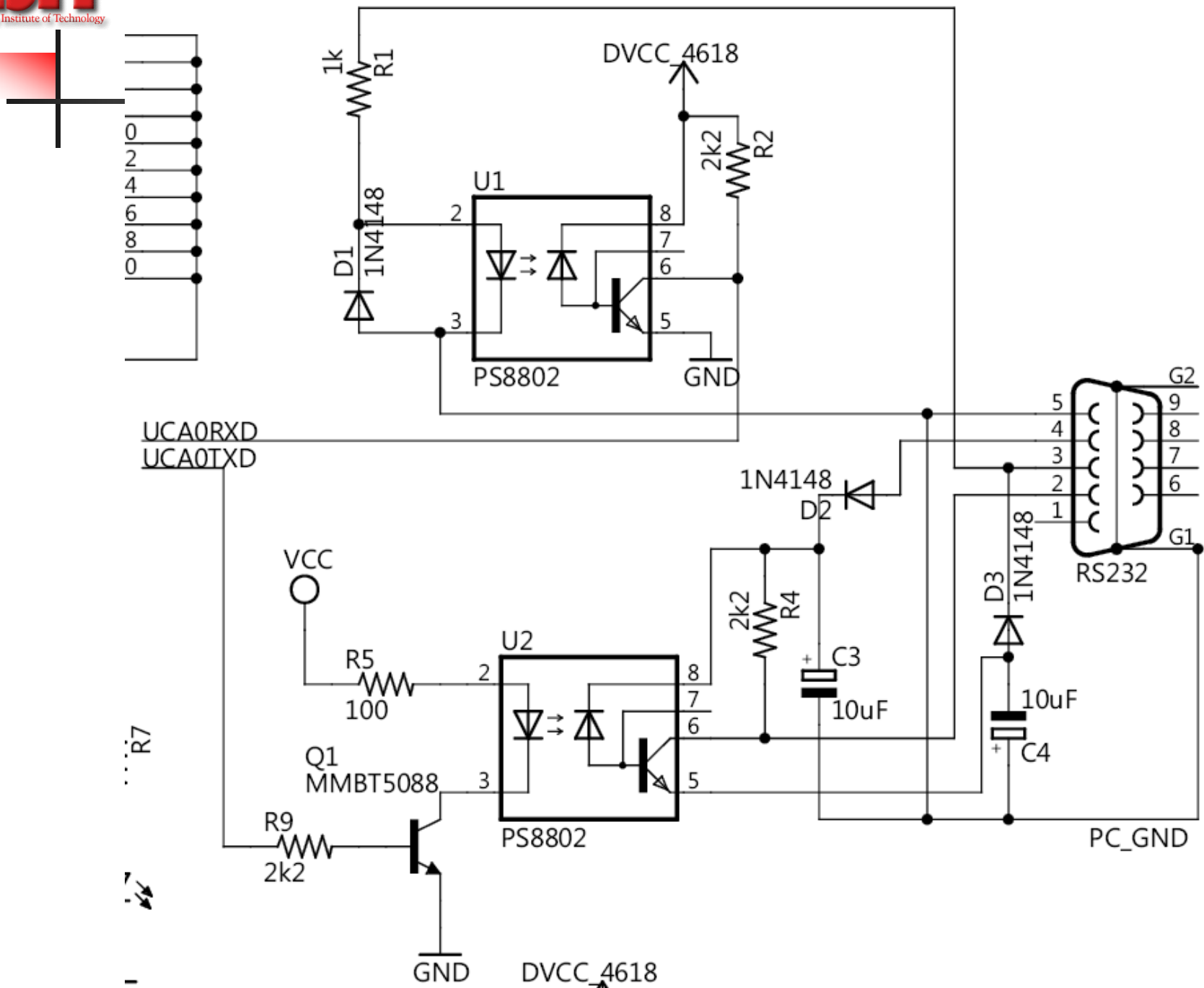




# Serial Communications

---

# Isolated RS232 Communication



# Ports and Pins

P2.4/ UCA0TXD	75	UCA0TXD
P2.5/UCA0RXD	74	UCA0RXD
P4.7	46	P4.7/S34/UCA0RXD
P4.6	47	P4.6/S35/UCA0TXD

# Universal Serial Communications Interface

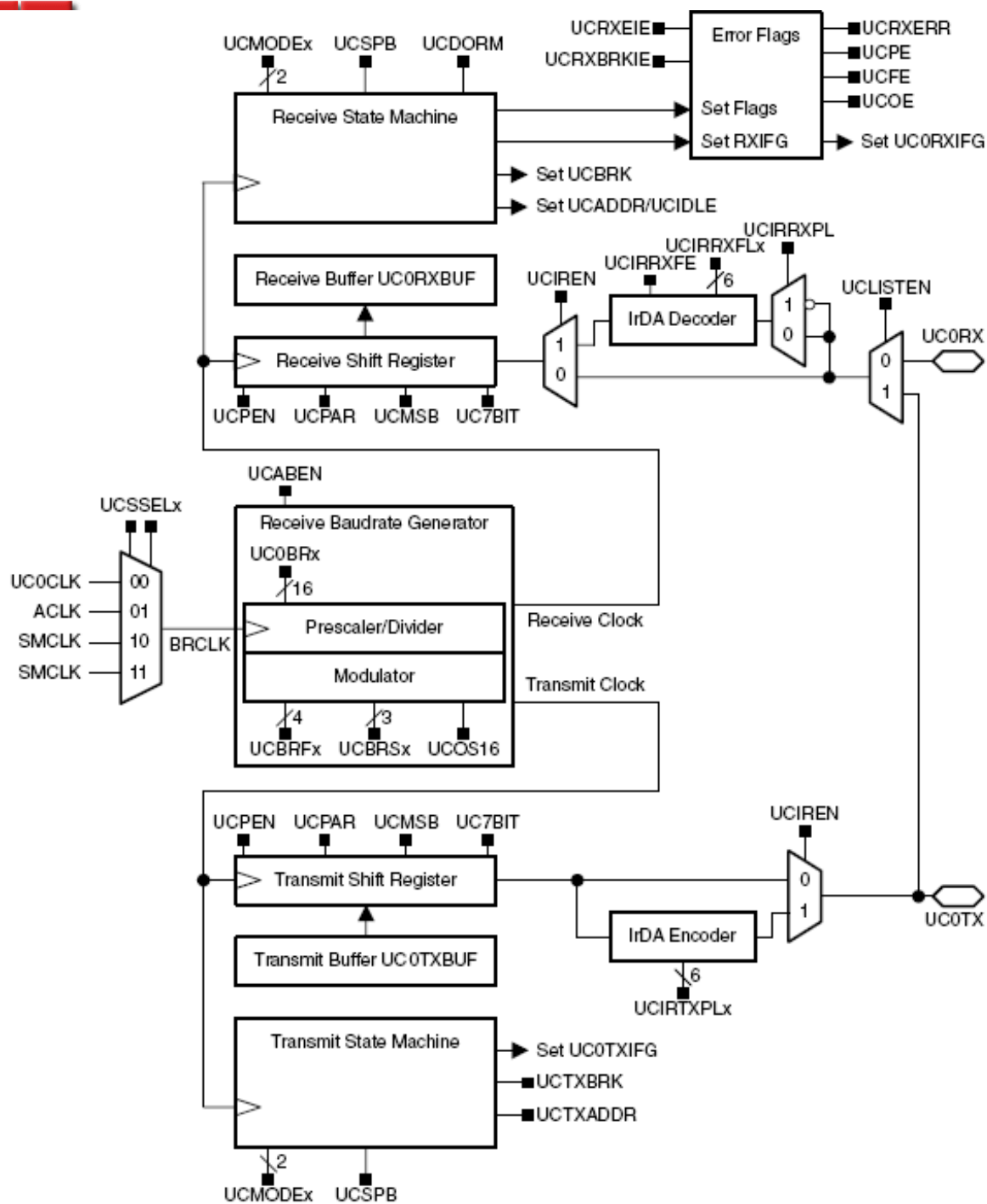
- USCI – UART Mode
- USCI\_Ax modules support several modes
  - UART, IrDA (pulse shaping), LIN, SPI
- Communicate via 2 pins
  - UCAxRXD
  - UCAxTXD
- Features
  - 7- or 8- bit data with odd, even, or non-parity
  - Independent RX and TC registers and buffers
  - RX and TX MSB or LSB first
  - Programmable baud rate
  - Status flags
  - Independent interrupt capability for RX and TX

# Initializing and RX/TXing

- USCI Initialization:
  - reset by a PUC (Power Up Clear) or
  - setting the UCSWRST (Software Reset Enable) bit.
  
