Application Note, V1.0, Jan. 2008

AP16130

XC2000 & XE166 Families

USIC Getting Started (IIC)

Microcontrollers

Never stop thinking

Edition 2008-02

Published by
Infineon Technologies AG
81726 Munich, Germany

AP16130

Revision History:	2008 - 01	V1.0
Previous Version:	none	

Page	Subjects (major changes since last revision)

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com

Table of Contents **Page**

1 Introduction

The USIC is a brand new serial communication module in Infineon Microcontroller XC2000/XE166 family. The Universal Serial Interface Channel module (USIC) is designed to support several serial communication protocols in a single module. This USIC module gives the flexibility to realize any one of the supported serial communication protocols.

This application node gives a brief overview of the USIC module and explains about realizing the Inter Integrated Circuit IIC bus using USIC module. The configurations of IIC Master mode and Slave mode and prototype applications for Master and Slave communction with XC2000 family Infineon Microcontrollers also discussed. For futher details about the product and module, please check at **www.infineon.com.**

2 Universal Serial Interface Channel (USIC)

The Universal Serial Interface Channel (USIC) is a flexible interface module that supports the protocols like ASC, SSC, IIC and IIS. The user can program during run-time desired protocols on the USIC communicaton channel. The protocol can also be changed during run-time without a reset.

The USIC concept is based on a generic data shift and data storage structure that is identical for many serial communication protocols. The protocol part, which is responsible for the generation of the shift clock, shift data and shift control signals, is independent from the general part.

2.1 USIC Channel Structure

The USIC module has two independent communication channels and each channel can be configured as one of the supported protocols. Each channel provides a vector of pins; the selected protocol can be configured with any of the pins from the vector. Each channel can have independent programmable baudrate configurations and also FIFO buffers for transmit and receive paths.

- Data buffer unit – has Transmit and Receive buffers and optional FIFO and bypass register.
- Data shift unit – serial-to-parallel for reception and the parallel-to-serial for transmission.
- Protocol Pre-processors – handle the protocol-specific actions.
- Input stages – for signal conditioning (input selection, polarity control, and digital input filter).
- Baud rate generator – Independent baud rate configuration for each channel.

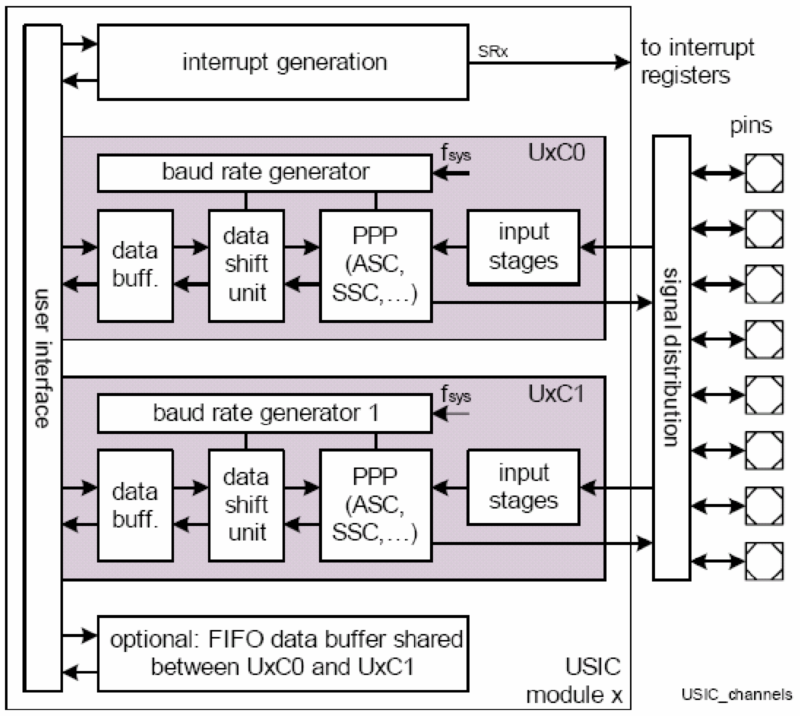

Figure 1 The USIC channel structure

2.2 Input Stages

There are three input stages to handle each one of the three input signals, the number of actually used inputs depends on the selected protocol. The input stages are, DX0 – handles shift data input, DX1 – handles shift clock input and DX2 – handles shift control input. These input stages provide the configurations to:

- Select the input signal from the input vector.
- Input switch between direct input signal and protocol pre-processor output.
- Digital filter and data synchronization.
- Signal polarity of the input signal.

2.3 Output Signals

There are up to 11 protocol related output signals available; the number of actually used output signals depends on the selected protocol. The output signals are classified according to their meaning for the protocols and the operation modes.

- Shift data output signal DOUT.
- Shift clock output signal SCLKOUT.
- Shift control outputs SELO [7:0].
- Master clock output MCLKOUT.

2.4 Baud Rate Generator

The Baud rate generator provides the frequencies needed for the different protocols. Each channel has the independent baud rate generator which contains:

- A fractional divider to generate the input frequency f_{PIN} = f_{FD} for baud rate generation based on the internal system frequency f_{SYS}.
- The DX1 input to generate the input frequency f_{PIN} = f_{DX1} for baud rate generation based on an external signal.
- A protocol-related counter to provide the master clock signal MCLK, the shift clock signal SCLK, and other protocol-related signals. It can also be used for time interval measurement, e.g. baud rate detection
- A time quanta counter associated with the protocol pre-processor defining protocol specific timings, such as shift control signals or bit timings, based on the input frequency f_{CTQIN}.
- The output signals MCLKOUT and SCLKOUT of the protocol-related divider that can be made available on the pins. In order to adapt to different applications, some output characteristics of these signals can be configured.

2.5 Channel Events and Interrupts

The channel events are classified based on data traffic and data handling,

- Data transfer events related to the transmission or reception of a data word.
- Protocol-specific events depending on the selected protocol.
- Data buffer events related to data handling by the optional FIFO data buffers.

2.6 Data Shifting and Handling

A data buffer structure and an independent data shift unit handle the data shifting and handling. The shift data, shift clock and shift control signals are input to data shift unit. The data handling comprises:

- A transmitter with a transmit shift register in the DSU (Data Shift Unit) and transmit data buffer. A data validation scheme allows triggering and gating of data transfers by external events under certain conditions.
- A receiver with two alternate receive shift registers in the DSU and a double receive buffer structure. The alternate receive shift registers support the reception of data streams and data frames longer than one data word.
- Optional transmit and receive FIFO buffers.

3 Inter Integrated Circuit (IIC)

The USIC channel can be programmed to work as IIC protocol with following features:

- Two wire interface, one for shift clock transfer SCL and other one for shift data SDA.
- Communcation in standard mode (100 kbps) and fast mode (400 kbps).
- Operation modes: Master, slave and multi-master.
- Addressing modes: 7 bit and 10 bit addressing.
- Powerful interrupt handling and efficient frame handling.

3.1 IIC Signals, Symbols and Frame format

The IIC connection is a two wire interface and it carries the signal SCL and SDA. The output driver for these signals must have open-drain characteristic to allow the wired-AND connection to form the IIC bus system.

- Shift data SDA: input handled by DX0 stage, output signal DOUT.
- Shift clock SCL: input handled by DX1 stage, output signal SCLKOUT.

