C programming, the USART and SPI interfaces

Schedule

MONDAY 28-NOV-05 (LECTURE)

- The USART and SPI interfaces

WEDNESDAY 30-NOV-05 (LECTURE)

- EXAM #3 – Open Book

FRIDAY 02-DEC-05 (LAB)

- Work on lab 10 (USART and SPI)

MONDAY 05-DEC-05 (LECTURE)

- Lab 10 due by 22:00
- No Class – Work on Projects
- Possibly FCQs

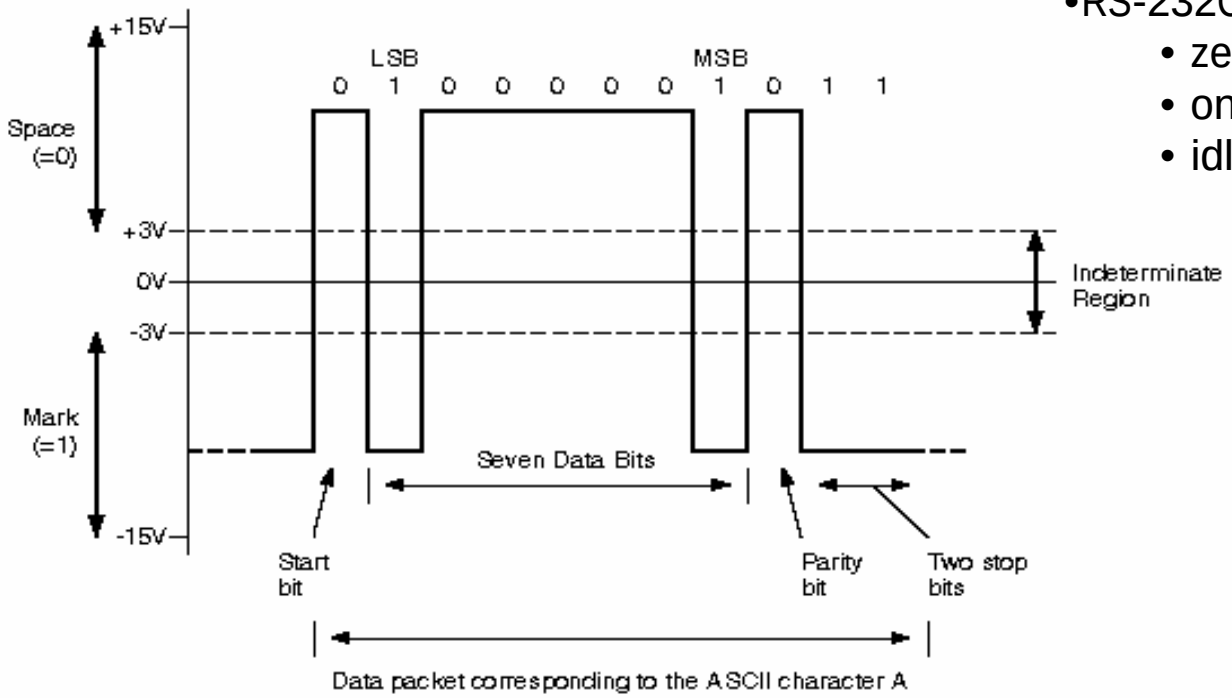
USART

- USART – Universal Synchronous Asynchronous Receiver Transmitter
- Can be used to generate many different serial asynchronous protocols (RS-232, RS-485, RS-422) and synchronous protocols
- We will only be concerned with the RS-232 interface
- Includes an internal clock generator
- Includes transmit and receive shift registers (serial to parallel conversion)

RS-232 signals

- Serial asynchronous protocol
  - One bit sent at a time (0 or 1)
  - No external clock is used (requires internal baud rate generation)
  - Peer to peer and point to point protocol (not distributed or master/slave)
- Hardware interface is full duplex and consists of
  - Transmit line
  - Receive line
  - Signal Ground
  - Other hardware handshaking signals (RTS, DCD, CTS, ...)
- A frame of data includes
  - Start bit (0) one baud in length
  - Up to 9 data bits (0 or 1) each one baud in length
  - Possibly a parity bit (even/odd) (0 or 1) on baud in length
  - Stop bit (1) can be 1, 1.5 or 2 bauds in length

