AN979

Interfacing I^{2}C^{TM} Serial EEPROMs to PIC18 Devices

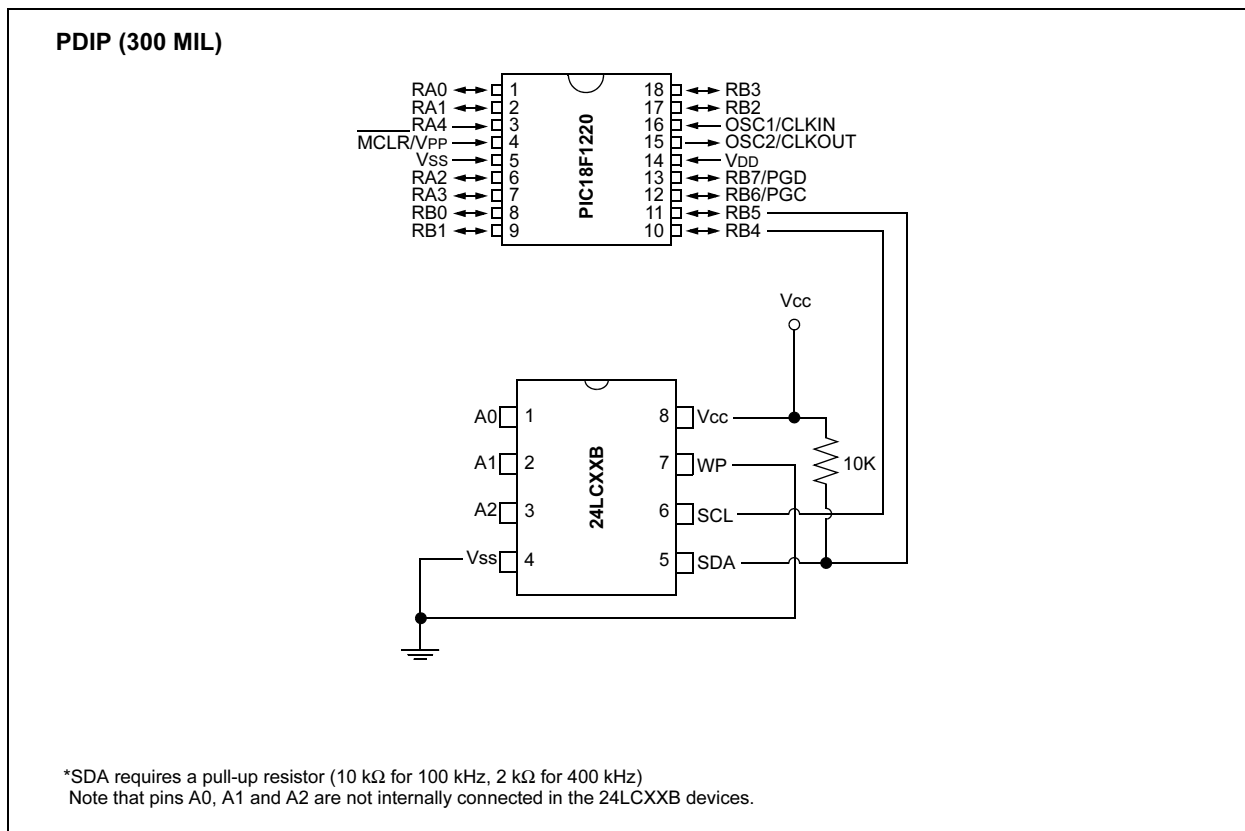| Author: | Chris Parris |
| | Microchip Technology Inc. |

INTRODUCTION

The 24LCXXB Serial EEPROMs from Microchip Technology are I^{2}C^{TM} compatible and support both the standard 100 kHz and the 400 kHz Fast mode. Many times when designing an application which utilizes a serial EEPROM device, it may be beneficial to use a microcontroller which does not feature a dedicated protocol-specific serial port. This can be due to several possible reasons, including size restrictions or costs. In these instances, it is required of the designer to write software routines capable of generating the proper signals for communicating with the EEPROM device.

This application note is intended to serve as a reference for manually communicating with Microchip's 24LCXXB serial EEPROM devices, that is, without relying on a hardware serial port to handle the I^{2}C operations. Also, the advantages of each I^{2}C operation will be discussed and compared. Source code for common data transfer modes is also provided.

Figure 1 describes the hardware schematic for the interface between Microchip's 24LCXXB devices and the PIC18F1220 PICmicro® microcontroller. The schematic shows the connections necessary between the microcontroller and the serial EEPROM as tested, and the software was written assuming these connections. The SDA pin is an open-drain terminal, and therefore requires a pull-up resistor to Vcc (typically 10 kΩ for 100 kHz and 2 kΩ for 400 kHz). Also, the WP pin is tied to ground because the write-protect feature is not used in the examples provided.

FIGURE 1: CIRCUIT FOR PIC18F1220 AND 24LCXXB DEVICE

*SDA requires a pull-up resistor (10 kΩ for 100 kHz, 2 kΩ for 400 kHz)
 Note that pins A0, A1 and A2 are not internally connected in the 24LCXXB devices.

FIRMWARE DESCRIPTION

The purpose of the firmware is to show how to generate specific I^2C transactions with software using a PICmicro microcontroller. In doing so, the specific details of the I^2C protocol are also shown and discussed, thus providing the building blocks for writing more complex programs later on. The firmware is organized into five separate programs:

• Byte write with a 5 ms calculated delay for writes
• Byte write with a polling routine for writes
• Page write
• Byte read
• Sequential read

These programs are also used to compare each of the different operations, as to provide a better understanding of when a particular operation would be more useful.

The code was tested using the 24LC16B serial EEPROM. This device features 8 blocks of 256 bytes and 16-byte pages. All operations occur in the first block of memory on the EEPROM device. The oscilloscope screen shots are labeled for ease in reading. The data sheet versions of the waveforms are shown below the oscilloscope screen shots. All timings are designed to meet the 100 kHz specs, and a 4 MHz crystal oscillator is used to clock the PIC18F1220. If a faster clock is used, the code must be modified to ensure the timing specs are met. All values represented in this application note are hex values unless otherwise noted.

Although this application note focuses on the 24LCXXB devices, the firmware is also compatible with the 24AAXX devices. Furthermore, the concepts discussed here apply to all 24XXX series devices. As such, with a small amount of modification, the firmware can be used with any 24XXX device.

BYTE WRITE WITH DELAY

The byte write operation has been broken down into the following components: the Start condition and control byte, the address byte, and the data byte and Stop condition.

All I^2C commands must begin with a Start condition. This consists of a high-to-low transition of the SDA line while the clock (SCL) is high. After the Start condition, the 8 bits of the control byte are clocked out to the EEPROM, with data being latched in on the rising edge of SCL. The device code (0xA for the 24LCXXB devices), the block address (3 bits) and the R/W̄ bit make up the control byte. Next, the EEPROM device must respond with an ACK bit by pulling the SDA line low for the ninth clock cycle.

Start Bit and Control Byte Transmission

Figure 2 shows the details of the Start condition and the control byte. The left marker shows the position of the Start bit, whereas the right marker shows the ACK bit. Note that the SDA line defaults to a high value due to the attached pull-up resistor. Therefore, SDA only goes low when transmitting a '0'.

