AT89LP2052/4052 Primer

1. Introduction

The Atmel® AT89LP2052/4052 microcontroller is a low-power, high-performance device featuring an enhanced single-cycle 8051 CPU, 2K/4K bytes of Flash memory, 256 bytes of RAM, configurable I/O, an analog comparator, dual pulse width modulation (PWM) outputs, a UART, a Serial Peripheral Interface (SPI) and a watchdog timer. The Flash program memory may be reprogrammed in-system via the SPI. These features, and others, are described in this application note. Code samples are provided. Additional information on the AT89LP2052/4052 microcontroller can be found in the datasheet. The AT89LP2052/4052 device is pin compatible with the AT89C2051/4051 device. The AT89LP2052/4052 can replace the AT89C2051/4051 in existing applications that require higher performance, lower power, or more data memory.

2. AT89LP Overview

The AT89LP family of Flash microcontrollers is built around an enhanced 8051 CPU core that can fetch a single instruction byte from memory every clock cycle, as compared to classic 8051 devices, which can only fetch a byte every 6 clock cycles. This enhancement allows the AT89LP to provide 6 to 12 times more throughput than the classic 8051 when operating at the same frequency because instructions take only 1 to 4 clock cycles to execute instead of 12, 24 or 48 clock cycles. All AT89LP devices are fully binary compatible with the MCS®51 instruction set.

The actual performance increase for any application depends on the mix of instructions. Out of 255 instructions, 81 have a speed-up of 6, 39 have a speed-up of 8, 134 have a speed-up of 12 and one instruction has a speed-up of 24. Users migrating existing applications to the AT89LP2052/4052 from older devices have the option of keeping the system frequency the same with the benefit of increased performance, or reducing the clock frequency to save power without sacrificing performance. For example, an AT89C2051/4051 running at 12 MHz can be replaced by an AT89LP2052/4052 at 2 MHz and still show some performance increase in most applications.

3. Memory Organization

The AT89LP2052/4052 has separate address spaces for program memory and data memory.

3.1 Program Memory

Program memory consists of 2K/4K bytes of internal Flash memory. External program memory is not sup-

Flash Microcontrollers
AT89LP Primer

Application Note

3658A–MICRO–1/07

ported. The internal Flash memory is accessed at addresses 0000H - 07FFH for AT89LP2052 and 0000H - 0FFFH for AT89LP4052. User application code should not access program locations at addresses higher than allowed by the capacity of the device.

The program memory is In-System Programmable using the 4-wire SPI interface. Note that the AT89LP2052/4052 requires the connection of the $\overline{SS}$ pin during ISP (In-System Programming). This is an **additional** connection as compared to Atmel's previous AT89S series of devices. The AT89ISP software and cable include support for the $\overline{SS}$ connection on the Atmel 10-pin ISP header. The $\overline{SS}$ input improves the robustness of the ISP interface and allows the AT89LP2052/4052 to coexist with other slave devices on an SPI bus used for programming.

3.2 Data Memory

The AT89LP2052/4052 data memory consists of 256 bytes of internal RAM and the Special Function Registers (SFRs). This is twice the amount of RAM of the AT89C2051/4051 device. The program memory can also be used to store constant data. The AT89LP2052/4052 does not support external data memory. However, the MOVX A,@DPTR and MOVX A,@Ri instructions are redirected to the program memory and can be used to access (read) constant data instead (i.e., they act as equivalent to the instructions MOVC A,@A+DPTR and MOVC A,@A+PC). The MOVX @DPTR,A and MOVX @Ri,A instructions have no effect.

4. Configurable I/O

The feature most readily apparent to new users of the AT89LP2052/4052 is the configurable I/O. Classic 8051s such as the AT89C2051/4051 use quasi-bidirectional I/Os that can act as both inputs and outputs without configuration. While these I/Os have reduced software overhead, they do not excel in applications requiring large current sourcing capability or high-impedance inputs. The AT89LP2052/4052 allows each I/O to be individually configured in one of four operating modes: quasi-bidirectional, high-impedance input, push-pull output, or open-drain.

During reset all ports of the AT89LP2052/4052 default to high-impedance inputs to prevent any contention with other devices. Before any port pins can be used as outputs they must be configured accordingly by modifying the port mode registers P1M0 (C2H), P1M1 (C3H), P3M0 (C6H), and P3M1 (C7H). Generally, the reset initialization routine should set up the ports as required by the application. Any unused pins should not be left in high-impedance mode, but must be configured for either quasi-bidirectional or push-pull output operation to prevent them from floating. Users migrating from AT89C2051/4051 to AT89LP2052/4052 can place the I/Os in compatibility mode by writing 03h to P1M0 and 00h to P3M0 as shown below.

4.1 Port Configuration Examples

```
; The P1M0 and P3M0 registers are not bit-addressable, so Boolean operations
; are used.
P1M0    DATA    C2H         ; port 1 low config register
P1M1    DATA    C3H         ; port 1 high config register
P3M0    DATA    C6H         ; port 3 low config register
P3M1    DATA    C7H         ; port 3 high config register

; Example 1: Enable port compatibility at reset. All bits of P1 and P3 are
; placed in quasi-bidirectional mode, with the exception of P1.0 and P1.1,
; which remain high-impedance.
        CSEG                ; code segment
```

```
        ORG     0000H       ; location of reset vector
        jmp     xreset      ; vector
        .
        .
        .
xreset:                     ; code for responding to reset
        .
        .
        .
        mov     P1M0, #03H  ; configure P1
        mov     P3M0, #00H  ; configure P3
```

5. Four-level Interrupt Controller

The AT89LP2052/4052 includes an enhanced interrupt controller with support for four priority levels. The additional priority levels allow greater control over the interrupt response sequence in multiple interrupt systems. Four priority levels require two priority bits per interrupt. The lower order bits are stored in the existing IP (B8H) SFR. The higher order bits are stored in the additional IPH (B7H) SFR. The priority bits for individual interrupts share the same position in both IP and IPH. The polling order for interrupts of the same priority remains the same as previous devices.

