Hi, my name is Barry Blixt, marketing manager for Microchip memory products. Welcome to this 20-minute web seminar in which we will discuss some recommended usage practices for SPI serial EEPROMs.

Here are some questions that I'll answer during this presentation:
First, are you looking for general information on the SPI protocol? We will discuss the advantages of the SPI bus.
• Are you working on an SPI system and looking for design tips? This web seminar offers suggestions that build upon the information available in our data sheets.
• Are you looking for ideas about the best way to design a board? We have several hardware recommendations.
SPI EEPROMs: Recommended Usage

- SPI bus advantages
- Beyond the data sheet
- Hardware recommendations
- Write Protect and Status Register

• The SPI bus has several software and hardware write protect options, some of which are controlled by the device’s status register. Do you need specifics about how to implement these often confusing options?
Most of the information in this seminar is taken from Microchip data sheets as well as our application note AN 1040 entitled “Recommended Usage of Microchip SPI Serial EEPROM devices.”

Now, let's look at our agenda.
We’ll begin this seminar with a summary of the major features of the SPI bus.

Then we’ll look at some hardware recommendations and details.

Following that, we will focus on understanding the status register, including its write protect options.

We will finish up with a quick summary and some places to find more information.

In total, we will cover 7 recommendations that represent some of the most often asked questions about SPI EEPROM devices.
Let’s look at some advantages of the SPI protocol, which is generally known for its robustness as well as its speed. In this slide, you can see a typical SPI EEPROM pinout. Pin 1 is chip select. Pin 2 is data out. Pin 3 is write protect. Pin 4 is ground. Pin 5 is data in. Pin 6 is the clock. Pin 7 is hold, and Pin 8 is voltage. Note that the Chip Select, Write Protect and HOLD pins are all active low.
SPI communication is controlled via hardware by the chip select pin. That means that the bus has very good noise resistance since spurious writes can be virtually eliminated by properly controlling the Chip Select pin and by properly using the write protect features.
Most SPI parts have a Write Protect pin that, along with the status register, is used to implement several write protect options. These options can protect ¼, ½, or all of the device array, as well as the status register.
The bus has a wide density range of 1 Kbit all the way up to 1 Mbit.
The SPI protocol is the fastest of the three EEPROM buses with most SPI devices having a maximum speed of 10 MHz. In comparison, Microwire devices have a maximum speed of 3 MHz, and I²CTM devices top out at 1 MHz.
Finally, SPI is a 4-wire bus, with Chip Select, Clock, Data Out and Data In all connected to the master with individual signal lines. This requires using several microcontroller pins, but it also means that the designer has more control of the bus and can design a very robust system.

On the other hand, the added functionality I just described does add to the die size, so SPI EEPROMs are slightly more expensive than the other 2 EEPROM protocols.
We'll begin the hardware section of this seminar by showing the recommended connections of an SPI EEPROM.

This slide shows a typical device. Four of its pins must be connected to the master: Chip Select, data out, clock and data in. These 4 pins are easily connected to the many microcontrollers that have built-in SPI ports, including many of Microchip's own 8-, 16-, and 32-bit microcontrollers. Alternatively, SPI EEPROMs can be connected to micros without SPI ports by bit banging, that is by using I/O lines programmed in firmware to match the SPI protocol. We have several app notes that describe, with pre-written code, communication via both methods.

This configuration shows the basic required connections. Now we'll begin our list of recommendations.
Our first recommendation is to use a pull-up resistor on chip select. Chip Select is controlled by the master with a low level needed to select the device. When Chip Select is pulled high, the device is deselected and will not accept commands.

But, there is a potential for the Chip Select pin to float during power up and power down operations. A floating pin can go either high or low depending on several factors. If Chip Select is allowed to float low, the device is selected, and random inputs could be interpreted as a write command, which could corrupt data. A pull-up resistor solves this problem by preventing floating during these indeterminate power levels.
Our 2nd recommendation is about the HOLD pin, here shown as pin 7. Hold is used to suspend transmission to the EEPROM in the middle of a sequence so it can be restarted later at the same point.

HOLD is active low, so it must be held high in order to have normal operations. Most applications do not use the Hold feature. But if it is used, hold must be controlled by the master. And, similar to the Chip Select pin, it should have a pull-up resistor so that it can not float during power-up or power-down.
Since it is fairly uncommon that designers use the hold pin, the more common implementation is to tie HOLD high to Vcc as I’ve shown here. Note that a pull-up resistor is still used. This connection scheme prevents the HOLD pin from floating, but it does disable the HOLD function. This configuration also frees up a microcontroller pin.
Our 3rd recommendation is a quick one. Remember good engineering practice by adding a decoupling capacitor of approximately 0.1 uF. It should be as close to the device as possible to help filter high-frequency noise from the power supply.
Write protect is the last input pin that we need to discuss as our 4th recommendation. If using the write protect pin is not required in an application, it can be permanently connected to Vcc. This configuration, while saving a microcontroller pin, does disable the write protect pin. And, a pull-up resistor must be used to prevent Write Protect from floating.
Alternatively, if hardware write protect is required in an application, the pin is controlled by the master. And, as with the other input pins, Write Protect can not be allowed to float. If the pin is being used, we recommend a pull-down resistor on Write Protect.