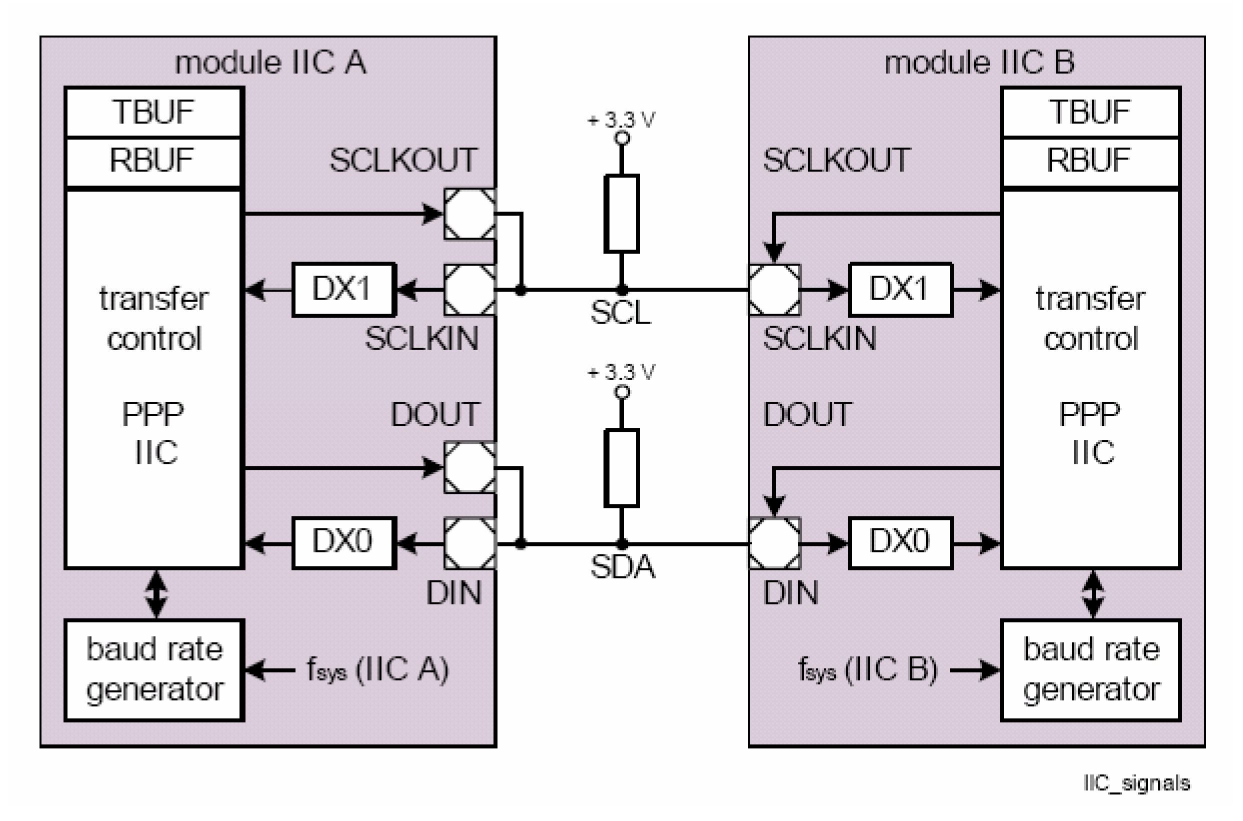

Figure 2 The USIC IIC signals

IIC Symbols

The IIC symbol is a sequence of edges on the lines SDA and SCL. As per the standard USIC generates signals for IIC symbols which contains 10 or 25 time quanta, depending on the selected baud rate.

- Bus idle – SDA and SCL are high. No data transfer takes place currently.
- Data bit symbol – SDA stable during the SCL high. During data transfers SDA may only change while SCL is low.
- Start symbol – SDA being high followed by a falling edge of SDA while SCL is high. This start condition initiates a data transfer over the IIC bus after the bus has been idle.
- Repeated start symbol – SDA is set high and SCL low, followed by a start symbol. This condition initiates a data transfer over the bus after a data symbol when the bus has not been idle.
- A rising edge on SDA while SCL is high indicates a stop condition. This stop condition terminates a data transfer to release the bus to idle state. Between a start and a stop condition an arbitrary number of bytes may be transferred.

IIC Frame format

According to the IIC specification, the IIC data has byte format. Every byte put on the bus must be 8-bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first. The data word can be address (after a start symbol) or data (after the address).

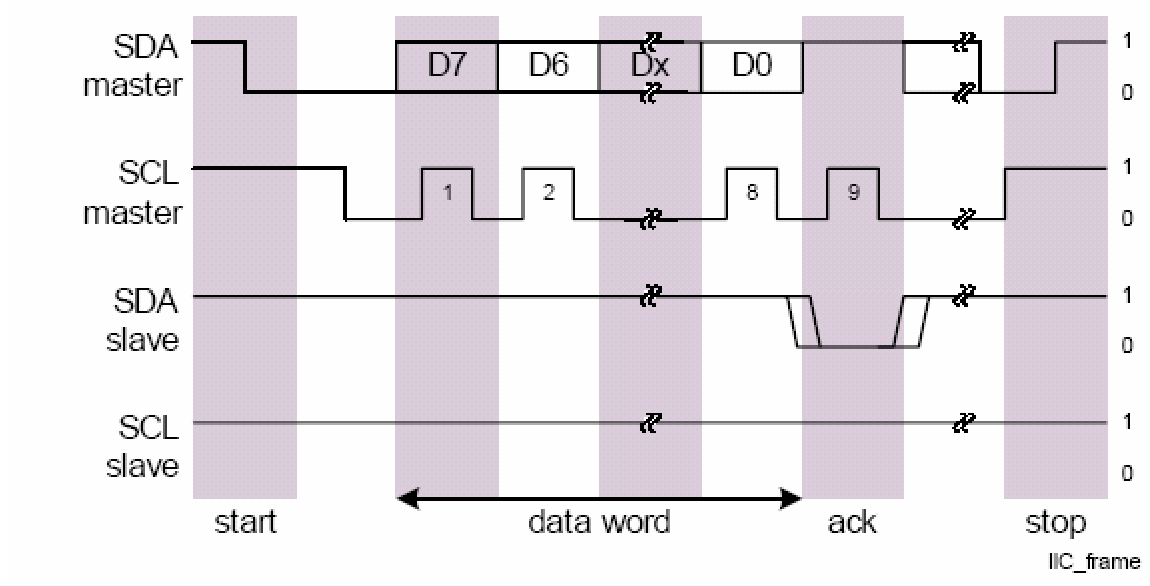

Figure 3 The USIC IIC frame format

3.2 IIC Addressing modes

3.2.1 7 bit address mode

The first byte after the START condition usually determines which slave will be selected by the master. In 7 bit addressing mode a slave address is sent out after the START condition. This address is 7 bits long followed by eighth bit which is a data direction bit (R/W: WRITE - a '0' indicates a transmission, READ - a '1' indicates a request for data).

