Lecture 11
Communications Introduction & USI Module

Texas Instruments Incorporated
University of Beira Interior (PT)

Pedro Dinis Gaspar, António Espírito Santo, Bruno Ribeiro, Humberto Santos
University of Beira Interior, Electromechanical Engineering Department
www.msp430.ubi.pt

>> Contents

Contents (1/2)

Contents (2/2)

Introduction

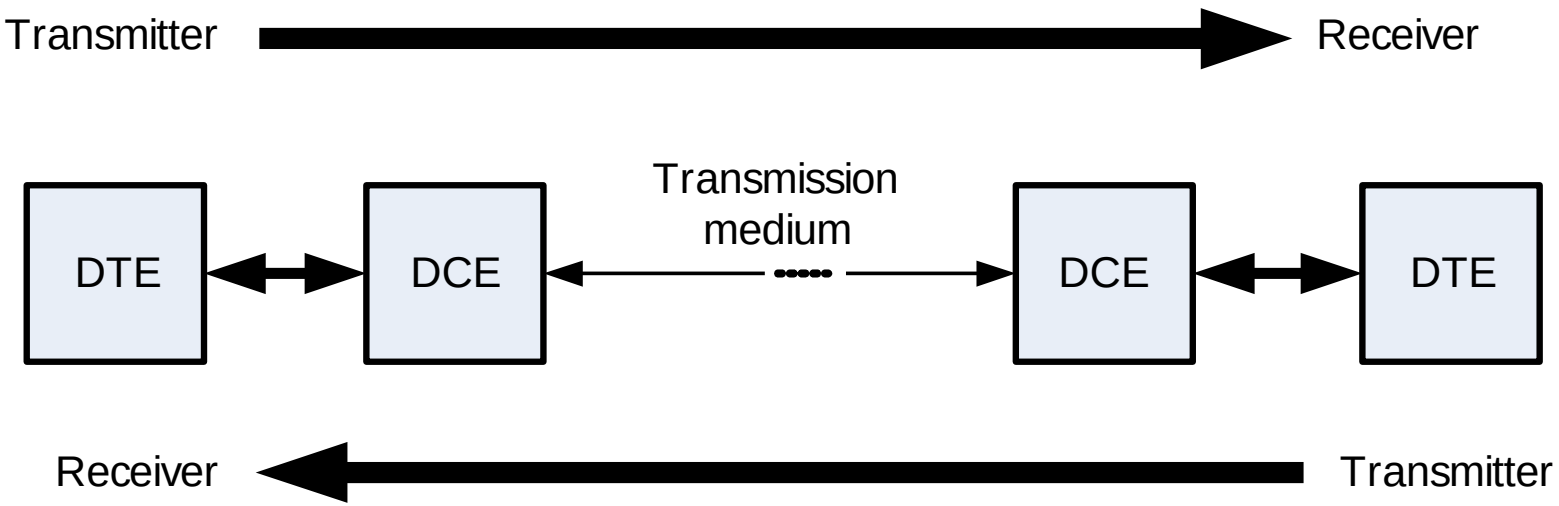
- An important feature of modern microprocessor based systems is their communication capability, that is, their ability to exchange information with other systems in the surrounding environment;
- At the low level, communications interfaces are used to download a firmware update or to set up local configurations (e.g. turn features on or off), amongst other tasks;
- At a higher level, communication interfaces are used to exchange information in distributed applications.

Communications system model (1/2)

- **Digital communication devices:**
  - ***Transmitter***: Has the task of putting the information into the appropriate format for subsequent transmission;
  - ***Receiver***: Is responsible for collecting the message that has been sent and extracting the original information;
  - ***Communication medium***: The physical medium through which the information flows and is commonly implemented as:
    - Twisted pair wire;
    - Fibre optic cable;
    - Radio frequency transmission.

Communications system model (2/2)

- **Devices participating in a digital communication system:**
  - DTE: Data Terminal Equipment;
  - DCE: Data Communications Equipment.



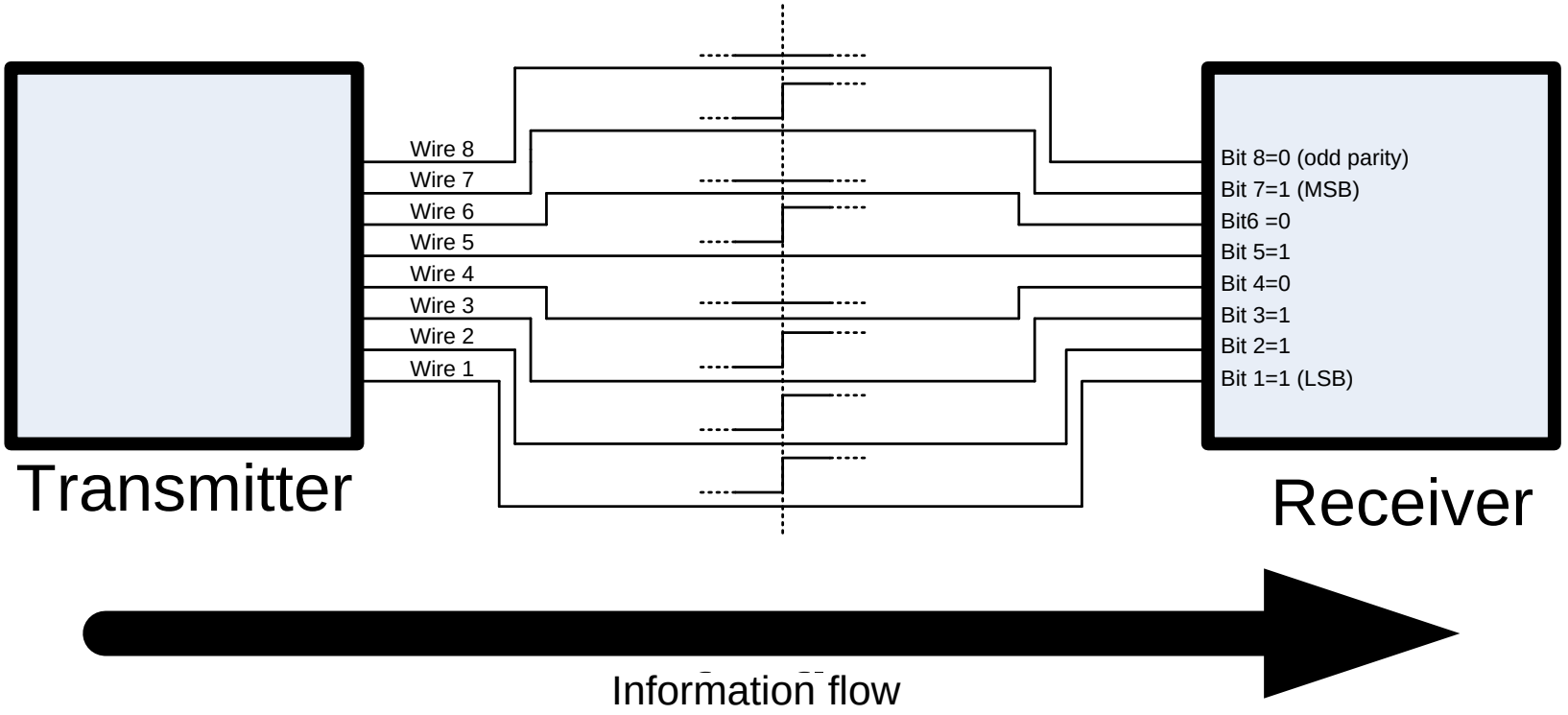
Transmission mode (1/5)

