

SPI™ Versus MICROWIRE™ EEPROM Comparison

National Semiconductor
Application Note 1012
Dovell Bonnett
September 1995



INTRODUCTION

The SPI (Serial Peripheral Interface) provides a simple eight bit serial port useful in communicating with external devices. Prior to the SPI EEPROMs introduction, engineers used the standard MICROWIRE EEPROMs to interface with the SPI port. MICROWIRE's diverse densities and data protection options offered highly versatile solutions. Recently, several manufacturers have developed serial EEPROMs that are now specifically designed to interface with this port. Due to SPI's faster clock speed and interface compatibility, this EEPROM device is increasing in popularity. The following application note will compare these two EEPROMs and where the advantages exist in each family.

To perform this comparison, a 4 kbit SPI and a 4 kbit MICROWIRE EEPROM are interfaced to the SPI port of a HC11. The NM25C04 SPI is a 512 by 8-bit serial EEPROM, while the NM93C66 MICROWIRE is a 256 by 16-bit.

HARDWARE INTERFACE

HC11 microcontroller's can be configured in one of two modes when utilizing the SPI port. The "master" mode sets the microcontroller to orchestrate data transfer to the peripheral device. The "slave" mode allows the peripheral device to command the HC11. In this application, the HC11 is set as the "master" to control the data transfers to and from the serial EEPROMs. *Figure 1* and *Figure 2* shows the typical SPI and MICROWIRE connection to a HC11 SPI port.

The HC11 SPI port has three lines that control data transfer and one extra general purpose I/O line to control the peripheral device's chip select. The MOSI (Master Out Slave In) line is used as data out, MISO (Master In Slave Out) line is the data input, and the SCK generates the serial clock. The general purpose PD5 line controls the EEPROMs chip select.

Both the SPI and the MICROWIRE devices are configurable as a four wire interface. Thus, neither of the EEPROMs offer a connection advantage.

DATA TRANSFER

The new SPI EEPROMs are specifically designed to interface with the SPI port. Furthermore, it operates at 2.1 MHz versus MICROWIRE's 1 MHz clock. The SPI communication protocol is broken down into byte size sequences containing instruction, address and data transfer information.

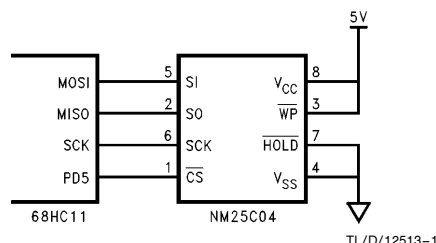


FIGURE 1. NM25C04 to HC11 Connections

MICROWIRE™ is a trademark of National Semiconductor Corporation.
SPI™ is a trademark of Motorola.

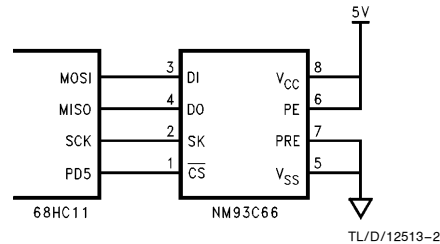


FIGURE 2. NM93C66 to HC11 Connections

Address transfers are dependent on the memory density. Standard MICROWIRE is organized in 16-bit words. To provide the SPI byte size data transfer protocol, three to five bits of additional instructions are required.

Furthermore, a design consideration is required when interfacing a MICROWIRE device to the SPI port. Data is valid relative to the clock signal. Data is written to the EEPROM on the rising edge of the clock. When reading from a MICROWIRE device data is valid on the falling edge.

Development of SPI compatibility is easily accomplished once the design issues are understood. Non-byte wide instruction and address issues can be dealt with by shifting in leading 0's before the required start bit. The leading 0's will not be recognized and can be used to fill out an instruction to byte length. The MICROWIRE word wide architecture can be made to appear byte accessible with clever software. The clock polarity issues can be handled by changing the configuration of the SPI port when receiving or transmitting data.

Both devices were driven with a 1 MHz clock from the SPI port. Communications between the HC11 and the serial EEPROMs took roughly the same amount of time.

SOFTWARE COMPARISON

Two software programs are included that demonstrate the interfacing difference between the HC11 microcontroller to a NM93C66 and a NM25C04. The MICROWIRE data transferring differences, as described in the previous section, only required an additional 38 bytes of program space.

The standard MICROWIRE family is organized in a 16-bit (word wide) manner. SPI is an 8-bit (byte wide) structure. Therefore, the additional software overhead that is required for converting a word organization into a byte organization.

However, there are devices available at National that offers the designer the option of either a x8 or x16 mode. The data size selection is determined by the ORG pin found on the NM93CxxA family. By setting the ORG pin low, the SPI and MICROWIRE devices are comparable in the amount of required programming space.

