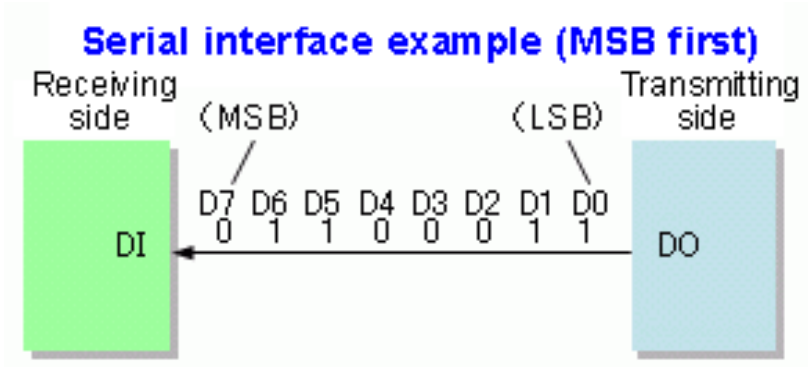
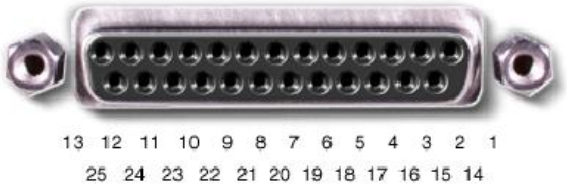
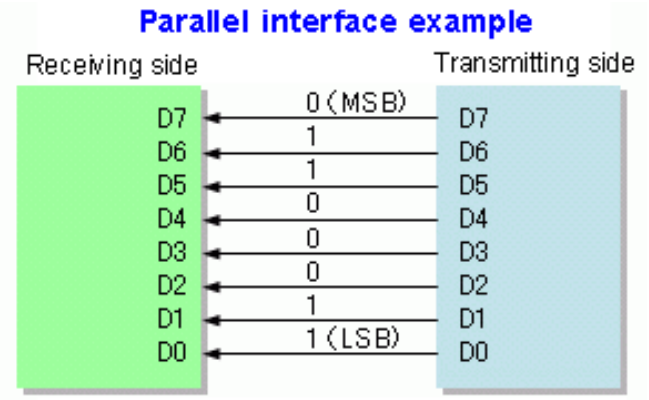
Digital Data Transmission

- Serial
  - Provides a low-cost (i.e., low wire/pin count) interface between devices
  - Many serial bus “standards”
    - RS-232
    - SPI
    - I^2C
    - USB
    - etc.



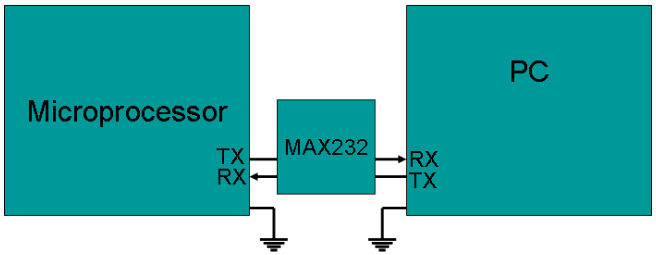
- Parallel
  - Can be a faster interface between devices if noise issues do not arrise.
  - Higher cost





RS-232

- Generally requires 3 wires
  - Transmit (Tx)
  - Receive (Rx)
  - Ground (Gnd)
- Signal levels are inverted
  - Logic high -> -5 V to -15 V
  - Logic low -> 5 V to 15 V
- Asynchronous
  - No common clock



ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

status of the Interrupt Identification Register. However I have never tested this.

Part 4 : Interfacing Devices to RS-232 Ports

RS-232 Waveforms

So far we have introduced RS-232 Communications in relation to the PC. RS-232 communication is asynchronous. That is a clock signal is not sent with the data. Each word is synchronized using it's start bit, and an internal clock on each side, keeps tabs on the timing.

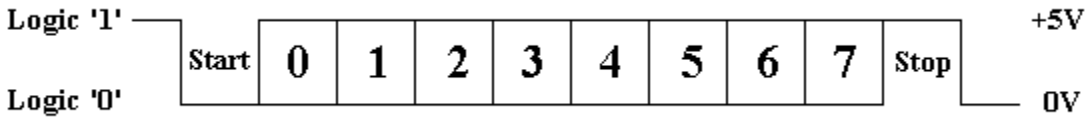


Figure 4 : TTL/CMOS Serial Logic Waveform

The diagram above, shows the expected waveform from the UART when using the common 8N1 format. 8N1 signifies 8 Data bits, No Parity and 1 Stop Bit. The RS-232 line, when idle is in the Mark State (Logic 1). A transmission starts with a start bit which is (Logic 0). Then each bit is sent down the line, one at a time. The LSB (Least Significant Bit) is sent first. A Stop Bit (Logic 1) is then appended to the signal to make up the transmission.

The diagram, shows the next bit after the Stop Bit to be Logic 0. This must mean another word is following, and this is it's Start Bit. If there is no more data coming then the receive line will stay in it's idle state(logic 1). We have encountered something called a "Break" Signal. This is when the data line is held in a Logic 0 state for a time long enough to send an entire word. Therefore if you don't put the line back into an idle state, then the receiving end will interpret this as a break signal.

The data sent using this method, is said to be *framed*. That is the data is *framed* between a Start and Stop Bit. Should the Stop Bit be received as a Logic 0, then a framing error will occur. This is common, when both sides are communicating at different speeds.

The above diagram is only relevant for the signal immediately at the UART. RS-232 logic levels uses +3 to +25 volts to signify a "Space" (Logic 0) and -3 to -25 volts for a "Mark" (logic 1). Any voltage in between these regions (ie between +3 and -3 Volts) is undefined. Therefore this signal is put through a "RS-232 Level Converter". This is the signal present on the RS-232 Port of your computer, shown below.