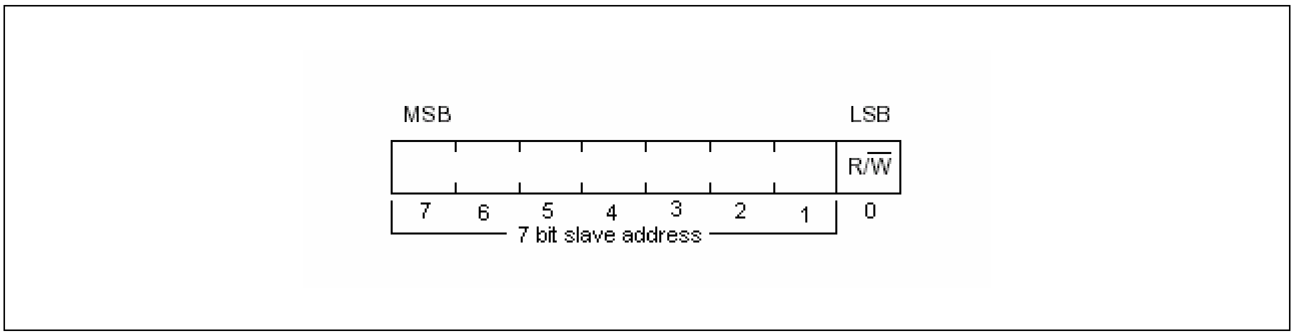

Figure 4 The 7 bit address format

When an address is sent by a master, each device in the system receives and compares the first seven bits with its own address. If the address match, the device becomes a slave (transmitter or receiver depends on the R/W data direction bit).

Master addresses the slave with 7 bit address and slave acknowledges to master.

Master WRITE: Master-transmitter transmits to slave-receiver and data transfer direction is not changed.

Master READ: Master-receiver receives from slave-transmitter immediately after the first byte. The data transfer direction is changed, so the master-transmitter becomes master-receiver and slave-receiver becomes slave-transmitter immediately after the first acknowledge which is generated by slave.

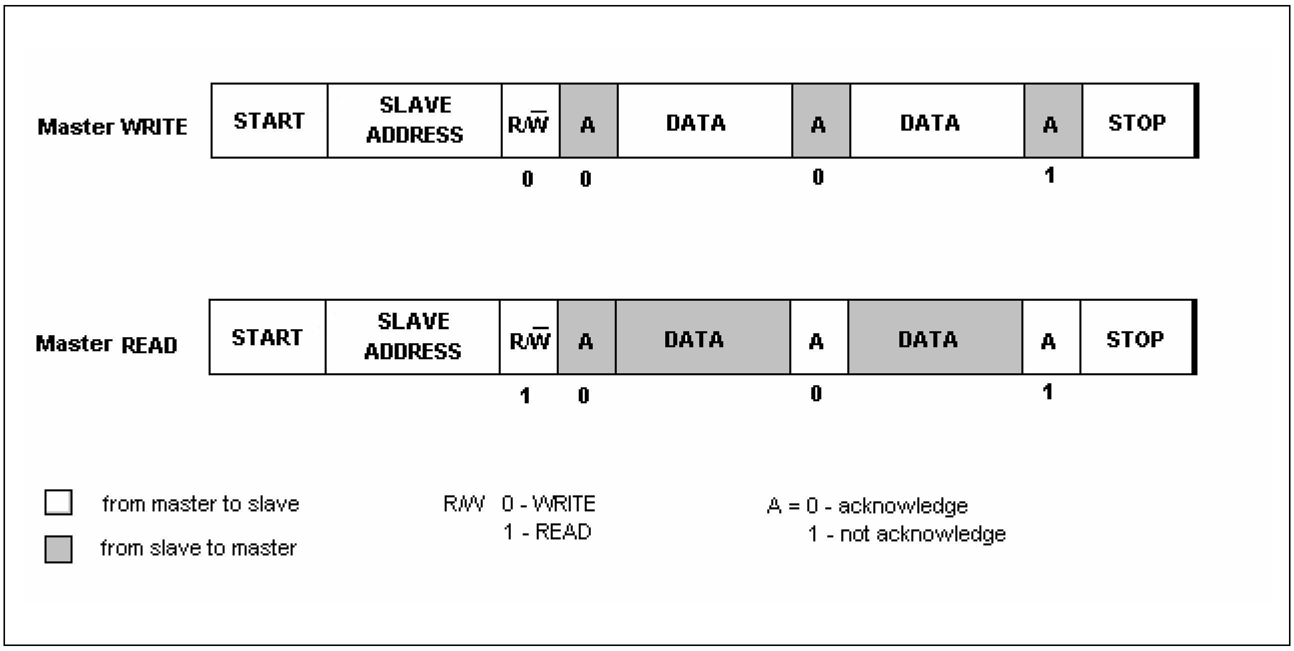

Figure 5 The 7 bit address mode – data transfer

3.2.2 10 bit address mode

The 10-bit slave address is transmitted as the first two bytes after a START condition or repeated START condition. The 10-bit address mode uses the 7-bit address mode's reserved combination 1111xxx for the first seven bits of the first address byte. But it exactly uses only the four combinations 11110xx.

The first seven bits of the first address byte are the combination 11110xx; the last two bits (xx) are the two MSBs of the 10-bit address; the eighth bit of the first byte is the R/W bit that determines the data transfer direction. The second address byte contains the remaining 8 bits of the 10-bit address.

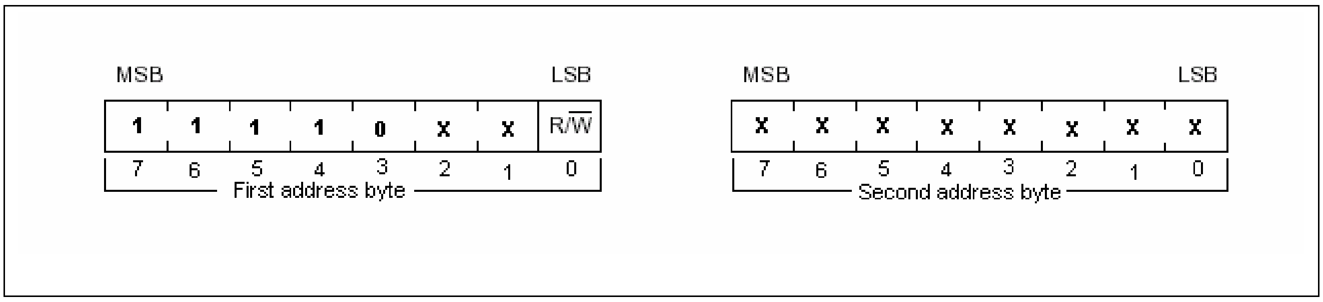

Figure 6 The 10 bit address format

Master WRITE:

- Master-transmitter transmits first address byte on the bus; the each device compares it with their own address and if it matches, acknowledges to master (**A1**).
- Master-transmitter transmits second address byte on the bus, if it matches; the device becomes the slave and acknowledges to master (**A2**).
- Master-transmitter transmits data bytes and gets acknowledges from slave and finally stops the communication. The data transfer direction is not changed.

Master READ:

- The Master READ sequence is also same as WRITE till A2, After A2 the Master transmits the first address byte again with R/W '1' to change the data transfer direction as READ – data from slave to master.
- Slave acknowledges to master (**A3**) and immediately starts to transmit data bytes to master and master acknowledges to slave and finally master stops the communication. Here the data transfer direction is changed immediately after **A3**.

