THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

CPE 323: MSP430 Serial Communication

Aleksandar Milenkovic

Electrical and Computer Engineering
The University of Alabama in Huntsville

milenka@ece.uah.edu

http://www.ece.uah.edu/~milenka

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

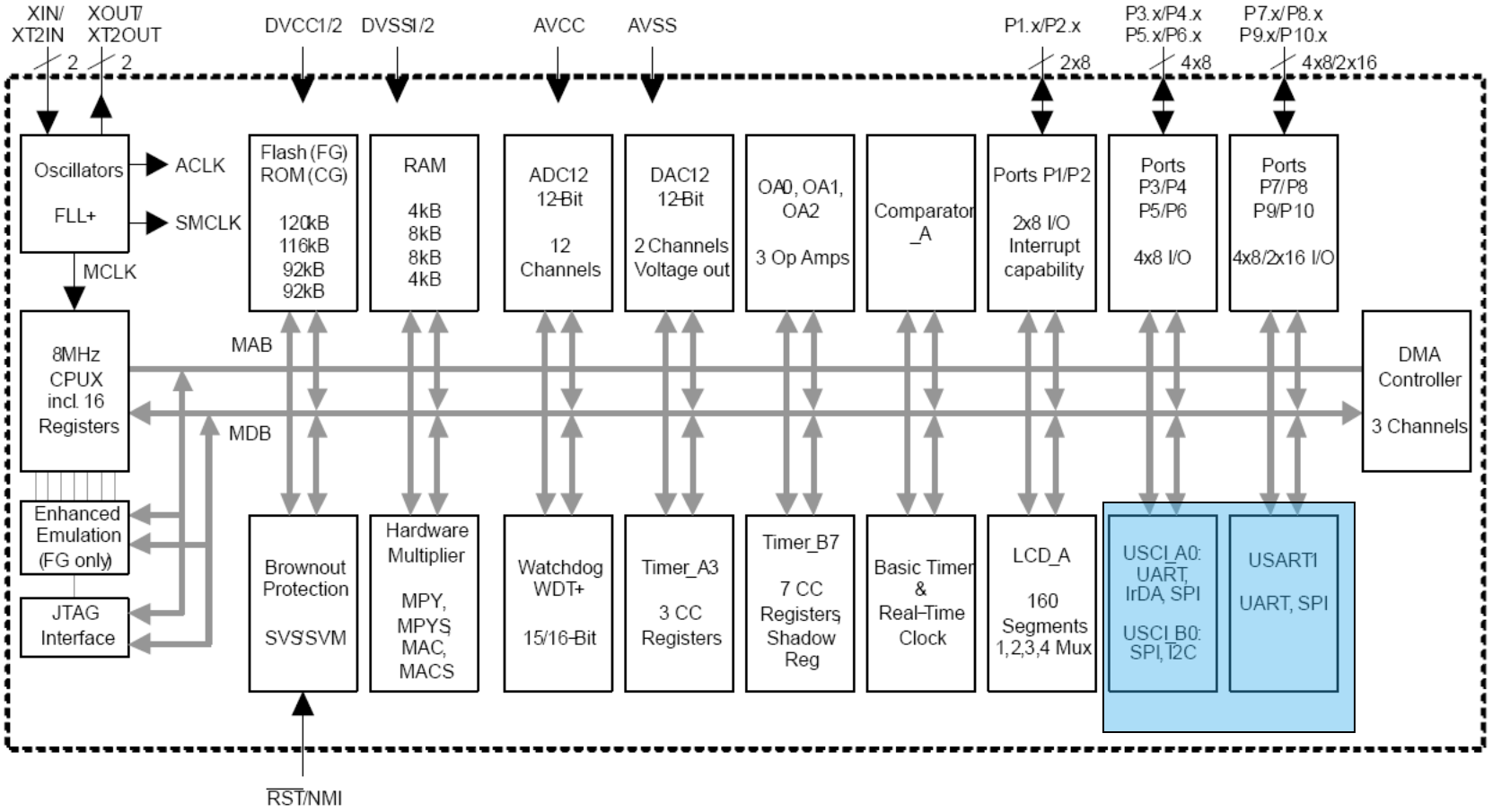
Outline

- Introduction
- UART: Universal Asynchronous Receiver/Transmitter
- UART Demo
- RS232
- SPI
- I^2C

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

MSP430xG461x Microcontroller



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Communication in Embedded Systems

- Sense => Process => Store/Memorize => Communicate => Act
- Internet-of-Things (connected things)
  - Transportation, Home, Infrastructure in Cities, …
- How to communicate with the development workstation?
  - Inter board comm. (several feet)
- How to communicate with other components on-board?
  - Inter-chip comm. (several millimeters)

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Inter-chip Communication

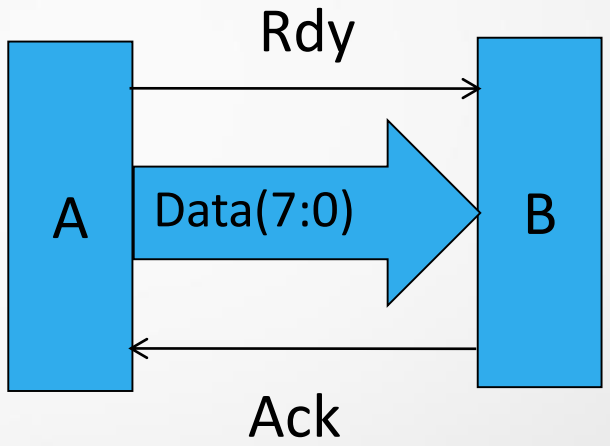
- Parallel vs. Serial
  - Metrics: Cost, Bandwidth, Latency
- Types of communication
  - Duplex
  - Half-duplex
  - Simplex
- Handshaking
  - RDY (ready data)
  - ACK (acknowledge data)

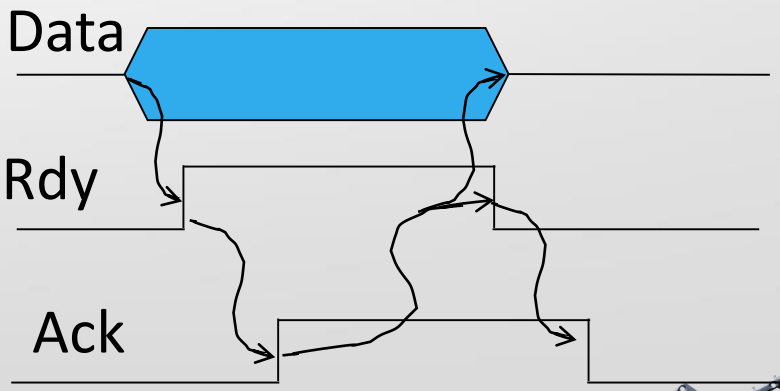
LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Handshaking Example

- Simplex communication (from A to B)
- Handshaking in time
  - A: set data
  - A: set Rdy
  - B: get data
  - B: set Ack
  - A: de-assert Rdy, Data
  - B: de-assert Ack





LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Serial Communication

- Common types of serial communication
  - Asynchronous serial communication (UART – Universal Asynchronous Receiver/Transmitter)
  - Serial Peripheral Interface (SPI)
  - Inter-integrated circuit (I^2C) bus

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

MSP430 Comm. Peripherals

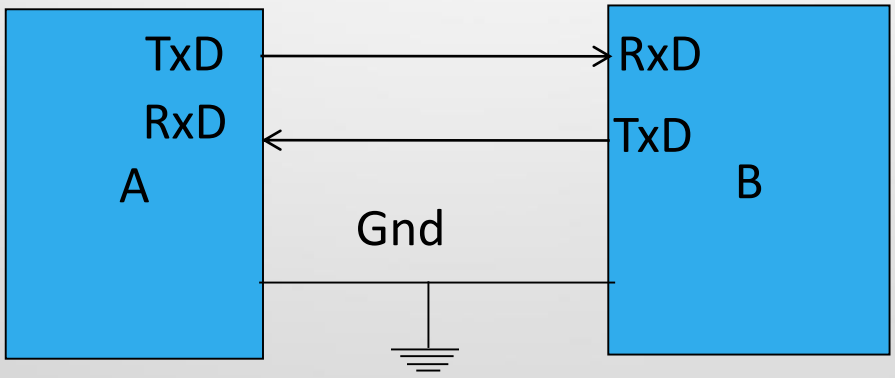
- Universal Serial Interface (USI)
  - Small module (F20x2, F20x3 devices)
  - SPI and I^2C (needs some help from SW)
- Universal Serial Communication Interface (USCI)
  - Contains two channels USCI_A, USCI_B
  - USCI_A: UART, IrDA, SPI
  - USCI_B: SPI, I^2C
- Universal Synchronous/Asynchronous Receiver/Transmitter (USART)
  - UART, SPI (and I^2C in some devices)

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Asynchronous Serial Communication

- Popular in embedded systems (uart)
- Single wire for each direction (RxD, TxD) and common ground (Gnd)
- Duplex link (data can be sent simultaneously in both directions)



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Asynchronous Serial Interface

- Asynchronous
  - Transmitted and received data are not synchronized over any extended period of time
  - No synchronization between receiver and transmitter clocks
- Serial
  - Usually character oriented
  - Data stream divided into individual bits at the transmitter side
  - Individual bits are grouped into characters at the receiving side
- Information is usually transmitted as ASCII-encoded characters
  - 7 or 8 bits of information plus control bits

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

How does it work?
Transmitter wants to send a byte to Receiver

- 1. Microcontroller (T side): Writes the byte into the Transmit Data Buffer (visible to the programmer) of the Serial I/O interface
- 2. Serial I/O interface (T side): Loads the data from the Transmit Data Buffer into a shift register (parallel-in/serial-out); The data are shifted out of this register using $T_{BITCLOCK}$ clock signal
- 3. Serial I/O interface (R side): The data from a single transmission line that goes from device T to device R are shifted in the serial-in/parallel out shift register. The data is shifted on every $R_{BITCLOCK}$ cycle. It is assumed that $T_{BITCLOCK} = R_{BITCLOCK}$
- 4. Serial I/O interface (R side): When all 8 bits are received in the shift register, the data is transferred to the Receive Data Buffer in parallel.
- 5. Microcontroller (R side): Reads the byte from the Receive Data Buffer (visible to the programmer) of the Serial I/O interface.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Format of Data

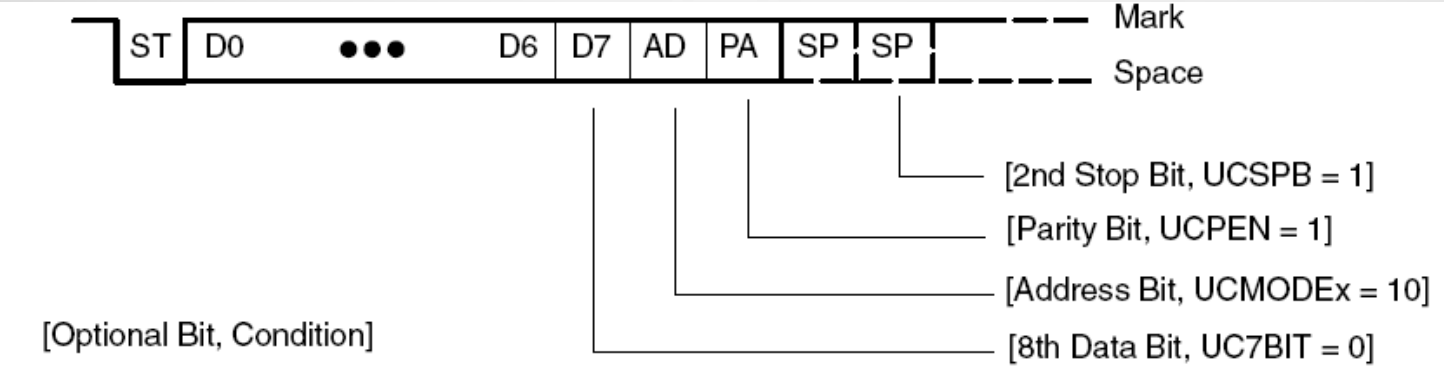
- Data are sent in relatively short frames (typically 1 byte)
- Logic levels: Logic one – MARK, Logic zero – SPACE
- If nothing to transmit, the transmitter holds the MARK level
- Start bit: to mark the beginning of a new character, the line changes from the mark to the space level for one bit time. This synchronizes the transmitter and receiver. When the receiver detects the start bit, it knows to start clocking in the serial data bits.
- Data bits: any number of bits can be theoretically sent. In practice, it is between 5 and 8 bits.
- Parity bit: 7 bits are needed to encode a character. Most UARTS allow 8 bits to be sent. The parity bit is added to the data to make the total of ones odd (ODD parity) or even (EVEN parity). The parity bit is used to detect possible errors in the data stream.
- Stop bit: The stop bit is added at the end of the data bits. This gives at least one bit time period between successive characters. Some systems require 2 or more stop bits.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Format of Data

- Line idles high (MARK)
- Character format
  - ST: Start bit (low, SPACE)
  - 7, 8 data bits (D0-D7), LSB goes first
  - SP: one stop bit (high, MARK)
- Baud rate: bits/sec



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

How does asynchronous comm. work?

- Transmitter and receiver run independently (no clock is shared)
- How do we synchronize them?
- Clock at the receiver side must run faster than the baud rate (e.g., 16x)
- Basic procedure for receiving a frame
  - 1. When falling edge of ST bit is detected, the receiver clock samples the input line half-way through the bit (be sure that is true ST bit)
  - 2. Sample the input after a half bit period to confirm the valid start bit
  - 3. Sample the input after the next bit period (lsb bit)
  - 4. Repeat this until all 8 bits are received
  - 5. Wait a further bit period and check that the input is high as expected (SP bit). Indicate a framing error if no valid stop bit is detected.