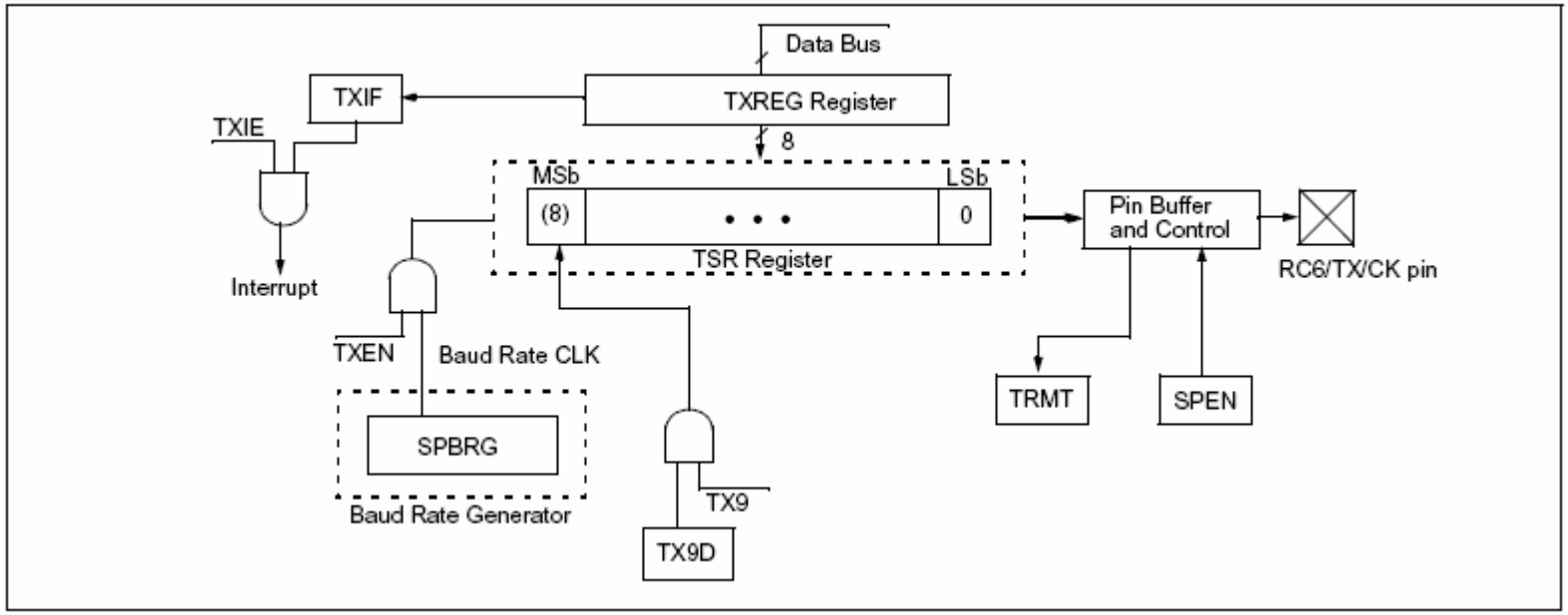
RS-232C electrical characteristics



- RS-232C
  - zero is +3 to +15 volts
  - one is -3 to -15 volts
  - idle state is one

USART Transmit

FIGURE 16-1: USART TRANSMIT BLOCK DIAGRAM



- TXREG to TSR register transfer occurs once previous stop bit starts transmission
- TXIF bit is set if TXEN is set (transmit enable) and once a TXREG to TSR transmit is complete
- SPBRG must be set correctly for baud rate generation
- SPEN must be set to enable serial port
- TSR is NOT a directly accessible register
- TRMT bit is set if TSR is empty and cleared when TSR is full

USART Transmit Configuration Register (TXSTA)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit

Asynchronous mode:
Don't care

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data
Can be Address/Data bit or a parity bit.

