide whether the data starts from the low or high bit. The slave should correspond with the master device.
- Step 3  
Set the SIMEN bit in the SIMCTL0 register to enable the SPI interface.
- Step 4  
Write data to SIMDR, check if WCOL. If WCOL=1 then a collision has occurred and jump to Step 4. If WCOL=0 then jump to Step 5.
- Step 5  
Check TRF or wait for the SPI serial function interrupt.
- Step 6  
Read data from the SIMDR register and place it in the temp3 data register.
- Step 7  
Clear TRF, return to Step 4.

**SPI Configuration Option**

Some configuration options must be set to enable certain register bits for the SPI interface function. One option is to enable the WCOL bit which is the data collision bit. The other is used to disable or enable the CSEN bit in the SIMCTL0 register. If the CSEN bit is disabled by the configuration option, none of the SPI function control will be affected by the CSEN register bit. The WCOL bit in the SIMCTL2 register is used to detect data collisions during data transfer. During data transmission if any attempt is made to write data to the SIMDR register, the WCOL bit will indicate a data collision and disable further write operations. The WCOL bit will be set to 1 by the hardware but should be cleared to zero by the application program. The WCOL bit can be disabled or enabled by a configuration option.

**Programming Considerations**

After the device enters the IDLE Mode, note that data reception/transmission depends upon the FSYSON bit which is located as the seventh bit of the WDTC register. When FSYSON=1, data reception and transmission are allowed in the IDLE Mode and the TRF bit will be used to generate an interrupt after data transmission/reception.

## Program Description

The example here is composed of a main program and an SPI service program. The main program will initialize the MCU, especially the initialization of the relevant SPI registers and the disabling of the pin-shared functions that share the same pins as the SPI function.

The spi\_Transmission subroutine in the SPI service program will send data in the temp1 register to the SPI function and save data from the SPI module to the temp3 register for users to modify. The user only needs to manage the data in the temp1 or temp3 registers to transmit and receive data.

The interrupt is not enabled in the program. If it is required, after checking that WOCL=0, then ignore the TRF bit. Enable the corresponding interrupt enable bit, ESIM, and the master interrupt enable bit, EMI, to wait for an interrupt to be generated.

In the configuration options, enable the SIM Function and the WOCL and CSEN bits before using the SPI function. Other options depend on actual user requirements. The HT56R24(SPI SIM).zip attachment is the circuit of the HT56R24 in the SPI operating mode together with the HT66F40.

## Program Example

Input/Output Program of the Master/Slave Device in the SPI(SIM) mode

```
;function:edit for SPI master
;MCU HT56R24
;option:
;OSC HIRC 4MHz
;SIM enable
;WCOL enable
;CSEN enable
;IO or resb IO
;WDT disable
;SPI1 disable
;SPI1 WCOL disable
;SPI1 CSEN disable
;pa5/6/7 VDD
;VDDIO disable
;pb4/5 IO
;pc6 IO
;others select by user
include HT56R24.inc
ds .section 'data'
temp1 db ?
temp2 db ?
temp3 db ?
temp4 db ?
temp5 db ?
temp6 db ?
temp7 db ?
temp8 db ?
cs .section 'code'
org 000h
jmp main
org 014h
reti
mmov macro h2,h1
mov a,h1
mov h2,a
endm
main:
call BK0
call BK2
call BK3
```

```

call    BK4
call    BK5          ;clr data of the bank0~5,except bank1
set     pec2         ;set pe2 as input
set     pepu2
set     pec3         ;set pe3 as input
set     pepu3
set     pec4         ;set pe4 as input
set     pepu4
clr     emi
clr     tmr1c
clr     intc0
clr     intc1
clr     simdr
clr     temp1
clr     temp2
clr     temp3
clr     temp4
clr     temp5
L0:sz   pe2          ;choose the transmit mode of SPI
      jmp    L1          ;pe2=1,slave
      jmp    L2          ;pe2=0,master,fsys/4
L1:clr  csen
      mov    a,0a0h
      jmp    L3
L2:set  csen
      mov    a,000h
L3:mov  simctl0,a
      clr    ckpol       ;high Base Level active rising edge
      clr    ckeg
      set    mls         ;data shift oder from msb to lsb
      set    simen       ;enable sim
      jmp    spi_Transmission
L:  sz   pe3
      jmp    Linput       ;pe3=1, pa port as input mode
      jmp    Loutput      ;pe3=0, pa port as output mode
Linput:
      ;set the pa port input mode one time in order not to inflict the
      transmission
      mov    a,temp4
      addm a,pcl
      jmp    Linput1
      jmp    Linput2
Linput1:
      set    pac         ;set pa port as input
      set    papu
Linput2:
      mmov temp1,pa      ;mov pa to temp1 register
      mmov temp4,1
      clr    temp5
      jmp    spi_Transmission
Loutput:
      mov    a,temp5
      addm a,pcl
      jmp    Loutput1
      jmp    Loutput2
Loutput1:
      clr    pac         ;set pa port as output
      set    pa
Loutput2:
      mmov pa,temp3      ;mov the simdr data into temp3 and display by pa port
      mmov temp5,1
      clr    temp4

