Microchip PIC18F452
Peripheral Hardware
Timer/Counters

(Data Sheet Sections 10-13)
Timer/Counters

- Four timers: Timer0 … Timer3
  - Timer0 can be 8-bit or 16-bit,
  - Timer1 and Timer3 are **16-bit only**
  - Timer2 is two separate 8-bit timers ⇒ CCP PWM mode

- Timers 0, 1 and 3 can act as counters
- Remember –
  - Timer counts ticks at a known frequency
  - Counter counts (usually asynchronous) pulses
Timer0

- Configurable as Timer or Counter by $T0CON<T0CS>$
- Configurable as 8-bit or 16-bit, by $T0CON<T08BIT>$
- Readable and writable
- Dedicated 8-bit programmable prescaler (divider)
- Internal ($F_{OSC}/4$) or external ($T0CKI/RA4$) clock source
- Edge select ($\uparrow$ or $\downarrow$) for external clock
- Interrupt-on-overflow ($0xFF \rightarrow 0$ or $0xFFFF \rightarrow 0$)
Timer0 Control Register

- **TMR0ON**: Enable/disable Timer0
- **T08BIT**: Select 8-bit or 16-bit
- **T0CS**: Select Timer0 clock source
- **T0SE**: Select positive/negative edge on external clock
- **PSA**: Enable/disable prescaler
- **T0PS2 : T0PS0**: Select prescaler value (256 to 2)
Timer0: 8-bit Mode

Note: Upon RESET, Timer0 is enabled in 8-bit mode with clock input from TOCKI max. prescale.
Timer0: 16-bit Mode

Hardware mechanism for updating all 16 bits simultaneously:

- **TMROH** is actually a buffered copy of the high byte of Timer0, latched when **TMR0L** is read – **read TMR0L first!**
- Write to Timer0 also occurs through this buffer. **TMROH** is written first, then Timer0 is updated when **TMR0L** is written – **write TMR0L last!**
- This mechanism is always enabled for Timer0

**Note:** Upon RESET, Timer0 is enabled in 8-bit mode with clock input from TOCKI max. prescale.
Timer1

- 16-bit Timer/Counter
- Register TMR1 = TMR1H:TMR1L readable and writable
- Three modes, set by T1CON<TMR1CS>, T1CON<T1SYNC>
  - Timer
  - Synchronised Counter – increments on Q4:Q1 edge
  - Asynchronous Counter – must meet timing requirements
- Internal (F_{osc}/4) or external (T1OSI/RC1) clock source
- Can be self-clocked by adding external crystal
  ⇒ can count when µP in sleep mode
- Interrupt-on-overflow (0xFFFF → 0)
- Can be reset by CCP special event trigger
**Timer1 Control Register**

- **RD16**: Enable 16-bit read/write mode
- **T1CKPS1**: Select prescale value (8 to 1)
- **T1OSCEN**: Enable/disable Timer1 external oscillator
- **#T1SYNC**: Synchronise external clock with system clock
- **TMR1CS**: Select Timer1 clock source
- **TMR1ON**: Enable/disable Timer1

---

**T1CON: Timer1 Module Control Register**

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>RD16</th>
<th>T1CKPS1</th>
<th>T1CKPS0</th>
<th>T1OSCEN</th>
<th>#T1SYNC</th>
<th>TMR1CS</th>
<th>TMR1ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO/BO Reset</td>
<td>0</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MCLR Reset</td>
<td>u</td>
<td>–</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
</tbody>
</table>

---

David Rye :: MTRX 3700  
PIC18F452 Peripheral Hardware :: Slide 9 of 74
Timer1: 16-bit Read/Write Mode

- Setting `T1CON<RD16>` enables a hardware mechanism for updating all 16 bits simultaneously.
- Similar to Timer0:
  - read `TMR1L` first!
  - write `TMR1L` last!
- If `RD16` is clear, `TMR1H` is directly accessed – take care of overflow.