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

MSP430 USCI in UART mode

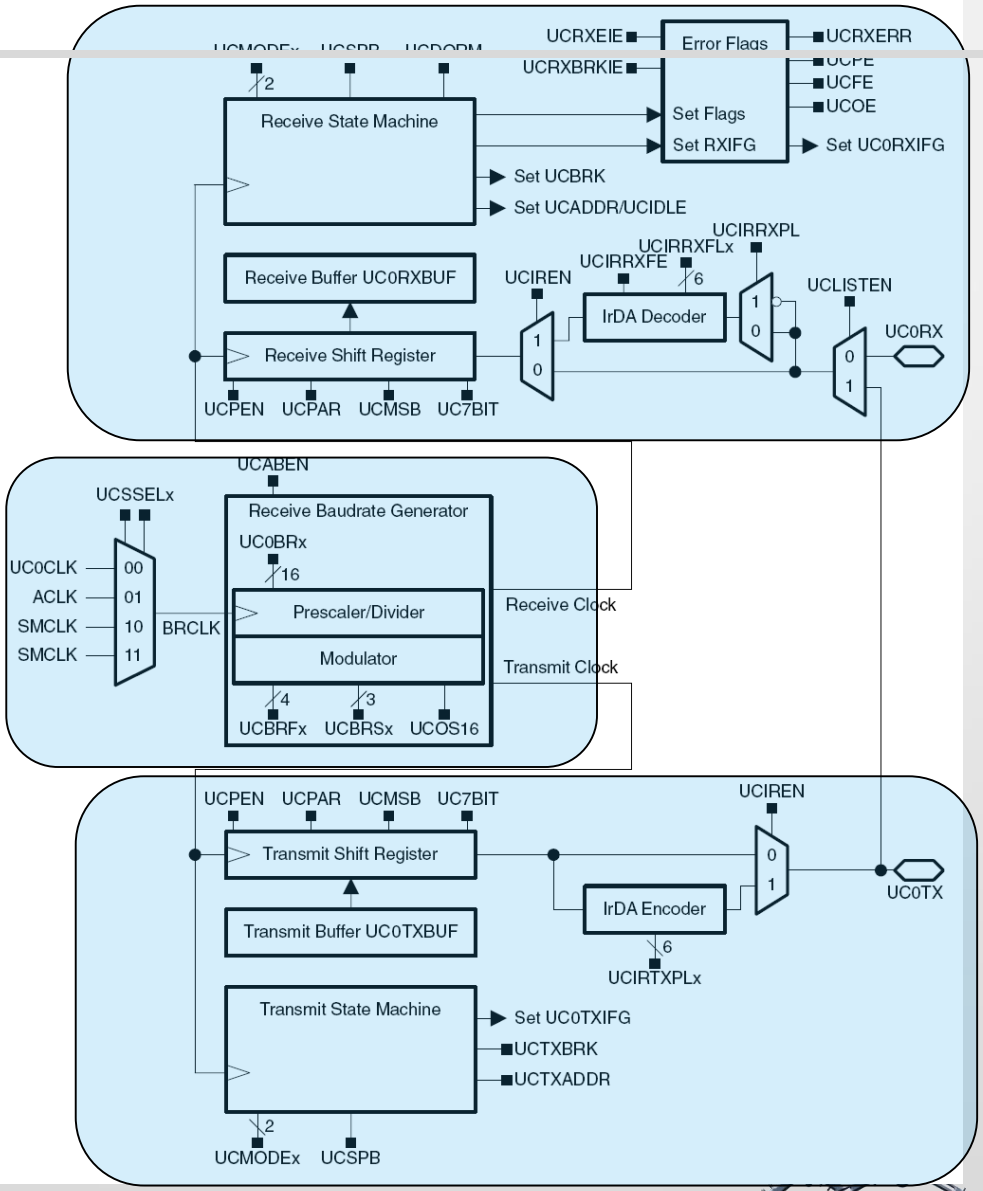
- 7- or 8-bit data with odd, even, or non-parity
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- LSB-first or MSB-first data transmit and receive
- Built-in idle-line and address-bit communication protocols for multiprocessor systems
- Receiver start-edge detection for auto-wake up from LPMx modes
- Programmable baud rate with modulation for fractional baud rate support
- Status flags for error detection and suppression
- *Status flags for address detection*
- Independent interrupt capability for receive and transmit

LaCASA

USCI Block

- Receive data buffer: UC0RXBUF
- Transmit data buffer: UCA0TXBUF
- Baud rate registers:
  UC0BR0, UC0BR1, Modulator

Figure 19–1. USCI_Ax Block Diagram: UART Mode (UCSYNC = 0)



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

USCI Initialization Process

- 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

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

USCI Registers: UART mode

Table 19–6. USCI_A0 Control and Status Registers

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_A0 Auto Baud control register	UCA0ABCTL	Read/write	05Dh	Reset with PUC
USCI_A0 IrDA Transmit control register	UCA0IRTCTL	Read/write	05Eh	Reset with PUC
USCI_A0 IrDA Receive control register	UCA0IRRCTL	Read/write	05Fh	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

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Receive Error Conditions

Table 19–1. Receive Error Conditions

Error Condition	Error Flag	Description
Framing error	UCFE	A framing error occurs when a low stop bit is detected. When two stop bits are used, both stop bits are checked for framing error. When a framing error is detected, the UCFE bit is set.
Parity error	UCPE	A parity error is a mismatch between the number of 1s in a character and the value of the parity bit. When an address bit is included in the character, it is included in the parity calculation. When a parity error is detected, the UCPE bit is set.
Receive overrun	UCOE	An overrun error occurs when a character is loaded into UCAxRXBUF before the prior character has been read. When an overrun occurs, the UCOE bit is set.
Break condition	UCBRK	When not using automatic baud rate detection, a break is detected when all data, parity, and stop bits are low. When a break condition is detected, the UCBRK bit is set. A break condition can also set the interrupt flag UCAxRXIFG if the break interrupt enable UCBRKIE bit is set.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

USCI_A Configuration

- UXA0CTL0
  - UCSYNC = 0 (async. mode)
  - UCMODExx
    - 00 – standard UART mode
    - 01 or 10 – multiprocessor modes
    - 11- automatic baud rate detection (LIN)
- Baud rate setting
  - BRCLK – the input clock (ACLK, SMCLK, UCLK)
  - BITCLK – $f_{BITCLK} = f_{Baud}$
  - BITCLK16 – $f_{BITCLK16} = 16*f_{BITCLK}$
- Challenge: BITCLK is rarely close to being perfect factor of BRCLK

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

USCI_A Configuration: Baud Rate

- Oversampling mode, UCOS16 = 1
  - BRCLK is first divided to give BITCLK16, which is further divided by a factor of 16
- Low frequency mode, UCOS16 = 0
  - BRCLK if used directly as the sampling clock and is divided to give BITCLK
- BRCLK division
  - UCA0BR0 and UCA0BR1: provides main divider
  - UCBRFx: modulates the divider that gives BITCLK16 in oversampling mode
  - UCBRSx: modulates the divider that gives BITCLK

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Oversampling Mode (UCOS16=1)

- Assume: f_{baud} = 9600 Hz, $f_{BRCLK} = 2^{20}$ Hz
  - $f_{BIT16CLK} = 16 * f_{baud}$ = 153.6 KHz
- $N = f_{BRCLK} / f_{baud} = 109.22 > 16$
- N/16 = 109.22/16 = 6.83
- UCBRx = INT(N/16) = 6
- UCBRFx= round ((N/16 – INT(N/16))*16) = 13
- => 13 BITCLK16 cycles will have 7 (or N+1 in general) BRCLK clocks and 3 BITCLK16 cycles will have 6 (or N in general) BRCLK clocks

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Commonly Used Baud Rates (UCOS16=1)

Table 19–5.Commonly Used Baud Rates, Settings, and Errors, UCOS16 = 1

BRCLK frequency [Hz]	Baud Rate [Baud]	UCBRx	UCBRSx	UCBRFx	Max. TX Error [%]		Max. RX Error [%]	
1,000,000	9600	6	0	8	–1.8	0	–2.2	0.4
1,000,000	19200	3	0	4	–1.8	0	–2.6	0.9
1,048,576	9600	6	0	13	–2.3	0	–2.2	0.8
1,048,576	19200	3	1	6	–4.6	3.2	–5.0	4.7
4,000,000	9600	26	0	1	0	0.9	0	1.1
4,000,000	19200	13	0	0	–1.8	0	–1.9	0.2
4,000,000	38400	6	0	8	–1.8	0	–2.2	0.4
4,000,000	57600	4	5	3	–3.5	3.2	–1.8	6.4
4,000,000	115200	2	3	2	–2.1	4.8	–2.5	7.3
8,000,000	9600	52	0	1	–0.4	0	–0.4	0.1
8,000,000	19200	26	0	1	0	0.9	0	1.1
8,000,000	38400	13	0	0	–1.8	0	–1.9	0.2
8,000,000	57600	8	0	11	0	0.88	0	1.6
8,000,000	115200	4	5	3	–3.5	3.2	–1.8	6.4
8,000,000	230400	2	3	2	–2.1	4.8	–2.5	7.3
12,000,000	9600	78	0	2	0	0	–0.05	0.05
12,000,000	19200	39	0	1	0	0	0	0.2
12,000,000	38400	19	0	8	–1.8	0	–1.8	0.1
12,000,000	57600	13	0	0	–1.8	0	–1.9	0.2
12,000,000	115200	6	0	8	–1.8	0	–2.2	0.4
12,000,000	230400	3	0	4	–1.8	0	–2.6	0.9

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Low-frequency Mode (UCOS16=0)

- Use when $f_{BRCLK} < 16*f_{baud}$
- E.g., ACLK = 32,768 Hz, f_{baud} = 9,600 Hz
- $N = f_{BRCLK} / f_{baud} = 3.41$ =>
  cannot make the length of each BITCLK correct
- UCBRx = INT (N) = 3
- UCBRSx = round ((N – INT(N))*8) = round((3.41 – 3)*8) = 3
- 5 bits (8 – UCBRSx) with duration 3 BRCLK (or N in general), UCBRSx bits with duration of 4 (or N+1 in general) BRCLK (~3.4 average)

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Commonly Used Baud Rates (UCOS16=0)

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
1,000,000	**9600**	104	1	0	–0.5	0.6	–0.9	1.2
1,000,000	**19200**	52	0	0	–1.8	0	–2.6	0.9
1,000,000	**38400**	26	0	0	–1.8	0	–3.6	1.8
1,000,000	**57600**	17	3	0	–2.1	4.8	–6.8	5.8
1,000,000	**115200**	8	6	0	–7.8	6.4	–9.7	16.1
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	**57600**	18	1	0	–4.6	3.3	–6.8	6.6
1,048,576	**115200**	9	1	0	–1.1	10.7	–11.5	11.3
4,000,000	**9600**	416	6	0	–0.2	0.2	–0.2	0.4
4,000,000	**19200**	208	3	0	–0.2	0.5	–0.3	0.8
4,000,000	**38400**	104	1	0	–0.5	0.6	–0.9	1.2
4,000,000	**57600**	69	4	0	–0.6	0.8	–1.8	1.1
4,000,000	**115200**	34	6	0	–2.1	0.6	–2.5	3.1
4,000,000	**230400**	17	3	0	–2.1	4.8	–6.8	5.8
8,000,000	**9600**	833	2	0	–0.1	0	–0.2	0.1
8,000,000	**19200**	416	6	0	–0.2	0.2	–0.2	0.4
8,000,000	**38400**	208	3	0	–0.2	0.5	–0.3	0.8
8,000,000	**57600**	138	7	0	–0.7	0	–0.8	0.6
8,000,000	**115200**	69	4	0	–0.6	0.8	–1.8	1.1

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

UCAxCTL0

UCAxCTL0, USCI_Ax Control Register 0

7	6	5	4	3	2	1	0
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC=0
rw–0	rw–0	rw–0	rw–0	rw–0	rw–0	rw–0	rw–0

UCPEN	Bit 7	Parity enable 0 Parity disabled. 1 Parity enabled. Parity bit is generated (UCAxTXD) and expected (UCAxRXD). In address-bit multiprocessor mode, the address bit is included in the parity calculation.
UCPAR	Bit 6	Parity select. UCPAR is not used when parity is disabled. 0 Odd parity 1 Even parity
UCMSB	Bit 5	MSB first select. Controls the direction of the receive and transmit shift register. 0 LSB first 1 MSB first
UC7BIT	Bit 4	Character length. Selects 7-bit or 8-bit character length. 0 8-bit data 1 7-bit data
UCSPB	Bit 3	Stop bit select. Number of stop bits. 0 One stop bit 1 Two stop bits
UCMODEx	Bits 2–1	USCI mode. The UCMODEx bits select the asynchronous mode when UCSYNC = 0. 00 UART Mode. 01 Idle-Line Multiprocessor Mode. 10 Address-Bit Multiprocessor Mode. 11 UART Mode with automatic baud rate detection.
UCSYNC	Bit 0	Synchronous mode enable 0 Asynchronous mode 1 Synchronous Mode

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

UCAxCTL1

UCAxCTL1, USCI_Ax Control Register 1

7	6	5	4	3	2	1	0
UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTXBRK	UCSWRST
rw–0	rw–0	rw–0	rw–0	rw–0	rw–0	rw–0	rw–1