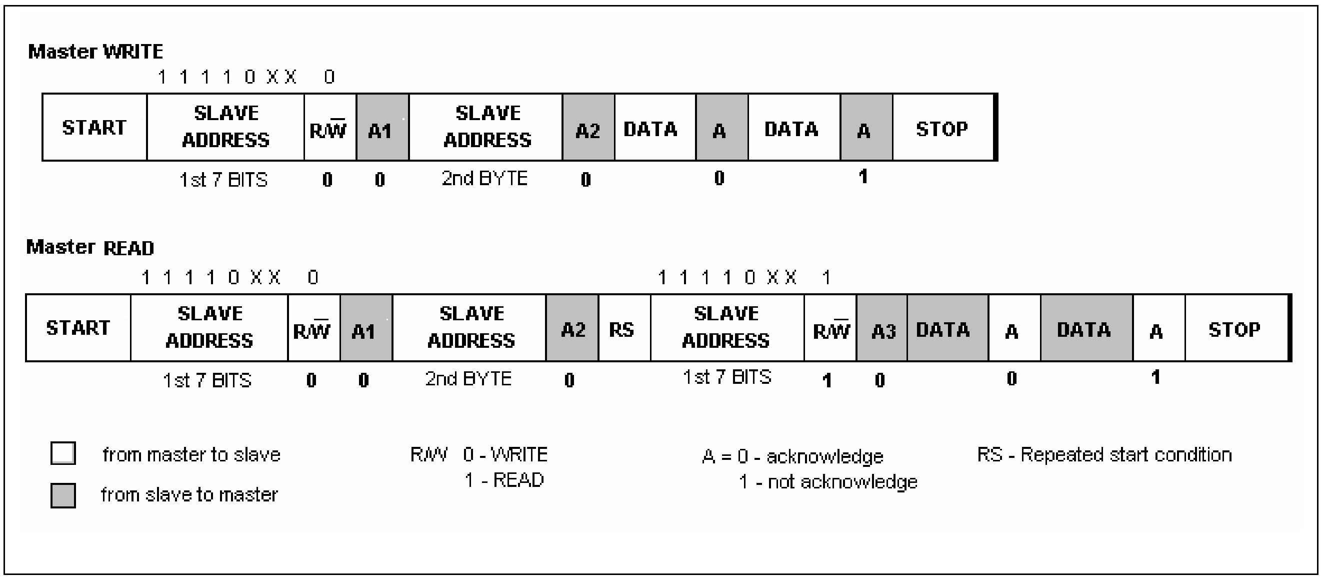

Figure 7 The 10 bit address mode – data transfer

3.2.3 General call address mode

The general call address is used for addressing all the devices on the IIC bus. When this address mode is used, all the devices should respond with an acknowledge. The address byte 00h indicates a general call address, which can be acknowledged if the slave is capable of handling the corresponding requests. The value 01h stands for a start byte generation that is not acknowledged (for details, please refer IIC specification, chapter 10)

The IIC channel of the USIC module can be made to acknowledge or ignore by bit ACK00 in the register PCRH. The address byte 00H is acknowledged if the bit PCRH.ACK00 is set.

3.3 USIC IIC slave address format

7 bit address:

- The 7 bit address should be programmed in the bitfield SLAD [15:9] of PCRL register, MSB at bit CTR15 and LSB at CTR9.
- The first 7 bits of a received first address byte are compared to the programmed slave address (PCRL.SLAD [15:9]). If these bits match, the slave sends an acknowledge.

10 bit address:

- The first two MSBs of the 10 bit address should be programmed in the bits CTR [10] and CTR [9] and remaining 8 bits at CTR [7:0] of PCRL register, the bit CTR8 is ignored.
- If the slave address is programmed to 1111 0XXb, the slave device waits for a second address byte and compares it to PCR.SLAD [7:0] and sends an acknowledge.

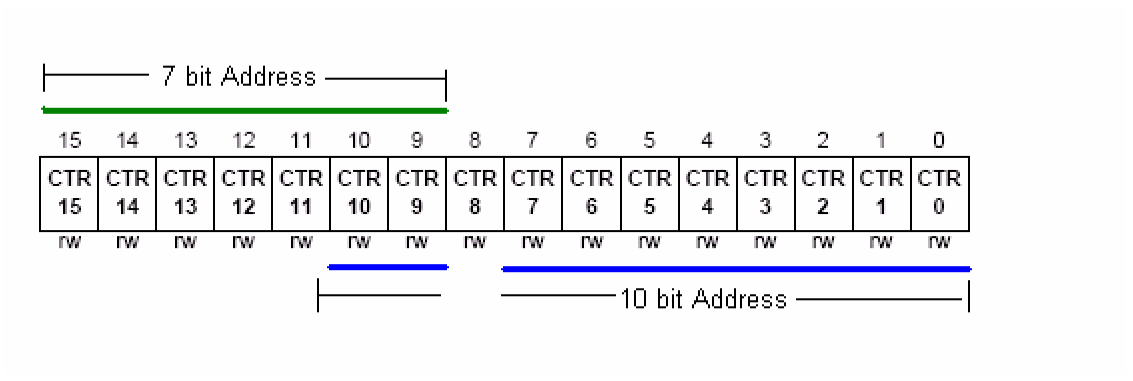

Figure 8 The USIC IIC slave address format

3.4 USIC IIC Protocol Registers

- PCRL – The protocol control register PCRL contrains the programmed slave address SLAD[15:0]. The corresponding bits in the first received address byte are compared to the bits SLAD[15:9] to check for address match (7 bit address mode). if SLAD[15:11] = 11110b (10 bit address mode), then the second address byte is also compared to SLAD[7:0].
- PCRH – The protocol control register PCRH contains the controls for Acknowledge to slave address 00h, Symbol timing for standard and fast mode and IIC protocol specific event interrupts enable control.
- PSR – The protocol status register PSR contains the status flags for the general data transfer events and IIC protocol specific events.
- PSCR – The protocol status clear register PSCR contains the control to clear the status flags in the PSR register.

3.5 USIC IIC Transmit Data Formats

The transmit data formats for master mode:

- TDF = 000b, Master send data byte:
  - The master sends data byte along with data byte (TBUF[7:0]), receives and checks the acknowledge bit sent by the slave.
- TDF = 010b, Master receive data byte and acknowledge 0:
  - The master acknowledges the transfer with a 0-level to continue the transfer. The TBUF[7:0] content is ignored.
- TDF = 011b, Master receive data byte and acknowledge 1:
  - The master acknowledges the transfer with a 1-level to finish the transfer. The TBUF[7:0] content is ignored.
- TDF = 100b, Start condition:
  - The start condition will be generated if the bus is idle, the content of TBUF[7:0] is taken as first address byte for the transmission. (TBUF[7:1] – address and TBUF[0] – read/write control).
- TDF = 101b, Repeated start condition:
  - The repeated start condition will be generated if SCL is low and byte transfer is not in progress. The content of TBUF[7:0] is taken as first address byte for the transmission. (TBUF[7:1] – address and TBUF[0] – read/write control).
- TDF = 110b, Stop condition:
  - The Stop condition will be generated if the master has finished its last byte transfer, the content of TBUF[7:0] is taken as first address byte for the transmission.

The transmit data formats for slave mode:

- TDF = 001b, Slave send data byte to master:
  - The slave only has to send data if it has been asked by the master. The slave sends its data byte (TBUF[7:0]) plus the acknowledge bit as a 1.