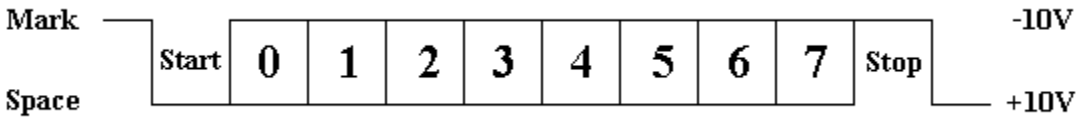


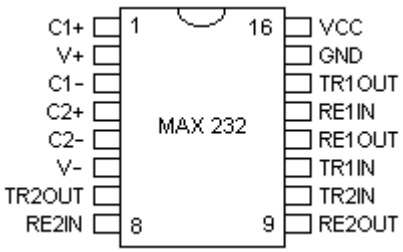
Figure 5 : RS-232 Logic Waveform

The above waveform applies to the Transmit and Receive lines on the RS-232 port. These lines carry serial data, hence the name Serial Port. There are other lines on the RS-232 port which, in essence are *Parallel* lines. These lines (RTS, CTS, DCD, DSR, DTR, RTS and RI) are also at RS-232 Logic Levels.

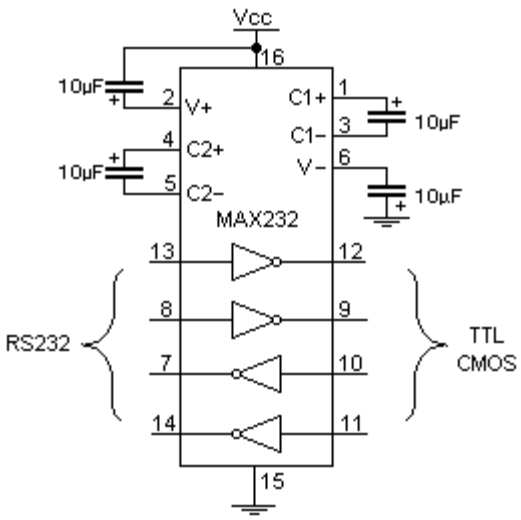
RS-232 Level Converters

Almost all digital devices which we use require either TTL or CMOS logic levels. Therefore the first step to connecting a device to the RS-232 port is to transform the RS-232 levels back into 0 and 5 Volts. As we have already covered, this is done by RS-232 Level Converters.

Two common RS-232 Level Converters are the 1488 RS-232 Driver and the 1489 RS-232 Receiver. Each package contains 4 inverters of the one type, either Drivers or Receivers. The driver requires two supply rails, +7.5 to +15v and -7.5 to -15v. As you could imagine this may pose a problem in many instances where only a single supply of +5V is present. However the advantages of these I.C's are they are cheap.



Above: (Figure 6) Pinouts for the MAX-232,
RS-232 Driver/Receiver.