UCSSELx	Bits 7-6	USCI clock source select. These bits select the BRCLK source clock. 00 UCLK 01 ACLK 10 SMCLK 11 SMCLK
UCRXEIE	Bit 5	Receive erroneous-character interrupt-enable 0 Erroneous characters rejected and UCAxRXIFG is not set 1 Erroneous characters received will set UCAxRXIFG
UCBRKIE	Bit 4	Receive break character interrupt-enable 0 Received break characters do not set UCAxRXIFG. 1 Received break characters set UCAxRXIFG.
UCDORM	Bit 3	Dormant. Puts USCI into sleep mode. 0 Not dormant. All received characters will set UCAxRXIFG. 1 Dormant. Only characters that are preceded by an idle-line or with address bit set will set UCAxRXIFG. In UART mode with automatic baud rate detection only the combination of a break and synch field will set UCAxRXIFG.
UCTXADDR	Bit 2	Transmit address. Next frame to be transmitted will be marked as address depending on the selected multiprocessor mode. 0 Next frame transmitted is data 1 Next frame transmitted is an address
UCTXBRK	Bit 1	Transmit break. Transmits a break with the next write to the transmit buffer. In UART mode with automatic baud rate detection 055h must be written into UCAxTXBUF to generate the required break/synch fields. Otherwise 0h must be written into the transmit buffer. 0 Next frame transmitted is not a break 1 Next frame transmitted is a break or a break/synch
UCSWRST	Bit 0	Software reset enable 0 Disabled. USCI reset released for operation. 1 Enabled. USCI logic held in reset state.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Baud Rate Control Registers

UCAxBR0, USCI_Ax Baud Rate Control Register 0

7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

UCAxBR1, USCI_Ax Baud Rate Control Register 1

7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

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

UCAxMCTL, USCI_Ax Modulation Control Register

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw–0	rw–0	rw–0	rw–0	rw–0	rw–0	rw–0	rw–0

UCBRFx	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.
UCBRSx	Bits 3–1	Second modulation stage select. These bits determine the modulation pattern for BITCLK. Table 19–2 shows the modulation pattern.
UCOS16	Bit 0	Oversampling mode enabled 0 Disabled 1 Enabled