**FIGURE 2:** START BIT AND CONTROL BYTE

AN979
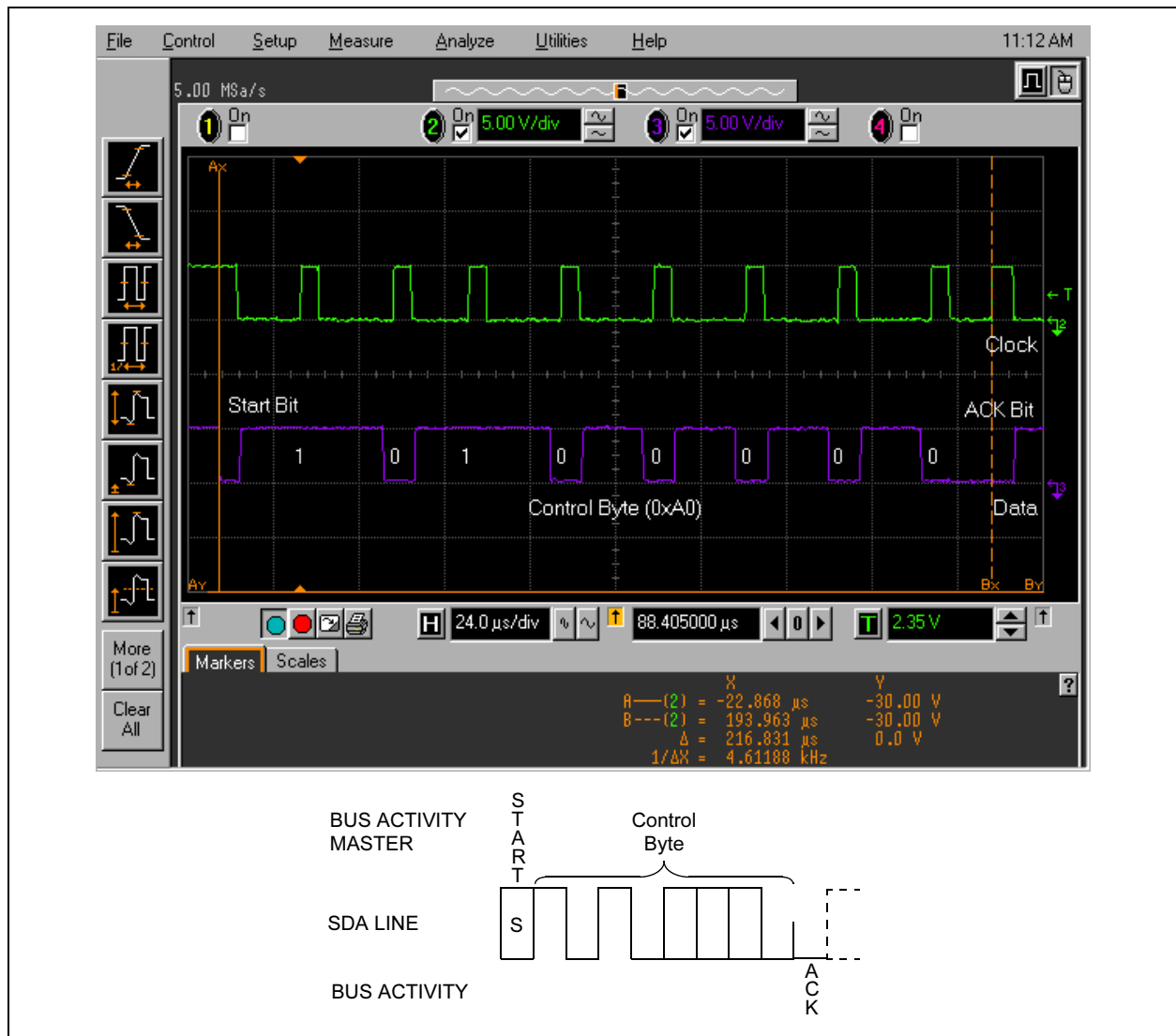
Sending the Word Address

After the EEPROM device has acknowledged receipt of the control byte, the master (PIC18F1220) begins to transmit the word address. This 8-bit value makes up the Least Significant bits of the Address Pointer on the 24LCXXB (the 3-bit block address makes up the Most Significant bits). After the word address has been transmitted, the device must respond with another ACK bit.

Figure 3 shows the address byte and corresponding ACK bit. For reference, the previous ACK bit (in response to the control byte) is shown by the left marker. Note that the word address chosen for this application note is 0x50, and so when combined with the block address, the 11-bit internal Address Pointer becomes 0x050.

FIGURE 3:   ADDRESS BYTE

© 2005 Microchip Technology Inc.
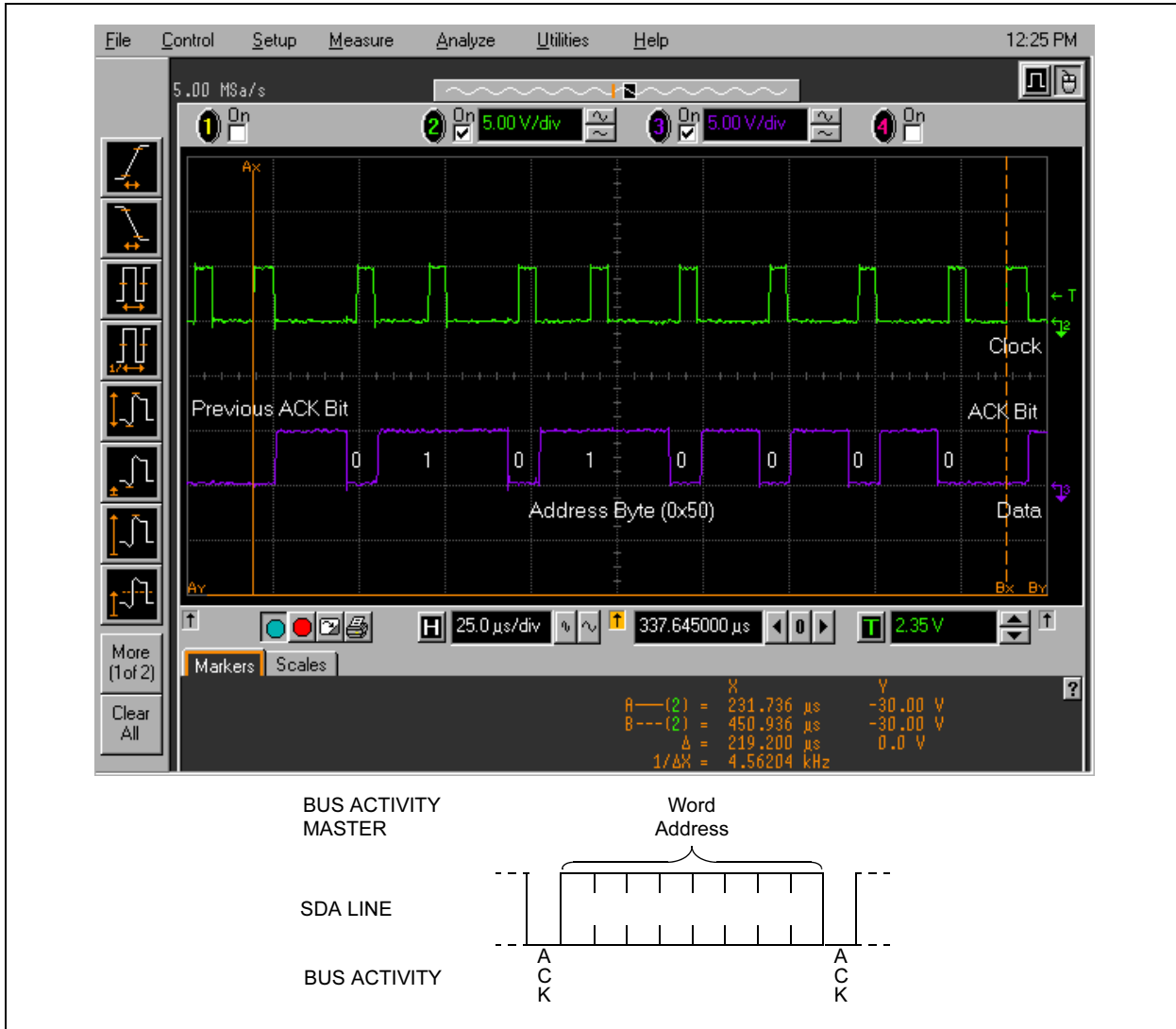
Data Byte and Stop Bit Transmission

Once the word address has been transmitted and the ACK bit has been received, the data byte can be sent. Once again, the EEPROM device must respond with another ACK bit. After this has been received, the master generates a Stop condition. This is achieved by creating a low-to-high transition of SDA while the clock (SCL) is high.

Figure 4 shows the transmission of the data byte, as well as the Stop condition indicating the end of the operation. Again, the left marker shows the previous ACK bit (that of the word address). The right marker denotes the Stop condition.