*Note 1:* When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.
Timer1: External Clocking

- When T1OSCEN is set, T1OSO and T1OSI are inputs for oscillator crystal (up to 200kHz)
- Note that this oscillator or the signal on T13CKI can also clock Timer3

**Note 1:** When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.
Timer3

- 16-bit Timer/Counter
- Registers TMR3H, TMR3L readable and writable
- Three modes, set by T3CON<TMR3CS>, T3CON<TSYNC>
  - Timer
  - Synchronised Counter – increments on Q4:Q1 edge
  - Asynchronous Counter
- Internal \((F_{OSC}/4)\) or external (\(\uparrow T13\text{CLKI}/RC0\ or T1OSI/RC1)\) clock source
- Can be self-clocked by adding external crystal (Same clock source as Timer1)
- Interrupt-on-overflow (FFFF\rightarrow0)
- Can be reset by CCP special event trigger
### Timer3 Control Register

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>RD16</th>
<th>T3CCP2</th>
<th>T3CKPS1</th>
<th>T3CKPS0</th>
<th>T3CCP1</th>
<th>#T3SYNC</th>
<th>TMR3CS</th>
<th>TMR3ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PO/BO Reset</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MCLR Reset</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
</tbody>
</table>

- **RD16**: Enable 16-bit read/write mode
- **T3CCP2**: Select CCP1 and CCP2 clock sources
- **T3CKPS1**: Select CCP1 clock source
- **T3CKPS0**: Select CCP2 clock source
- **#T3SYNCH**: Synchronise external clock with system clock
- **TMR3CS**: Select Timer3 clock source
- **TMR3ON**: Enable/disable Timer3
Timer3 Block Diagram

- Circuit is near-identical to Timer1

Note 1: When the T1OSCN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.
Timer2

- Two separate 8-bit registers
  - Timer 2: TMR2
  - Period Register: PR2  period of PWM output
- TMR2 clocked only by internal \((F_{\text{osc}}/4)\) clock
- TMR2 increments until it matches PR2, then resets to zero on next increment
- Both TMR2 and PR2 readable and writable
- Programmable input prescaler (divider)
- Programmable output postscaler (divider), clocked by match of TMR2 match to PR2
- Interrupt on TMR2 match to PR2
**Timer2 Control Register**

### T2CON: Timer2 Module Control Register

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>0xFCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>P/C/BO</td>
<td>MCLR</td>
</tr>
<tr>
<td>Reset</td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
</tr>
<tr>
<td>TOUTPS3</td>
<td>TOUTPS2</td>
</tr>
<tr>
<td>TOUTPS1</td>
<td>TOUTPS0</td>
</tr>
<tr>
<td>TMR2ON</td>
<td>T2CKPS1</td>
</tr>
<tr>
<td>T2CKPS0</td>
<td></td>
</tr>
</tbody>
</table>

- **TOUTPS3 : TOUTPS0**: Select postscaler value (16 to 1)
- **TMR2ON**: Enable/disable Timer2
- **T2CKPS1 : T2CKPS0**: Select prescaler value (16 to 1)

- Status of the other two registers on all resets is
  - **TMR2**: 0000 0000
  - **PR2**: 1111 1111
Timer2 Block Diagram

- If \( PR2 = 0 \), \( TMR2 \) will not increment (0 match 0 now, set = 0 next tick)
- Both Prescaler and Postscaler counters are cleared on
  - Any reset
  - Write to \( TMR2 \)
  - Write to \( T2CON \)
- Note: when \( T2CON \) is written, \( TMR2 \) is not cleared
Best Practice: Configuring Hardware

- Subroutine for configuring a hardware module should
  - Configure pins if needed (TRISx registers)
  - Configure hardware
  - Clear any counters, etc if needed