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Demo #1 (Echo a Character)

```
#include  <msp430xG46x.h>
void main(void)
{
  volatile unsigned int i;

  WDTCTL = WDTPW+WDTHOLD; // Stop WDT

  P2SEL |= BIT4 + BIT5; // P4.7,6 = USCI_A0 RXD/TXD
  UCA0CTL1 |= UCSSEL_2; // SMCLK
  UCA0BR0 = 0x09;       // 1MHz 115200
  UCA0BR1 = 0x00;       // 1MHz 115200
  UCA0MCTL = 0x01;      // Modulation
  UCA0CTL1 &= ~UCSWRST; // **Initialize USCI state machine**
  IE2 |= UCA0RXIE; // Enable USCI_A0 RX interrupt

  _BIS_SR(LPM0_bits + GIE); // Enter LPM0, interrupts enabled
}

//  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
}
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Interface Standards

- UART mode for on-board communication
  - Logic levels: VSS (logic 0), VCC (logic 1)
  - Devices share ground and power supply
- Communication between separate devices (e.g., dev. board and PC)
  - Need a standard interface (e.g. RS232)
  - Defines logic levels, connector types, …
- External circuits are needed to connect MSP430 to an RS-232 port (e.g. RS232 line driver MAX3222 + charge pumps that make ± 2*V_{CC})

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

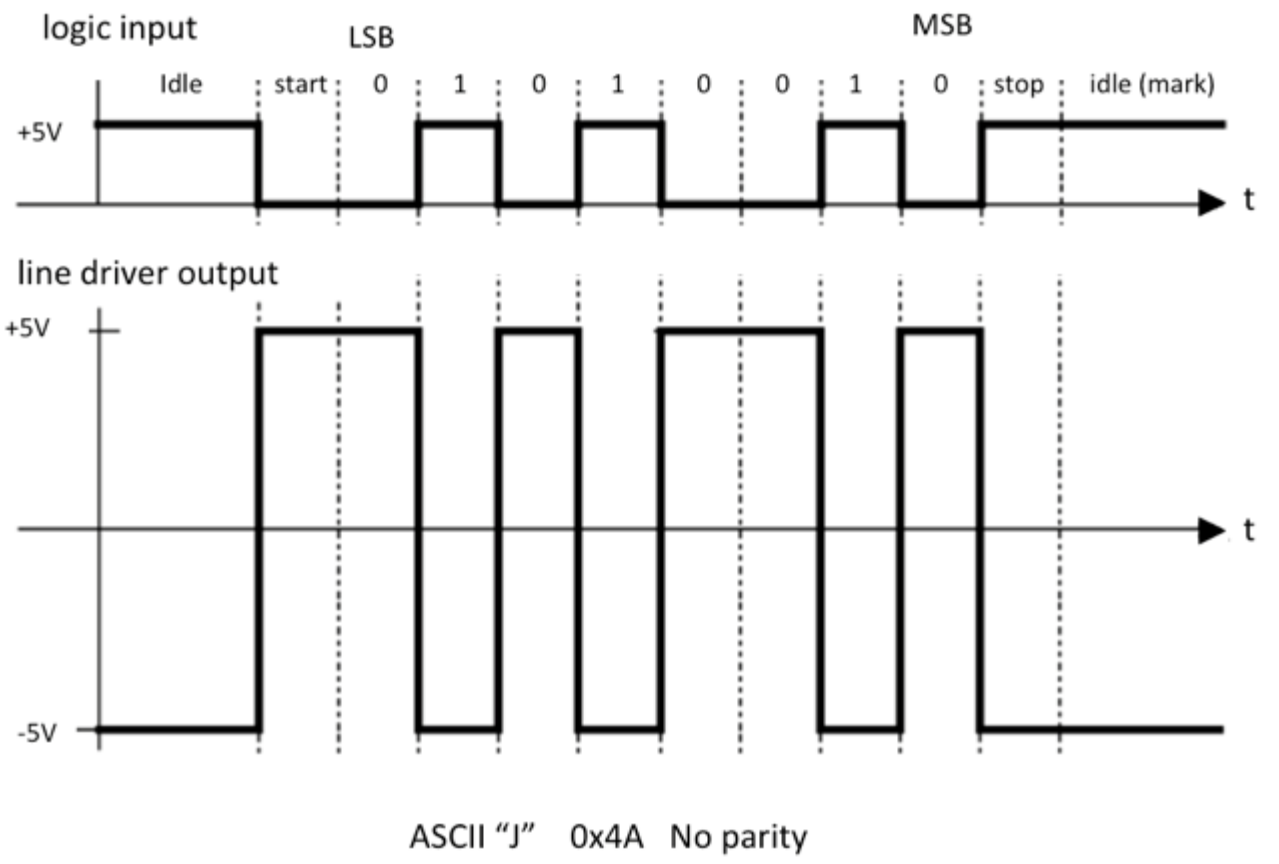
RS-232 Interface Standard

- Bi-polar:
  - +3 to +12V (ON, 0-state, or SPACE condition)
  - -3 to –12V (OFF, 1-state, or MARK condition)
- Modern computers accept 0V as MARK
- “Dead area” between –3V and 3V is designed to absorb line noise
- Originally developed as a standard for communication between computer equipment and modems
- From the point of view of this standard:
  - MODEM: data communications equipment (DCE)
  - Computer equipment: data terminal equipment (DTE)
- Therefore, RS-232C was intended for DTE-DCE links (not for DTE-DTE links, as it is frequently used now)

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

RS-232 Interface Standard



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

RS-232 Interface Standard

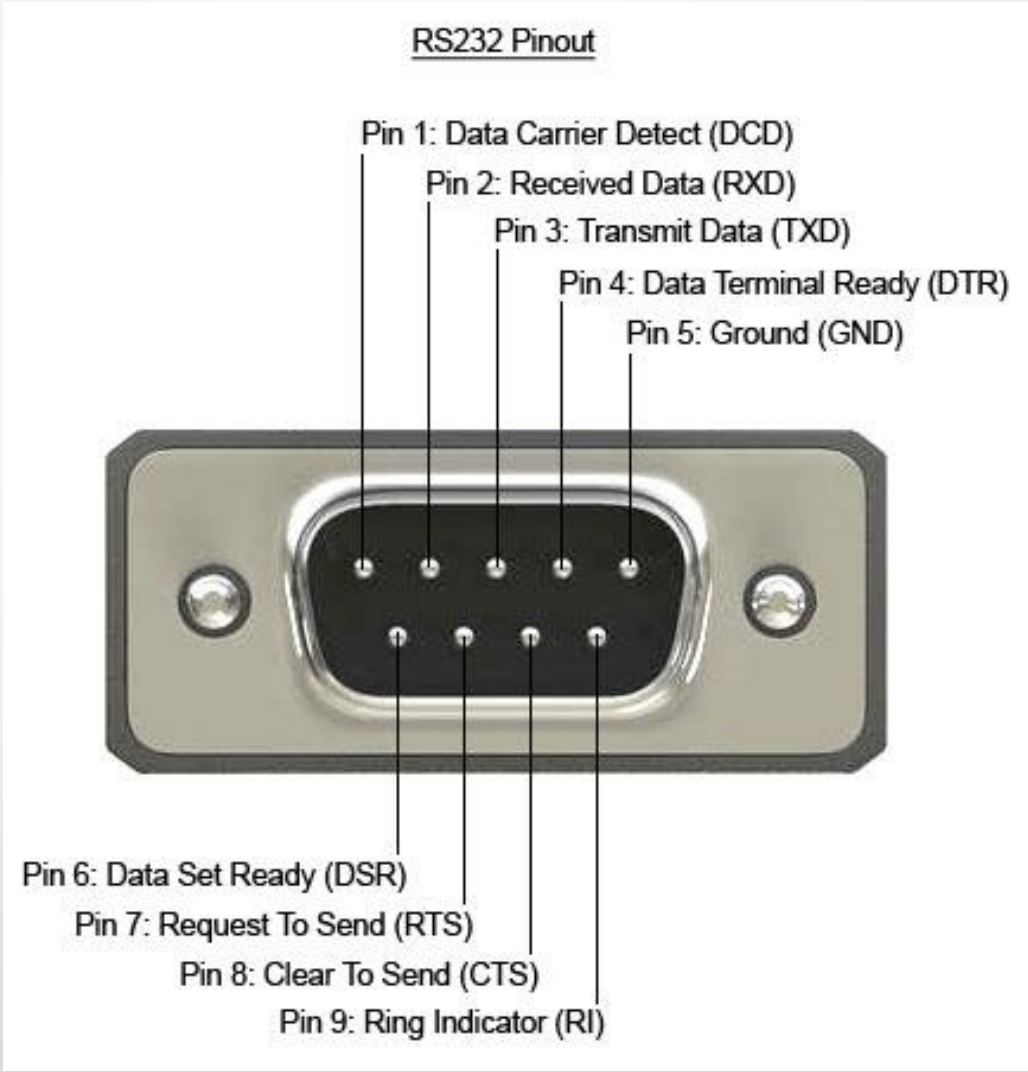
- Each manufacturer may choose to implement only a subset of functions defined by this standard
- Two widely used connectors: DB-9 and DB-25
- Three types of link
  - Simplex
  - Half-duplex
  - Full-duplex
- Basic control signals
  - RTS (Request to send): DTE indicates to the DCE that it wants to send data
  - CTS (Clear to send): DCE indicates that it is ready to receive data
  - DSR (Data set ready): indication from the DCE (i.e., the modem) that it is on
  - DTR (Data terminal ready): indication from the DTE that it is on

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

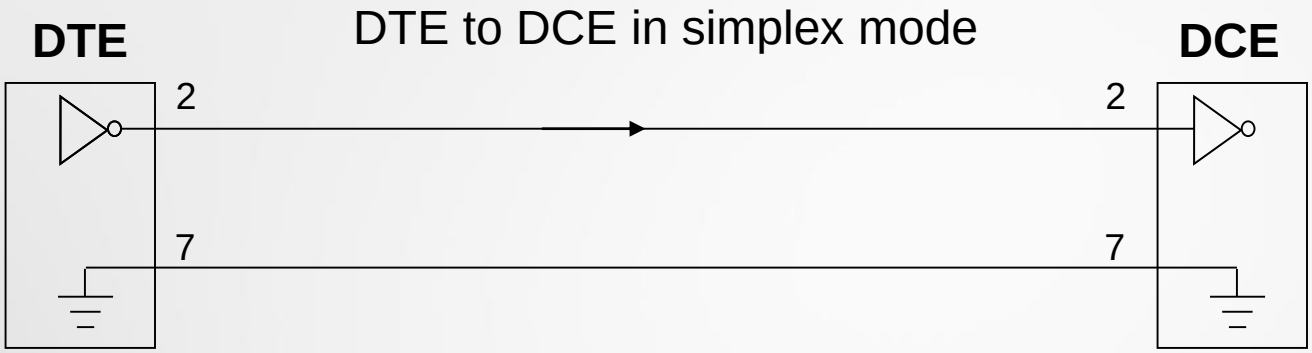
RS-232 Interface Standard

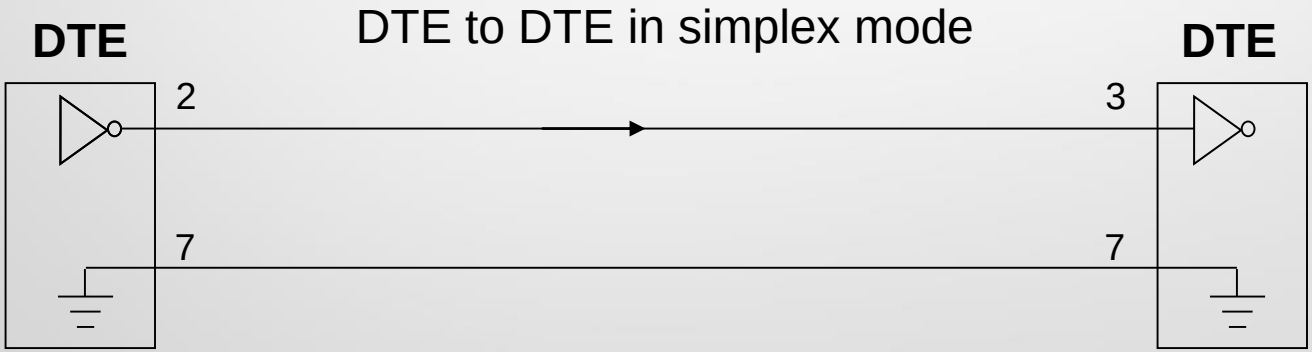
- Let's take a look at DB-9



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

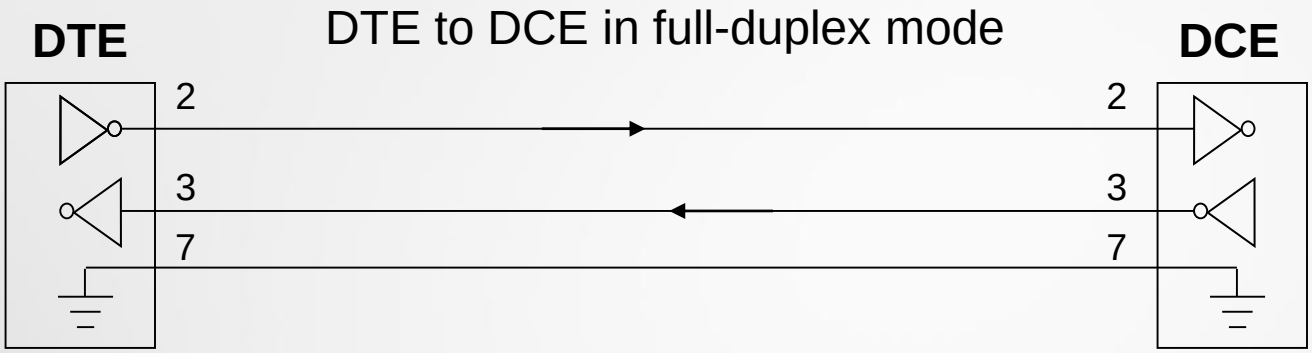
The Minimal RS-232 Function

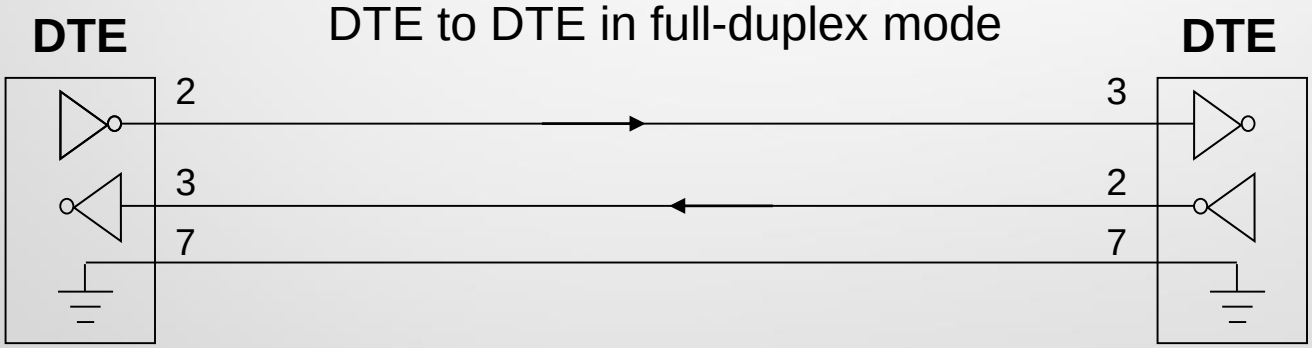




THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

The Minimal RS-232 Function

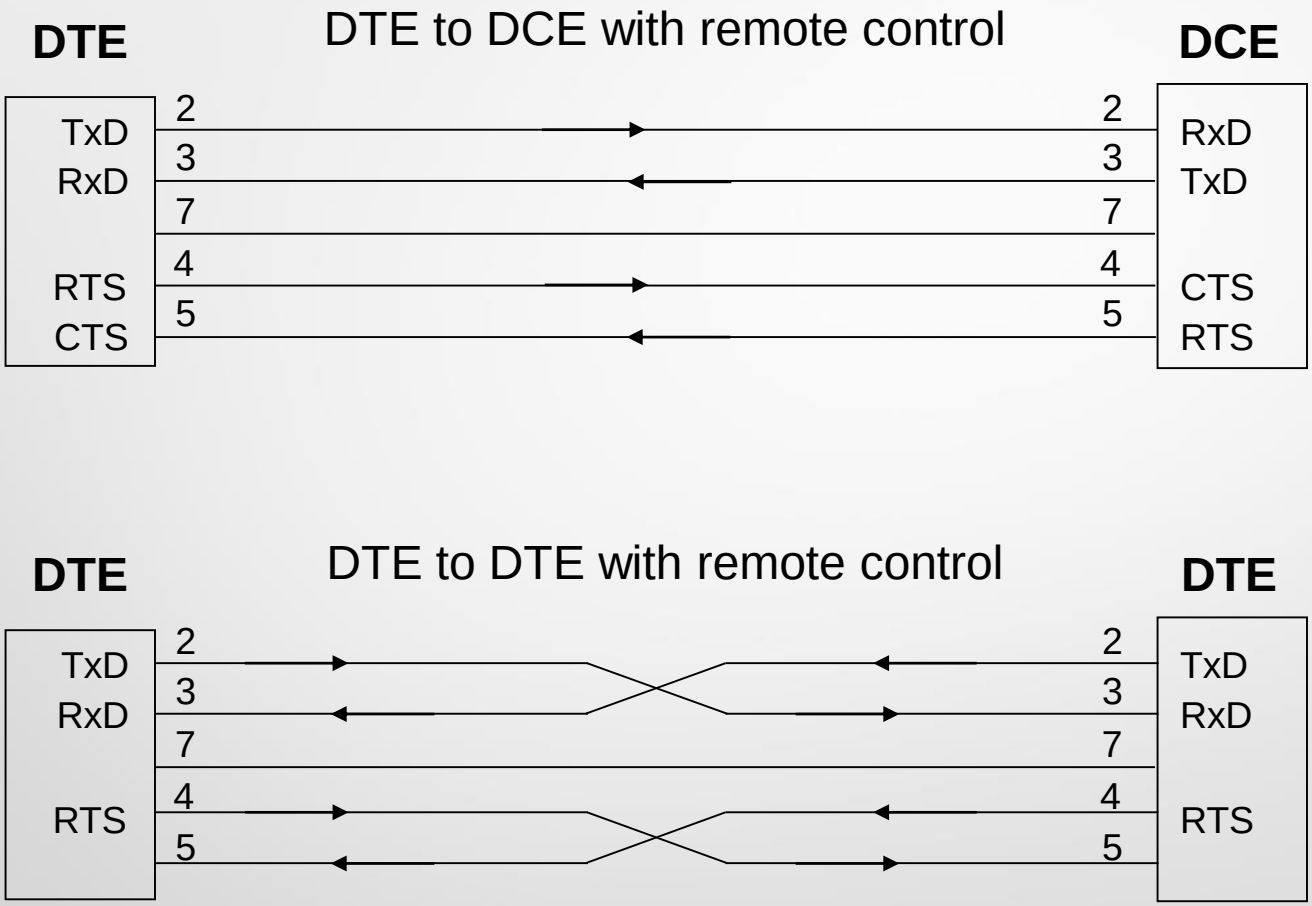




LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

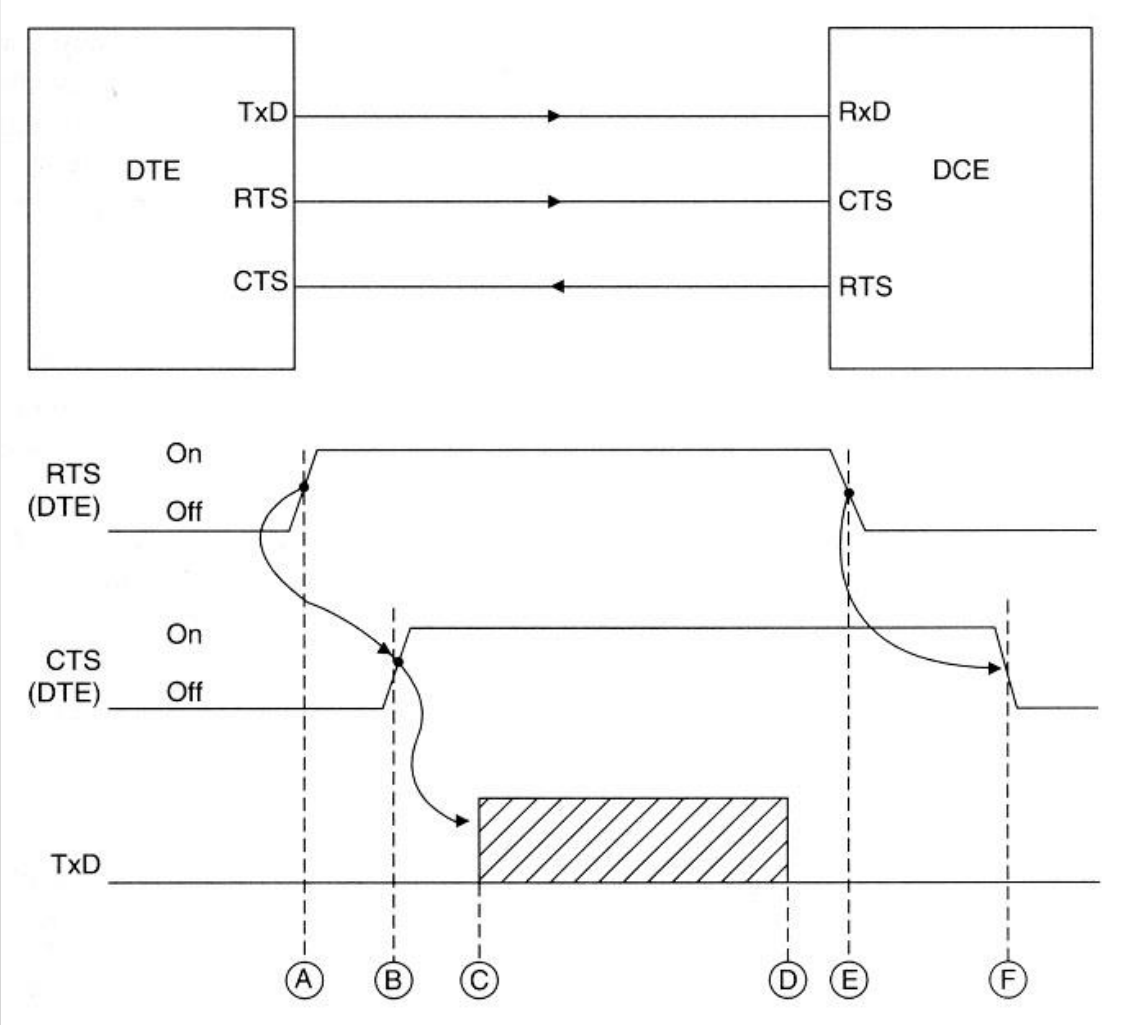
The Minimal RS-232 Function



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Handshaking Between RTS and CTS

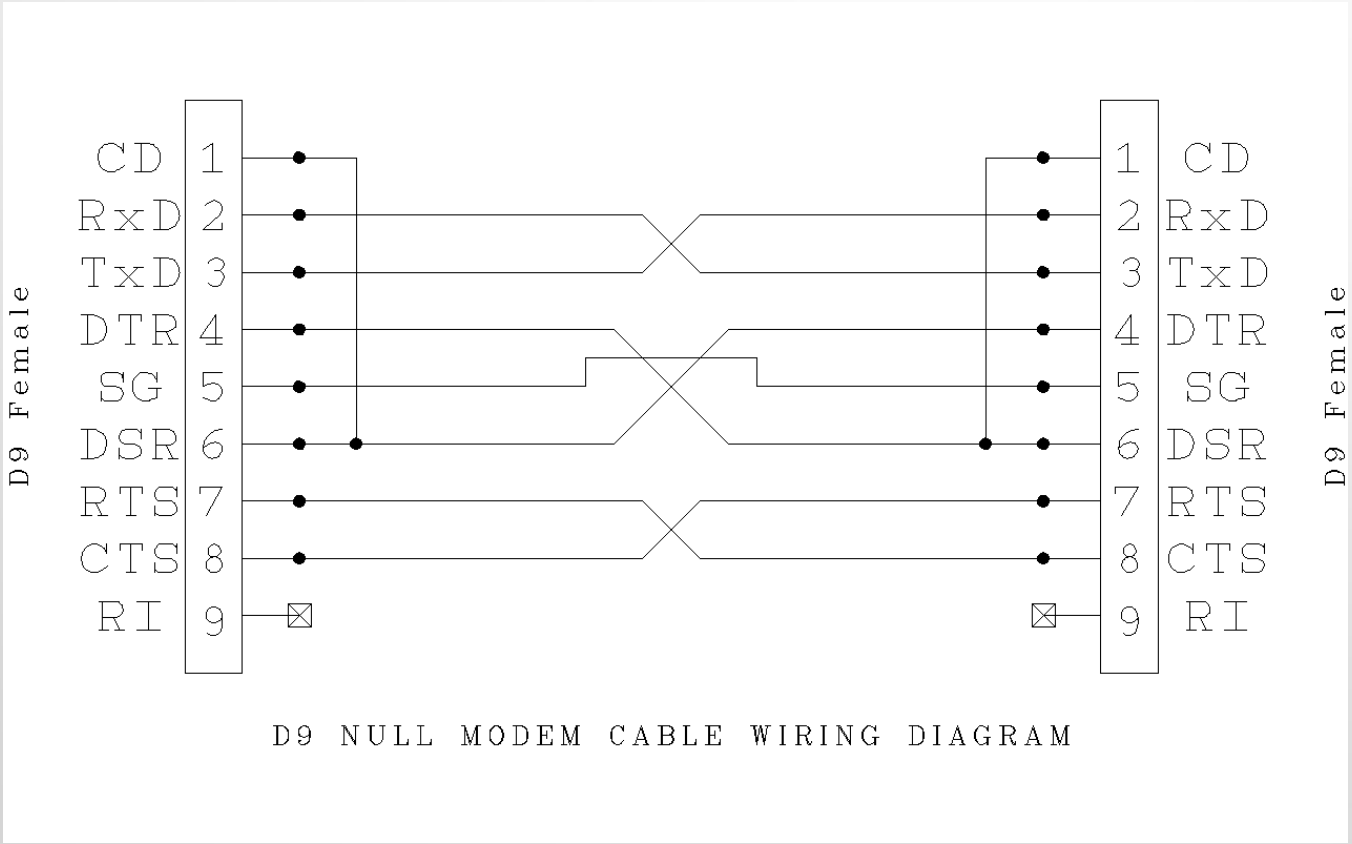


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Null Modem

- Null-modem simulates a DTE-DCE-DCE-DTE circuit



D9 NULL MODEM CABLE WIRING DIAGRAM

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Serial Peripheral Interface

- Serial Peripheral Interface – SPI
  - It is a synchronous serial data link standard named by Motorola that operates in full duplex mode
  - Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines.
- SPI specifies four logic signals
  - SCLK — Serial Clock (output from master)
  - MOSI/SIMO — Master Output, Slave Input (output from master)
  - MISO/SOMI — Master Input, Slave Output (output from slave)
  - SS — Slave Select (active low; output from master)

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

SPI Mode in USCI: Signal Definition

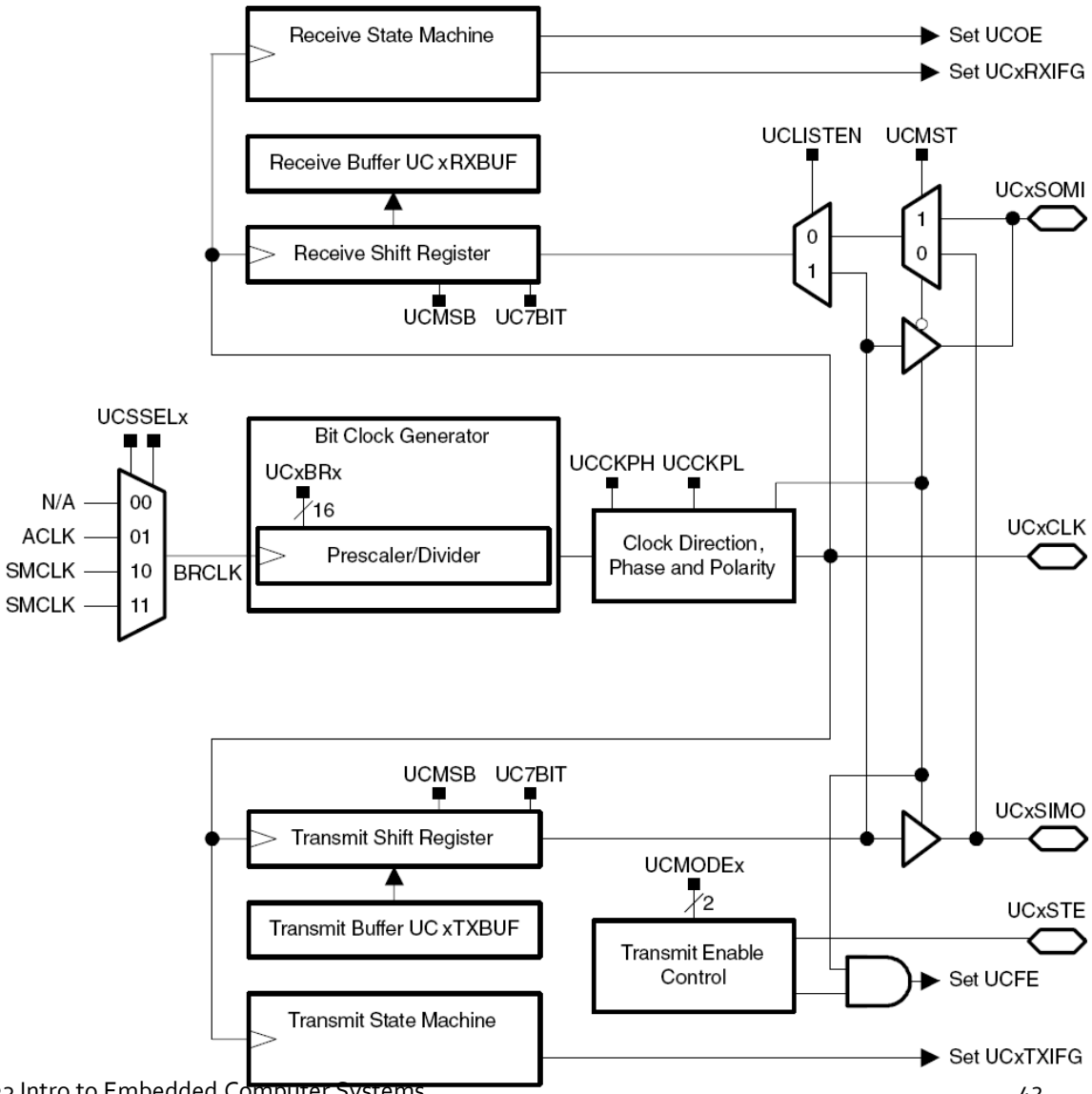
- UCxSIMO Slave in, master out
  - Master mode: UCxSIMO is the data output line
  - Slave mode: UCxSIMO is the data input line
- UCxSOMI Slave out, master in
  - Master mode: UCxSOMI is the data input line
  - Slave mode: UCxSOMI is the data output line
- UCxCLK USCI SPI clock
  - Master mode: UCxCLK is an output
  - Slave mode: UCxCLK is an input
- UCxSTE Slave transmit enable.
  - Used in 4-pin mode to allow multiple masters on a single bus. Not used in 3-pin mode.

UCMODEx	UCxSTE Active State	UCxSTE	Slave	Master
01	high	0	inactive	active
		1	active	inactive
10	low	0	active	inactive
		1	inactive	active

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

USCI: SPI Mode



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

SPI Mode: Initialization Sequence

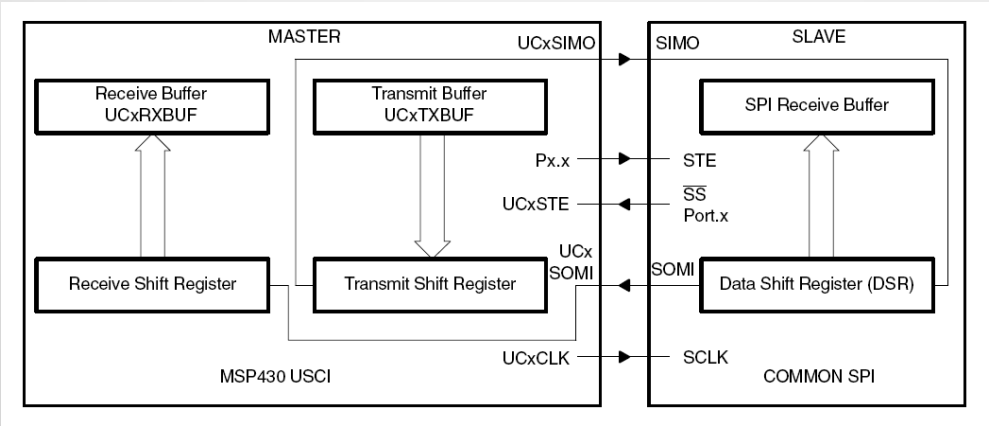
- 1) Set UCSWRST (BIS.B #UCSWRST,&UCxCTL1)
- 2) Initialize all USCI registers with UCSWRST=1 (including UCxCTL1)
- 3) Configure ports
- 4) Clear UCSWRST via software (BIC.B #UCSWRST,&UCxCTL1)
- 5) Enable interrupts (optional) via UCxRXIE and/or UCxTXIE

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

SPI Master Mode

- 1) Move data to UCxTXBUF to initiate transfer
- 2) UCxTXBUF data is moved to the TX shift register when the TX shift register is empty, initiating data transfer on UCxSIMO starting with either the MSB or LSB
- 3) Data on UCxSOMI is shifted into the receive shift register on the opposite clock edge
- 4) When the character is received, the receive data is moved from the RX shift register to UCxRXBUF and the receive interrupt flag, UCxRXIFG, is set, indicating the RX/TX operation is complete
- (Note #1: A set transmit interrupt flag, UCxTXIFG, indicates that data has moved from UCxTXBUF to the TX shift register and UCxTXBUF is ready for new data. It does not indicate RX/TX completion)
- (Note #2: To receive data into the USCI in master mode, data must be written to UCxTXBUF because receive and transmit operations operate concurrently)



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

4-pin SPI Master Mode

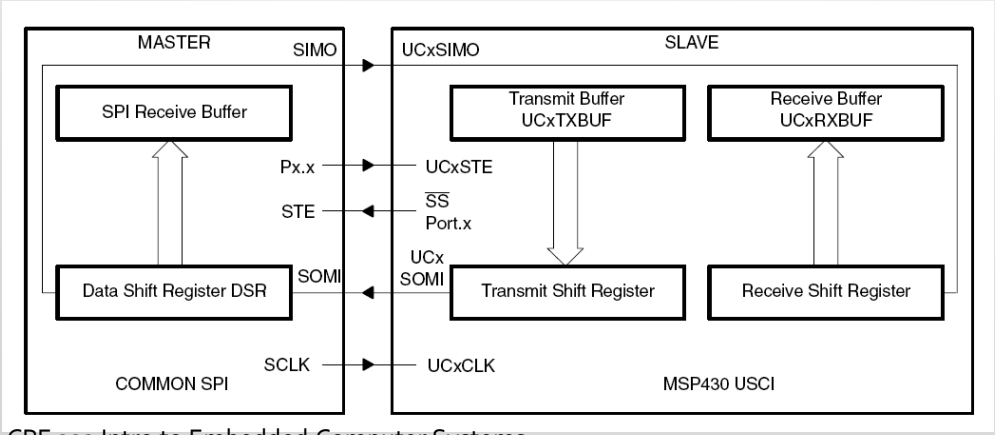
- UCxSTE is used to prevent conflicts with another master and controls the master
- When UCxSTE is in the master-inactive state:
  - UCxSIMO and UCxCLK are set to inputs and no longer drive the bus
  - The error bit UCFE is set indicating a communication integrity violation to be handled by the user
  - The internal state machines are reset and the shift operation is aborted
- If data is written into UCxTXBUF while the master is held inactive by UCxSTE, it will be transmit as soon as UCxSTE transitions to the master-active state.
- If an active transfer is aborted by UCxSTE transitioning to the master-inactive state, the data must be re-written into UCxTXBUF to be transferred when UCxSTE transitions back to the master-active state.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

SPI Slave Mode

- UCxCLK is used as the input for the SPI clock and must be supplied by the external master
- UCxTXBUF moved to the TX shift register before the start of UCxCLK
- Transmitted on UCxSOMI.
- Data on UCxSIMO is shifted into the receive shift register on the opposite edge of UCxCLK and moved to UCxRXBUF when the set number of bits are received
- When data is moved from the RX shift register to UCxRXBUF, the UCxRXIFG interrupt flag is set, indicating that data has been received
- (Note #1: The overrun error bit UCOE is set when the previously received data is not read from UCxRXBUF before new data is moved to UCxRXBUF).



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

4-pin SPI Slave Mode

- In 4-pin slave mode, UCxSTE is used by the slave to enable the transmit and receive operations and is provided by the SPI master
- When UCxSTE is in the slave-active state, the slave operates normally
- When UCxSTE is in the slave-inactive state:
  - Any receive operation in progress on UCxSIMO is halted UCxSOMI is set to the input direction
  - The shift operation is halted until the UCxSTE line transitions into the slave transmit active state

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

SPI Mode

- When the USCI module is enabled by clearing the UCSWRST bit it is ready to receive and transmit
- In master mode the bit clock generator is ready, but is not clocked nor producing any clocks.
- In slave mode the bit clock generator is disabled and the clock is provided by the master.
- A transmit or receive operation is indicated by UCBUSY = 1. The UCBUSY flag is set by writing UCxTXBUF in master mode and in slave mode with UCCKPH=1. In slave mode with UCCKPH=0 UCBUSY is set with the first UCLK edge. UCBUSY is reset by the following conditions:
  - In master mode when transfer completed and UCxTXBUF empty.
  - In slave mode with UCCKPH=0 when transfer completed.
  - In slave mode with UCCKPH=1 when transfer completed and UCxTXBUF empty.
- Transmit Enable
  - In master mode, writing to UCxTXBUF activates the bit clock generator and the data will begin to transmit.
  - In slave mode, transmission begins when a master provides a clock and, in 4-pin mode, when the UCxSTE is in the slave-active state.
- Receive Enable
  - The SPI receives data when a transmission is active. Receive and transmit operations operate concurrently.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

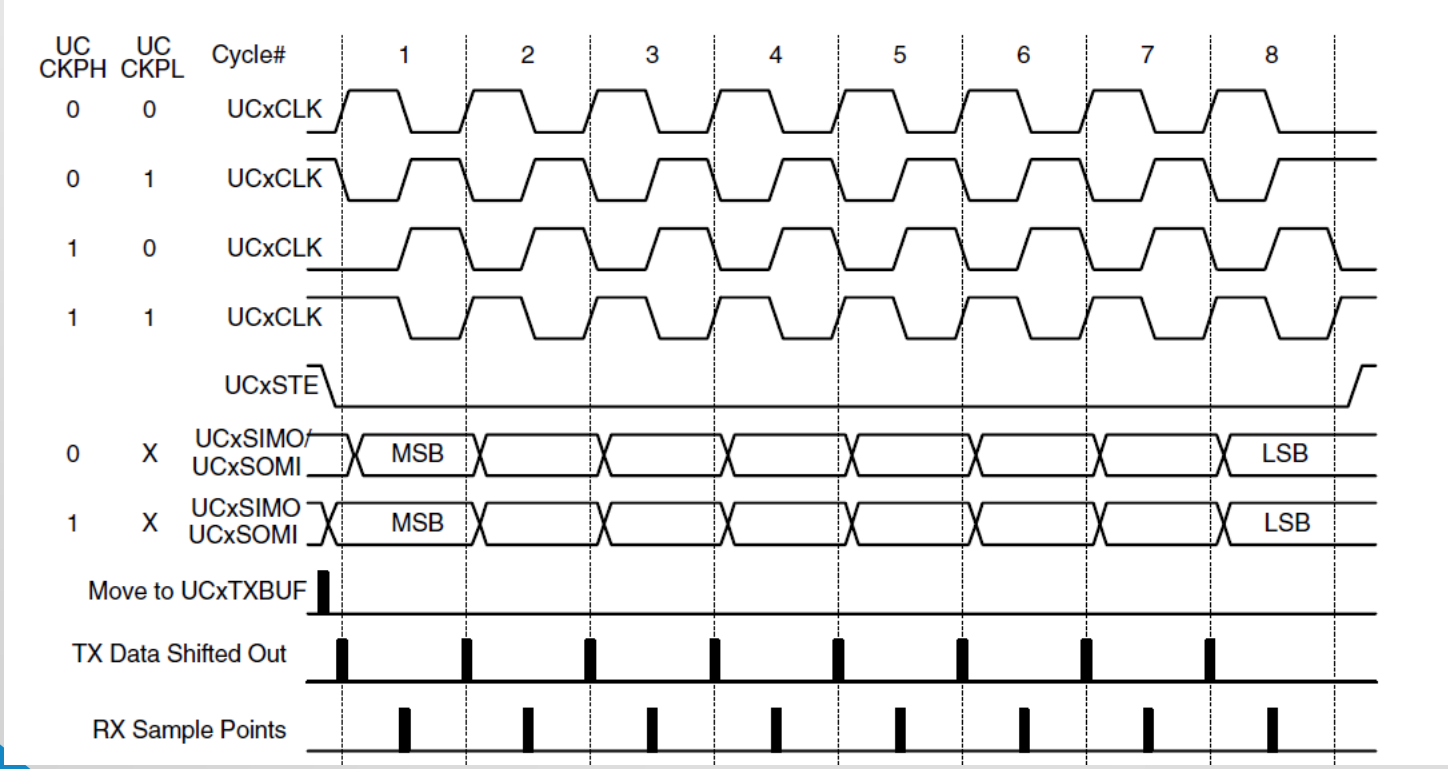
Serial Clock Control

- UCxCLK is provided by the master on the SPI bus
  - When UCMST = 1, the bit clock is provided by the USCI bit clock generator on the UCxCLK pin. The clock used to generate the bit clock is selected with the UCSSELx bits.
  - When UCMST = 0, the USCI clock is provided on the UCxCLK pin by the master, the bit clock generator is not used, and the UCSSELx bits are don't care.
- The 16-bit value of UCBRx in the bit rate control registers UCxxBR1 and UCxxBR0 is the division factor of the USCI clock source, BRCLK. The maximum bit clock that can be generated in master mode is BRCLK.
- Modulation is not used in SPI mode and UCAxMCTL should be cleared when using SPI mode for USCI_A. The UCAxCLK/UCBxCLK frequency is given by F(BRCLK)/UCBRx

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

USCI SPI Timing

- Polarity and phase of UCxCLK can be controlled by UCCKPH and UCCKPL control bits



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

SPI Registers

Table 20–2.USCI_A0 and USCI_B0 Control and Status Registers

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

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

SPI Registers (cont'd)

Register	Short Form	Register Type	Address	Initial State
USCI_A1 control register 0	UCA1CTL0	Read/write	0D0h	Reset with PUC
USCI_A1 control register 1	UCA1CTL1	Read/write	0D1h	001h with PUC
USCI_A1 Baud rate control register 0	UCA1BR0	Read/write	0D2h	Reset with PUC
USCI_A1 Baud rate control register 1	UCA1BR1	Read/write	0D3h	Reset with PUC
USCI_A1 modulation control register	UCA1MCTL	Read/write	0D4h	Reset with PUC
USCI_A1 status register	UCA1STAT	Read/write	0D5h	Reset with PUC
USCI_A1 Receive buffer register	UCA1RXBUF	Read	0D6h	Reset with PUC
USCI_A1 Transmit buffer register	UCA1TXBUF	Read/write	0D7h	Reset with PUC
USCI_B1 control register 0	UCB1CTL0	Read/write	0D8h	001h with PUC
USCI_B1 control register 1	UCB1CTL1	Read/write	0D9h	001h with PUC
USCI_B1 Bit rate control register 0	UCB1BR0	Read/write	0DAh	Reset with PUC
USCI_B1 Bit rate control register 1	UCB1BR1	Read/write	0DBh	Reset with PUC
USCI_B1 status register	UCB1STAT	Read/write	0DDh	Reset with PUC
USCI_B1 Receive buffer register	UCB1RXBUF	Read	0DEh	Reset with PUC
USCI_B1 Transmit buffer register	UCB1TXBUF	Read/write	0DFh	Reset with PUC
USCI_A1/B1 interrupt enable register	UC1IE	Read/write	006h	Reset with PUC
USCI_A1/B1 interrupt flag register	UC1IFG	Read/write	007h	00Ah with PUC

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

UCAxCTL0, USCI_Ax Control Register 0
UCBxCTL0, USCI_Bx 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	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

UC7BIT Bit 4 Character length. Selects 7-bit or 8-bit character length.
- 0 8-bit data
- 1 7-bit data

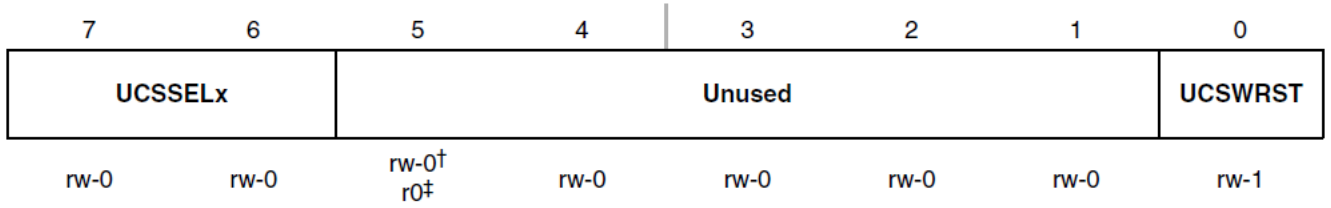
UCMST Bit 3 Master mode select
- 0 Slave mode
- 1 Master mode

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

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

UCAxCTL1, USCI_Ax Control Register 1
UCBxCTL1, USCI_Bx Control Register 1



† UCAxCTL1 (USCI_Ax)
‡ UCBxCTL1 (USCI_Bx)

UCSSELx	Bits 7-6	USCI clock source select. These bits select the BRCLK source clock in master mode. UCxCLK is always used in slave mode. 00 NA 01 ACLK 10 SMCLK 11 SMCLK
Unused	Bits 5-1	Unused in synchronous mode (UCSYNC=1).
UCSWRST	Bit 0	Software reset enable 0 Disabled. USCI reset released for operation. 1 Enabled. USCI logic held in reset state.

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Demo #1 (SPI Master)