FIGURE 4: **DATA BYTE AND STOP BIT**
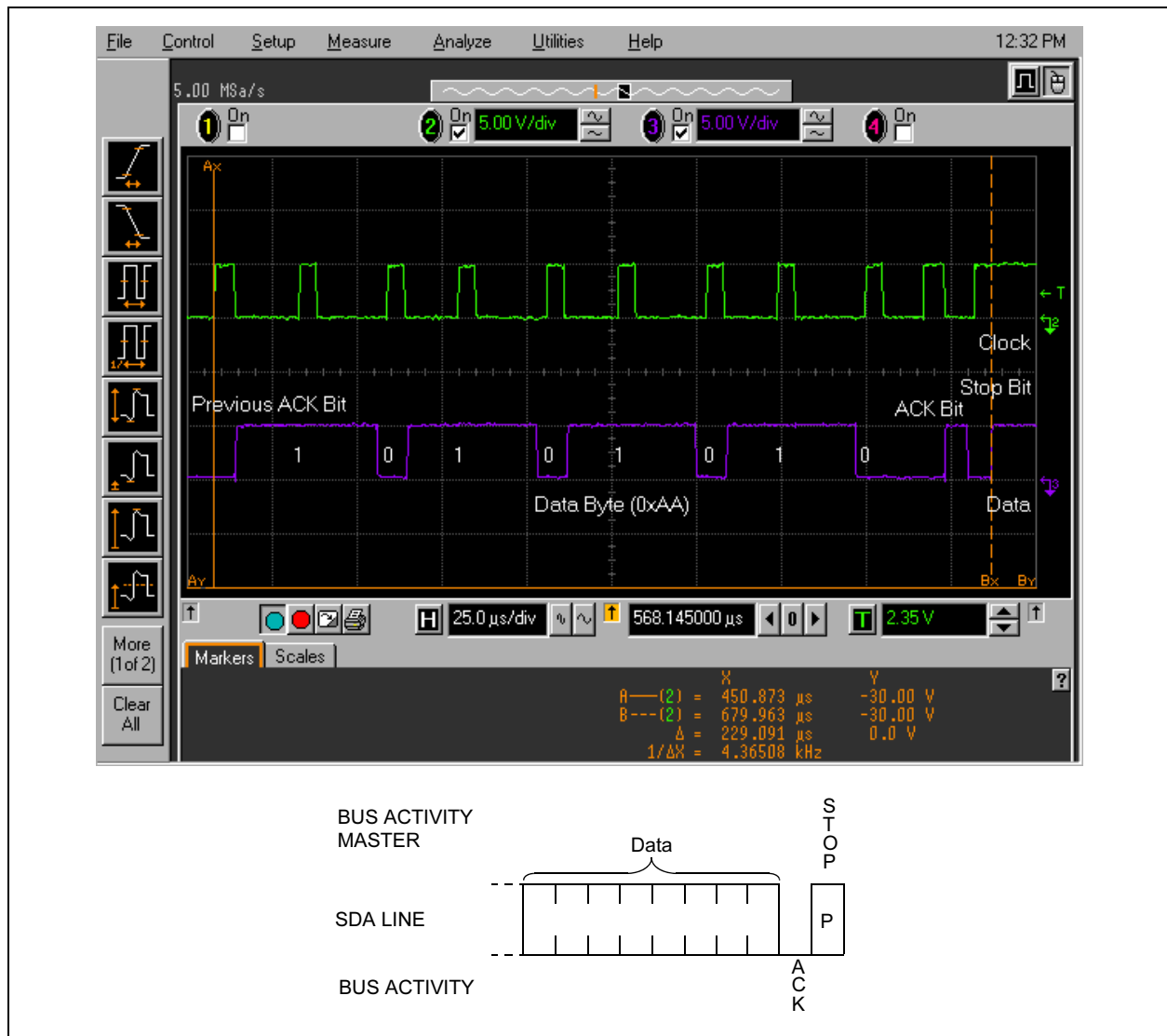
Byte Write Transfer Time, Delay Version

Figure 5 shows the entire byte write operation, including the calculated 5 ms delay. Note that the entire operation took 5.690 ms in this example. So in order to transfer 16 bytes (1 full page on the 24LC16B), it would take approximately 91 ms, using byte writes.

FIGURE 5:   ENTIRE BYTE WRITE TRANSFER (1 BYTE, INCLUDES DELAY)
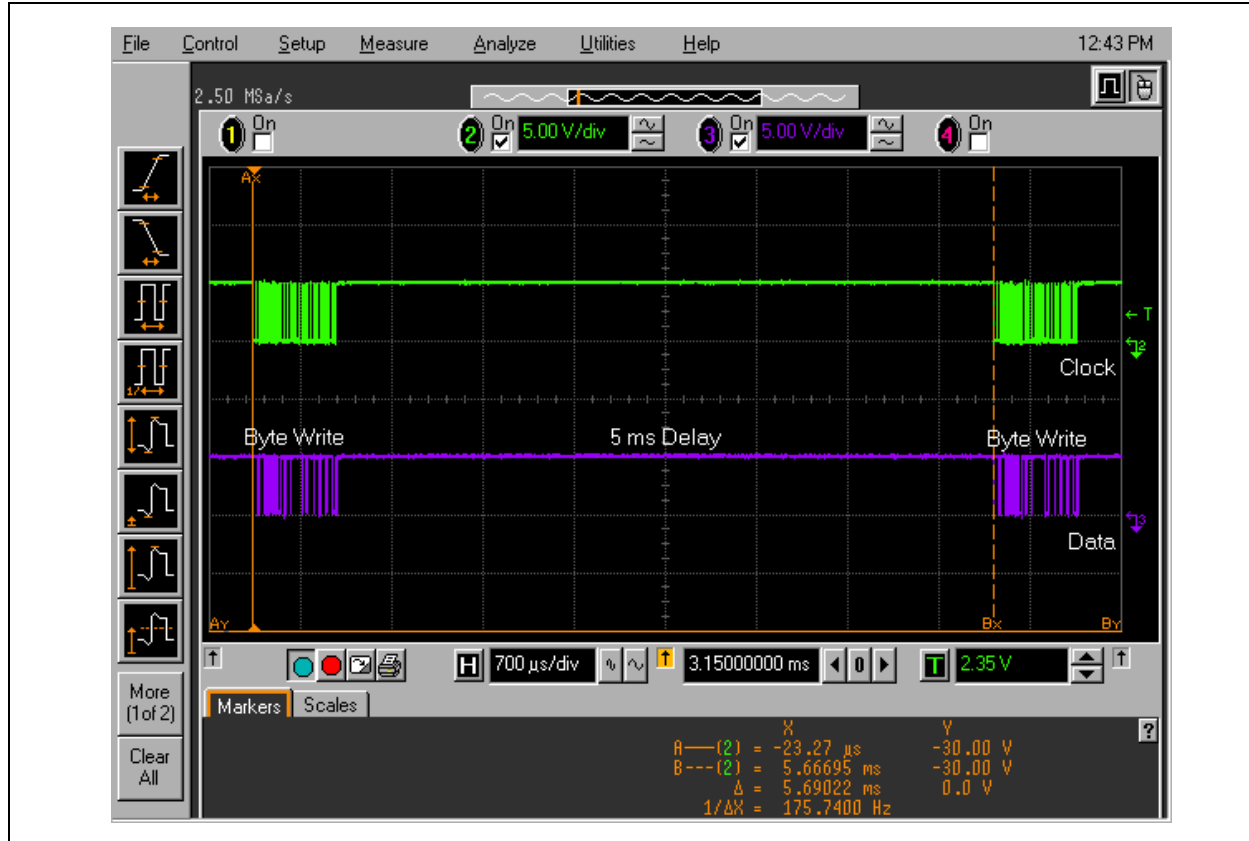
BYTE WRITE WITH POLLING

The byte write operations in the previous section were somewhat inefficient when writing large blocks of data all at once. One reason for this is that the delay used at the end of each operation was a calculated one based on the maximum characteristics of the 24LCXXB devices. But it is not uncommon for these devices to finish their write cycle before the maximum specified time. As such, using the previously shown delay method results in a period of time in which the EEPROM has finished writing, but the master is still waiting.

In order to eliminate this extra period of time, and therefore operate much more efficiently, it is highly recommended to use the Acknowledge Polling feature. Since the 24LCXXB devices will not acknowledge during a write cycle, the device can continuously be polled until an ACK bit is received. This is done after the Stop condition takes place to initiate the internal write cycle of the device.

