AN1183

Interfacing PIC18 MCUs with UNI/O® Bus-Compatible Serial EEPROMs

Author: Chris Parris
Microchip Technology Inc.

INTRODUCTION

As embedded systems become smaller, a growing need exists to minimize I/O pin usage for communication between devices. Microchip has addressed this need by developing the UNI/O® bus, a low-cost, easy-to-implement solution requiring only a single I/O pin for bidirectional communication.

UNI/O bus-compatible serial EEPROMs can be used to enhance any application facing restrictions on available I/O. Such restrictions can potentially stem from connectors, board space, or from the master device itself.

The 11XXX family is the newest addition to Microchip Technology's broad serial EEPROM product line, and is compatible with the newly developed UNI/O bus.

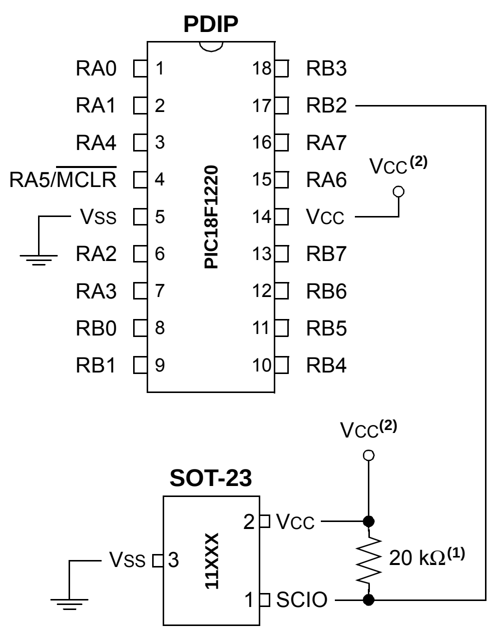
The main features of 11XXX serial EEPROMs are:

- Single I/O pin used for communication
- EEPROM densities from 1 Kb to 16 Kb
- Extremely small packages
- Bus speed from 10 kHz up to 100 kHz
- Voltage range from 1.8V to 5.5V
- Low-power operation
- Temperature range from -40°C to +125°C
- Over 1,000,000 erase/write cycles

This application note is part of a series that provide source code to help the user implement the protocol with minimal effort.

Figure 1 describes the hardware schematic for the interface between the Microchip 11XXX series of UNI/O bus-compatible serial EEPROMs and the PIC18F1220 microcontroller. The schematics show the connections necessary between the microcontroller and the serial EEPROM as tested. The software was written assuming these connections. The single I/O connection between the microcontroller and the serial EEPROM includes a recommended pull-up resistor.

FIGURE 1: CIRCUIT FOR PIC18F1220 AND 11XXX SERIAL EEPROM



Note 1: A pull-up resistor (typically 20 kΩ) on SCIO is recommended to ensure bus idle during power-up.
2: Decoupling capacitors (typically 0.1 µF) should be used to filter noise on Vcc.

FIRMWARE DESCRIPTION

The purpose of the firmware is to show how to generate specific UNI/O bus transactions using a general I/O pin on the microcontroller. The focus is to provide the designer with a strong understanding of communication with the 11XXX serial EEPROMs, thus allowing for more complex programs to be written in the future. The firmware was written in assembly language and tested using the Microchip PICDEM™ 4 development board. The code can easily be modified to use any I/O pin that is available.

No additional libraries are required with the provided code. The main program is organized into five sections:

- Initialization
- Write Enable
- Page Write
- WIP Polling
- Sequential Read

The program utilizes the WIP polling feature for detecting the completion of the write cycle after the page write operation. The read operation allows for verification that the data was properly written. No method of displaying the input data is provided, but an oscilloscope can be used.

The code was tested using the 11LC160 serial EEPROM. This device features 2K x 8 (16 Kbits) of memory and 16-byte pages. Oscilloscope screen shots are labeled for ease in reading. The data sheet versions of the waveforms are shown below the oscilloscope screen shots. The internal 8 MHz RC oscillator is used to clock the microcontroller. If a different clock is used, the code must be modified to generate the proper timings. All values represented in this application note are hex values unless otherwise noted.

BIT PERIOD TIMING

Subroutine Overhead

For this application note, a timer module on the PIC® microcontroller was not used. Therefore, in order to maintain accurate timing, all instructions executed during communications must be taken into account. All of the provided subroutines have been designed to have the same amount of overhead. This means that the same number of instructions must be used between calls to each subroutine. The necessary number of instructions is defined as a constant named '`USERCODE`', located within the 'UNIO PIC18.inc' file. The constants '`PRE`' and '`POST`' specify the overhead within the subroutines, and should not be modified unless the subroutines themselves are changed. In Example 1, '`USERCODE`' is set to 3, and so a '`BRA`' instruction is required to ensure 3 instructions are executed between subroutine calls.

Figure 2 shows how the '`PRE`', '`USERCODE`', and '`POST`' constants determine the bit period, and Equation 1 shows how to calculate the bit period based on these constants. In this example, because each half of the period must be balanced, one period contains 54 instructions. With T_{CY} = 500 ns, this equates to 27 µs per bit period, or 37.04 kbps. If additional instructions are needed between subroutine calls, then the '`USERCODE`' constant can be modified. It is important that the proper number of instructions, as defined by '`USERCODE`', are always used between subroutine calls within a command. Note that changing the number will also affect the bit period.