3.6 USIC IIC Protocol Interrupt Events

IIC Protocol interrupt events:

- Start condition received interrupt.
- Repeated start condition received interrupt.
- Stop condition received interrupt.
- Non Acknowledge received interrupt.
- Arbitration lost interrupt.
- Slave read request interrupt.
- Error interrupts (wrong TDF, IIC symbols at unexpected position in a frame).

4 IIC Communication

The IIC communication is established by connecting the devices with two bus lines; a serial data line (SDA) and a serial clock line (SCL). The IIC is a multi-master bus, the number of devices connected is limited only by bus capacitance.

A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered as slave. The transmitter sends data to the bus and the receiver gets the data from the bus. Depending on the direction of the data transfer, master and slave can be a transmitter or receiver.

The master initiates the communication. The sequence of events:

1. The Master issues a start condition. This condition informs all the slaves to listen to the serial data line for the instructions.
2. The Master sends the address of the target salve and a read/write flag.
3. The other devices receive the address and compare it with their own address.
   I. If it doesn’t match they simply wait until the bus is released by the stop condition.
   II. If the address matches, the slave responds with an acknowledge signal.
4. The Master and the Salve communication proceeds on the bus. Both the master and the slave can receive or transmit data depending on the data transfer direction (read or write). The transmitter sends 8 bits of data to the receiver which replies with a 1-bit acknowledgement.
5. When the communication is complete, the master issues a stop condition indicating that everything is done.

Let us consider IIC data transfer between our USIC channels, USIC channel U1C1 as master and U0C0 as slave. The different data transfer scenarios of IIC master – salve communication are:

- Master transmit and slave receive with 7 bit address mode.
- Master receive and slave transmit with 7 bit address mode.
- Master transmit and slave receive with 10 bit address mode.
- Master receive and slave transmit with 10 bit address mode.

Software code example

The software code examples explain and provide the configuration for the IIC mode initialization and also test code for IIC Master – Slave communication. The Master mode and Slave mode initialization differs only with slave address assignment to the PCRL register. So the mode initialization can be done with a single function and two macros for Master and Slave mode.

The examples explained in the below sections can run with Infineon XC2000/XE16x Starter Kit. For more information about the Starter Kit, please see **www.infineon.com**

4.1 Master mode configuration

Input stages

- Input stage DX0 is used for shift data input SDA signal (DIN – DX0CR.DSEL)
- Input stage DX1 is used for Shift clock input SCL signal (SCLKIN – DX1CR.DSEL) – optional

Output signals

- Shift data output DOUT is used as SDA output signal
- Shift clock output SCLKOUT is used as SCL output signal

Baud rate generation

The baud rate calculation for IIC (fractional divider mode, CLKSEL = 0, PPPEN = 0 & CTQSEL = 0)

$$f_{IIC} = f_{SYS} * STEP/_{1024} * 1/_{PDIV+1} * 1/_{PCTQ+1} * 1/_{DCTQ+1}$$

- f_{SYS} – The system frequency
- STEP – The step value for the fractional divider mode (FDRL.STEP)
- PDIV – The ratio between the input frequency and the divider frequency (BRGH.PDIV)
- PCTQ – Pre-Divider for time quanta counter (BRGL.PCTQ)
- DCTQ – Denominator for time quanta counter (BRGL.DCTQ)

Shift control

- Transmit/Receive shift MSB first (SCTRL.SDIR = 1b)
- The passive data level is 1 when no data transmission (SCTRL.PDL = 1b)
- Shift data value DOUT without inversion (SCTRL.DOCFG = 0b)
- Data transfer is active and actual signal level is not considered (SCTRL.TRM =11b)
- Data format – 8 data bits (SCTRH.WLE = 7h), unlimited data flow (SCTRH.FLE = 3Fh)

Transfer control

- TBUF data single shot mode – data valid for transfer only once (TCSRL.TDSSM = 1b)
- TBUF data transmission starts if TDV = 1 (TCSRL.TDEN = 1b)

Symbol timing

- A symbol contains 10 time quanta, standard mode – 100 kBaud (PCRH.STIM = 0b)
- A symbol contains 25 time quanta, fast mode – 400 kBaud (PCRH.STIM = 1b)
- The delay to compensate the internal treatment of the SCL signal in order to respect the SDA hold time (PCRH.HDEL)

Pin configuration

- Port pins should be configured to support input, output or both (Pn_IOCRxx_PC)
- The same pin to act as input as well as output ,
  - SDA: SCTRL.DOCFG = 0b, DX0CR.DPOL = 0b and DX0CR.INSW = 0b
  - SCL: BRGH.SCLKCFG = 0b, DX1CR.DPOL = 0b and DX1CR.INSW = 0b

Example code for Master mode configuration

```
void U1C1_IIC_Master_vInit(void)
{

  ///  -----------------------------------------------------------------------
  ///  Channel Control Configuration:
  ///  -----------------------------------------------------------------------
  ///  KSCFG_MODEN = 1, KSCFG_BPMODEN = 1,    - Enable Module
  ///  CCR_MODE    = 0, Channel disabled while initialization

  U1C1_KSCFG     =  0x0003;      // MODEN=1, BPMODEN=1
  U1C1_CCR       =  0x0000;      // MODE=0

  ///  -----------------------------------------------------------------------
  ///  Baudrate Generator Configuration:
  ///  -----------------------------------------------------------------------
  ///  BaudRate is 100.000 kbaud
  ///  FDRL_DM   = 2,    - The Fractional divider mode is selected
  ///  FDRL_STEP = 791   - The step value STEP = 791
  ///  BRGL_PCTQ = 1,    - PreDivider for CTQ
  ///  BRGL_DCTQ = 9,    - The Denominator for CTQ in standard mode
  ///  BRGH_PDIV = 24,   - The Divider factor PDIV = 24

  U1C1_FDRL      =  0x8317;      // DM=2, STEP=0x317
  U1C1_BRGL      =  0x2500;      // PCTQ=1, DCTQ=9
  U1C1_BRGH      =  0x0018;      // PDIV = 24

  ///  -----------------------------------------------------------------------
  ///  Input Control Configuration:
  ///  -----------------------------------------------------------------------
  ///  DX0CR_DSEL = 0,   - The data input DX0A (P0.6) is selected
  ///  DX1CR_DSEL = 0,   - The clock input DX1A (P0.5) is selected

  U1C1_DX0CR     =  0x0000;      // DSEL=0, P0.6 for SDA
  U1C1_DX1CR     =  0x0000;      // DSEL=0, P0.5 for SCL

  ///  -----------------------------------------------------------------------
  ///  Interrupt Node Pointer Configuration:
  ///  -----------------------------------------------------------------------
  ///  Interrupts not used

  U1C1_INPRL     =  0x0000;      // AINP=0, RINP=0, TBINP=0, TSINP=0
  U1C1_INPRH     =  0x0000;      // PINP=0

  ///  -----------------------------------------------------------------------
  ///  Shift Control Configuration:
  ///  -----------------------------------------------------------------------
  ///  SCTRL_SDIR = 1,   - Transmit/Receive shift MSB first
  ///  SCTRL_PDL  = 1,   - The passive data level is 1
  ///  SCTRL_TRM  = 3,   - The shift control signal is active, data frame transfer is possible
  ///  SCTRH_WLE  = 7,   - The word length is 7 (8 data bits)
  ///  SCTRH_FLE  = 3F,  - The frame length is 63 (unlimited data flow)

  U1C1_SCTRL     =  0x0303;      // TRM=3, PDL=1, SDIR=1
  U1C1_SCTRH     =  0x073F;      // WLE=7, FLE=3F

  ///  -----------------------------------------------------------------------
  ///  Transmit Control Configuration:
  ///  -----------------------------------------------------------------------
  ///  TCSRL_TDSSM = 1,   - TBUF data single shot mode: TBUF data transferred only once
  ///  TCSRL_TDEN  = 1,   - TBUF data transmission starts if TDV = 1

  U1C1_TCSRL     =  0x0500;      // TDEN=1, TDSSM=1
  U1C1_TCSRH     =  0x0000;
  ///  -----------------------------------------------------------------------
  ///  Protocol Control Configuration:
  ///  -----------------------------------------------------------------------
  ///  PCRH_STIM  = 0,  - The Symbol Timing, A symbol contains 10 time quanta
  ///  PCRH_HDEL  = 12, - The HW delay to compensate internal treatment of SCL

  U1C1_PCRL      =  0x0000;      //
  U1C1_PCRH      =  0x3000;      // STIM=0, HDEL=12

  ///  -----------------------------------------------------------------------
  ///  Port Pins Configuration:
  ///  -----------------------------------------------------------------------
  ///  IOCR06_PC  = D,  - P0.6 is used for USIC1Channel1 Shift Data output(DOUT)
  ///  IOCR05_PC  = D,  - P0.5 is used for USIC1Channel1 Shift Clock output1(SCLKOUT)

  P0_IOCR06 = 0x00D0;    // PC=D
  P0_IOCR05 = 0x00D0;    // PC=D

  ///  -----------------------------------------------------------------------
  ///  Mode & Interrupts Configuration:
  ///  -----------------------------------------------------------------------
  ///  CCR_xxIEN  = 0,  - Interrupts are not enabled
  ///  CCR_PM     = 0,  - The parity generation is disabled
  ///  CCR_MODE   = 4,  - The IIC protocol is selected

  U1C1_CCR       =  0x0004;      // PM=0, MODE=4

}
```

