Interfacing the NM24C16 Serial EEPROM to the 8031 Microcontroller

National Semiconductor
Application Note 957
Anne Gregory
August 1994

INTRODUCTION

This applications note describes an interface between the National Semiconductor NM24C16 serial EEPROM and an 8031 microcontroller. The interface between the devices uses 2 of the 8031 general purpose I/O port lines. Software has been developed that demonstrates how the NM24C16 can be accessed through the I/O port bits. The circuit and software has been bench tested and is ready to be used in an end user application.

NM24C16 DESCRIPTION

The NM24C16 is a 16k serial EEPROM that has a 2k by 8-bit architecture. The NM24C16 uses the industry standard I^2C serial protocol for data transfers.

The I^2C protocol allows several devices to share the same two wire clock and data bus. Devices that are compatible with the protocol fall into the categories of being either a master or a slave. A master device controls the transfer of data, and a slave device responds to the commands issued by a master. The NM24CXX family of devices always fall into the category of slave devices since they can not initiate data transfers.

The I^2C protocol uses a clock (SCL) and a bidirectional data line (SDA). When the NM24C16 is transmitting data an open drain transistor is used to control the state of the SDA line. The SDA I/O pulls the line low for a zero state, or places the line in high impedance for a one state. An external pull-up resistor ensures a ''high'' condition exists when the SDA line is in a high impedance state.

Data is transfered back and forth by using predefined bit sequences. All transfers are initiated with a START condition (SDA going low with SCL high) and terminated with a STOP condition (SDA going high with SCL high). If an unexpected STOP is ever detected the NM24C16 will return to the standby mode. Because transitions of SDA when SCL is high have been defined as STOP and START conditions, the SDA line must change only when SCL is low while transfers are being performed.

DATA TRANSFERS

There are just two types of data transfers used on the NM24C16, a page write operation, and a sequential read operation. Byte write and byte read operations are simply truncated versions of a page write or sequential read.

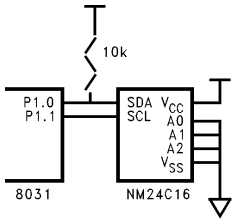
The page write allows up to 16 bytes in a single page to be altered during a single write operation. It is important to note that all addresses to be altered must reside in the same 16 byte page. A byte write is the same as a page write with the data in a single address being altered.

The sequential read operation will allow read operations starting at a user defined address and then allow successive addresses to be read as long as the user continues to indicate that the read operation is to continue. The byte read is simply a sequential read from only a single address.

8031 INTERFACE DESCRIPTION

The interface to the 8031 uses 2 general purpose port lines. One of the lines is used to drive the SCL input of the NM24C16, and the other is used as an I/O port connected to the SDA line. The 8031 has very weak pull-ups on the output ports that provide a high state. When an 8031 port bit is sending a high, the bit can be driven externally and used as an input.

Port 1 of the 8031 provides the 2 I/O bits for the interface. *Figure 1* shows how the NM24C16 is connected to the 8031. The port bits that were chosen for this interface are not especially significant. Any 2 available port bits could be used as long as 1 can be configured an output and 1 as an I/O with the weak pull-up. Changes in the interface software to implement different port placements would only require a change in the SDA and SCL port definition at the top of the program.