In addition to the extra priority levels, the AT89LP2052/4052 can respond to interrupts up to four times faster than the AT89C2051/4051 due to its single-cycle architecture. Whereas AT89C2051/4051 can take between 18 and 54 clock cycles to vector to the interrupt service routine, AT89LP2052/4052 takes only 5 to 13 clock cycles in a single interrupt environment.

The following example configures the interrupts with different priority levels.

5.1 Interrupt Priority Example

```
; The IPH register is not bit-addressable, so Boolean operations are used.
IP      DATA    B8H         ; low priority register
IPH     DATA    B7H         ; high priority register
PX0H    EQU     00000001B   ; external interrupt 0
PT0H    EQU     00000010B   ; timer 0 overflow
PX1H    EQU     00000100B   ; external interrupt 1
PT1H    EQU     00001000B   ; timer 1 overflow
PSH     EQU     00010000B   ; serial interrupts
PCH     EQU     01000000B   ; comparator interrupt

; Set up priority between interrupts
;  priority 3: Comparator (Highest)
;  priority 2: INT1, Timer 0
;  priority 1: Serial
;  priority 0: INT0, Timer 1(Lowest)

        setb    ps          ; serial = level 1
        orl     IPH, #PX1H  ; INT1 = level 2
        orl     IPH, #PT0H  ; timer 0 = level 2
        setb    pc          ; comparator = level 1
        orl     IPH, #PCH   ; comparator = level 3
```

6. Enhanced Timer 0/1

The AT89LP2052/4052 has two 16-bit timer/counters: Timer 0 and Timer 1. The timers of the AT89LP2052/4052 increment every clock cycle. This rate is 12 times faster than on AT89C2051/4051. Users migrating to the AT89LP2052/4052 from older devices will need to update the timer values and/or reduce the clock frequency to achieve the same overflow rate. To facilitate long timer counts the 16-bit timer mode includes a full 16-bit auto-reload.

6.1 16-bit Auto-Reload

Mode 1 of Timer 0/1 functions as a 16-bit Auto-Reload timer. The reload value is stored in the register pairs {RH0 (94H), RL0 (92H)}/{RH1 (95H), RL1 (93H)}. When the timer reaches FFFFh, THx and TLx are reloaded with the value of RHx and RLx, respectively. By default the reload value is 0000h for compatibility with the old 16-bit timer mode. The time-out period for Mode 1 is given by the following equation using Timer 0 as an example:

$$\text{Mode 1:} \qquad \text{Time-out Period} = \frac{(65536 - \{RH0, RL0\})}{\text{Oscillator Frequency}}$$

6.2 Pulse Width Modulation

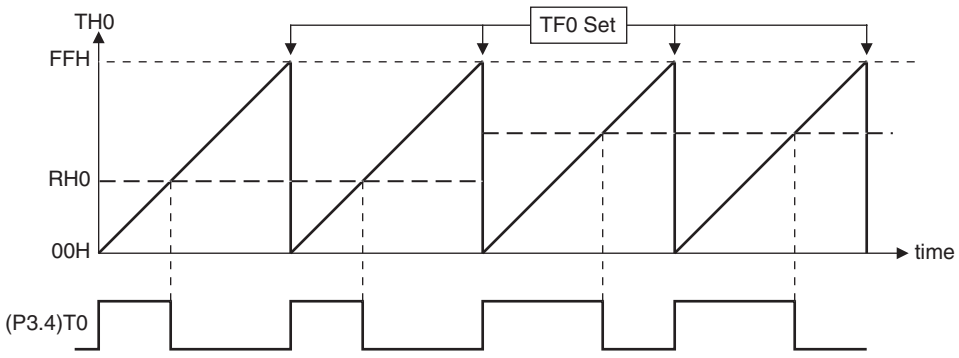
Timer 0 and Timer 1 can be independently configured to output an asymmetrical pulse width modulated (PWM) signal on their respective timer pins, T0 and T1, by setting the PWM0EN and PWM1EN bits in TCONB (91H). An example PWM waveform is shown in Figure 6-1. PWM is generally used with Mode 1 operation. In Mode 1 PWM, RLx determines the output frequency and RHx determines the duty cycle as shown in the following equations:

$$\text{Mode 1:} \qquad f_{out} = \frac{\text{Oscillator Frequency}}{256 \times (256 - RL0)}$$

$$\text{Mode 1:} \qquad \text{Duty Cycle \%} = 100 \times \frac{RH0}{256}$$

Writes to RHx are buffered such that updated duty cycle values will only take effect after the FFh to 00h overflow in THx. The timer overflow flags, interrupts and gating inputs will continue to function while in PWM Mode. The timer interrupt routine can be used to update the duty cycle value for the next period. The timer gating inputs can halt the PWM in its current state. The timer count and output state will remain constant until the gate input returns high. The PWM outputs can always be forced low immediately by clearing the P3.4 and P3.5 bits, respectively. These bits must be set high again before T0 or T1 can generate an output high level.