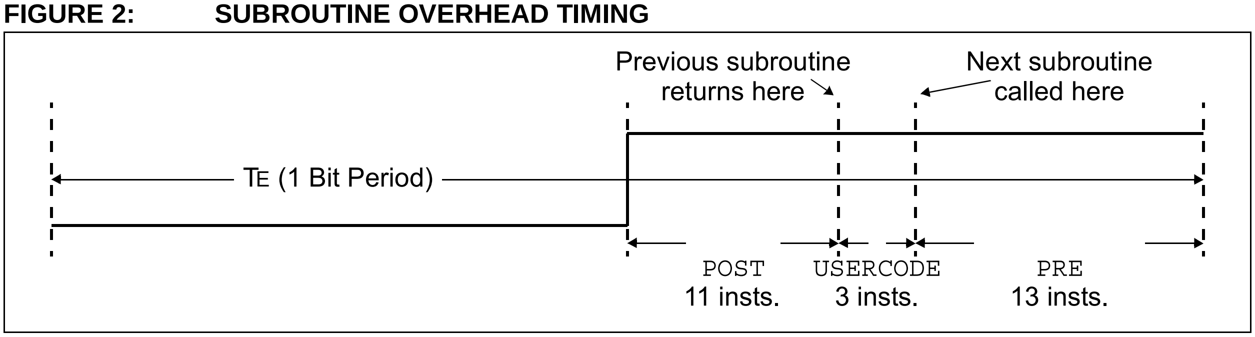
EQUATION 1: BIT PERIOD

$$T_E = 2 \cdot (PRE + POST + USERCODE) \cdot T_{CY}$$

EXAMPLE 1: SUCCESSIVE SUBROUTINE CALLS

```
RCALL    OutputByte        ; Output byte
MOVLW    WRITE_CMD         ; Load command into WREG (1 inst)
BRA      $+2               ; Delay to ensure 3 insts. between calls (2 insts)
RCALL    OutputByte        ; Output byte
```

FIGURE 2: SUBROUTINE OVERHEAD TIMING



Achieving Necessary Delays

In order to ensure the proper timings are met, loops have been placed at the necessary locations within the code. A simple macro, shown in Example 2, was developed to achieve these loops.

The total number of instructions necessary for the desired delay is passed as the 'numinsts' argument, while a unique label is passed as the 'looplabel' argument. The macro will calculate the number of loops necessary to achieve the specified delay, and will also generate an additional NOP or GOTO instruction to account for errors in rounding.

To enable the constants shown above to be modified easily, equations have been used for each location where the macro is called. These equations should not be modified unless the subroutine code has been changed and a different delay is needed.

EXAMPLE 2: DELAYLOOP MACRO

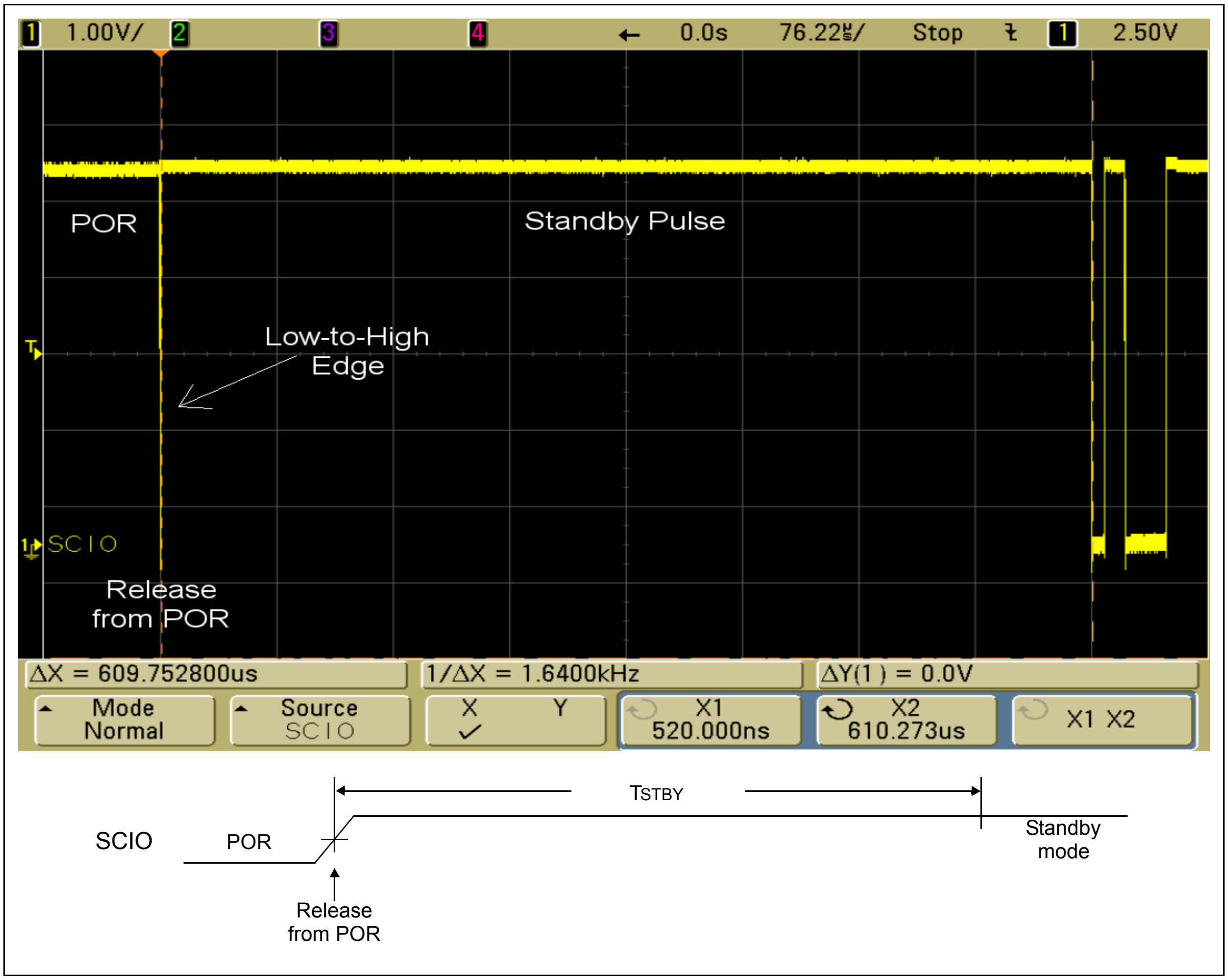
```
DELAYLOOP      MACRO numinsts, looplabel
     MOVLW     (numinsts-.1)/.3      ; Load count into WREG
     MOVWF     delayCount            ; Copy WREG to delayCount
looplabel      ; Each loop is 3 inst. (2 for last loop)
     DECFSZ    delayCount,F          ; Decrement delayCount, check if 0
     BRA       looplabel             ; If not 0, keep looping

; Now account for miscalculations by adding instructions. This also accounts
; for the loop executing only 2 instructions for the last count value.
#if (numinsts%.3)==.0                ; Account for 2-inst miscalculation
     BRA       $+2
#else
#if (numinsts%.3)==.2                ; Account for 1-inst miscalculation
     NOP
#endif
#endif
    endm
```

INITIALIZATION

Before initiating communication with the 11XXX, the master device (MCU) must generate a low-to-high edge on SCIO to release the serial EEPROM from Power-On Reset (POR). Because bus idle is high, the MCU creates a high-low-high pulse on SCIO. Once the serial EEPROM has been released from POR, a standby pulse with a minimum timing of T_{STBY} is performed to place the serial EEPROM into Standby mode, as shown in Figure 3.