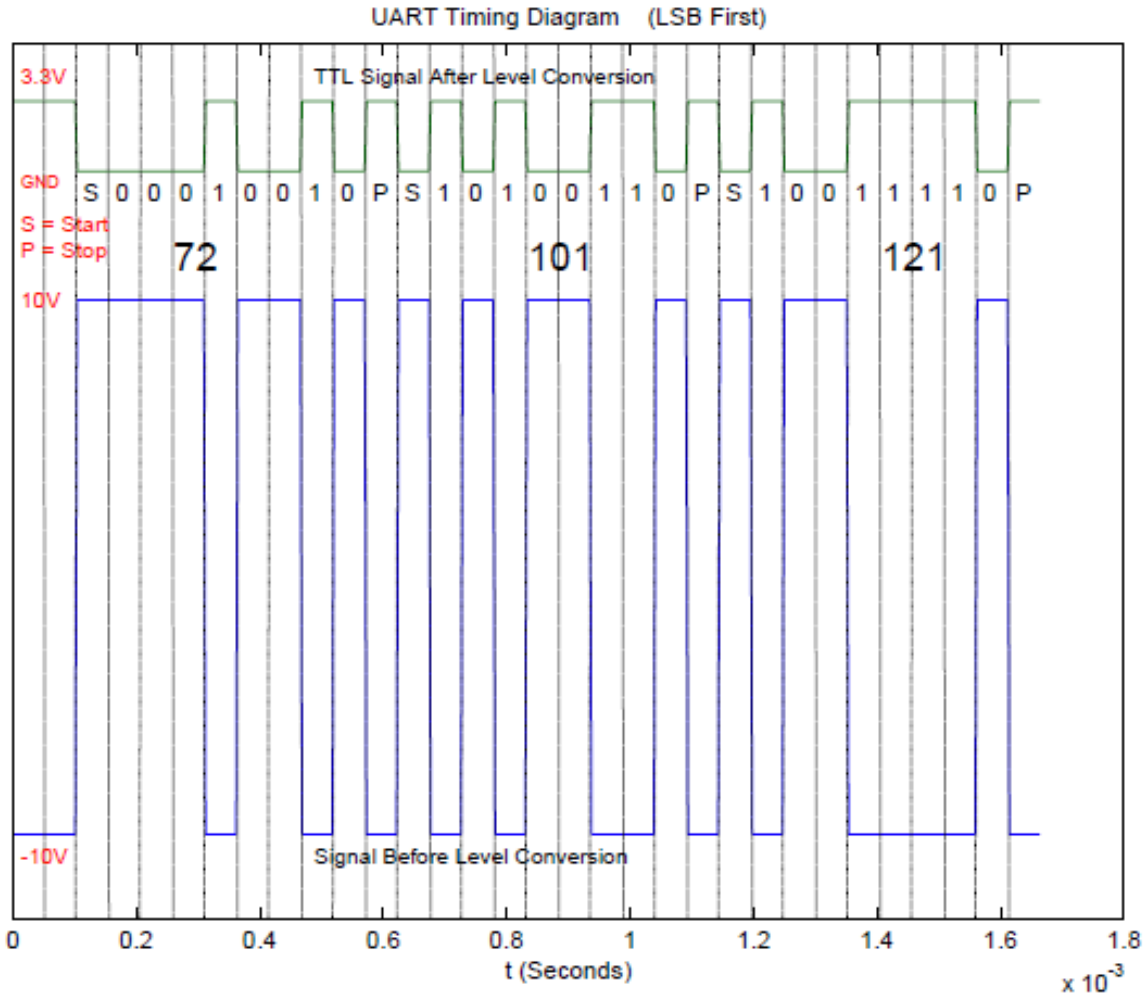


Right: (Figure 7) Typical MAX-232 Circuit.

Another device is the MAX-232. It includes a Charge Pump, which generates +10V and -10V from a single 5v supply. This I.C. also includes two receivers and two transmitters in the same package. This is handy in many cases when you only want to use the Transmit and Receive data Lines. You don't need to use two chips, one for the receive line and one for the transmit. However all this convenience comes at a price, but compared with the price of designing a new power supply it is very

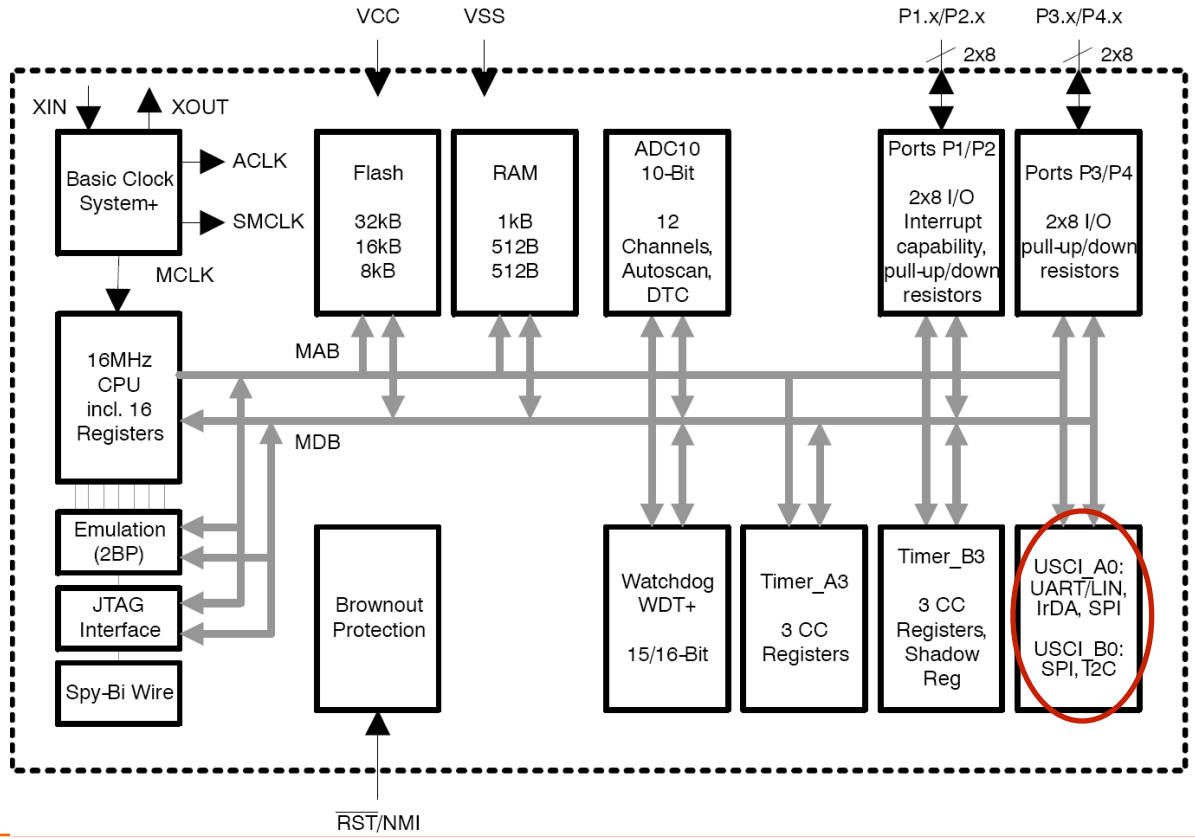
MSP430 Serial Communication

```
RS232 Serial Port sending the string 'H','e','y'.
ascii characters 72 (0x48), 101 (0x65) and 121 (0x79).
```