Figure 6-1. Asymmetrical Pulse Width Modulation Waveform



6.3 Timer Examples

```
; The TCONB and TMOD registers are not bit-addressable, so Boolean
; operations are used.
TCONB    DATA    91H          ; timer config b register
RL0      DATA    92H          ; timer 0 low reload register
RL1      DATA    93H          ; timer 1 low reload register
RH0      DATA    94H          ; timer 0 high reload register
RH1      DATA    95H          ; timer 1 high reload register
T0M0     EQU     00000001B    ; timer 0 mode 0
T1M0     EQU     00010000B    ; timer 1 mode 0
PWM0EN   EQU     01000000B    ; timer 0 pwm mode
PWM1EN   EQU     10000000B    ; timer 1 pwm mode

; Example 1: Configure timer 0 for 1 ms time-out at 12 MHz using 16-bit
; auto-reload mode 1.

        orl     TMOD, #T0M0  ; enable mode 1
        mov     RL0, #02CH   ; 1ms at 12 MHz =
        mov     RH0, #0CFH   ; 12500 x 80 ns
        mov     TL0, RL0     ; initialize timer
        mov     TH0, RH0     ;
        setb    TR0          ; start timer

; Example 2: Configure timer 1 for ~20 kHz PWM output at 25% duty cycle when
operating at 16 MHz.

        orl     TMOD, #T1M0    ; enable mode 1
        mov     RL1, #0FDH     ; 20.8 kHz = 16 MHz / (3 x 256)
        mov     RH1, #040H     ; 25%
        mov     TL1, RL1       ; init timer just before
        mov     TH1, #0FFh     ; rollover
        orl     TCONB, #PWM1EN ; pwm mode
        setb    TR1            ; start timer
```

7. Analog Comparator

The analog comparator of the AT89LP2052/4052 has several improvements over the AT89C2051/4051. The comparator can be disabled to save power, it can selectively operate during Idle, and it can generate an interrupt under a variety of output conditions. The analog comparator is controlled by the ACSR SFR (97H).

Table 7-1. ACSR – Analog Comparator Control & Status Register

Bit	7	6	5	4	3	2	1	0
97H	–	–	CIDL	CF	CEN	CM2	CM1	CM0
Reset Value	X	X	0	0	0	0	0	0

The analog comparator is disabled by reset. To enable/disable the comparator the CEN bit in ACSR should be set/cleared. The comparator interrupt flag, CF, is set whenever the output of the comparator matches the conditions specified by the CM_{2-0} bits. The comparator will be disabled during Idle unless the CIDL bit is set. When CIDL is set, the comparator interrupt may be used to exit Idle mode.

7.1 Analog Comparator Example

```
; The ACSR register is not bit-addressable, so Boolean operations are used.
ACSR    DATA    97H          ; comp control register
CM0     EQU     00000001B    ; mode bit 0
CM1     EQU     00000010B    ; mode bit 1
CM2     EQU     00000100B    ; mode bit 2
CEN     EQU     00001000B    ; enable
CF      EQU     00010000B    ; flag
CIDL    EQU     00100000B    ; enable during idle

; Enable port compatibility at reset. All bits of P1 and P3 are
; placed in quasi-bidirectional mode, with the exception of P1.0 and P1.1,
; which remain high-impedance.
        CSEG                 ; code segment

        ORG     0000H        ; location of reset vector
        jmp     main         ;
        .
        .
        .
        ORG     0033H        ; location of comparator vector
        jmp     cmp_int      ; vector
        .
        .
        .
   cmp_int:                  ; code for responding to comparator
        anl    ACSR, #(~CF)  ; clear flag
        .
        .
        .
        reti

   main:                     ;
        orl    ACSR, #CM2    ; set negative edge trigger
        orl    ACSR, #CEN    ; enable comparator
        .
        anl    ACSR, #(~CF)  ; clear spurious flag
        setb   EC            ; enable comparator interrupt
        setb   EA            ; enable global interrupts
        .
        .
        .
```

8. Watchdog Timer

The AT89LP2052/4052 features a watchdog timer which allows control of the microcontroller to be regained, should it be lost. When enabled, the timer will reset the microcontroller after a specified period has elapsed, unless prevented from doing so by the intervention of the firmware. The AT89LP2052/4052 watchdog always operates in hardware mode similar to AT89S52. Note that the watchdog on AT89LP2052/4052 is controlled by SFR WDTCON at address A7H as compared with WMCON (96H) in AT89S52.

To enable the watchdog timer, the sequence 1EH/E1H must be written to SFR WDTRST (A6H). The watchdog can only be disabled by a device reset. The WDTEN bit will be forced high when the hardware watchdog mode is enabled. Once the hardware watchdog timer is enabled, the firmware must write the 1EH/E1H sequence to WDTRST before the reset period elapses to prevent the timer from resetting the microcontroller. Each time WDTRST is written with the correct sequence, a new reset period begins, requiring another response from the firmware.

The watchdog timer reset period varies from 16K to 2048K system clock cycles, as specified by bits PS0, PS1 and PS2 in WDTCON. Refer to the AT89LP2052/4052 datasheet for the nominal reset periods corresponding to the bit settings. The timer reset period of the AT89LP2052/4052 depends on the frequency of the clock source driving the microcontroller and is 12 times faster than classic 8051s like AT89S52 for a given frequency. Reset (including reset generated by the watchdog timer) clears WDTEN, WDIDLE, SP0, SP1 and SP2, disabling the watchdog timer. A watchdog reset will set the WDTOVF bit. WDTOVF can be used to determine if a device reset was caused by the watchdog. WDTOVF must be cleared by software.