- It should NOT
  - Assume that any configuration registers are in their reset states
  - Make any assumptions regarding the configuration of other hardware modules, nor change their state
Best Practice: Configuring Hardware

- Example: Configure Timer0 as (external) counter

```assembly
    call  setupTimer0      ; call subroutine

setupTimer0:
    bsf   TRISA, T0CKI    ; T0CKI pin is input
    movlw 0b'10101000'    ; enable T0, no prescale
    movwf  T0CON          ; clock on T0CKI rising
    CLRF   TMR0H          ; zero Timer0
    CLRF   TMR0L          ; must clear lo byte last
    return
```
Capture/Compare/PWM Modules

(Data Sheet Section 14)
CCP Modules

- Two (nearly) independent hardware modules

- Capture
  - Look for an event
  - If it occurs, store the 16-bit time as measured by a timer

- Compare
  - Program an event to occur at some future time as measured by a timer
  - When the timer value matches the programmed time, cause the designated event to occur

- PWM
  - Adjust PWM duty cycle (and base frequency) of PWM output

- Events can be logic levels on pins (capture, compare) or internal changes (compare)
Each of the two Capture/Compare/PWM (CCP) modules has a 16-bit register that can function as a
- 16-bit (timer value) Capture register – or –
- 16-bit (timer value) Compare register – or –
- 10-bit PWM master/slave duty cycle register

- Operation of CCP1 is identical to that of CCP2 – either can be configured for Capture or Compare or PWM
- Timer1 or Timer3 can be the reference time for CCPx¹, capture or compare modes, Timer2 is the reference time for PWM mode.

- BUT there are some interactions…

1. CCPx stands for either CCP1 or CCP2
## CCPx/CCPy Interactions

<table>
<thead>
<tr>
<th>CCP Mode</th>
<th>Timer Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture</td>
<td>Timer1 or Timer3</td>
</tr>
<tr>
<td>Compare</td>
<td>Timer1 or Timer3</td>
</tr>
<tr>
<td>PWM</td>
<td>Timer2 only</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCPx Mode</th>
<th>CCPy Mode</th>
<th>Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture</td>
<td>Capture</td>
<td>Timebase TMR1 or TMR3. Timebases can be different</td>
</tr>
<tr>
<td>Capture</td>
<td>Compare</td>
<td>The compare can use the special event trigger to clear TMR1 or TMR3, depending on timebase used.</td>
</tr>
<tr>
<td>Compare</td>
<td>Compare</td>
<td>The compares can use the special event trigger to clear TMR1 or TMR3, depending on timebase used.</td>
</tr>
<tr>
<td>PWM</td>
<td>PWM</td>
<td>PWMs will have the same frequency and update rate</td>
</tr>
<tr>
<td>PWM</td>
<td>Capture</td>
<td>No interaction between CCP1 and CCP2</td>
</tr>
<tr>
<td>PWM</td>
<td>Compare</td>
<td>No interaction between CCP1 and CCP2</td>
</tr>
</tbody>
</table>
CCPx Control Register

- **DCxB1 : DCxB0**: PWM Duty Cycle bits 1 and 0 (used in PWM mode only)
- **CCPxM3 : CCPxM0**: Select CCPx Mode (choose between 4 Capture Modes, 4 Compare Modes, and PWM Mode)
- **CONFIG3H<CCP2MX>** can be used to associate CCP2 with RB3
CCP – Capture Mode

- **CCPR1H**: **CCPR1L** captures the 16-bit value of **TMR1** or **TMR3** when a defined event occurs on pin **CCP1/RC2**
- Defined events are selected by **CCPxCON<3:0>** and are:
  - A falling edge
  - A rising edge
  - Every 4th rising edge
  - Every 16th rising edge
CCP – Capture Mode

- The timebase used (Timer1 or Timer3) must be enabled and running in Timer Mode or Synchronous Counter Mode for reliable captures
- Timebase configured by \texttt{T3CON<T3CCP2:T3CCP1>}
- If the \texttt{CCPx} pin is configured as an output, writing to the port can cause a capture to occur

