

Interfacing the X84641/129 MPS E²PROM to 8051 Microcontrollers

by Applications Staff, May 1997

The MPS E²PROM is an attractive alternative to the 'add-on' parallel or serial nonvolatile memories that have traditionally been used in microcontroller-based systems. It features serial E²PROM memory that is accessed via a standard parallel bus. The serial architecture of the X84641/129 provides a cost and space saving replacement for low density parallel devices. Its unique parallel bus compatible interface also offers an I/O-saving alternative to serial devices that typically

require two or three control pins for communication purposes. Although the X84641/129 can be used to interface with practically any 8, 16, or 32-bit microcontroller, the code provided here is for the widely used 8051 family.

Additional code can be found on the Xicor web site that will implement interfaces between the 8051 microcontroller family and other Xicor serial devices. Xicor application notes are available from the WWW using the URL: <http://www.xicor.com>.

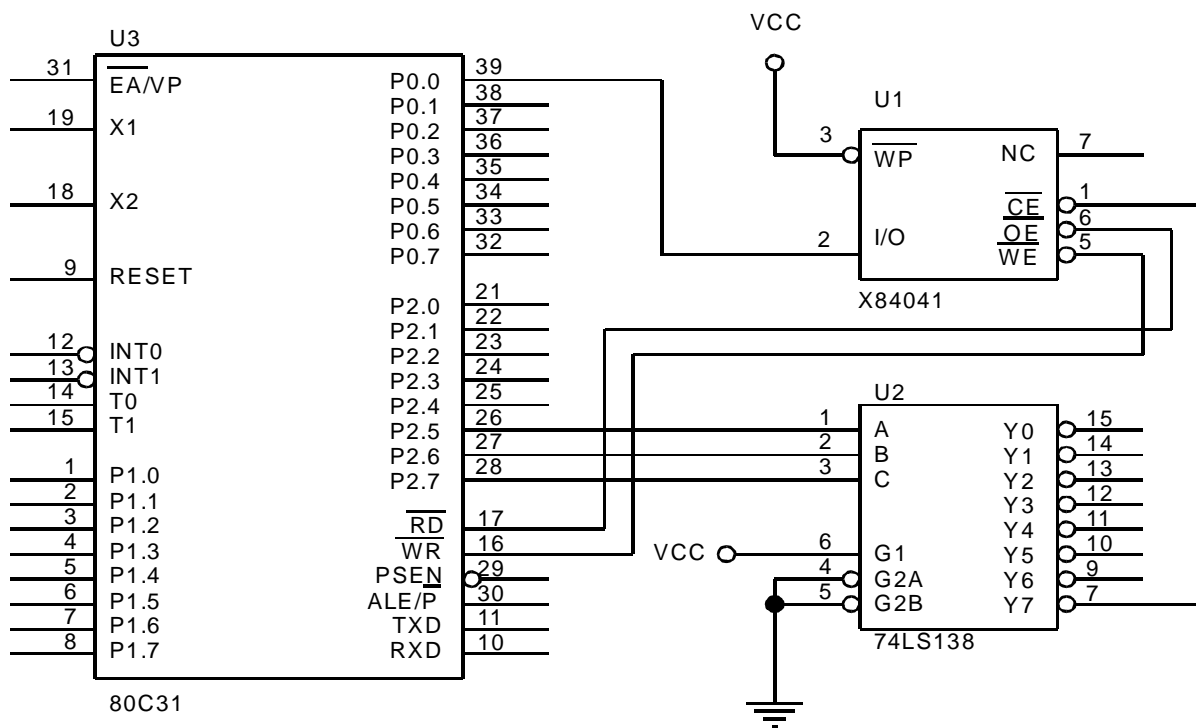


Figure 1. One way to interface an X84641/129 to 8051 microcontrollers

```

$ TITLE(X84xxx/8031/1.0)
;*****
;* The purpose of this code is to provide routines to interface the Xicor MPS EEPROM
;* with the 8031 microcontroller. The interface uses the 8031's parallel bus and connects the
;* microcontroller's /RD strobe to the MPS EEPROM's /OE pin, /WR strobe to the /WE pin, and
;* decodes address pins A13, A14, A15 for /CS. Address space E000 - FFFF is used to select the
;* MPS EEPROM. All MPS EEPROM commands are provided. These are :-
;* 1. Single Byte Write
;* 2. Single Byte Read
;* 3. Page Write
;* 4. Sequential Read
;* The code performs a byte write of 11H to EEPROM address 0; a byte read from EEPROM address
;* 0; a page write of 22H, 33H, 44H to EEPROM addresses 100H, 101H, 102H; and a sequential
;* read from EEPROM addresses 100H, 101H, 102H.
;*****
;* CONSTANTS
BYTE_ADDR      equ    0F000H      ; Address used for single byte operation
BYTE_DATA      equ    11H        ; Data used for single byte write operation
PAGE_ADDR      equ    0F100H      ; Address used for page operation
PAGE_DATA1     equ    22H        ; 1st data used for page write operation
PAGE_DATA2     equ    33H        ; 2nd data used for page write operation
PAGE_DATA3     equ    44H        ; 3rd data used for page write operation