Note that once a command has successfully executed – indicated by the reception of a Slave Acknowledge (SAK) following the No Master Acknowledge (NoMAK) – the serial EEPROM enters Standby mode immediately and a standby pulse is not necessary. In this case, only the Start Header Setup time (T_{SS}) must be observed before the MCU may initiate another command to the same device.

FIGURE 3: STANDBY PULSE



WRITE ENABLE

Before a write operation to the array or the STATUS register can occur, the Write Enable Latch (WEL) must be set. This is done by issuing a Write Enable (`WREN`) instruction.

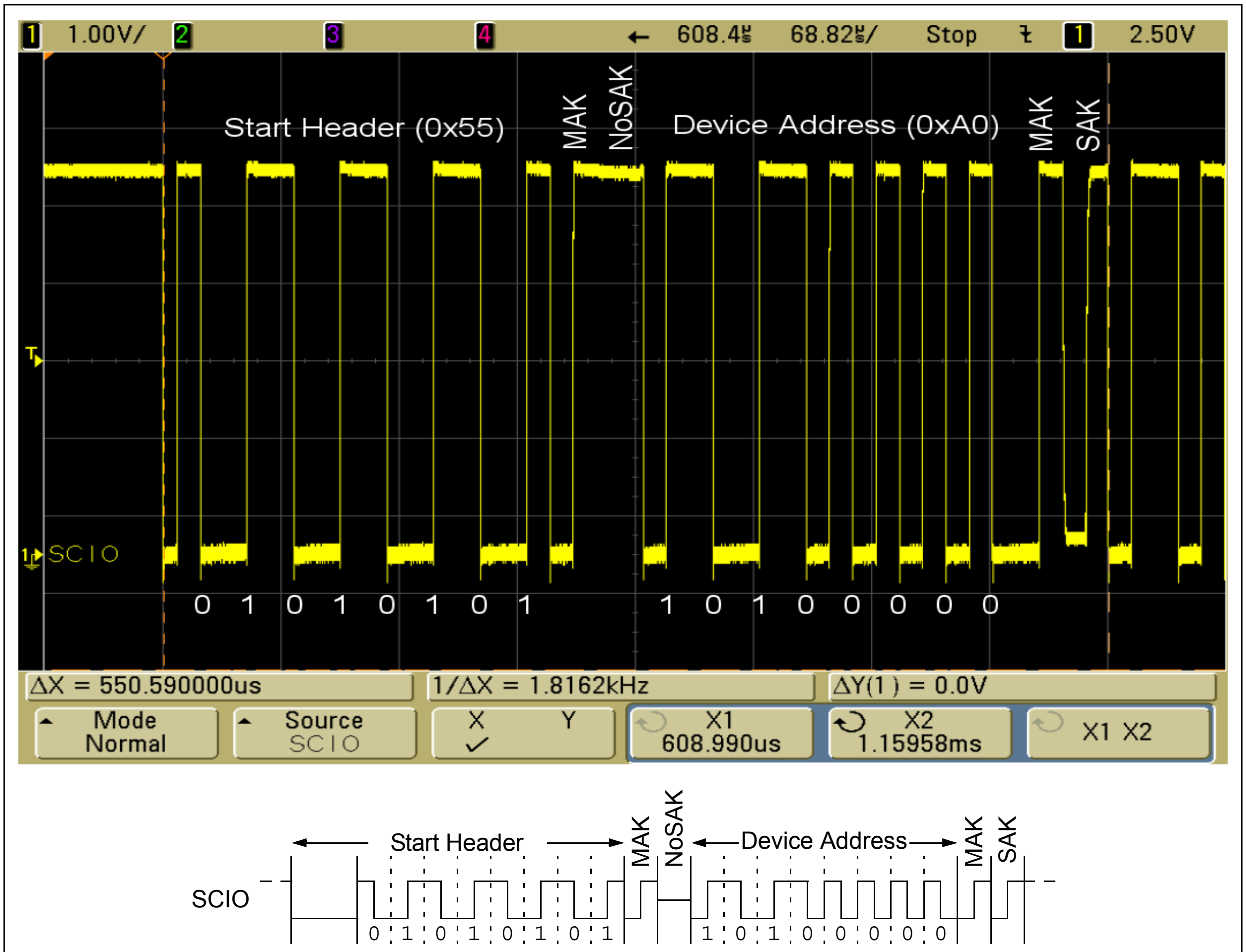
The WEL can be cleared by issuing a Write Disable (`WRDI`) instruction. It is also cleared upon termination of a write cycle to either the array or STATUS register, and upon POR.

The Write Enable operation has been broken down into the following components: the start header, which is followed by the device address and the command byte.

Start Header and Device Address

To issue a `WREN` instruction, the MCU transmits the start header. This consists of a low pulse (T_{HDR}), followed by `'01010101'`, and a Master Acknowledge (MAK), followed by a NoSAK. Next, the MCU transmits the device address (`'10100000'`) and another MAK. The serial EEPROM then responds with a SAK if the start header and device address were received correctly. Figure 4 shows the details of the start header and device address.