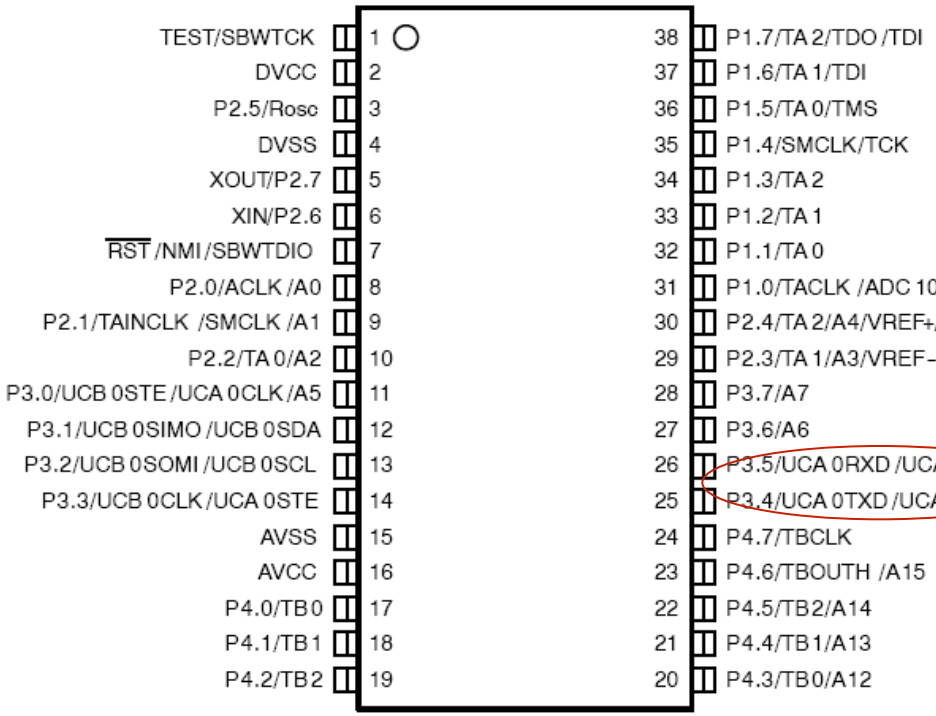
MSP430x22x2 USCI Module

- Universal Serial Communications Interface
- USCI_A0
  - UART, SPI, IrDA
- USCI_B0
  - SPI, I²C



MSP430x22x2 USI Module

MSP430x22x2 device pinout, DA package



PIN NAME (P3.X)	X	FUNCTION	CONTROL BITS / SIGNALS	
			P3DIR.x	P3SEL.x
P3.4/ UC0TXD/UC0SIMO	1	P3.4† (I/O)	I: 0; O: 1	0
		UC0TXD/UC0SIMO (see Note 3)	X	1
P3.5/ UC0RXD/UC0SOMI	1	P3.5† (I/O)	I: 0; O: 1	0
		UC0RXD/UC0SOMI (see Note 3)	X	1

MSP430x22x2 USCI Module

- USCI Registers: SPI Mode

Register	Short Form	Register Type	Address	Initial State
USCI_A0 control register 0	UCA0CTL0	Read/write	060h	Reset with PUC
USCI_A0 control register 1	UCA0CTL1	Read/write	061h	001h with PUC
USCI_A0 baud rate control register 0	UCA0BR0	Read/write	062h	Reset with PUC
USCI_A0 baud rate control register 1	UCA0BR1	Read/write	063h	Reset with PUC
USCI_A0 modulation control register	UCA0MCTL	Read/write	064h	Reset with PUC
USCI_A0 status register	UCA0STAT	Read/write	065h	Reset with PUC
USCI_A0 receive buffer register	UCA0RXBUF	Read	066h	Reset with PUC
USCI_A0 transmit buffer register	UCA0TXBUF	Read/write	067h	Reset with PUC
USCI_B0 control register 0	UCB0CTL0	Read/write	068h	001h with PUC
USCI_B0 control register 1	UCB0CTL1	Read/write	069h	001h with PUC
USCI_B0 bit rate control register 0	UCB0BR0	Read/write	06Ah	Reset with PUC
USCI_B0 bit rate control register 1	UCB0BR1	Read/write	06Bh	Reset with PUC
USCI_B0 status register	UCB0STAT	Read/write	06Dh	Reset with PUC
USCI_B0 receive buffer register	UCB0RXBUF	Read	06Eh	Reset with PUC
USCI_B0 transmit buffer register	UCB0TXBUF	Read/write	06Fh	Reset with PUC
SFR interrupt enable register 2	IE2	Read/write	001h	Reset with PUC
SFR interrupt flag register 2	IFG2	Read/write	003h	00Ah with PUC

MSP430x22x2 USCI Module

- Configuration

Note: Initializing or Re-Configuring the USCI Module

The recommended USCI initialization/re-configuration process is:

1) Set UCSWRST (`BIS.B #UCSWRST,&UCAxCTL1`)

2) Initialize all USCI registers with UCSWRST = 1 (including UCAxCTL1)

3) Configure ports.

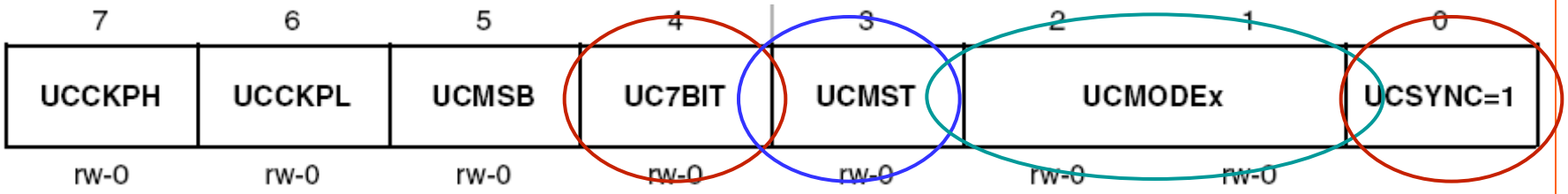
4) Clear UCSWRST via software (`BIC.B #UCSWRST,&UCAxCTL1`)

5) Enable interrupts (optional) via UCAxRXIE and/or UCAxTXIE

MSP430 User's Guide (pg. 15-5)

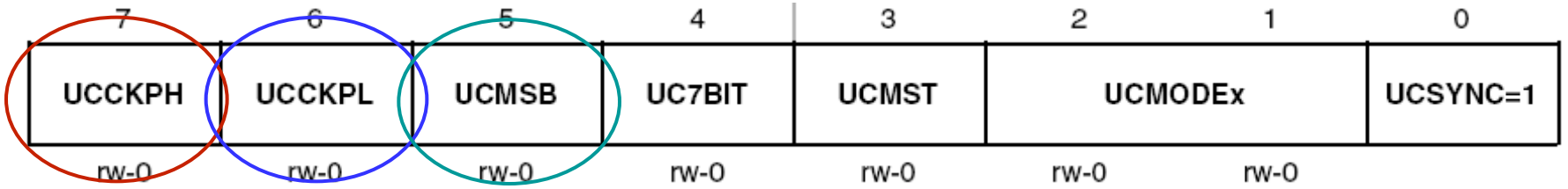
There are many of them!