4.2 Slave mode configuration

Input stages

- Input stage DX0 is used for shift data input SDA signal (DIN – DX0CR.DSEL)
- Input stage DX1 is used for shift clock input SCL signal (SCLKIN – DX1CR.DSEL)

Output signals

- Shift data output DOUT is used as SDA output signal
- Shift clock output SCLKOUT is used as SCL output signal

Baud rate generation

The baud rate calculation for IIC (fractional divider mode, CLKSEL = 0, PPPEN = 0 & CTQSEL = 0)

$$f_{IIC} = f_{SYS} * STEP/_{1024} * 1/_{PDIV+1} * 1/_{PCTQ+1} * 1/_{DCTQ+1}$$

- f_{SYS} – The system frequency
- STEP – The step value for the fractional divider mode (FDRL.STEP)
- PDIV – The ratio between the input frequency and the divider frequency (BRGH.PDIV)
- PCTQ – Pre-Divider for time quanta counter (BRGL.PCTQ)
- DCTQ – Denominator for time quanta counter (BRGL.DCTQ)

Shift control

- Transmit/Receive shift MSB first (SCTRL.SDIR = 1b)
- The passive data level is 1 when no data transmission (SCTRL.PDL = 1b)
- Shift data value DOUT without inversion (SCTRL.DOCFG = 0b)

- Data transfer is active and actual signal level is not considered (SCTRL.TRM =11b)
- Data format – 8 data bits (SCTRH.WLE = 7h), unlimited data flow (SCTRH.FLE = 3Fh)

Transfer control

- TBUF data single shot mode – data valid for transfer only once (TCSRL.TDSSM = 1b)
- TBUF data transmission starts if TDV = 1 (TCSRL.TDEN = 1b)

Symbol timing

- A symbol contains 10 time quanta, standard mode – 100 kBaud (PCRH.STIM = 0b)
- A symbol contains 25 time quanta, fast mode – 400 kBaud (PCRH.STIM = 1b)
- The delay to compensate the internal treatment of the SCL signal in order to respect the SDA hold time (PCRH.HDEL)

Pin configuration

- Port pins should be configured to support input, output or both (Pn_IOCRxx_PC)
- The same pin to act as input as well as output ,
  - SDA: SCTRL.DOCFG = 0b, DX0CR.DPOL = 0b and DX0CR.INSW = 0b
  - SCL: BRGH.SCLKCFG = 0b, DX1CR.DPOL = 0b and DX1CR.INSW = 0b

Slave address format

- 7 bit slave address should be programmed in PCRL.SLAD[15:9]
- 10 bit slave address first two MSBs in PCRL.SLAD [10:9] and remaining bits in PCRL.SLAD [7:0] and PCRL.SLAD [15:11] = 11110b is programmed as per the IIC standard.

Example code for Slave mode configuration

```
void U0C0_IIC_Slave_vInit(void)
{

  ///  -----------------------------------------------------------------------
  ///  Channel Control Configuration:
  ///  -----------------------------------------------------------------------
  ///  KSCFG_MODEN = 1, KSCFG_BPMODEN = 1,   - Enable Module
  ///  CCR_MODE    = 0, Channel disabled while initialization

  U0C0_KSCFG     =  0x0003;      // MODEN=1, BPMODEN=1
  U0C0_CCR       =  0x0000;      // MODE=0

  ///  -----------------------------------------------------------------------
  ///  Baudrate Generator Configuration:
  ///  -----------------------------------------------------------------------
  ///  BaudRate is 100.000 kbaud
  ///  FDRL_DM   = 2,   - The Fractional divider mode is selected
  ///  FDRL_STEP = 791  - The step value STEP = 791
  ///  BRGL_PCTQ = 1,   - PreDivider for CTQ
  ///  BRGL_DCTQ = 9,   - The Denominator for CTQ in standard mode
  ///  BRGH_PDIV = 24,  - The Divider factor PDIV = 24

  U0C0_FDRL      =  0x8317;      // DM=2, STEP=0x317
  U0C0_BRGL      =  0x2500;      // PCTQ=1, DCTQ=9
  U0C0_BRGH      =  0x0018;      // PDIV = 24
  /// ----------------------------------------------------------------------
  /// Input Control Configuration:
  /// ----------------------------------------------------------------------
  /// DX0CR_DSEL = 1,  - The data input DX0A (P10.1) is selected
  /// DX1CR_DSEL = 1,  - The clock input DX1A (P10.2) is selected

  U0C0_DX0CR     =  0x0001;      // DSEL=1, P10.1 for SDA
  U0C0_DX1CR     =  0x0001;      // DSEL=1, P10.2 for SCL

  /// ----------------------------------------------------------------------
  /// Interrupt Node Pointer Configuration:
  /// ----------------------------------------------------------------------
  /// Interrupts not used

  U0C0_INPRL     =  0x0000;      // AINP=0, RINP=0, TBINP=0, TSINP=0
  U0C0_INPRH     =  0x0000;      // PINP=0

  /// ----------------------------------------------------------------------
  /// Shift Control Configuration:
  /// ----------------------------------------------------------------------
  /// SCTRL_SDIR = 1,  - Transmit/Receive shift MSB first
  /// SCTRL_PDL  = 1,  - The passive data level is 1
  /// SCTRL_TRM  = 3,  - The shift control signal is active, data frame transfer is possible
  /// SCTRH_WLE  = 7,  - The word length is 7 (8 data bits)
  /// SCTRH_FLE  = 3F, - The frame length is 63 (unlimited data flow)

  U0C0_SCTRL     =  0x0303;      // TRM=3, PDL=1, SDIR=1
  U0C0_SCTRH     =  0x073F;      // WLE=7, FLE=3F

  /// ----------------------------------------------------------------------
  /// Transmit Control Configuration:
  /// ----------------------------------------------------------------------
  /// TCSRL_TDSSM = 1,  - TBUF data single shot mode: TBUF data transferred only once
  /// TCSRL_TDEN  = 1,  - TBUF data transmission starts if TDV = 1

  U0C0_TCSRL     =  0x0500;      // TDEN=1, TDSSM=1
  U0C0_TCSRH     =  0x0000;

  /// ----------------------------------------------------------------------
  /// Protocol Control Configuration:
  /// ----------------------------------------------------------------------
  /// PCRH_STIM  = 0,  - The Symbol Timing, A symbol contains 10 time quanta
  /// PCRH_HDEL  = 12, - The HW delay to compensate internal treatment of SCL

  U0C0_PCRL      =  0x0A00;      // SLAD[15:9] = 0x0A, 7 bit address
  U0C0_PCRH      =  0x3000;      // STIM=0, HDEL=12

  /// ----------------------------------------------------------------------
  /// Port Pins Configuration:
  /// ----------------------------------------------------------------------
  /// IOCR01_PC  = D,  - P0.6 is used for USIC1Channel1 Shift Data output(DOUT)
  /// IOCR02_PC  = D,  - P0.5 is used for USIC1Channel1 Shift Clock output1(SCLKOUT)

  P10_IOCR01 = 0x00D0;    // PC=D
  P10_IOCR02 = 0x00D0;    // PC=D

  /// ----------------------------------------------------------------------
  /// Mode & Interrupts Configuration:
  /// ----------------------------------------------------------------------
  /// CCR_xxIEN  = 0,  - Interrupts are not enabled
  /// CCR_PM     = 0,  - The parity generation is disabled
  /// CCR_MODE   = 4,  - The IIC protocol is selected

  U0C0_CCR       =  0x0004;      // PM=0, MODE=4
}
```