```
//******************************************************************************
//   MSP430xG46x Demo - USCI_A0, SPI 3-Wire Master Incremented Data
//
//   Description: SPI master talks to SPI slave using 3-wire mode. Incrementing
//   data is sent by the master starting at 0x01. Received data is expected to
//   be same as the previous transmission.  USCI RX ISR is used to handle
//   communication with the CPU, normally in LPM0. If high, P5.1 indicates
//   valid data reception.  Because all execution after LPM0 is in ISRs,
//   initialization waits for DCO to stabilize against ACLK.
//   ACLK = 32.768kHz, MCLK = SMCLK = DCO ~ 1048kHz.  BRCLK = SMCLK/2
//
//   Use with SPI Slave Data Echo code example.  If slave is in debug mode, P5.2
//   slave reset signal conflicts with slave's JTAG; to work around, use IAR's
//   "Release JTAG on Go" on slave device.  If breakpoints are set in
//   slave RX ISR, master must stopped also to avoid overrunning slave
//   RXBUF.
//
//                   MSP430FG4619
//                -----------------
//            /|\|              XIN|-
//             | |                 |  32kHz xtal
//             --|RST          XOUT|-
//               |                 |
//               |             P7.1|-> Data Out (UCA0SIMO)
//               |                 |
//         LED <-|P5.1         P7.2|<- Data In (UCA0SOMI)
//               |                 |
//  Slave reset <-|P5.2         P7.3|-> Serial Clock Out (UCA0CLK)
//
//
//   K. Quiring/ M. Mitchell
//   Texas Instruments Inc.
//   October 2006
//   Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.41A
//******************************************************************************
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Demo #1 (SPI Master)

```
#include  <msp430xG46x.h>
unsigned char MST_Data,SLV_Data;

void main(void) {
  volatile unsigned int i;
  WDTCTL = WDTPW+WDTHOLD;                   // Stop watchdog timer
  FLL_CTL0 |= XCAP14PF;                     // Configure load caps
  // Wait for xtal to stabilize
  do {
    IFG1 &= ~OFIFG;                           // Clear OSCFault flag
    for (i = 0x47FF; i > 0; i--);             // Time for flag to set
  }
  while ((IFG1 & OFIFG));                   // OSCFault flag still set?

  for(i=2100;i>0;i--);                      // Now with stable ACLK, wait for
                                            // DCO to stabilize.
  P5OUT = 0x04;                             // P5 setup for LED and slave reset
  P5DIR |= 0x06;                            //
  P7SEL |= 0x0E;                            // P7.3,2,1 option select
  UCA0CTL0 |= UCMST+UCSYNC+UCCKPL+UCMSB;    //3-pin, 8-bit SPI master
  UCA0CTL1 |= UCSSEL_2;                     // SMCLK
  UCA0BR0 = 0x02;                           // /2
  UCA0BR1 = 0;                              //
  UCA0MCTL = 0;                             // No modulation
  UCA0CTL1 &= ~UCSWRST;                     // **Initialize USCI state machine**
  IE2 |= UCA0RXIE;                          // Enable USCI_A0 RX interrupt

  P5OUT &= ~0x04;                           // Now with SPI signals initialized,
  P5OUT |= 0x04;                            // reset slave

  for(i=50;i>0;i--);                        // Wait for slave to initialize

  MST_Data = 0x001;                         // Initialize data values
  SLV_Data = 0x000;                         //

  UCA0TXBUF = MST_Data;                     // Transmit first character

  _BIS_SR(LPM0_bits + GIE);                 // CPU off, enable interrupts
}
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Demo #1 (SPI Master)

