

Both the X5043 and X5045 feature a power on reset circuit, low voltage reset controller, programmable watchdog timer, and 4K bits of high speed, three-wire serial, nonvolatile EEPROM in a single 8-pin package.

Power-On-Reset (POR)

The X5043, X5045 Power-On-Reset circuit holds the RESET pin active for 250ms when the system power is applied. This prevents the microcontroller from operating while the power supply is stabilizing. This improves the reliability of system start up.

Low Voltage Reset (LVR)

During operation, the low voltage reset circuit monitors the supply voltage. If the voltage drops below a specified minimum, the X5043, X5045 drives the RESET pin active. This stops the operation of the microcontroller to prevent unexpected operation. If the microcontroller operates at voltages that are too low, the microcontroller or a peripheral device may fail, causing the system to "lock-up" or resulting in data corruption.

Watchdog Timer

While the POR and LVR circuits help prevent system problems, the Watchdog Timer helps the system recover when there is a problem. The Watchdog Timer works by resetting the system when there is a time-out. The microcontroller continually resets the timer, as part of the software loop, before the timer times-out. If there is ever a software problem, such as an infinite loop or an operation that waits for a peripheral device, the Watchdog timer expires and resets the microcontroller.

Hardware Implementation

The circuit shown in Figure. 1 includes both a manual and X5043 controlled reset. R1 serves as a pull-up resistor for the X5043 open-drain (i.e. active LOW) reset output. The 2N7000 N-MOSFET is used to invert the active LOW reset, to directly control the 8031 RST pin. The circuit shown in Figure. 2 has both a manual and X5045 controlled reset. The circuit in Figure 2 is preferable because the X24045 has the correct reset polarity for the 8051.

Software Implementation

The following routines are included for implementing an interface to the X5043, X5045:

wren_cmd – This routine sets the write enable latch, which must be set before writing to either the EEPROM memory array or the status register. The WEL bit is automatically reset after a write operation.

wrdi_cmd – This command resets the write enable latch.

wrsr_cmd – This operation writes the watchdog timeout period bits (WD0, WD1) and the Block Protect bits (BP0, BP1) in the status register.

rdsr_cmd – This routine reads the status register.

byte_write – This command writes a single byte to the EEPROM memory array.

byte_read – This command reads a single byte from the EEPROM memory array.

page_write – This operation writes 3 consecutive bytes to the EEPROM memory array. It can easily be modified to write an entire page (maximum of 16 bytes).

sequ_read – This routine reads three consecutive bytes from the EEPROM memory array. It can be easily modified to read any number of bytes.

rst_wdog – This routine is used to reset the watchdog timer without sending a command.

