Chapter 14

Communications

The MSP430 contains built-in features for both parallel and serial data communication. This chapter describes the operation of these peripherals, and discusses the protocols, data formats and specific techniques for each type of data communication.

The communication modules available for the MSP430 family of microcontrollers are USART (Universal Synchronous/Asynchronous Receiver/Transmitter), USCI (Universal Serial Communication Interface) and USI (Universal Serial Interface). These provide asynchronous data transmission between the MSP430 and other peripheral devices when configured in UART mode. They also support data transmission synchronized to a clock signal through a serial I/O port in Serial Peripheral Interface (SPI) and Inter Integrated Circuit (I^2C) modes.

Topic **Page**

14.1 Introduction

An important feature of modern microprocessor based systems is their communication capability, that is, their ability to exchange information with other systems in the surrounding environment. The communications interfaces can be used for firmware update or loading local parameters. At a higher level, these interfaces can be used to exchange information in applications with distributed processes.

14.2 Communications system model

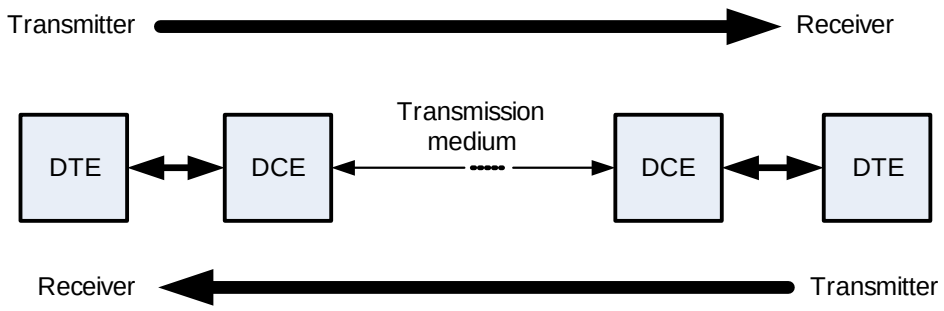
Any digital communication system has three devices:

- Transmitter: Has the task of processing the information into the appropriate format for subsequent transmission;
- Receiver: Is in charge of collecting the information and extracting the original data;
- Communication medium: Provides the physical medium through which the information flows and is commonly implemented as twisted pair wire, optical fibre cable or radio frequency network.

Figure 14-1 shows two devices participating in a digital communication system:

- DTE: Data Terminal Equipment;
- DCE: Data Communications Equipment.

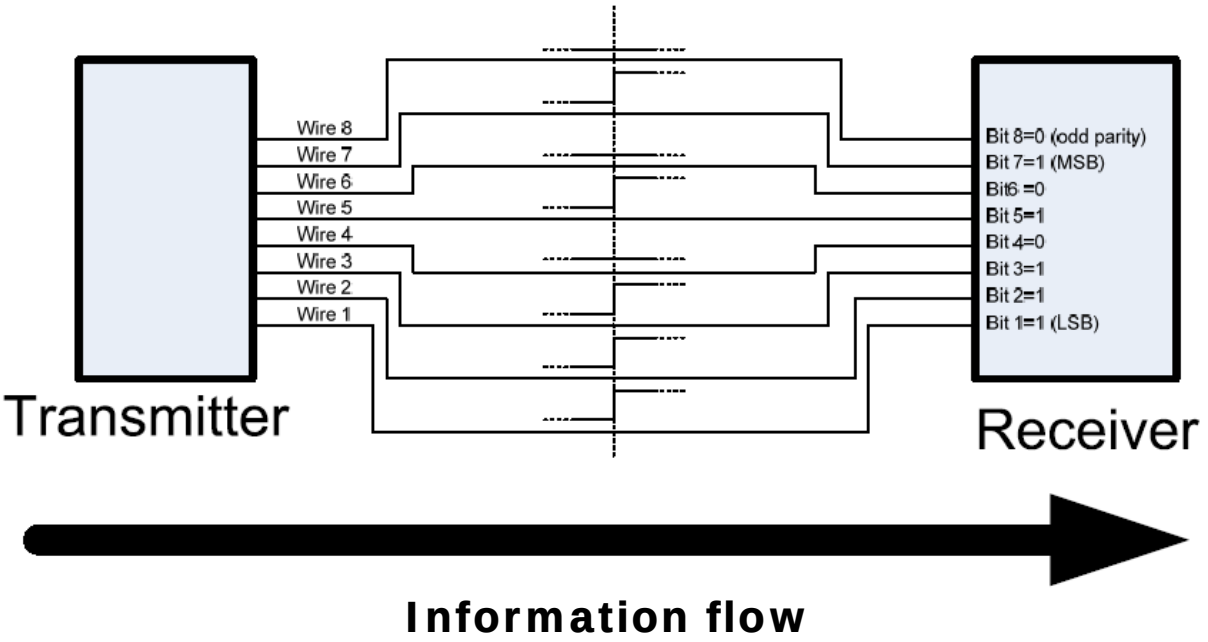
Figure 14-1. Communication system.



14.3 Transmission mode

Communications between digital devices are divided into parallel communications and serial communications. In parallel communications systems, the physical transmission medium has independent signal lines for each of the bits of the transmitted digital value. The information transmitted at a given moment is represented by the value formed by logical levels on the various lines (see *Figure 14-2*).

Figure 14-2. Character ASCII "W" parallel transmission.

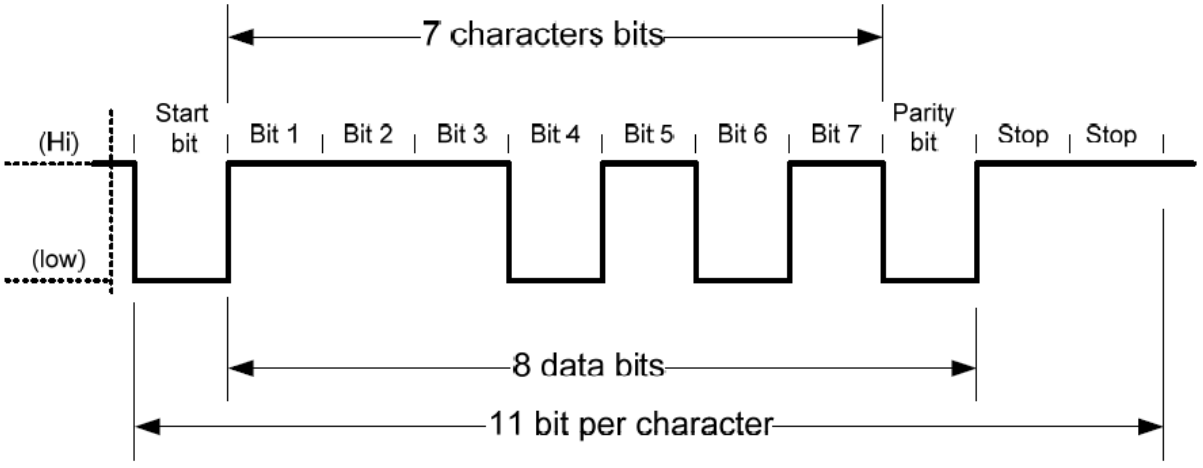


In serial communications, the physical transmission medium needs only one signal line. The information is sent by the transmitter as a sequence of bits, at a common rate established between the transmitter and the receiver. Additional information is needed to enable synchronization between the communication participants:

- Start bit: Added to the beginning of the information transmitted, the function of which is to the identify the beginning of a new word;
- Stop bit: Indicates completion of the transfer and is added to the end of the information transmitted.

Figure 14-3 gives an example of the ASCII character "W" being sent by serial transmission.

Figure 14-3. Character ASCII "W" serial transmission.



The two communication modes both have their advantages and disadvantages (see *Table 14-1*), but the parallel communication medium has been losing importance to serial communication. Serial communications, due to technological advances, have now achieved high transmission rates, making them more attractive for the most applications.

Table 14-1. Advantages and disadvantages of parallel and serial communication modes.

Characteristic	Parallel	Serial
Bus line	One line per bit	One line
Sequence	All bits of one word simultaneously	Sequence of bits
Transmission rate	High	Low
Bus length	Short distances	Short and long distances
Cost	High	Low
Critical characteristics	Synchronisation between the different bits is demanding	Asynchronous transmission needs start and stop bits Synchronous transmission needs some other synchronisation

14.4 Synchronous and asynchronous serial communications

Serial communications may be:

- Asynchronous: Where the transmission rate (baud rate) is fixed by the transmitter. The receiver must know this rate and synchronize itself to the transmitter when the start bit is detected;
- Synchronous: Where there is a synchronization clock signal between the receiver and the transmitter.

In synchronous communication, there is one unit that assumes the role of master and one or more units that assume the role(s) of slave. The clock signal generated by the master is used by the slave units to carry out the loading/unloading of TX and RX registers. In this communication mode, it is possible to transmit and receive simultaneously. In synchronous communications, both the sender and receiver are synchronized with a clock or a signal encoded into the data stream.

Asynchronous communication requires nothing more than a transmitter, a receiver and a wire. It is thus the simplest of serial communication protocols, and the least expensive to implement. As the name implies, asynchronous communication is performed between two (or more) devices which operate on independent clocks. Therefore, even if the two clocks agree for a time, there is no guarantee that they will continue to agree over extended periods.

14.5 Asynchronous (UART) communications

As shown in *Figure 14-3*, the start bit identifies the beginning of a transfer and is generated by a high-to-low transition on the bus. Following the start bit there are the seven or eight data bits (in this example, the ASCII code for the text transfer uses seven bits). The verification bit (parity bit) is sent after the data bits. To terminate the transmission, one or two stop bits are used.

14.5.1 Parity bit

The parity bit verifies the integrity of information transmitted. The bit is added by the transmitter and indicates whether the total number of bits at level "1" in the message data are odd or even. The transmissions can be configured for odd or even parity (see *Figure 14-4*).

Figure 14-4. Odd or even parity.

	7 bit ASCII code								
Bit	1	2	3	4	5	6	7	Parity bit odd	Parity bit even
B	0	1	0	0	0	0	1	1	0
Q	1	0	0	0	1	0	1	0	1
3	1	1	0	0	1	1	0	1	0
z	0	1	0	1	1	1	1	0	1

14.5.2 Baud rate

The transmission of the ASCII character "W" 7-bit character requires eleven bits to be sent, with four additional bits being used for control. This corresponds to a baud value of 11. If the character transmission rate is 10 characters per second, it will give a baud rate of 10x11 = 1100 baud/second.

The baud rate is a basic parameter in serial asynchronous communications. After the master and slave are synchronized through the start bit, they must use the same baud rate in order to know the appropriate moments to write (master) and read (slave) the various bits of the word transmitted.

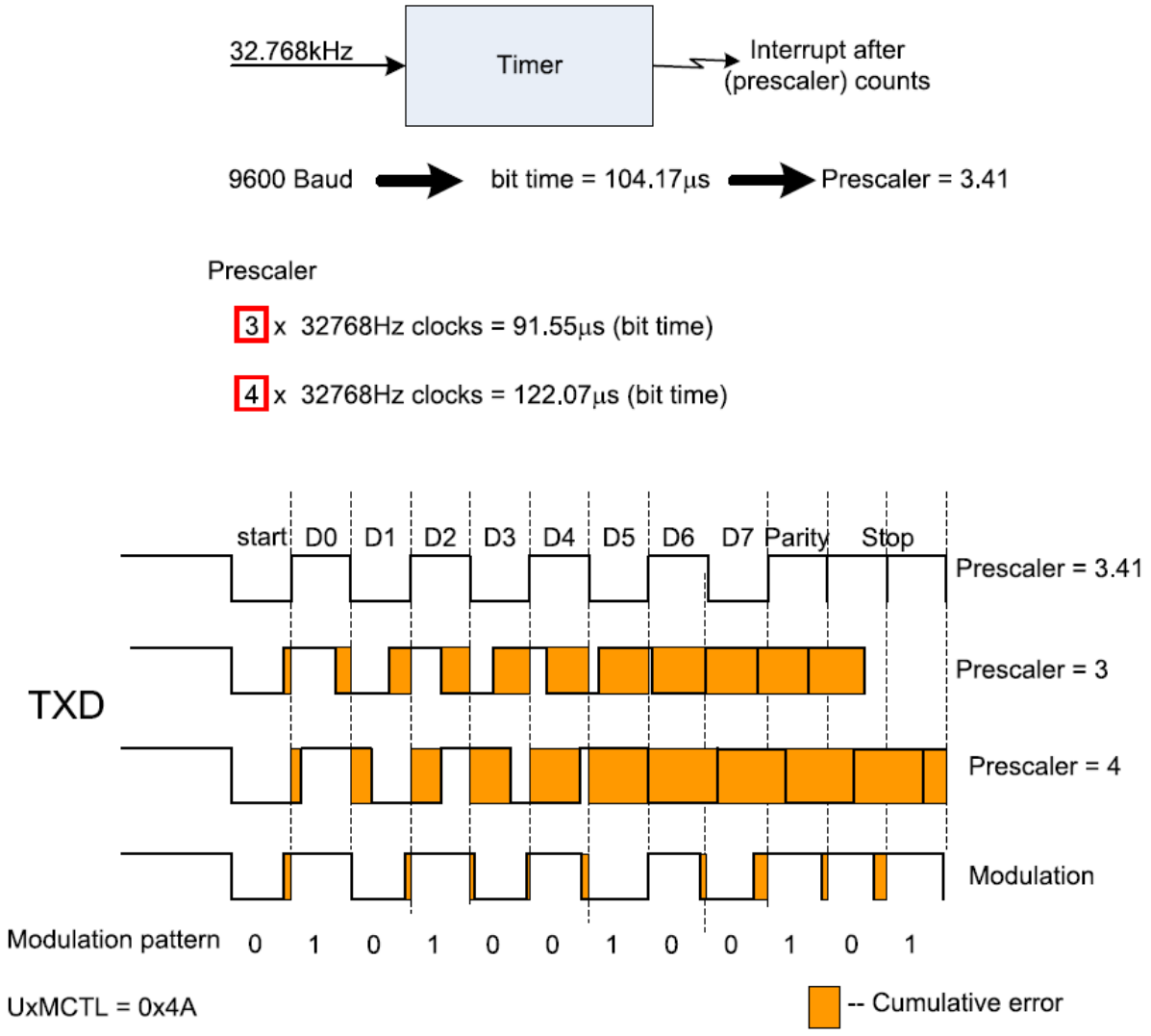
The most commonly used baud rates are shown in *Table 14-2*. The usual problem is that these baud rates are not generally sub-multiples of the unit's clock sources. The example in *Figure 14-5* illustrates this problem and shows how to solve it. The correct bit timing is obtained in two steps:

- Clock initial division through the counter;
- Using a method to correct the problem of the inexact integer division of the clock signal by the baud rate.

Table 14-2. Standardized common baud rates.

Baud Rate	Divide by		A: BRCLK = 32,768 Hz						B: BRCLK = 1,048,576 Hz				
	A:	B:	UxBR1	UxBR0	UxMCTL	Max. TX Error %	Max. RX Error %	Synchr. RX Error %	UxBR1	UxBR0	UxMCTL	Max. TX Error %	Max. RX Error %
1200	27.31	873.81	0	1B	03	−4/3	−4/3	±2	03	69	FF	0/0.3	±2
2400	13.65	436.91	0	0D	6B	−6/3	−6/3	±4	01	B4	FF	0/0.3	±2
4800	6.83	218.45	0	06	6F	−9/11	−9/11	±7	0	DA	55	0/0.4	±2
9600	3.41	109.23	0	03	4A	−21/12	−21/12	±15	0	6D	03	−0.4/1	±2
19,200		54.61							0	36	6B	−0.2/2	±2
38,400		27.31							0	1B	03	−4/3	±2
76,800		13.65							0	0D	6B	−6/3	±4
115,200		9.1							0	09	08	−5/7	±7

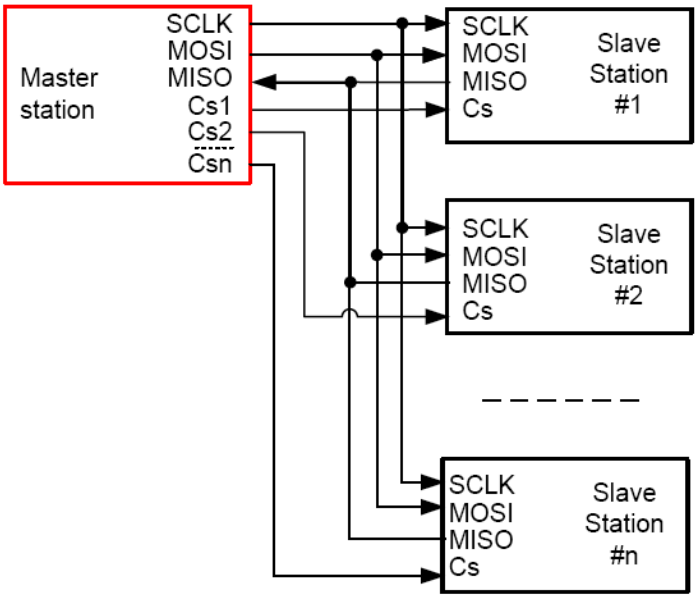
Figure 14-5. Example: Sub-multiples of the unit's clock source.



14.6 Serial Peripheral Interface (SPI) communication

The SPI (Serial Peripheral Interface) bus is a standard for synchronous serial communication developed by Motorola, which operates in full duplex mode. The devices have a master/slave relationship and the communication is always initiated by the master.

Figure 14-6. Typical SPI communication system.



The SPI communications system shown in *Figure 14-6* only supports one master, but can support more than a slave. The distance between units should be minimized, ideally limited to a single PCB. Special attention should be given to the polarity and phase of the clock signal.

14.7 I^2C (Inter-Integrated Circuit) protocol

The I^2C protocol is a multi-master synchronous serial computer bus developed by Philips Semiconductors, with the main objective of establishing links between integrated circuits and to connect low-speed peripherals.

The protocol is based on hardware using two bi-directional open-drain bus lines pulled up with resistors:

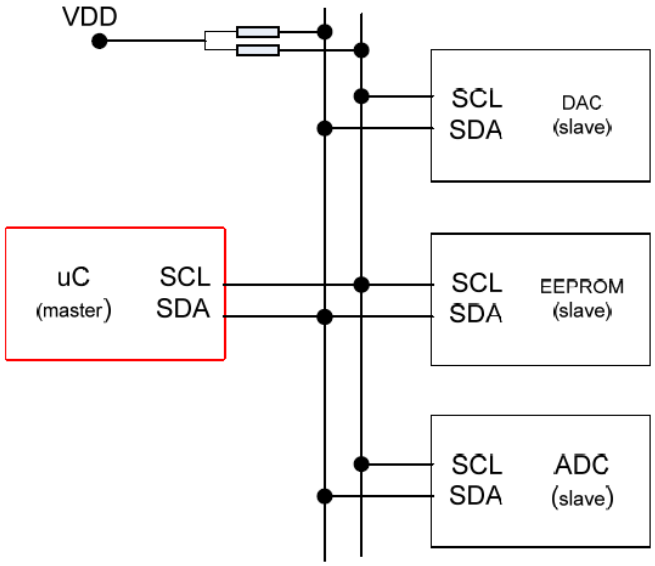
- SDA: Serial Data;
- SCL: Serial clock.

Typical voltages used are +5.0 V or +3.3 V, although systems with other voltages are possible.

The communications are always initiated and completed by the master, which is responsible for generating the clock signal. In more complex applications, the I^2C system can operate in multi-master mode. The slave selection by the master is made by the seven-bit address of the target slave.

The master (in transmit mode) sends a start bit followed by the 7-bit address of the slave it wishes to communicate with, followed by a single bit representing whether it wishes to write (0) to or read (1) to/from the slave. The target slave will acknowledge its address.

Figure 14-7. Typical I^2C communication system.



14.8 MSP430 communications interfaces

The MSP430 microcontroller family is equipped with three different types of serial communication modules:

- USART;
- USCI;
- USI.

The comparison between the three MSP430 communication modules is shown in *Table 14-3*.

Table 14-3. Comparison of MSP430 communication modes.

USART	USCI	USI
UART: - Only one modulator - n/a - n/a - n/a	**UART:** - Two modulators support n/16 timings - Auto baud rate detection - IrDA encoder & decoder - Simultaneous USCI_A and USCI_B (2 channels)	
SPI: - Only one SPI available - Master and Slave Modes - 3 and 4 Wire Modes	**SPI:** - Two SPI (one on each USCI_A and USCI_B) - Master and Slave Modes - 3 and 4 Wire Modes	**SPI:** - Only one SPI available - Master and Slave Modes
I²C: *(on '15x/'16x only)* - Master and Slave Modes - up to 400kbps	**I²C:** - Simplified interrupt usage - Master and Slave Modes - up to 400kbps	**I²C:** - SW state machine needed - Master and Slave Modes

14.8.1 USART module

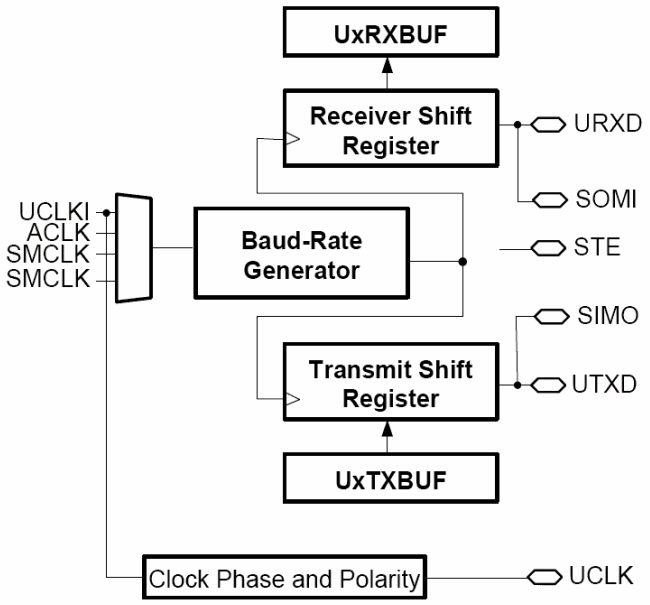
The USART (Universal Synchronous/Asynchronous Receiver/Transmitter) module is a base unit for serial communications, supporting both asynchronous communications (RS232) and synchronous communications (SPI).