- Interrupt flag \texttt{CCPxIF} is set when a capture occurs
- A false interrupt can occur when changing mode – first clear \texttt{PIE<CCPxIE>}, then clear \texttt{CCPxIF} after mode change
CCP – Compare Mode

- A 16-bit value representing a future time relative to TMR1 or TMR3 is written to the register CCPRxH:CCPRxL
- This value is continuously compared to the count in TMR1/TMR3
- When a match occurs
  - CCP1IF is set  – and –
  - A defined event occurs
CCP – Compare Mode Events

- Which event occurs is defined by \( \text{CCP}x\text{CON}<3:0> \)
- The event can be
  - \( \text{CCP}x \) pin driven High – or –
  - \( \text{CCP}x \) pin driven Low – or –
  - \( \text{CCP}x \) pin state Toggled – or –
  - \( \text{CCP}x \) pin state unchanged “Software Interrupt” – or –
  - \( \text{CCP}x \) pin state unchanged “Special Event Trigger”
    - Resets TMR1 or TMR3, \textit{without} setting TMR1IF or TMR3IF, – and –
    - Sets GO/DONE bit (ADCON0<2>), which starts an A/D conversion if A/D is enabled via ADCON0<ADON> (CCP2 only)
CCP – PWM Mode

- 8-bit Period written to PR2
- 10-bit On-time written to CCPRxL, CCPxCON<5:4>
- 10-bit on-time count formed from TMR2 + Q Clocks

- Duty cycle = OnTime / Period
- Must enable the CCPx pin as an output

Note: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time-base.
CCP – PWM Mode - Notes

- What the Data Sheet calls “duty cycle” is really the “on-time”

- The On-time counter $TMR2$ runs 4x faster than other Timers.
- Clearing $CCPxCON$ will force the $CCPx$ PWM output latch low. This latch is not the PortC I/O Latch
- The Timer2 Postscaler is not used in PWM Mode. It could be used to generate $TMR2$ interrupts at a lower frequency than the PWM frequency – e.g. for running a control loop
- If PWM On-Time $TMR2$ is greater than Period $PR2$, $CCPx$ pin is always on (high)

- See Data Sheet for PWM calculations and setup sequence
Universal Synchronous/Asynchronous Receiver/Transmitter (USART)

(Data Sheet Section 16)
Can be configured as
- Asynchronous, full duplex
- Synchronous – Master, half duplex
- Synchronous – Slave, half duplex

**Asynchronous** ⇒ Receiver (Rc) and transmitter (Tx) are clocked by different clocks (differences in frequency & phase)

**Synchronous** ⇒ Rc and Tx are clocked by a common clock

**Full duplex** ⇒ Both ends Rc and Tx simultaneously

**Half duplex** ⇒ Transmission in one direction and any time

**Master** ⇒ Supplies the synchronising clock
TXSTA: USART Transmit Status and Control Register

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>U-0</th>
<th>R/W</th>
<th>R</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td></td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
</tr>
<tr>
<td>bit 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bit Names:**
- **TX9:** 9-bit transmission (8-bit plus Parity)
- **TXEN:** Enable Transmitter
- **SYNC:** Select Asynchronous/Synchronous
- **BRGH:** Select high/low speed baud rate
- **TRMT:** Transmit Shift Register Status
- **TX9D:** 9th bit of transmit data (parity or address bit)
**USART Rx Status/Control Register**

- **SPEN**: Serial Port enable
- **RX9**: Enable 9-bit Receive
- **SREN**: Single Synchronous Receive Enable
- **CREN**: Continuous Receive Enable
- **ADDEN**: Enable Address bit detect
- **PERR**: Receive Parity Error
- **OERR**: Overrun Error
- **RX9D**: 9th bit of received data (parity or address bit)
USART Baud Rate Generator