When WDIDLE = 0, the watchdog timer continues to operate even when the microcontroller is in Idle mode. To prevent the watchdog from timing out during Idle, an interrupt such as a timer overflow must be used to wake up the device periodically and reset the watchdog, or the WDIDLE bit may be set to disable the watchdog during Idle. The watchdog is always disabled during Power-down mode. To prevent the watchdog from timing out immediately upon exit from Power-down (or Idle with WDIDLE set), the watchdog should be reset just before entering Power-down (or Idle).

Table 8-1. WDTCON – Watchdog Control Register

Bit	7	6	5	4	3	2	1	0
A7H	PS2	PS1	PS0	WDIDLE	–	–	WDTOVF	WDTEN
Reset Value	0	0	0	0	X	X	X	0

A typical application of the watchdog timer is outlined in Section 8.1.

8.1 Watchdog Timer Example

```
; Use the hardware watchdog timer to regain control of the microcontroller
; if an operation takes longer than expected. The operation is expected to
; take less than 2 ms to complete and the reset period chosen is 2.7 ms.
; Adequate margin must be allowed between the desired reset period and the
; selected period to allow for the slop present in the timer. The WDTOVF
; flag is checked to determine if an error occurred and the appropriate
; routine is executed. The details of the operation and error routines are
; not shown. The WDTCON register is not bit-addressable, so Boolean
; operations are used.

WDTCON   DATA    A7H          ; watchdog control register
WDTRST   DATA    A6H          ; watchdog reset register
WDTEN    EQU     00000001B    ; watchdog timer enable bit
WDTOVF   EQU     00000010B    ; watchdog timer overflow flag
WDIDLE   EQU     00000100B    ; watchdog timer disable during idle bit
PS0      EQU     00100000B    ; watchdog timer period select bits
PS1      EQU     01000000B    ;
PS2      EQU     10000000B    ;

         CSEG                 ; code segment

         ORG     0000H        ; location of reset vector
         jmp     xreset       ; vector
         .
         .
         .
   xreset:                    ; code for responding to reset
         .
         .
         .
         mov     a, WDTCON    ; get Watchdog Control register
         anl     a, #WDTOVF   ; test Overflow Flag
         jnz     WDT_ERROR    ; WDTOVF=0 indicates error

         orl     WDTCON, #PS0  ; select 2.7 ms period at 12 MHz
         orl     WDTCON, #HWDT ; enable hardware watchdog
         mov     WDTRST, #01EH ; enable watchdog sequence 1
         mov     WDTRST, #0E1H ; enable watchdog sequence 2

     loop:
         ; Do something which normally takes less than 2 ms.
         .
         .
         .

         ; keep watchdog at bay
         mov     WDTRST, #01EH ; feed watchdog sequence 1
         mov     WDTRST, #0E1H ; feed watchdog sequence 2
         jmp     loop
```

9. Power Off Flag

The Power Off Flag (POF) indicates that power has been removed from the AT89LP2052/4052. This allows the firmware to differentiate between reset due to the application of power and reset due to the watchdog timer, or a logic high on the RST pin. POF is set when power is applied to the microcontroller and is not affected by the watchdog timer or by activity on RST. POF is located at bit four in SFR PCON (87H), and may be read, set, or cleared by firmware. Note that PCON is not bit-addressable.

A typical application of the Power Off Flag is outlined in Section 9.1.

9.1 Power Off Flag Example

```
; After reset, the microcontroller begins executing code at program memory
; address 0000H. POF is tested to determine if the controller was reset
; by the application of power (cold start) or by the watchdog timer or a
; high on RST (warm start).
; Code for the cold start and warm start routines is not shown.

POF     EQU     00010000B     ; Power Off Flag bit

        CSEG                  ; code segment

        ORG     0000H         ; location of reset vector
        jmp     xreset        ; vector
        .
        .
        .
   xreset:                    ; code for responding to reset
        .
        .
        .
        mov     a, PCON       ; get Power Control register
        anl     a, #POF       ; test Power Off Flag
        jz      WARM_START    ; POF=0 indicates reset from
                              ;   watchdog timer or RST
        xrl     PCON, #POF    ; clear POF for next time
        jmp     COLD_START    ; POF=1 indicates reset from power
```

10. Enhanced UART

The serial port (UART) of the AT89LP2052/4052 includes the enhanced features of framing error detection and automatic address recognition.

10.1 Framing Error Detection

Invalid stop bits can flag errors in the serial communication stream of the UART. The UART looks for missing stop bits in the communication. A missing stop bit will set the FE bit in the SCON register. The FE bit shares the SCON.7 bit position with SM0 and the function of SCON.7 is determined by SMOD0 (PCON.6). SCON.7 functions as SM0 when SMOD0 is cleared. If SMOD0 is set then SCON.7 functions as FE. FE will be set by missing stop bits regardless of the state of SMOD0. FE can only be cleared by software, i.e. a framing error will be remembered, even if subsequent communication is valid, until the FE bit is cleared by the software.

To use framing error detection with Modes 2 or 3, first set SM0 while SMOD0 = 0 and then set SMOD0 to map FE into SCON.