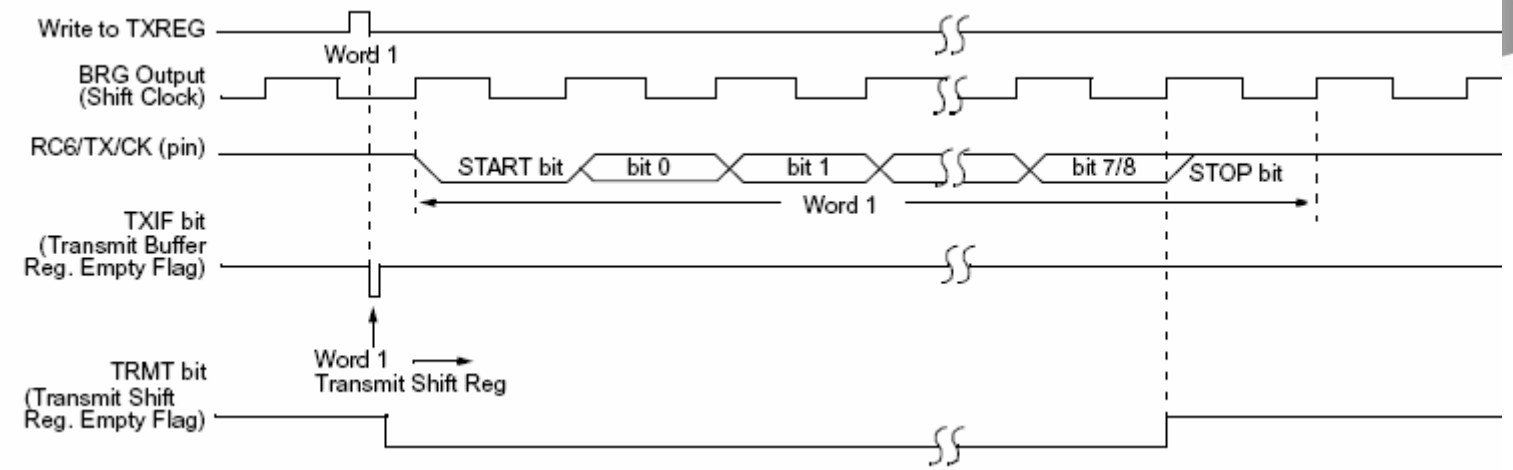
USART Transmit Registers

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

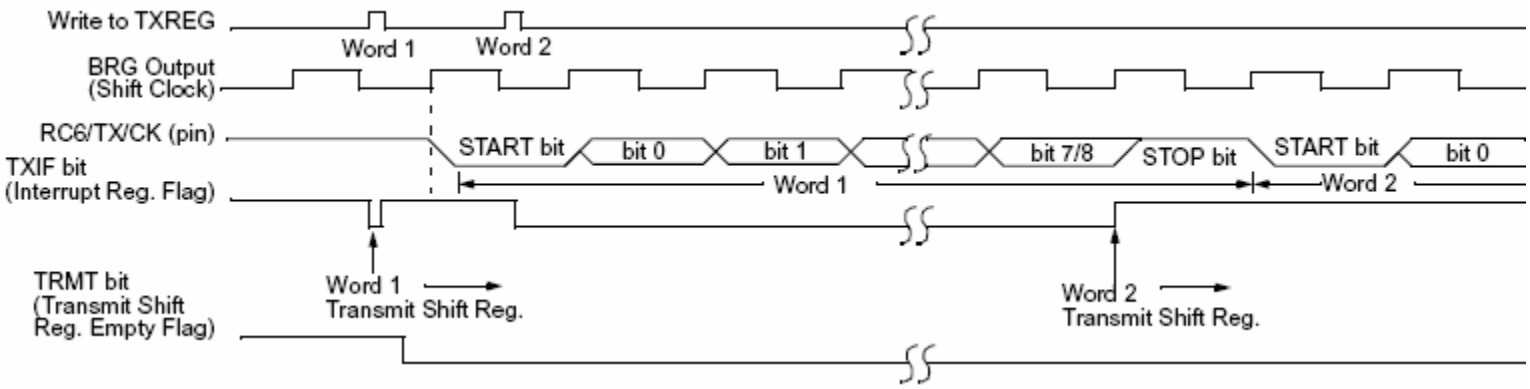
Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Asynchronous Transmission.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