- Baud Rate is number of bits per second
- Dedicated Baud Rate generator supports both Asynchronous and Synchronous modes
- Baud Rate = \( \frac{F_{\text{OSC}}}{(64 \times (X+1))} \) \(\Rightarrow\) solve for integer \(X\) that gives approximately correct baud rate
- Allowable clock frequency mismatch is about 3%
- Program by writing ‘X’ to `SPBRG` Register
- Bit shift clock is 16x or 64x baud rate, depending on `TXSTA<BRGH>` (Baud Rate Generate High speed)

- Rx data pin is sampled 3 times per bit, majority vote
USART Asynchronous Transmit

- Transmit is
  - One START bit, 8 or 9 Data bits, one STOP bit
  - Parity generation not supported in hardware ⇒ can use 9th bit for parity
- User writes byte to TXREG
- Tx Shift Register (TSR) loaded when previous STOP bit transmitted
- TXIF bit signals Transmit Buffer empty when TSR loaded
- TRMT bit shows status of the TSR
USART Asynchronous Transmit

- See Data Sheet for Tx, Rx configuration & operating sequences

- TSR Register is not mapped into data memory ⇒ not accessible
- When TXEN becomes set, TXIF will also be set, since the Tx buffer is not yet full (Tx data can be moved to the Tx buffer)
USART Asynchronous Receive

- Data Recovery block is shift register clocked at 16x baud rate
- Data is sampled 3 times per bit, majority vote
- When the STOP bit is sampled, RSR contents transferred to RCREG
- RCIF is set if the transfer completes
- Receive is “double-buffered” (2-deep FIFO) – can hold 2 received bytes whilst 3rd is shifting in.

- If RCREG is not read, 3rd byte will overwrite 2nd, and OERR flag is set
- FERR (Framing Error) flag is set if the STOP bit is read as low
- Rx bit 9 and FERR are also double-buffered ⇒ reading RCREG loads new values of RX9D and FERR, so must read RCREG first
USART Synchronous Modes

- Defer until later…
Analog to Digital Converter

(Data Sheet Section 17)
A/D Converter

- 8 (or 5) Channels of 10-bit A/D
- Multiplexed into sample-and-hold
- Successive-approximation converter
  - Takes ‘n’+2 A/D conversion clocks to convert ‘n’ bits
  - A/D conversion clock period must be ≥ 1.6us
- Ratiometric converter:
  - count = 1024 * \( V_{in}/V_{ref} \)
  - Can use external \( V_{REF} \) ⇒ loose 2 input channels

* These channels are implemented only on the PIC18F4X2 devices.
Hardware Requirements

- Sampling capacitor is about 120pF
- Internal impedance affects the charging time
- Change input channel, then must wait (acquisition time) for sampling capacitor to charge before starting conversion. Data Sheet has equation for min acquisition time
- Sampling cap disconnected from source when conversion begins
- **Maximum** recommended source impedance is 2k5 Ohm
A/D Registers

Registers associated with A/D
- ADCON0 – A/D Control0
- ADCON1 – A/D Control1
- ADRESH – A/D Result High
- ADRESL – A/D Result Low
A/D Control Register 0

- **ADCS1**: A/D conversion clock select (bit 2 in ADCON1)
  - Note: Select A/D conversion clock relative to system clock speed so that A/D conversion clock period is at least 1.6 us

- **CHS2**: Channel select (1 of 8 multiplexer)

- **GO/##DONE**: Start conversion

- **ADON**: Enable A/D converter
A/D Control Register 1

<table>
<thead>
<tr>
<th>ADCON1: A/D Control Register 1</th>
<th>0xFC1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Names</td>
<td></td>
</tr>
<tr>
<td>PO/BO Reset</td>
<td>0</td>
</tr>
<tr>
<td>MCLR Reset</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>R/W</th>
<th>R/W</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADFM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADCS2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCFG3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCFG2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCFG1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCFG0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **ADFM**: Result format select (left or right justified)
- **ADCS2**: A/D conversion clock select (bits 0,1 in ADCON0)
- **PCFG3 : PCFG0**: A/D Port Configuration
A/D Sequence and Result