TL/D/12084–1

FIGURE 1. NM24C16 to 8031 Connections

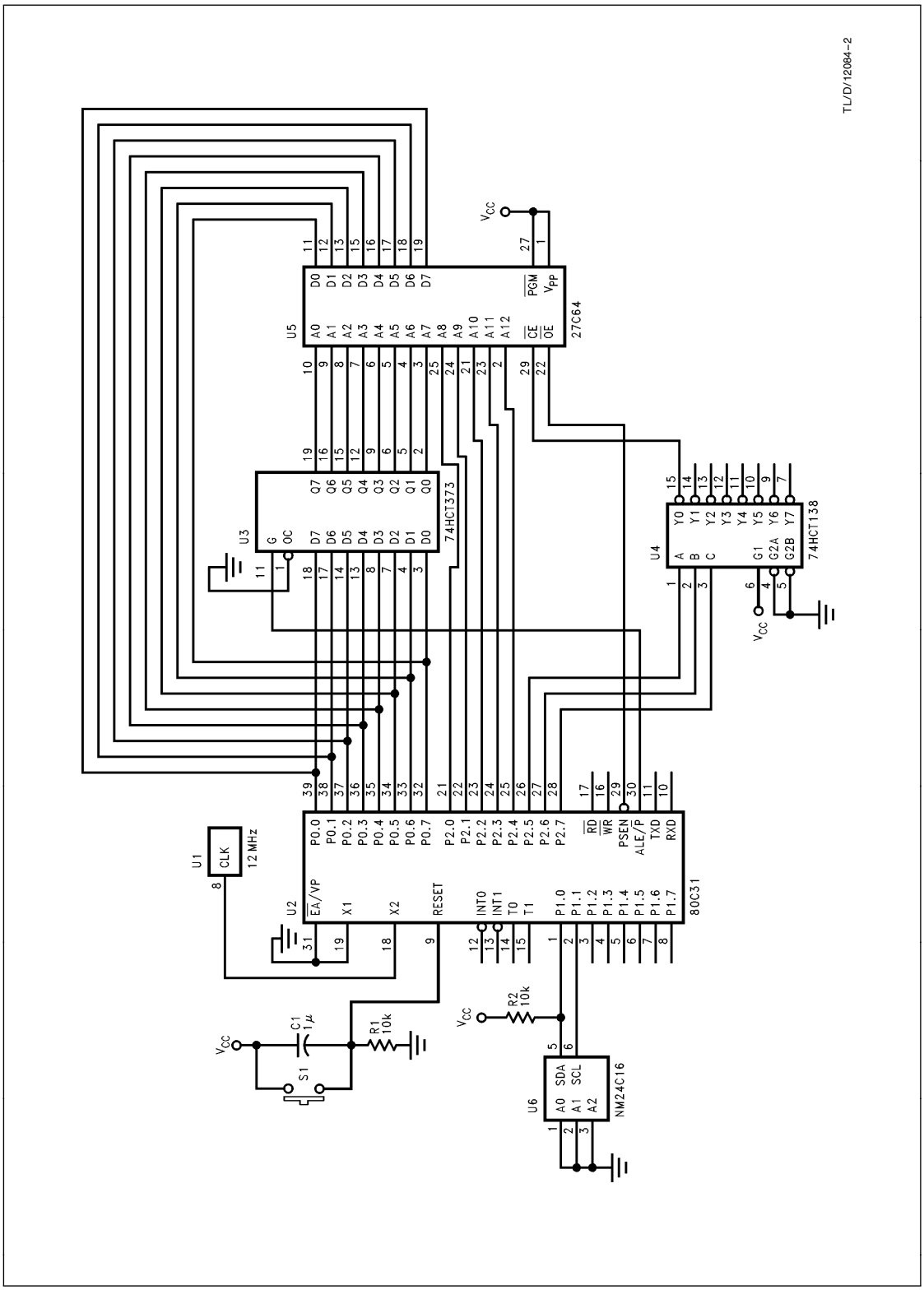
SOFTWARE DESCRIPTION

The software listing demonstrates a byte read and byte write operation. The read and write operations are implemented in separate subroutines. Parameters to be passed into the subroutines are stored in the SRAM portion of the 8031. The passed parameters include address (hi-order and low-order) and data (single byte) information. The variables are sometimes modified during subroutine operation so they must be initialized immediately prior to a subroutine call. Expansions of the byte read and write routines to implement sequential read and page write should be straight forward.

The software also implements acknowledge (ACK) polling to indicate when a write operation has completed. While the NM24C16 is actually changing the state of the EEPROM bits all input pins are ignored. Once a write cycle has concluded the NM24C16 will return an acknowledge when a valid slave address is issued. The ACK polling routine repeatedly sends a slave address and check to see if the X24C16 returns an acknowledge. A STOP condition is issued once an ACK is received to return the NM24C16 to the standby mode. Using acknowledge polling can significantly reduce the effective Write Cycle Time because the actual time required is typically much less than the maximum specified in the data sheet.

CONCLUSION

This applications note has shown how the NM24C16 can easily be interfaced to the 8031 microcontroller. Interface resources are minimal with only 2 I/O port pins and 200 bytes of code required. Although this applications note describes an interface to the 8031, the issues discussed can be used to implement an 8031 interface to any general purpose microcontroller.