Acknowledge Polling

The process of acknowledge polling entails sending a Start condition and then a Write command to the EEPROM device, then simply checking to see if the ACK bit was received. If it was not, then the device is still performing its write cycle.

Figure 6 shows an example of acknowledge polling to check if a write operation has finished. In this example, the device did not acknowledge the poll (the ACK bit is high), which indicates that it has not yet completed the write cycle.

FIGURE 6: **ACKNOWLEDGE POLLING ROUTINE (SHOWING NO ACK BIT)**

AN979
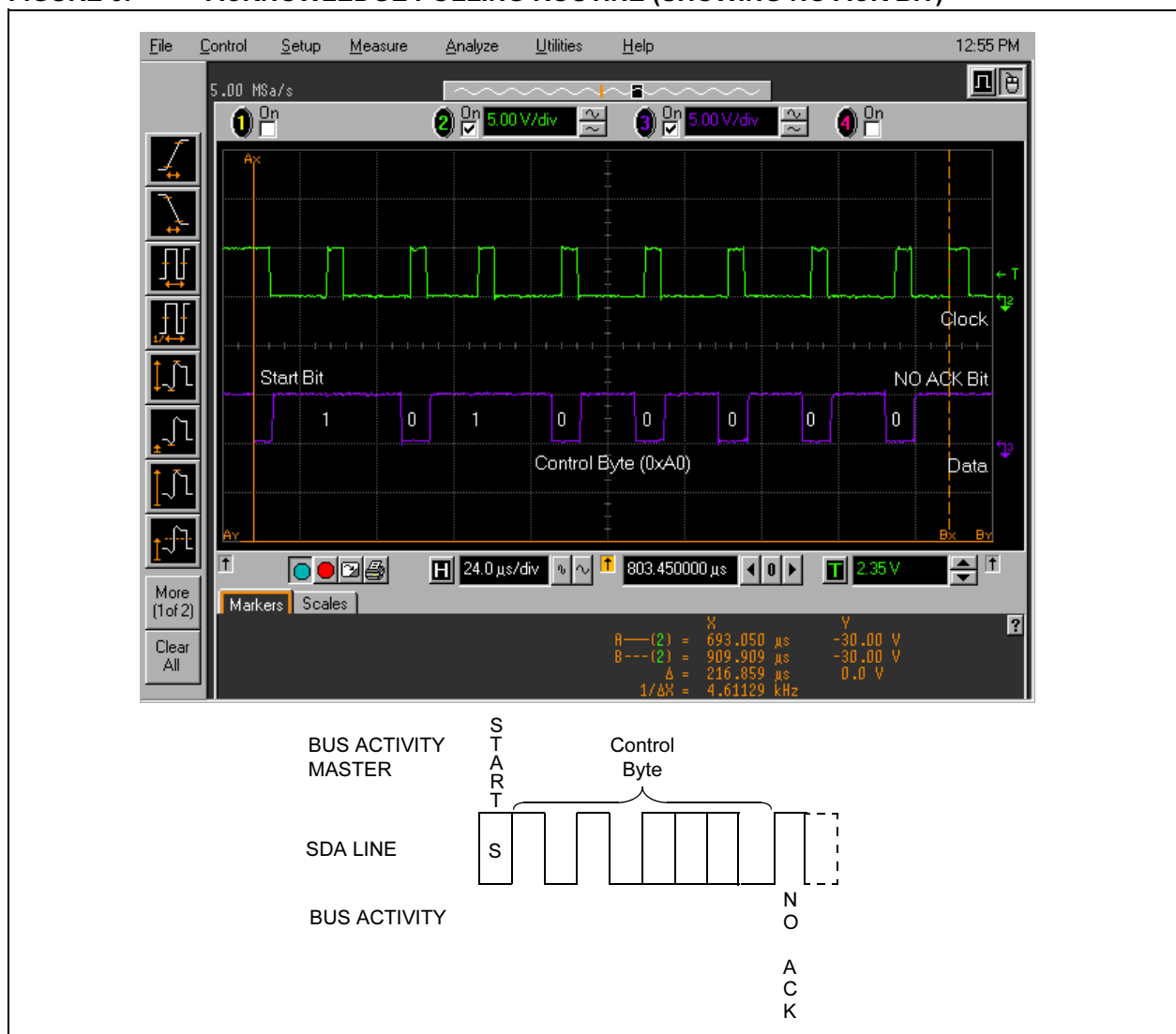
Response to Acknowledge Polling

Figure 7 shows the final acknowledge poll after a write operation, in which the device responds with an ACK bit, indicating that the write cycle has completed and the device is ready to continue.

**FIGURE 7:** **ACKNOWLEDGE POLLING FINISHED (SHOWING ACK BIT)**

© 2005 Microchip Technology Inc.
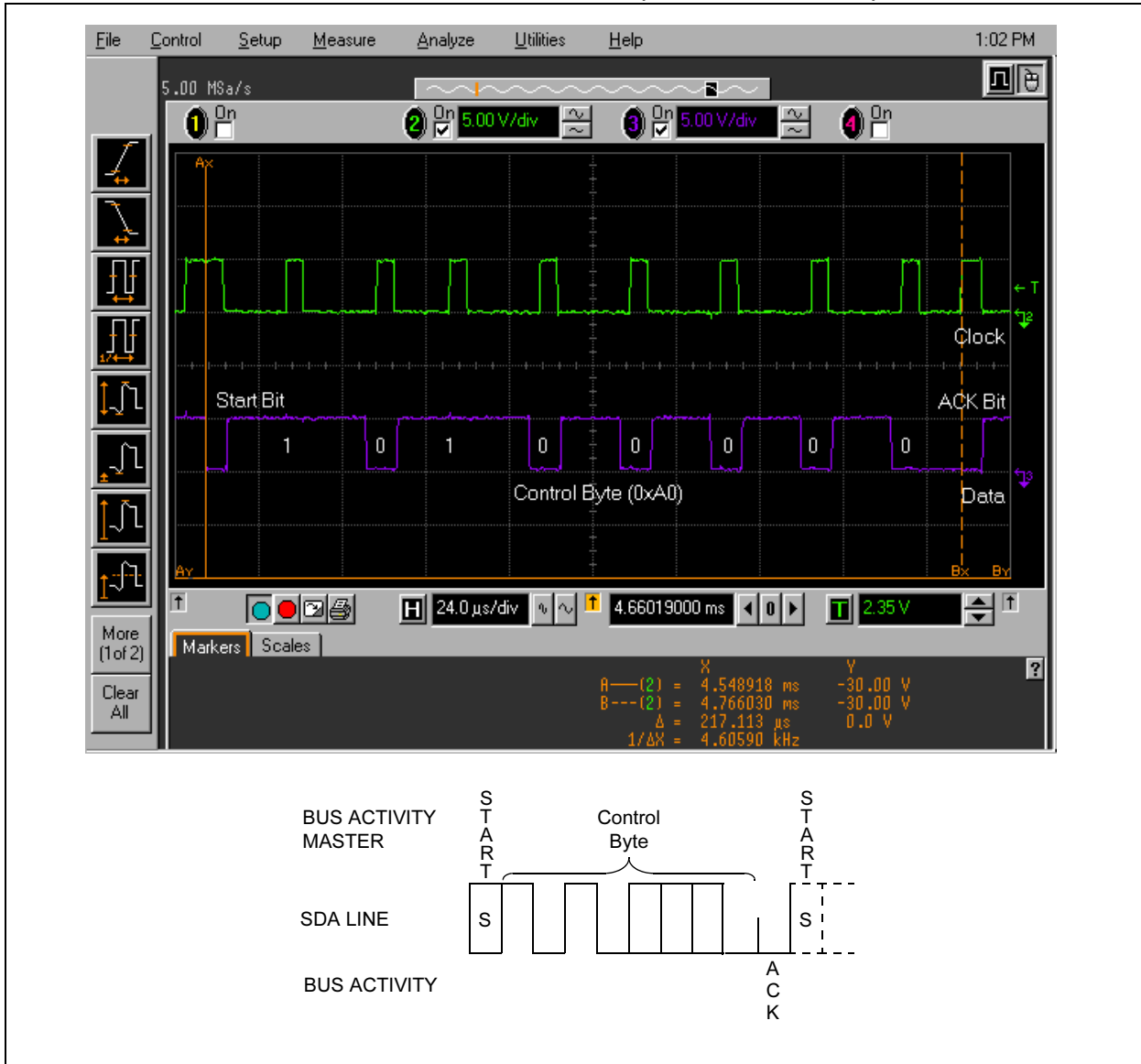
Byte Write Transfer Time, Polling Version

Figure 8 shows an entire byte write operation, including the acknowledge polling at the end. Note that this took 4.751 ms from start to finish in this example. Therefore, to write 16 bytes of data (1 full page on the 24LC16B), it would require roughly 76 ms. Compare this to the 91 ms necessary for 16 bytes using the calculated 5 ms delay and it becomes clear how much time can be saved using this simple feature.