Application Note 21

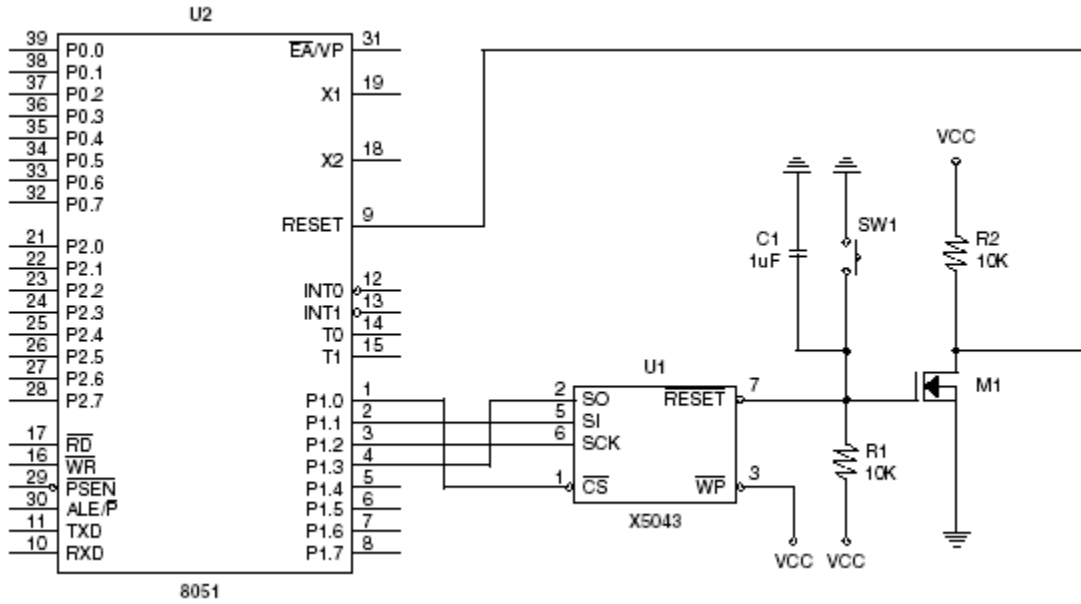


FIGURE 1. CONNECTING AN X5043 TO AN 8051 MICROCONTROLLER, WITH MANUAL RESET CONTROL

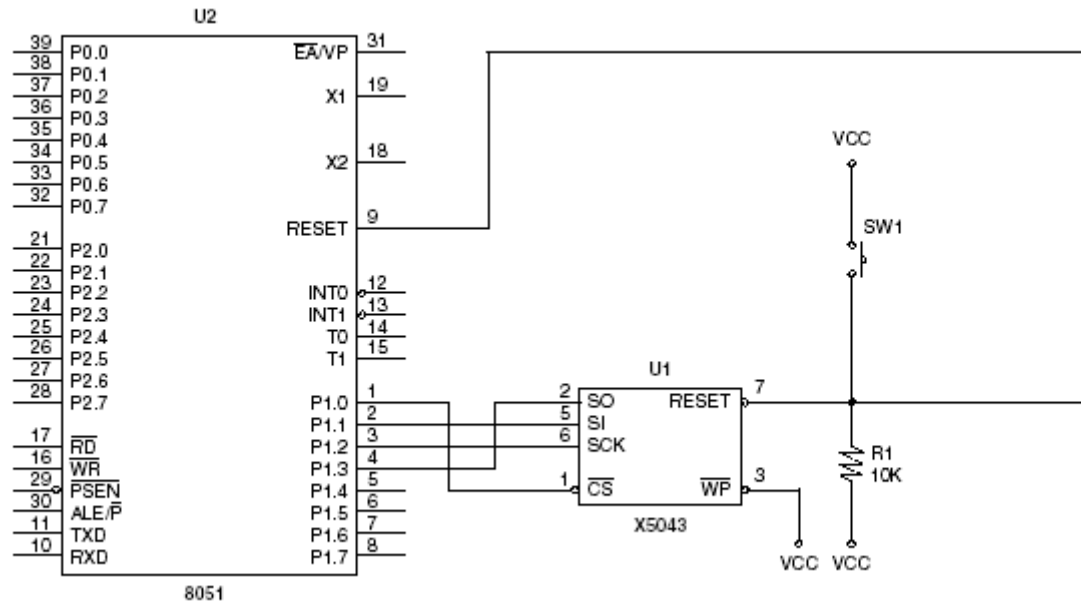


FIGURE 2. CONNECTING AN X5045 TO AN 8051 MICROCONTROLLER, WITH MANUAL RESET CONTROL

Application Note 21

```
*****
;*Copyright (c) 1994 Xicor, Inc.
;*AUTHOR: Richard Downing
*****
;* The purpose of this code is to provide routines to interface the Xicor X5043 with the 8031
;* microcontroller. The interface uses the 8031's general purpose parallel port 1 and connects
;* P1.0 to the chip select line (/CS), P1.1 to the serial input data line (SI), P1.2 to the
;* serial clock line (SCK) and P1.3 to the serial output data line (SO).
;*
;* All X5043 commands are provided. These are :-
;*
;* 1. Set Write Enable Latch
;* 2. Reset Write Enable Latch
;* 3. Write Status Register
;* 4. Read Status Register
;* 5. Single Byte Write
;* 6. Single Byte Read
;* 7. Page Write
;* 8. Sequential Read
;* 9. Reset Watchdog Timer
;*
;* The code writes 00H to the Status Register; reads the Status Register; writes 11H to
;* address 55H in Byte Mode; performs a single Byte Read from address 55H; writes 22H,
;* 33H, 44H to addresses 1F0H, 1F1H, 1F2H in Page Mode; performs a Sequential Read
;* from addresses 1F0H, 1F1H, 1F2H; and resets watchdog timer. This code can also be used with
;* the X5045 which is identical to the X5043, except for its RESET output polarity.
*****

;* CONSTANTS
cs          bit    P1.0   ; Port 1 bit 0 used for chip select (/CS)
si          bit    P1.1   ; Port 1 bit 1 used for serial input (CI)
sck        bit    P1.2   ; Port 1 bit 2 used for serial clock (SCK)
so         bit    P1.3   ; Port 1 bit 3 used for serial output (SO)
WREN_INST  equ    06H    ; Write enable latch instruction (WREN)
WRDI_INST  equ    04H    ; Write disable latch instruction (WRDI)
WRSR_INST  equ    01H    ; Write status register instruction (WRSR)
RDSR_INST  equ    05H    ; Read status register instruction (RDSR)
WRITE_INST equ    02H    ; Write memory instruction (WRITE)
READ_INST  equ    03H    ; Read memory instruction (READ)
BYTE_ADDR  equ    55H    ; Memory address for byte mode operations
BYTE_DATA  equ    11H    ; Data byte for byte write operation
PAGE_ADDR  equ    1F0H   ; Memory address for page mode operations
PAGE_DATA1 equ    22H    ; 1st data byte for page write operation