TL/D/12084–2
U1
CLK
12 MHz
U2
80C31
U3
74HCT373
U4
74HCT138
U5
27C64
U6
NM24C16
C1
1μ
R1
10k
R2
10k
S1
VCC

```
;*********************************************************************;
;* This code was developed to demonstrate how the NM24C16 serial EEPROM  *
;* can be interfaced to the 8031 microcontroller.  The software includes *
;* byte read and a byte write routines.                    *
;*                                  *
;* The 8031 interfaces to the NM24C16 by using 2 general purpose I/O    *
;* port lines.  Port 1 is used with one line driving the NM24C16 SCL    *
;* input and a second port line used in a bidirectional mode for SDA.   *
;*                                  *
;* The mainline was used to test the functionality of the subroutines.  *
;* The subroutines can be copied directly into a customer's program and  *
;* be expected to operated as described.  The final mainline only       *
;* performs a byte write, acknowledge polling and finaly a byte read.   *
;*********************************************************************
;
;************************
;* BIT POSITION EQUATES *
;************************

SDA    BIT    P1.0                    ;SDA position in port 1
SCL    BIT    P1.1                    ;SCL position in port 1

;***********************
;* VARIABLE REGISTERS *
;***********************

ADDLO  EQU    R2                      ;low order address pointer
ADDHI  EQU    R3                      ;high order address pointer
COUNT  EQU    R4                      ;loop counter
TDATA  EQU    R5                      ;scratch register
RWDATA EQU    R6                      ;read and write data register

;****************
;* RESET VECTOR *
;****************

     ORG    0000H
     LJMP   BEGIN                     ;reset vector to address 0100H

;******************************
;* PROGRAM STARTING LOCATION *
;******************************

     ORG    0100H
BEGIN: MOV    SP,#60H                 ;initialize stack pointer
```

TL/D/12084–3

```
;************
;* MAINLINE *
;************

      MOV    A,#001H
      MOV    ADDHI,A
      MOV    A,#023H
      MOV    ADDLO,A
      MOV    A,#096H
      MOV    RWDATA,A
      LCALL  WRITE                    ;write data 96H to address 0123H
      LCALL  POLL                     ;perform acknowledge polling
      LCALL  READ                     ;read data from address 0123H
DONE: LJMP   DONE                     ;wait until reset loop

;*****************************
;* NM24C16 FUNCTIONAL ROUTINES *
;*****************************

;*************************************************************;*
* WRITE performs a byte write operation into the NM24C16. The routine *
;* expects the address to modify to be specified in the ADDLO and ADDHI *
;* variables. The new data value is specified in the RWDATA variable. *
;*    ************************************************************
WRITE: LCALL  START                   ;issue START condition
      MOV    A,ADDHI                  ;build slave address
      RL     A
      ORL    A,#0A0H
      LCALL  SENDB                    ;send slave address
      LCALL  ACK                      ;get ACK from NM24C16
      MOV    A,ADDLO
      LCALL  SENDB                    ;send low order address
      LCALL  ACK                      ;get ACK from NM24C16
      MOV    A,RWDATA
      LCALL  SENDB                    ;send data value to write
      LCALL  ACK                      ;get ACK from NM24C16
      LCALL  STOP                     ;issue STOP condition
      RET

;*************************************************************;*
* POLL performs acknowledge to determine when a write cycle has      *
;* completed. The routine repeatedly issues a dummy slave address and *
;* checks for an acknowledge from the NM24C16. Once the NM24C16       *
;* responds with an acknowledge the routine terminates.               *
;*************************************************************
POLL:  LCALL  START                   ;issue a START condition
      MOV    A,#0A0H
      LCALL  SENDB                    ;send the dummy slave address
      LCALL  ACK                      ;look for acknowledge from NM24C16
      JC     POLL                     ;loop until acknowledge is received
      LCALL  STOP                     ;issue STOP condition
      RET
```