FIGURE 8:   ENTIRE BYTE WRITE TRANSFER (1 BYTE, INCLUDES POLLING)
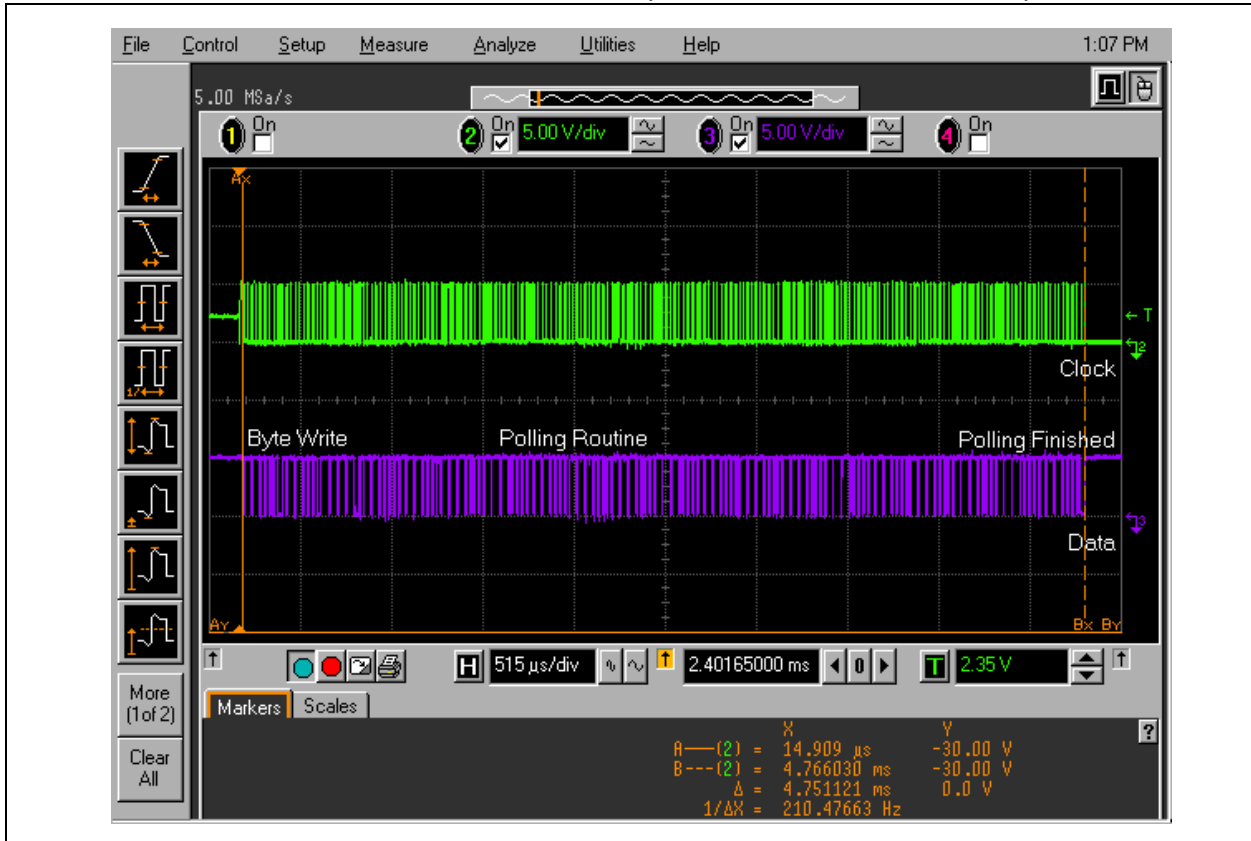
PAGE WRITE

Another method for increasing throughput when writing large blocks of data is to use page write operations. Most of the 24LCXXB serial EEPROMs feature 16-byte pages, except for the 24LC01B/02B devices, which feature 8-byte pages. Using the page write feature, up to 1 full page of data can be written consecutively with the control and word address bytes being transmitted only once. It is very important to point out, however, that page write operations are limited to writing bytes within a single physical page, regardless of the number of bytes actually being written. Physical page boundaries start at addresses which are integer multiples of the page size, and end at addresses which are [integer multiples of the page size] minus 1. Any attempts to write across a page boundary will result in the data being wrapped back to the beginning of the current page, thus overwriting any previously stored data there.

The page write operation is very similar to the byte write operation. However, instead of generating a Stop condition after the first data byte has been transmitted, the master continues to send more data bytes, up to 1 page total. The 24LCXXB will automatically increment the internal Address Pointer with receipt of each byte. As with the byte write operation, the internal write cycle is initiated by the Stop condition.

Sending Multiple Bytes Successively

Figure 9 shows two consecutive data bytes during a page write operation. The entire transfer cannot be shown legibly due to length, but this screen shot shows the main difference between a page write and a byte write. Notice that after the device acknowledges the first data byte (0x10 in this example), the master immediately begins transmitting the second data byte (0x0F in this example).

FIGURE 9:   PAGE WRITE (TWO CONSECUTIVE DATA BYTES)
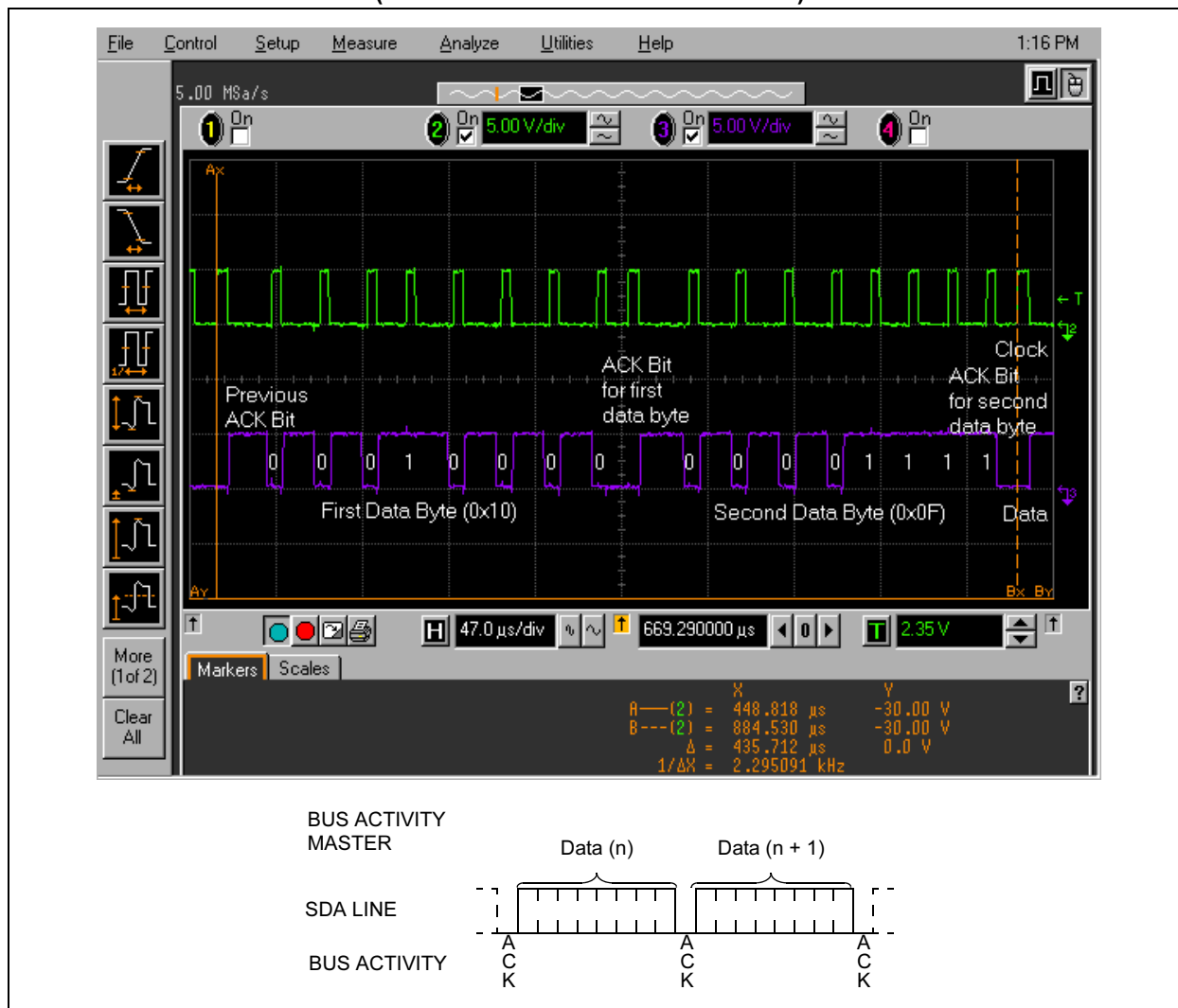
16-Byte Page Write Transfer Time