10.2 Automatic Address Recognition

Automatic Address Recognition is a feature which allows the UART to recognize certain addresses in the serial bit stream by using hardware to make the comparisons. This feature saves a great deal of software overhead by eliminating the need for the software to examine every serial address which passes by the serial port. This feature is enabled by default when setting the SM2 bit in SCON to use multiprocessor communication mode. In the 9-bit UART modes, Mode 2 and Mode 3, the Receive Interrupt flag (RI) will be automatically set when the received byte contains either the “Given” address or the “Broadcast” address. The 9-bit mode requires that the 9th information bit is a 1 to indicate that the received information is an address and not data.

In 8-bit Mode 1 the RI flag will be set if SM2 is enabled and the information received has a valid stop bit following the 8 address bits and the information is either a Given or Broadcast address.

Using the Automatic Address Recognition feature allows a master to selectively communicate with one or more slaves by invoking the given slave address or addresses. All of the slaves may be contacted by using the Broadcast address. Two special function registers are used to define the slave’s address, SADDR (A9H), and the address mask, SADEN (B9H). SADEN is used to define which bits in SADDR are to be used and which bits are “don’t care”. The SADEN mask can be logically ANDed with the SADDR to create the “Given” address which the master will use for addressing each of the slaves. Use of the Given address allows multiple slaves to be recognized while excluding others. The following examples will help to show the versatility of this scheme:

Slave 0:
SADDR = 1100 0000
SADEN = 1111 1101
Given = 1100 00X0

Slave 1:
SADDR = 1100 0000
SADEN = 1111 1110
Given = 1100 000X

In the previous example, SADDR is the same and the SADEN data is used to differentiate between the two slaves. Slave 0 requires a 0 in bit 0 and it ignores bit 1. Slave 1 requires a 0 in bit 1 and bit 0 is ignored. A unique address for slave 0 would be 1100 0010 since slave 1 requires a 0 in bit 1. A unique address for slave 1 would be 1100 0001 since a 1 in bit 0 will exclude slave 0. Both slaves can be selected at the same time by an address which has bit 0 = 0 (for slave 0) and bit 1 = 0 (for slave 1). Thus, both could be addressed with 1100 0000.

In a more complex system, the following could be used to select slaves 1 and 2 while excluding slave 0:

Slave 0:
SADDR = 1100 0000
SADEN = 1111 1001
Given = 1100 0XX0

Slave 1:
SADDR = 1110 0000
SADEN = 1111 1010
Given = 1110 0X0X

Slave 2:
SADDR = 1110 0000
SADEN = 1111 1100
Given = 1110 00XX

In the previous example, the differentiation among the 3 slaves is in the lower 3 address bits. Slave 0 requires that bit 0 = 0 and it can be uniquely addressed by 1110 0110. Slave 1 requires that bit 1 = 0 and it can be uniquely addressed by 1110 and 0101. Slave 2 requires that bit 2 = 0 and its unique address is 1110 0011. To select Slaves 0 and 1 and exclude Slave 2, use address 1110 0100, since it is necessary to make bit 2 = 1 to exclude slave 2.

The Broadcast Address for each slave is created by taking the logical OR of SADDR and SADEN. Zeros in this result are treated as don't-cares. In most cases, interpreting the don't-cares as ones, the broadcast address will be FFH.

Upon reset SADDR and SADEN are loaded with 0s. This produces a given address of all "don't cares" as well as a Broadcast address of all "don't cares". This effectively disables the Automatic Addressing mode and allows the microcontroller to use standard 80C51-type UART drivers which do not make use of this feature.

11. Serial Peripheral Interface

The Serial Peripheral Interface (SPI) permits compatible devices to communicate serially over a high-speed, synchronous bus. Devices resident on the bus act as masters or slaves, with only one master and one slave active at any one time. Data transfers are always initiated by a master, and are actually data exchanges, with data flowing from the master to the slave and from the slave to the master simultaneously.

SPI-compatible devices have four pins in common: SCK, MOSI, MISO, and $\overline{SS}$. All devices in a system have their SCK, MOSI, and MISO pins tied together. Data flows from master to slave via MOSI (Master Out Slave In) and from slave to master via MISO (Master In Slave Out). Data transfers are synchronized to a clock generated by the master and output on its SCK pin. SCK is an input for devices configured as slaves. A master device will always drive its SCK and MOSI pins, even when not currently transmitting. Inactive masters must be reconfigured as slaves to prevent them from driving their SCK and MOSI pins.

The $\overline{SS}$ (Slave Select) pins on the devices in the system are not bussed. Each slave is connected to its master by a select line from its $\overline{SS}$ input to a general-purpose output on the master. If a slave has multiple masters, the multiple select lines must be gated to its $\overline{SS}$ input. Masters do not utilize their $\overline{SS}$ pins during SPI data transfers, freeing them for use as general-purpose outputs.

To initiate an SPI data transfer, the active master selects a slave by applying a logic low to the slave's $\overline{SS}$ input. The master starts the serial clock, which it outputs on its SCK pin, and shifts out a byte on its MOSI pin, synchronized to the clock. Simultaneously, the slave shifts out a byte on its MISO pin, synchronized to the clock. When the master and slave have exchanged data, the transfer is complete. The master stops the serial clock and may deselect the slave. Slaves which are not selected ignore their SCK inputs and float their MISO outputs to avoid contention with the active output of the selected slave.

In the AT89LP2052/4052, the SPI is configured via SFR SPCR (D5H), the SPI Control Register. The frequency of the serial clock, the ordering of the serial data, and the relationship between the clock and the shifting and sampling of data are all programmable, as described below.