Asynchronous transmission timing diagram

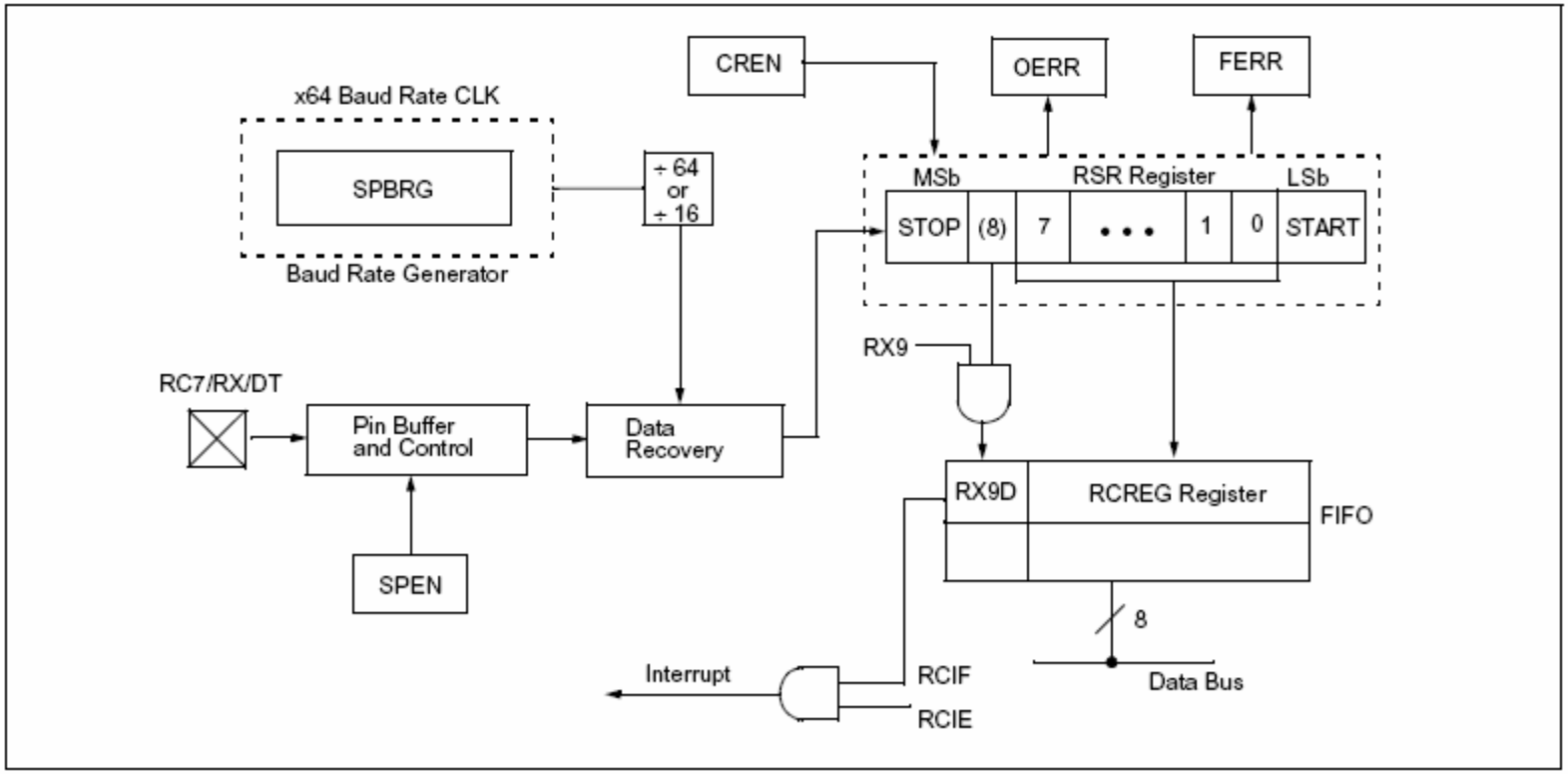


Single byte



Multiple byte

USART Receive



- Continuous receive (CREN) bit must be set to enable reception
- RXIF bit is set once a RSR to RCREG transfer is complete
- RCREG is a two byte FIFO (first in first out)
- SPBRG must be set correctly for baud rate generation
- SPEN must be set to enable serial port

USART Receive Configuration Register (RCSTA)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - Slave:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be Address/Data bit or a parity bit, and must be calculated by user firmware.