4.3 Master Transmit and Slave Receive – 7 bit address mode

The sequence of events:

- START: The Master U1C1 initiates the communication by a start condition.
  - Transfer Data Format TDF_MStart to generate a start condition on the bus.
  - IICSlaveAddr is slave address that master wants to communicate.
  - IICWRITE is data transfer type which indicates that master wants to send data.
  - Function vSendData() to write the whole data (TDF, Address and transfer type) to TBUF.
  - Slave acknowledges to master if its address match and sets PSR.SLSEL flag
- TRANSMIT: The Master U1C1 transmits the data to slave which is acknowledged for the address.
  - Transfer Data Format TDF_MTxData to transmit data to slave.
  - IICMasterSendData[] buffer to transmit an array of data to slave.
  - Function vSendDate() to write the whole data (TDF and Data) to TBUF.
  - Slave receives the data and stores it in IICSlaveReceiveData[] buffer with uwGetData() function.
- STOP: The Master issues a stop condition to finish the communication.
  - Transfer Data Format TDF_MStop to terminate the communication.
  - Function vSendData() to write the whole data (TDF, Dummy data) to TBUF.

Example code

```
//****************************************************************************
// USIC IIC:  U1C1_Master Transmit - U0C0_Slave Receive, 7 bit address mode
//****************************************************************************
void IIC_MTSR_7bit_mode(void)
{
    U1C1_IIC_vSendData(U1C1TDF_MStart, uwIICSlaveAddr + U1C1IIC_WRITE);

    While(!(U0C0_PSR & 0x0001));    // PSR_SLSEL - wait for slave select

    for(uwIICCount=0; uwIICCount < IIC_DATA_MAX ; uwIICCount++)
    {
      U1C1_IIC_vSendData(U1C1TDF_MTxData, uwIICMasterSendData[uwIICCount]);

      uwIICSlaveReceiveData[uwIICCount] = (ubyte) U0C0_IIC_uwGetData();
    }
     U1C1_IIC_vSendData(U1C1TDF_MStop, uwFFFFU1C1);
}
```

4.4 Master Receive and Slave Transmit – 7 bit address mode

The sequence of events:

- START: The Master U1C1 initiates the communication by a start condition.
  - Transfer Data Format TDF_MStart to generate a start condition on the bus.
  - IICSlaveAddr is slave address that master wants to communicate.
  - IICREAD is data transfer type which indicates that master wants to read from slave.
  - Function vSendData() to write the whole data (TDF, Address and transfer type) to TBUF.
  - Slave acknowledges to master if its address match and sets PSR.SLSEL flag
- RECEIVE: The Master U1C1 receives the data from slave which is acknowledged for the address.
  - Slave uses TDF_STxData to transmit data to master.
  - IICSlaveSendData[] buffer to transmit an array of data to master.
  - Function vSendDate() to write the whole data (TDF and Data) to TBUF.
  - Master receives the data and store it in IICSlaveReceiveData[] buffer with uwGetData() function.
  - Master acknowledges to slave by TDF MRxAck0 and dummy data. if the last data byte, Master acknowledges by TDF MRxAck1 to finish the communication.
- STOP: The Master issues a stop condition to finish the communication.
  - Transfer Data Format TDF_MStop to terminate the communication.
  - Function vSendData() to write the whole data (TDF, Dummy data) to TBUF.

Example code

```
//****************************************************************************
// USIC IIC:  U1C1_Master Reveice - U0C0_Slave Transmit, 7 bit address mode
//****************************************************************************
void IIC_MRST_7bit_mode (void)
{
    U1C1_IIC_vSendData(U1C1TDF_MStart, uwIICSlaveAddr + U1C1IIC_READ);

    While(!(U0C0_PSR & 0x0001));    // PSR_SLSEL - wait for slave select
    While(!(U0C0_PSR & 0x0080));    // PSR_SRR - wait for slave read request

    for (uwIICCount=0; uwIICCount < IIC_DATA_MAX ; uwIICCount++)
    {                                                    // Slave Tx  Master READ
       U0C0_IIC_ vSendData(U0C0TDF_STxData, uwIICSlaveSendData[uwIICCount]);

       if (uwIICCount == IIC_DATA_MAX - 1)                // Master ACK
         U1C1_IIC_vSendData(U1C1TDF_MRxAck1, uwFFFFU1C1);
       else
         U1C1_IIC_vSendData(U1C1TDF_MRxAck0, uwFFFFU1C1);

       uwIICMasterReceiveData[uwIICCount] = (ubyte) U1C1_IIC_ uwGetData();
    }
    U1C1_IIC_vSendData(U1C1TDF_MStop, uwFFFFU1C1);
}
```

4.5 Master Transmit and Slave Receive – 10 bit address mode

The sequence of events:

- START: The Master U1C1 initiates the communication by a start condition.
  - Transfer Data Format TDF_MStart to generate a start condition on the bus.
  - IICSubAddr is first address byte of slave that master wants to communicate.
  - IICWRITE is data transfer type which indicates that master wants to send data.
  - Function vSendData() to write the whole data (TDF, Address and transfer type) to TBUF.
- ADDRESS: The Master U1C1 transmits the second address byte of the 10 bit address.
  - IICSlaveAddr is second address byte of slave that master wants to communicate.
  - Slave acknowledges to master if its address match and sets PSR.SLSEL flag
- TRANSMIT: The Master U1C1 transmits the data to slave which is acknowledged for the address.
  - Transfer Data Format TDF_MTxData to transmit data to slave.
  - IICMasterSendData[] buffer to transmit an array of data to slave.
  - Function vSendDate() to write the whole data (TDF and Data) to TBUF.
  - Slave receives the data and stores it in IICSlaveReceiveData[] buffer with uwGetData() function.
- STOP: The Master issues a stop condition to finish the communication.
  - Transfer Data Format TDF_MStop to terminate the communication.
  - Function vSendData() to write the whole data (TDF, Dummy data) to TBUF.