To enable the SPI feature, the SPE bit in SFR SPCR must be set; to disable the SPI, SPE is cleared. When the SPI is enabled, microcontroller pins P1.4, P1.5, P1.6 and P1.7 become $\overline{SS}$, MOSI, MISO, and SCK, respectively. The SPI may not operate correctly unless pins P1.4 - P1.7 are first programmed high. Reset sets pins P1.4 - P1.7 high and clears SPE, disabling the SPI. Note that because P1.7 starts high, enabling the SPI in master mode with CPOL = 0 will result in

a falling edge on SCK and clearing either MSTR or SPE will generate a rising edge on SCK. To prevent the slaves from being clocked by these edges, all slaves should be disabled with either $\overline{SS}$ = 1 or SPE = 0 when enabling/disabling the master.

The MSTR bit in SFR SPCR configures the microcontroller as a SPI master when set, and as a slave when cleared. Reset clears MSTR. When the microcontroller is configured as a SPI master, $\overline{SS}$ (P1.4) is not utilized and may be used as a general-purpose, programmable output.

When the microcontroller is configured as a SPI master, the frequency of the serial clock is determined by bits SPR0 and SPR1 in SFR SPCR. The frequency of the serial clock is the frequency of the microcontroller's clock source divided by the selected divisor. The divisor must be selected to produce a serial clock frequency which is compatible with the master's slaves. Refer to the AT89LP2052/4052 datasheet for the divisors corresponding to the settings of bits SPR0 and SPR1.

The DORD bit in SFR SPCR determines the order in which the bits in the serial data are transferred. Data is transferred least-significant bit (LSB) first when DORD is set; most-significant bit (MSB) first when DORD is cleared. Reset clears DORD. Note that only MSB-first data transfers are shown in the diagrams in the AT89LP2052/4052 datasheet.

The polarity of the SPI serial clock is determined by the CPOL bit in SFR SPCR. Setting CPOL specifies serial clock high when idle; clearing CPOL specifies serial clock low when idle. Reset clears CPOL.

The CPHA bit in SFR SPCR controls the phase of the SPI serial clock, which defines the relationship between the clock and the shifting and sampling of serial data. Setting CPHA specifies that data is to be shifted on the leading edge of the clock and sampled on the trailing edge. Clearing CPHA specifies that data is to be sampled on the leading edge of the clock and shifted on the trailing edge. Reset clears CPHA. Examples of SPI serial clock phase and polarity are shown in the diagrams in the AT89LP2052/4052 datasheet.

Only an AT89LP2052/4052 configured as an SPI master may initiate a data transfer. A data transfer is triggered by a byte written to SFR SPDR (86H), the SPI Data Register. As data is shifted out of the master, data from the selected slave is simultaneously shifted in, replacing the data in SPDR. When a data transfer is complete, the SPIF bit is set in SFR SPSR (AAH), the SPI Status Register. The data received from the slave may then be read from SPDR. Writing SPDR during a data transfer sets the Write Collision bit (WCOL) in SPSR. The progress of the data transfer is not affected by a collision. The SPIF and WCOL bits cannot be cleared directly by software. To clear bits SPIF and WCOL, first read SPSR and then read or write SPDR. This operation generally occurs within the normal operation of the SPI routine, e.g., polling the status of SPIF reads the SPSR register and then reading the received value from SPDR or sending the next value by writing SPDR automatically clears the flags.

An interrupt may be generated as an alternative to polling SPIF to determine the end of a SPI data transfer. To enable the SPI interrupt, three bits must be set. The first is the SPIE bit in SPCR, which causes an interrupt to be generated when SPIF is set. The second and third are the EA and ES bits in SFR IE (A8H). EA is the global interrupt enable bit. The SPI shares an interrupt vector with the UART, so the UART enable ES must also be set. When an SPI interrupt occurs, the SPI/UART interrupt service routine must determine the source of the interrupt. An SPI interrupt is indicated when the SPIF bit in SPSR is set.

11.1 Buffered Mode

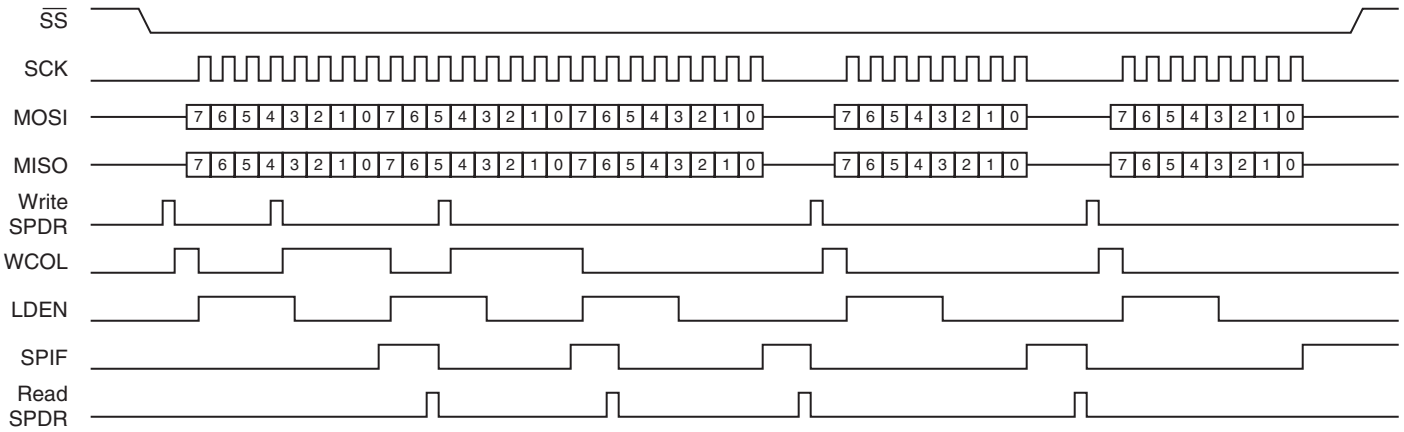
The SPI data register is normally double-buffered in the read direction but only single-buffered in the write direction. This means that received data from the previous transfer can be read while the current transfer is in progress, but the next byte to transmit cannot be queued while the current transfer is in progress. The AT89LP2052/4052 has the ability to also double-buffer SPDR in the write direction by setting the ENH bit in SPSR. Double-buffered mode is referred to as Enhanced Mode.