- **Communications between digital devices can be divided into two types :**
  - Parallel communications;
  - Serial communications.

- **Parallel communications:**
  - The physical transmission medium has independent signal lines in numbers equal to the transmitted digital word bits;
  - The information transmitted at any given instant is the data word formed by the logical levels on the various signal lines.

Transmission mode (2/5)

- **Parallel communications:**
  - Example: Character ASCII "W" parallel transmission.

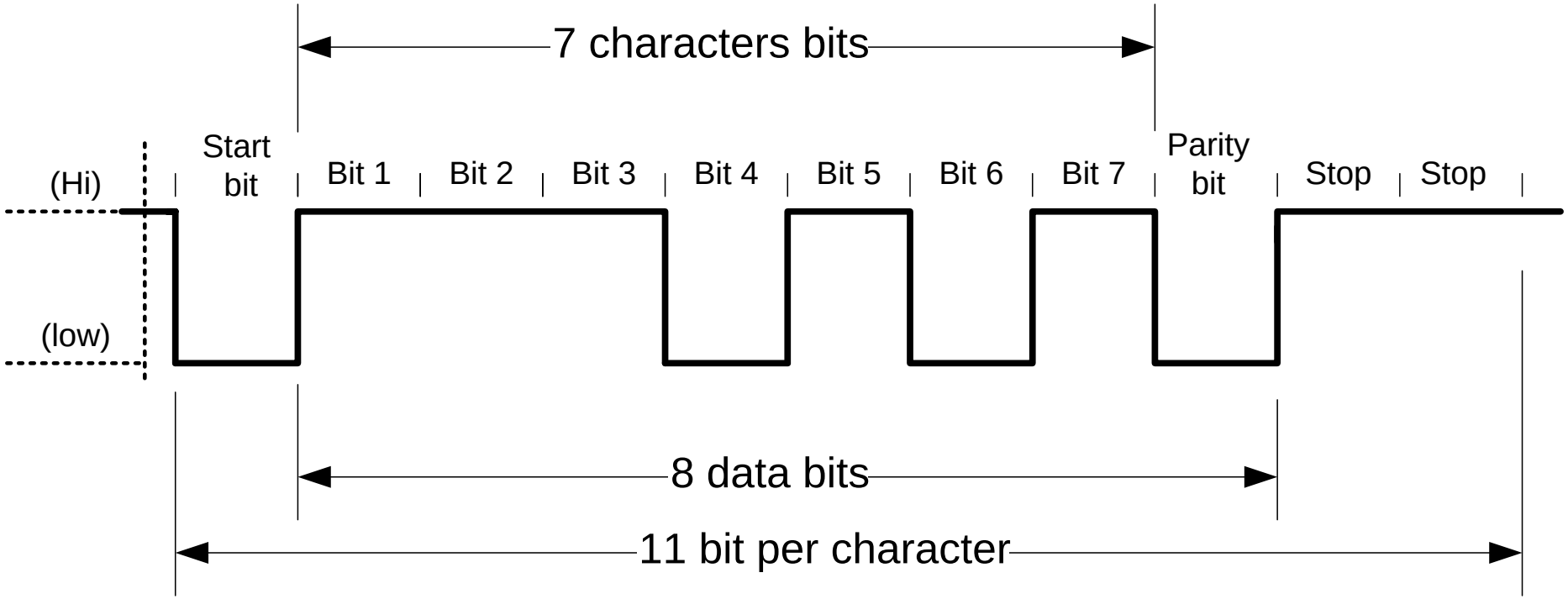


Transmission mode (3/5)

- **Serial communications:**
  - Physical transmission medium needs only one signal line;
  - The information transmitted is provided by the transmitter as a sequence of bits, sent at the rate established between the transmitter and the receiver;
  - Additional information is needed to enable the synchronization between the receiver and transmitter:
    - Start bit: Added to the beginning of the information transmitted, so that the receiver can identify the beginning of a new transmission;
    - Stop bit(s): Added to the end of the information transmitted to indicate that the data value is complete.

Transmission mode (4/5)

- **Serial communications:**
  - Example: Character ASCII "W" serial transmission:



Transmission mode (5/5)

- **Advantages and disadvantages of parallel and serial communication:**

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

Serial communications (1/3)

- The start bit identifies the beginning of a data transfer and is generated by a high-to-low transition on the bus;
- Following the start bit are the data bits. In this example, the ASCII code for the text transfer uses seven data bits;
- The error-checking bit (parity bit) is sent after the data bits;
- To terminate the transmission, one or two stop bits are issued;
- Using seven data bits, the complete message can use one or two stop bits. Using eight data bits, only one stop bit is available for transmission.