- See Data Sheet for detailed configuration and operating sequence

- Conversion triggered by GO bit, or CCP Compare Special Event Trigger

- Note: do not set the GO bit in the same instruction that enables the A/D

- Can poll the GO/#DONE bit for completion, or use ADIF interrupt

- Result can be right or left-justified, depending on ADFM
Master Synchronous Serial Port (MSSP)

(Data Sheet Section 15)
MSSP Modes

- Used for communicating with peripheral ICs or μPs
- Highly flexible in operation
- Configurable as
  - Serial Peripheral Interface (SPI) Bus
  - Inter-Integrated Circuit (I²C) Bus
    - Master Mode
    - Multi-Master Mode
    - Slave Mode
Serial Peripheral Interface (SPI) Bus

A Three-Wire, Hardware-Addressed Synchronous Serial Bus
Serial Peripheral Interface (SPI) Bus

- SPI - Serial Peripheral Interface, defined by Motorola in 68HCXX (SPI is a Trademark)

- Allows simple synchronous serial (3 wire) interface to
  - Parallel I/O – 74HC165 SIPO, 74HC166 PISO, 74HC595 SIPO/latch
  - A/D and D/A – see Maxim, LT, AD, etc
  - Real-time clocks (Dallas), Sensors, etc
  - MAX7219 8-Digit LED drive

- Known as “synchronous serial port expansion” – see Peatman, schematics following

  - Note: Conventionally, “n-wire” = n wires + ground…
SPI Bus

- Microchip’s pin names:
  - SCK: serial clock
  - SDI: serial data in
  - SDO: serial data out
  - #SS: slave select

- Motorola’s pin names:
  - SCLK: serial clock
  - MISO: master-in slave-out data
  - MOSI: master-out slave-in data
  - #SS: slave select

- Master/Slave interface – Shift clock driven by “bus master”
- Synchronous clock shifts data in/out in 8-bit blocks
- Low to medium speed (1 Mbit/sec)
- Full duplex – simultaneous data transmission in both directions
SPI Bus – One Slave

- Transmit initiated by *writing* to SSPBUF
- Receive is initiated by *writing* (anything) to SSPBUF
- Master generates a burst of 8 clocks
- Data are transferred MSb first
- Note that an input byte and an output byte are transferred *simultaneously*
SPI Bus – Many Slaves

- Multiple ICs connected to the same bus
  - Use slave select (SS) or chip select (CS) to select one IC
  - CSs driven by general-purpose output bits
  - Expansion limited only by the need to select 1-of-N

- Master always
  - Selects one Slave
  - Initiates transmissions
SPI Configuration

**SSPCON1 Register sets:**
- SPI Master or Slave
- Shift clock source
- Shift clock frequency

**SSPSTAT Register sets:**
- Transmit on rising/falling edge of SCK
- Sample input at middle/end of bit time

- Note: both control and status registers have control and status functions!
SSPCON1 Register – SPI Mode

**SSPCON1: MSSP Control Register1 (in SPI Mode)**

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCOL</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>SSPOV</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>SSPEN</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>CKP</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>SSPM3</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>SSPM2</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>SSPM1</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>SSPM0</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

- **WCOL**: Write collision detected (next byte written before previous sent)
- **SSPOV**: SSPBUF overwritten on receive
- **SSPEN**: SSP Enable
- **CKP**: Shift clock polarity (shift on rising/falling edge)
- **SSPM3 : SSPM0**: SPI Master/Slave Mode, clock source

PO/BO Reset

<table>
<thead>
<tr>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

MCLR Reset

<table>
<thead>
<tr>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**SSPSTAT Register – SPI Mode**