FIGURE 4: START HEADER AND DEVICE ADDRESS

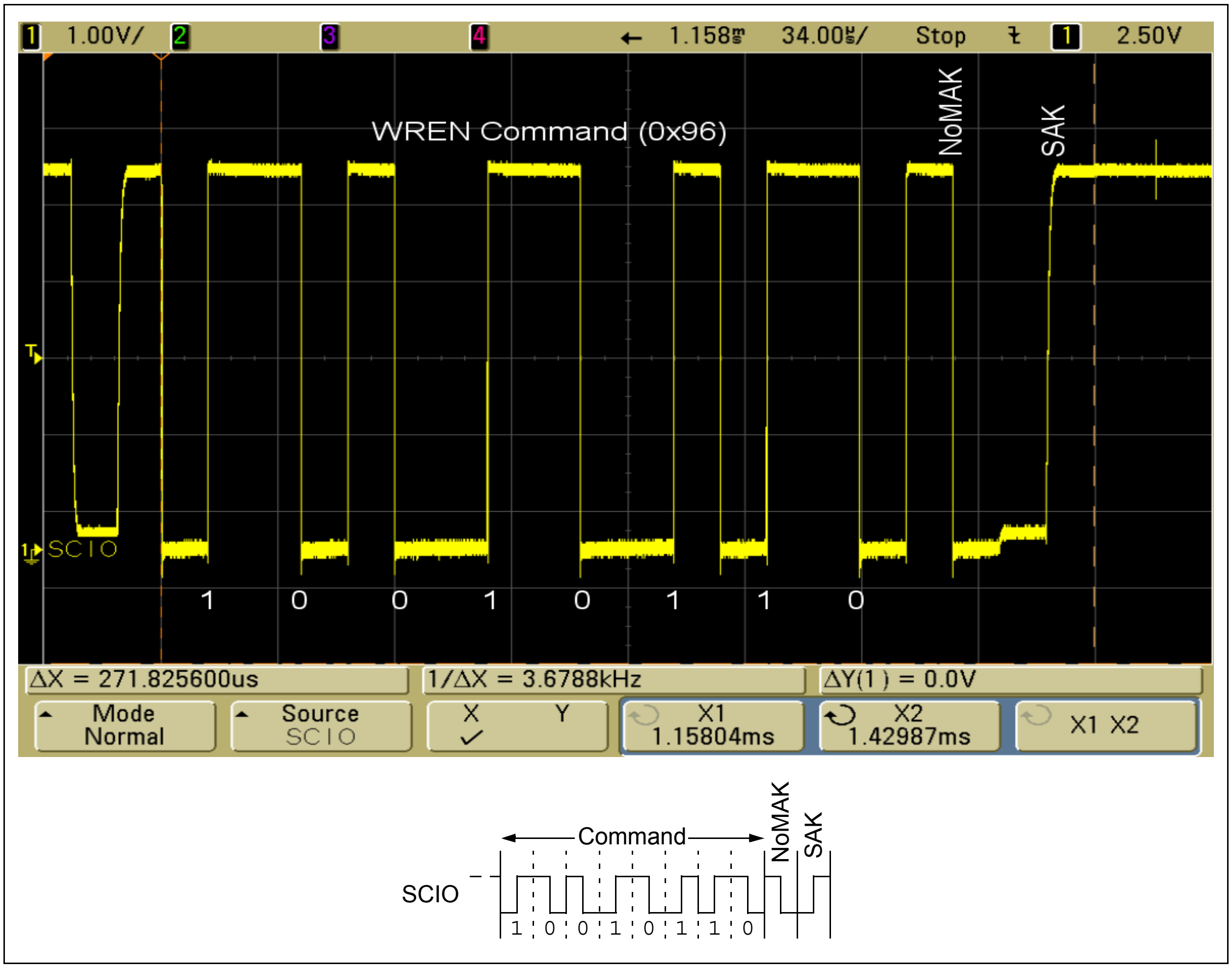


Write Enable (WREN) Command Byte

Once the SAK is received following the device address, the MCU sends the WREN command byte ('`10010110`' or 0x96) and performs a final Acknowledge sequence. During this last sequence, the MCU sends a NoMAK to signal the end of the operation. Once again, the serial EEPROM responds with a SAK, indicating it received the byte successfully.

Figure 5 shows an example of the WREN command byte.

FIGURE 5: WRITE ENABLE COMMAND



PAGE WRITE

Once the WREN instruction has been performed, a page write operation can be executed to write data to the array. The serial EEPROM features a 16-byte page, so up to 16 bytes of data can be written within a single operation.

The page write operation consists of the following components: the Write command, followed by the word address and the data bytes. Note that the start header and device address are not illustrated in this section but are still required to initiate the operation.

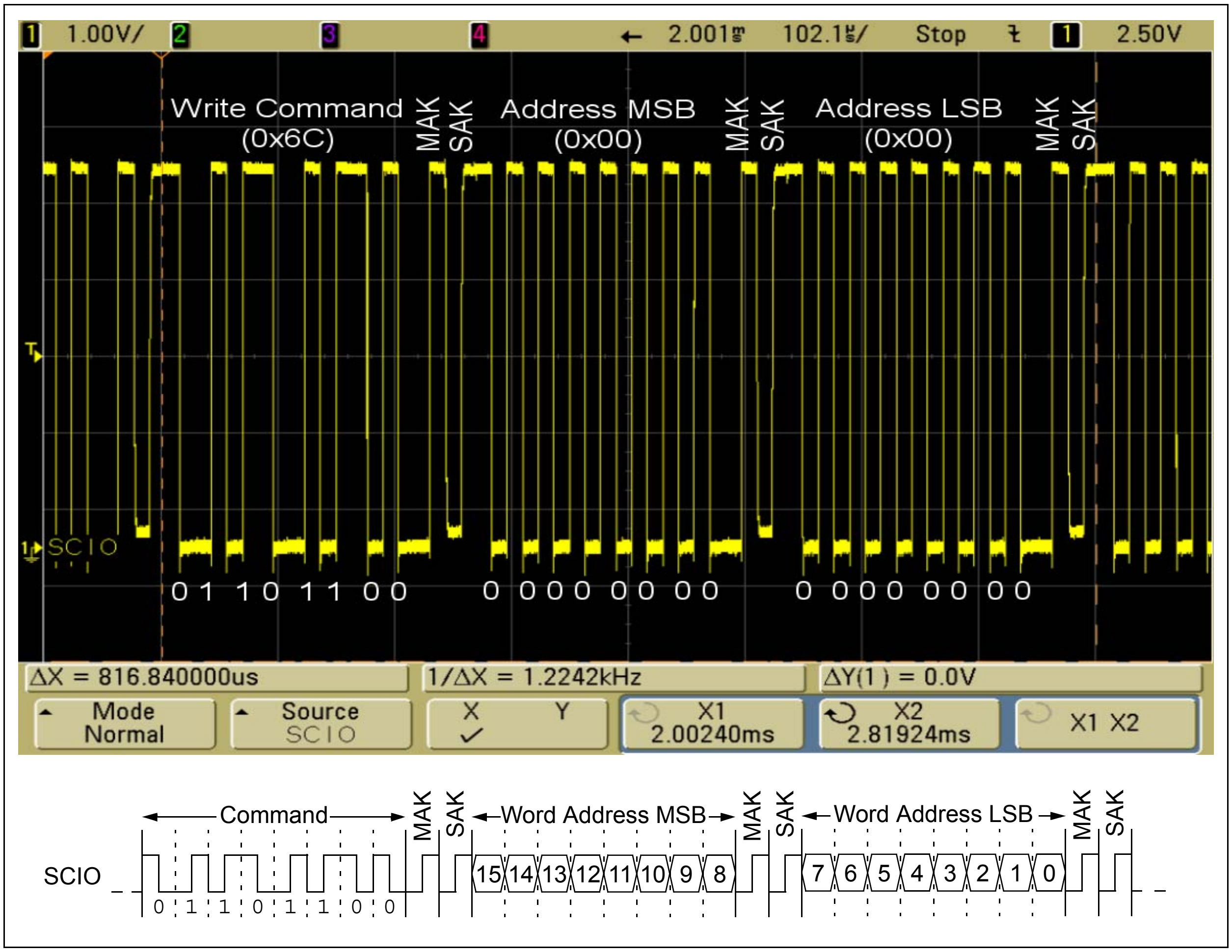
Before beginning the WRITE instruction, a period of TSS must be observed following the WREN operation. This period can be used in place of the standby pulse after a command has been executed successfully when addressing the same slave device. After the TSS period, the start header and device address are transmitted as described on page 5.