In enhanced mode, the Serial Peripheral Interface (SPI) on the AT89LP2052/4052 uses a double-buffered transmitter with WCOL as the buffer full flag. When data is written to SPDR, the transmit buffer is loaded with the data and the WCOL flag is set to show that the buffer is full. If the shift register is empty, as is the case for the first byte in a sequence, the contents of the buffer are transferred immediately to the shift register and WCOL is cleared low to flag buffer empty. Afterwards the user may queue data bytes in the transmit buffer by writing to SPDR while WCOL is low. WCOL is set high when SPDR is written, but the data is not transferred to the shift register. When the current byte transfer completes, the queued data in the buffer is moved to the shift register, WCOL is cleared, and the next transfer starts immediately. If a byte transfer completes with the buffer empty, the transmission halts until data is written to SPDR.

If the user waits until SPIF is set before loading the next data byte, gaps may occur between byte transfers. For higher utilization of the bus, the user may poll the status of WCOL to determine when to load the next data byte and thereby keep the transmitter full occupied.

The AT89LP2052/4052 includes the LDEN bit in SPSR to flag the first four bit slots of a transfer. The LDEN signal is not required for SPI operation. However, the LDEN signal may be used on slave devices to determine when an SPI transfer starts.

Figure 11-1. SPI in Enhanced Mode

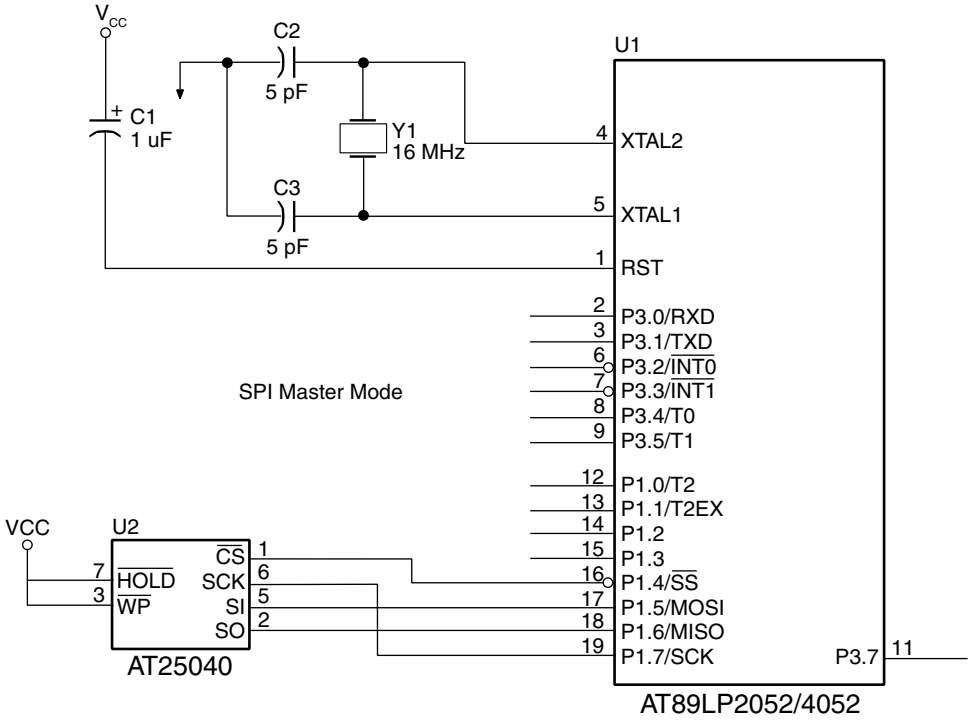


11.2 SPI Example

In the application shown below, the AT89LP2052/4052 is configured as an SPI master and interfaces to an Atmel AT25040 SPI-compatible EEPROM. The EEPROM provides 512 bytes of rewritable, nonvolatile storage while requiring only a four-pin interface to the microcontroller. The microcontroller and EEPROM are wired as shown in Figure 11-2. Note that the microcontroller's $\overline{SS}$ pin is used as a slave select, since it is unused when the microcontroller is configured as a SPI master. Additional EEPROMs may be connected to the microcontroller's SCK, MISO and MOSI pins, but each device must have its own select line.

Sample code for the application is shown in Section 11.2. A SPI master must be configured to meet the requirements of its slaves. The AT25040 datasheet states that the maximum clock rate for the device is 2 MHz. The microcontroller's clock source is a 16-MHz crystal (Figure 11-2), so a SPI serial clock divisor of 8 was chosen to produce a serial clock of 2 MHz. As shown in the AT25040 datasheet, the device's chip select ($\overline{CS}$) input must remain active (low) for the duration of an operation, which may include multiple data transfers. Also, the serial clock must be low when idle and data is transferred most-significant bit first. Therefore, CPHA = 1, CPOL = 0 and DORD = 0. In the example, SPI interrupts are not used.