SLIC           equ    030H        ; Address location of SLIC

;*****
;* INTERNAL RAM
;*****

STACK_TOP      equ    060H        ; Stack top

;*****
;* CODE
;*****

                ORG    0000H      ; Reset vectors to this location
                jmp    main
                ORG    0100H

MAIN:
                mov    SP, #STACK_TOP    ; Initialize stack pointer
                clr    EA                ; Disable interrupts
                lcall  byte_write        ; Write 11H to address 0 (single byte write)
                lcall  byte_read         ; Read from address 0 (single byte read)
                lcall  page_write        ; Write 22/33/44H to addresses 100/1/2H (page write)
                lcall  sequ_read         ; Read from addresses 100H,101H,102H (sequential read)
                ljmp   SLIC

```

```

;*****
;* Name: BYTE_WRITE
;* Description: Single byte write
;* Function: This routine sends the command sequence to write a single byte to EEPROM
;* Calls: rst, load_byte, nv_start, nv_poll
;* Input: None
;* Outputs: None
;* Register Usage: A
;*****

```

```
byte_write:
```

```

    mov    DPTR, #BYTE_ADDR
    lcall  rst           ; Issue a reset sequence
    mov    A, DPH
    lcall  load_byte    ; Load high order address byte
    mov    A, DPL
    lcall  load_byte    ; Load low order address byte
    mov    A, #BYTE_DATA
    lcall  load_byte    ; Load data byte
    lcall  nv_start     ; Start nonvolatile write cycle
    lcall  nv_poll      ; Poll for completion of write cycle
    ret

```

```

;*****
;* Name: PAGE_WRITE
;* Description: Page write
;* Function: This routine sends the command sequence to write three consecutive bytes
;* Calls: rst, load_byte, nv_start, nv_poll
;* Input: None
;* Outputs: None
;* Register Usage: A
;*****

```

```
page_write:
```

```

    mov    DPTR, #PAGE_ADDR
    lcall  rst           ; Issue a reset sequence
    mov    A, DPH
    lcall  load_byte    ; Load high order address byte
    mov    A, DPL
    lcall  load_byte    ; Load low order address byte
    mov    A, #PAGE_DATA1
    lcall  load_byte    ; Load 1st data byte
    mov    A, #PAGE_DATA2
    lcall  load_byte    ; Load 2nd data byte
    mov    A, #PAGE_DATA3
    lcall  load_byte    ; Load 3rd data byte
    lcall  nv_start     ; Start nonvolatile write cycle
    lcall  nv_poll      ; Poll for completion of write cycle
    ret