The USART module is available in the 4xx series devices, particularly in the sub-series MSP430x42x and MSP430x43x. The sub-series MSP430x44x and MSP430FG461x have a second USART unit.

The USART module supports:

- Low power operating modes (with auto-start);
- UART or SPI mode (I^2C on 'F15x/'F16x only);
- Double buffered TX/RX;
- Baud rate generator;
- DMA enabled;
- Error detection.

Figure 14-8. USART block diagram.



14.8.2 USCI module

Although supporting RS232, SPI and I^2C, the USCI (Universal Serial Communication Interface) module is a communications interface designed to interconnect to industrial protocols:

- LIN (Local interconnect Network), used in cars (door modules, alarm, sunroof, etc.);
- IrDA (Infrared Data Association), used for remote controllers.

The USCI module is available in the following devices:

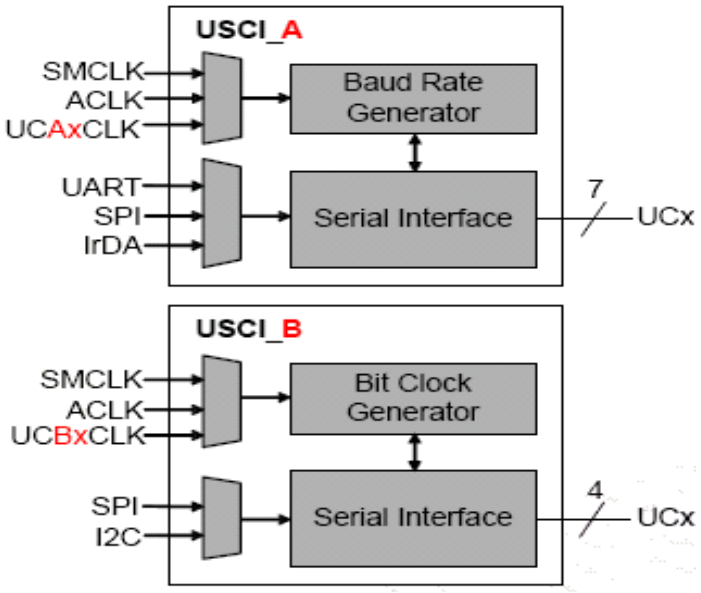
- MSP430F5xx;
- MSP430F4xx and MSP430FG461x;
- MSP430F2xx.

The USCI module supports:

- Low power operating modes (with auto-start);
- Two individual blocks:
  - USCI_A:
    - UART with Lin/IrDA support;
    - SPI (Master/Slave, 3 and 4 wire modes).
  - USCI_B:
    - SPI (Master/Slave, 3 and 4 wire mode);
    - I^2C (Master/Slave, up to 400 kHz).
- Double buffered TX/RX

- Baud rate/Bit clock generator with:
  - Auto-baud rate detect;
  - Flexible clock source.
- RX glitch suppression;
- DMA enabled;
- Error detection.

Figure 14-9. USCI block diagram.



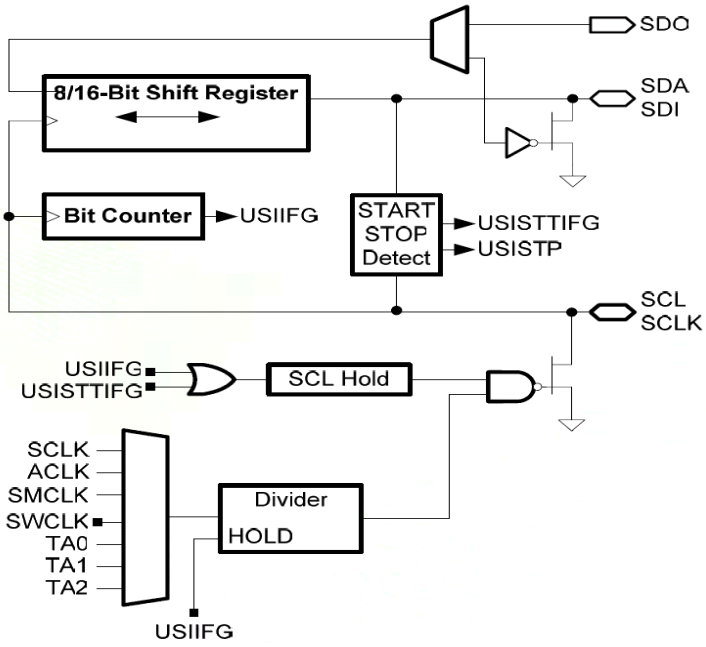
14.8.3 USI module

The USI (Universal Serial Interface) module offers basic support for synchronous serial communications SPI and I^2C. It is available in the MSP430x20xx devices family.

The USI module supports:

- SPI mode:
  - Programmable data length (8/16-bit shift register);
  - MSB/LSB first.
- I^2C mode:
  - START/STOP detection;
  - Arbitration for lost detection;
- Interrupt driven;
- Reduces CPU load;
- Flexible clock source selection;

Figure 14-10. USI block diagram.



14.9 Initialization sequence

The MSP430x4xx and MSP430x2xx MSP430 User's Guides recommended the initialization/re-configuration process for USART, USCI and USI modules as given below:

14.9.1 USART module

The recommended USART initialization/re-configuration process is:

- Set SWRST (`BIS.B #SWRST,&UxCTL`);
- Initialize all USART registers with SWRST = 1 (including UxCTL);
- Enable USART module via the MEx SFRs (URXEx and/or UTXEx);
- Clear SWRST via software (`BIC.B #SWRST,&UxCTL`);
- Enable interrupts (optional) via the IEx SFRs (URXIEx and/or UTXIEx);

14.9.2 USCI module

The recommended USCI initialization/re-configuration process is:

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

14.9.3 USI module

The recommended USI initialization process is:

- ❑ Set the USIPEx bits in the USI control register. This will select the USI function for the pin and maintains the PxIN and PxIFG functions for the pin as well;
- ❑ Set the direction of the receive and transmit shift register (MSB or LSB first) by USILSB bit;
- ❑ Select the mode (master or slave) by USIMST bit;
- ❑ Enable or disable output data by USIOE;
- ❑ Enable USI interrupts setting USIIE;
- ❑ Set up USI clock configuring the USICKCTL control register.
- ❑ Enable USI by setting USISWRST bit;
- ❑ Read port input levels via the PxIN register by software;
- ❑ Incoming data stream to generate port interrupts on data transitions.

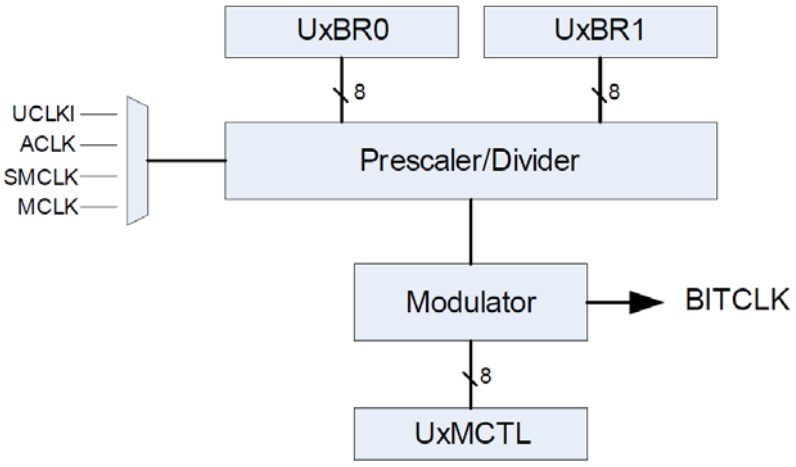
14.10 Baud Rate

The following sub-section describes the method for setting the baud rate for the supported asynchronous modules.

14.10.1 USART module

The USART module uses a prescaler/divider and a modulator as shown in *Figure 14-11*. The bit timing (BITCLK) of this module must be smaller than a third of the clock signal (BRCLK in *Figure 14-11*).

Figure 14-11. Example: USART module block diagram.



The bit timing is implemented in two stages. For the BRCLK, the division factor *N* is given by:

$$N = \frac{BRCLK}{baudrate}$$

The factor *N* may not be an integer, but its integer part will be treated by the first phase of the bit time. The fractional part in this factor will be treated by the modulator. The new definition of N is given by:

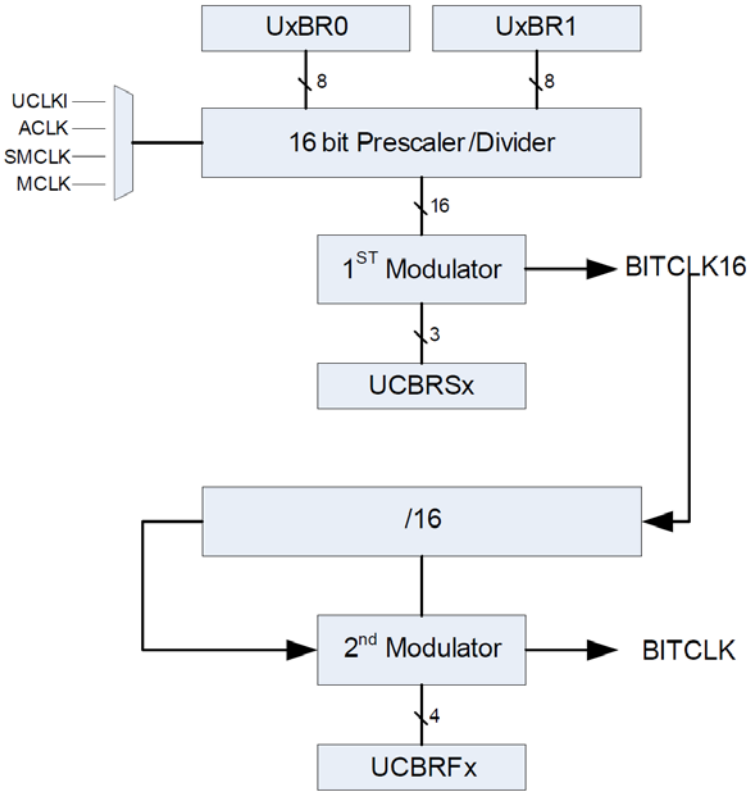
$$N = UxBR + \frac{1}{n}\sum_{i=0}^{n-1} m_i \quad [1]$$

Where:

$$baudrate = \frac{BRCLK}{N} = \frac{BRCLK}{UxBR + \frac{1}{n}\sum_{i=0}^{n-1} m_i}$$

14.10.2 USCI module

Figure 14-12. USCI module block diagram.



[1] Additional details in section 17.2.6 of the MSP430x4xx User's Guide.

For a specific clock source frequency, the divider value is given by:

$$N = \frac{BRCLK}{baudrate}$$

Typically, the value of *N* is not an integer value, so it is necessary to use a modulator.

The USCI module has two ways to generate the baud rate:

Low-Frequency Baud Rate Generation

The "Low-Frequency Baud Rate Generation" mode is selected when UCOS16 = 0. The baud rate generation mode is useful for lowering power consumption, since it uses a low frequency clock source (32.768 kHz crystal).

The baud rate is obtained through a prescaler and a modulator, similar to the one used in the USART module.

The registers are configured using the equations given below. However, for confirmation, it is recommended that the tables given in the User's Guide be consulted.

$$UCBRx = \text{int}(N) \times prescaler$$

$$UCBRSx = round((N - \text{int}(N)) \times 8) \times (fractional\ portion)$$

Oversampling Baud Rate Generation

The "Oversampling Baud Rate Generation" mode is selected when UCOS16 = 1. This mode allows precise bit timing. It requires clock sources 16x higher than the desired baud rate.

The baud rate is generated in two steps:

The clock source is divided by 16, and results in the BITCLK16, being the source signal divided by the prescaler, and is applied to the first modulator;

The BITCLK is defined by BITCLK16, through division by 16 and a second modulator.

The registers are configured using the equations given below. However, for confirmation, it is recommended that the tables given in the User's Guide be consulted.

$$UCBRx = \text{int}\left(N/16\right) \times prescaler$$

$$UCBRFx = round\left(\left(\frac{N}{16} - \text{int}\left(\frac{N}{16}\right)\right) \times 16\right) \times (\textit{fractional portion})$$

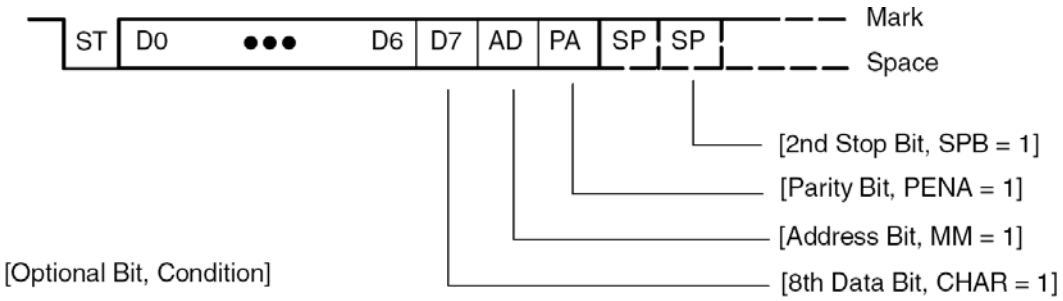
14.11 Serial communication modes operation

The following sub-sections describe the different operating modes of the communication interfaces supported by the MSP430.

14.11.1 USART operation: UART mode

- Transmits and receives characters at an asynchronous bit rate to/from other devices;
- Timing for each character is based on the selected baud rate;
- Transmit and receive functions use the same baud rate frequency;
- Initialization follows the sequence given earlier;
- Define the character format for the sequence of characters:
  - Start bit;
  - Seven or eight data bits;
  - Even/odd/no parity bit;
  - Address bit (address-bit mode);
  - One or two stop bits.

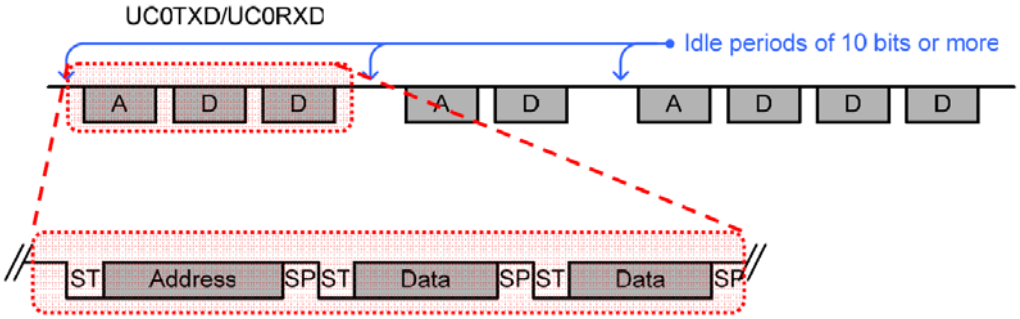
Figure 14-13. Character format.



- Define the asynchronous communication protocol:
  - Idle-line multiprocessor communication protocol for a minimum of two devices (see *Figure 14-14*):
    - IDLE is detected after > 10 periods of continuous marks after the stop bit;
    - The first character after IDLE is an address;

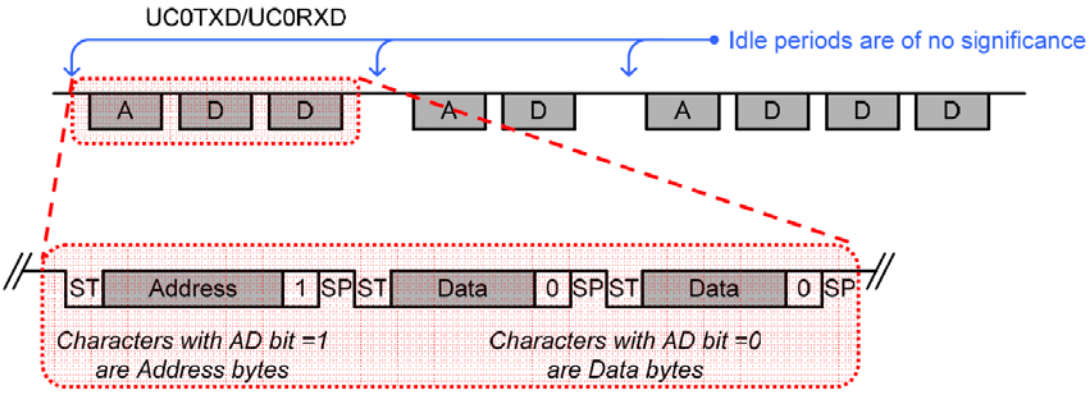
- UART can be programmed to receive only address characters.

Figure 14-14. UART idle-line multiprocessor communication protocol.



- Address-bit multiprocessor communication protocol for a minimum of three devices (see *Figure 14-15*):
  - Extra bit state in the received character marks an address character;
  - UART can be programmed to receive only address characters.

Figure 14-15. UART address-bit multiprocessor communication protocol.



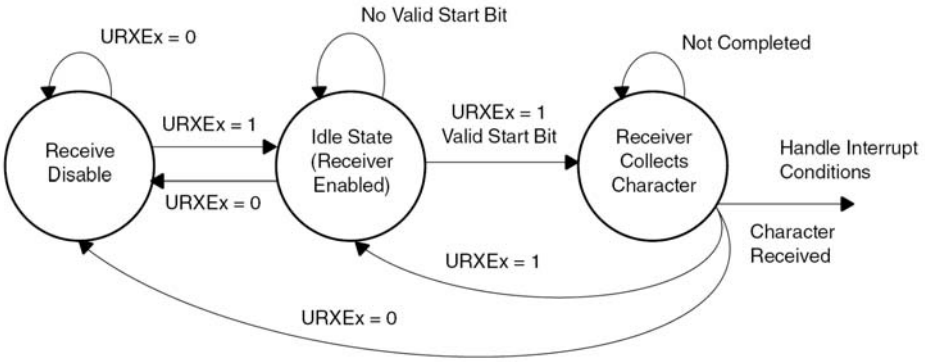
- Automatic error detection:
  - Framing error FE:
    - FE is set if the stop bit is missing from a received frame.
  - Parity error PE:
    - PE is set if there is a parity mismatch in a received frame.
  - Receive overrun error OE:
    - OE is set if UxRXBUF is overwritten.

- Break condition BRK:
  - BRK is set if all bits in the received frame = 0;
- Glitch suppression prevents the USART from being accidentally started;
- Any pulse on UxRXD shorter than the deglitch time (approximately 300 ns) will be ignored.

❑ Enable the USART receive enable bit URXEx:

- The receive-data buffer, UxRXBUF, contains the character transferred from the RX shift register after the character is received.

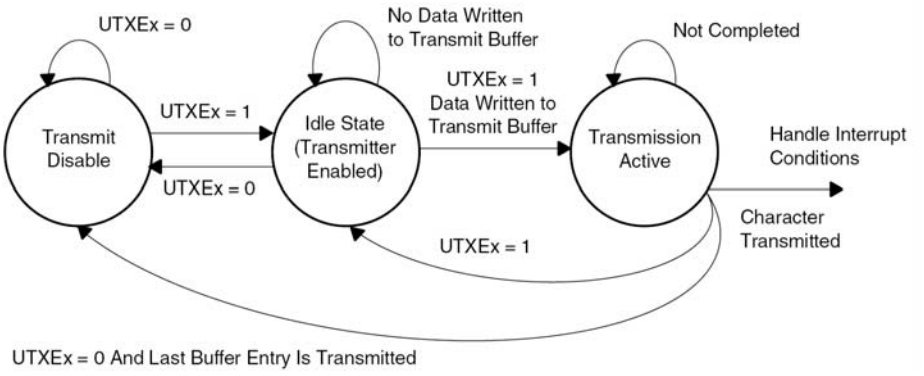
Figure 14-16. State diagram of receive enable.



❑ Enable the USART transmit enable bit UTXEx:

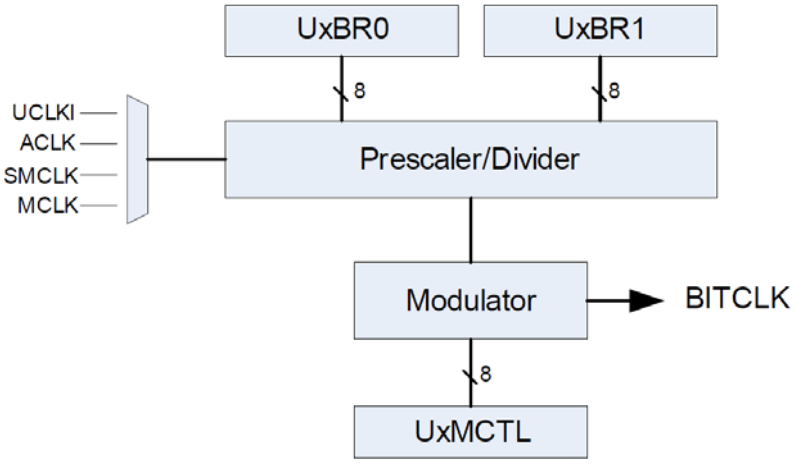
- Transmission is initiated by writing data to UxTXBUF;
- The data is then moved to the transmit shift register on the next BITCLK after the TX shift register is empty, and transmission begins.

Figure 14-17. State diagram of transmit enable.



❑ Define the USART baud rate generation (standard baud rates from non-standard source frequencies) as described earlier (See *Figure 14-18*);

Figure 14-18. USART operation: Baud rate generator.



❑ Set USART interrupts (one interrupt vector for transmission and one interrupt vector for reception):

- UART transmit interrupt operation:
  - UTXIFGx interrupt flag is set by the transmitter to indicate that UxTXBUF is ready to accept another character;
  - An interrupt request is also generated if UTXIEx and GIE are set;
  - UTXIFGx is automatically reset if the interrupt request is serviced or if a character is written to UxTXBUF.
- UART receive interrupt operation:
  - URXIFGx interrupt flag is set each time a character is received and loaded into UxRXBUF;
  - An interrupt request is also generated if URXIEx and GIE are set;
  - URXIFGx and URXIEx are reset by a system reset PUC signal or when SWRST = 1;
  - URXIFGx is automatically reset if the pending interrupt is serviced (when URXSE = 0) or when UxRXBUF is read.

❑ Receive-start edge detect feature (URXSE bit). Should be used when:

- BRCLK is sourced by the DCO;
- DCO is off due to low-power mode operation.