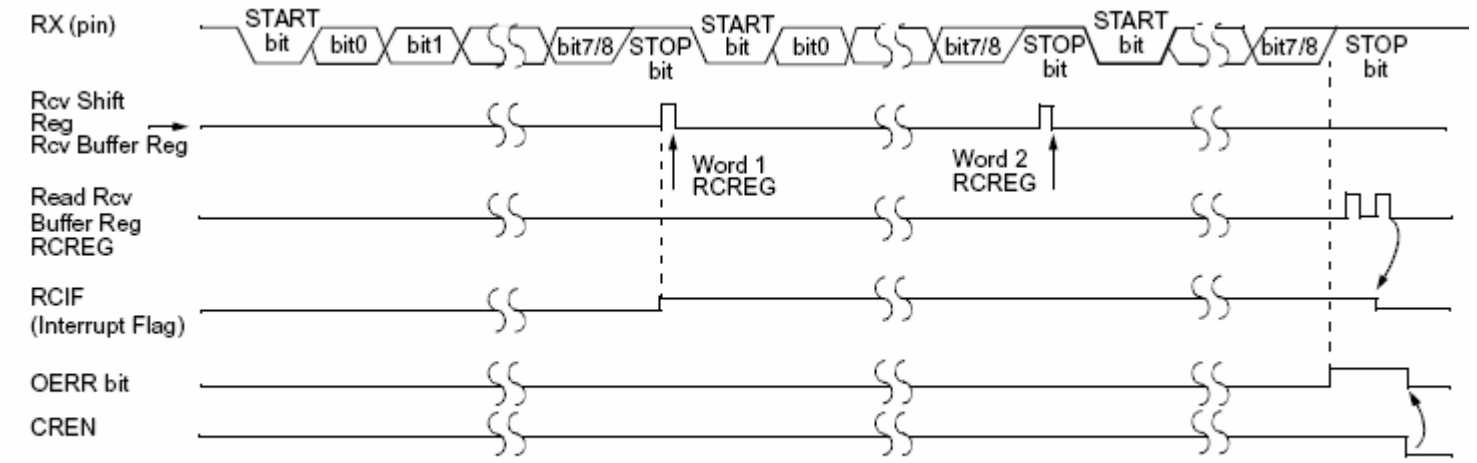
USART Receive Registers

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Asynchronous Reception.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

Asynchronous reception timing diagram



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

Baud Rate

- The baud rate defines the rate that data is transferred via the serial port. These values typically range from 110 to 115200 bps (bits/second)
- Each 0 or 1 is defined as a baud and the length of this baud is 1/(baud rate).
- Both the transmitter and receiver must generate their own baud rate independently. The start bit is used for synchronization
- Example:
  - At a baud rate of 9600 each baud would be 1/9600 = 104.2 usec in length

Baud Rate Generation

- The USART baud rate on the PIC is generated by dividing the system clock operating a F_{osc} = 10MHz.
- There are two options
  - BRGH=1 (high speed baud rate)
    - Baud rate = F_{osc}/(16*(SPRG+1))
  - BRGH=0 (low speed baud rate)
    - Baud rate = F_{osc}/(64*(SPRG+1))
- Example:
  - BRGH=1, Baud rate = 9600
    - SPRG = (10MHz/(16*9600))-1 = 64.104 (truncate to 64)
  - If BRGH=1 and SPRG = 64
    - Baud Rate = 10MHz/(16*(64+1)) = 9615.38 (0.16% error)

USART C Functions

- **void OpenUSART(auto unsigned char, auto unsigned);**
  - Function to open and configure the USART
  - The first byte contains the port configuration values
  - The second word (int) is the SPBRG value

- **#define DataRdyUSART() (PIR1bits.RCIF);**
  - Macro to check the RCIF bit and determine if new data is available

- **char ReadUSART(void);**
  - Function to read the value of the USART receive register RCREG

- **void WriteUSART(auto char);**
  - Function to write a byte from RAM to the USART transmit register TXREG

- **#define BusyUSART() (!TXSTAbits.TRMT);**
  - Macro to check if the USART is busy transmitting