```
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCIA0RX_ISR (void)
{
  volatile unsigned int i;

  while (!(IFG2 & UCA0TXIFG));              // USART1 TX buffer ready?
  if (UCA0RXBUF==SLV_Data)                  // Test for correct character RX'd
    P5OUT |= 0x02;                          // If correct, light LED
  else
    P5OUT &= ~0x02;                         // If incorrect, clear LED

  MST_Data++;                               // Increment data
  SLV_Data++;
  UCA0TXBUF = MST_Data;                     // Send next value

  for(i=30;i>0;i--);                        // Add time between transmissions to
}                                           // make sure slave can keep up
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Demo #1 (SPI Slave)

```
//******************************************************************************
//   MSP430xG46x Demo - USCI_A0, SPI 3-Wire Slave Data Echo
//
//   Description: SPI slave talks to SPI master using 3-wire mode. Data received
//   from master is echoed back.  USCI RX ISR is used to handle communication,
//   CPU normally in LPM4.  Prior to initial data exchange, master pulses
//   slaves RST for complete reset.
//   ACLK = 32.768kHz, MCLK = SMCLK = DCO ~ 1048kHz
//
//   Use with SPI Master Incremented Data code example.  If the slave is in
//   debug mode, the reset signal from the master will conflict with slave's
//   JTAG; to work around, use IAR's "Release JTAG on Go" on slave device.  If
//   breakpoints are set in slave RX ISR, master must stopped also to avoid
//   overrunning slave RXBUF.
//
//                MSP430FG4619
//             -----------------
//         /|\|              XIN|-
//          | |                 |   32kHz xtal
//          | |             XOUT|-
// Master---+-|RST              |
//            |             P7.1|<- Data In (UCA0SIMO)
//            |                 |
//            |             P7.2|-> Data Out (UCA0SOMI)
//            |                 |
//            |             P7.3|<- Serial Clock In (UCA0CLK)
//
//
//   K. Quiring/ M. Mitchell
//   Texas Instruments Inc.
//   October 2006
//   Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.41A
//******************************************************************************
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Demo #1 (SPI Slave)

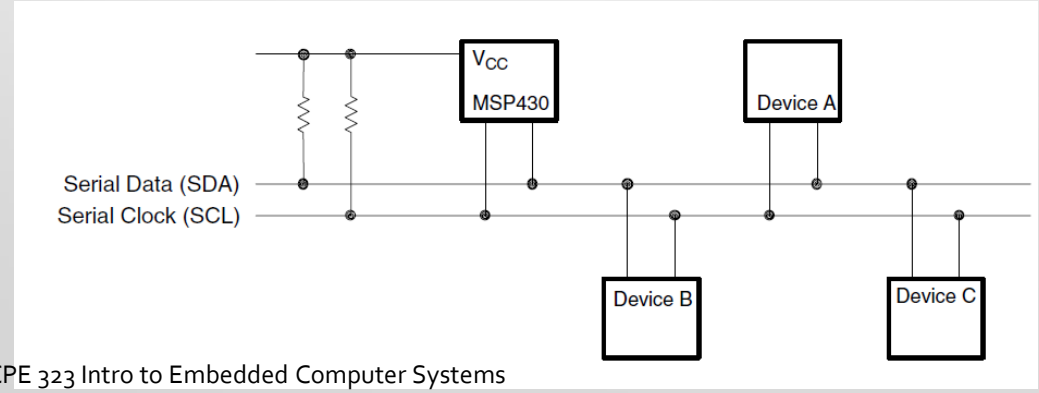
```
#include  <msp430xG46x.h>

void main(void) {
  volatile unsigned int i;