Write Command and Word Address

After the start header and device address have been sent, the MCU transmits the Write command ('01101100' or 0x6C) and the word address. The serial EEPROM uses a 16-bit word address to access the array, so two bytes must be transmitted for the entire word address, with the Most Significant Byte sent first. After every byte, the MCU transmits a MAK and the serial EEPROM responds with a SAK.

Figure 6 shows an example of the Write command and the word address.

FIGURE 6: WRITE COMMAND AND WORD ADDRESS



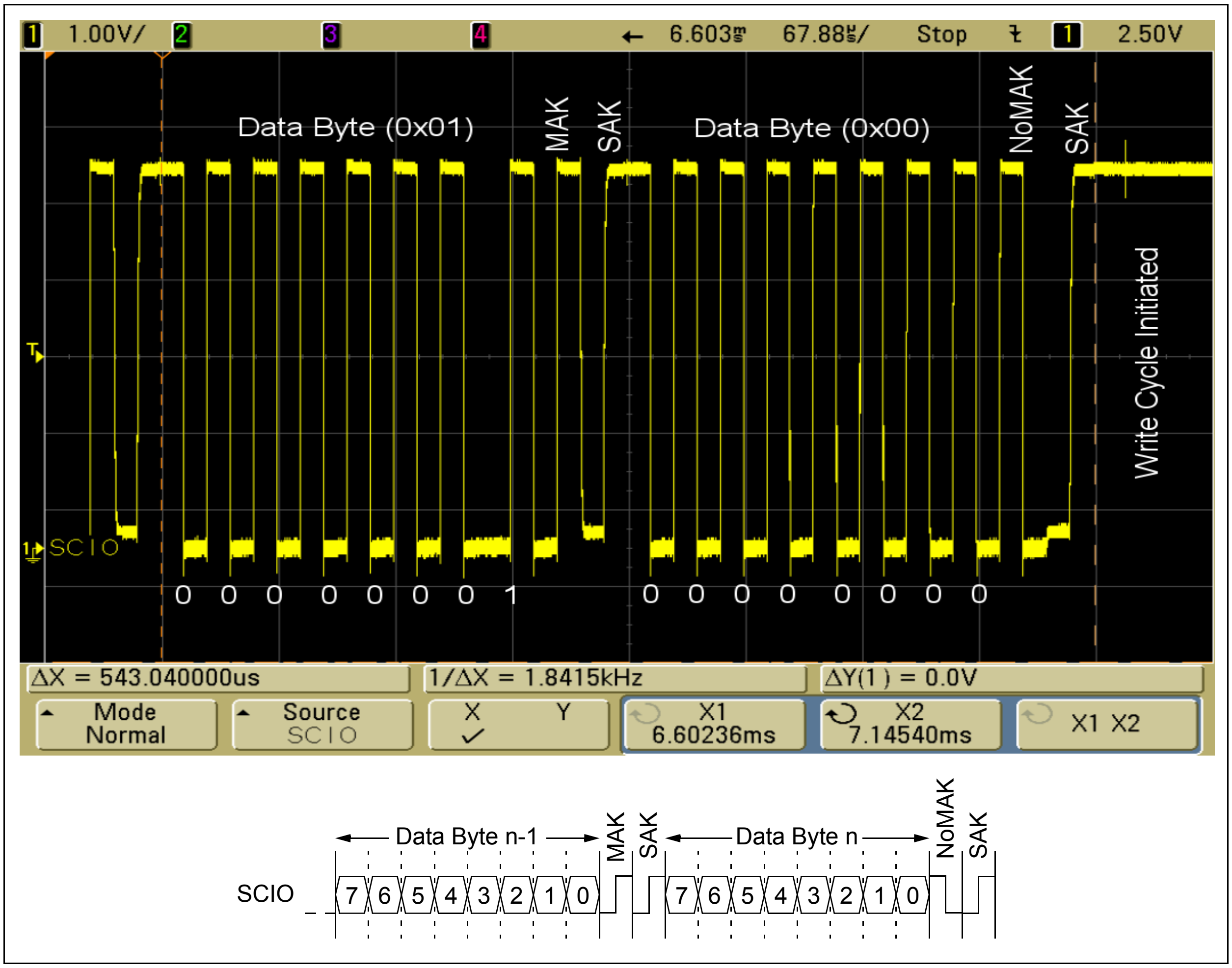
Data Bytes

Once the word address has been transmitted and the last SAK has been received, the data bytes can be sent. Up to 16 bytes of data can be sent within a single operation. After each byte is transmitted, the MCU sends a MAK and the serial EEPROM responds with a SAK. If at any point a NoSAK is received, then an error has occurred and the operation must be restarted, beginning with a standby pulse.