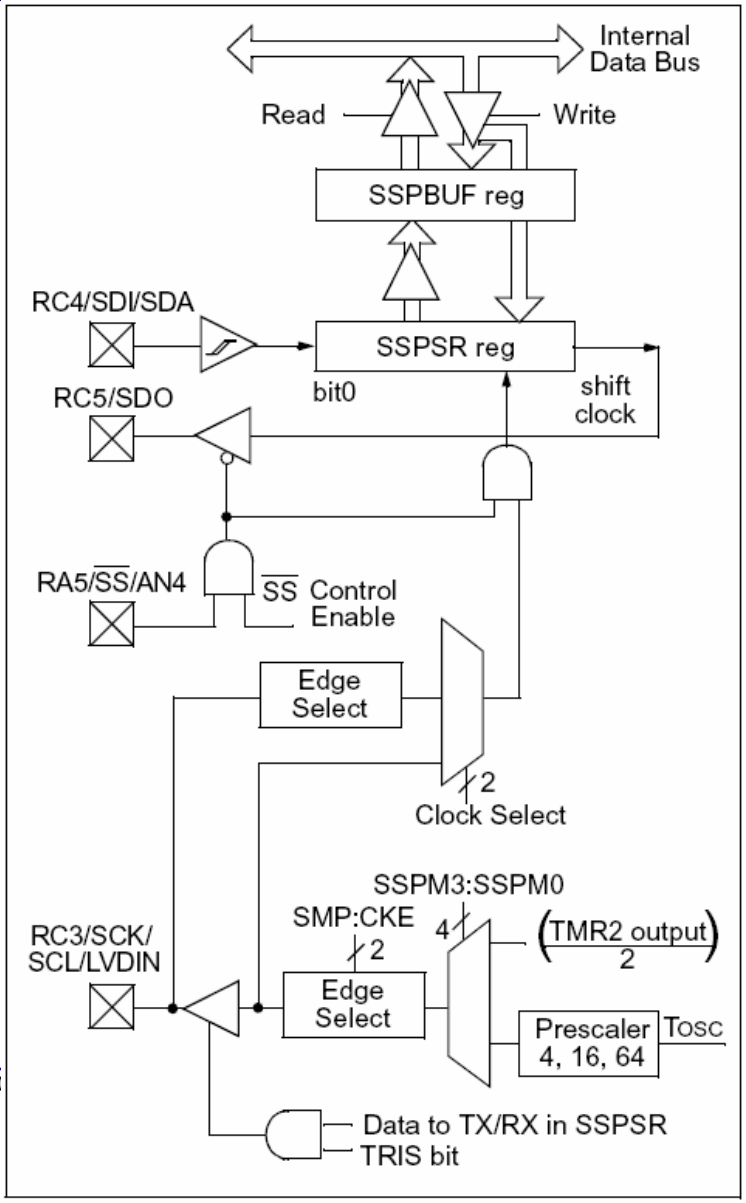
OpenUSART bit masks


```
#define USART_TX_INT_ON       0b11111111
#define USART_TX_INT_OFF      0b01111111
#define USART_RX_INT_ON       0b11111111
#define USART_RX_INT_OFF      0b10111111
#define USART_ASYNCH_MODE     0b11111110
#define USART_SYNCH_MODE      0b11111111
#define USART_EIGHT_BIT       0b11111101
#define USART_NINE_BIT        0b11111111
#define USART_SYNC_SLAVE      0b11111011
#define USART_SYNC_MASTER     0b11111111
#define USART_SINGLE_RX       0b11110111
#define USART_CONT_RX         0b11111111
#define USART_BRGH_HIGH       0b11111111
#define USART_BRGH_LOW        0b11101111
```

Master Synchronous Serial Port (MSSP)

- The MSSP on the PIC can be used to generate SPI and I^2C compatible signals
- SPI – serial peripheral interface
  - A serial full-duplex synchronous interface capable of transfer speeds of 5MHz
  - Operates in master or slave mode with multiple peripherals
  - Chip select is use for peripheral addressing
- I^2C – Inter-integrated circuit
  - Similar to SPI
  - Only half-duplex (same line is used for transmit and receive)
  - Addressing is done in software using an address configuration byte

MSSP block diagram in SPI mode



MSSP Status Register (SSPSAT)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit

SPI Master mode:

1 = Input data sampled at end of data output time

0 = Input data sampled at middle of data output time

SPI Slave mode:

SMP must be cleared when SPI is used in Slave mode

bit 6 **CKE:** SPI Clock Edge Select

When CKP = 0:

1 = Data transmitted on rising edge of SCK

0 = Data transmitted on falling edge of SCK

When CKP = 1:

1 = Data transmitted on falling edge of SCK

0 = Data transmitted on rising edge of SCK

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit

Used in I^2C mode only

bit 4 **P:** STOP bit

Used in I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** START bit

Used in I^2C mode only

bit 2 **R/$\overline{W}$:** Read/Write bit information

Used in I^2C mode only

bit 1 **UA:** Update Address

Used in I^2C mode only

bit 0 **BF:** Buffer Full Status bit (Receive mode only)

1 = Receive complete, SSPBUF is full

0 = Receive not complete, SSPBUF is empty

MSSP Control Register (SSPCON1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit

SPI Slave mode:

1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode.The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).