USCI_B0—UCB0CTL0 Control Register 0



UC7BIT	Bit 4	Character length. Selects 7-bit or 8-bit character length. 0 8-bit data 1 7-bit data
UCMST	Bit 3	Master mode select 0 Slave mode 1 Master mode
UCMODEx	Bits 2-1	USCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00 3-Pin SPI 01 4-Pin SPI with UCxSTE active high: slave enabled when UCxSTE = 1 10 4-Pin SPI with UCxSTE active low: slave enabled when UCxSTE = 0 11 I^2C Mode
UCSYNC	Bit 0	Synchronous mode enable 0 Asynchronous mode 1 Synchronous Mode

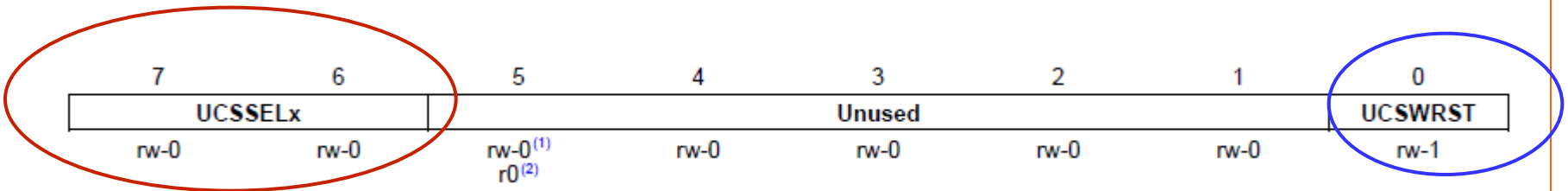
USCI_B0—UCB0CTL0 Control Register 0



7	6	5	4	3	2	1	0
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx		UCSYNC=1
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	

UCCKPH	Bit 7	Clock phase select. 0 Data is changed on the first UCLK edge and captured on the following edge. 1 Data is captured on the first UCLK edge and changed on the following edge.
UCCKPL	Bit 6	Clock polarity select. 0 The inactive state is low. 1 The inactive state is high.
UCMSB	Bit 5	MSB first select. Controls the direction of the receive and transmit shift register. 0 LSB first 1 MSB first

USCI_B0—UCB0CTL1 Control Register 1



UCSSELx	Bits	USCI clock source select. These bits select the BRCLK source clock.	
	7-6	00	UCLK
		01	ACLK
		10	SMCLK
		11	SMCLK

UCSWRST	Bit 0	Software reset enable	
		0	Disabled. USCI reset released for operation.
		1	Enabled. USCI logic held in reset state.

MSP430 Serial Communication

```
// USCI Transmit ISR
// Called when TXBUF is empty (ready to accept another character)
#pragma vector=USCIABOTX_VECTOR
__interrupt void USCI0TX_ISR(void) {

  if(IFG2&UCA0TXIFG) {
        if(printf_flag) {
          if (currentindex == txcount) {
              senddone = 1;
              printf_flag = 0;
              IFG2 &= ~UCA0TXIFG;
            }
          else {
              UCA0TXBUF = printbuff[currentindex];
              currentindex++;
            }
        } else if(UART_flag) {
                if(!donesending) {
                UCA0TXBUF = txbuff[txindex];
                if(txbuff[txindex] == 255) {
                  donesending = 1;
                  txindex = 0;
                }
                else txindex++;
                }
        }
          IFG2 &= ~UCA0TXIFG;
        }
        if(IFG2&UCB0TXIFG) {
                IFG2 &= ~UCB0TXIFG;
        }
}
```