Once all data bytes have been sent, the MCU terminates the command by generating a NoMAK in place of the MAK, and the serial EEPROM again responds with a SAK. This also initiates the internal write cycle (T_{WC}).

Figure 7 shows the final two data bytes sent by the MCU, as well as the NoMAK and SAK.

FIGURE 7: WRITE COMMAND FINAL TWO DATA BYTES



WRITE-IN-PROCESS POLLING

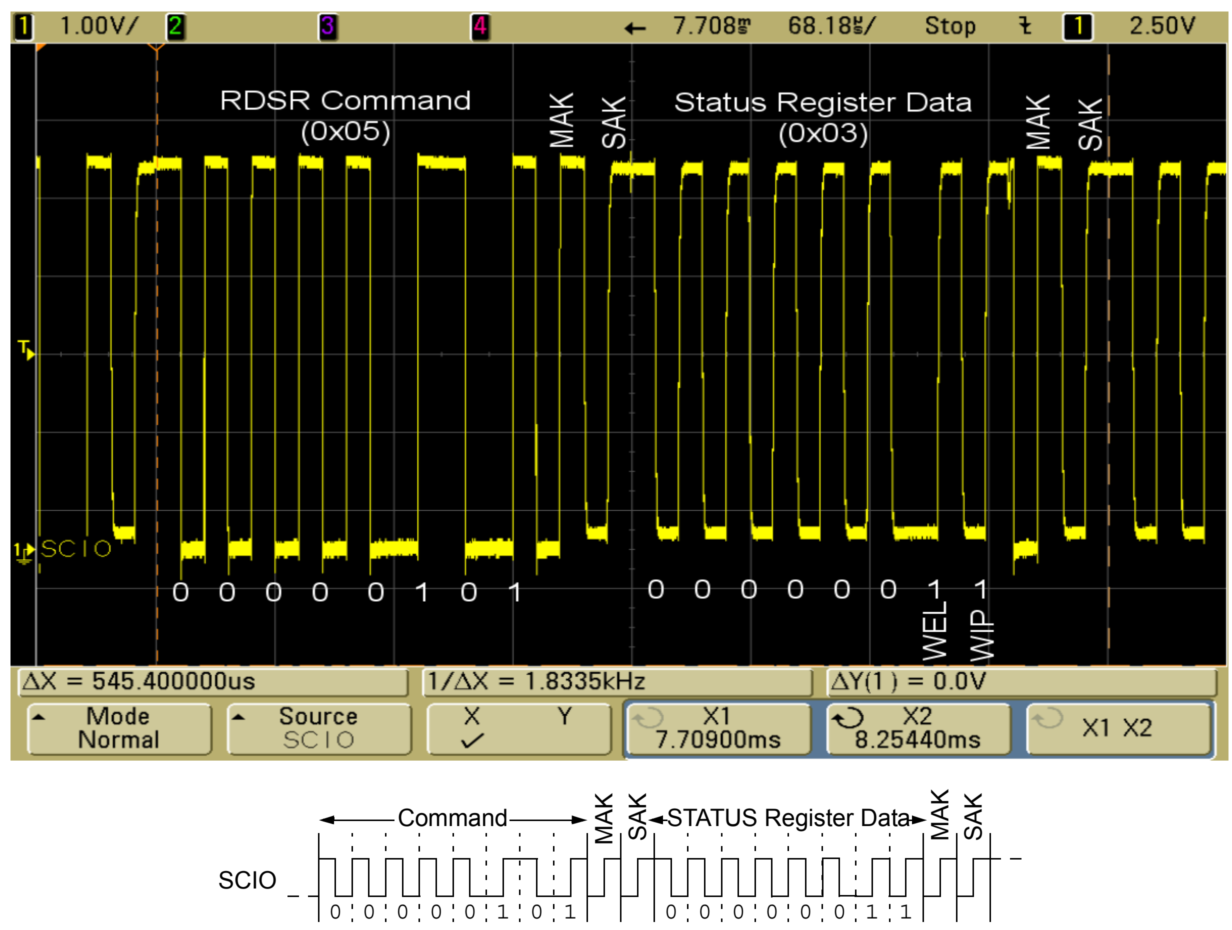
After an array or STATUS register write instruction is executed, the MCU must observe a write cycle time (T_{WC}). Write cycle time is a maximum, so the actual time required is typically less. Therefore, to transfer data as efficiently as possible, using the Write-In-Process (WIP) polling feature is highly recommended. Because the STATUS register can be read during a write cycle, the WIP bit can be continuously monitored to determine the completion of the write cycle.

Write-In-Process Polling Routine

The process of WIP polling consists of the MCU sending a start header and device address after observing the T_{SS} period. The MCU follows this by sending the Read Status Register (RDSR) command (`00000101' or 0x05). After sending the subsequent SAK, the serial EEPROM transmits the STATUS register. At this point, the STATUS register can be requested again by sending a MAK. The WEL and WIP values sent are updated dynamically, so the MCU can continuously check the STATUS register. Sending a NoMAK terminates the command.

Figure 8 shows an example of WIP polling to check if a write operation has finished. In this example, the WIP bit is set ('1'), which indicates that the write cycle has not yet completed.