0 = No overflow

Note: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables serial port and configures SCK, SDO, SDI, and $\overline{SS}$ as serial port pins

0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, these pins must be properly configured as input or output.

bit 4 **CKP:** Clock Polarity Select bit

1 = IDLE state for clock is a high level

0 = IDLE state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin

0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled

0011 = SPI Master mode, clock = TMR2 output/2

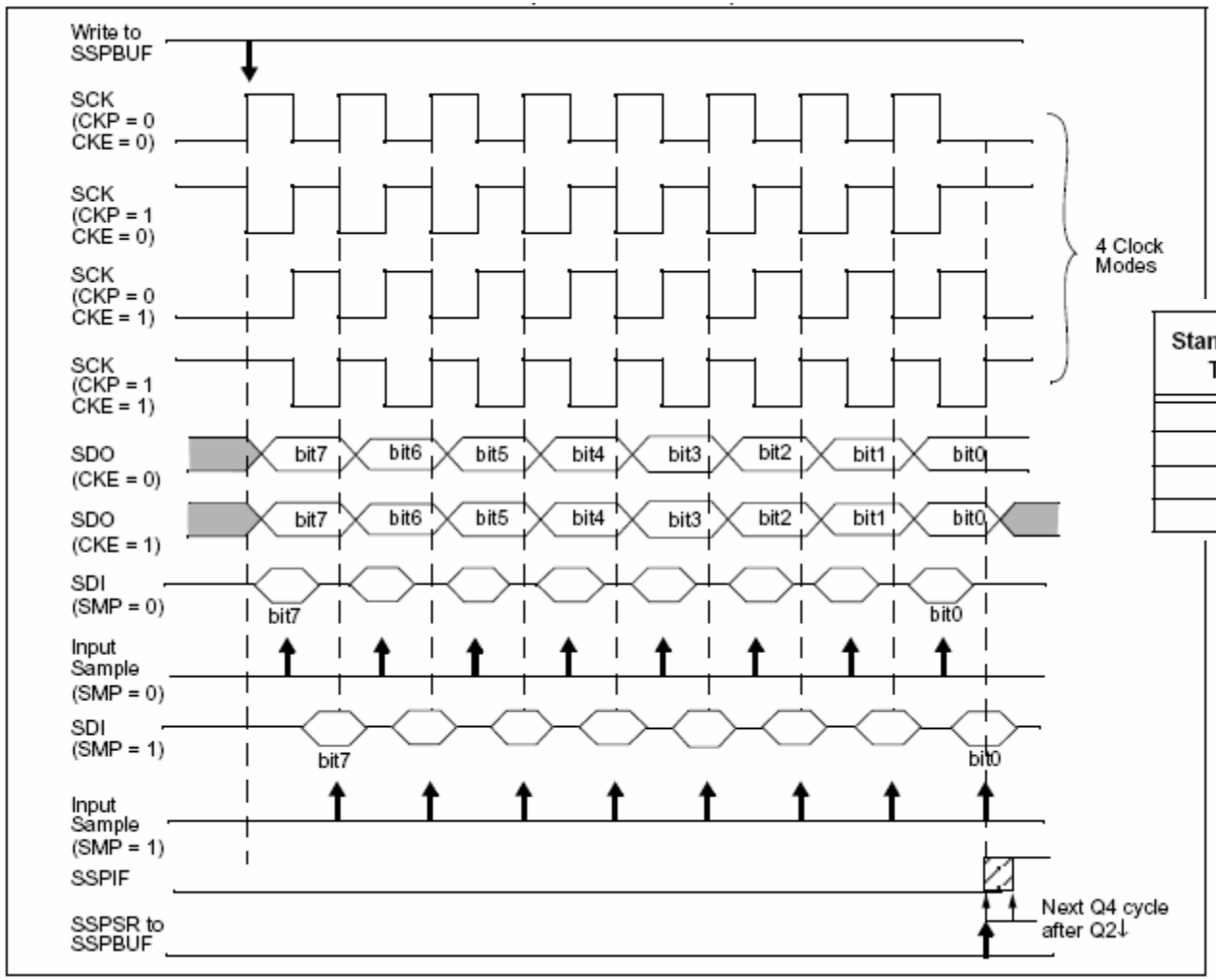
0010 = SPI Master mode, clock = Fosc/64

0001 = SPI Master mode, clock = Fosc/16

0000 = SPI Master mode, clock = Fosc/4

Note: Bit combinations not specifically listed here are either reserved, or implemented in I^2C mode only.