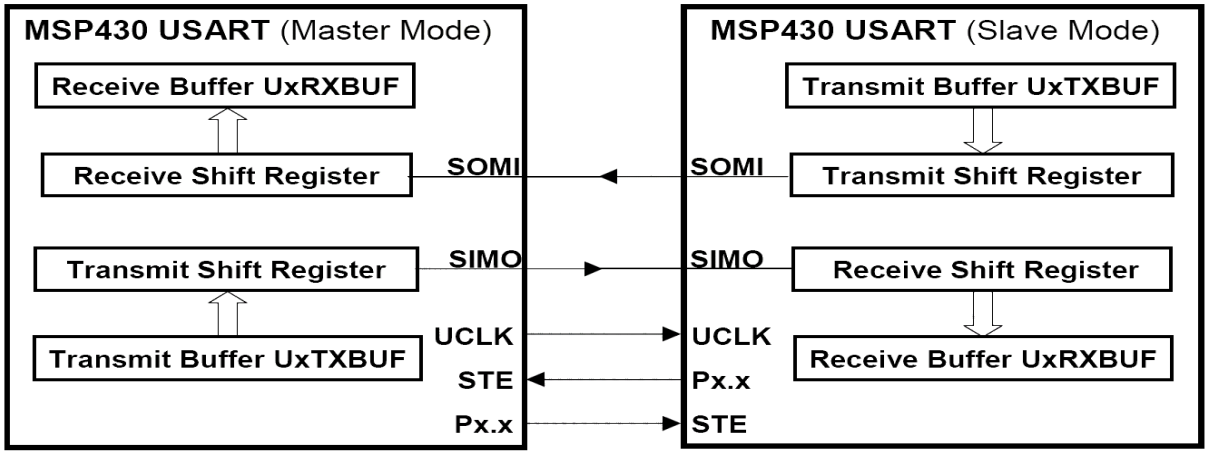
14.11.2 USART operation: SPI mode

- Serial data transmitted and received by multiple devices using a shared clock provided by the master;

- STE bit (controlled by the master) enables a device to receive and transmit data;

- Three or four signals are used for SPI data exchange (see *Figure 14-19*):
  - SIMO: Slave In, Master Out;
  - SOMI Slave Out, Master In;
  - UCLK USART SPI clock;
  - STE slave transmit enable.

Figure 14-19. USART operation in SPI mode.



- Initialization follows the sequence given earlier;

- Define mode: Master or Slave;

- Enable SPI transmit/receive, USPIEx;

Figure 14-20. State diagram of transmit enable for SPI master mode.

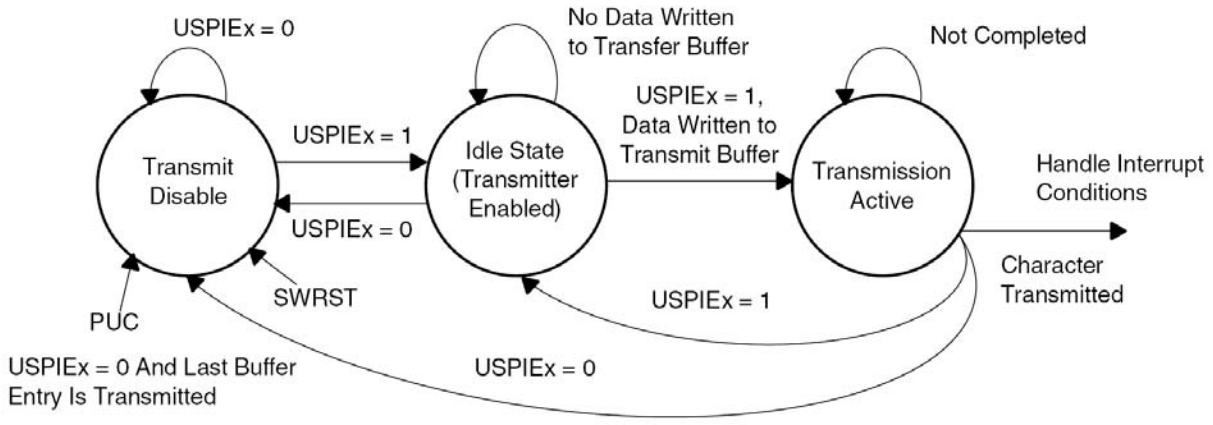


Figure 14-21. State diagram of transmit enable for SPI slave mode.

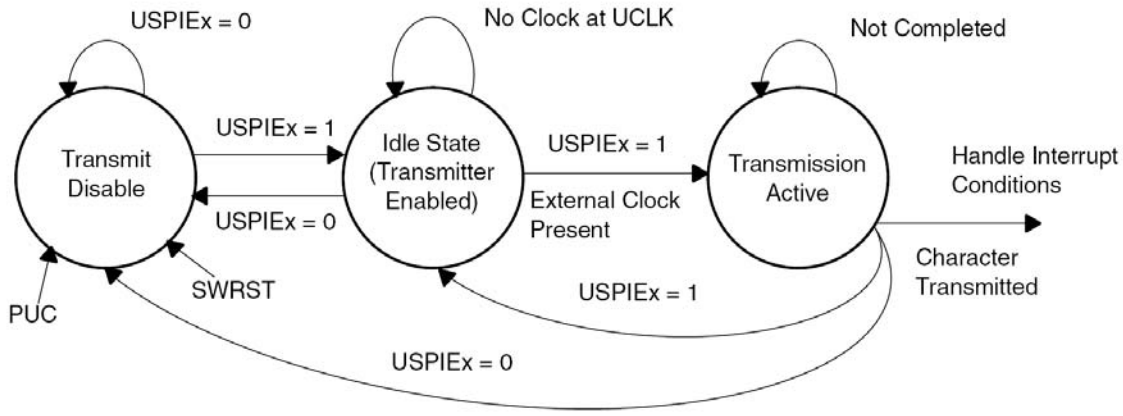


Figure 14-22. State diagram of receive enable for SPI master mode.

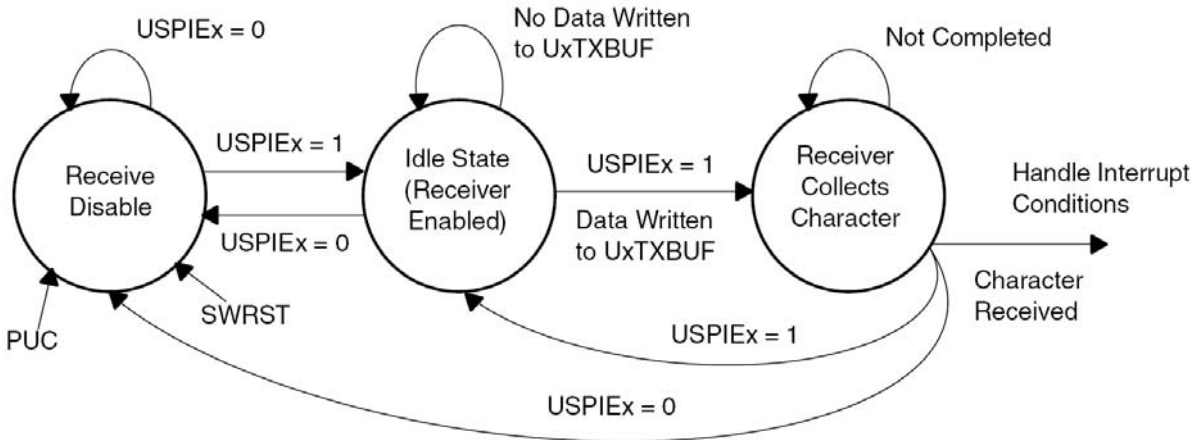
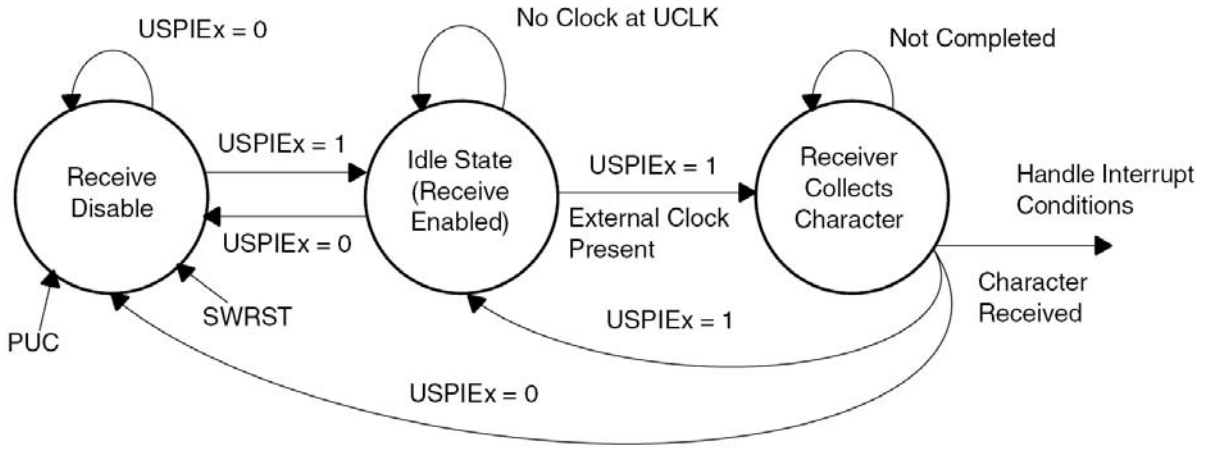


Figure 14-23. State diagram of receive enable for SPI slave mode.



- Define serial clock control:
  - UCLK is provided by the master on the SPI bus.

- Define serial clock polarity (CKPL bit) and phase (CKPH bit);

- Set USART interrupts (one interrupt vector for transmission and one interrupt vector for reception):
  - SPI transmit interrupt operation (as UART mode);
  - SPI receive interrupt operation (as UART mode).

14.11.3 USCI operation: UART mode

- In asynchronous mode, the USCI_Ax modules connect the MSP430 to an external system via two external pins, UCAxRXD and UCAxTXD;

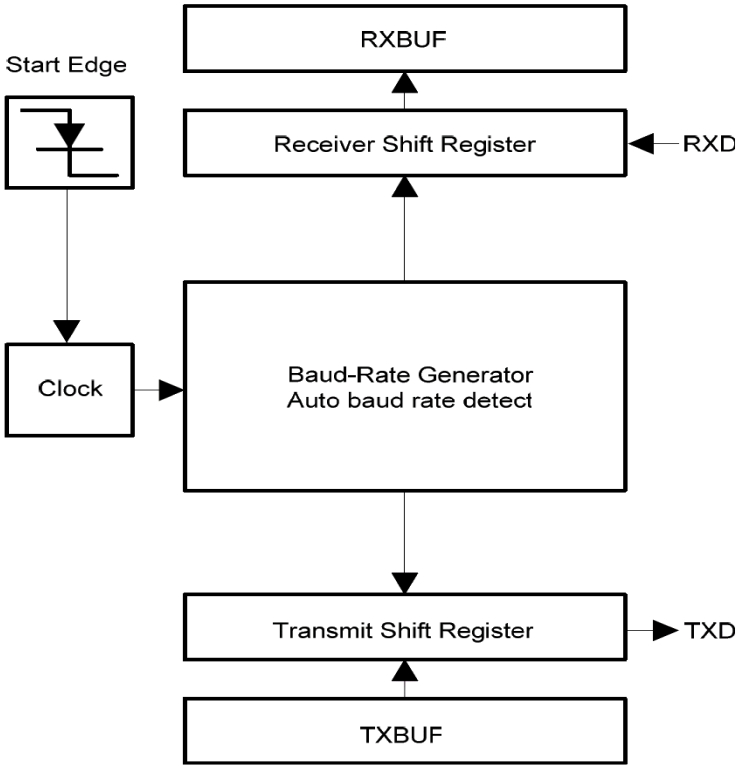
- UART mode is selected when the UCSYNC bit is cleared;

- USCI transmits and receives characters asynchronously at a bit rate the same as other devices;

- Timing for each character is based on the selected baud rate of the USCI;

- The transmit and receive functions use the same baud rate frequency;

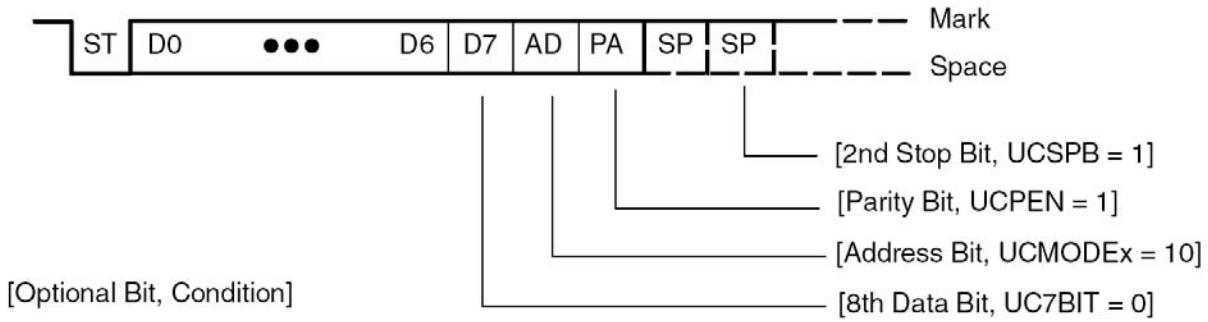
Figure 14-24. USCI operation in UART mode.



- Initialization follows the sequence given earlier;

- Define the character format specified as USART in UART mode:
  - UCMSB bit controls the direction of the transfer and selects LSB (usual in UART communication) or MSB first.

Figure 14-25. Character format in USCI operation: UART mode.

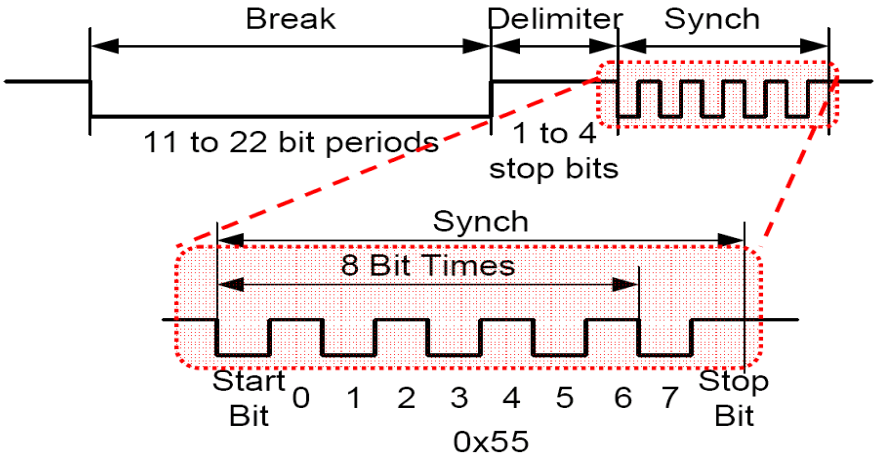


- Define the asynchronous communication format as USART in UART mode;

❑ If appropriate, define automatic baud rate detection (UCMODEx = 11) as shown in *Figure 14-26*:

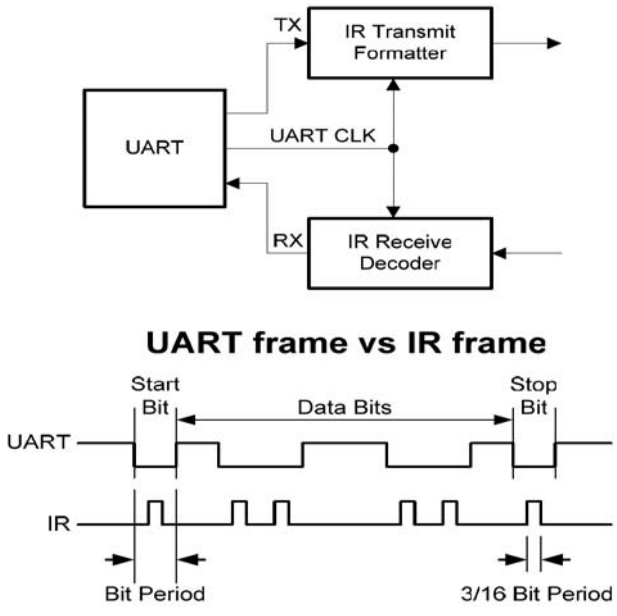
- Data frame is preceded by a synchronization sequence:
  - Break: Detected when 11 or more continuous zeros (spaces) are received;
  - Synch field: Data 055h inside a byte field.
- The baud rate is calculated from a valid SYNC;
- Auto baud rate value stored in UxBR1, UxBR0 and UxMCTL (modulation pattern);
- BREAK time-out detect in hardware;
- Programmable delimiter time;

Figure 14-26. USCI automatic baud rate detection.



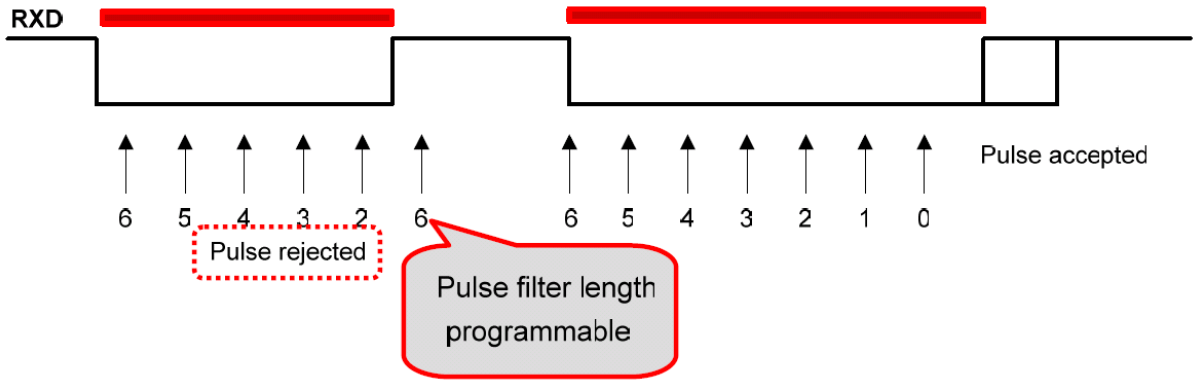
❑ If appropriate, use the IrDA encoder and decoder (UCIREN = 1), as shown in *Figure 14-27*:

Figure 14-27. USCI in UART mode: IrDA operation.



- IrDA encoding:
  - Encoder sends a pulse for every zero bit in the transmit bit stream coming from the UART;
  - Pulse duration (defined by UCIRTXPLx bits) specifies the number of half clock periods of the clock (UCIRTXCLK);
  - Oversampling baud rate generator allows the selection of the IrDA standard 3/16 bit length.
- IrDA decoding:
  - Programmable low (see *Figure 14-28*) or high pulse detection (UCIRRXPL) by the decoder;
  - Programmable received pulse length filter adds noise filter capability in addition to the glitch detector.

Figure 14-28. USCI in UART mode: low pulse detection - IrDA decoding operation.



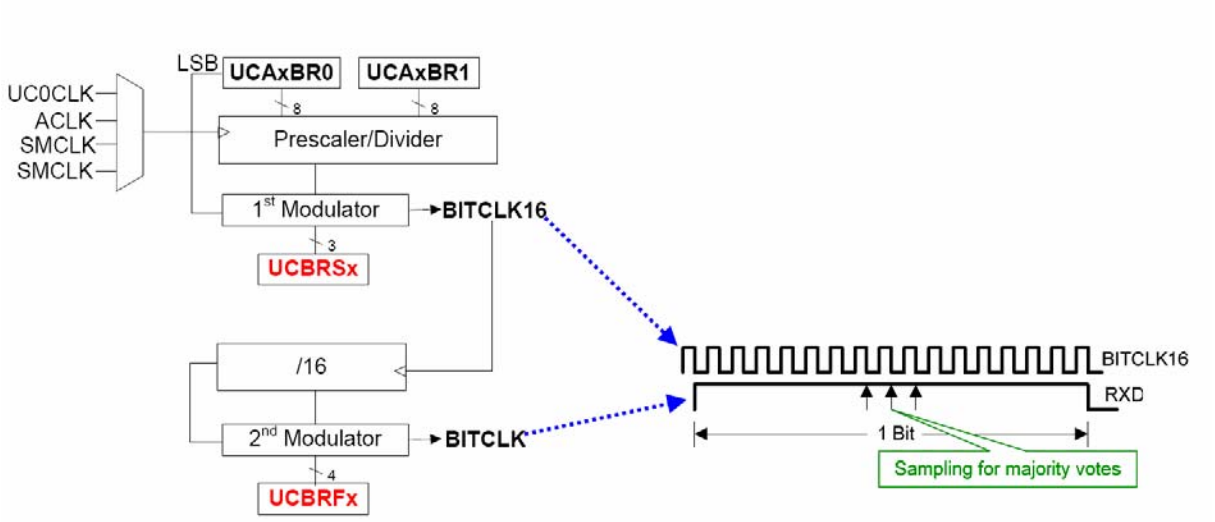
- Automatic error detection:
  - Framing error UCFE:
    - UCFE is set if the stop bit is missing from a received frame.
  - Parity error UCPE:
    - UCPE is set if there is parity mismatch in the received frame.
  - Receive overrun error UCOE:
    - UCOE is set if RXBUF is overwritten.
  - Break condition UCBRK:
    - UCBRK is set if all bits in the received frame = 0;
    - If UCBRKIE is set, then UCAxRXIFG is set if BRK=1.
  - Glitch suppression prevents the USCI from being accidentally started;
  - Any pulse on UCAxRXD shorter than the deglitch time (approximately 150 ns) will be ignored.
  - UART can be programmed to transfer only error free characters to UCAxRXBUF.

- USCI receive enable: Clear UCSWRST;
  - The falling edge of the start bit enables the baud rate generator;
  - If a valid start bit is detected, a character will be received.

- USCI transmit enable: Clear UCSWRST;
  - Transmission is initiated by writing data to UCAxTXBUF;
  - The baud rate generator is enabled;
  - The data in UCAxTXBUF is moved to the transmit shift register on the next BITCLK after the transmit shift register is empty;
  - UCAxTXIFG is set when new data can be written to UCAxTXBUF.

- Define UART baud rate generation (standard baud rates from non-standard source frequencies) as given earlier (see *Figure 14-29*). Two modes of operation (UCOS16 bit):
  - Low-frequency baud rate;
  - Oversampling baud rate.