- Enabling USCI module for RX or TX:
  - clear the UCSWRST bit
  - baud rate generator is in a ready state but is not clocked producing clocks
  
- USCI Receive Enable:
  - Enable USCI module
  - If valid start bit detected, character will be received into UCAxRXBUF
  - (Your job – get it out quick!)
  
- USCI Transmit Enable:
  - Enable USCI module
  - Write data to UCAxTXBUF
  - Data xmission continues until UCAxTXBUF clear
  - (Your job – make sure next character ready and stored into UCAxTXBUF quick!)

# Baud Rate

- Two ranges
  - Low frequency mode: 32768 Hz xtal for 9600 baud
  - Saves power
  - Max rate is 1/3 UART source clock frequency BRCLK
  - One prescaler/divider
  - One modulator
- Oversampling
  - Supports higher rates
- Baud (signaling events per second)
  - NOT Bits/sec, which is often faster
  - (one symbol may require several bits)



# Registers

---

- UCA0CTL1: USCI\_A1 Control Register
- UCA0BR0: Baud rate control register 0 - (prescaler)
- UCA0BR1: Baud rate control register 0 - (prescaler)
- UCA0MCTL: Modulation control register - (modulator)
- UCA0RXBUF: Receive buffer register
- UCA0TXBUF: Transmit buffer register
- UCA0STAT: Status register

# Baud Rate

- Setting baud rate

- Division factor, N, often non-integer  $N = \frac{f_{BRCLK}}{\text{Baudrate}}$
- Integer part  $UCBRx = \text{INT}(N)$
- Fractional part  $UCBRSx = \text{round}((N - \text{INT}(N)) \times 8)$

- Desired: 9600 baud

- $N = 32768/9600 = 3.41333333$
- $USBRx = \text{INT}(3.413333) = 3$
- $UCBRSx = \text{round}((3.41333333 - 3) * 8) = 3$

**Therefore,**

- prescaler = 3 (into UCxBR0)
- modulator = 3 (into UCxMCTL)

- 2400 baud?

# Values for Common Baud Rates

*Table 19–4. Commonly Used Baud Rates, Settings, and Errors, UCOS16 = 0*

BRCLK Frequency [Hz]	Baud Rate [Baud]	UCBRx	UCBRSx	UCBRFx	Max TX Error [%]	Max RX Error [%]		
32,768	1200	27	2	0	-2.8	1.4	-5.9	2.0
32,768	2400	13	6	0	-4.8	6.0	-9.7	8.3
32,768	4800	6	7	0	-12.1	5.7	-13.4	19.0
32,768	9600	3	3	0	-21.1	15.2	-44.3	21.3
1,048,576	9600	109	2	0	-0.2	0.7	-1.0	0.8
1,048,576	19200	54	5	0	-1.1	1.0	-1.5	2.5
1,048,576	38400	27	2	0	-2.8	1.4	-5.9	2.0
1,048,576	56000	18	6	0	-3.9	1.1	-4.6	5.7
1,048,576	115200	9	1	0	-1.1	10.7	-11.5	11.3
1,048,576	128000	8	1	0	-8.9	7.5	-13.8	14.8
1,048,576	256000	4	1	0	-2.3	25.4	-13.4	38.8

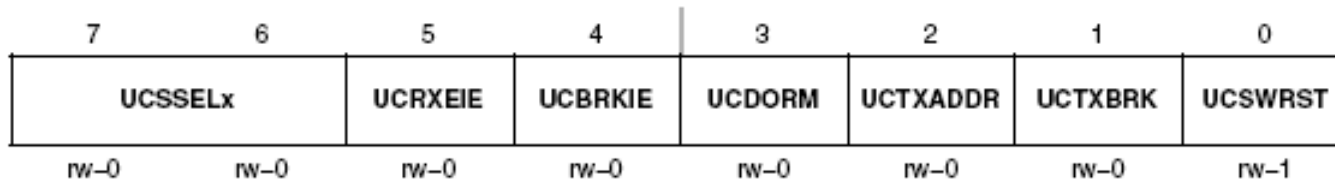
# mcp430xG46x\_uscia0\_duplex\_9600.c

```

P1OUT = 0x000;           // P1.0/1 setup for LED output
P1DIR |= 0x003;         //
P4SEL |= 0x0C0;         // P4.7,6 UART option select
UCA0CTL1 |= UCSSEL_1;   // CLK = A0CLK
UCA0BR0 = 0x03;         // 32k/9600 - 3.41
UCA0BR1 = 0x00;         //
UCA0MCTL = 0x06;        // Modulation
UCA0CTL1 &= ~UCSWRST;   // **Initialize USCI state machine**
IE2 |= UCA0RXIE+UCA0TXIE; // Enable USCI_A0 TX/RX interrupt