CONCLUSION

This application note has shown that both the SPI and MICROWIRE families of devices can be effectively interfaced to the HC11. MICROWIRE performance compares favorably with the newer SPI EEPROMs. A few extra bytes of software is the most significant disadvantage when using word wide MICROWIRE mode. In applications where the extra software storage space is available this becomes a non-issue.

When considering availability and price per bit, both families of EEPROMs are quite competitive.

```

*****
* This code was developed to demonstrate how the NM25C04 serial EEPROM
* can be interfaced to the MC68HC11 microcontroller. Basic read and
* write operations have been developed. The software demonstrates the
* following commands:
*
* READ :Read a byte
* WREN :Enable write operations
* WRDI :Disable write operations
* WRITE :Program a byte
*
* The SPI port in Port D is used to interface the NM25C04 to the
* 68HC11. The SPI port provides the clock (SCK), data out (MOSI) and
* the data in (MISO) lines. The 25C04 CS line is driven by a general
* purpose Port D I/O line.
*
* 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 operate as described. The final mainline only
* performs a write enable, write, write disable and finally a read.
*****

*****
* ADDRESS LOCATION EQUATES
*****

        DDRD    EQU        $09        port D direction register = $1009
        PORTD   EQU        $08        port D data register      = $1008
        SPCR    EQU        $28        SPI control register
        SPDR    EQU        $29        SPI status register
        SPDR    EQU        $2A        SPI data register

*****
* BIT POSITION EQUATES
*
*****

        CSBIT   EQU        $20        CS position in port D = bit 5

*****
* VARIABLE ADDRESS EQUATES
*****

        HIADD   EQU        $0180      high order page pointer
        LOADD   EQU        $0181      low order page pointer
        DATVAL  EQU        $0182      data transfer register

*****
* RESET VECTOR
*
*****

        ORG     $FFFE                reset vector to $E000
        FDB     $E000

```

```

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

```

```

      ORG      $E000          program execution begins at $E000
BEGIN: LDS     #$01FF          initialize stack pointer
      LDX      #$1000          initialize "address index register"
      LDAA     #$EF            initialize I/O ports (CS = 1, SCK = 0)
      STAA     PORTD, X
      LDAA     #$3F
      STAA     DDRD, X          SPI bits set to outputs
      LDAA     #$54
      STAA     SPCR, X          initialize SPI port CPOL = 0, CPHA = 1
      LDAA     SPSR, X          reset SPIF bit

```

```

*****
* MAINLINE
*****

```

```

      JSR      WREN            enable write operations
      LDAA     #$01
      STAA     HIADD
      LDAA     #$23
      STAA     LOADD           address = 123H
      LDAA     #$96
      STAA     DATVAL          data = 96H
      JSR      WRITE           write data 96H into address 0123H
      JSR      WRDI            disable writes
      LDAA     #$01
      STAA     HIADD
      LDAA     #$23
      STAA     LOADD           address = 123H
      JSR      READ            read address 123H
LOOP:  BRA     LOOP            wait until reset loop

```

```

*****
* NM25C04 FUNCTIONAL ROUTINES
*****

```

```

*****
* WRITE performs a byte write operation into the 25C04. The routine
* expects the address to modify to be specified in the HIADD and LOADD
* variables. The new data value is specified in the DATVAL variable.
* The 25C04 must be in the write enabled state for this function to be
* executed successfully.
*****

```

```

WRITE: BCLR    PORTD, X  #CSBIT    enable device
        LDAA    HIADD
        ASLA
        ASLA                move high order address to
        ASLA                proper bit location
        ORAA    #$02          OR in WRITE instruction
        JSR     SENDB         send WRITE instruction
        LDAA    LOADD
        JSR     SENDB         send in low order address
        LDAA    DATVAL
        JSR     SENDB         send in data value
        BSET    PORTD, X  #CSBIT    disable device
        JSR     BUSY          wait until write has completed
        RTS

```

```

*****
* READ performs a byte read operation from the NM25C04. The routine *
* expects the address to read to be specified in the HIADD and LOADD *
* variables. The data in the specified address is returned in the *
* DATVAL variable. *
*****

```

```

READ: BCLR    PORTD, X  #CSBIT    enable device
        LDAA    HIADD
        ASLA
        ASLA                move high order address to
        ASLA                proper bit location
        ORAA    #$03          OR in READ instruction
        JSR     SENDB         send READ instruction
        LDAA    LOADD
        JSR     SENDB         send in low order address
        JSR     SENDB         read byte from NM25C04
        STAA    DATVAL
        BSET    PORTD, X  #CSBIT    disable device