PAGE_DATA2 equ    33H    ; 2nd data byte for page write operation
PAGE_DATA3 equ    44H    ; 3rd data byte for page write operation
STATUS_REG equ    00H    ; Status register
MAX_POLL   equ    99H    ; Maximum number of polls
INIT_STATE equ    09H    ; Initialization value for control ports
USER       equ    030H   ; Address location of User Code
*****

;* INTERNAL RAM

STACK_TOP  equ    060H   ; Stack top

*****

;* CODE

          ORG    0000H   ; Reset vectors to this location
          ljmp   main

          ORG    0100H
```

Application Note 21

```
main:
    mov     SP,#STACK_TOP      ; Initialize stack pointer
    clr     EA                  ; Disable interrupts
    mov     P1, #INIT_STATE    ; Init control lines (/CS & SO =1, SCK & SI =0)
    lcall   wren_cmd           ; Set write enable latch
    lcall   wrsr_cmd          ; Write 00H to status register
    lcall   wren_cmd           ; Set write enable latch
    lcall   byte_write        ; Write 11H to address 55H (Byte Write)
    lcall   byte_read         ; Read from address location 55H (Byte Read)
    lcall   wren_cmd           ; Set write enable latch
    lcall   page_write        ; Page write 22H/33H/44H to addresses 1F0/1/2H
    lcall   sequ_read         ; Seq. Read from address locations 1F0/1/2H
    lcall   rst_wdog          ; Reset Watchdog timer
    jmp     USER

;*****
;* Name: WREN_CMD
;* Description: Set write enable latch
;* Function: This routine sends the command to enable writes to the EEPROM memory array or
;* status register
;* Calls: outbyt
;* Input: None
;* Outputs: None
;* Register Usage: A
;*****
wren_cmd:
    clr     sck                ; Bring SCK low
    clr     cs                  ; Bring /CS low
    mov     A, #WREN_INST
    lcall   outbyt             ; Send WREN instruction
    clr     sck                ; Bring SCK low
    setb    cs                  ; Bring /CS high
    ret

;*****
;* Name: WRDI_CMD
;* Description: Reset write enable latch
;* Function: This routine sends the command to disable writes to the EEPROM memory array or
;* status register
;* Calls: outbyt
;* Input: None
;* Outputs: None
;* Register Usage: A
;*****
wrdi_cmd:
    clr     sck                ; Bring SCK low
    clr     cs                  ; Bring /CS low
    mov     A, #WRDI_INST
    lcall   outbyt             ; Send WRDI instruction
    clr     sck                ; Bring SCK low
    setb    cs                  ; Bring /CS high
    ret
```