UCSI interrupt vector for A0 & B0 Tx

Shared interrupt register—check specific flag

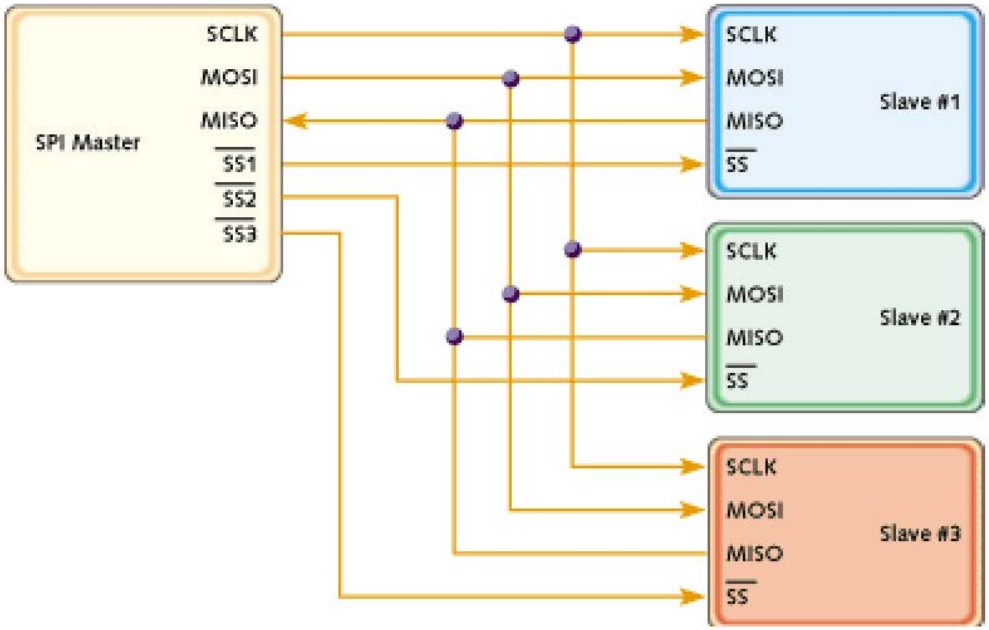
Do something here and…

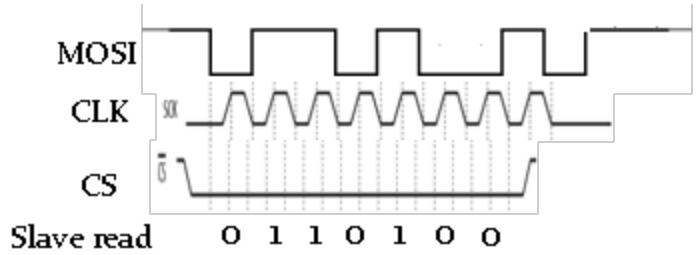
…reset interrupt flag (if necessary)

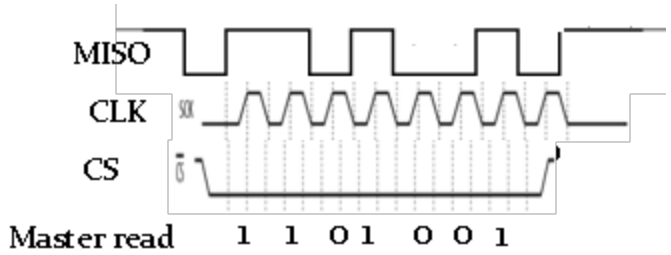
Check for module B0 Tx interrupt

MSP430x22x2 USCI Module

- SPI (Serial Peripheral Interface)