Figure 14-29. USCI operation: Baud rate generator.



- Transmit bit timing:
  - The timing for each character is the sum of the individual bit timings;
  - The modulation feature of the baud rate generator reduces the cumulative bit error.

- Two error sources for receive bit timing:
  - Bit-to-bit timing error;
  - Error between a start edge occurring and the start edge being accepted by the USCI module.

- Set USCI interrupts (one interrupt vector for transmission and one interrupt vector for reception):
  - USCI transmit interrupt:
    - UCAxTXIFG interrupt flag is set by the transmitter to indicate that UCAxTXBUF is ready to accept another character;
    - An interrupt request is generated if UCAxTXIE and GIE are also set;
    - UCAxTXIFG is automatically reset if a character is written to UCAxTXBUF.
  - USCI receive interrupt:
    - UCAxRXIFG interrupt flag is set each time a character is received and loaded into UCAxRXBUF;
    - An interrupt request is also generated if UCAxRXIE and GIE are set;
    - UCAxRXIFG and UCAxRXIE are reset by a system reset PUC signal or when UCSWRST = 1;
    - UCAxRXIFG is automatically reset when UCAxRXBUF is read.

14.11.4 USCI operation: SPI mode

- Flexible interface:
  - 3- or 4-pin SPI;
  - 7- or 8-bit data length;
  - Master or slave;
  - LSB or MSB first.
- S/W configurable clock phase and polarity;
- Programmable SPI master clock;
- Double buffered TX/RX;
- Interrupt driven TX/RX (USCI_A and USCI_B share TX and RX vector);
- DMA enabled;
- LPMx operation.

Figure 14-30. USCI operation: SPI mode.

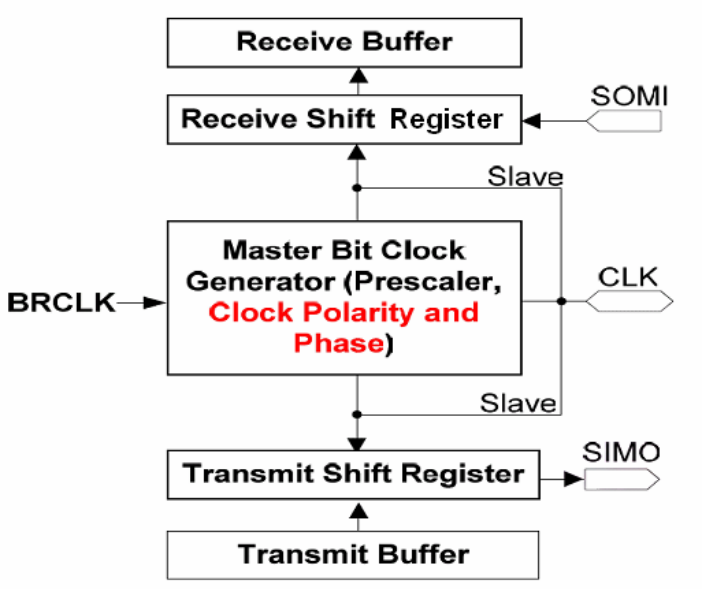
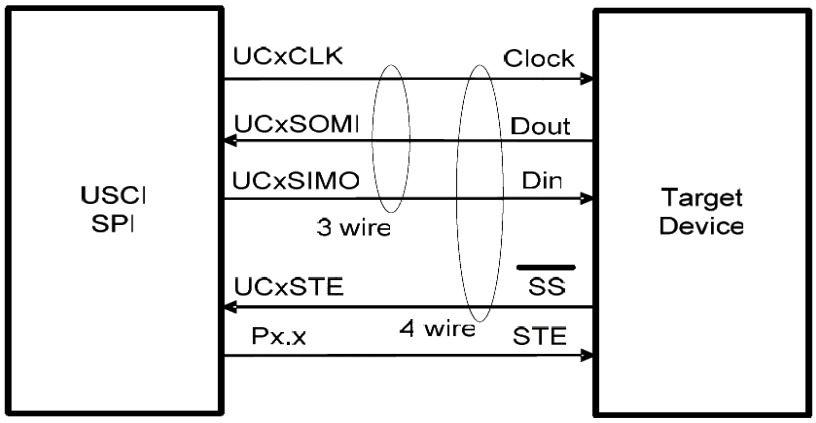


Figure 14-31. USCI operation: SPI connections.



❑ Serial data is transmitted and received by multiple devices using a shared clock signal that is generated by the master;

❑ Three or four signals are used for SPI data exchange:

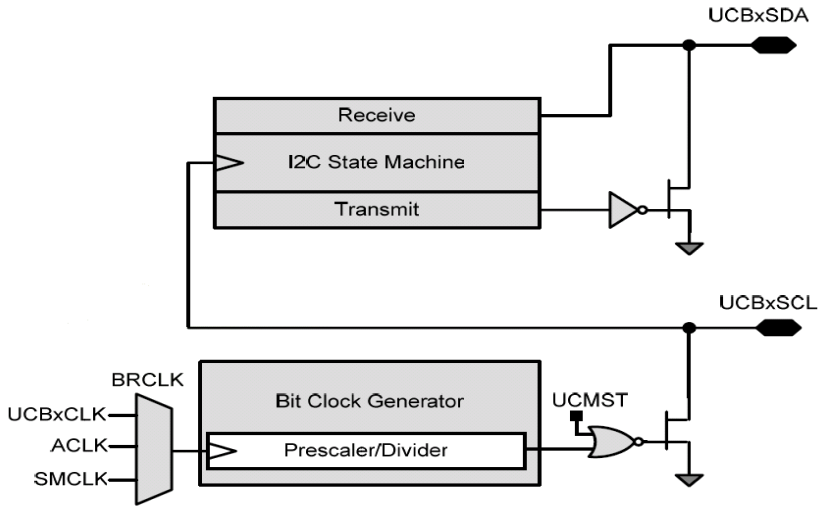
- UCxSIMO: Slave in, master out;
- UCxSOMI: Slave out, master in;
- UCxCLK: USCI SPI clock;
- UCxSTE: Slave transmit enable:
  - Enables a device to receive and transmit data and is controlled by the master;
  - 4 wire master, senses conflicts with other master(s);
  - 4 wire slave, externally controls TX and RX.

- Initialization follows the sequence given earlier;
- Define the character format as given earlier;
- Define mode: Master or Slave;
- Enable SPI transmit/receive by clearing the UCSWRST bit:
- Define serial clock control:
  - UCxCLK is provided by the master on the SPI bus;
  - Configure serial clock polarity and phase (UCCKPL and UCCKPH bits).
- Set USCI interrupts (one interrupt vector for transmit and one interrupt vector for receive):
  - SPI transmit interrupt operation:
    - UCxTXIFG interrupt flag is set by the transmitter to indicate that UCxTXBUF is ready to accept another character;
    - An interrupt request is also generated if UCxTXIE and GIE are set;
    - UCxTXIFG is automatically reset if the interrupt request is serviced or if a character is written to UCxTXBUF.
  - SPI receive interrupt operation.
    - UCxRXIFG interrupt flag is set each time a character is received and loaded into UCxRXBUF;
    - An interrupt request is also generated if UCxRXIE and GIE are set;
    - UCxRXIFG and UCxRXIE are reset by a system reset PUC signal or when SWRST = 1;
    - UCxRXIFG is automatically reset if the pending interrupt is serviced (when UCSWRST = 1) or when UCxRXBUF is read.

14.11.5 USCI operation: I^2C mode

- The I^2C mode supports any I^2C compatible master or slave device (specification v2.1);

Figure 14-32. USCI operation: I²C mode.

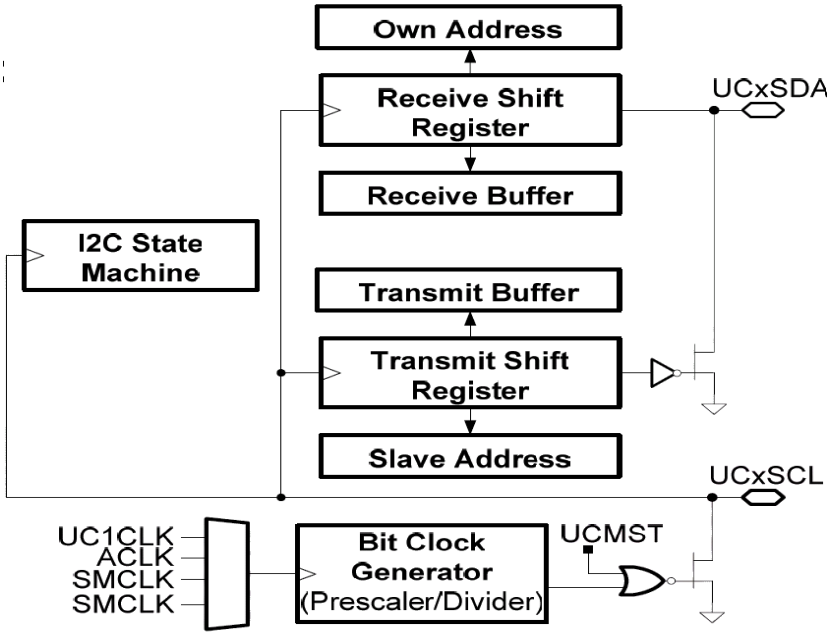


- Each I²C device is identified by a unique address and can operate either as a transmitter or a receiver, and either as the master or the slave;

- A master initiates data transfers and generates the clock signal SCL. Any device addressed by a master is taken to be a slave;

- Communication uses the bi-directional serial data (SDA) and serial clock (SCL) pins;

Figure 14-33. USCI operation: I²C block diagram.



- Initialization follows the sequence given earlier;

- I^2C serial data:
  - One clock pulse is generated by the master device for each data bit transferred;
  - Operates with byte data (MSB transferred first);
  - The first byte after a START condition consists of a 7-bit slave address and a R/W bit:
    - R/W = 0: Master transmits data to a slave;
    - R/W = 1: Master receives data from a slave.
  - The acknowledge (ACK) bit is sent from the receiver after each byte on the 9th SCL clock;

- I^2C addressing modes (7-bit and 10-bit addressing modes);

- I^2C module operating modes:
  - Master transmitter;
  - Master receiver;
  - Slave transmitter;
  - Slave receiver.

- An arbitration procedure is invoked if two or more master transmitters simultaneously start a transmission on the bus;

- I^2C Clock generation and synchronization:
  - SCL is provided by the master on the I^2C bus;
  - Master mode: BITCLK is provided by the USCI bit clock generator;
  - Slave mode: the bit clock generator is not used.

- Set USCI interrupts (one interrupt vector for transmission and one interrupt vector for reception):
  - I^2C transmit interrupt operation:
    - UCBxTXIFG interrupt flag is set by the transmitter to indicate that UCBxTXBUF is ready to accept another character;
    - An interrupt request is also generated if UCBxTXIE and GIE are set;
    - UCBxTXIFG is automatically reset if a character is written to UCBxTXBUF or a NACK is received.

- I^2C receive interrupt operation.
  - UCBxRXIFG interrupt flag is set each time a character is received and loaded into UCxRXBUF;
  - An interrupt request is also generated if UCBxRXIE and GIE are set;
  - UCBxRXIFG and UCBxRXIE are reset by a system reset PUC signal or when SWRST = 1;
  - UCxRXIFG is automatically reset when UCBxRXBUF is read.

- I^2C state change interrupt flags:
  - Arbitration-lost, UCALIFG: Flag set when two or more transmitters start a transmission simultaneously, or a device operates as master, but is addressed as a slave by another master;
  - Not-acknowledge interrupt, UCNACKIFG: Flag set when an acknowledge is expected but is not received;
  - Start condition detected interrupt, UCSTTIFG: Flag set when the I^2C module detects a START condition, together with its own address while in slave mode;
  - Stop condition detected interrupt, UCSTPIFG: Flag set when the I^2C module detects a STOP condition while in slave mode.

14.11.6 USI operation (SPI and I^2C modes)

- Shift register and bit counter that includes logic to support SPI and I^2C communication;

- USISR shift register (up to 16 bits supported):
  - Directly accessible by software;
  - Contains the data to be transmitted or the data received (TX and RX is simultaneous);
  - MSB or LSB first.

- Bit counter:
  - Controls the number of TX or RX bits;
  - Counts the number of sampled bits;
  - Sets the USI interrupt flag USIIFG when the USICNTx value becomes zero (decrementing or writing zero to USICNTx bits);
  - Writing USICNTx > 0 automatically clears USIIFG when USIIFGCC = 0 (automatically stops clocking after the last bit).

- USI initialization:
  - Reset USISWRST;
  - Set USIPEx bits (USI function for the pin and maintains the PxIN and PxIFG functions for the pin):
    - Port input levels can be read from the PxIN register by software;
    - The incoming data stream can generate port interrupts on data transitions.

- USI clock generation (see *Figure 14-34*):
  - Clock selection multiplexer:
    - Internal clocks ACLK or SMCLK;
    - External clock SCLK;
    - USISWCLK (software clock input bit);
    - Timer_A CAP/COM outputs.
  - Configurable divider;
  - Auto-stop on interrupt: USIIFG;
  - Selectable phase and polarity.

Figure 14-34. USI clock generator block diagram.

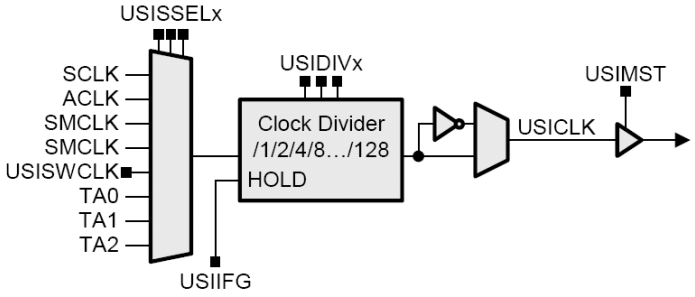
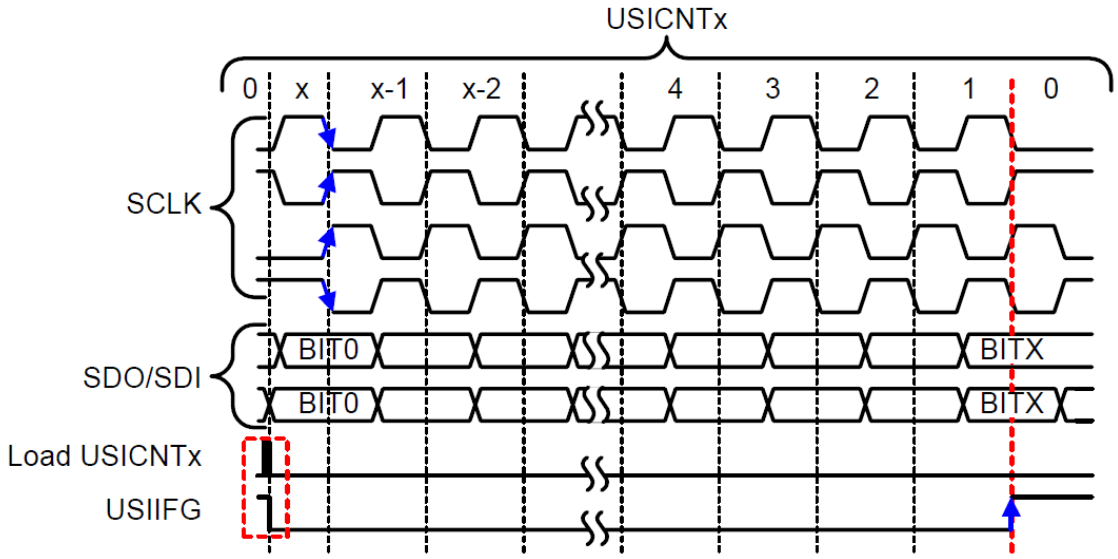


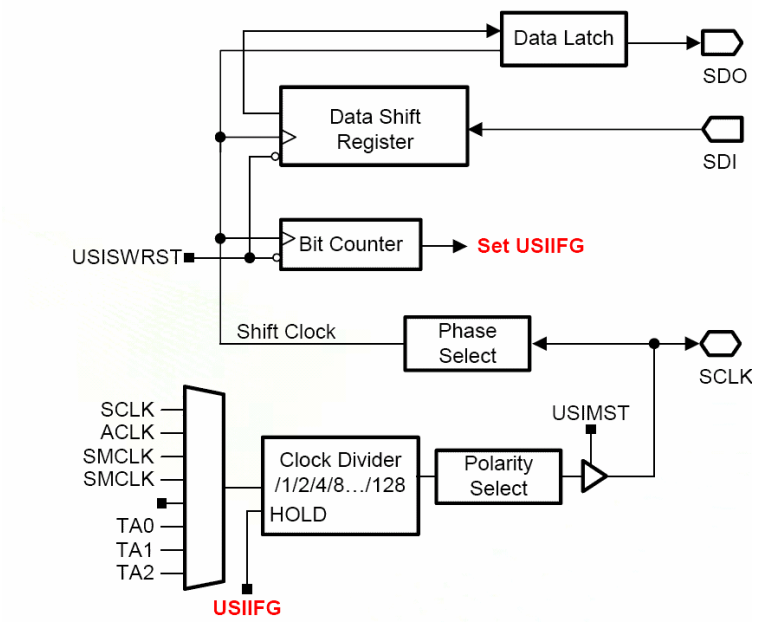
Figure 14-35. USI operation- SPI mode: Clock and data handling.



SPI mode

- Configure USI module in SPI mode (USII2C = 0);

Figure 14-36. USI – SPI mode block diagram.



- Configure USICKPL. Selects the inactive level of the SPI clock (rising or falling edge data latching);

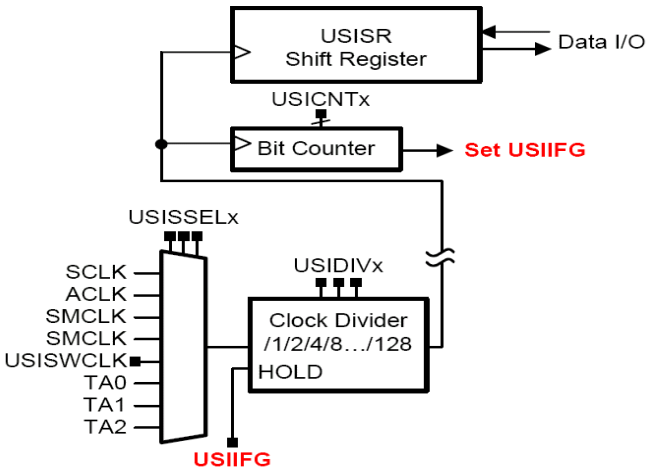
- Configure USICKPH. Selects the clock edge on which SDO is updated and SDI is sampled (idle high or low supported).

- Configure mode:
  - SPI master:
    - Set USIMST bit and clear USII2C bit;
    - Select clock source;
    - Configure SCLK as output.
  - SPI slave:
    - Clear the USIMST and the USII2C bits;
    - SCLK is automatically configured as an input;
    - Receives the clock externally from the master.

- SPI interrupts:
  - One interrupt vector associated with the USI module;
  - One interrupt flag, USIIFG:
    - Set when bit counter counts to zero;

- Generates an interrupt request when USIIE = 1;
- Cleared when USICNTx > 0 (USIIFGCC = 0), or directly by software;
- Stops the clock when set.

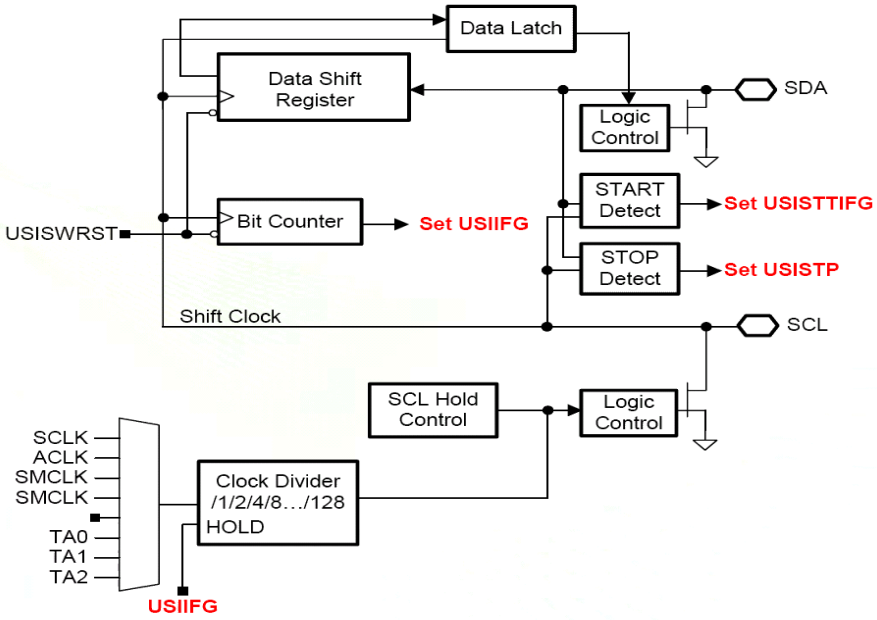
Figure 14-37. USI interrupts- SPI mode.



I²C mode

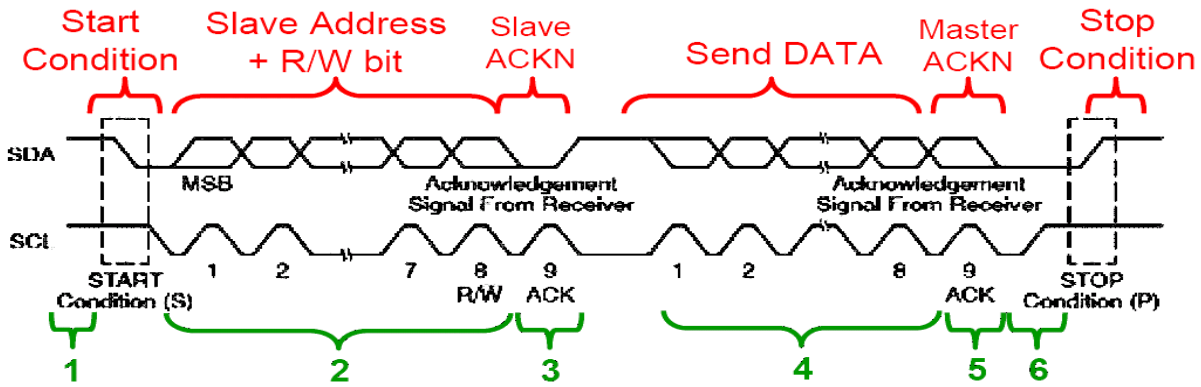
- Configure USI module in I²C mode (USII2C =1, USICKPL = 1, and USICKPH = 0);
- Clear USILSB and USI16B (I²C data compatibility);
- Set USIPE6 and USIPE7 (enables SCL and SDA port functions);

Figure 14-38. USI operation- I²C mode.