Application Note 21

```
*****
;* Name: WRSR_CMD
;* Description: Write Status Register
;* Function: This routine sends the command to write the WD0, WD1, BP0 and BP0 EEPROM
;* bits in the status register
;* Calls: outbyt, wip_poll
;* Input: None
;* Outputs: None
;* Register Usage: A
*****
wrsr_cmd:
    clr    sck            ; Bring SCK low
    clr    cs             ; Bring /CS low
    mov    A, #WRSR_INST
    lcall  outbyt         ; Send WRSR instruction
    mov    A, #STATUS_REG
    lcall  outbyt         ; Send status register
    clr    sck            ; Bring SCK low
    setb   cs             ; Bring /CS high
    lcall  wip_poll       ; Poll for completion of write cycle
    ret

*****
;* Name: RDSR_CMD
;* Description: Read Status Register
;* Function: This routine sends the command to read the status register
;* Calls: outbyt, inbyt
;* Input: None
;* Outputs: A = status register
;* Register Usage: A
*****
rdsr_cmd:
    clr    sck            ; Bring SCK low
    clr    cs             ; Bring /CS low
    mov    A, #RDSR_INST
    lcall  outbyt         ; Send RDSR instruction
    lcall  inbyt          ; Read status register
    clr    sck            ; Bring SCK low
    setb   cs             ; Bring /CS high
    ret

*****
;* Name: BYTE_WRITE
;* Description: Single Byte Write
;* Function: This routine sends the command to write a single byte to the EEPROM memory array
;* Calls: outbyt, wip_poll
;* Input: None
;* Outputs: None
;* Register Usage: A, B
*****
byte_write:
    mov    DPTR, #BYTE_ADDR ; Set address of byte to be written
    clr    sck            ; Bring SCK low
    clr    cs             ; Bring /CS low
    mov    A, #WRITE_INST
    mov    B, DPH
    mov    C, B.0
    mov    ACC.3, C
    lcall  outbyt         ; Send WRITE instruction including MSB of address
    mov    A, DPL
    lcall  outbyt         ; Send 8 LSBs of address
```

Application Note 21

```
mov    A, #BYTE_DATA
lcall  outbyt          ; Send data byte
clr    sck            ; Bring SCK low
setb   cs             ; Bring /CS high
lcall  wip_poll       ; Poll for completion of write cycle
ret

;*****
;* Name: BYTE_READ
;* Description: Single Byte Read
;* Function: This routine sends the command to read a single byte from the EEPROM memory array
;* Calls: outbyt, inbyt
;* Input: None
;* Outputs: A = read byte
;* Register Usage: A, B
;*****
byte_read:
mov    DPTR, #BYTE_ADDR    ; Set address of byte to be read
clr    sck                ; Bring SCK low
clr    cs                 ; Bring /CS low
mov    A, #READ_INST
mov    B, DPH
mov    C, B.0
mov    ACC.3, C
lcall  outbyt            ; Send READ instruction including MSB of address
mov    A, DPL
lcall  outbyt            ; Send 8 LSBs of address
lcall  inbyt             ; Read data byte
clr    sck                ; Bring SCK low
setb   cs                 ; Bring /CS high
ret

;*****
;* Name: PAGE_WRITE
;* Description: Page Write
;* Function: This routine sends the command to write three consecutive bytes to the EEPROM
;* memory array using page mode
;* Calls: outbyt, wip_poll
;* Input: None
;* Outputs: None
;* Register Usage: A, B
;*****
page_write:
mov    DPTR, #PAGE_ADDR    ; Set address of 1st byte to be written
clr    sck                ; Bring SCK low
clr    cs                 ; Bring /CS low
mov    A, #WRITE_INST
mov    B, DPH
mov    C, B.0
mov    ACC.3, C
lcall  outbyt            ; Send WRITE instruction including MSB of address
mov    A, DPL
lcall  outbyt            ; Send 8 LSBs of address
mov    A, #PAGE_DATA1
lcall  outbyt            ; Send 1st data byte
mov    A, #PAGE_DATA2
lcall  outbyt            ; Send 2nd data byte
mov    A, #PAGE_DATA3
lcall  outbyt            ; Send 3rd data byte
clr    sck                ; Bring SCK low
setb   cs                 ; Bring /CS high
```