  WDTCTL = WDTPW+WDTHOLD;                   // Stop watchdog timer
  FLL_CTL0 |= XCAP14PF;                     // Configure load caps

  // Wait for xtal to stabilize
  do
  {
    IFG1 &= ~OFIFG;                         // Clear OSCFault flag
    for (i = 0x47FF; i > 0; i--);           // Time for flag to set
  }
  while ((IFG1 & OFIFG));                   // OSCFault flag still set?

  for(i=2100;i>0;i--);                      // Now with stable ACLK, wait for
                                            // DCO to stabilize.
  while(!(P7IN&0x08));                      // If clock sig from mstr stays low,
                                            // it is not yet in SPI mode
  P7SEL |= 0x00E;                           // P7.3,2,1 option select
  UCA0CTL1 = UCSWRST;                       // **Put state machine in reset**
  UCA0CTL0 |= UCSYNC+UCCKPL+UCMSB;          // 3-pin, 8-bit SPI master
  UCA0CTL1 &= ~UCSWRST;                     // **Initialize USCI state machine**
  IE2 |= UCA0RXIE;                          // Enable USCI_A0 RX interrupt

  _BIS_SR(LPM4_bits + GIE);                 // Enter LPM4, enable interrupts
}

// Echo character
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCIA0RX_ISR (void)
{
  while (!(IFG2 & UCA0TXIFG));              // USCI_A0 TX buffer ready?
  UCA0TXBUF = UCA0RXBUF;
}
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

I2C: Inter-Integrated Circuit Bus

- I^2C – introduced by Philips (NXP) Semiconductors
  - a.k.a. two-wire interface
- True bus, with well-defined specification
- Two bidirectional lines
  - SDA: Serial Data
  - SCL: Serial Clock
- More complex than SPI, more HW resources (e.g., in USCI_B), slower than SPI

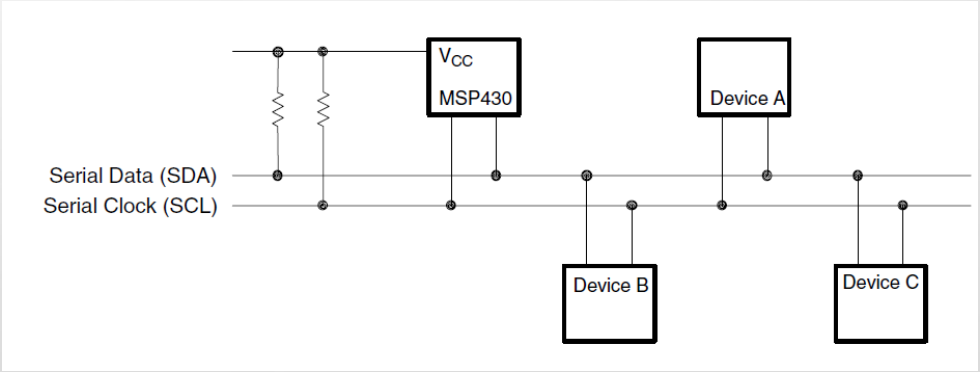


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

I²C Operation

- Each I²C device is recognized by a unique address and can operate as either a transmitter or a receiver.
- A device connected to the I²C bus can be considered as the master or the slave when performing data transfers
  - A master initiates a data transfer and generates the clock signal SCL.
  - Any device addressed by a master is considered a slave.
- I²C uses two bidirectional lines connected to a positive supply voltage through a pullup transistor (wired-and)
  - Serial data pin (SDA)
  - Serial clock pin (SCL)

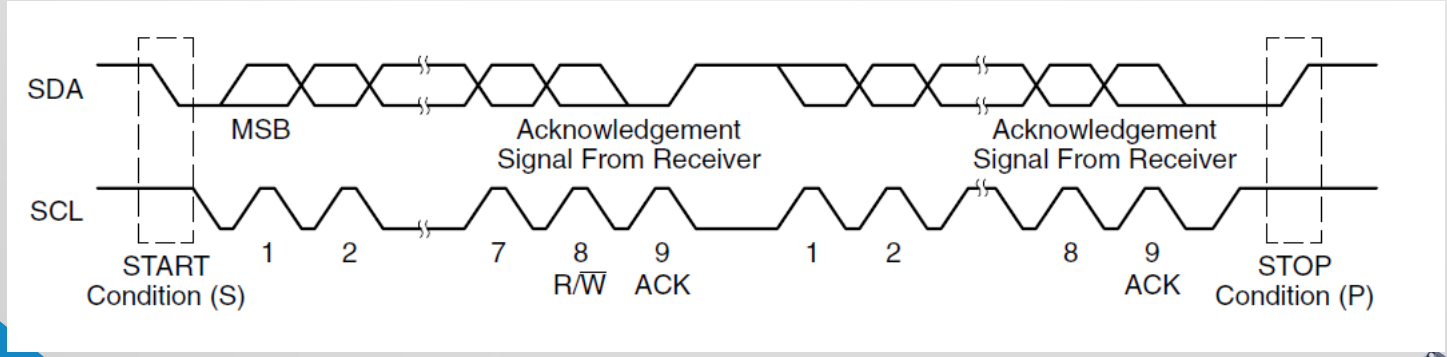


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

I²C Serial Transfer

- One clock pulse is generated by the master device for each data bit transferred. The I²C mode operates with byte data. Data is transferred most significant bit first.
- The first byte after a START condition consists of a 7-bit slave address and the R/#W bit
  - When R/#W = 0, the master transmits data to a slave.
  - When R/#W = 1, the master receives data from a slave.
- The ACK bit is sent from the receiver after each byte on the 9th SCL clock.

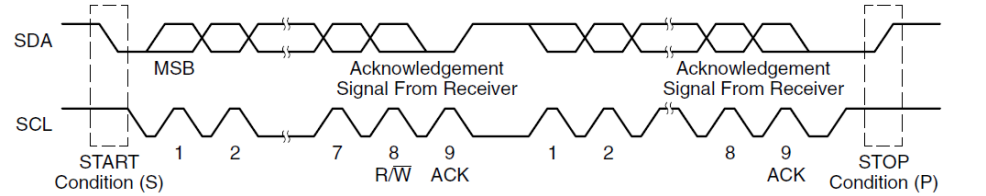


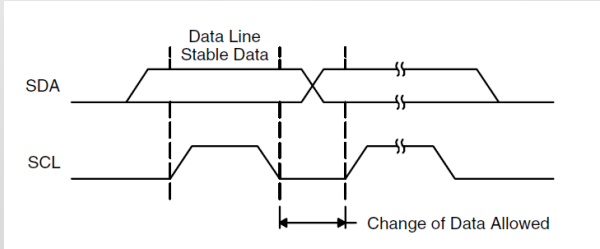
LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

I²C Serial Transfer (cont'd)

- START and STOP conditions are generated by the master
- A START condition is a high-to-low transition on the SDA line while SCL is high.
- A STOP condition is a low-to-high transition on the SDA line while SCL is high.
- The bus busy bit, UCBBUSY, is set after a START and cleared after a STOP.
- Data on SDA must be stable during the high period of SCL
  - The high and low state of SDA can only change when SCL is low, otherwise START or STOP conditions will be generated.