Figure 10 shows an entire page write operation, including the acknowledge polling at the end. Note that this consumed 8.101 ms total in this example for writing a 16-byte page. Compared to the 76 ms required for transferring the same amount of data using byte writes, the page write operation is obviously the most efficient method of writing large blocks of data. The only disadvantage is that page write operations cannot cross physical page boundaries. But with careful design, the gains in bus throughput should more than make up for it.

FIGURE 10:   PAGE WRITE TRANSFER (16 BYTES, INCLUDES POLLING)
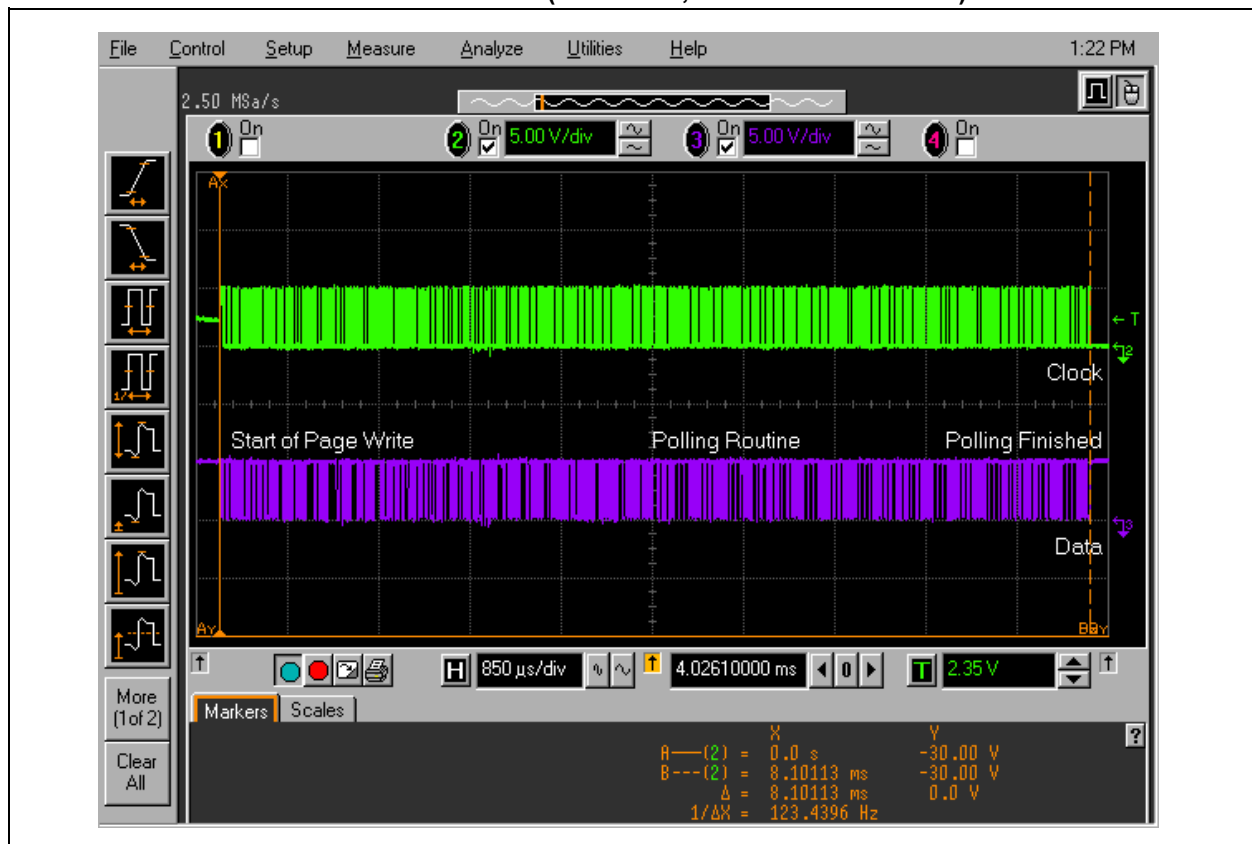
BYTE READ

In order to read data from the 24LCXXB device in a random access manner, the byte read operation can be used. It is similar to the byte write operation, but more complex. The word address must still be transmitted, and to do this, a control byte with the R/$\overline{W}$ bit set low must be sent first. This conflicts with the desired operation, that is, to read data. Therefore, after the word address has been sent, a new Start condition and a control byte with R/$\overline{W}$ set high must be transmitted. Note that a Stop condition is not generated after sending the word address.

After the data byte has been read back from the 24LCXXB device, the master must respond with a NO ACK bit, that is, leaving the SDA line high in place of an ACK bit. This indicates to the device that no more data will be read. Finally, the master generates a Stop condition to end the operation.

Writing Word Address for Read

Figure 11 shows an example of the first control byte and word address of a byte read operation. The left marker indicates the Start bit, and the right marker indicates the ACK bit after receipt of the word address (0x50 in this example). Once again, the R/$\overline{W}$ bit must be low in order to transmit the word address.