SPI Master Clocking modes



Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

The DAC uses mode_00

SPI Registers

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
TRISA	—	PORTA Data Direction Register							-111 1111	-111 1111
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the MSSP in SPI mode.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18C2X2 devices; always maintain these bits clear.

SPI C Functions

- **void OpenSPI(auto unsigned char sync_mode, auto unsigned char bus_mode, auto unsigned char smp_phase);**
  - Function to open and configure the MSSP as SPI
  - The first byte configures the clock rate
    - SPI_FOSC_4 // SPI Master mode, clock = Fosc/4
    - SPI_FOSC_16 // SPI Master mode, clock = Fosc/16
    - SPI_FOSC_64 // SPI Master mode, clock = Fosc/64
    - SPI_FOSC_TMR2 // SPI Master mode, clock = TMR2 output/2
    - SLV_SSON // SPI Slave mode, /SS pin control enabled
    - SLV_SSOFF // SPI Slave mode, /SS pin control disabled
  - The second byte configures the data transmission synchronization
    - mode_00 // CKE=1, CKP=0
    - mode_01 // CKE=0, CKP=0
    - mode_10 // CKE=1, CKP=1
    - mode_11 // CKE=0, CKP=1
  - The last bye configures the input data sampling
    - SMPEND // Sample data input at end of data out
    - SMPMID // Sample data input at middle of data out

SPI C Functions

- **unsigned char WriteSPI(auto unsigned char data_out);**
  - Function to write a byte to the SPI interface
  - Returns a zero on successful completion
  - Returns a -1 if a bus collision occurred

- Note: SPI also requires an additional chip select hardware line (CS) to select which peripheral should be listening to the bus. This signal is active low.

DAC Details

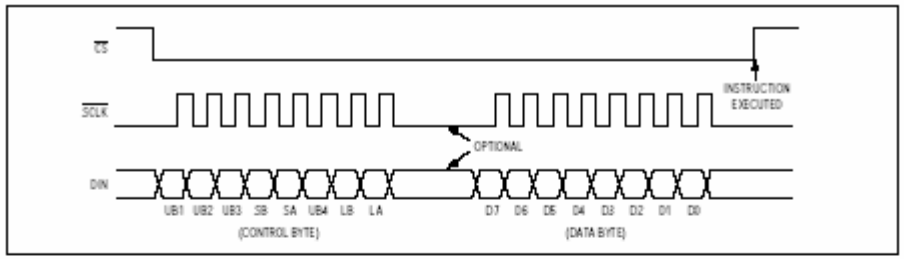


Figure 2. MAX522 3-Wire Serial-Interface Timing Diagram

Table 2. Serial-Interface Programming Commands

CONTROL								DATA								FUNCTION
UB1	UB2	UB3	SB	SA	UB4	LB	LA	B7 MSB	B6	B5	B4	B3	B2	B1	B0 LSB	
X	X	1	*	*	0	0	0	X	X	X	X	X	X	X	X	No Operation to DAC Registers
X	X	1	*	*	0	0	0									Unassigned Command
X	X	1	*	*	0	1	0				8-Bit DAC Data					Load Register to DAC B
X	X	1	*	*	0	0	1				8-Bit DAC Data					Load Register to DAC A
X	X	1	*	*	0	1	1				8-Bit DAC Data					Load Both DAC Registers
X	X	1	0	0	0	*	*	X	X	X	X	X	X	X	X	All DACs Active
X	X	1	0	0	0	*	*	X	X	X	X	X	X	X	X	Unassigned Command
X	X	1	1	0	0	*	*	X	X	X	X	X	X	X	X	Shut Down DAC B
X	X	1	0	1	0	*	*	X	X	X	X	X	X	X	X	Shut Down DAC A
X	X	1	1	1	0	*	*	X	X	X	X	X	X	X	X	Shut Down All DACs

X = Don't care.
* = Not shown, for the sake of clarity. The functions of loading and shutting down the DACs and programming the logic can be combined in a single command.

Table 3. Example of a 16-Bit Input Word

Loaded in First															Loaded in Last
UB1	UB2	UB3	SB	SA	UB4	LB	LA	B7	B6	B5	B4	B3	B2	B1	B0
X	X	1	0	0	0	1	1	1	0	0	0	0	0	0	0

- DAC output will not change until CS line is driven high after data transfer