- Configure mode:
  - I^2C master:
    - Set USIMST and USII2C bits;
    - Select clock source (output to SCL line while USIIFG = 0).
  - I^2C slave:
    - Clear the USIMST;
    - SCL is held low if USIIFG = 1, USISTTIFG = 1 or if USICNTx = 0.
  - I^2C transmitter:
    - Data value is first loaded into USISRL;
    - Set USIOE to enable output and start transmission (writes 8 into USICNTx);
    - Send Start (or repeated Start);
    - Define address and set R/W;
    - Slave ACK: (Data TX/RX + ACK for N bytes);
    - SCL is generated in master mode or released from being held low in slave mode;
    - USIIFG is set after the transmission of all 8 bits (stops clock signal on SCL in master mode or held low at the next low phase in slave mode);
    - Stop (or repeated Start).

Figure 14-39. USI operation- I^2C mode: Clock and data handling.

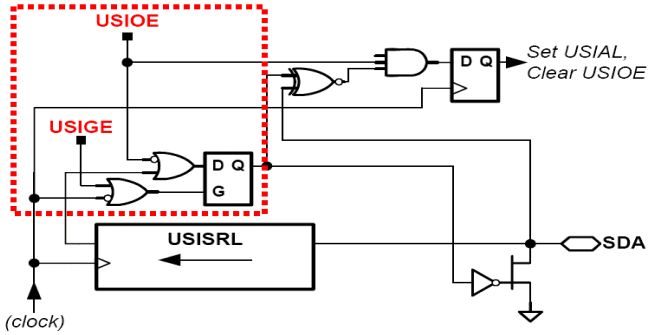


  - I^2C receiver:
    - Clear USIOE (disable output);
    - Receive by writing 8 into USICNTx (USIIFG = 0);
    - SCL is generated in master mode or released from being held low in slave mode;

- USIIFG is set after 8 clocks (stops the clock signal on SCL in master mode or holds SCL low at the next low phase in slave mode).

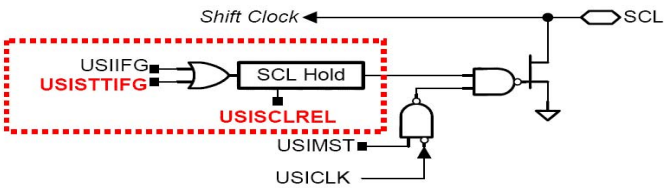
- SDA configuration:
  - Direction;
  - Used for TX/RX, ACK/NACK handling and START/STOP generation;
  - USIGE: Output latch control;
  - USIOE: Data output enable.

Figure 14-40. USI operation- I²C mode: SDA control.



- SCL control:
  - SCL automatically held low in slave mode if USIIFG = 1 or USISTTIFG = 1;
  - Requires I²C compliant master supporting clock stretching;
  - SCL can be released by software with USISCLREL = 1.

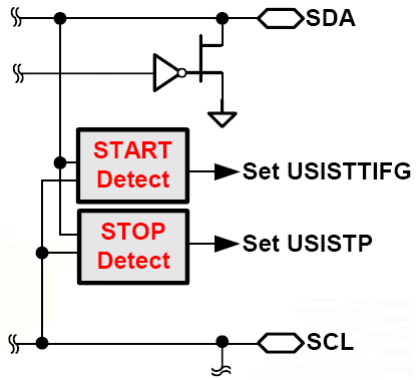
Figure 14-41. USI operation- I²C mode: SCL control.



  - START condition (high-to-low transition on SDA while SCL is high);
    - Clear MSB of the shift register;
    - USISTTIFG set on start (Sources USI interrupt).

- STOP condition (low-to-high transition on SDA while SCL is high):
  - Clear the MSB in the shift register and load 1 into USICNTx (finishes the acknowledgment bit and pull SDA low);
  - USISTP set on stop (CPU-accessible flag).

Figure 14-42. USI operation- I²C mode: Start/Stop detection.



- Receiver ACK/NACK generation:
  - After address/data reception;
  - SDA = output;
  - Output 1 data bit: 0 = ACK, 1 = NACK.
- Transmitter ACK/NACK Detection:
  - After address/data transmission;
  - SDA = input;
  - Receive 1 data bit: 0 = ACK, 1 = NACK.

❑ Arbitration procedure (in multi-master I²C systems);

❑ I²C Interrupts:

- One interrupt vector associated with the USI;
- Two interrupt flags, USIIFG and USISTTIFG;
- Each interrupt flag has its own interrupt enable bit, USIIE and USISTTIE;
- When an interrupt is enabled, and the GIE bit is set, a set interrupt flag will generate an interrupt request;
- USIIFG is set (USICNTx = 0);
- USISTTIFG is set (START condition detection).

Figure 14-43. USI operation- I²C mode: Interrupts management.

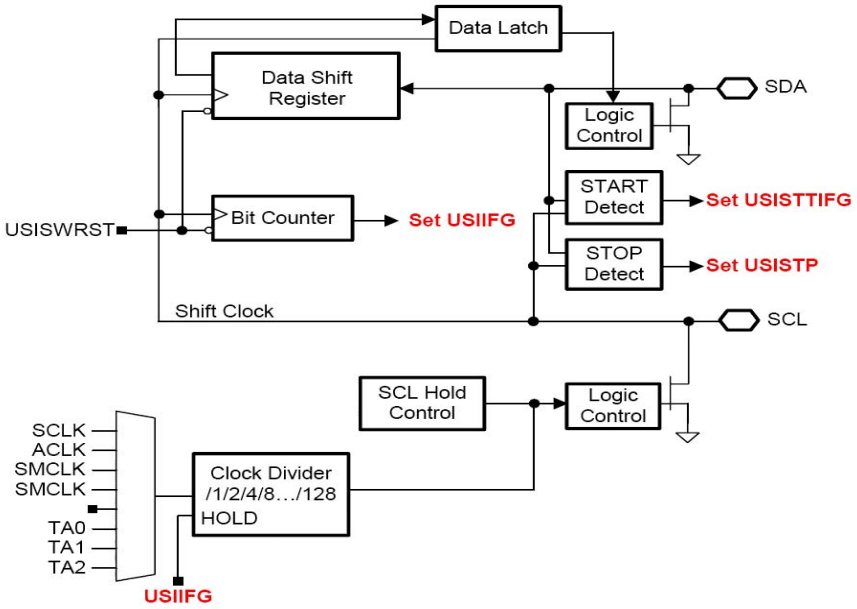


Table 14-4 gives the procedure for I²C communication between a Master TX and a Slave RX.

Table 14-4. Example: Communication procedure between Master TX and Slave RX.

Master TX	Slave RX
1: Send Start, Address and R/W bit	1: Detect Start, receive address and R/W
2: Receive (N)ACK	2: Transmit (N)ACK
3: Test (N)ACK and handle TX data	3: Data RX
4: Receive (N)ACK	4: Transmit (N)ACK
5: Test (N)ACK and prepare Stop	5: Reset for next Start
6: Send Stop	

14.12 Other documentation

This chapter only covers the main features of the different communication modules included in the MSP430. The TI web page contains Application Reports and Presentations covering the communications peripherals, recommendations for correct handling and user applications.

Annex E that provide a detailed analysis of each module. A brief overview of the documents is included here:

❑ Introduction to MSP430 Communication Interfaces <slap117.pdf>

- Makes a comparison between the features of the USART, USCI and USI communication modules. Gives the characteristics of the RS232, SPI and I^2C communication modules.

❑ In-Depth with MSP430's New Communication Interfaces <slap110.pdf>

- Presents the differences between the USCI and USI interfaces. Provides information concerning the USCI and USI communication modes (SPI and I^2C).

❑ Hands-on: The New MSP430 Communication Peripherals <slap120.pdf>

- A dedicated laboratory for SPI using the USI and I^2C using the USCI.

❑ New High Performance, Dual Communication Module – USCI

- Describes the features of the USCI module and its different communication modes: UART/LinBUS asynchronous mode, SPI synchronous mode and I^2C synchronous mode. Highlights bus and device selection.

❑ Powerful Yet Simple: Low-Cost Serial Communication with the New USI Module

- Provides an overview of the USI and gives the characteristics of SPI and I^2C communication modes using this module. It also presents the system-level benefits of using the USI.

❑ RF Basics, RF for Non-RF Engineers <slap127.pdf>

- Explains RF basics: building blocks of an RF system; RF Parameters; RF Measurement and equipment needed.

❑ Selecting the Right RF Protocol for your MSP430 Application

- Provides an introduction to ultra-low power wireless networking and presents the low power protocol selection criteria, focusing on TI devices: 802.15.4, ZigBee and SimpliciTI. Also recommends the appropriate protocol and some application examples.

❑ Implementing IrDA With The MSP430 <slaa202a.pdf>

- Implementation of the IrDA Lite protocol (IrPHY, IrLAP, and IrLMP) on the MSP430, as well as Tiny Transfer Protocol (TTP) and IrCOMM 3-wire services as a passive, secondary-only device. IrPHY implementations are provided using a Timer_A-based approach as well as using the USCI_A hardware module.

- Automatic Baud Rate Detection on the MSP430 <slaa215.pdf>
  - Presents the implementation of the interface to the user through a serial terminal via RS-232, using Automatic Baud Rate (ABR) detection to match of baud rates between the communication terminals.

- Software I^2C Slave Using the MSP430 <slaa330.pdf>
  - Describes the design of a software I^2C slave that can run up to 100-kbps using an MSP430.

- Using the USCI I^2C Master <slaa382.pdf>
  - Overview of the use of the I^2C master function set (single-master transmitter/receiver mode using 7-bit device addressing) for MSP430 devices with the USCI module.

- Using the USCI I^2C Slave <slaa383.pdf>
  - Overview of the use of the I^2C slave function (to handle both transmit and receive requests from I^2C master) for MSP430 devices with the USCI module.

- Using the USI I^2C Code Library <slaa368.pdf>
  - Overview of the master and slave code libraries for I^2C communication using the USI module of the MSP430F20xx.

14.13 Registers

14.13.1 USART Peripheral Interface (UART and SPI modes)

The universal synchronous/asynchronous receive/transmit (USART) peripheral interface supports two serial modes, in a singe hardware module.

This section provides the register bit definitions for both USART peripheral interfaces:

- Asynchronous UART mode;
- Synchronous SPI mode.

In this section, the registers for both modes are described simultaneously, taking into account that some of them use the same mnemonic, only differentiated by the register number (“UART” for UART mode and “SPI” for SPI mode). The registers exclusively used for each mode are described separately.

UART and SPI modes: UxCTL, USART Control Register

Mode	7	6	5	4	3	2	1	0
UART	PENA	PEV	SPB	CHAR	LISTEN	SYNC	MM	SWRST
SPI	Unused	Unused	I^2C(1)	CHAR	LISTEN	SYNC	MM	SWRST

(1) Not implemented in 4xx devices.

Bit		UART mode description		SPI mode description
7	PENA	Parity enable when PENA = 1 Parity bit is generated (UTXDx) and expected (URXDx).	Unused	
6	PEV	Parity select: PEV = 0 ⇒ Odd parity PEV = 1 ⇒ Even parity	Unused	
5	SPB	Stop bit select: SPB = 0 ⇒ One stop bit SPB = 1 ⇒ Two stop bits	I^2C	I^2C or SPI mode select when SYNC = 1. I^2C = 0 ⇒ SPI mode I^2C = 1 ⇒ I^2C mode
4	CHAR	Character length: CHAR = 0 ⇒ 7-bit data CHAR = 1 ⇒ 8-bit data	CHAR	As UART mode
3	LISTEN	Listen enable when LISTEN = 1. The transmit signal is internally fed back to the receiver.	LISTEN	As UART mode
2	SYNC	Synchronous mode enable: SYNC = 0 ⇒ UART mode SYNC = 1 ⇒ SPI Mode	SYNC	As UART mode
1	MM	Multiprocessor mode select MM = 0 ⇒ Idle-line multiprocessor protocol MM = 1 ⇒ Address-bit multiprocessor protocol	MM	Master mode: MM = 0 ⇒ USART is slave MM = 1 ⇒ USART is master
0	SWRST	Software reset enable: SWRST = 0 ⇒ Disabled. USART reset released for operation SWRST = 1 ⇒ Enabled. USART logic held in reset state	SWRST	As UART mode

UART and SPI modes: UxTCTL, USART Transmit Control Register

Mode	7	6	5	4	3	2	1	0
UART	Unused	CKPL	SSELx		URXSE	TXWAKE	Unused	TXEPT
SPI	CKPH	CKPL	SSELx		Unused	Unused	STC	TXEPT

Bit		UART mode description		SPI mode description
7	Unused		CKPH	
6	CKPL	Clock polarity select: CKPL = 0 ⇒ UCLKI = UCLK CKPL = 1 ⇒ UCLKI = inverted UCLK	CKPL	Clock polarity select: CKPL = 0 ⇒ UCLKI = The inactive state is low. CKPL = 1 ⇒ UCLKI = The inactive state is high.
5-4	SSELx	BRCLK source clock: SSEL1 SSEL0 = 00 ⇒ UCLKI SSEL1 SSEL0 = 01 ⇒ ACLK SSEL1 SSEL0 = 10 ⇒ SMCLK SSEL1 SSEL0 = 11 ⇒ SMCLK	SSELx	BRCLK source clock: SSEL1 SSEL0 = 00 ⇒ External UCLK (slave mode only) SSEL1 SSEL0 = 01 ⇒ ACLK (master mode only) SSEL1 SSEL0 = 10 ⇒ SMCLK (master mode only) SSEL1 SSEL0 = 11 ⇒ SMCLK (master mode only)
3	URXSE	UART receive start-edge enable when URXSE = 1	Unused	
2	TXWAKE	Transmitter wake: TXWAKE = 0 ⇒ Next frame transmitted is data TXWAKE = 1 ⇒ Next frame transmitted is an address	Unused	
1	Unused		STC	Slave transmit control: STC = 0 ⇒ 4-pin SPI mode: STE enabled. STC = 1 ⇒ 3-pin SPI mode: STE disabled.
0	TXEPT	Transmitter empty flag: TXEPT = 0 ⇒ UART is transmitting data and/or data is waiting in UxTXBUF TXEPT = 1 ⇒ Transmitter shift register and UxTXBUF are empty or SWRST=1	TXEPT	Transmitter empty flag: TXEPT = 0 ⇒ UART is transmitting data and/or data is waiting in UxTXBUF TXEPT = 1 ⇒ UxTXBUF and TX shift register are empty

UART and SPI modes: UxRCTL, USART Receive Control Register

Mode	7	6	5	4	3	2	1	0
UART	FE	PE	OE	BRK	URXEIE	URXWIE	RXWAKE	RXERR
SPI	FE	Unused	OE	Unused	Unused	Unused	Unused	Unused

Bit		UART mode description		SPI mode description
7	FE	Framing error flag: FE = 0 ⇒ No error FE = 1 ⇒ Character received with low stop bit	FE	Master mode framing error flag: (MM = 1 and STC = 0) FE = 0 ⇒ No conflict detected FE = 1 ⇒ Bus conflict (STE's negative edge)
6	PE	Parity error flag: PE = 0 ⇒ No error PE = 1 ⇒ Character received with parity error	Unused	
5	OE	Overrun error flag: OE = 0 ⇒ No error OE = 1 ⇒ A character was transferred into UxRXBUF before the previous character was read.	OE	As UART mode
4	BRK	Break detect flag: BRK = 0 ⇒ No break condition BRK = 1 ⇒ Break condition occurred	Unused	
3	URXEIE	Receive erroneous-character interrupt-enable: URXEIE = 0 ⇒ Err. characters rejected (URXIFGx=0) URXEIE = 1 ⇒ Err. characters received (URXIFGx=1)	Unused	
2	URXWIE	Receive wake-up interrupt-enable: URXWIE = 0 ⇒ All received characters set URXIFGx URXWIE = 1 ⇒ Received address characters set URXIFGx	Unused	
1	RXWAKE	Receive wake-up flag: RXWAKE = 0 ⇒ Received character is data RXWAKE = 1 ⇒ Received character is an address	Unused	
0	RXERR	Receive error flag: RXERR = 0 ⇒ No receive errors detected RXERR = 1 ⇒ Receive error detected	Unused	

UART and SPI modes: UxBR0, USART Baud Rate Control Register 0

Mode	7	6	5	4	3	2	1	0
UART / SPI	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

UART and SPI modes: UxBR1, USART Baud Rate Control Register 1

Mode	7	6	5	4	3	2	1	0
UART / SPI	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8

Bit		UART mode description		SPI mode description
7	UxBRx	The valid baud-rate control range is $3 \leq$ UxBR $<$ 0FFFFh, where UxBR = {UxBR1+UxBR0}. Unpredictable receive/transmit timing occurs if UxBR < 3.	UxBRx	The baud-rate generator uses the contents of {UxBR1+UxBR0} to set the baud rate. Unpredictable SPI operation occurs if UxBR < 2.

UART and SPI modes: UxMCTL, USART Modulation Control Register

Mode	7	6	5	4	3	2	1	0
UART / SPI	m7	m6	m5	m4	m3	m2	m1	m0

Bit		UART mode description		SPI mode description
7	UxMCTLx	Selects the modulation for BRCLK.	UxMCTLx	Not used in SPI mode and should be set to 00h.

UART and SPI modes: UxRXBUF, USART Receive Buffer Register

Mode	7	6	5	4	3	2	1	0
UART / SPI	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Bit		UART mode description		SPI mode description
7	UxRXBUFx	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UxRXBUF resets the receive-error bits, the RXWAKE bit, and URXIFGx. In 7-bit data mode, UxRXBUF is LSB justified and the MSB is always cleared.	UxRXBUFx	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UxRXBUF resets the OE bit and URXIFGx flag. In 7-bit data mode, UxRXBUF is LSB justified and the MSB is always cleared.

UART and SPI modes: UxTXBUF, USART Transmit Buffer Register

Mode	7	6	5	4	3	2	1	0
UART / SPI	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Bit		UART mode description		SPI mode description
7	UxTXBUFx	The transmit data buffer is user accessible and holds the data waiting to be moved to the transmit shift register and transmitted on UTXDx. Writing to the transmit data buffer clears UTXIFGx. The MSB of UxTXBUF is not used for 7-bit data and is cleared.	UxTXBUFx	The transmit data buffer is user accessible and contains current data to be transmitted. When seven-bit character-length is used, the data should be MSB justified before being moved into UxTXBUF. Data is transmitted MSB first. Writing to UxTXBUF clears UTXIFGx.

UART and SPI modes: ME1, Module Enable Register 1

Mode	7	6	5	4	3	2	1	0
UART	UTXE0	URXE0						
SPI		USPIE0						

Bit		UART mode description		SPI mode description
7	UTXE0	USART0 transmit enable: UTXE0 = 0 ⇒ Module not enabled UTXE0 = 1 ⇒ Module enabled		
6	URXE0	USART0 receive enable: URXE0 = 0 ⇒ Module not enabled URXE0 = 1 ⇒ Module enabled	USPIE0	USART0 SPI enable: USPIE0 = 0 ⇒ Module not enabled USPIE0 = 1 ⇒ Module enabled

The remaining bits may be used by other modules. See device-specific data sheet.

UART and SPI modes: ME2, Module Enable Register 2

Mode	7	6	5	4	3	2	1	0
UART			UTXE1	URXE1				
SPI				USPIE1				

Bit		UART mode description		SPI mode description
7	UTXE1	USART1 transmit enable: UTXE1 = 0 ⇒ Module not enabled UTXE1 = 1 ⇒ Module enabled		
6	URXE1	USART1 receive enable: URXE1 = 0 ⇒ Module not enabled URXE1 = 1 ⇒ Module enabled	USPIE1	USART1 SPI enable: USPIE1 = 0 ⇒ Module not enabled USPIE1 = 1 ⇒ Module enabled

UART and SPI modes: IE1, Interrupt Enable Register 1

Mode	7	6	5	4	3	2	1	0
UART / SPI	UTXIE0	URXIE0						

Bit		UART mode description		SPI mode description
7	UTXIE0	USART0 UTXIFG0 transmit interrupt enable: UTXIE0 = 0 ⇒ Interrupt not enabled UTXIE0 = 1 ⇒ Interrupt enabled	UTXIE0	As UART mode
6	URXIE0	USART0 URXIFG0 receive interrupt enable: URXIE0 = 0 ⇒ Interrupt not enabled URXIE0 = 1 ⇒ Interrupt enabled	URXIE0	As UART mode

The remaining bits may be used by other modules. See device-specific data sheet.

UART and SPI modes: IE2, Interrupt Enable Register 2

Mode	7	6	5	4	3	2	1	0
UART / SPI			UTXIE1	URXIE1				

Bit		UART mode description		SPI mode description
7	UTXIE1	USART1 UTXIFG1 transmit interrupt enable: UTXIE1 = 0 ⇒ Interrupt not enabled UTXIE1 = 1 ⇒ Interrupt enabled	UTXIE1	As UART mode
6	URXIE1	USART1 URXIFG1 receive interrupt enable: URXIE1 = 0 ⇒ Interrupt not enabled URXIE1 = 1 ⇒ Interrupt enabled	URXIE1	As UART mode

The remaining bits may be used by other modules. See device-specific data sheet.

UART and SPI modes: IFG1, Interrupt Flag Register 1

Mode	7	6	5	4	3	2	1	0
UART / SPI	UTXIFG0	URXIFG0						

Bit		UART mode description		SPI mode description
7	UTXIFG0	USART0 transmit interrupt flag. UTXIFG0 is set when U0TXBUF is empty. UTXIFG0 = 0 ⇒ No interrupt pending UTXIFG0 = 1 ⇒ Interrupt pending	UTXIFG0	As UART mode
6	URXIFG0	USART0 receive interrupt flag. URXIFG0 is set when U0RXBUF has received a complete character. URXIFG0 = 0 ⇒ No interrupt pending URXIFG0 = 1 ⇒ Interrupt pending	URXIFG0	As UART mode

The remaining bits may be used by other modules. See device-specific data sheet.