Serial communications (2/3)

- **Parity bit:**
  - Used to verify the integrity of information transmitted;
  - The bit is added by the transmitter and indicates whether the total sum of the numbers "1" in the message data is odd or even;
  - The transmissions can be configured for 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

Serial communications (3/3)

- **Baud rate example:**
  - The transmission of "W":
    - Character uses seven data bits;
    - Four bits are used for control, making a total of 11 bits.
    - This corresponds to 11 baud;
    - If the characters are transmitted at a rate of 10 characters per second, the baud rate will be:

      10x11 = 1100 baud/s.

Synchronous and asynchronous serial communications (1/2)

- **Serial communications may be:**
  - Asynchronous: where the transmission rate (baud rate) is fixed by the transmitter and the receiver works at the same baud rate, using the transmitted start bit to synchronize the start of a new message;
  - Synchronous: where there is a separate synchronization clock signal connected between the receiver and the transmitter.

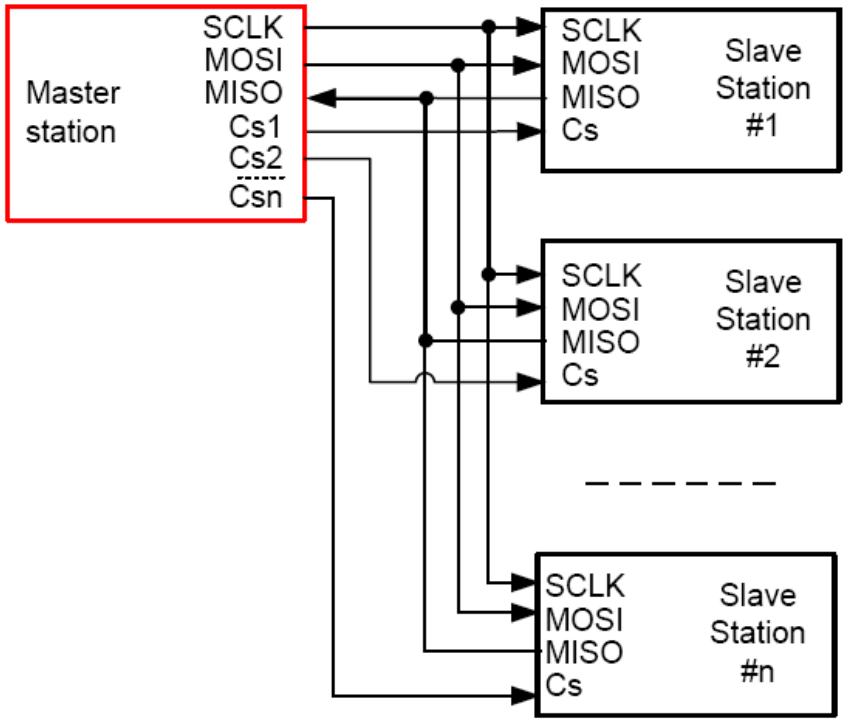
- **Synchronous communications:**
  - Normally one unit assumes the role of master and one or more of the other units take the role of slaves;
  - The clock signal generated by the master is used by the slave units to transfer data in/out of the TX and RX registers;
  - It is possible for a device to transmit and receive simultaneously.

Synchronous and asynchronous serial communications (2/2)

- **Asynchronous communications:**
  - Characterised by the absence of any synchronization clock signal between the units;
  - The transmission in this mode does not allow simultaneous transmission and reception, that is, when one device transmits the other devices just listen.

Serial Peripheral Interface (SPI) protocol (1/2)

- The Serial Peripheral Interface (SPI) bus is a standard form of synchronous serial communication;
- Developed by Motorola;
- Operates in full duplex mode;
- Master/Slave relationship;
- Communication is always initiated by the master.
- Low cost.



Peripheral Interface Serial (SPI) protocol (2/2)

- Supports only one master;
- Can support more than a slave;
- Short distance between devices, e.g. on a printed circuit boards (PCBs);
- Special attention needs to be observed to the polarity and phase of the clock signal;
- The master sends data on one edge of clock and reads data on the other edge. Therefore, it can send/receive at the same time.

I²C (Inter-Integrated Circuit) protocol (1/3)

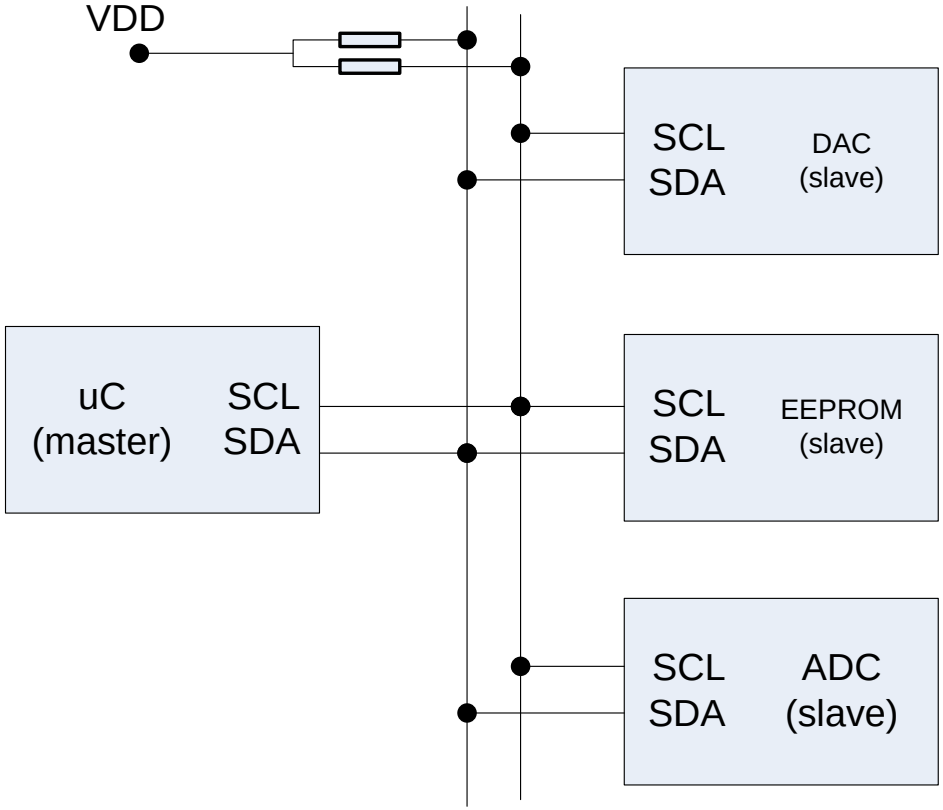
- **Multi-master synchronous serial computer bus;**
- **Invented by Philips Semiconductors;**
- **Developed with the main objective of establishing links between integrated circuits and to connect low-speed peripherals;**
- **Based on two bi-directional open-drain lines pulled up with resistors:**
  - SDA: Serial Data;
  - SCL: Serial clock.
- **Typical voltages used are +5.0 V or +3.3 V, although systems with other voltages are possible.**