And this completes our hardware recommendations.
Let’s review those hardware recommendations now.
First, there is a pull-up resistor on the chip select pin.
Second, the hold pin is either connected to the master or, more commonly, HOLD is disabled by being tied high.
Third, a decoupling capacitor is utilized.
Finally, the Write Protect pin is either disabled by being pulled high to Vcc or is connected to the master via a pull-down resistor, as shown here.

Now let's move on to a discussion of the status register.
SPI devices have an 8-bit status register, shown in table form on this slide. The top row of the table shows the 8 bits labeled 0 through 7. The 2nd row shows the name of each bit. And the third row shows the value of each bit, here all 0’s, except for bits 4 through 6 which are “don’t care.” I will introduce each of the bits here, and talk in more detail about them later on.
Bit 7 is the write protect enable, or WPEN bit. It is only present in 8 Kb and larger devices.
Bits 2 and 3 are the block protect bits. They determine which part of the array – ¼, ½, all or none – is protected.

The 3 bits we’ve just discussed are read/write bits that can be changed by writing directly to the status register. The next two bits are read only.
**Status Register Overview**

<table>
<thead>
<tr>
<th>Bit number:</th>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit name:</td>
<td>WPEN BP1 BP0 WEI WIP</td>
</tr>
<tr>
<td>Bit value(^1):</td>
<td>0 x x 0 0 0 0</td>
</tr>
</tbody>
</table>

- **Write Protect Enable bit** (8Kb +)
- **Write-In-Process bit** (Read-only)
  - 0 = no write
  - 1 = write occurring
- **Block Protect Bits**

\(^1x = \text{Don't care}\)

Bit zero is the read-only Write In Process or WIP bit. When no write is occurring, the WIP bit is set to “0.” During a write to either the array or to the status register, it is set to “1.” The WIP bit can be used to determine when a write is complete. Since the status register can be read at any time, even while a write is in progress, a routine can be set up to monitor the bit. If the WIP bit is “1,” a write is currently in progress. When the WIP bit changes to “0,” the write cycle is complete, and the next command can be sent. WIP polling is a simple method to increase throughput by decreasing wait time.
Bit 1 is the Write Enable Latch bit, or WEL bit, which shows the status of the Write Enable Latch. The latch must be set to 1 in order to write to the device. If the WEL bit is 0, writes are prohibited. It can only be modified by the user with a write enable or write disable command. We'll go through more details on the latch's functionality on the next slide.
Here we’ll go through the steps required to set the write enable latch in order to allow writes. We will also look at how the status register changes with each step.
To begin, we’ll look at an un-programmed status register in step 1. The WPEN bit is set to 0 as are the two block protect bits, BP1 and BP0. The WIP bit is also 0 since no write is occurring. The main point here is that the WEL bit is 0, meaning that the write enable latch is cleared and writes are not allowed to the status register or to the array.
Before writing to the part, we first need to set the write enable latch by using a Write Enable, or WREN, command, shown here as step 2. After the command has been completed, the WEL bit is 1, so the latch is now set to allow writes.
In our 3rd step, a write command is sent, which is accepted since the Write enable latch is set. Note that the WIP bit is 1 during the write cycle to allow WIP polling.
Step 4 shows us the status register contents after the write is complete. The WEL bit has changed back to '0' since in SPI devices, the Write Enable Latch is automatically cleared after a successful write. So now writes are not allowed to the array or to the status register until another write enable command has been sent. The latch is automatically cleared after a write command, as shown here, as well as after writes to the status register or after a 'write disable' command.

Compare the operation of this protocol to Microwire EEPROMs, which also have a write enable command. But once the write enable command has been sent in a Microwire device, writes are allowed until a write disable command is sent, leaving the part vulnerable to unwanted writes. The fact that the WEL is cleared after writes makes SPI parts a good choice for noisy environments.
Our 5th recommendation, then, is to only set the Write Enable Latch just before a write command in order to minimize the chance of undesired writes.

Next, we’ll talk about how the status register’s block protect bits can protect a portion of the array.
Once again, here is our status register, and now we’re highlighting block protect bits 1 and 0. The bits can be set in 4 different combinations to protect different parts of the array.
If the bits are 0,0, none of the array is protected. If the bits are changed to “0, 1”, the upper quarter of the array is protected. If the bits are set to “1,0”, the upper half of the array is protected. If the bits are set to “1,1”, the entire array is write protected.

The Block Protect bits are both writable, so they can be changed by writing to the status register. Let’s run through the steps needed to change them.
Step 1 shows the status register after a write enable command has been sent. Remember that this command is required to set the Write Enable Latch to 1 to allow writes. Now we can write to the part.
We want to change the block protect bits to “1,0” to protect the upper half of the array. Writes to the status register are done with a Write Status Register, or WRSR, command shown as step 2. During the write to the status register, the WIP bit is “1”.