**FIGURE 11:   BYTE READ (CONTROL BYTE AND ADDRESS)**
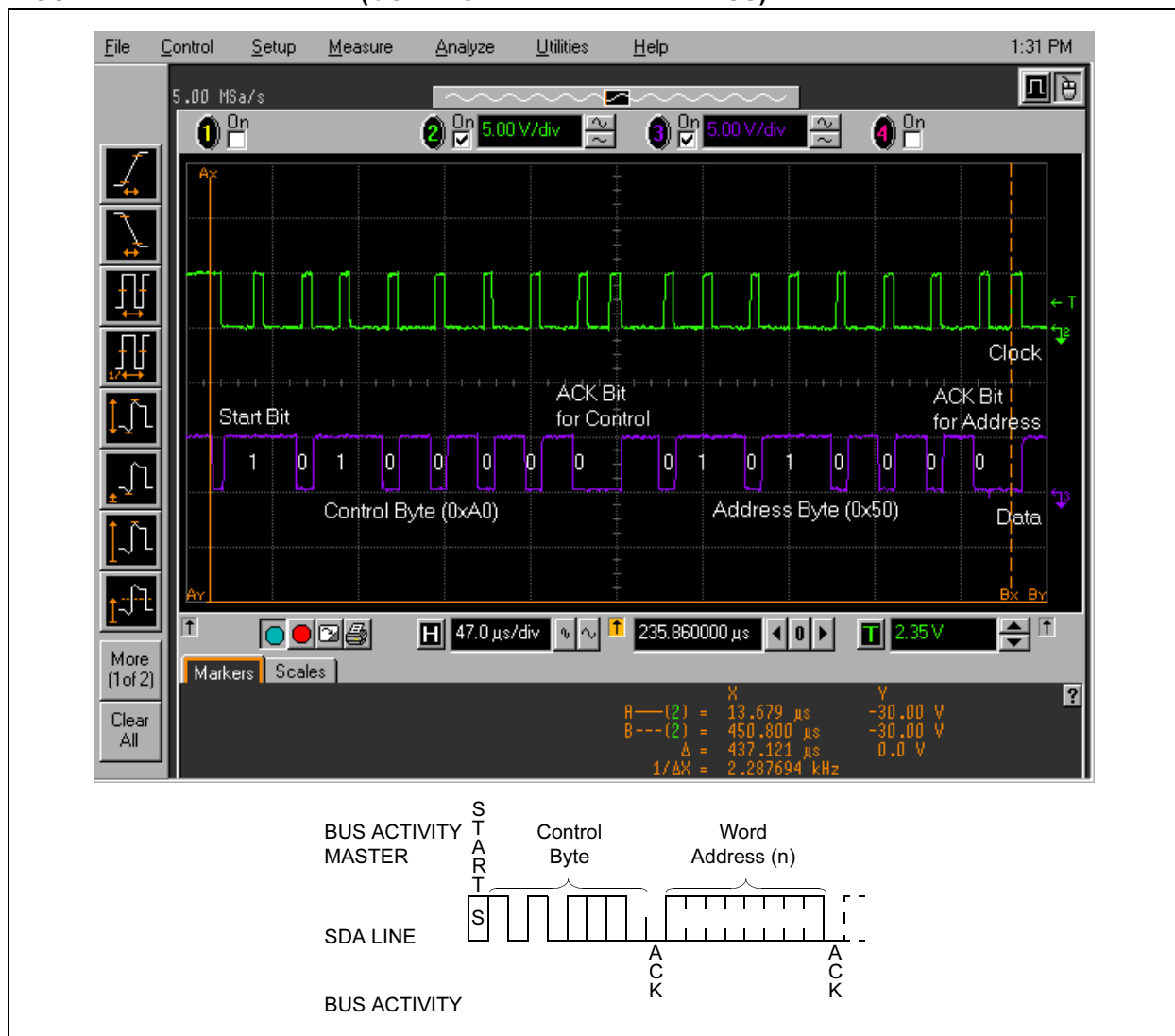
Reading Data Byte Back

Figure 12 shows the control byte and data byte during the actual read part of the operation. A new Start condition is generated immediately after receipt of the previous ACK bit, and is marked with the left marker. At the end of the transfer, the master indicates that no more data will be read by the use of a NO ACK bit (holding SDA high in place of an ACK bit); this is shown by the right marker. After the NO ACK bit has been sent, the master generates a Stop condition to end the operation.

Overall, the byte read operation takes approximately 835 µs for 1 byte. And so, 16 bytes would take ~13.4 ms.

FIGURE 12:   BYTE READ (CONTROL BYTE AND DATA)
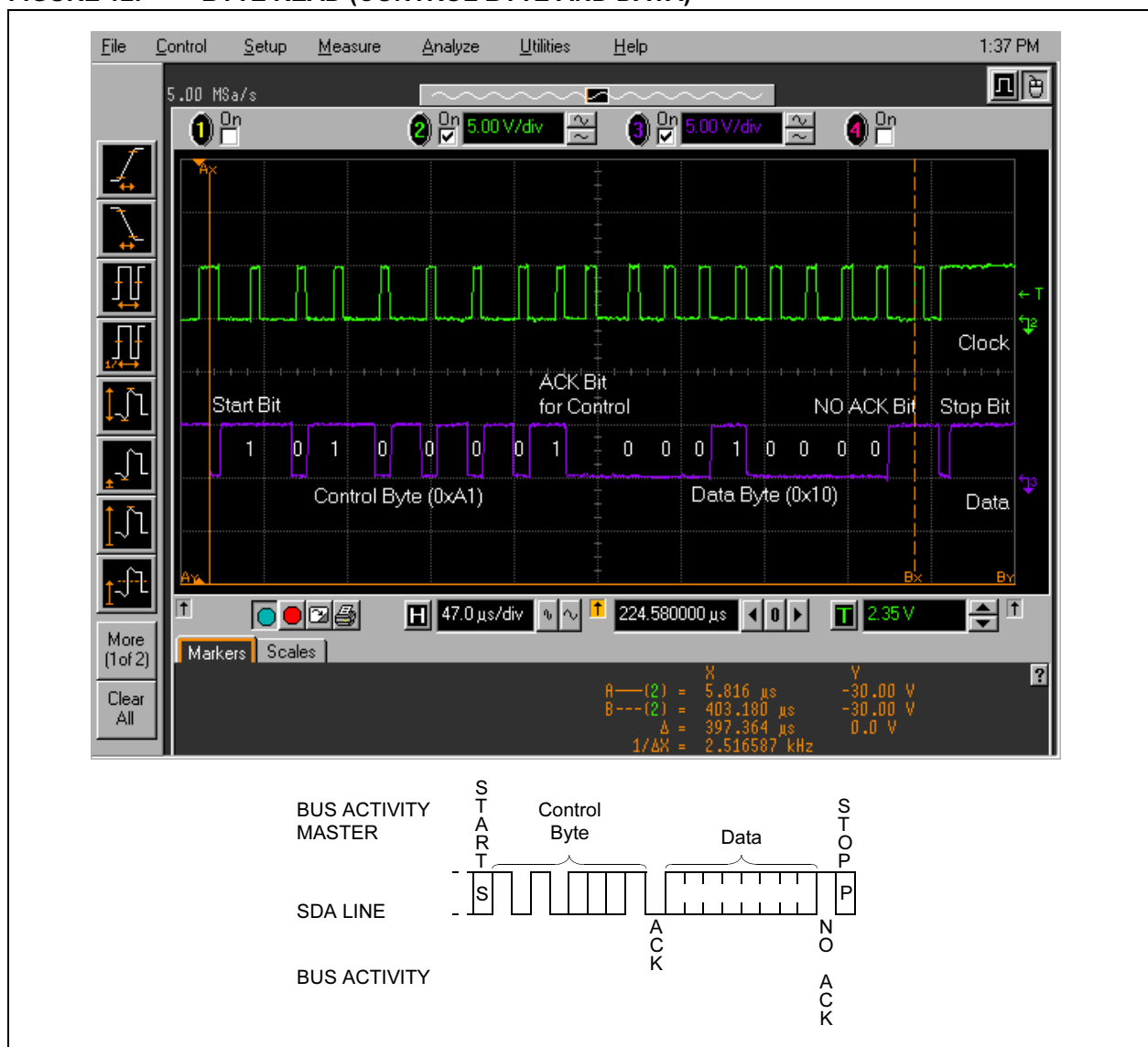
SEQUENTIAL READ

Just as the page write operation exists to allow for more efficient write operations, the sequential read operation exists to allow for more efficient read operations. While the page write is limited to writing within a single physical page, the sequential read operation can read the entire contents of memory in a single operation.

The sequential read operation is very similar to the byte read operation, except that the master must pull SDA low after receipt of each data byte to send an ACK bit back to the 24LCXXB device. This ACK bit indicates that more data is to be read. As long as this ACK bit is transmitted, the master can continue to read back data without the need for generating Start/Stop conditions or for sending more control/address bytes.

Reading Data Bytes Successively

Figure 13 shows the last two bytes of a 16-byte sequential read operation. Note that the master pulls SDA low to transmit an ACK bit after the first data byte, but leaves SDA high to transmit a NO ACK bit after the final data byte. As with all other operations, a Stop condition is generated to end the operation.