Example code

```
//*****************************************************************************
// USIC IIC:  U1C1_Master Transmit – U0C0_Slave Receive, 10 bit address mode
//*****************************************************************************
void IIC_MTSR_10bit_mode(void)
{
    U0C1_IIC_vSendData(U0C1TDF_MStart, uwIICSubAddr + U0C1IIC_WRITE);

     U0C1_IIC_vSendData(U0C1TDF_MTxData, uwIICSlaveAddr);

     While(!(U0C0_PSR & 0x0001));   // PSR_SLSEL – wait for slave select

    for (uwIICCount=0; uwIICCount < IIC_DATA_MAX ; uwIICCount++)
    {
       U0C1_IIC_vSendData(U0C1TDF_MTxData, uwIICMasterSendData[uwIICCount]);

      uwIICSlaveReceiveData[uwIICCount] = (ubyte) U2C1_IIC_uwGetData();
    }
     U0C1_IIC_vSendData(U0C1TDF_MStop, uwFFFFU0C1);
}
```

4.6 Master Receive and Slave Transmit – 10 bit address mode

The sequence of events:

- START: The Master U1C1 initiates the communication by a start condition.
  - Transfer Data Format TDF_MStart to generate a start condition on the bus.
  - IICSubAddr is first address byte of slave that master wants to communicate.
  - IICWRITE is data transfer type which indicates that master wants to send data.
  - Function vSendData() to write the whole data (TDF, Address and transfer type) to TBUF.
- ADDRESS: The Master U1C1 transmits the second address byte of the 10 bit address.
  - IICSlaveAddr is second address byte of slave that master wants to communicate.
  - Slave acknowledges to master if its address match and sets PSR.SLSEL flag
- REPEATED START: The Master U1C1 retransmits the first address byte.
  - TDF_MRStart to generate a repeated start condition on the bus.
  - IICREAD is data transfer type whether master wants to read from slave.
- RECEIVE: The Master U1C1 receives the data from slave which is acknowledged for the address.
  - Slave uses TDF_STxData to transmit data to master.
  - IICSlaveSendData[] buffer to transmit an array of data to master.
  - Function vSendDate() to write the whole data (TDF and Data) to TBUF.
  - Master receives the data and store it in IICSlaveReceiveData[] buffer with uwGetData() function.
  - Master acknowledges to slave by TDF MRxAck0 and dummy data. if the last data byte, Master acknowledges by TDF MRxAck1 to finish the communication.
- STOP: The Master issues a stop condition to finish the communication.
  - Transfer Data Format TDF_MStop to terminate the communication.
  - Function vSendData() to write the whole data (TDF, Dummy data) to TBUF.

Example code

```
//****************************************************************************
// USIC IIC:  U1C1_Master Reveice - U0C0_Slave Transmit, 10 bit address mode
//****************************************************************************
void IIC_MRST_10bit_mode (void)
{
     U1C1_IIC_vSendData(U1C1TDF_MStart, uwIICSubAddr + U1C1IIC_WRITE);
     U1C1_IIC_vSendData(U1C1TDF_MTxData, uwIICSlaveAddr);
     U1C1_IIC_vSendData(U1C1TDF_MRStart, uwIICSubAddr + U1C1IIC_READ);
     U1C1_IIC_vSendData(U1C1TDF_MRxAck0, uwFFFFU1C1);

     While(!(U0C0_PSR & 0x0001));   // PSR_SLSEL - wait for slave select
     While(!(U0C0_PSR & 0x0080));   // PSR_SRR - wait for slave read request

    for (uwIICCount=0; uwIICCount < IIC_DATA_MAX ; uwIICCount++)
    {                                                     // Slave Tx Master READ
      U0C0_IIC_ vSendData(U0C0TDF_STxData, uwIICSlaveSendData[uwIICCount]);

      if (uwIICCount == IIC_DATA_MAX - 1)                 // Master ACK
        U1C1_IIC_vSendData(U1C1TDF_MRxAck1, uwFFFFU1C1);
      else
        U1C1_IIC_vSendData(U1C1TDF_MRxAck0, uwFFFFU1C1);

      uwIICMasterReceiveData[uwIICCount] = (ubyte) U1C1_IIC_uwGetData();
    }
    U1C1_IIC_vSendData(U1C1TDF_MStop, uwFFFFU1C1);
}
```

Transmit data formats

```
#define U1C1TDF_MTxData      0x00     // Master transmits data
#define U1C1TDF_STxData      0x01     // Slave transmits data
#define U1C1TDF_MRxAck0      0x02     // Master acknowledge with 0
#define U1C1TDF_MRxAck1      0x03     // Master acknowledge with 1
#define U1C1TDF_MStart       0x04     // Start condition by master
#define U1C1TDF_MRStart      0x05     // Repeated start condition by master
#define U1C1TDF_MStop        0x06     // Stop condition by master
```

Data transfer type

```
#define U1C1IIC_WRITE        0          // Master write to slave
#define U1C1IIC_READ         1          // Master reads from slave
#define uwFFFFU1C1         0xFFFF       // dummy data
```

Data transfer buffers

```
#define IIC_DATA_MAX          9          // data buffer size
uword uwIICCount;                        // counter
uword uwIICMasterReceiveData[IIC_DATA_MAX];      // Master receive data buffer
uword uwIICSlaveReceiveData[IIC_DATA_MAX];       // Slave receive data buffer
uword uwIICMasterSendData[9] = {0x90,0x80,0x70,0x60,0x50,0x40,0x30,0x20,0x10};
uword uwIICSlaveSendData[9] = {0x15,0x25,0x35,0x45,0x55,0x65,0x75,0x85,0x95};
```

Slave address

```
uword uwIICSlaveAddr=0x0A;               // Slave address 7 bit format
uword uwSubAddr = 0xF2;                  // Slave address 10 bit  format - first byte
```

Send data function

```
void UxCy_IIC_vSendData(uword uwTDF, uword uwData)
{
  while(UxCy_TCSRL & 0x0080);
  UxCy_PSCR    |= 0x2000;                                       // clear PSR_TBIF
  UxCy_TBUF00  = ((uwTDF << 8) & 0x0700) | (uwData & 0x00FF);  // load TBUF00
}
```

Get data function

```
uword UxCy_IIC_uwReadData(void)
{
  while(!((UxCy_PSR & 0x8000) || (UxCy_PSR & 0x4000)));
  UxCy_PSCR    |= 0xC000;          // clear PSR_AIF & PSR_RIF
  return(UxCy_RBUF);               // return receive buffer register
}
```