```

```

;*****
;* Name: BYTE_READ
;* Description: Single byte read
;* Function: This routine sends the command sequence to read a single byte from the EEPROM
;* Calls: rst, load_byte, rec_byte
;* Input: None
;* Outputs: A = byte read
;* Register Usage: A
;*****
byte_read:
        mov     DPTR, #BYTE_ADDR
        lcall  rst             ; Issue a reset sequence
        mov     A, DPH
        lcall  load_byte      ; Load high order address byte
        mov     A, DPL
        lcall  load_byte      ; Load low order address byte
        lcall  rec_byte       ; Receive data byte
        ret

;*****
;* Name: SEQU_READ
;* Description: Sequential read
;* Function: This routine sends the command sequence to read three consecutive bytes
;* Calls: rst, load_byte, rec_byte
;* Input: None
;* Outputs: A = last byte read
;* Register Usage: A
;*****
sequ_read:
        mov     DPTR, #PAGE_ADDR
        lcall  rst             ; Issue a reset sequence
        mov     A, DPH
        lcall  load_byte      ; Load high order address byte
        mov     A, DPL
        lcall  load_byte      ; Load low order address byte
        lcall  rec_byte       ; Receive 1st data byte
        lcall  rec_byte       ; Receive 2nd data byte
        lcall  rec_byte       ; Rceive 3rd data byte
        ret

;*****
;* Name: LOAD_BYTE
;* Description: Loads byte to EEPROM
;* Function: This routine loads a byte, MSB first, to the EEPROM
;* Calls: None
;* Input: None
;* Outputs: None
;* Register Usage: R0, A
;*****
load_byte:
        mov     R0, #08             ; Set bit counter to eight

loop1:
        rl      A                 ; Rotate accumulator left

```

```

        movx  @DPTR, A           ; Load LSB of accumulator
        djnz  R0, loop1        ; Finish if last data bit
        ret

;*****
;* Name: REC_BYTE
;* Description: Receives byte from EEPROM
;* Function: This routine receives a byte, MSB first, from the EEPROM
;* Calls: None
;* Input: None
;* Outputs: A = received byte
;* Register Usage: R0, R1, A
;*****
rec_byte:
        mov   R0, #08          ; Set bit counter to eight
        mov   R1, #00          ; Set register 1 to zero
loop2:
        clr   C                ; Set carry bit to '0'
        movx  A, @DPTR         ; get data bit and store in LSB of acc
        jnb  ACC.0, zero_bit   ; If LSB of accumulator '1' set carry bit to
'1'
        setb  C
zero_bit:
        mov   A, R1            ; Load contents of register 1 to accumulator
        rlc   A                ; Rotate accumulator left through carry
        mov   R1, A            ; Load contents of accumulator to register 1
        djnz  R0, loop2        ; Finish if last data bit
        ret

;*****
;* Name: RST
;* Description: Reset sequence
;* Function: This routine sends the reset command sequence to the EEPROM
;* Calls: None
;* Input: None
;* Outputs: None
;* Register Usage: A
;*****
rst:
        movx  A, @DPTR         ; Read data bit
        mov   A, #00H
        movx  @DPTR, A         ; Write data bit '0'
        movx  A, @DPTR         ; Read data bit
        ret

;*****
;* Name: NV_START
;* Description: Start nonvolatile write
;* Function: This routine sends the command sequence to start the EEPROM nonvolatile write
;* Calls: None
;* Input: None
;* Outputs: None
;* Register Usage: A

```

```
*****
nv_start:
    movx   A, @DPTR           ; Read data bit
    mov    A, #01H
    movx   @DPTR, A          ; Write data bit '1'
    movx   A, @DPTR           ; Read data bit
    ret

*****
;* Name: NV_POLL
;* Description: Nonvolatile write polling
;* Function: This routine polls for the completion of a nonvolatile EEPROM write cycle
;* Calls: None
;* Input: None
;* Outputs: None
;* Register Usage: R0, A
*****
nv_poll:
    mov    R0, #0FFH         ; Set counter to FFH

loop3:
    movx   A, @DPTR         ; Read data bit
    nop                    ; Delay
    nop                    ; Delay
    nop                    ; Delay
    jb     ACC.0, end_loop   ; If data bit '1' nonvolatile cycle complete
    djnz   R0, loop3        ; If data bit '0' nonvolatile cycle in progress

end_loop:
    ret
END
```