```

```

*****
* WREN enables the NM25C04 to perform a write operation. This *
* function along with the WRDI (write disable) function helps to *
* prevent against inadvertant data changes. *
*****

```

```

WREN: BCLR    PORTD, X  #CSBIT    enable device
        LDAA    #$06
        JSR     SENDB         send EWEN instruction
        BSET    PORTD, X  #CSBIT    disable device
        RTS

```

```

*****
* WRDI disables the NM25C04 from further write operations. This *
* function prevents against inadvertant data changes. *
*****

```

```

WRDI:  BCLR   PORTD, X  #CSBIT    enable device
        LDAA   #$04
        JSR    SENDB          send EWDS instruction
        BSET   PORTD, X  #CSBIT    disable device
        RTS

*****
* BUSY is used to pause until a write operation has completed.
*
*****

BUSY:   BCLR   PORTD, X  #CSBIT
        LDAA   #$05
        JSR    SENDB          send "read status reg" instruction
        JSR    SENDB          read status register
        BSET   PORTD, X  #CSBIT
        ANDA   #$01
        BNE    BUSY          loop until RDY bit in status is low
        RTS

*****
* SENDB is used to send a byte to the NM25C04 and also read the data
* that has been returned to the 68HC11.
*
*****

SENB:   STAA   SPDR, X          send byte in A register
PAUSE:  BRCLR  SPSR, X  #$80 PAUSE wait until byte has been sent
        LDAA   SPDR, X          read byte into a register
        RTS

*****
* This code was developed to demonstrate how the NM93C66 serial EEPROM
* can be interfaced to the MC68HC11 microcontroller SPI port. The
* software includes several subroutines that perform various interface
* functions. The internal architecture of the NM93C66 is configured
* with word wide data registers. This applications code demonstrates
* storage of data in a byte wide manner. Odd byte addresses are stored
* in the D8-D15 bits of each word and even address data is stored in
* the D0-D7 bits. The software demonstrates the following commands:
*
*
*   READ   :Read a byte
*   WEN    :Enable write and erase operations
*   WDS    :Disable write and erase operations
*   WRITE  :Program a byte
*
*
* The 68HC11 interfaces to the NM93C66A by using 3 lines from the SPI
* port and a general purpose I/O port bit. The 68HC11 SCK, MOSI and
* MISO pins are used to drive the NM93C66 SK, DI and D0 pins
* respectively. Port D bit 5 is used to drive the CS line of the
* NM93C66.
*
*
* 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 operate as described. The final mainline only
* performs a write enable, write, write disable and finally a read.
*
*****

```

```

*****
* ADDRESS LOCATION EQUATES
*****

        DDRD    EQU    $09        port D direction register = $1009
        PORTD   EQU    $08        port D data register    = $1008
        SPCR    EQU    $28        SPI control register
        SPSR    EQU    $29        SPI status register
        SPDR    EQU    $2A        SPI data register

*****

* BIT POSITION EQUATES
*
*****

        CSBIT   EQU    $20        CS position in port D = bit 5

*****

* VARIABLE ADDRESS EQUATES
*****

        HIADD   EQU    $0180      high order page pointer
        LOADD   EQU    $0181      low order page pointer
        DATVAL   EQU    $0182      data transfer register
        TEMP    EQU    $0183      temporary "scratch" register

*****

* RESET VECTOR
*
*****

        ORG     $FFFE            reset vector to $E000
        FDB     $E000

*****

* PROGRAM STARTING LOCATION
*****

        ORG     $E000            program execution begins at $E000
BEGIN: LDS     #$01FF            initialize stack pointer
        LDX     #$1000            initialize "address index register"
        LDAA    #$CF             initialize I/O ports (CS = 0, SCK = 0)
        STAA    PORTD, X
        LDAA    #$3F
        STAA    DDRD, X          SPI bits set to outputs
        LDAA    #$50
        STAA    SPCR, X          initialize SPI port
        LDAA    SPSR, X          reset SPIF bit

*****

* MAINLINE
*
*****

```

```

        JSR      WREN                enable write operation
        LDAA     #$01
        STAA     HIADD
        LDAA     #$23
        STAA     LOADD
        LDAA     #$96
        STAA     DATVAL
        JSR      WRITE                write data 96H into address 0123H
        JSR      WDS                  disable writes
        LDAA     #$01
        STAA     HIADD
        LDAA     #$23
        STAA     LOADD
        JSR      READ                  read address 123H
LOOP:   BRA      LOOP                wait until reset loop