I²C (Inter-Integrated Circuit) protocol (2/3)

- **Communications is always initiated and completed by the master, which is responsible for generating the clock signal;**
- **In more complex applications, I²C can operate in multi-master mode;**
- **The slave selection by the master is made using the seven-bit address of the target slave;**
- **The master (in transmit mode) sends:**
  - Start bit;
  - 7-bit address of the slave it wishes to communicate with;
  - A single bit representing whether it wishes to write (0) to or read (1) from the slave;
  - The target slave will acknowledge its address.

I²C (Inter-Integrated Circuit) protocol (3/3)

- **Example of an I²C communication system:**



>> Contents

MSP430 communications interfaces (1/2)

- **Equipped with three serial communication interfaces:**
  - **USART** (Universal Synchronous/Asynchronous Receiver/Transmitter):
    - UART mode;
    - SPI mode;
    - I^2C (on 'F15x/'F16x only).
  - **USCI** (Universal Serial Communication Interface):
    - UART with Lin/IrDA support;
    - SPI (Master/Slave, 3 and 4 wire modes);
    - I^2C (Master/Slave, up to 400 kHz).
  - **USI** (Universal Serial Interface):
    - SPI (Master/Slave, 3 & 4 wire mode);
    - I^2C (Master/Slave, up to 400 kHz).

MSP430 communications interfaces (2/2)

❑ Comparison between the communication modules:

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

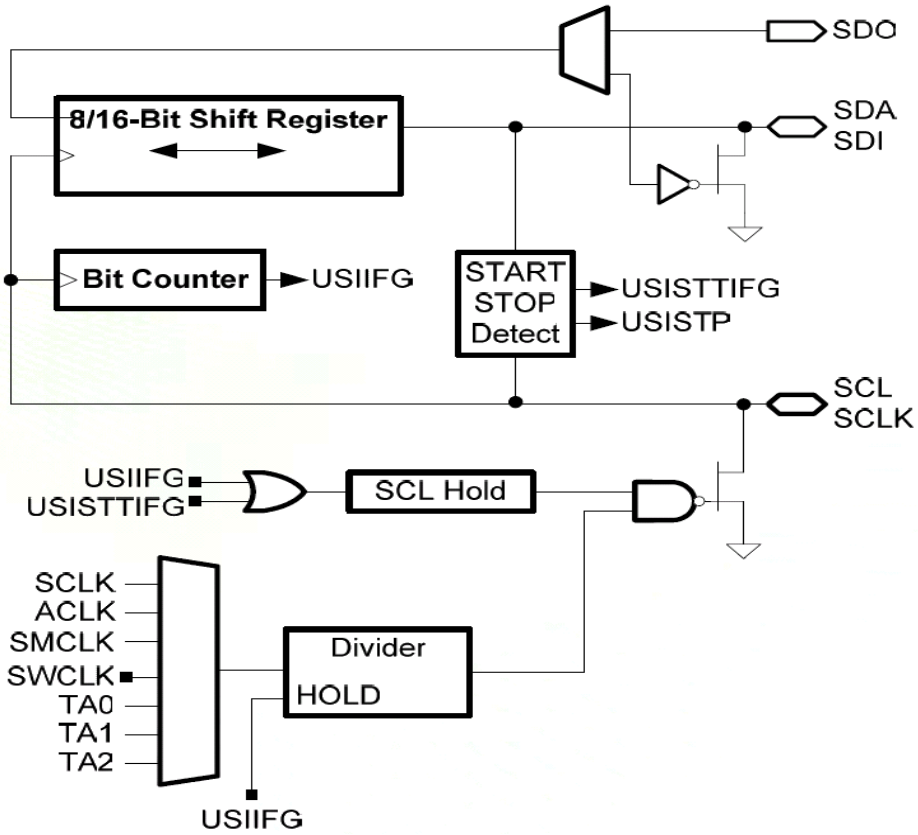
USI module introduction (1/2)

- **The USI (Universal Serial Interface) module supports basic SPI and I²C synchronous serial communications;**
- **It is available in the MSP430x20xx family of devices;**
- **The USI module supports:**
  - SPI or I²C modes;
  - Interrupt driven;
  - Reduces CPU load;
  - Flexible clock source selection.

USI module introduction (2/2)

USI block diagram:

- SPI mode:
  - Programmable data length (8/16-bit shift register);
  - MSB/LSB first.
- I²C mode:
  - START/STOP detection;
  - Arbitration lost detection.
- Interrupt driven;
- Reduces CPU load;
- Flexible clock source.



USI operation: SPI and I²C modes (1/5)

- **Shift register and bit counter that include logic to support SPI and I²C communication;**
- **USISR shift register (up to 16 bits supported):**
  - Directly accessible by software;
  - Contains the data to be transmitted/received (simultaneously);
  - MSB or LSB first.
- **Bit counter:**
  - Controls the number of bits transmitted/received;
  - Counts the number of sampled bits;
  - Sets USIIFG when the USICNTx = 0 (decrementing or writing zero to USICNTx bits);
  - Writing USICNTx > 0 automatically clears USIIFG when USIIFGCC = 0 (automatically stops clocking after last bit).