FIGURE 13: SEQUENTIAL READ (LAST TWO DATA BYTES)
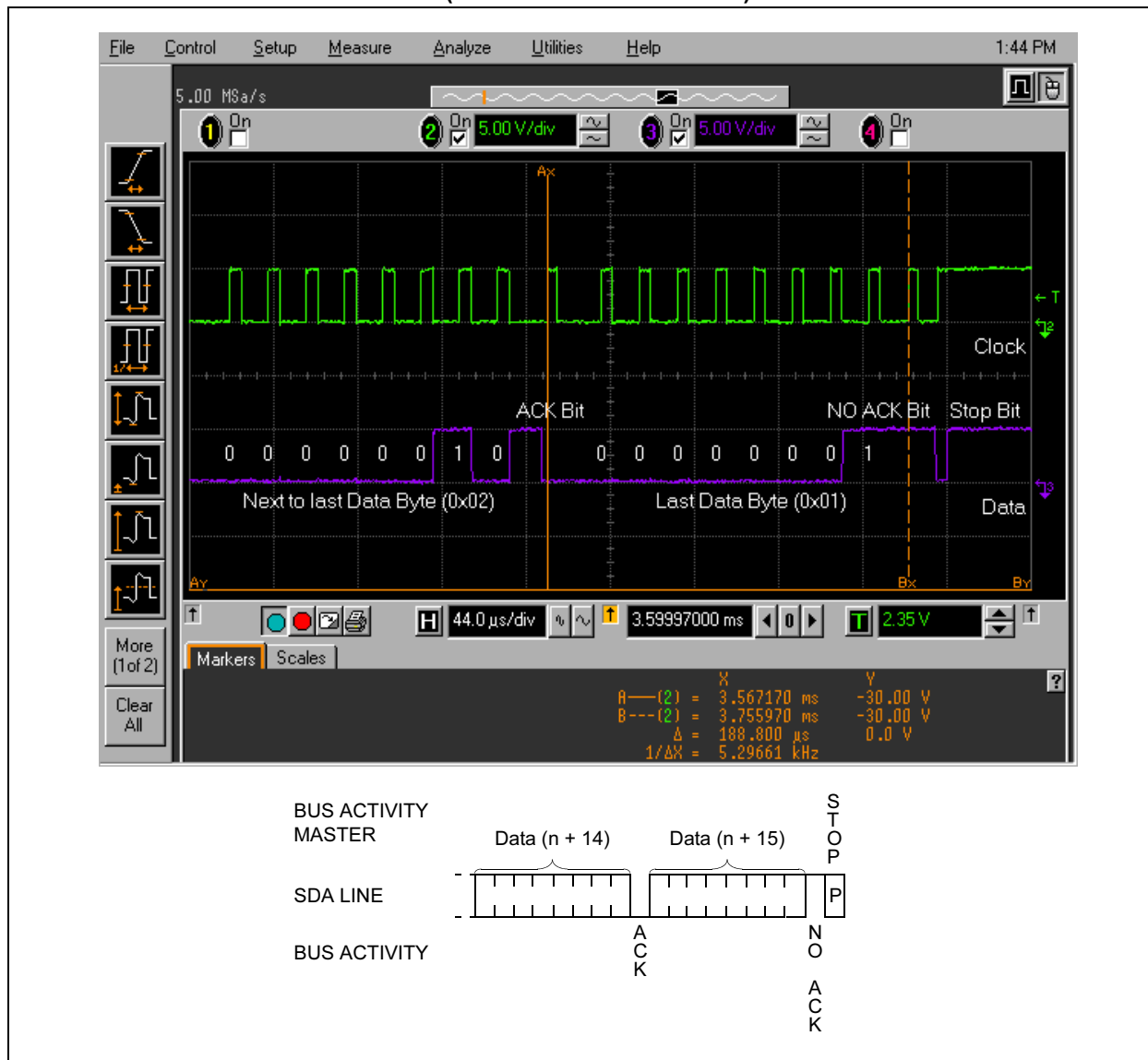
16-Byte Sequential Read Transfer Time

Figure 14 shows a 16-byte sequential read operation, which took 3.799 ms for this example. This is much less than the ~13.4 ms consumed by reading 16 bytes with the byte read operation. Also, since sequential reads are not restricted by page boundaries, they become more efficient than byte reads when reading anything over a single byte of data.

FIGURE 14:   16-BYTE SEQUENTIAL READ TRANSFER
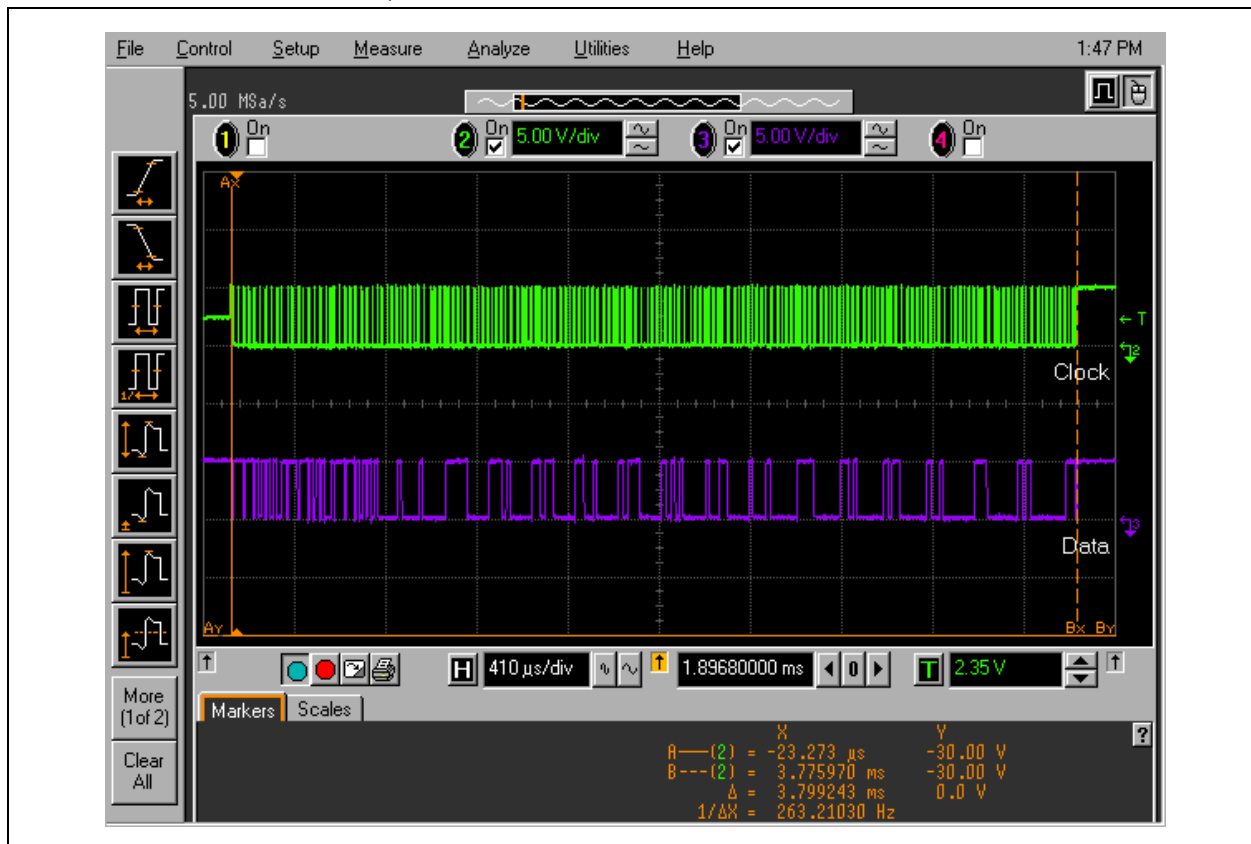
CONCLUSION

All of the operations used to communicate with Microchip's 24LCXXB serial EEPROM devices have certain characteristics which make them more efficient in specific situations. Byte operations can be more efficient when only small amounts of data need to be transmitted, whereas page writes and sequential reads are more efficient for larger amounts of data. Acknowledge polling allows for better throughput by shortening the amount of time spent waiting on the write cycle to finish, but it requires the bus to remain busy while polling. This is not an issue when a single I^2C peripheral resides on the bus, but could be a disadvantage when communication is desired between many different peripherals.

This application note illustrates the main characteristics of I^2C communications with Microchip's 24XXX series serial EEPROM devices, focusing specifically on the 24LCXXB devices. The assembly code provided is highly portable and can be used on many PICmicro microcontrollers with only minor modifications. The code was tested on Microchip's PICDEM™ 2 Plus 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.

QUALITY MANAGEMENT SYSTEM

CERTIFIED BY DNV

ISO/TS 16949:2002

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
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westford, MA
Tel: 978-692-3848
Fax: 978-692-3821

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

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

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

ASIA/PACIFIC

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: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

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

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 - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

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

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

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

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

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

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

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

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

Germany - Ismaning
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

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/20/04