Application Note 21

```
    lcall wip_poll          ; Poll for completion of write cycle
    ret

;*****
;* Name: SEQU_READ
;* Description: Sequential Read
;* Function: This routine sends the command to read three consecutive bytes from the EEPROM
;* memory array using sequential mode
;* Calls: outbyt, inbyt
;* Input: None
;* Outputs: A = last byte read
;* Register Usage: A, B
;*****
sequ_read:
    mov    DPTR, #PAGE_ADDR    ; Set address of 1st byte to be read
    clr    sck                 ; Bring SCK low
    clr    cs                   ; Bring /CS low
    mov    A, #READ_INST
    mov    B, DPH
    mov    C, B.0
    mov    ACC.3, C
    lcall  outbyt              ; Send READ instruction with MSB of address
    mov    A, DPL
    lcall  outbyt              ; Send low order address byte
    lcall  inbyt               ; Read 1st data byte
    lcall  inbyt               ; Read 2nd data byte
    lcall  inbyt               ; Read 3rd data byte
    clr    sck                 ; Bring SCK low
    setb   cs                  ; Bring /CS high
    ret

;*****
;* Name: RST_WDOG
;* Description: Reset Watchdog Timer
;* Function: This routine resets the watchdog timer without sending a command
;* Calls: None
;* Input: None
;* Outputs: None
;* Register Usage: None
;*****
rst_wdog:
    clr    cs                   ; Bring /CS low to reset watchdog timer
    setb   cs                  ; Bring /CS high
    ret

;*****
;* Name: WIP_POLL
;* Description: Write-In-Progress Polling
;* Function: This routine polls for completion of a nonvolatile write cycle by examining the
;* WIP bit of the status register
;* Calls: rdsr_cmd
;* Input: None
;* Outputs: None
;* Register Usage: R1, A
;*****
wip_poll:
    mov    R1, #MAX_POLL       ; Set maximum number of polls
wip_poll1:
    jnb    ACC.0, wip_poll2    ; If WIP bit '0' write cycle completed
    djnz   R1, wip_poll1      ; If WIP bit '1' continue polling
wip_poll2:
```

Application Note 21

```
ret

;*****
;* Name: OUTBYT
;* Description: Sends byte to EEPROM
;* Function: This routine shifts out a byte, starting with the MSB, to the EEPROM
;* Calls: None
;* Input: A = byte to be sent
;* Outputs: None
;* Register Usage: R0, A
;*****
outbyt:
    mov    R0, #08        ; Set bit counter to eight
outbyt1:
    clr    sck            ; Bring SCK low
    rlc    A              ; Shift byte left through carry
    mov    si, C          ; Send data bit in carry
    setb   sck            ; Bring SCK high
    djnz   R0, outbyt1    ; Finish if last data bit
    clr    si             ; Place SI in known condition
    ret

;*****
;* Name: INBYT
;* Description: Recieves byte from EEPROM
;* Function: This routine recieves a byte, MSB first, from the EEPROM
;* Calls: None
;* Input: None
;* Outputs: A = recieved byte
;* Register Usage: R0, A
;*****
inbyt:
    mov    R0, #08        ; Set bit counter to eight
inbyt1:
    setb   sck            ; Bring SCK high
    clr    sck            ; Bring SCK low
    mov    C, so          ; Receive data bit and store in carry
    rlc    A              ; Shift byte left through carry
    djnz   R0, inbyt1    ; Finish if last data bit
    ret

END
```

Intersil Corporation reserves the right to make changes in circuit design, software and/or specifications at any time without notice. Accordingly, the reader is cautioned to verify that the Application Note or Technical Brief is current before proceeding.

For information regarding Intersil Corporation and its products, see www.intersil.com