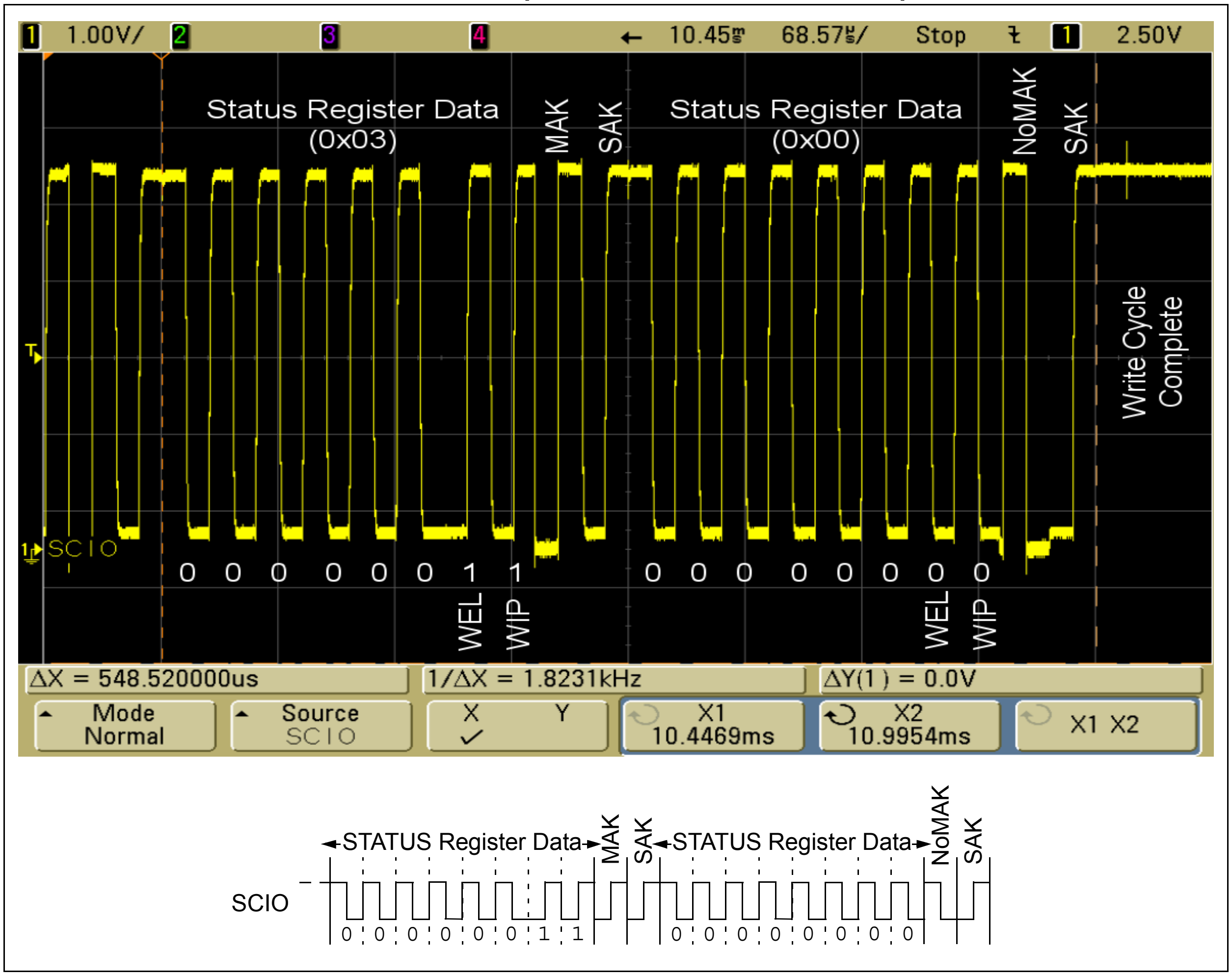
FIGURE 8: WIP POLLING ROUTINE (SHOWING WRITE-IN-PROCESS)



WIP Polling Complete

Figure 9 shows the final read of the STATUS register after the page write operation, in which the WIP bit is clear (‘0’). This indicates that the write cycle is complete and the serial EEPROM is ready to continue.

FIGURE 9: WIP POLLING FINISHED (SHOWING WRITE COMPLETE)