USI operation: SPI and I²C modes (2/5)

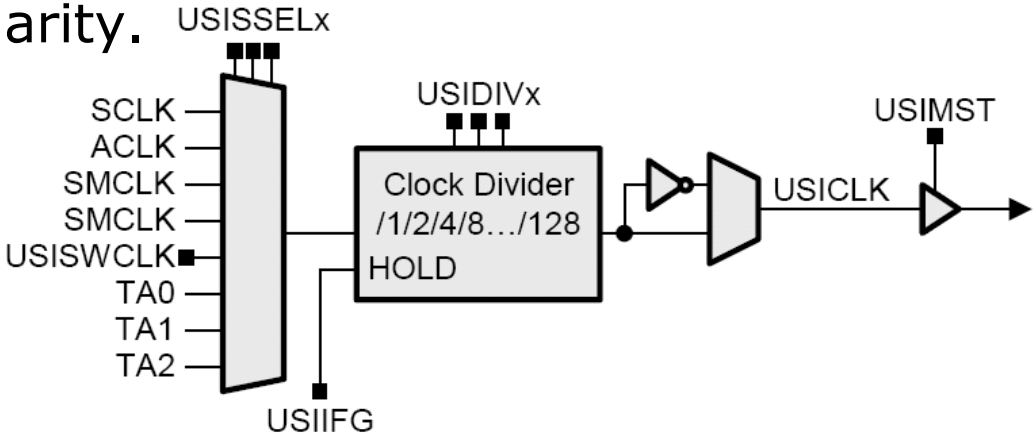
- **USI initialization:**
  - Reset USISWRST;
  - Set USIPEx bits (USI function for the pin and maintains the PxIN and PxIFG functions for the pin):
    - Port input levels can be read via the PxIN register by software;
    - Incoming data stream can generate port interrupts on data transitions.

USI operation: SPI and I²C modes (3/5)

- **Recommended USI initialization process:**
  - Set the USIPEx bits in the USI control register (USI function for the pin and set up the PxIN and PxIFG functions for the pin as well);
  - Set the direction of the RX and TX shift register (MSB or LSB first) by USILSB bit;
  - Select the mode (master or slave) by USIMTS bit;
  - Enable or disable output data by USIOE bit;
  - Enable USI interrupts by setting USIIE bit;
  - Set up USI clock by configuring the USICKCTL control register;
  - Enable USI by setting USISWRST bit;
  - Read port input levels via the PxIN register by software;
  - Incoming data stream will generate port interrupts on data transitions.

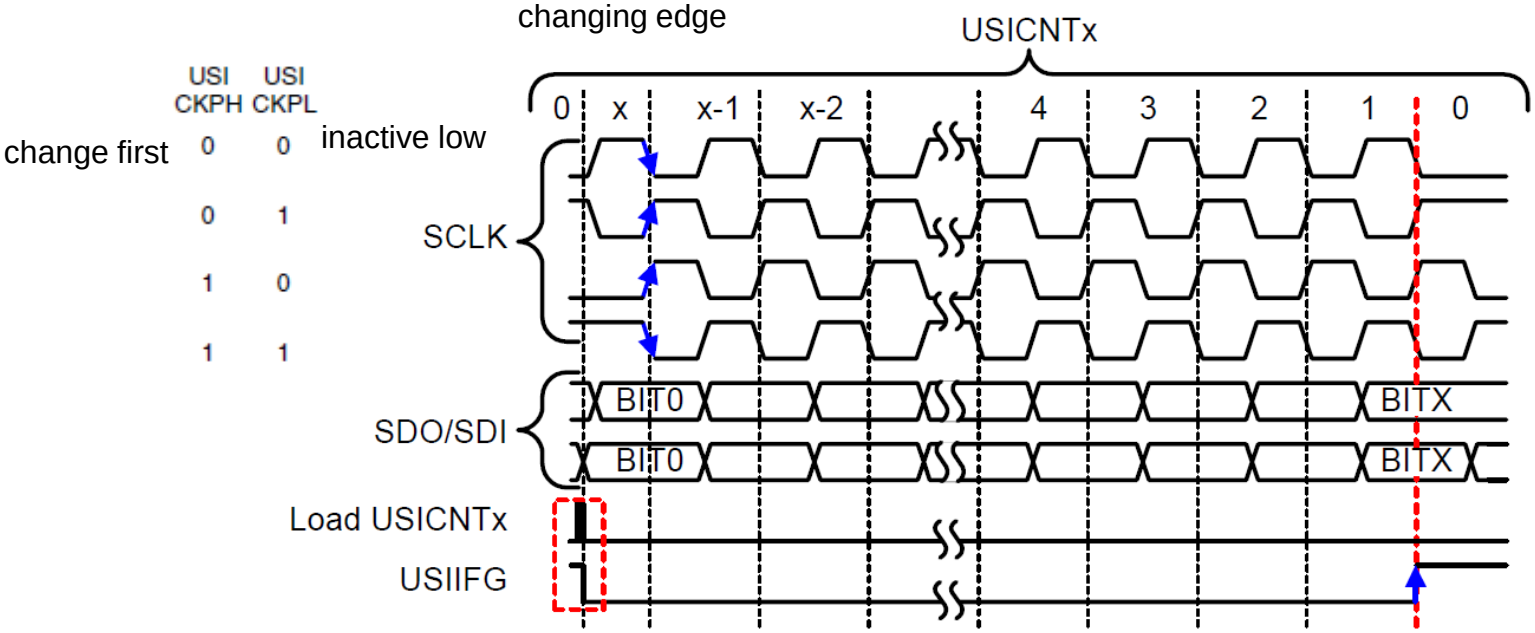
USI operation: SPI and I²C modes (4/5)

- **USI clock generation:**
  - 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.