UART and SPI modes: IFG2, Interrupt Flag Register 2

Mode	7	6	5	4	3	2	1	0
UART / SPI			UTXIFG1	URXIFG1				

Bit		UART mode description		SPI mode description
7	UTXIFG1	USART1 transmit interrupt flag. UTXIFG1 is set when U1TXBUF is empty. UTXIFG1 = 0 ⇒ No interrupt pending UTXIFG1 = 1 ⇒ Interrupt pending	UTXIFG1	As UART mode
6	URXIFG1	USART1 receive interrupt flag. URXIFG1 is set when U1RXBUF has received a complete character. URXIFG1 = 0 ⇒ No interrupt pending URXIFG1 = 1 ⇒ Interrupt pending	URXIFG1	As UART mode

The remaining bits may be used by other modules. See device-specific data sheet.

14.13.2 USCI peripheral interface (UART, SPI and I²C modes)

The universal serial communication interface (USCI) modules support multiple serial communication modes. Different USCI modules support different modes. Each USCI module is identified by a different letter (USCI_A or USCI_B).

If more than one identical USCI module is implemented on one device, the modules are identified with incrementing numbers (USCI_A0 and USCI_A1). See the device-specific data sheet to determine which USCI module, if any, is implemented on which device.

The USCI_Ax modules support:

- UART mode;
- Pulse shaping for IrDA communications;
- Automatic baud rate detection for LIN communications;
- SPI mode.

The USCI_Bx modules support:

- SPI mode;
- I^2C mode.

This section provides the register bit definitions for the USCI module interfaces:

- Asynchronous UART mode;
- Synchronous SPI mode;
- Synchronous I^2C mode;

The registers for the implemented MSP430 modes are described simultaneously, taking into account that some of them use the same mnemonic, only differentiated by the register number ("UART" for UART mode, "SPI" for SPI mode and "I^2C" for I^2C mode). The registers exclusively used for each mode are presented separately.

UART mode: UCAxCTL0, USCI_Ax Control Register 0
SPI mode: UCAxCTL0, USCI_Ax Control Register 0 and **UCBxCTL0 USCI_Bx Control Register 0**
I²C mode: UCBxCTL0 USCI_Bx Control Register 0

Mode	7	6	5	4	3	2	1	0
UART	UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC=0
SPI	UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx		UCSYNC=1
I²C	UCA10	UCSLA10	UCMM	Unused	UCMST	UCMODEx=11		UCSYNC=1

Bit		UART mode description		SPI mode description		I²C mode description	
7	UCPEN	Parity enable when UCPEN = 1	UCCKPH	Clock phase select: UCCKPH = 0 ⇒ Data is changed on the 1st UCLK edge and captured on the next one. UCCKPH = 1 ⇒ Data is captured on the 1st UCLK edge and changed on the next one.	UCA10	Own addressing mode select: UCA10= 0 ⇒ 7-bit address UCA10= 1 ⇒ 10-bit address
6	UCPAR	Parity select: UCPAR = 0 ⇒ Odd parity UCPAR = 1 ⇒ Even parity	UCCKPL	Clock polarity select. UCCKPL = 0 ⇒ Inactive state: low. UCCKPL = 1 ⇒ Inactive state: high.	UCSLA10	Slave addressing mode select: UCSLA10= 0 ⇒ 7-bit address UCSLA10= 1 ⇒ 10-bit address
5	UCMSB	MSB first select: UCMSB = 0 ⇒ LSB first UCMSB = 1 ⇒ MSB first	UCMSB	As UART mode	UCMM	Multi-master environment select: UCMM= 0 ⇒ Single master UCMM= 1 ⇒ Multi master
4	UC7BIT	Character length: UC7BIT = 0 ⇒ 8-bit data UC7BIT = 1 ⇒ 7-bit data	UC7BIT	As UART mode	Unused	
3	UCSPB	Stop bit select: UCSPB = 0 ⇒ One stop bit UCSPB = 1 ⇒ Two stop bits	UCMST	Master mode: UCMST = 0 ⇒ USART is slave UCMST = 1 ⇒ USART is master	UCMST	Master mode select. UCMST = 0 ⇒ Slave mode UCMST = 1 ⇒ Master mode
2-1	UCMODEx	USCI asynchronous mode: = 00 ⇒ UART = 01 ⇒ Idle-Line Multiprocessor. = 10 ⇒ Address-Bit Multiprocessor. = 11 ⇒ UART with ABR.	UCMODEx	USCI synchronous mode: = 00 ⇒ 3-Pin SPI = 01 ⇒ 4-Pin SPI (slave enabled when UCxSTE = 1) = 10 ⇒ 4-Pin SPI (slave enabled when UCxSTE = 0) = 11 ⇒ I²C	UCMODEx =11	USCI Mode: = 00 ⇒ 3-Pin SPI = 01 ⇒ 4-Pin SPI (master/slave enabled if STE = 1) = 10 ⇒ 4-Pin SPI (master/slave enabled if STE = 0) = 11 ⇒ I²C
0	UCSYNC=0	Synchronous mode enable: UCSYNC = 0 ⇒ Asynchronous UCSYNC = 1 ⇒ Synchronous	UCSYNC=1	As UART mode	UCSYNC=1	As UART mode

UART mode: UCAxCTL1, USCI_Ax Control Register 1
SPI mode: UCAxCTL1, USCI_Ax Control Register 1 and **UCBxCTL0 USCI_Bx Control Register 1**
I²C mode: UCBxCTL1 USCI_Bx Control Register 1

Mode	7	6	5	4	3	2	1	0
UART	UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTXBRK	UCSWRST
SPI	UCSSELx		Unused	Unused	Unused	Unused	Unused	UCSWRST
I²C	UCSSELx		Unused	UCTR	UCTXNACK	UCTXSTP	UCTXSTT	UCSWRST

Bit		UART mode description		SPI mode description		I²C mode description
7-6	UCSSELx	BRCLK source clock: = 00 ⇒ UCLK = 01 ⇒ ACLK = 10 ⇒ SMCLK = 11 ⇒ SMCLK	UCSSELx	BRCLK source clock: = 00 ⇒ N/A = 01 ⇒ ACLK = 10 ⇒ SMCLK = 11 ⇒ SMCLK	UCSSELx	BRCLK source clock: = 00 ⇒ UCLKI = 01 ⇒ ACLK = 10 ⇒ SMCLK = 11 ⇒ SMCLK
5	UCRXEIE	Receive erroneous-character IE: = 0 ⇒ Rejected (UCAxRXIFG not set) = 1 ⇒ Received (UCAxRXIFG set)	Unused		Unused	Slave addressing mode select: UCSLA10= 0 ⇒ 7-bit address UCSLA10= 1 ⇒ 10-bit address
4	UCBRKIE	Receive break character IE: = 0 ⇒ Not set UCAxRXIFG. = 1 ⇒ Set UCAxRXIFG.	Unused		UCTR	Transmitter/Receiver select: = 0 ⇒ Receiver = 1 ⇒ Transmitter
3	UCDORM	Dormant. Puts USCI into sleep mode: = 0 ⇒ Not dormant = 1 ⇒ Dormant	Unused		UCTXNACK	Transmit a NACK: = 0 ⇒ Acknowledge normally = 1 ⇒ Generate NACK
2	UCTXADDR	Transmit address: = 0 ⇒ Next frame transmitted is data = 1 ⇒ Next frame transmitted is address	Unused		UCTXSTP	Transmit STOP condition in master mode: = 0 ⇒ No STOP generated = 1 ⇒ Generate STOP
1	UCTXBRK	Transmit break: = 0 ⇒ Next frame transmitted is not a break = 1 ⇒ Next frame transmitted is a break or a break/synch	Unused		UCTXSTT	Transmit START condition in master mode: = 0 ⇒ No START generated = 1 ⇒ Generate START
0	UCSWRST	Software reset enable =0 ⇒ Disabled. USCI reset released for operation 1 ⇒ Enabled. USCI logic held in reset state	UCSWRST	As UART mode	UCSWRST	As UART mode

UART mode: UCAxBR0, USCI_Ax Baud Rate Control Register 0
SPI mode: UCAxBR0, USCI_Ax Bit Rate Control Register 0 and **UCBxBR0, USCI_Bx Bit Rate Control Register 0**
I²C mode: UCBxBR0, USCI_Bx Baud Rate Control Register 0

Mode	7	6	5	4	3	2	1	0
UART / SPI / I²C	UCBRx – low byte							

UART mode: UCAxBR1, USCI_Ax Baud Rate Control Register 1
SPI mode: UCAxBR1, USCI_Ax Bit Rate Control Register 1 and **UCBxBR1, USCI_Bx Bit Rate Control Register 1**
I²C mode: UCBxBR1, USCI_Bx Baud Rate Control Register 1

Mode	7	6	5	4	3	2	1	0
UART / SPI / I²C	UCBRx – high byte							

Bit	UART mode description		SPI mode description		I²C mode description	
7-6	UCBRx	Clock prescaler setting of the baud rate generator: Prescaler value (16-bit value) = {UCAxBR0 + UCAxBR1 x 256}	UCBRx	Bit clock prescaler setting: Prescaler value (16-bit value) = {UCAxBR0 + UCAxBR1 × 256}	UCBRx	As SPI mode

UART mode: UCAxSTAT, USCI_Ax Status Register
SPI mode: UCAxSTAT, USCI_Ax Status Register and **UCBxSTAT, USCI_Bx Status Register**
I²C mode: UCBxSTAT, USCI_Bx Status Register

Mode	7	6	5	4	3	2	1	0
UART	UCLISTEN	UCFE	UCOE	UCPE	UCBRK	UCRXERR	UCADDR UCIDLE	UCBUSY
SPI	UCLISTEN	UCFE	UCOE	Unused	Unused	Unused	Unused	UCBUSY
I²C	Unused	UCSCLLOW	UCGC	UCBBUSY	UCNACKIFG	UCSTPIFG	UCSTTIFG	UCALIFG

Bit	UART mode description		SPI mode description		I²C mode description	
7	UCLISTEN	Listen enable: = 0 ⇒ Disabled = 1 ⇒ UCAxTXD is internally fed back to receiver	UCLISTEN	Listen enable: = 0 ⇒ Disabled = 1 ⇒ The transmitter output is internally fed back to receiver	Unused	
6	UCFE	Framing error flag: = 0 ⇒ No error = 1 ⇒ Character with low stop bit	UCFE	Framing error flag: = 0 ⇒ No error = 1 ⇒ Bus conflict (4w master)	UCSCLLOW	SCL low: = 0 ⇒ SCL is not held low = 1 ⇒ SCL is held low
5	UCOE	Overrun error flag: = 0 ⇒ No error = 1 ⇒ Overrun error	UCOE	As UART mode	UCGC	General call address received: = 0 ⇒ No general call address = 1 ⇒ General call address
4	UCPE	Parity error flag: = 0 ⇒ No error = 1 ⇒ Character with parity error	Unused		UCBBUSY	Bus busy: = 0 ⇒ Bus inactive = 1 ⇒ Bus busy
3	UCBRK	Break detect flag: = 0 ⇒ No break condition = 1 ⇒ Break condition occurred	Unused		UCNACKIFG	NACK received interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending
2	UCRXERR	Receive error flag. = 0 ⇒ No receive errors detected = 1 ⇒ Receive error detected	Unused		UCSTPIFG	Stop condition interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending
1	UCADDR UCIDLE	Address-bit multiprocessor mode: = 0 ⇒ Received character is data = 1 ⇒ Received character is an address Idle-line multiprocessor mode: = 0 ⇒ No idle line detected = 1 ⇒ Idle line detected	Unused		UCSTTIFG	Start condition interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending
0	UCBUSY	USCI busy: = 0 ⇒ USCI inactive = 1 ⇒ USCI transmit/receive	UCBUSY		UCALIFG	Arbitration lost interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending

UART mode: UCAxRXBUF, USCI_Ax Receive Buffer Register
SPI mode: UCAxRXBUF, USCI_Ax Receive Buffer Register and **UCBxRXBUF, USCI_Bx Receive Buffer Register**
I²C mode: UCBxRXBUF, USCI_Bx Receive Buffer Register

Mode	7	6	5	4	3	2	1	0
UART / SPI / I²C	UCRXBUFx							

Bit		UART mode description		SPI mode description		I²C mode description
7-0	UCRXBUFx	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets receive-error bits, UCADDR/UCIDLE bit and UCAxRXIFG. In 7-bit data mode, UCAxRXBUF is LSB justified and the MSB is always cleared.	UCRXBUFx	As UART mode Reading UCxRXBUF resets the receive-error bits, and UCxRXIFG	UCRXBUFx	As SPI mode

UART mode: UCAxTXBUF, USCI_Ax Transmit Buffer Register
SPI mode: UCAxTXBUF, USCI_Ax Transmit Buffer Register and **UCBxTXBUF, USCI_Bx Transmit Buffer Register**
I²C mode: UCBxTXBUF, USCI_Bx Transmit Buffer Register

Mode	7	6	5	4	3	2	1	0
UART / SPI / I²C	UCTXBUFx							

Bit		UART mode description		SPI mode description		I²C mode description
7-0	UCTXBUFx	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted on UCAxTXD. Writing to the transmit data buffer clears UCAxTXIFG.	UCTXBUFx	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCxTXIFG.	UCTXBUFx	As SPI mode

UART, SPI and I²C modes: IE2, Interrupt Enable Register 2

Mode	7	6	5	4	3	2	1	0
UART							UCA0TXIE	UCA0RXIE
SPI					UCB0TXIE	UCB0RXIE	UCA0TXIE	UCA0RXIE
I²C					UCB0TXIE	UCB0RXIE		

Bit	UART mode description		SPI mode description		I²C mode description	
3			UCB0TXIE	USCI_B0 transmit interrupt enable: UTXIE1 = 0 ⇒ Disabled UTXIE1 = 1 ⇒ Enabled	UCB0TXIE	As SPI mode
2			UCB0RXIE	USCI_B0 receive interrupt enable: URXIE1 = 0 ⇒ Disabled URXIE1 = 1 ⇒ Enabled	UCB0RXIE	As SPI mode
1	UCA0TXIE	USCI_A0 transmit interrupt enable: UTXIE1 = 0 ⇒ Disabled UTXIE1 = 1 ⇒ Enabled	UCA0TXIE	As UART mode		
0	UCA0RXIE	USCI_A0 receive interrupt enable: URXIE1 = 0 ⇒ Disabled URXIE1 = 1 ⇒ Enabled	UCA0RXIE	As UART mode		

The remaining bits may be used by other modules. See device-specific data sheet.

UART, SPI and I²C modes: IFG2, Interrupt Flag Register 2

Mode	7	6	5	4	3	2	1	0
UART							UCA0TXIFG	UCA0RXIFG
SPI					UCB0TXIFG	UCB0RXIFG	UCA0TXIFG	UCA0RXIFG
I²C					UCB0TXIFG	UCB0RXIFG		

Bit	UART mode description		SPI mode description		I²C mode description	
3			UCB0TXIFG	USCI_B0 transmit interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCB0TXIFG	As SPI mode
2			UCB0RXIFG	USCI_B0 receive interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCB0RXIFG	As SPI mode
1	UCA0TXIFG	USCI_A0 transmit interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCA0TXIFG	As UART mode		
0	UCA0RXIFG	USCI_A0 receive interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCA0RXIFG	As UART mode		

The remaining bits may be used by other modules. See device-specific data sheet.

UART mode: UC1IE, USCI_A1 Interrupt Enable Register
SPI mode: UC1IE, USCI_A1/USCI_B1 Interrupt Enable Register
I²C mode: UC1IE, USCI_B1 Interrupt Enable Register

Mode	7	6	5	4	3	2	1	0
UART	Unused	Unused	Unused	Unused			UCA1TXIE	UCA1RXIE
SPI	Unused	Unused	Unused	Unused	UCB1TXIE	UCB1RXIE	UCA1TXIE	UCA1RXIE
I²C	Unused	Unused	Unused	Unused	UCB1TXIE	UCB1RXIE		

Bit	UART mode description		SPI mode description		I²C mode description	
3			UCB1TXIE	USCI_B1 transmit interrupt enable: UTXIE1 = 0 ⇒ Disabled UTXIE1 = 1 ⇒ Enabled	UCB1TXIE	As SPI mode
2			UCB1RXIE	USCI_B1 receive interrupt enable: URXIE1 = 0 ⇒ Disabled URXIE1 = 1 ⇒ Enabled	UCB1RXIE	As SPI mode
1	UCA1TXIE	USCI_A1 transmit interrupt enable: UTXIE1 = 0 ⇒ Disabled UTXIE1 = 1 ⇒ Enabled	UCA1TXIE	As UART mode		
0	UCA1RXIE	USCI_A1 receive interrupt enable: URXIE1 = 0 ⇒ Disabled URXIE1 = 1 ⇒ Enabled	UCA1RXIE	As UART mode		

The remaining bits may be used by other modules. See device-specific data sheet.

UART mode: UC1IFG, USCI_A1 Interrupt Flag Register
SPI mode: UC1IFG, USCI_A1/USCI_B1 Interrupt Flag Register
I²C mode: UC1IFG, USCI_B1 Interrupt Flag Register

Mode	7	6	5	4	3	2	1	0
UART							UCA1TXIFG	UCA1RXIFG
SPI					UCB1TXIFG	UCB1RXIFG	UCA1TXIFG	UCA1RXIFG
I²C					UCB1TXIFG	UCB1RXIFG		

Bit		UART mode description		SPI mode description		I²C mode description	
3			UCB1TXIFG	USCI_B1 transmit interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCB1TXIFG	As SPI mode
2			UCB1RXIFG	USCI_B1 receive interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCB1RXIFG	As SPI mode
1	UCA1TXIFG	USCI_A1 transmit interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCA1TXIFG	As UART mode		
0	UCA1RXIFG	USCI_A1 receive interrupt flag: = 0 ⇒ No interrupt pending = 1 ⇒ Interrupt pending	UCA1RXIFG	As UART mode		

The remaining bits may be used by other modules. See device-specific data sheet.

UART mode: UCAxMCTL, USCI_Ax Modulation Control Register

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16

Bit		UART mode description
7-4	UCBRFx	First modulation pattern for BITCLK16 when UCOS16 = 1 (See *Table 19-3* of the MSP430x4xx User's Guide)
3-1	UCBRSx	Second modulation pattern for BITCLK (See *Table 19-2* of the MSP430x4xx User's Guide)
0	UCOS16	Oversampling mode enabled when UCOS16 = 1

UART mode: UCAxIRTCTL, USCI_Ax IrDA Transmit Control Register

7	6	5	4	3	2	1	0
UCIRTXPLx						UCIRTXCLK	UCIREN

Bit		UART mode description
7-2	UCIRTXPLx	Transmit pulse length: t_{PULSE} = (UCIRTXPLx + 1) / (2 x $f_{IRTXCLK}$)
1	UCIRTXCLK	IrDA transmit pulse clock select: UCIRTXCLK = 0 ⇒ BRCLK UCIRTXCLK = 1 ⇒ BITCLK16, when UCOS16 = 1 ⇒ BRCLK,otherwise
0	UCIREN	IrDA encoder/decoder enable: UCIREN = 0 ⇒ IrDA encoder/decoder disabled UCIREN = 1 ⇒ IrDA encoder/decoder enabled

UART mode: UCAxIRRCTL, USCI_Ax IrDA Receive Control Register

7	6	5	4	3	2	1	0
UCIRRXFLx						UCIRRXPL	UCIRRXFE

Bit		UART mode description
7-2	UCIRRXFLx	Receive filter length (minimum pulse length): t_{MIN} = (UCIRRXFLx + 4) / (2 × $f_{IRTXCLK}$)
1	UCIRRXPL	IrDA receive input UCAxRXD polarity. When a light pulse is seen: UCIRRXPL = 0 ⇒ IrDA transceiver delivers a high pulse UCIRRXPL = 1 ⇒ IrDA transceiver delivers a low pulse
0	UCIRRXFE	IrDA receive filter enabled: UCIRRXFE = 0 ⇒ Disabled UCIRRXFE = 1 ⇒ Enabled

UART mode: UCAxABCTL, USCI_Ax Auto Baud Rate Control Register

7	6	5	4	3	2	1	0
Reserved		UCDELIMx		UCSTOE	UCBTOE	Reserved	UCABDEN

Bit		UART mode description
5-4	UCDELIMx	Break/synch delimiter length: UCDELIM1 UCDELIM0 = 00 ⇒ 1 bit time UCDELIM1 UCDELIM0 = 01 ⇒ 2 bit times UCDELIM1 UCDELIM0 = 10 ⇒ 3 bit times UCDELIM1 UCDELIM0 = 11 ⇒ 4 bit times
3	UCSTOE	Synch field time out error: UCSTOE = 0 ⇒ No error UCSTOE = 1 ⇒ Length of synch field exceeded measurement time
2	UCBTOE	Break time out error: UCBTOE = 0 ⇒ No error UCBTOE = 1 ⇒ Length of break field exceeded 22 bit times.
0	UCABDEN	Automatic baud rate detect enable: UCABDEN = 0 ⇒ Baud rate detection disabled UCABDEN = 1 ⇒ Baud rate detection enabled

I²C mode: UCBxI2COA, USCIBx I2C Own Address Register

15	14	13	12	11	10	9	8
UCGCEN	0	0	0	0	0	I2COAx	

7	6	5	4	3	2	1	0
I2COAx							

Bit		UART mode description
15	UCGCEN	General call response enable: UCGCEN = 0 ⇒ Do not respond to a general call UCGCEN = 1 ⇒ Respond to a general call
9-0	I2COAx	I²C own address (local address of the USCI_Bx I²C controller) ⇒ Right-justified address ⇒ 7-bit address ⇒ Bit 6 is the MSB, Bits 9-7 are ignored. ⇒ 10-bit address ⇒ Bit 9 is the MSB.

I²C mode: UCBxI2CSA, USCI_Bx I2C Slave Address Register

15	14	13	12	11	10	9	8
0	0	0	0	0	0	I2CSAx	

7	6	5	4	3	2	1	0
I2CSAx							

Bit		UART mode description
9-0	I2CSAx	I²C slave address (slave address of the external device to be addressed by the USCI_Bx module) ⇒ Only used in master mode ⇒ Right-justified address ⇒ 7-bit address ⇒ Bit 6 is the MSB, Bits 9-7 are ignored. ⇒ 10-bit address ⇒ Bit 9 is the MSB.