**SSPSTAT: MSSP Status Register (in SPI Mode)**  
0xFC7

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>R/W</th>
<th>R/W</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CKE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D/#A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R/#W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SMP**: Sample input at middle/end of bit time
- **CKE**: Data transmitted on rising/falling edge of SCK
- **BF**: SSPBUF Full (for receive only)
SPI Master Mode

Transmit
- CKP ⇒ SCK idles high/low
- CKE ⇒ Tx on ↑ or ↓ edge

Receive
- SMP ⇒ Rx sample in middle/end of Tx bit time
SPI Slave Mode

Would normally run the SPI in Master Mode – *but* – when using SPI between two μPs, one must be in Slave Mode

In Slave Mode,

- SS pin and SSPEN bit must be asserted to enable shift clock and transmission output buffer
- Then, data will shift in and out of SDI and SDO as clock pulses appear on SCK

- See Data Sheet for SPI Slave Mode variants
MICROWIRE

- Serial interface defined by National Semiconductor
- Also used by Microchip, Fairchild, etc. for serial EEPROMs
- Subset of the SPI interface: half duplex, SPI mode 0

- Microwire devices can often be interfaced directly to a μC SPI interface (with appropriate configuration)
- Sometimes additional "glue" logic is needed
Inter-Integrated Circuit (I\textsuperscript{2}C) Bus

A Two-Wire,
Software-Addressed
Synchronous Serial Bus
Inter-Integrated Circuit (IIC or I²C) Bus

- I²C invented by Philips in early 1980s
- Connect a CPU to peripheral chips inside TV sets
- Specification now at version 2.1 (released in 2000)

- Philips now has more than 150 types of I²C chips
- Now more than 1,000 I²C-compatible chips from about 50 manufacturers

- Smart Management Bus (SMBus) is an extension of the I²C Bus developed by industry group – Smart Battery System Implementer’s Forum
Inter-Integrated Circuit (IIC or I²C) Bus

- Two wire synchronous serial interface
- Both wires are bidirectional
  - **SDA**: Serial Data
  - **SCL**: Serial Clock
- Master/Slave bus
- Master controls the clock, and signal transfers between devices (including Slave-to-Slave)
- Each device is software addressable with a unique 7-bit device address (now 10-bit)
- Any device can talk (Rx/Tx) with the master
- Multi-Master, with (hardware) procedure for clock synch
- No of devices limited only by max bus capacitance of 400pF – typical device capacitance is 10pF
- Data rates: 100kb/s (standard) and up to 3.4Mb/s (high-speed)
MSSP – I²C Mode

MSSP fully implements
- All Master and Slave functions
- Interrupt on START and STOP bus conditions to allow Multi-Mastering
- Standard Mode (100kb/s)
- 7- and 10-bit addressing
SSPSTAT Register – I²C Mode

SSPSTAT: MSSP Status Register (in I²C Mode)  0xFC7

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>R/W</th>
<th>R/W</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CKE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D/#A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R/#W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SMP**: Slew Rate Control control bit
- **CKE**: SMBus enable control bit
- **D/#A**: Data/#Address status bit (Slave Mode only)
- **P**: STOP Condition detected last status bit
- **S**: START Condition detected last status bit
- **R/#W**: Transmit in Progress status bit (Master) or Read/Write status bit (Slave)
- **UA**: Update Address status bit (10-bit Slave Mode only)
- **BF**: SSPBUF Full status bit

<table>
<thead>
<tr>
<th>PO/BO Reset</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLR Reset</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

0xFC7
SSPCON1 Register – I²C Mode

SSPCON1: MSSP Control Register1 (in I²C Mode) 0xFC6

- **WCOL**: Write collision status bit – indicates SSPBUF was written under incorrect conditions
- **SSPOV**: SSPBUF overwritten on receive status bit
- **SSPEN**: SSP Enable control bit
- **CKP**: Free-run clock/enable clock stretching control bit (for setup time)
- **SSPM3:SSPM0**: I²C Master/Slave Mode, 7/10-bit address, enable interrupts on START and STOP Condition
SSPCON2 Register – I²C Mode