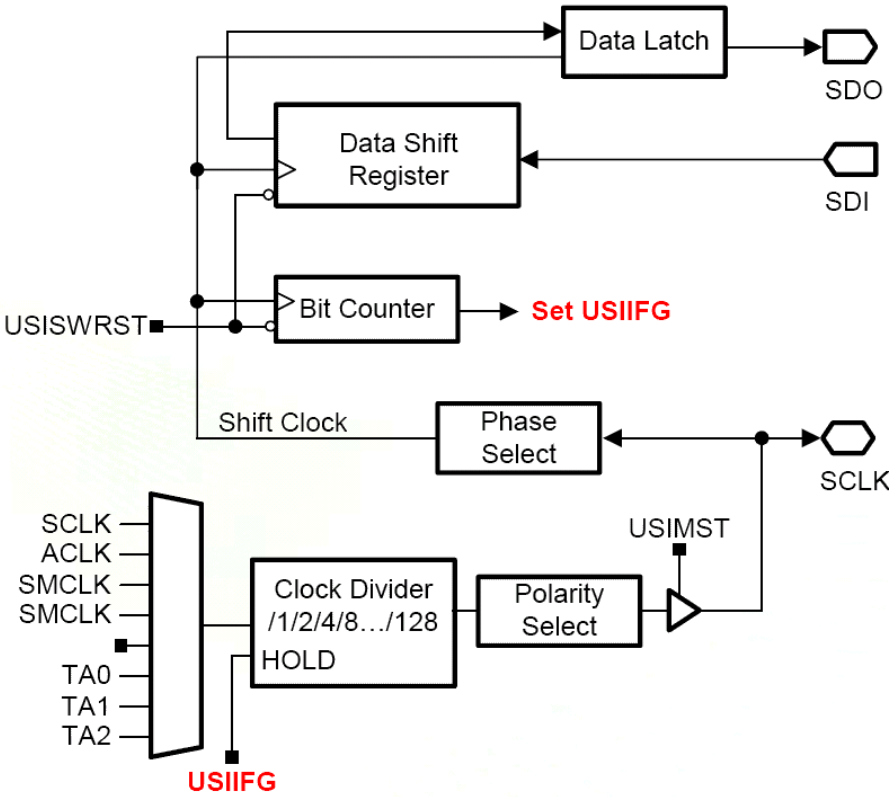
USI operation: SPI and I²C modes (5/5)

- **USICKPL: Selects the inactive level of the SPI clock (data latching on rising or falling edge);**
- **USICKPH: Selects the clock edge on which SDO is updated and SDI is sampled (idle high or low support).**
- **USIIFG automatically cleared and set by USICNTx;**
- **Clock stop on IFG: USIIFG and USISTTIFG.**



USI operation: SPI mode (1/2)

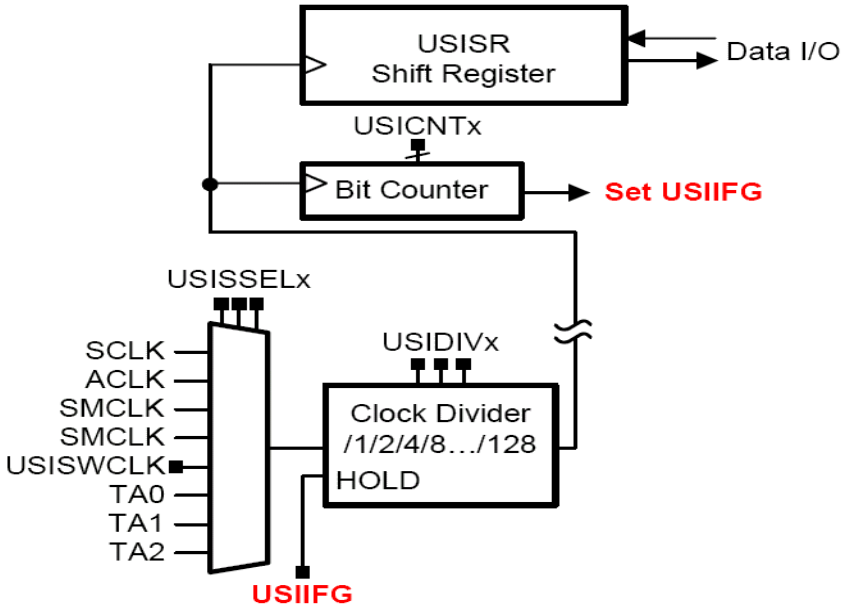
- **Configure SPI mode:**
  - SPI master:
    - USIMST = 1;
    - USII2C = 0;
    - Select clock source;
    - SCLK -> output.
  - SPI slave:
    - USIMST = 0;
    - USII2C = 0;
    - SCLK -> input;
    - Receives the clock externally from the master.



- **USIPEx bits enable data and clock pins;**
- **Port logic functions, including interrupts as normal;**
- **Data output latched on shift clock.**