I²C mode: UCBxI2CIE, USCI_Bx I²C Interrupt Enable Register

7	6	5	4	3	2	1	0
Reserved				UCNACKIE	UCSTPIE	UCSTTIE	UCALIE

Bit		UART mode description
3	UCNACKIE	Not-acknowledge interrupt enable: UCNACKIE = 0 ⇒ Interrupt disabled UCNACKIE = 1 ⇒ Interrupt enabled
2	UCSTPIE	Stop condition interrupt enable: UCSTPIE = 0 ⇒ Interrupt disabled UCSTPIE = 1 ⇒ Interrupt enabled
1	UCSTTIE	Start condition interrupt enable: UCSTTIE = 0 ⇒ Interrupt disabled UCSTTIE = 1 ⇒ Interrupt enabled
0	UCALIE	Arbitration lost interrupt enable: UCALIE = 0 ⇒ Interrupt disabled UCALIE = 1 ⇒ Interrupt enabled

14.13.3 USI peripheral interface (SPI and I²C modes)

The Universal Serial Interface (USI) module provides SPI and I²C serial communication using a single hardware module.

This module is implemented in the MSP430x20xx devices, i.e., in the eZ430-F2013 hardware development kit and in the Experimenter's board.

USICTL0, USI Control Register 0

7	6	5	4	3	2	1	0
USIPE7	USIPE6	USIPE5	USILSB	USIMST	USIGE	USIOE	USISWRST

Bit		Description
7	USIPE7	USI SDI/SDA port enable: ⇒ SPI mode ⇒ Input ⇒ I²C mode ⇒ Input or open drain output USIPE7 = 0 ⇒ USI function disabled USIPE7 = 1 ⇒ USI function enabled
6	USIPE6	USI SDO/SCL port enable: ⇒ SPI mode ⇒ Output ⇒ I²C mode ⇒ Input or open drain output USIPE6 = 0 ⇒ USI function disabled USIPE6 = 1 ⇒ USI function enabled
5	USIPE5	USI SCLK port enable: ⇒ SPI slave mode ⇒ Input ⇒ SPI master mode ⇒ Output ⇒ I²C mode ⇒ Input USIPE5 = 0 ⇒ USI function disabled USIPE5 = 1 ⇒ USI function enabled
4	USILSB	LSB first select (direction of the receive and transmit shift register): USILSB = 0 ⇒ MSB first USILSB = 1 ⇒ LSB first
3	USIMST	Master select: USIMST = 0 ⇒ Slave mode USIMST = 1 ⇒ Master mode
2	USIGE	Output latch control: USIGE = 0 ⇒ Output latch enable depends on shift clock USIGE = 1 ⇒ Output latch always enabled and transparent
1	USIOE	Data output enable: USIOE = 0 ⇒ Output disabled USIOE = 1 ⇒ Output enabled
0	USIWRST	USI software reset: USIWRST = 0 ⇒ USI released for operation USIWRST = 1 ⇒ USI logic held in reset state

USICTL1, USI Control Register 1

7	6	5	4	3	2	1	0
USICKPH	USII2C	USISTTIE	USIIE	USIAL	USISTP	USISTTIFG	USIIFG

Bit		Description
7	USICKPH	Clock phase select: USICKPH = 0 ⇒ Data is changed on the first SCLK edge and captured on the following edge USICKPH = 1 ⇒ Data is captured on the first SCLK edge and changed on the following edge
6	USII2C	I^2C mode enable: USII2C = 0 ⇒ I^2C mode disabled USII2C = 1 ⇒ I^2C mode enabled
5	USISTTIE	START condition interrupt-enable: USISTTIE = 0 ⇒ Interrupt on START condition disabled USISTTIE = 1 ⇒ Interrupt on START condition enabled
4	USIIE	USI counter interrupt enable: USIIE = 0 ⇒ Interrupt disabled USIIE = 1 ⇒ Interrupt enabled
3	USIAL	Arbitration lost: USIAL = 0 ⇒ No arbitration lost condition USIAL = 1 ⇒ Arbitration lost
2	USISTP	STOP condition received: USISTP = 0 ⇒ No STOP condition received USISTP = 1 ⇒ STOP condition received
1	USISTTIFG	START condition interrupt flag: USISTTIFG = 0 ⇒ No interrupt pending USISTTIFG = 1 ⇒ Interrupt pending
0	USIIFG	USI counter interrupt flag: USIIFG = 0 ⇒ No interrupt pending USIIFG = 1 ⇒ Interrupt pending

USICKCTL, USI Clock Control Register

7	6	5	4	3	2	1	0
USIDIVx			USISSELx			USICKPL	USISWCLK

Bit		Description
7-5	USIDIVx	Clock divider select: USIDIV2 USIDIV1 USIDIV0 = 000 ⇒ Divide by 1 USIDIV2 USIDIV1 USIDIV0 = 001 ⇒ Divide by 2 USIDIV2 USIDIV1 USIDIV0 = 010 ⇒ Divide by 4 USIDIV2 USIDIV1 USIDIV0 = 011 ⇒ Divide by 8 USIDIV2 USIDIV1 USIDIV0 = 100 ⇒ Divide by 16 USIDIV2 USIDIV1 USIDIV0 = 101 ⇒ Divide by 32 USIDIV2 USIDIV1 USIDIV0 = 110 ⇒ Divide by 64 USIDIV2 USIDIV1 USIDIV0 = 111 ⇒ Divide by 128
4-2	USISSELx	Clock source select. Not used in slave mode. USISSEL2 USISSEL1 USISSEL0 = 000 ⇒ SCLK (1) USISSEL2 USISSEL1 USISSEL0 = 001 ⇒ ACLK USISSEL2 USISSEL1 USISSEL0 = 010 ⇒ SMCLK USISSEL2 USISSEL1 USISSEL0 = 011 ⇒ SMCLK USISSEL2 USISSEL1 USISSEL0 = 100 ⇒ USISWCLK bit USISSEL2 USISSEL1 USISSEL0 = 101 ⇒ TACCR0 USISSEL2 USISSEL1 USISSEL0 = 110 ⇒ TACCR1 USISSEL2 USISSEL1 USISSEL0 = 111 ⇒ TACCR2 (2) (1) Not used in SPI mode (2) Reserved on MSP430F20xx devices
1	USICKPL	Clock polarity select: USICKPL = 0 ⇒ Inactive state is low USICKPL = 1 ⇒ Inactive state is high
0	USISWCLK	Software clock: USISWCLK = 0 ⇒ Input clock is low USISWCLK = 1 ⇒ Input clock is high

USICNT, USI Bit Counter Register

7	6	5	4	3	2	1	0
USISCLREL	USI16B	USIIFGCC	USICNTx				

Bit		Description
7	USISCLREL	SCL line release from low to idle: USISCLREL = 0 ⇒ SCL line is held low if USIIFG is set USISCLREL = 1 ⇒ SCL line is released
6	USI16B	16-bit shift register enable: USI16B = 0 ⇒ 8-bit shift register mode. (Uses USISRL low byte) USI16B = 1 ⇒ 16-bit shift register mode (Uses both USISRx bytes)
5	USIIFGCC	USI interrupt flag clear control: USIIFGCC = 0 ⇒ USIIFG automatically cleared on USICNTx update USIIFGCC = 1 ⇒ USIIFG is not cleared automatically
4-0	USICNTx	USI bit count (Number of bits to be received or transmitted)

USISRL, USI Low Byte Shift Register

7	6	5	4	3	2	1	0
USISRLx							

Bit		Description
7-0	USISRLx	Contents of the USI low byte shift register

USISRH, USI High Byte Shift Register

7	6	5	4	3	2	1	0
USISRHx							

Bit		Description
7-0	USISRHx	Contents of the USI high byte shift register

14.14 Laboratory 10: Echo test

14.14.1 Lab10a: Echo test using the UART mode of the USCI module

Project files

- C source files: **Chapter 14 > Lab10 > Lab10a_student.c**

  Solution file: **Chapter 14 > Lab10 > Lab10a_solution.c**

Overview

This laboratory explores the USCI module in UART mode that will be connected to a Code Composer Essentials (CCE) IO console. When the connection is established, the character sequence written on the keyboard to the console will be displayed again on the console.

A. Resources

This laboratory uses the USCI module in asynchronous mode. The RX interrupt activates the service routine that reads the incoming character and sends it out again to the PC (computer), allowing the instantaneous display (echo) of the written character.

The resources used are:

- USCI module;
- Interrupts;
- IO ports:
- System clock.

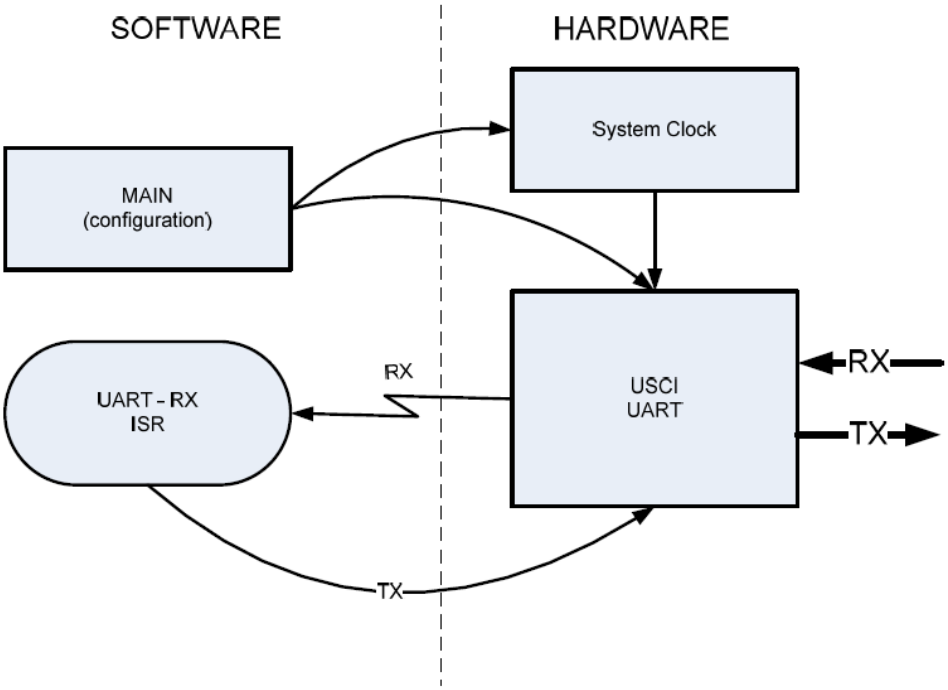
With the objective of allowing the generation of two different baud rates, a function has been added that configures the FLL+ and selects the base frequency for the UART. In this example it will be 8 MHz.

B. Software application organization

The proposed software is organized as shown in *Figure 14-44*. The main routine performs the necessary hardware configuration. Then, the hardware takes command of the software through the interrupt service routine generated by the reception of a new character.

The initial configuration sets the system clock to a frequency of 8 MHz.

Figure 14-44. Lab10a: Software application organization.



C. UART configuration

- **Configure the control registers**

The connection will operate in the following mode:

- Parity disabled;
- LSB first;
- 8-bit data;
- One stop bit.

The module will operate in the following mode:

- Asynchronous;
- SMCLK source clock;
- No Receive erroneous-character interrupt-enable;
- No Receive break character interrupt-enable.

Configure the following control registers based on these characteristics:

```
UCA0CTL0 = _________________;
UCA0CTL1 = _________________;
```

❑ Baud rate generation

The module has an 8 MHz clock source and the objective is to establish a connection at 9600 Baud. It is necessary to select the baud rate generation in oversampling mode. Configure the following registers:

```
UCA0BR0  = ________________;
UCA0BR1  = ________________;
UCA0MCTL = ________________;
```

❑ Port configuration

In order to set the external interfaces at the USCI module, it is necessary to configure the I/O ports. Select the USCI peripheral in UART mode following the connections provided on the Experimenter's board:

```
P2SEL = ____________________;
```

❑ RX interrupt enable

To finish the module configuration, it is necessary to enable the receive interrupts:

```
IE2 = _____________________;
```

D. Analysis of operation

Once the USCI module is configured in accordance with the previous steps, initiate the experiment by completing the file **Lab10a_student.c**, compiling it and running it on the Experimenter's board. The complete solution can be found in the file **Lab10a_solution.c**.

For the correct operation, there must be a connection between the Experimenter's board and the PC. If the CCE console is disabled, go to **Window > Show View > Console** to enable it. If necessary, configure the CCE console options in accordance to the connection details.

❑ Verification

Once the program code is running, any character key pressed in the PC keyboard will be displayed on the CCE console.

MSP-EXP430FG4618 **SOLUTION**

Using USCI module in UART mode included in the MSP-EXP430FG4618 Development Tool, develop a procedure to connect the development tool to the CCE console. When the connection is established, the character sequence written by the keyboard to the console will be displayed again on the console.

- Configure the control registers:

```
UCA0CTL0 = 0x00;
  // UCA0CTL0 =
  //UCPEN|UCPAR|UCMSB|UC7BIT|UCSPB|UCMODEx|UCSYNC|
  //UCPEN (Parity)              = 0b  ->  Parity disabled
  //UCPAR (Parity select)            = 0b  ->  Odd parity
  //UCMSB (MSB first select)          = 0b  ->  LSB first
  //UC7BIT (Character length)        = 0b  ->  8-bit data
  //UCSPB (Stop bit select)        = 0b  ->  One stop bit
  //UCMODEx (USCI mode)               = 00b ->  UART Mode
  //UCSYNC                     = 0b  ->  Asynchronous mode

UCA0CTL1 = 0x81;
// UCA0CTL1 =
//UCSSELx|UCRXEIE|UCBRKIE|UCDORM|UCTXADDR|UCTXBRK|UCSWRST|
  //UCSSELx (USCI clock source select)    = 10b ->  SMCLK
  //UCRXEIE       = 0b  ->  Erroneous characters rejected
  //UCBRKIE       = 0b  ->  Received break characters set
  //UCDORM                          = 0b  ->  Not dormant
  //UCTXADDR     = 0b  ->  Next frame transmitted is data
  //UCTXBRK   = 0b  ->  Next frame transmitted is no break
  //UCSWRST               = 1b  ->  normally Set by a PUC
```

- Baud rate generation:

```
UCA0BR0 = 0x34;
UCA0BR1 = 0x00;
  //Prescaler = 8MHz/(16 x 9600) = 52 = 0x34
  //9600 from 8MHz -> SMCLK

UCA0MCTL = 0x11;
// UCA0MCTL =
  //UCBRFx|UCBRSx|UCOS16|
  //UCBRFx (1st modulation stage)    = 0001b -> Table 19-4
  //UCBRSx (2nd modulation stage)    = 000b  -> Table 19-4
  //UCOS16 (Oversampling mode)         = 1b    -> Enabled
```

- Configuration of ports:

```
P2SEL |= 0x30;
  //P2.4,P2.5 = USCI_A0 TXD,RXD
```

- RX interrupt enable:

```
IE2 |= UCA0RXIE;
  //Enable USCI_A0 RX interrupt
```

14.14.2 Lab10b: Echo test using SPI

Project files

- C source files: **Chapter 14 > Lab10 > Lab10b1_student.c**
  Chapter 14 > Lab10 > Lab10b2_student.c
  Solution files: **Chapter 14 > Lab10 > Lab10b1_solution.c**
  Chapter 14 > Lab10 > Lab10b2_solution.c

Overview

This laboratory explores the USCI and USI communication interfaces in SPI mode. The MSP430 devices included on the Experimenter's board will exchange messages between themselves, one being the MSP430FG4618 (master) that will control operation of the other MSP430F2013 device (slave). The master, by reading the current state of the slave, will drive the slave to the new desired state, controlling its activity. In this particular case, switching the state of LED3 will be implemented.

A. Resources

This laboratory uses the USCI module of the MSP430FG4618 device and the USI module included on the MSP430F2013. Both units operate in SPI mode.

The Basic Timer1 of the master device is programmed to switch the status of the slave device once every 2 seconds.

The slave is notified of the arrival of information through the counting end interrupt of the USI module.

The resources used are:

- USCI module;
- USI module;
- Basic Timer1;
- Interrupts;
- I/O ports.

B. Software application organization

The software architecture for this laboratory is shown in Figure 14-45.

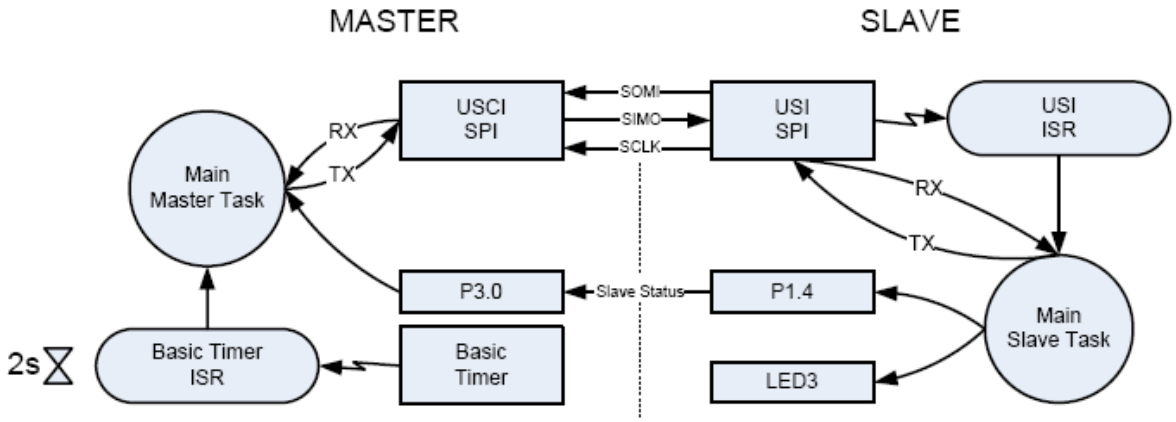
The master unit is composed of two software modules:

- The "Main master task" module contains the operation algorithm of master unit;
- The "ISR Basic Timer" module wakes the "Main master task" once every 2 seconds.

The slave unit is also composed of two modules:

- The "Main slave task" module contains the operation algorithm of the slave unit;
- The "USI ISR" module reads the data received, prepares the USI module for new reception and wakes the "Main slave task" to execute the algorithm associated with the reception of the new command.

Figure 14-45. Lab10b: Software architecture.



C. configuration

- **Configure the control registers USCI_B (master)**

The SPI connection will operate in the following mode:

- Clock phase -> Data value is updated on the first UCLK edge and captured on the following edge;
- Clock polarity -> the inactive state is low;
- MSB first;
- 8-bit data;
- Master mode;
- 3-Pin SPI;
- Source clock -> SMCLK.

Configure the following control registers based on these characteristics:

```
UCB0CTL0 = _________________:
UCB0CTL1 = _________________;
```

- **Data rate USCI_B (master)**

The system clock is configured to operate with a frequency of ~ 1048 kHz from the DCO. This frequency will be the working base frequency of the USCI module. The connection operates at a clock frequency of ~ 500 kHz. Configure the following registers:

```
UCB0BR0= _________________;
UCB0BR1= _________________;
```

- **Port configuration USCI_B (master)**

In order to set the external interfaces at the USCI module, it is necessary to configure the I/O ports. Select the USCI peripheral in SPI mode, matching the connections provided at the Experimenter's board:

```
P3SEL = ___________________;
```

- **Configure the control registers USI (slave)**

The SPI connection will operate on the following mode:

- MSB first;
- 8-bit data.
- Slave mode;
- Clock phase -> Data is changed on the first SCLK edge and captured on the following edge;
- USI counter interrupt enable.

Configure the following control registers based on these characteristics:

```
USICTL0 = _________________;
USICTL1 = _________________;
```


D. Analysis of operation

Once the USCI module is configured in accordance with the previous steps, initiate the experiment by completing the files **Lab10b1_student.c** (master – MSP430FG4618) and **Lab10b2_student.c** (slave – MSP430F2013), compiling them and running them on the Experimenter's board. The complete solution can be found in the files **Lab10b1_solution.c** and **Lab10b2_soluction.c**.

For this laboratory, it is necessary to set the following jumper settings:

- PWR1/2, BATT, LCL1/2, JP2;
- SPI: H1- 1&2, 3&4, 5&6, 7&8.

- **Verification**

Once the program code is running in the two microcontrollers, monitor LED3 on the Experimenter's board. It will blink with a period of 4 seconds.

MSP-EXP430FG4618 (master) **SOLUTION**

Using USCI module in SPI mode included in the FG4618 (configured as master) of the Experimenter's board, establish a connection to the F2013 by its USI module in SPI mode. The data exchanged is displayed by the LED blinking.

- Configure the control registers USCI_B (master):

```
UCB0CTL0 = 0x29;
//UCB0CTL0 =
// UCCKPH|UCCKPL|UCMSB|UC7BIT|UCMST|UCMODEx|UCSYNC|
  //UCCKPH (Clock phase) = 0b   ->  Data is changed on the
  //   first UCLK edge and captured on the following edge.
  //UCCKPL (Clock polarity) = 0b  -> Inactive state is low
  //UCMSB (MSB first select) = 1b            ->  MSB first
  //UC7BIT (Character length) = 0b          ->  8-bit data
  //UCMST (Master mode) = 1b               ->  Master mode
  //UCMODEx (USCI mode) = 00b                 -> 3-Pin SPI
  //UCSYNC (Synch. mode enable) = 1b  ->  Synchronous mode
```

```
UCB0CTL1 = 0x81;
//UCB0CTL1 =
// UCSSELx |                Unused                |UCSWRST|
  //UCSSELx (USCI clock source select)= 10b      ->  SMCLK
  //UCSWRST (Software reset) = 1b -> normally set by a PUC
```

❑ Configure the data rate USCI_B (master):

```
UCB0BR0 = 0x02;
UCB0BR1 = 0x00;
// DATA RATE
  // Data rate = SMCLK/2 ~= 500kHz
  // UCB0BR1 = 0x00 & UCB0BR0 = 0x02
```

❑ Configure I/O ports:

```
P3SEL |= 0x0E;
// P3.3, P3.2, P3.1 option select
```

MSP-EXP430F2013 (slave)

SOLUTION

Using USCI module in SPI mode included in the FG4618 (configured as master) of the Experimenter's board, establish a connection to the F2013 by its USI module in SPI mode. The data exchanged is displayed on the LED blinking.