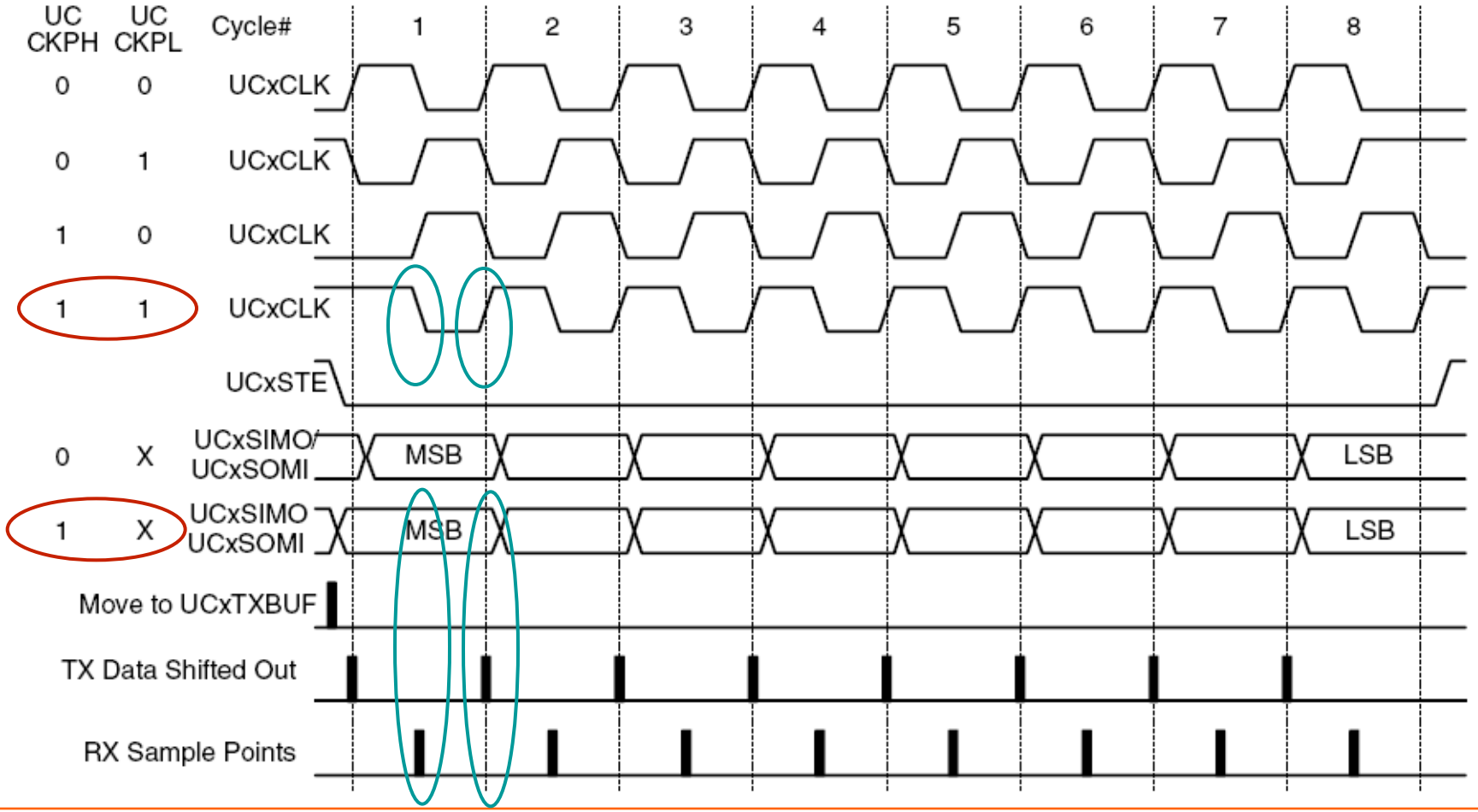






MSP430x22x2 USCI Module

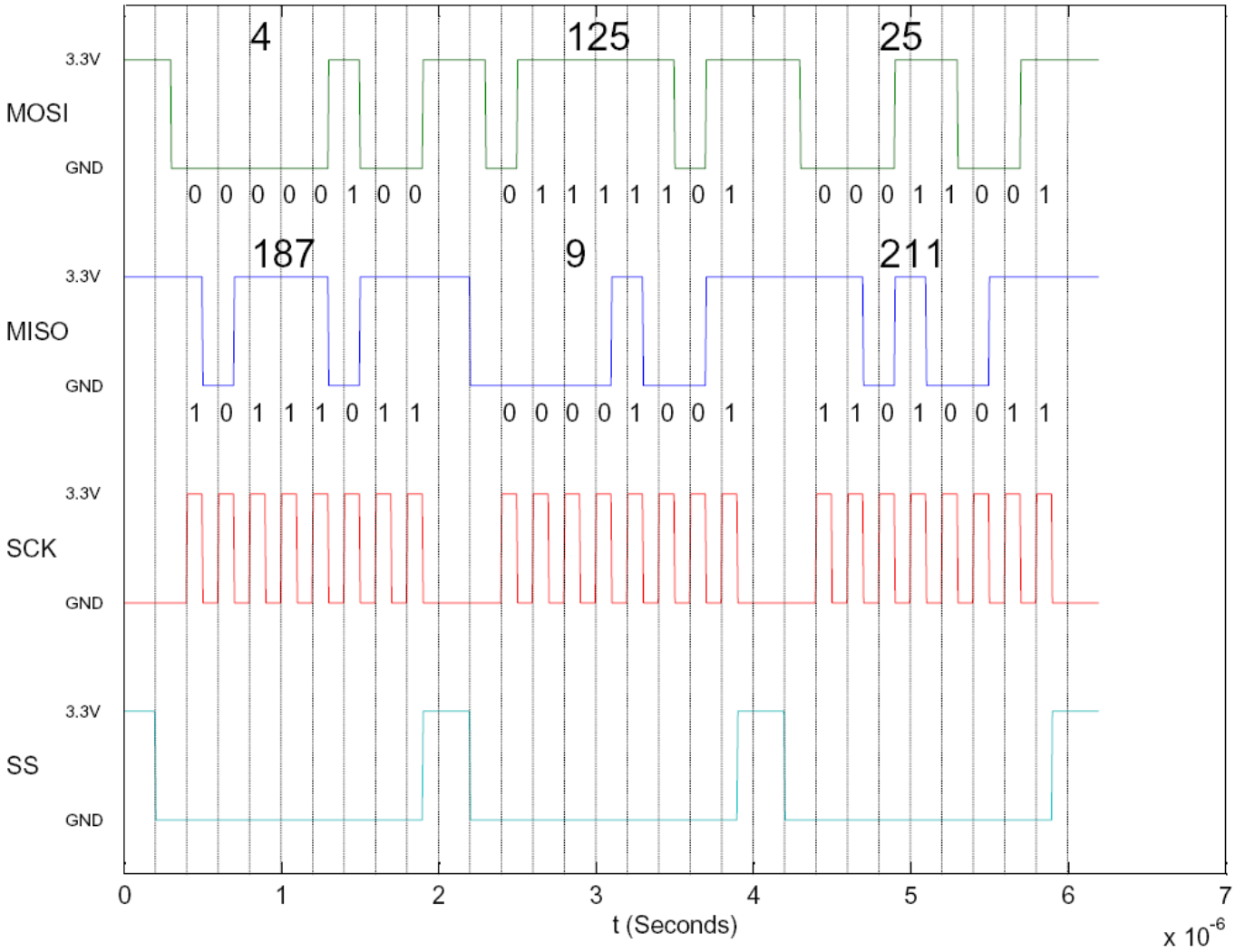
- USCI SPI Timing with UCMSB =1



MSP430x22x2 USCI Module

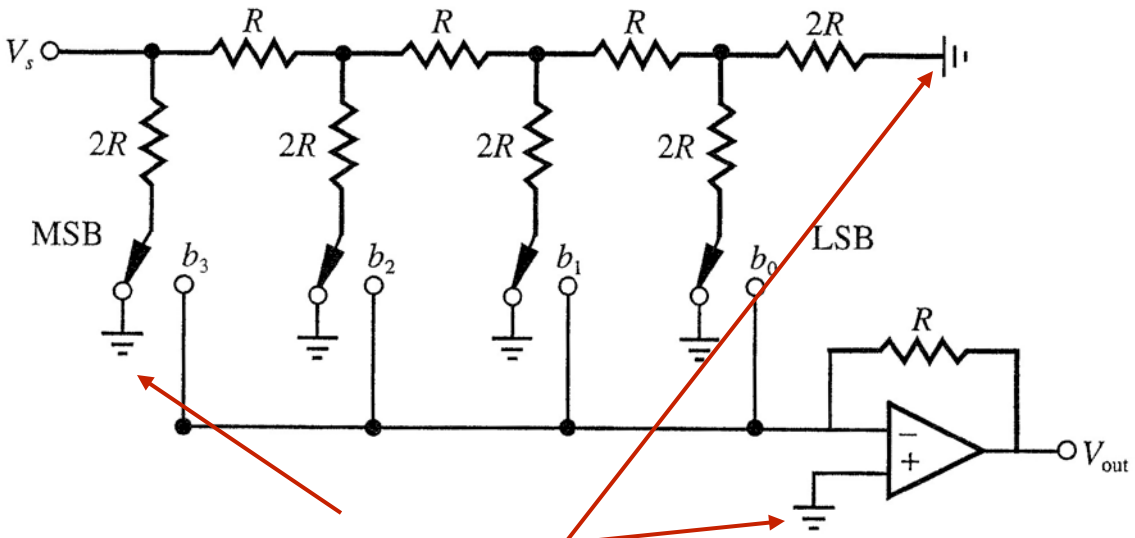
- A Test!