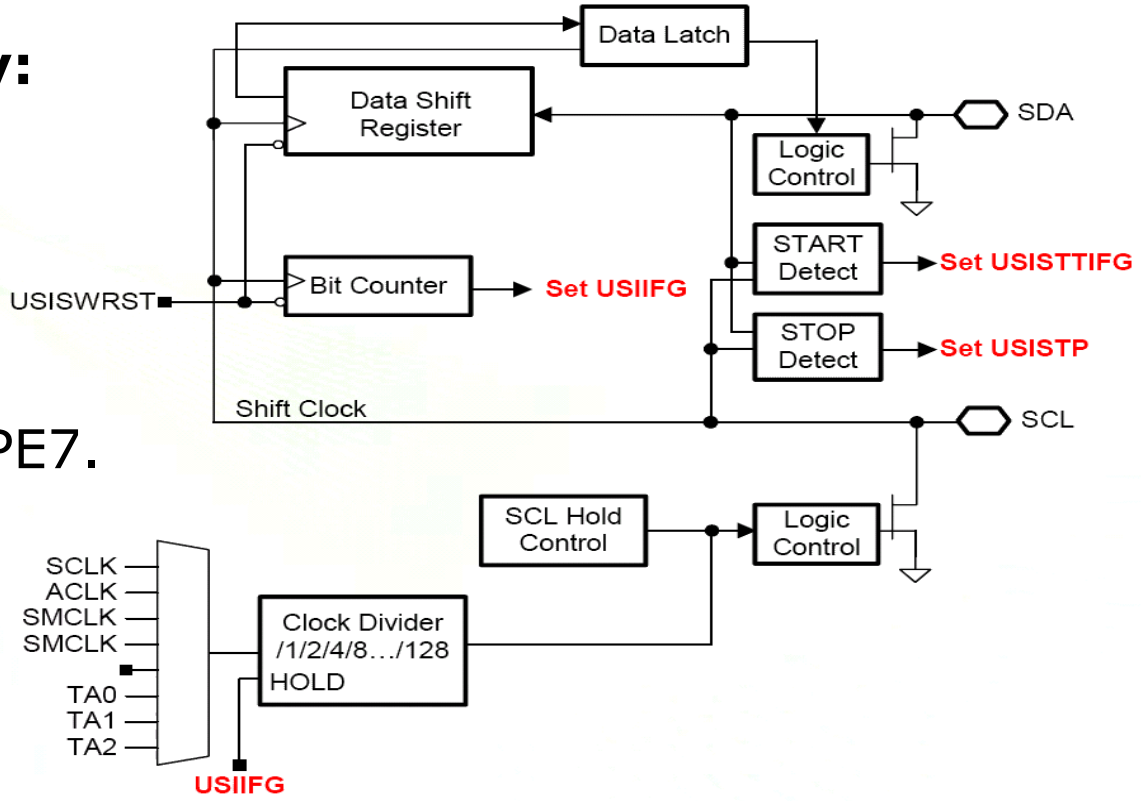
USI operation: SPI mode (2/2)

- **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 clock when set.



USI operation: I²C mode (1/10)

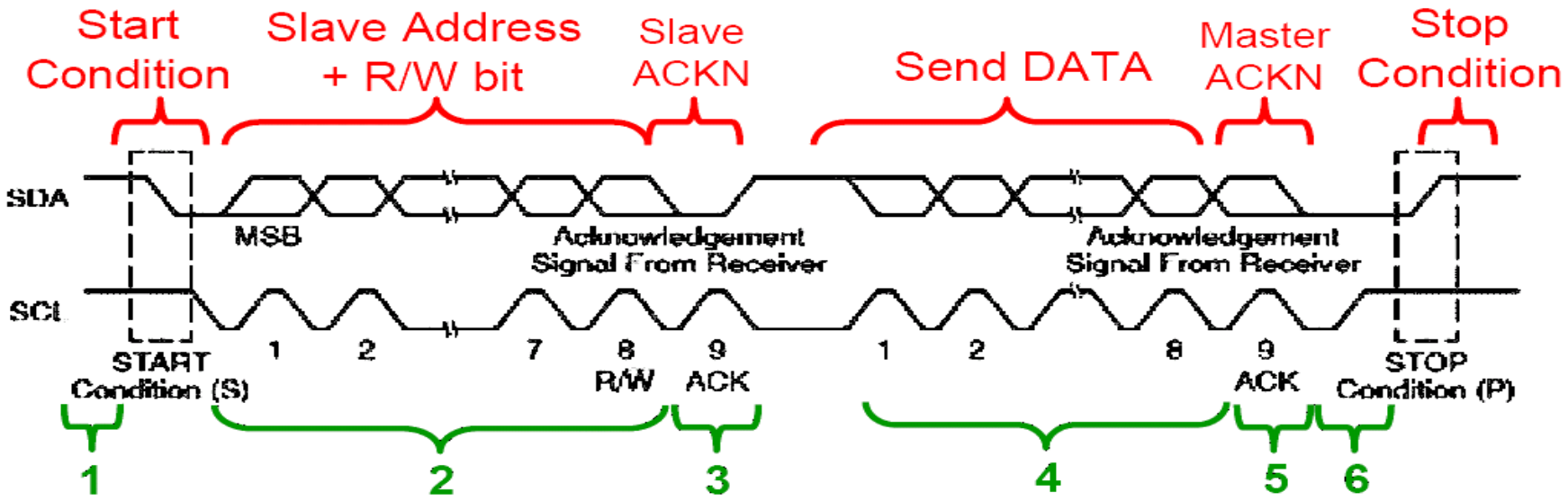
- **Configure USI module in I²C mode:**
  - USII2C =1;
  - USICKPL = 1;
  - USICKPH = 0;
- **I²C data compatibility:**
  - USILSB = 0;
  - USI16B = 0;
- **Enable SCL and SDA port functions:**
  - Set USIPE6 and USIPE7.



USI operation: I²C mode (2/10)

- **I²C master:**
  - USIMST = 1 and USII2C = 1;
  - Select clock source (output to SCL line while USIIFG = 0).

- **I²C slave:**
  - USIMST = 0;
  - SCL is held low if USIIFG=1, USISTTIFG=1 or if USICNTx=0.



USI operation: I²C mode (3/10)