❑ USI (slave) control registers:

```
USICTL0 = 0xE3;
//USICTL0 =
//USIPE7|USIPE6|USIPE5|USILSB|USIMST|USIGE|USIOE|USISWRST|
  //USIPE7 (USI SDI/SDA port enable) = 1b   -> USI enabled
  //USIPE6 (USI SDO/SCL port enable) = 1b  ->  USI enabled
  //USIPE5 (USI SCLK port enable) = 1b     ->  USI enabled
  //USILSB (LSB first) = 0b                  ->  MSB first
  //USIMST (Master) = 0b                    ->  Slave mode
  //USIGE (Output latch control) = 0b     ->  Output latch
  //                          enable depends on shift clock
  //USIOE (Serial data output enable) = 1b     ->  Output
enabled
  //USISWRST (USI software reset) = 1b  ->  Software reset
```



```
USICTL1 = 0x10;
//USICTL1 =
//USICKPH|USII2C|USISTTIE|USIIE|USIAL|USISTP|USISTTIFG|…
//…USIIFG|
  //USICKPH (Clock phase select) = 0b  ->  Data is changed
  //  on the first SCLK edge and captured on the following
  //  edge
  //USII2C (I2C mode enable) = 0b    ->  I2C mode disabled
  //USISTTIE (START condition interrupt) = 0b ->  Not used
  //USIIE (USI counter) = 1b          -> Interrupt enabled
  //USIAL (Arbitration lost) = 0b             ->  Not used
  //USISTP (STOP condition received) = 0b     ->  Not used
  //USISTTIFG (START condition int. flag) = 0b -> Not used
  //USIIFG (USI counter int. flag) = 0b -> No int. pending
```

14.14.3 Lab10c: Echo test using I²C

Project files

- C source files: **Chapter 14 > Lab10 > Lab10c1_student.c**
  Chapter 14 > Lab10 > Lab10c2_student.c
  Solution files: **Chapter 14 > Lab10 > Lab10c1_solution.c**
  Chapter 14 > Lab10 > Lab10c2_solution.c

Overview

This laboratory explores the USCI and USI communication interfaces in I²C mode. It uses the two MSP430 devices included on the Experimenter's board: MSP430FG4618 as the master and the MSP430F2013 as the slave. The master receives a single byte from the slave as soon as a button connected to P1.0 is pressed.

A. Resources

This laboratory uses the USCI module of the MSP430FG4618 device and the USI module included in the MSP430F2013. Both units operate in I²C mode.

The interrupts on the slave unit are generated exclusively by the USI module. They are:

- START condition in the I²C bus;
- Data reception and transmission.

The interrupts on the master unit are provided by the USCI module. They are:

- Data reception;
- Interrupt on Port1.

The resources used are:

- USCI module;
- USI module;
- Interrupts;
- I/O ports.

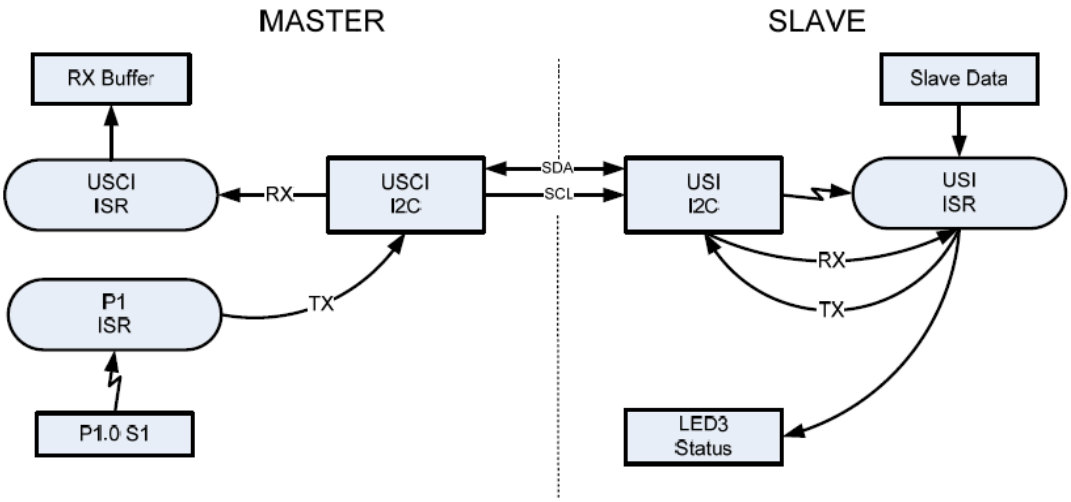
B. Software application organization

The software architecture for this laboratory is shown in Figure *14-46*.

The master task is composed of two interrupt service routines:

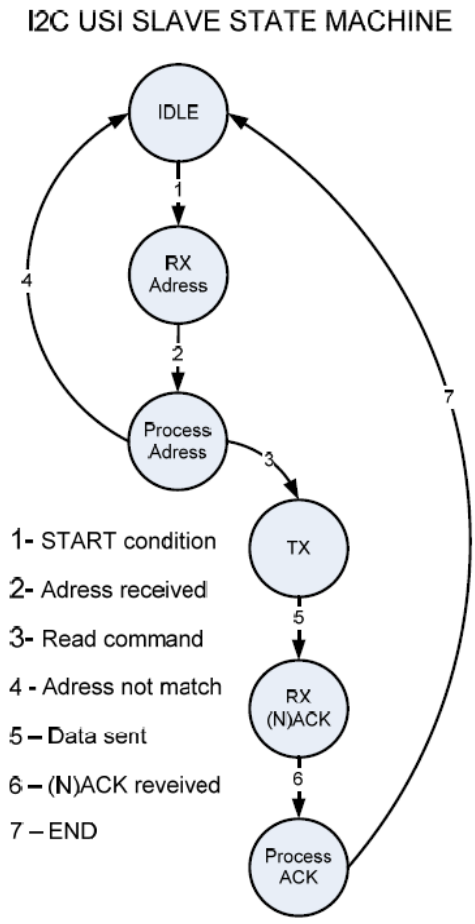
- S1 switch service routine used to receive a new frame from the slave;
- USCI module interrupt service routine that reads the data sent by the slave.

Figure 14-46. Lab10c: Software architecture.



For the operational capability of the slave unit based on the USI module, it is necessary to implement a state machine as shown in *Figure 14-47*. It is important to note that the states “RX Address” and “RX (N)ACK" are transient states that ensure the USI module is prepared for the next activity.

Figure 14-47. Lab10c: Slave state machine.



C. Configuration

- **Configure the control registers USCI_B (master)**

The connection via I^2C bus will operate in the following mode:

- Address slave with 7-bit address;
- Master mode;
- Single master;
- USCI clock source is SMCLK;

Configure the following control registers based on these characteristics:

```
UCB0CTL0 = _________________;
UCB0CTL1 = _________________;
```

❑ Data rate USCI_B (master)

The system clock is configured to operate with a frequency of ~ 1048 kHz from the DCO. This frequency will be the working base frequency of the USCI module. The connection operates at a clock frequency of ~ 95.3 kHz. Configure the following registers:

```
UCB0BR0= _________________;
UCB0BR1= _________________;
```

❑ Port configuration USCI_B (master)

In order to set the external interfaces at the USCI module, it is necessary to configure the I/O ports. Select the USCI peripheral in I^2C mode matching the connections provided at the Experimenter's board:

```
P3SEL = ___________________;
```

❑ Configure the control registers USI (slave)

The connection via I^2C bus will operate in the following mode:

- ❑ Slave mode;
- ❑ USI counter interrupt enable (RX and TX);
- ❑ START condition interrupt-enable;
- ❑ USIIFG is not cleared automatically.

Configure the following control registers based on these characteristics:

```
USICTL0 = _________________;
USICTL1 = _________________;
USICNT = _________________;
```

The slave unit interrupt service routine is not complete. The portion related to the "I2C_TX" state needs to be completed:

- ❑ Configure the USI module as output;
- ❑ Insert the information to transmit using the transmission register;
- ❑ Configure the bit counter.

```
USICTL0 |=_________________;
USISRL =_____________________;
USICNT |=_____________________;
```


D. Analysis of operation

Once the USCI module is configured in accordance with the previous steps, initiate the experiment by completing the files **Lab10c1_student.c** (master – MSP430FG4618) and **Lab10c2_student.c** (slave – MSP430F2013), compiling them and running them on the Experimenter's board. The complete solution can be found in the files **Lab10c1_solution.c** and **Lab10c2_soluction.c**.

For this laboratory, the following jumper settings are required:

- PWR1/2, BATT, LCL1/2, JP2;
- SPI: H1- 1&2, 3&4.

- **Verification**

The slave data is sent and increments from 0x00 with each transmitted byte, which is verified by the Master. The LED is off for address or data Acknowledge and the LED turns on for address or data Not Acknowledge. LED3 blinks at each data request. It is turned on with a START condition and it is turned off by the data transmit acknowledge by the slave (Note: the I^2C bus is not released by the master since the successive START conditions are interpreted as "repeated START").

Verify the value received setting a breakpoint in the line of code "**RxBuffer = UCB0RXBUF;**" of the USCI interrupt.

MSP-EXP430FG4618 (master) **SOLUTION**

Using USCI module in I^2C mode included in the FG4618 (configured as master) of the Experimenter's board, establish a connection to the F2013 using its USI module in I^2C mode. The master receives a single byte from the slave as soon as a button connected to P1.0 is pressed.

- USCI (master) control registers:

```
UCB0CTL0 = 0x0F;


//UCB0CTL0 =
//UCA10|UCSLA10|UCMM|Unused|UCMST|UCMODEx|UCSYNC|
  //UCA10 (Own address) = 0b        -> Own address (7-bit)
  //UCSLA10 (Slave address) = 0b   ->  7-bit slave address
  //UCMM (Multi-master) = 0b             ->  Single master
  //Unused
  //UCMST (Master mode) = 1b               ->  Master mode
  //UCMODEx (USCI mode) = 11b ->                  I2C Mode
  //UCSYNC (Synchronous mode enable) = 1b ->   Synchronous
```

```
UCB0CTL1 = 0x81;

//UCB0CTL1 =
//UCSSELx |Unused|UCTR|UCTXNACK|UCTXSTP|UCTXSTT|UCSWRST|
  //UCSSELx (USCI clock source select) = 10b     ->  SMCLK
  //Unused
  //UCTR (Transmitter/Receiver) = 0b          ->  Receiver
  //UCTXNACK (Transmit a NACK) = 0b       ->  ACK normally
  //UCTXSTP (Transmit STOP condition) = 0b     ->  No STOP
  //UCTXSTT (Transmit START condition) = 0b   ->  No START
  //UCSWRST (Software reset) = 1b               -> Enabled
```

- Data rate:

```
  // DATA RATE
  // data rate -> fSCL = SMCLK/11 = 95.3kHz
UCB0BR0 = 0x0B; // fSCL = SMCLK/11 = 95.3kHz
UCB0BR1 = 0x00;
```

- Configure ports:

```
P3SEL |=0x06;  // Assign I2C pins to USCI_B0
```

MSP-EXP430F2013 (slave)

SOLUTION

Using USCI module in I^2C mode included in the FG4618 (configured as master) of the Experimenter's board, establish a connection to the F2013 using its USI module in I^2C mode. The master receives a single byte from the slave as soon as a button connected to P1.0 is pressed.

- USI (slave) control registers:

```
  USICTL0 = 0XC1;

//USICTL0 =
//USIPE7|USIPE6|USIPE5|USILSB|USIMST|USIGE|USIOE|USISWRST|
  //USIPE7 (USI SDI/SDA port enable) = 1b   -> USI enabled
  //USIPE6 (USI SDO/SCL port enable) = 1b   -> USI enabled
  //USIPE5 (USI SCLK port enable) = 0b     -> SCLK disable
  //USILSB (LSB first) = 0b                  -> MSB first
  //USIMST (Master) = 0b                    -> Slave mode
```



```
  //USIGE (Output latch control) = 0b      -> Output latch
  //                          enable depends on shift clock
  //USIOE (Serial data output enable) = 0b      -> Output
enabled
  //USISWRST (USI software reset) = 1b   -> Software reset


  USICTL1 = 0x70;

//USICTL1 =
//|USICKPH|USII2C|USISTTIE|USIIE|USIAL|USISTP|USISTTIFG|…
//…USIIFG|
  //USICKPH (Clock phase select) = 0b   -> Data is changed
  //  on the first SCLK edge and captured on the following
  //                                                edge.
  //USII2C (I2C mode enable) = 1b      -> I2C mode enabled
  //USISTTIE = 1b  -> Interrupt on START condition enabled
  //USIIE = 1b             -> USI counter interrupt enable
  //USIAL (Arbitration lost) = 0b              -> Not used
  //USISTP (STOP condition received) = 0b      -> Not used
  //USISTTIFG (START condition int. flag) = 0b -> Not used
  //USIIFG (USI counter int. flag) = 0b -> No int. pending
```

- USI Bit Counter Register:

```
  USICNT |= 0x20;

//USICNT =
//USISCLREL| USI16B |USIIFGCC |USICNTx|
  //USISCLREL (SCL release) = 0b   -> SCL line is held low
  //                                    if USIIFG is set
  //USI16B (16-bit shift register enable) = 0b    -> 8-bit
  //                                  shift register mode
  //USIIFGCC (USI int. flag clear control) = 1b  -> USIIFG
  //                           is not cleared automatically
  //USICNTx (USI bit count) = 00000b (not relevant)
```

- I^2C state machine:

```
USICTL0 |= USIOE;                          // SDA = output
USISRL = SlaveData;                       // Send data byte
USICNT |= 0x08;               // Bit counter = 8, TX data
```


14.15 Quiz

1. In a parallel communication transmission mode:

(a) The data is transferred slower;

(b) Each bit of the data has its own line;

(c) All of above;

(d) None of above.

2. In an asynchronous serial communication transmission mode:

(a) The data bits arrive sequentially;

(b) The digital data is transferred faster;

(c) All of above;

(d) None of above.

3. The serial transmission mode is the most widely used digital data communication method because:

(a) Increasing bit transfer rates are achieved;

(b) It is cheaper than parallel transmission mode;

(c) All of above;

(d) None of above.

4. In serial transmission communications, the frame must include:

(a) Start and stop bits.

(b) Parity bit.

(c) All of above;

(d) None of above.

5. Even parity means that an additional bit is:

(a) Added to the data to make the sum of the “1” bits even;

(b) Subtracted from the data to make the sum of the “1” bits even;

(c) All of above;

(d) None of above.

6. A USART is used:

(a) Only for asynchronous transmissions;

(b) Only for synchronous transmissions;

(c) In parallel transmission communications;

(d) In serial transmission communications.

7. Synchronous communication performed between two USART devices requires:

(a) A common clock either in the transmitter or the receiver;

(b) No common clock.

(c) An independent clock in the transmitter;

(d) An independent clock in the receiver.

8. Asynchronous communication performed by two USART devices requires:

(a) A common clock in the transmitter and the receiver;

(b) An independent clock in the transmitter and the receiver;

(c) A common clock in the transmitter or the receiver;

(d) An independent clock in the transmitter or the receiver.

9. I^2C is a bus:

(a) Synchronous with a master and a slave where both can be the transmitter or receiver;

(b) Where the master generates the clock;

(c) All of above;

(d) None of above.

10. The USART supports the following communication modes:

(a) UART and I^2C;

(b) SPI and I^2C;

(c) UART and SPI;

(d) None of above.

11. The USART module has:

(a) One SPI module;

(b) Two SPI modules;

(c) Three SPI modules;

(d) None.

12. The UART:

(a) Transmits and receives characters at a bit rate synchronous to another device;

(b) Transmits characters at a bit rate synchronous to another device and receives characters at a bit rate asynchronous;

(c) Transmits characters at a bit rate asynchronous to another device and receives characters at a bit rate synchronous;

(d) Transmits and receives characters at a bit rate asynchronous to another device.

13. The USART character format is composed of:

(a) {Start bit, Seven data bits, Parity bit, Stop bit}.

(b) {Start bit, Eight data bits, Parity bit, Stop bits}.

(c) {Start bit, Seven data bits, Parity bit, Address bit; Stop bit}

(d) Each of the above is possible.

14. The asynchronous communication formats supported by the USART module are:

(a) Idle-line multiprocessor communication protocol;

(b) Address bit multiprocessor communication protocol;

(c) All of above;

(d) None of above.

15. The automatic error detection recognizes:

(a) Framing, Parity, Receive Overrun and Break condition errors;

(b) Framing and Parity errors;

(c) Receive Overrun and Break condition errors;

(d) Framing, Parity, Receive Overrun errors.

16. The serial clock control in SPI mode when MM = 1 is provided by the:

(a) ACLK pin on the master;

(b) BITCLK USART baud rate generator on the UCLK;

(c) All of above;

(d) UCLK pin on the master.

17. The USCI module has:

(a) One module;

(b) Two modules;

(c) Three modules;

(d) None.

18. The USCI module in UART mode supports:

(a) LIN;

(b) IrDA;

(c) All of above;

(d) None of above.

19. The UCMSB bit controls:

(a) The direction of the transfer;

(b) Selects LSB or MSB first;

(c) All of above;

(d) None of above.

20. The automatic baud rate detection is composed by a break character, which is:

(a) Detected when 11 or more continuous “0”s are received;

(b) Detected when 4 or more continuous “0”s are received;

(c) Detected when 8 or more continuous “0”s are received;

(d) None.

21. The automatic baud rate detection is composed by a synch field, which is represented by:

(a) Data 022h inside a byte field;

(b) Data 055h inside a byte field;

(c) Data 044h inside a byte field;

(d) None.

22. The USCI module in UART mode for IrDA decoding detects:

(a) Low pulse;

(b) High pulse;

(c) All of above;

(d) None.

23. The baud rate can be generated by:

(a) Low-frequency;

(b) Oversampling;

(c) All of above;

(d) None of above.

24. In USCI I^2C communication, the ACK bit is sent from the receiver after:

(a) Each bit on the 9th SCL clock;

(b) Each byte on the 2nd SCL clock;

(c) Each bit on the 2nd SCL clock;

(d) Each byte on the 9th SCL clock.

25. The operating modes provided by the I^2C mode are:

(a) Master transmitter and Slave receiver;

(b) Slave transmitter and Master receiver;

(c) All of above;

(d) None of above.

26. The I^2C state change interrupt flags are:

(a) Arbitration-lost and Not-acknowledge;

(b) Start and stop conditions;

(c) All of above;

(d) None of above.

27. The USI module has:

(a) SPI;

(b) I^2C;

(c) All of above;

(d) None of above.

28. The internal USI clock generation can use:

(a) ACLK and SMCLK;

(b) ACLK and MCLK;

(c) SMCLK and MCLK;

(d) None of above.

29. The USISR shift register supports:

(a) 8 bits;

(b) 16 bits;

(c) All of above;

(d) None of above.

30. The USIIFG is set when:

(a) Bit counter counts up to 0xFF;

(b) Bit counter counts down to 0x00;

(c) Bit counter counts up to 0x80;

(d) Bit counter counts up to 0x08.

31. After address/data reception the receiver ACK/NACK is:

(a) SDA = input: 0 = ACK, 1 = NACK;

(b) SDA = output: 0 = ACK, 1 = NACK;

(c) SDA = input: 1 = ACK, 0 = NACK;

(d) SDA = output: 1 = ACK, 0 = NACK.

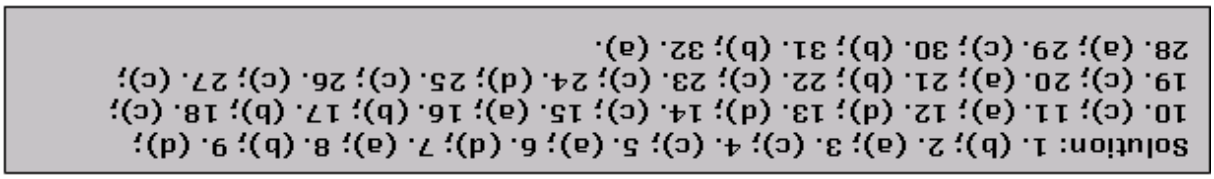
32. After address/data transmission the transmitter ACK/NACK is:

(a) SDA = input: 0 = ACK, 1 = NACK;

(b) SDA = output: 0 = ACK, 1 = NACK;

(c) SDA = input: 1 = ACK, 0 = NACK;

(d) SDA = output: 1 = ACK, 0 = NACK.



14.16 FAQs

1. How is the transmission mode re-enabled by the USART's receiver in UART mode?

If the receiver is disabled (URXEx = 0), re-enabling it is asynchronous to any data stream that may be present on URXDx at that time. Synchronization can be performed by testing for an idle line condition, before receiving a valid character. Set the URXWIE and the URXEIE bits of the UxRCTL, USART Receive Control Register. In this case, only the received address characters will set URXIFGx.

2. Can a break be detected when the receive start-edge detect feature in USART is used in UART mode?

No. If the UART Clock (BRCLK source) is off, the break condition is not detected.

3. Why can't I get the correct data transmission by writing to the UxTXBUF using the USART in SPI mode?

It may be an issue of configuration errors. Try first to find if UTXIFGx = 0 and USPIEx = 1 when writing data to the UxTXBUF. This may result in erroneous data transmission.

4. What is the default character format for USCI in SPI mode?

The default SPI character transmission is LSB first. Note that the MSB-first mode may be required for communication with other SPI interfaces.

5. Is there any voltage limit for the MSP430 SDA and SCL pins levels when using the USCI in I^2C mode?

The MSP430 SDA and SCL pins must not be pulled up above the MSP430 VCC level.

6. How can I ensure that when initiated by the master in a multiple consecutive transactions, the current transaction is completed, without using a repeated start condition in the USCI I^2C mode?

Ensure that the transmit stop condition flag UCTXSTP is cleared before the next I^2C transaction is initiated by setting UCTXSTT = 1. Otherwise, the current transaction might be affected.

7. When working with I^2C datasheets, it says the peripheral does not ACK. How do I implement this?

I^2C always uses an acknowledge of some type. Here it means that the peripheral performs **not ACK** (negative acknowledge).