TL/D/12084–4

```
;*************************************************************;*
*  READ performs a byte read from the address specified in the ADDHI  *
;* and ADDLO variables.  The data that is read is returned in the     *
;* variable RWDATA.                                                   *
;*************************************************************
READ:  LCALL  START                    ;issue a START condition
       MOV    A,ADDHI                  ;issue slave address with R/W=0
       RL     A
       ORL    A,#0A0H
       LCALL  SENDB                    ;send slave address
       LCALL  ACK                      ;get ACK from NM24C16
       MOV    A,ADDLO
       LCALL  SENDB                    ;send low order address
       LCALL  ACK                      ;get ACK from NM24C16
       LCALL  START                    ;issue a START condition
       MOV    A,ADDHI
       RL     A
       ORL    A,#0A1H
       LCALL  SENDB                    ;issue slave address with R/W=1
       LCALL  ACK                      ;get ACK from NM24C16
       LCALL  READB                    ;read data byte
       MOV    RWDATA,A                 ;put data into RWDATA variable
       SETB   SDA                      ;clock in a 1 (no acknowledge)
       LCALL  CLOCK
       LCALL  STOP                     ;issue a STOP condition
       RET

;*************************************************************;*
* START issues a START condition to the NM24C16.  The routine makes *
;* sure that both SDA and SCL are high.  Then bring SDA low first     *
;* followed by bringing SCL low.                                      *
;*************************************************************
START: SETB   SDA                      ;make sure SDA and SCL are high
       SETB   SCL
       CLR    SDA                      ;bring SDA low
       NOP                             ;NOPs assure correct timing
       NOP
       NOP
       NOP
       NOP
       CLR    SCL                      ;bring SCL low
       RET
```

TL/D/12084–5

```
;*************************************************************
;* STOP issues a STOP condition to the NM24C16.  The routine makes *
;* sure that the SDA line is low before trying to issue the STOP.  *
;* The routine then brings SCL high followed by bringing SDA high. *
;*************************************************************
STOP:  CLR    SDA                    ;make sure SDA is low
       SETB   SCL                    ;bring SCL high
       NOP                           ;NOPs assure correct timing
       NOP
       NOP
       NOP
       NOP
       SETB   SDA                    ;bring SDA high
       RET

;*************************************************************;*
*  CLOCK issues a clock pulse to the NM24C16.  The state of SDA is *
;* sampled before the clock pulse is issued.                  *
;*************************************************************
CLOCK: MOV    C,SDA                  ;sample SDA and put state in carry flag
       SETB   SCL                    ;bring SCL high
       NOP                           ;NOPs assure correct timing
       NOP
       NOP
       NOP
       NOP
       CLR    SCL                    ;bring SCL low
       RET

;*************************************************************;*
* ACK allows the NM24C16 to send an acknowledge back to the 8031. *
;*************************************************************
ACK:   SETB   SDA                    ;bring SDA high
       LCALL  CLOCK                  ;issue a clock pulse
       RET

;*************************************************************;*
*  SENDB sends a byte to the NM24C16.  The routine receives the data *
;* to send in the A register.                                 *
;*************************************************************
SENDB: MOV    TDATA,A                ;move data to send into TDATA
       MOV    A,#8                   ;8 bits to send
       MOV    COUNT,A                ;store 8 in down counter
       MOV    A,TDATA                ;return data to send into A register
NEXTR: RLC    A                      ;send most significant bit first
       MOV    SDA,C                  ;move bit to SDA port
       LCALL  CLOCK                  ;issue clock pulse
       DJNZ   COUNT,NEXTR            ;loop 8 times
       RET
```

TL/D/12084–6

```
;*************************************************************;*
 * READB reads a byte from the NM24C16.  The routine returns the byte *
;* that is read in the A register.                             *
;*************************************************************
READB:  MOV     A,#8                    ;8 bits to read
        MOV     COUNT,A                 ;store 8 in down counter
NEXTW:  LCALL   CLOCK                   ;issue clock pulse
        RLC     A                       ;store and shift data from SDA
        DJNZ    COUNT,NEXTW             ;loop 8 times
        RET

        END
```

TL/D/12084–7

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

http://www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 180-530 85 85
English Tel: +49 (0) 180-532 78 32
Français Tel: +49 (0) 180-532 93 58
Italiano Tel: +49 (0) 180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2308
Fax: 81-043-299-2408