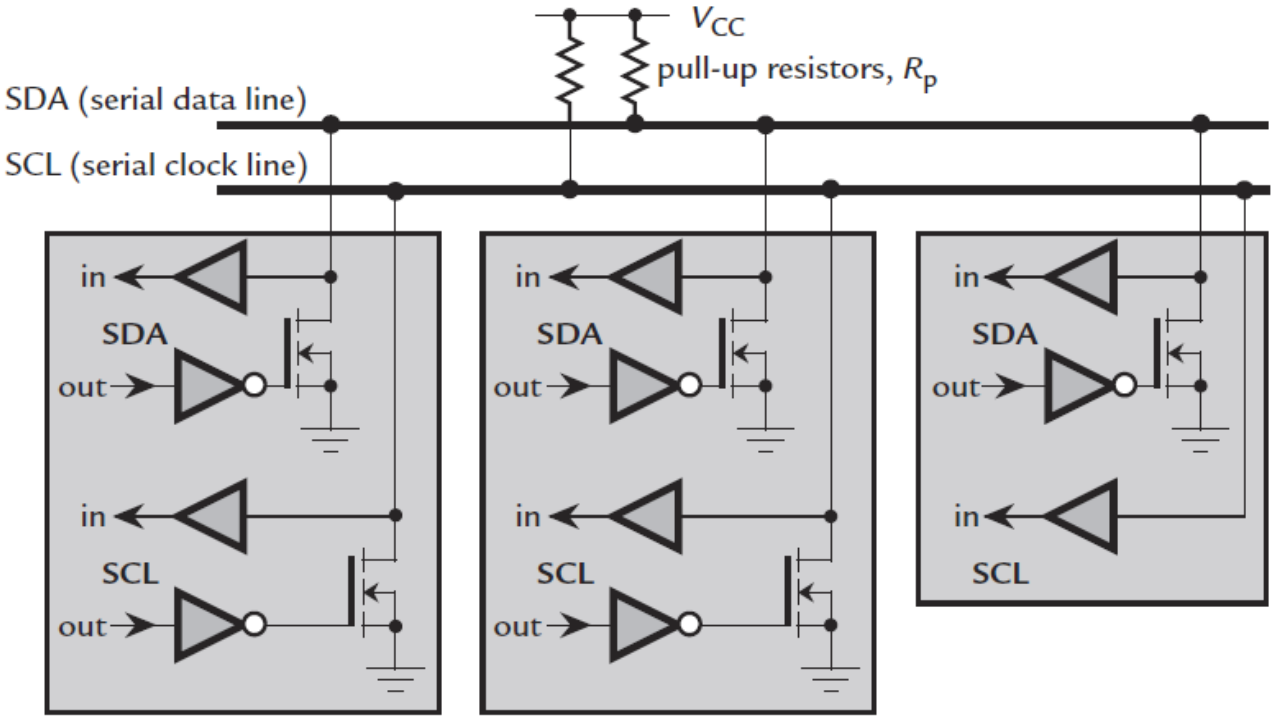




LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

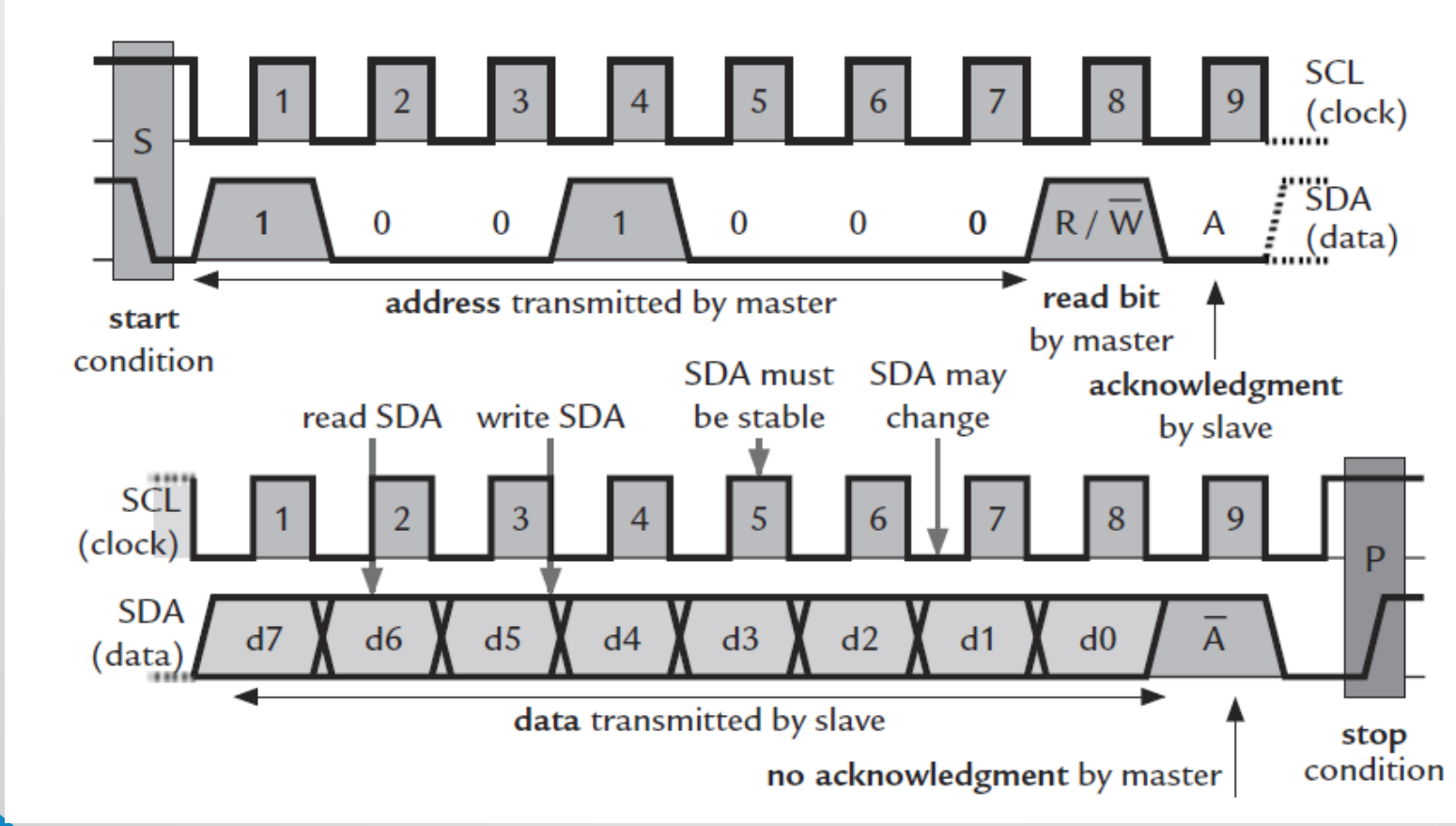
Hardware for I²C



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Master Reads from a Slave



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

I²C Addressing Modes

Figure 21–5. I²C Module 7-Bit Addressing Format

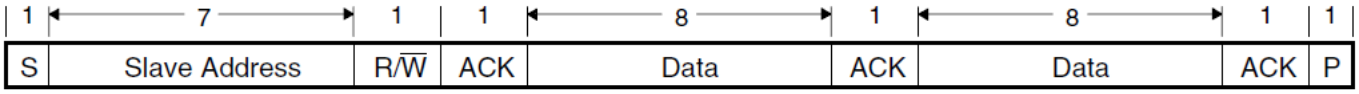


Figure 21–6. I²C Module 10-Bit Addressing Format

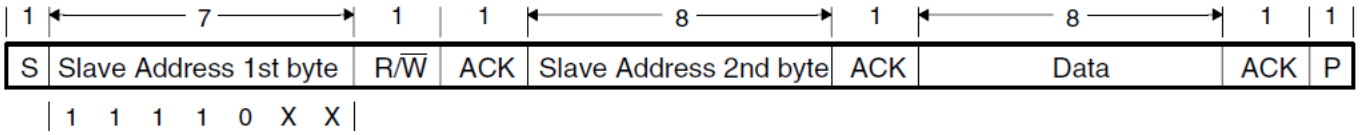
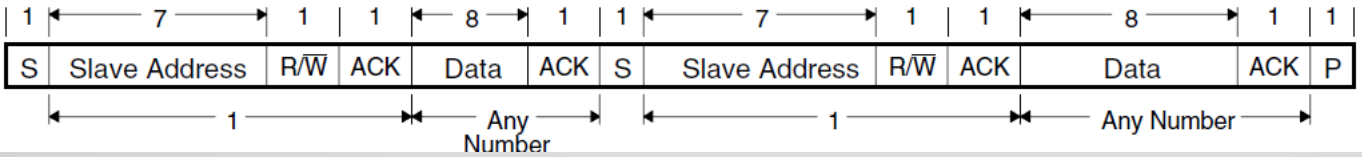


Figure 21–7. I²C Module Addressing Format with Repeated START Condition



LaCASA

Slave Transmitter Mode

- Slave transmitter mode is entered when the slave address transmitted by the master is identical to its own address with a set R/W bit (read).
- The USCI module is automatically configured as a transmitter and UCTR and UCBxTXIFG become set
  - The slave transmitter shifts the serial data out on SDA with the clock pulses that are generated by the master device
- The slave device does not generate the clock, but it will hold SCL low while intervention of the CPU is required after a byte has been transmitted.
- The SCL line is held low until the first data to be sent is written into the transmit buffer UCBxTXBUF. Then the address is acknowledged, the UCSTTIFG flag is cleared, and the data is transmitted.

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Slave Transmitter Mode (cont'd)

- The SCL line is held low until the first data to be sent is written into the transmit buffer UCBxTXBUF. Then the address is acknowledged, the UCSTTIFG flag is cleared, and the data is transmitted.
- As soon as the data is transferred into the shift register the UCBxTXIFG is set again. After the data is acknowledged by the master the next data byte written into UCBxTXBUF is transmitted or if the buffer is empty the bus is stalled during the acknowledge cycle by holding SCL low until new data is written into UCBxTXBUF.
- If the master sends a NACK succeeded by a STOP condition the UCSTPIFG flag is set. If the NACK is succeeded by a repeated START condition the USCI I2C state machine returns to its address-reception state.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Slave Transmitter Mode

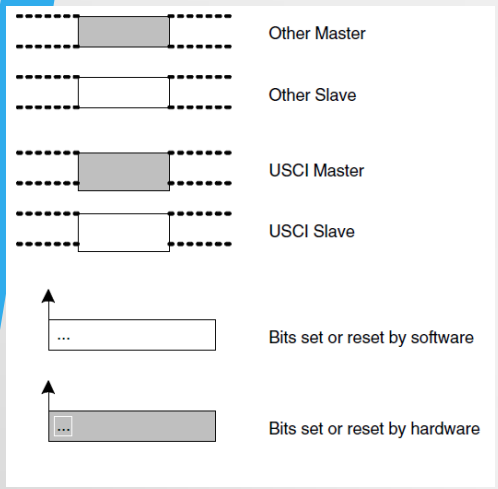
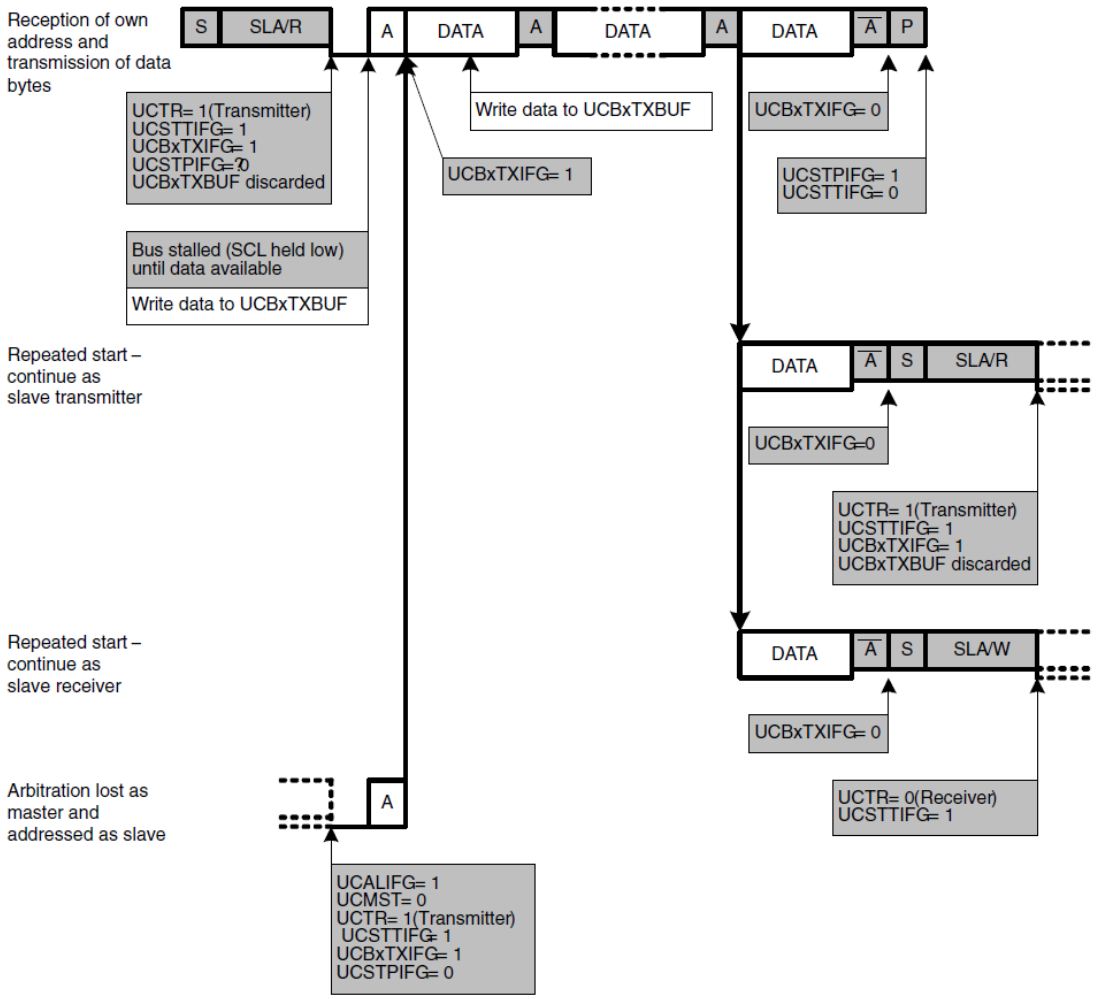


Figure 21–9. I²C Slave Transmitter Mode



LaCASA

Slave Receiver Mode

- Slave receiver mode is entered when the slave address transmitted by the master is identical to its own address and a cleared R/#W bit is received.
- The USCI module is automatically configured as a receiver and UCTR is cleared. After the first data byte is received the receive interrupt flag UCBxRXIFG is set.
- Serial data bits received on SDA are shifted in with the clock pulses that are generated by the master device
  - The slave device does not generate the clock, but it can hold SCL low if intervention of the CPU is required after a byte has been received.
- The USCI module automatically acknowledges the received data and can receive the next data byte.
- If the previous data was not read from the receive buffer UCBxRXBUF at the end of a reception, the bus is stalled by holding SCL low. As soon as UCBxRXBUF is read the new data is transferred into UCBxRXBUF, an acknowledge is sent to the master, and the next data can be received.

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Slave Receiver Mode (cont'd)

- Setting the UCTXNACK bit causes a NACK to be transmitted to the master during the next acknowledgment cycle. A NACK is sent even if UCBxRXBUF is not ready to receive the latest data.
- If the UCTXNACK bit is set while SCL is held low the bus will be released, a NACK is transmitted immediately, and UCBxRXBUF is loaded with the last received data.
- Since the previous data was not read that data will be lost. To avoid loss of data the UCBxRXBUF needs to be read before UCTXNACK is set.
- When the master generates a STOP condition the UCSTPIFG flag is set.
- If the master generates a repeated START condition the USCI I2C state machine returns to its address reception state.

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Slave Receiver Mode

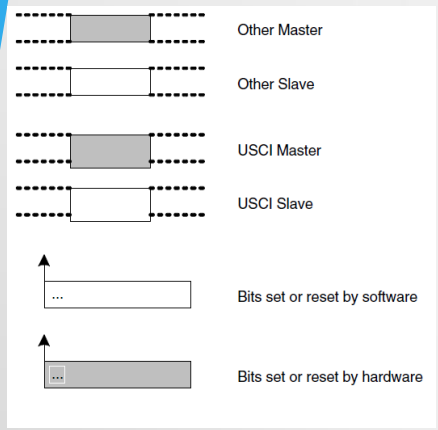
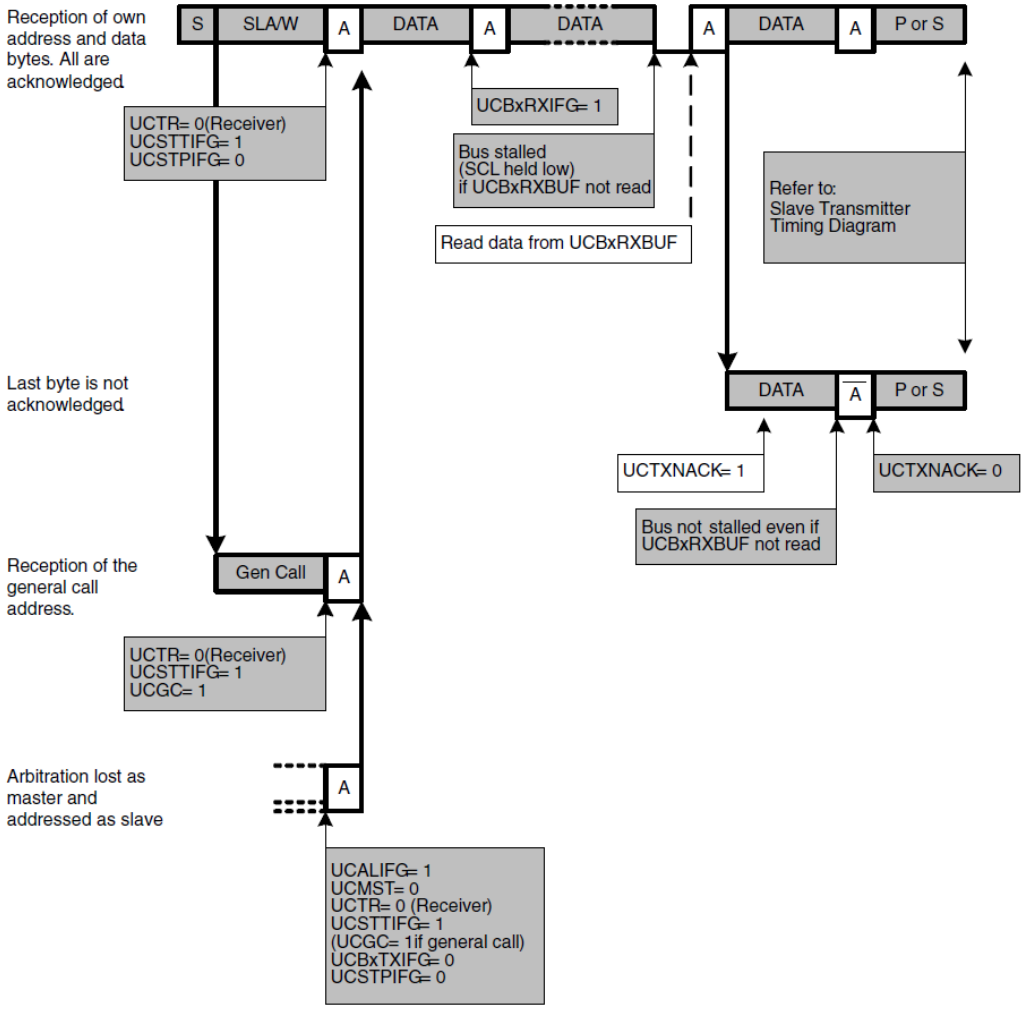


Figure 21–10. I²C Slave Receiver Mode



LaCASA