UCCKPH = 1
UCCKPL = 0
UCMSB = 1
UC7BIT= 0



Digital to Analog Converters

- Resistor Ladder Network
  - Simplest type of DAC
  - R-2R configuration requires only two precision resistor values

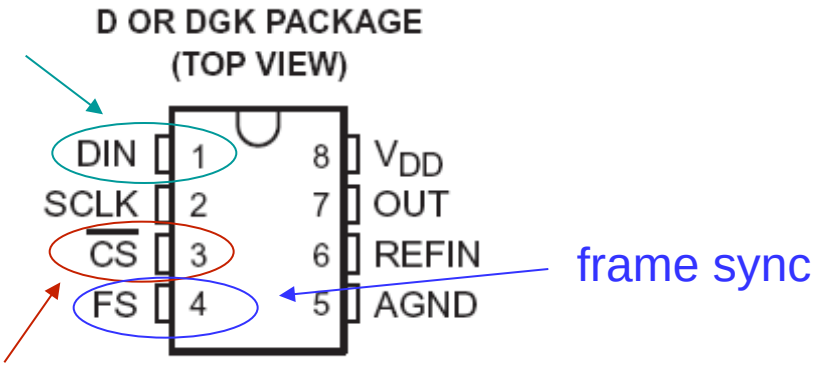


Bipolar operation possible by replacing all ground connections with -V_s connections.

TLV5606

TLV5606
2.7-V TO 5.5-V LOW POWER 10-BIT DIGITAL-TO-ANALOG CONVERTERS WITH POWER DOWN

data in



frame sync

chip select

The TLV5606 is a 10-bit voltage output digital-to-analog converter (DAC) with a flexible 4-wire serial interface. The 4-wire serial interface allows glueless interface to TMS320, SPI, QSPI, and Microwire serial ports. The TLV5606 is programmed with a 16-bit serial string containing 4 control and 10 data bits. Developed for a wide range of supply voltages, the TLV5606 can operate from 2.7 V to 5.5 V.

TLV5606 Timing

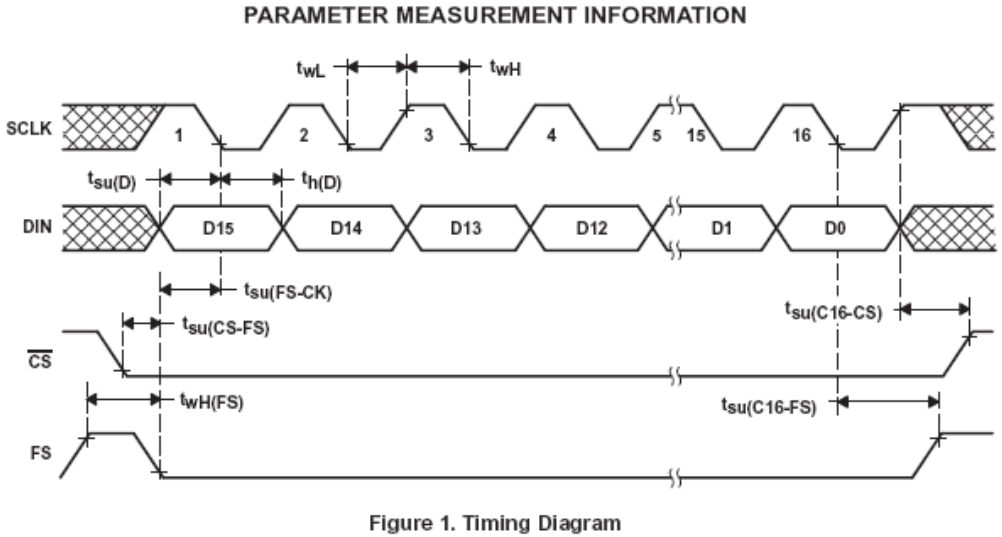


Figure 1. Timing Diagram

digital input timing requirements

		MIN	NOM	MAX	UNIT
$t_{su(CS-FS)}$	Setup time, $\overline{CS}$ low before FS↓	10			ns
$t_{su(FS-CK)}$	Setup time, FS low before first negative SCLK edge	8			ns
$t_{su(C16-FS)}$	Setup time, sixteenth negative edge after FS low on which bit D0 is sampled before rising edge of FS	10			ns
$t_{su(C16-CS)}$	Setup time, sixteenth positive SCLK edge (first positive after D0 is sampled) before $\overline{CS}$ rising edge. If FS is used instead of the sixteenth positive edge to update the DAC, then the setup time is between the FS rising edge and $\overline{CS}$ rising edge.	10			ns
t_{wH}	Pulse duration, SCLK high	25			ns
t_{wL}	Pulse duration, SCLK low	25			ns
$t_{su(D)}$	Setup time, data ready before SCLK falling edge	8			ns
$t_{h(D)}$	Hold time, data held valid after SCLK falling edge	5			ns
$t_{wH(FS)}$	Pulse duration, FS high	20			ns