SSPCON2: MSSP Control Register2 (I²C Mode only) 0xFC5

<table>
<thead>
<tr>
<th>Bit Names</th>
<th>GCEN</th>
<th>ACKSTAT</th>
<th>ACKDT</th>
<th>ACKEN</th>
<th>RCEN</th>
<th>PEN</th>
<th>RSEN</th>
<th>SEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bit 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- GCEN: Enable Interrupt on General Call Address (0x00) control bit
- ACKSTAT: Acknowledge from Slave status bit
- ACKDT: Acknowledge Data control bit
- ACKEN: Acknowledge sequence enable bit
- RCEN: Receive Enable control bit
- PEN: STOP Condition Enable
- RSEN: Repeated START Condition Enable
- SEN: START Condition/Stretch Enable
**I²C Transaction Protocol**

- Master always controls the bus, generates $SCK$
- Transaction initiated by the Master
  - Multi-byte transmission sequence
  - Must begin with the start condition and end with the stop condition or the restart condition
  - Last bit of the first byte indicates if the Master is going to transmit (write) or receive (read) from the device that is addressed

- Multi-Master Bus
  - If there are two Masters, an arbitration procedure comes into effect if both try to take control
  - Once a master has control, no other master can take control until a stop condition is sent and the bus placed in idle state
I\textsuperscript{2}C Bus States (Conditions)

- **F (FREE)**
  - The bus is free or idle
  - Data line S\textsubscript{DA} and clock line S\textsubscript{CL} are both high

- **S (START) or S\textsubscript{R} (Repeated START)**
  - Beginning of data transmission
  - S\textsubscript{DA} data line changes from high to low while the clock line S\textsubscript{CL} remains high
  - Bus becomes ‘busy’

- **C (CHANGE)**
  - While the clock line S\textsubscript{CL} remains low, data bit can be placed on data line S\textsubscript{DA}
  - S\textsubscript{DA} can transition as long as S\textsubscript{CL} remains low

- **D (DATA)**
  - Data bit (high or low) on S\textsubscript{DA} is valid during S\textsubscript{CL} high
  - Data bit must be maintained stable whenever S\textsubscript{CL} is high

- **P (STOP)**
  - Signals end of data transmission
  - S\textsubscript{DA} data line changes from low to high while the clock line S\textsubscript{CL} remains high
  - Bus becomes free again
I²C Bus States

FREE, START and STOP States

DATA and CHANGE States (Single Bit Transfer)
I²C Bus States

Data Transfer
- MSb First

![Diagram of I²C Bus States](image)
I²C Write Data (Tx by Master)

Master waits for ACK from the receiver on the last byte, then sets the STOP condition.

F = FREE  
S = START  
R = Repeated START  
C = CHANGE  
D = DATA  
P = STOP

Data Direction

Transmitter (Master)  
Receiver (Slave)

<n data bytes>  
<last data byte>
I²C Read Data (Rx by Master)

Master does not acknowledge the last byte \( \Rightarrow \) this signals the slave that the read is finished. Master then sets the STOP condition.

F = FREE
S = START
R = Repeated START
C = CHANGE
D = DATA
P = STOP

Data Direction

Receiver (Master)  Transmitter (Slave)

F S Slave Addr R A Data A . . . Data A P F

<last data byte>
Other Features

(Data Sheet Other Sections)
Other Features

- Configuration Bytes
- LVD (Low Voltage Detect)
- BOR (Brown-out Reset)
- WDT (Watch Dog Timer)
- Oscillator: 8 configurable oscillator modes
- Sleep mode
- PSP (Parallel Slave Port)