```

```

        jmp      spi_Transmission
spi_Transmission:      ;the transmission of SPI programme
        set     simen
        clr     trf
        clr     wcol
        mmov   simdr,temp1      ;mov the register data of temp1 into simdr,ready to transmit
        sz     wcol
        jmp    $-4
        snz    trf
        jmp    $-1
        mmov   temp3,simdr
        snz    pe4
        jmp    L0      ;pe4=0,return to judge the state of pe2,and judge the transmit
slave or master mode
        jmp    L      ;pe4=1,return to judge the state of pe3,and judge the pd input
or output mode

```

### Notes:

- **pe2, pe3:**
  - 00: master device, output, data sent from the slave device and received by the master device will be seen by the master device.
  - 10: master device, input, data sent from the master device and received by the slave device will be seen by the slave device.
  - 01: slave device, output, data sent from the master device and received by the slave device will be seen by the slave device.
  - 11: slave device, input, data sent from the slave device and received by the master device will be seen by the master device.
- **pe4:**
  - 0: check pe2 high/low condition; reset the master/slave mode for SPI communication.
  - 1: enter pe3 high/low detection, setup all the input/output modes of the SIMDR register.

### Slave/Master device input/output program in the SPI mode:

```

;function: edit for SPI master
;MCU: HT66F40
;option:
;OSC  $\Delta$  HIRC + 4M + LIRC 32K
;SIM enable
;WCOL enable
;CSEN enable
;I/O or RESB $\rightarrow$  RESB
;WDT Enable two instructions
;Others select by user
include HT66F40.INC
.listinclude
use_data .section 'data'
        data1 db      ?
        data2 db      ?
        data3 db      ?
        temp1 db      ?
        temp2 db      ?
maincode .section at 00 'code'
        org    00h
        jmp   main_start
mmov    macro h2,h1
mov     a,h1
mov     h2,a
endm
main_start:
        call  initial      ;initial the program

```

```

    set     pac
; the transmission of SPI programme
spi_server:
    set     csen           ;SPI SCS pin Control--enable
    set     mls           ;SPI Data shift order---MSB
    sz     pa.0
    jmp     L1
    jmp     L2
L1:mov     a,0a0h         ;pa.0=1 spi slave mode
    jmp     L3
L2:mov     a,020h         ;pa.0=0 spi master mode
L3:mov     simc0,a
    set     simen
    jmp     spi_Transmission
L: snz     pa.1
    jmp     pc_as_output  ;pa1=0,pc port as output
    jmp     pc_as_input   ;pa1=1,pc port as input
pc_as_output:
    mov     a,temp1
    addm a,pcl
    jmp     Loutput1
    jmp     Loutput2
Loutput1:
    clr     pcc
    clr     pc
Loutput2:
    mmov pc,data2
    mmov temp1,1
    clr     temp2
    jmp     spi_Transmission
pc_as_input:
    mov     a,temp2
    addm a,pcl
    jmp     Linput1
    jmp     Linput2
Linput1:
    set     pcc
Linput2:
    mmov data1,pc
    mmov temp2,1
    clr     temp1
    jmp     spi_Transmission
spi_Transmission:
    clr     trf
    clr     wcol
    mov     a,data1
    mov     simd,a        ;move data1 to simd register
    sz     wcol          ;check if the mode of write has a conflict
    jmp     $-4
    snz     trf          ;check if the transmission has finished
    jmp     $-1
    mmov data2,simd     ;read data from simd
    snz     pa2
    jmp     spi_server   ;pa2=0 ' return to check the condition of the pa1 port
    jmp     L            ;pa2=1 ' return to check the condition of the pa0 port

```

**Note:**

- **pa0, pa1:**  
00: master device, output, data sent from the slave device and received by the master device will be seen by the master device.

10: master device, input, data sent from the master device and received by the slave device will be seen by the slave device.

01: slave device, output, data sent from the master device and received by the slave device will be seen by the slave device.

11: slave device, input, data sent from the slave device and received by the master device will be seen by the master device.

- pa2:

0: check the pa0 high/low condition; reset the master/slave mode for SPI communication.

1: enter the pa1 high/low detection, setup all the input/output mode of the SIMDR register.

## Conclusion

This application has provided an operational description and special considerations regarding the SPI in the HT56R24. The operation and special notes for the SPI1 are similar with that of the SPI. Users, when using the SPI1, may setup the corresponding registers as shown below:

SPI(SIM)	SIMCTL0	SIMCTL2	SIMDR
SPI1(Voice)	SPICTL0	SPICTL1	SPIDR

The SPICTL0 register can only setup the master device, frequency or slave device without the I<sup>2</sup>C option. The SPI1, WCOL1 and CSEN1 functions in the configuration options should all be enabled. If an SPI1 interrupt is required, ignore the TRF1 bit after checking that WCOL1=0. Enable the corresponding interrupt enable bit, ESPI, and the master interrupt enable bit, EMI, to wait for an interrupt to be generated. Note that the SPI (SIM) interrupt vector is located at 014H and the SPI1 (Voice) at 018H. Other options can be setup according to the related register configuration options listed above. Users may insert this IP directly into their own programs and modify the program according to their actual requirement with the description provided above. The SPI (SIM) and SPI1(Voice) communication and circuit of the HT56R24 and HT66F40 are enclosed.