Figure 11-2. AT89LP2052/4052 as an SPI Master



```
; Write/Read AT25C040 EEPROM via the Serial Peripheral Interface (SPI).
; Completion of AT25C040 programming is determined by polling the device.
; SPI interrupt is not used. Works with a microcontroller clock of 16 MHz
; (or slower).
;
; The AT25040 routines ("read_status", "enable_write", "read_byte",
; "write_byte") are excerpted from code previously made available by Atmel
; for use with the AT89Cx051 microcontrollers. In that code, access to the
; AT25040 was via "bit banging". The two routines which shifted the serial
; data in/out have been replaced by the single SPI routine "masterIO".

; Microcontroller registers and bit definitions.

SPCR    DATA    0d5H        ; SPI control register
SPSR    DATA    0aaH        ; SPI status register
SPIF    EQU     10000000B   ; interrupt flag
SPDR    DATA    86H         ; SPI data register
P1M0    DATA    C2H         ; port 1 low config register
P1M1    DATA    C3H         ; port 1 high config register

; Microcontroller connections to AT25040.

CS_     BIT     p1.4        ; AT25040 slave select
MOSI    BIT     p1.5        ; SPI
MISO    BIT     p1.6        ; SPI
SCK     BIT     p1.7        ; SPI

; AT25040 device command and bit definitions.

RDSR    EQU     05H         ; Read Status Register
WRSR    EQU     01H         ; Write Status Register
READ    EQU     03H         ; Read Data from Memory
WRITE   EQU     02H         ; Write Data to Memory
WREN    EQU     06H         ; Write Enable
WRDI    EQU     04H         ; Write Disable

A8      BIT     acc.3       ; MSB of address
NRDY    BIT     acc.0       ; high = write cycle in progress

main:
     ; SPI master mode initialization code.

        anl    P1M0, #04FH    ; configure spi pins
        orl    P1M1, #0B0H    ; sck, mosi, ss\ to output
        orl    P1M0, #040H    ; miso to input
        anl    P1M1, #0BFH    ;

        setb    CS_           ; deselect AT25040

        setb    MOSI          ; initialize SPI pins
        setb    MISO          ;
        setb    SCK           ;
```

```
        mov     SPCR, #01010101B    ; initialize SPI master
                                    ; interrupt disable, pin enable,
                                    ; MSB first, polarity 0, phase 1,
                                    ; clock rate /8
; Write one byte to AT25040 and verify (read and compare).
; Code to handle verification failure is not shown.
; Needs timeout to prevent write error from causing an infinite loop.

        call    enable_write   ; must precede each byte write
        mov     a, #DATA       ; data
        mov     dptr, #ADDRESS ; address
        call    write_byte     ; write
   wchk:
        call    read_status    ; check write status
        jb      NRDY, wchk     ; loop until done

        mov     dptr, #ADDRESS ; address
        call    read_byte      ; read
        cjne    a, #DATA, ERROR ; jump if data compare fails
        .
        .
        .
read_status:

   ; Read device status.
   ; Returns status byte in A.

        clr     CS_            ; select device
        mov     a, #RDSR       ; get command
        call    masterIO       ; send command
        call    masterIO       ; get status
        setb    CS_            ; deselect device
        ret

enable_write:

   ; Enable write.
   ; Does not check for device ready before sending command.
   ; Returns nothing. Destroys A.

        clr     CS_            ; select device
        mov     a, #WREN       ; get command
        call    masterIO       ; send command
        setb    CS_            ; deselect device
        ret

read_byte:

   ; Read one byte of data from specified address.
   ; Does not check for device ready before sending command.
   ; Called with address in DPTR.
   ; Returns data in A.
```

```
        clr     CS_             ; select device
        mov     a, dph          ; get high byte of address
        rrc     a               ; move LSB into carry bit
        mov     a, #READ        ; get command
        mov     A8, c           ; combine command and high bit of addr
        call    masterIO        ; send command and high bit of address
        mov     a, dpl          ; get low byte of address
        call    masterIO        ; send low byte of address
        call    masterIO        ; get data
        setb    CS_             ; deselect device
        ret

write_byte:

    ; Write one byte of data to specified address.
    ; Does not check for device ready or write enabled before sending
    ; command. Does not wait for write cycle to complete before returning.
    ; Called with address in DPTR, data in A.
    ; Returns nothing.

        clr     CS_             ; select device
        push    acc             ; save data
        mov     a, dph          ; get high byte of address
        rrc     a               ; move LSB into carry bit
        mov     a, #WRITE       ; get command
        mov     A8, c           ; combine command and high bit of address
        call    masterIO        ; send command and high bit of address
        mov     a, dpl          ; get low byte of address
        call    masterIO        ; send low byte of address
        pop     acc             ; restore data
        call    masterIO        ; send data
        setb    CS_             ; deselect device
        ret

masterIO:
    ; Send/receive data through the SPI port.
    ; A byte is shifted in as a byte is shifted out,
    ; receiving and sending simultaneously.
    ; Waits for shift out/in complete before returning.
    ; Expects slave already selected.
    ; Called with data to send in A. Returns data received in A.

        mov     SPDR, a         ; write output data
    bbb:
        mov     a, SPSR         ; get status
        anl     a, #SPIF        ; check for done
        jz      bbb             ; loop until done

        move    a, SPDR         ; read input data
        ret
```

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-47-50
Fax: (33) 4-76-58-47-60

Literature Requests
www.atmel.com/literature