<table>
<thead>
<tr>
<th>Block Protect Bits</th>
<th>Status Register</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td>W</td>
<td>P</td>
</tr>
<tr>
<td>1. After WREN</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>2. Initiate WRSR</td>
<td>0</td>
<td>x</td>
</tr>
</tbody>
</table>

BP1, BP0: 0,0 0,1 1,0 1,1

Array Protection: None Upper 1/4 Upper 1/2 All
After the Write Status Register command is complete in step 3, the block protect bits have been changed to “1,0.” And, the WEL bit has been automatically cleared to 0, so writes are prohibited to the status register and to the entire array until the Write Enable Latch is re-set. And, even if the latch is set to 1 to allow writes, the upper half of the array is still protected by the block protect bits.

Before we move on to the hardware write protect options, remember that the protected segments of the array as set by Block Protect bits are always protected. Let’s now move on to the Write Protect pin.
One often confusing aspect of SPI serial EEPROMs is that the write protect pin works differently on 1 through 4 Kbit parts than on 8 Kbit and larger ones.

In 1 – 4 Kb devices, the Write Protect pin – here pin 3 - acts as a normal hardware write protect. Recall that the WP pin is active low; so if the pin is held low, as shown here on the left side of this slide, write protect is enabled, and writes are prohibited to the array and to the status register.
WP Pin: 1-4 Kb
Normal hardware write protect

**WP LOW:** Writes prohibited to array and status register

**WP HIGH:** Writes allowed to:
- status register
- unprotected array segments (as defined by BP bits)

If the Write Protect pin is high, as shown on the right, writes are permitted to the device’s status register. Writes are also permitted to the array itself, except to those segments which have been protected by the status register’s block protect bits. And remember that the Write Enable Latch must be set for writes to occur.
In 8Kb and larger devices, the Write Protect pin only protects the status register. This write protect feature is enabled by the following 2-step process.
First, the Write Protect Enable, or WPEN bit, must be set to 1. Recall that it is bit 7 of the status register. While the WPEN bit is 0, the write protect pin is disabled. Changing the WPEN bit to a 1 with a write status register command enables the write protect pin.

WP Pin: 8 Kb-1 Mb
Protects Status Register Only

1. Set WPEN to 1

WPEN = 1 ⇒ WP pin enabled
Once the Write Protect pin has been enabled by the Write Protect Enable bit, the pin controls writes to the status register only. If the pin is pulled low, as shown here, writes to the status register are prohibited. Note that the pin does not affect writes to the array itself.

To summarize: in the lower density parts, the Write Protect pin acts as a hardware write protect that can prevent writes to the array and the Status Register. For higher density parts, the Write Protect Enable bit and Write Protect pin act together to determine whether the status register is protected. Note that in all cases, protected blocks – as determined by the Block Protect bits - remain write protected. And in all cases, the Write Enable Latch must be set to 1, or no writes at all can occur.
We have a lot more information about SPI EEPROMs available on our web site.

Our data sheets are an excellent source that describe how SPI devices work.

We also have several educational app notes. I have been referencing AN 1040 throughout this seminar. Between it and the product data sheets, you have an excellent baseline to understand EEPROM operations and recommended design practices.

We also have dozens of app notes explaining how to interface a Microchip SPI EEPROM to many of Microchip’s PIC® microcontrollers. In most cases, these app notes also include downloadable source code.

Both the data sheets and app notes can be found at www.microchip.com/memory.

Finally, we have several other web seminars on EEPROMS, including an introductory overview, our memory design kit, EEPROM endurance, and the many small package options that are available. And, our web seminar about I²C™ EEPROM recommended usage also is posted.
And that completes this web seminar in which we’ve discussed 7 recommendations for SPI serial EEPROM designs. Let’s quickly review them now.
Summary

Hardware Recommendations

1. Pull-up resistor on Chip Select
2. Tie HOLD properly
3. Decoupling capacitor
4. Tie Write Protect properly

The first 4 recommendations are all hardware based.

First, make sure the chip select pin has a pull-up resistor and is not left floating.
Next, make sure the hold pin is either disabled by being tied high or is connected to the master. In either case, use a pull-up resistor to prevent floating the pin.
3rd, use a decoupling capacitor.
4th, make sure write protect is not floating.
Summary

● **Hardware Recommendations**
  1. Pull-up resistor on Chip Select
  2. Tie HOLD properly
  3. Decoupling capacitor
  4. Tie Write Protect properly

● **Status Register Recommendations**
  5. WEL operation
  6. WIP polling
  7. S/W and H/W Write Protect options

Here are the Status register recommendations:
Don’t forget that the Write Enable Latch is cleared after writes are completed and must be re-set to 1 before the next write command.

6th, use the status register’s WIP bit to perform WIP polling to improve throughput.

Lastly, take advantage of the hardware and software write protect options that we just discussed. Remember that the block protect bits can be programmed in the status register to protect various segments of the array. And, remember that the write protect pin only protects the status register in 8Kb and larger devices.

To get more details on any of these concepts, please look through our app notes and data sheets.

Thanks a lot for your time.