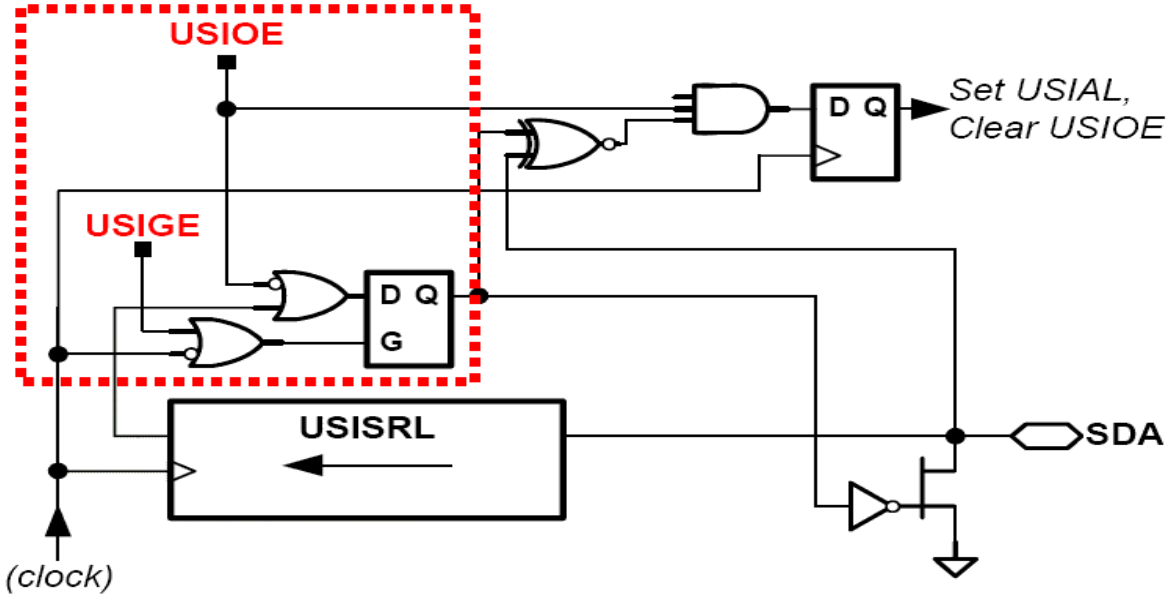
- **I²C transmitter:**
  - Data value is first loaded into USISRL;
  - USIOE= 1: Enable output and start transmission (writes 8 into USICNTx);
  - Send Start (or repeated Start);
  - Define address and set R/W;
  - Slave ACK: (Data TX/RX + ACK for N bytes);
  - SCL is generated in master mode or released 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).

USI operation: I²C mode (4/10)

- **I²C receiver:**
  - Clear USIOE (disable output);
  - Enable reception 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).

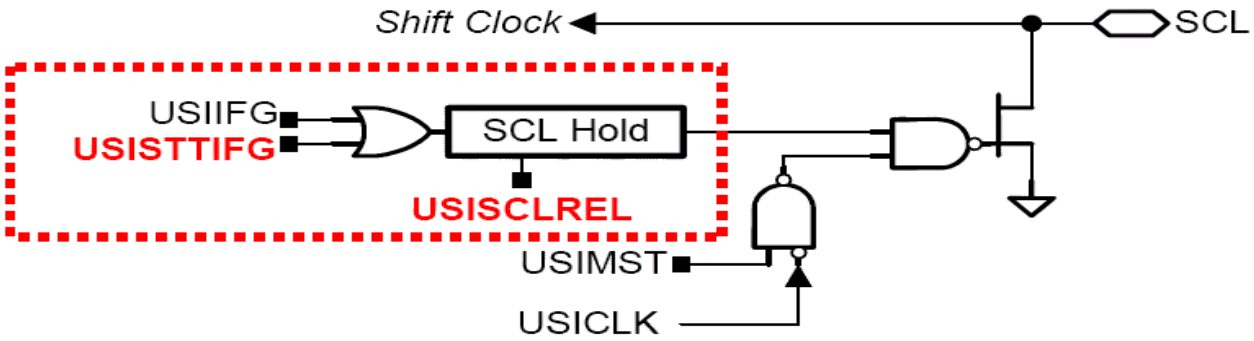
USI operation: I²C mode (5/10)

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



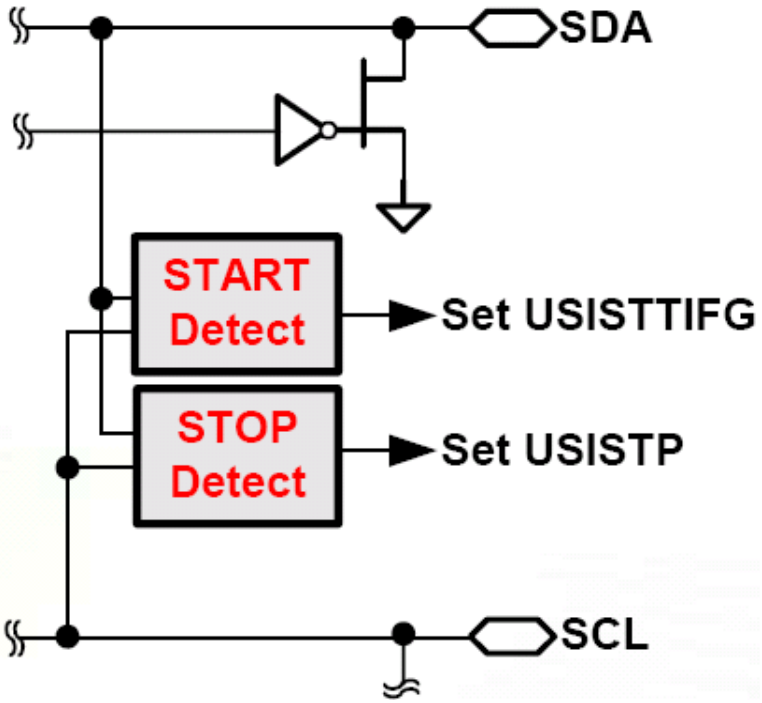
USI operation: I²C mode (6/10)

- **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).



USI operation: I²C mode (7/10)

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



USI operation: I²C mode (8/10)

- **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);**

USI operation: I²C mode (9/10)

- **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).

USI operation: I²C mode (10/10)

- **Example:**
  - Procedure for I²C communication between a Master TX and a 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	

USI registers (SPI and I²C modes) (1/8)

❑ USICTL0, USI Control Register 0

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

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

USI registers (SPI and I²C modes) (2/8)

❑ USICTL0, USI Control Register 0 (continued)

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

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

USI registers (SPI and I²C modes) (3/8)

❑ 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	I2C mode enable: USII2C = 0 ⇒ I2C mode disabled USII2C = 1 ⇒ I2C 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

USI registers (SPI and I²C modes) (4/8)

❑ USICTL1, USI Control Register 1 (continued)

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

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

USI registers (SPI and I²C modes) (5/8)

❑ 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

USI registers (SPI and I²C modes) (6/8)

❑ USICKCTL, USI Clock Control Register (continued)

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

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

USI registers (SPI and I²C modes) (7/8)

❑ 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)

USI registers (SPI and I²C modes) (8/8)

❑ 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