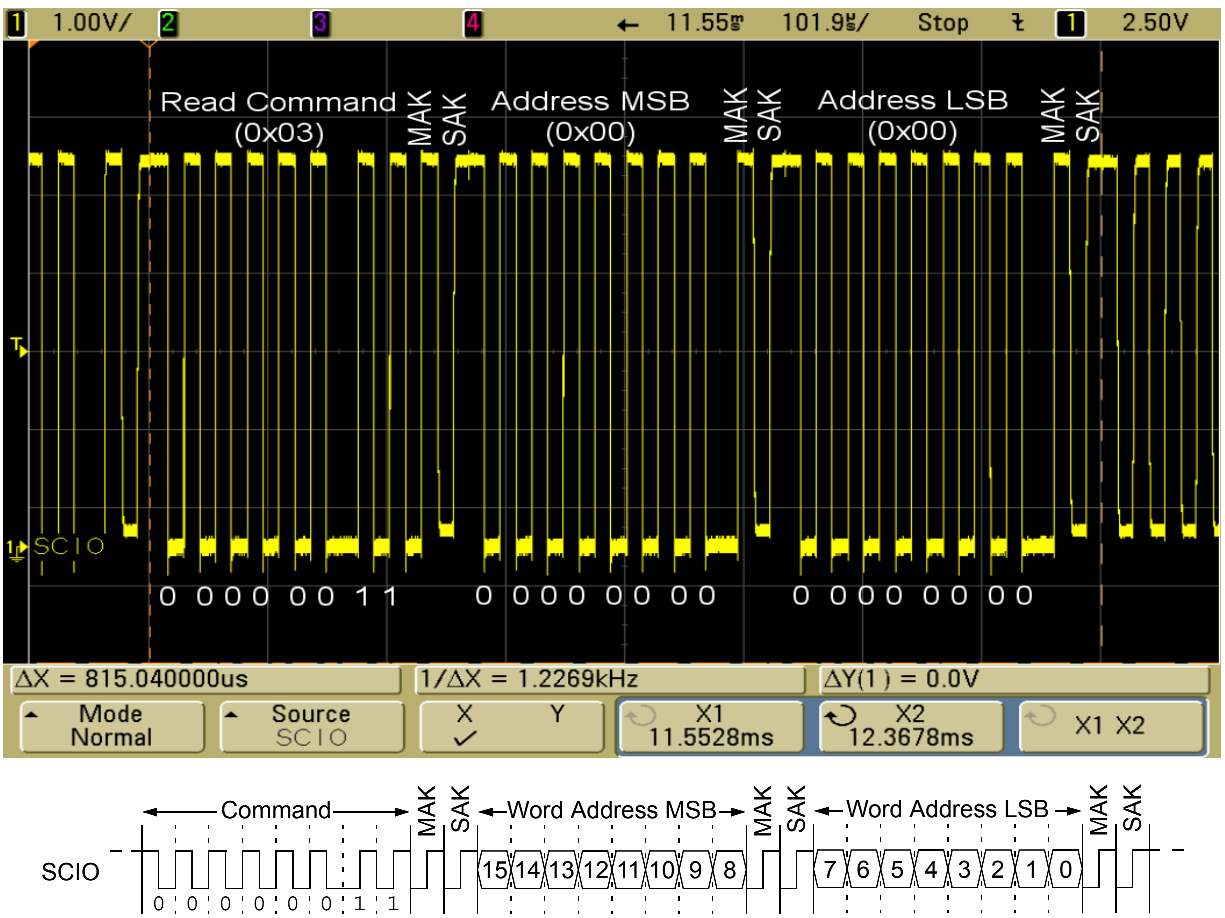
SEQUENTIAL READ

The serial EEPROM allows data to be read from the array in a random access manner. Reading data from the array is very similar to the write operation, except that the read is not limited to a single page. In order to read from the array, the start header and device address must first be sent after observing the TSS period. The Read command byte and word address bytes are transmitted next. The MCU generates a MAK after every byte, and the serial EEPROM responds with a SAK if no errors occurred.

Command and Word Address for Read

Figure 10 shows an example of the Read command (`'00000011'` or 0x03) followed by the word address.

FIGURE 10: READ – COMMAND BYTE AND WORD ADDRESS



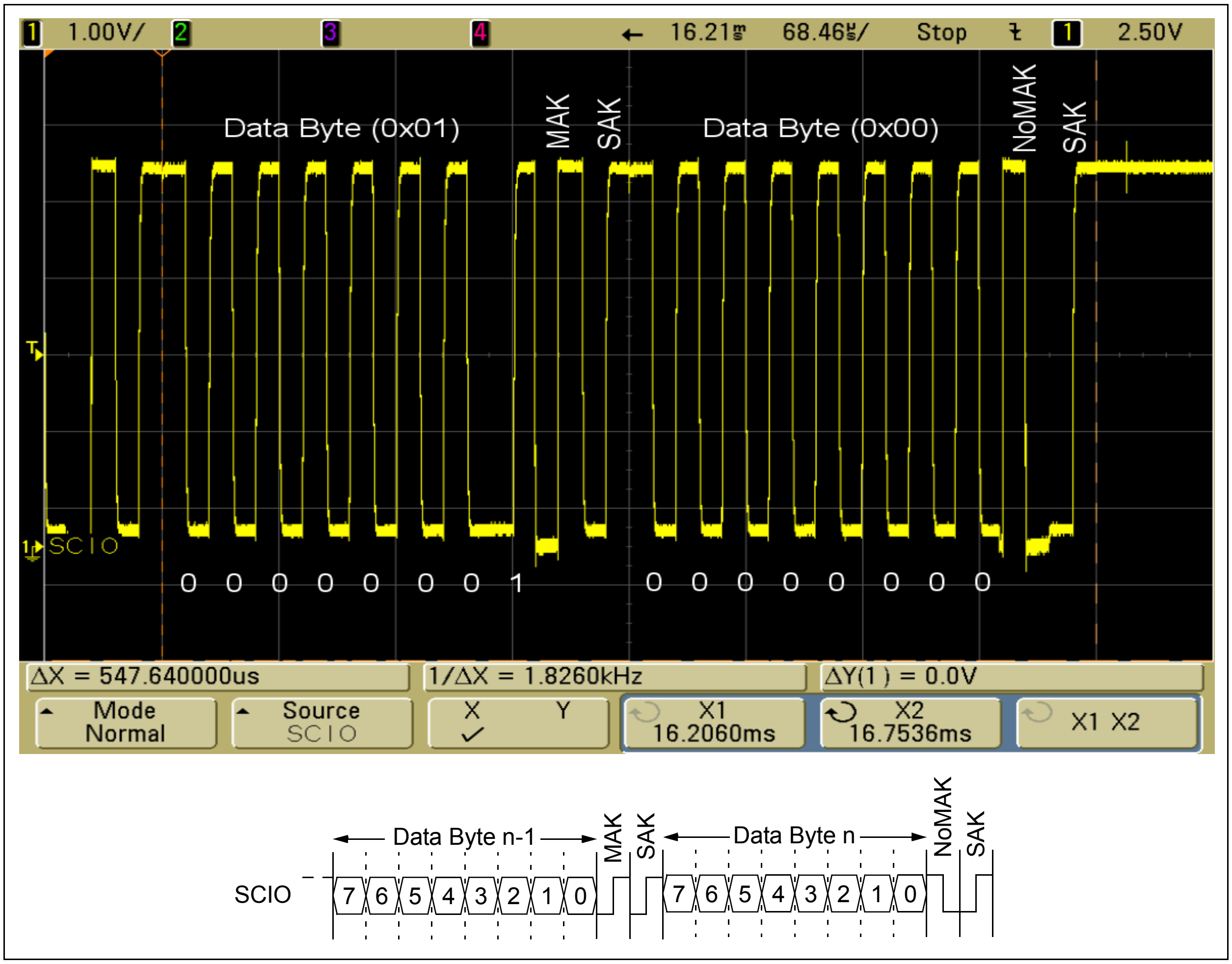
Reading Data Back

After the Read command and word address have been sent and acknowledged, the serial EEPROM sends the first data byte from the array, starting at the address specified. In order to continue the read, the MCU must send a MAK after each data byte, with the serial EEPROM responding with a SAK if there are no errors. After each data byte has been sent, the serial EEPROM automatically increments the internal word address to output the next data byte.

The read operation is not limited to a single page, so the entire array can be read within a single operation if the MCU continues to request data. At the end of the array, the internal word address is automatically reset back to 0x000. A NoMAK terminates the operation.

Figure 11 shows the MCU reading the final two bytes of data. The MCU sends a NoMAK after the last byte to indicate that no more data is requested and to terminate the command.

FIGURE 11: READ – FINAL TWO DATA BYTES



CONCLUSION

This application note provides examples of the basic commands for communicating with the UNI/O bus-compatible family of serial EEPROMs. These functions are designed to be used in an end application with very little modification. The code generated for this application note was tested using the PICDEM4 demonstration board with the connections shown in Figure 1.

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/02/08