INTRODUCTION

There are many different microcontrollers on the market today that are being used in embedded control applications. Many of these embedded control systems need nonvolatile memory. Because of their small footprint, byte level flexibility, low I/O pin requirement, low power consumption and low cost, serial EEPROMs are a popular choice for nonvolatile storage.

Microchip Technology has addressed these needs by offering a full line of serial EEPROMs covering industry standard serial communication protocols for two-wire (I2C™), three-wire (Microwire), and SPI™ communication. Serial EEPROM devices are available in a variety of densities, operational voltage ranges, and packaging options.

FIGURE 1: CIRCUIT FOR PIC16F54 AND 93 SERIES (MICROWIRE) DEVICE

Note 1: CS should always have a pull-down resistor to protect against data corruption during power-up or power-down of the microcontroller.

2: PE pin available only on 93XX76X and 93XX86X devices. Pull-up resistor suggested ~10K ohm.

3: ORG pin available only on 93XXXXC devices.
FIRMWARE DESCRIPTION

The purpose of the program is to show individual features of the Microwire protocol and give code samples of the Start bit, opcodes and addressing schemes so that the basic building blocks of a program can be shown. The firmware performs five basic operations:

• Erase/Write Enable command
• Write command for one word of data
• Ready/Busy polling
• Read command for one word of data
• Erase/Write Disable command

The code was tested using the 93LC66B serial EEPROM. This device features 256 x 16 (4 Kbit) of memory and 16-bit organization. A 10 MHz crystal oscillator is used to clock the PIC16F54. If a faster clock is used, the code must be modified to ensure all timing specs are met. The waveforms provided are shown from CS active to CS disable so an entire instruction can be seen. To ease the interpretation of the serial data, the data sheet waveforms are provided below the oscilloscope screen shots. All values represented in this application note are decimal values unless otherwise noted.

Device Setup

Although this application note focuses on the 93LC66B device, the firmware supports all 93XXXX devices. This is done by setting the following two constants at the beginning of the source code:

• NUMBITS
• ORGVAL

The NUMBITS constant is used in the output subroutines to determine how many bits are required to be output. More specifically, it is used by the EWEN and EWDS commands to calculate the number of dummy bits required. It is also used by the Read and Write subroutines to skip over the unused bits in the address word. This constant must equal the required number of clock cycles for an EWEN command (11 for the 93XX66B).

Note: On devices of the same density, the required number of clock cycles differs between 8-bit and 16-bit organizations. Therefore, if the organization is changed, NUMBITS must also be updated appropriately.

The ORGVAL constant specifies the data organization. This value must be set to either 8 or 16, depending upon which device organization is being used. The 93XXXXA devices use 8-bit organization, whereas the 93XXXXB devices use 16-bit organization. Furthermore, the 93XXXXC devices allow a selectable word size (either 8 or 16-bit) through the use of the ORG pin.
WRITE ENABLE

Figure 2 shows an example of the Erase/Write Enable (EWEN) command. Chip Select is brought high (active), and the Start bit and four-bit opcode ('0011') are sent out first, with the required number of dummy bits (6 for the 93XX66B) following. The EWEN command must be given before a write is attempted. The device will be enabled for writes until an Erase/Write Disable command is given or the device is powered down.

FIGURE 2: ERASE/WRITE ENABLE (EWEN)
WRITE COMMAND (START BIT, OPCODE, ADDRESS AND DATA)

Figure 3 shows an example of the Write command. The device is selected and the Start bit, opcode and the word address are sent out. Next, the data is clocked out to the device. When the Chip Select is toggled, the internal write cycle is initiated.

Once the internal write cycle has begun, the Ready/Busy signal can be polled on the DO pin to check when the write finishes. A 6 ms delay needs to be added if the Ready/Busy status is not being polled. This code uses Ready/Busy polling.

FIGURE 3: WRITE COMMAND, ADDRESS AND DATA
READY/BUSY POLLING

After a valid Write command is given, the DO line of the 93XXXX can be monitored to check if the internal write cycle has been successfully initiated and, if so, to determine when the write cycle is complete. The oscilloscope plot below shows that the device is selected and the DO line is low for approximately 3.2 ms before the device brings the DO line high, indicating that the write cycle is complete.

Note that the 93AAXX and 93LCXX devices have a maximum program cycle time (Twc) of 6 ms, but in this example, the write cycle only lasted 3.2 ms. This illustrates that the write cycle typically is much shorter than the specified maximum. Therefore, it can be highly beneficial to take advantage of the Ready/Busy polling feature, so as to increase efficiency when writing multiple words of data to the device.
READ COMMAND (START BIT, OPCODE, ADDRESS AND DATA)

Figure 5 shows an example of the Read command. The device is selected and the Start bit, opcode and the word address are sent out. At this point, the device gets ready to transmit data. The microcontroller must generate the clock signals, and read DO on each falling clock edge. In this example, the data being read is 0x55AA.

FIGURE 5: READ COMMAND
ERASE/WRITE DISABLE COMMAND

Once the internal write cycle is complete, the Write Disable (EWDS) command should be given (see Figure 6). This command consists of a Start bit and the four-bit opcode ('0000'), followed by the appropriate number of dummy bits (6 for the 93XX66B). The EWDS command should always be sent to the device after completing a write or prior to powering down the device/system.

FIGURE 6: ERASE/WRITE DISABLE COMMAND

![Diagram showing the ERASE/WRITE DISABLE COMMAND process. The diagram includes waveforms for CS, CLK, and DI, with annotations indicating Start Bit, Dummy bits, and EWDS instruction.](image-url)
CONCLUSION

These are some of the basic features of Microwire communications on one of Microchip's PIC16 devices without the use of a hardware serial port. The code is highly portable and can be used on many PICmicro® microcontrollers with very minor modifications. Using the code provided, designers can begin to build their own Microwire libraries to be as simple or as complex as needed.

The code was tested on Microchip's PICDEM™ 2 Plus Demonstration Board with the connections shown in Figure 1.
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.