```

## UCAxCTL1, USCI\_Ax Control Register 1



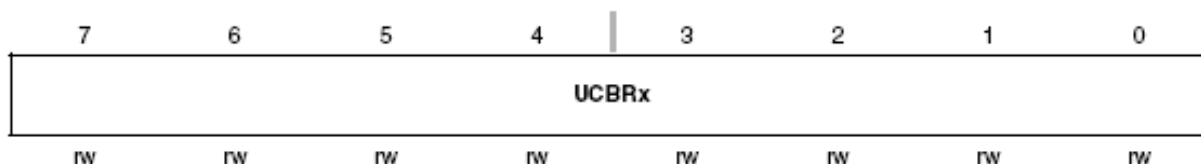
UCSSELx    Bits    USCI clock source select. These bits select the BRCLK source clock.

7-6	00	UCLK
	01	ACLK
	10	SMCLK
	11	SMCLK

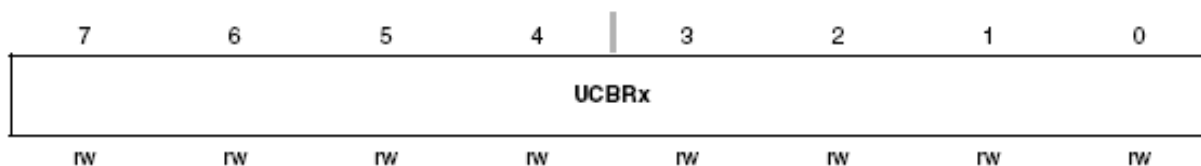
**UCSSEL\_1: 0x40**

# Baud Rate

UCAxBR0, USCI\_Ax Baud Rate Control Register 0



UCAxBR1, USCI\_Ax Baud Rate Control Register 1



UCBRx

Clock prescaler setting of the Baud rate generator. The 16-bit value of (UCAxBR0 + UCAxBR1 × 256) forms the prescaler value.

UCA0BR0 = 0x03;  
UCA0BR1 = 0x00;

**prescaler = 3 ( into UCAxBR0)**

# Modulation (Baud Rate Stuff)

UCAxMCTL, USCI\_Ax Modulation Control Register



<b>UCBRFx</b>	Bits 7-4	First modulation stage select. These bits determine the modulation pattern for BITCLK16 when UCOS16 = 1. Ignored with UCOS16 = 0. Table 19-3 shows the modulation pattern.
<b>UCBRSx</b>	Bits 3-1	Second modulation stage select. These bits determine the modulation pattern for BITCLK. Table 19-2 shows the modulation pattern.
<b>UCOS16</b>	Bit 0	Oversampling mode enabled 0 Disabled 1 Enabled

UCA0MCTL = 0x06;

**modulator = 3 (into UCAxMCTL)**

```
UCA0CTL1 &= ~UCSWRST;  
IE2 |= UCA0RXIE+UCA0TXIE;
```

```
// **Initialize USCI state machine**  
// Enable USCI_A0 TX/RX interrupt
```

```
#define USCWRST      0x01           // USCI Software Reset  
#define IE2         0x0001        // Interrupt Enable 2  
#define UCA0RXIE    (0x01)  
#define UCA0TXIE    (0x02)
```

# mcp430xG46x\_uscia0\_uart\_9600.c

```
// Echo back RXed character, confirm TX buffer is ready first
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCIA0RX_ISR (void)
{
    while(!(IFG2&UCA0TXIFG));
    UCA0TXBUF = UCA0RXBUF;           // TX -> RXed character
}
```

- When a char received, it is put into transmit buffer, which will cause it to immediately be sent.
- UCA0TXIFG is set when TX buffer clear

# RX, TX Interrupts

```
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI_A0_rx (void)
{
    P1OUT = UCA0RXBUF;           // RXBUF1 to TXBUF1
}

#pragma vector=USCIAB0TX_VECTOR
__interrupt void USCI_A0_Tx (void)
{
    volatile unsigned int i;
    unsigned int delay;

    for(delay=240;delay>0;delay--); // Add small gap between TX'ed bytes
    i = P1IN;
    i = i >> 4;
    UCA0TXBUF = i;               // Transmit character
}
```

- RX: When char received, outputs to Port 1
- TX: Reads Port 1 and sends it 1 nibble at a time