```

```

*****
* 98C66 FUNCTIONAL ROUTINES *
*****

```

```

*****
* WRITE performs a byte write operation into the 93C66. The routine *
* expects the address to modify to be specified in the HIADD and LOADD *
* variables. The new data value is specified in the DATVAL variable. *
* The 93C66 must be in the write enabled state for this function to be *
* executed successfully. Each word in the NM93C66 contains 2 bytes with *
* the high order byte used for "odd" addresses and the low order byte used *
* for "even" addresses. The word addresses to be accessed is determined by *
* dividing the byte address specified in HIADD and LOADD by two. The data *
* to be written is determined by reading the specified word address and *
* mapping in the new byte value into the specified high or low order byte. *
*****

```

```

WRITE: LDAA    DATVAL                save the data value
      PSHA
      LDAA    HIADD                  save the address
      PSHA
      LDAA    LOADD
      PSHA
      EORA    #$01                  determine the byte address in the
      STAA    LOADD                  word that will not be modified
      JSR     READ                  read the valid byte
      PULA
      STAA    LOADD                  retrieve address to modify
      PULA
      STAA    HIADD
      PULA
      STAA    TEMP                  retrieve new data value
      ROR     HIADD                  calculate word address
      ROR     LOADD
      BCC     NOFLIP                if carry = 0 then we are set
      PSHA                                save new data value
      LDAA    DATVAL                transfer valid byte in word
      STAA    TEMP                  into TEMP
      PULA                                store new data byte value
      STAA    DATVAL                into DATVAL
NOFLIP: BSET    PORTD, X    #CSBIT    enable device
      LDAA    #$05
      JSR     SENDB                send WRITE instruction
      LDAA    LOADD
      JSR     SENDB                send word address to modify
      LDAA    DATVAL
      JSR     SENDB                send high order data byte
      LDAA    TEMP
      JSR     SENDB                send low order data byte
      BCLR    PORTD, X    #CSBIT    disable device
      JSR     BUSY                wait until write has completed
      RTS

```

```

*****
* READ performs a byte read operation from the 93C66. The routine *
* expects the address to read to be specified in the HIADD and LOADD *
* variables. The data in the specified address is returned in the *
* DATVAL variable. Each word in the NM93C66 contains 2 bytes with the *
* high order byte used for "odd" addresses and the low order byte used *
* for "even" addresses. The word address to be accessed is determined *
* by dividing the byte address specified in HIADD and LOADD by two. *
*****

```

```

READ:  BSET   PORTD, X  #CSBIT    enable device
        LDAA   #$06
        JSR    SENDB          send READ instruction
        ROR    HIADD
        ROR    LOADD
        LDAA   LOADD
        JSR    SENDB          send address
        LDAA   #$54
        STAA   SPCR, X        sample on falling edge when reading
        JSR    SENDB          read data value
        BCS    DONER
        JSR    SENDB
DONER:  STAA   DATVAL
        BCLR   PORTD, X  #CSBIT    disable device
        LDAA   #$50
        STAA   SPCR, X        data valid on rising edge
        RTS

*****
* WEN enables the 93C66 to perform a write operation. This function      *
* along with the WDS (write disable) function helps to prevent against   *
* inadvertant data changes.                                              *
*****

WEN:    BSET   PORTD, X  #CSBIT    enable device
        LDAA   #$04
        JSR    SENDB          send WEN instruction
        LDAA   #$C0
        JSR    SENDB          send instruction + dummy address
        BCLR   PORTD, X  #CSBIT    disable device
        RTS

*****
* WDS disables the 93C66 from further write operations. This            *
* function prevents against inadvertant data changes.                    *
*****

WDS:    BSET   PORTD, X  #CSBIT    enable device
        LDAA   #$04
        JSR    SENDB          send WDS instruction
        LDAA   #$00
        JSR    SENDB          send instruction + dummy address
        BCLR   PORTD, X  #CSBIT    disable device
        RTS

*****
* SUPPORT ROUTINES                                                         *
*****

```

```

*****
* BUSY is used to pause until a write or erase operation has completed. *
*****

```

```

BUSY:  BSET    PORTD, X  #CSBIT    enable device
TWC:   BRCLR   PORTD, X  #$04      TWC wait until write cycle has finished
      BCLR    PORTD, X  #CSBIT    disable device
      RTS

```

```

*****
* SENDB is used to send a byte to the NM93C66 and also read the data *
* that has been returned to the 68HC11. *
*****

```

```

SENDER: STAA    SPDR, X      send byte in A register
PAUSE:  BRCLR   SPSR, X  #$80 PAUSE wait until byte has been sent
      LDAA     SPDR, X      read byte in A register
      RTS

```

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	National Semiconductor Europe Fax: (+49) 0-180-530 85 86 Email: cnjwge@tevm2.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. 19th 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: 81-043-299-2309 Fax: 81-043-299-2408
---	---	---	--	--

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.