LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information
For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (www.infineon.com).

Warnings
Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.
We Listen to Your Comments

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

ipdoc@infineon.com
# Table of Contents

1. Introduction ................................................................................................................... 5

2. Introduction to the IIC (I2C) Bus .................................................................................... 6

3. Digital Temperature Sensor (LM75A) ............................................................................... 7

4. Interfacing the XC22xx Device with LM75A ................................................................. 8

5. IIC Configuration using DAvE ......................................................................................... 10

6. Example C Code for the XC2265N Device .................................................................... 16

7. Abbreviations .................................................................................................................. 19

8. Related Documents ....................................................................................................... 19
1 Introduction

This application note focuses on I2C (IIC) communication with a Temperature Sensor (LM75A in this example), driven by a Microcontroller.

The IIC functionality is described, along with an example on how to set up the IIC using the Infineon DAvE tool with an XC2265N device.

Figure 1  Block Diagram of XC22xx Starter Kit with Temperature Sensor
2 Introduction to the IIC (I2C) Bus

The IIC bus is a bi-directional two line bus, enabling communication between any kind of integrated circuit that supports this protocol, either by hardware or software. Examples of such ICs are EEPROMs, RAMs, and Temperature Sensors, data converters or general purpose microcontrollers.

The main advantage of this protocol is its two line interface, as shown in figure 2.

![Diagram of IIC Communication](image)

**Figure 2** IIC Communication

The two-line bus consists of an SDA (Serial Data Line) and an SCL (Serial Clock Line). These lines are to be connected to a positive supply via pull-up resistors (Note: Please read the IIC bus specification).

The multi-master capability of the IIC bus is not used in this application example. There is only one master, the Infineon XC22xx microcontroller, and a slave (the Temperature Sensor).
3 Digital Temperature Sensor (LM75A)

The LM75A is an industry-standard digital temperature sensor with an integrated Sigma-Delta analog-to-digital converter and I2C® interface.

The LM75A provides 9-bit digital temperature readings with an accuracy of ±2°C, from -25°C to 100°C and ±3°C over -55°C to 125°C.

<table>
<thead>
<tr>
<th>Label</th>
<th>Pin #</th>
<th>Function</th>
<th>Typical Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDA</td>
<td>1</td>
<td>I2C Serial Bi-Directional Data Line, Open Drain</td>
<td>From Controller, tied to a pull-up resistor or current source</td>
</tr>
<tr>
<td>SCL</td>
<td>2</td>
<td>I2C Clock Input</td>
<td>From Controller, tied to a pull-up resistor or current source</td>
</tr>
<tr>
<td>O.S.</td>
<td>3</td>
<td>Over temperature Shutdown, Open Drain Output</td>
<td>Pull-up Resistor, Controller Interrupt Line</td>
</tr>
<tr>
<td>GND</td>
<td>4</td>
<td>Power Supply Ground</td>
<td>Ground</td>
</tr>
<tr>
<td>+Vs</td>
<td>8</td>
<td>Positive Supply Voltage Input</td>
<td>DC Voltage from 2.7V to 5.5V 100nF bypass capacitor with 10uF bulk capacitance in the near vicinity</td>
</tr>
<tr>
<td>A0-A2</td>
<td>7, 6, 5</td>
<td>User-Set I2C Address Input</td>
<td>Ground (Low, “0”) or +Vs (High, “1”)</td>
</tr>
</tbody>
</table>

![LM75A Pin Description Diagram](image)
4 Interfacing the XC22xx Device with LM75A

4.1 LM75A

The LM75A sensor operates with a single supply from +2.7V to +5.5V. Communication is via a 2-wire interface which operates up to 400 kHz. The sensor has three address pins (User-Set I2C Address Inputs), allowing up to eight LM75A devices to operate on the same 2-wire bus.

The LM75A temperature sensor incorporates a band-gap type temperature sensor and 9-bit ADC (Sigma-Delta Analog to-Digital Converter). The temperature data output of the LM75A is available at all times via the I2C bus.

4.1.1 IIC Bus Interface

The LM75A operates as a slave on the I2C bus, so the SCL line is an input (no clock is generated by the LM75A) and the SDA line is a bi-directional serial data path.

According to the I2C bus specification, the LM75A has a 7-bit slave address. The four most significant bits of the slave address are hard wired inside the LM75A and are "1001". The three least significant bits of the address are assigned to pins A2–A0, and are set by connecting these pins to ground for a low, (0); or to +Vs for a high, (1).

The complete slave address is:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1.2 Temperature Data Format

Temperature data is represented by a 9-bit, two's complement word with an LSB (Least Significant Bit) equal to 0.5°C:

Table 2 Temperature Data Format

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Digital Output</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Binary</td>
<td></td>
</tr>
<tr>
<td>+125°C</td>
<td>0 1111 1010</td>
<td>0FAh</td>
</tr>
<tr>
<td>+25°C</td>
<td>0 0011 0010</td>
<td>032h</td>
</tr>
<tr>
<td>+0.5°C</td>
<td>0 0000 0001</td>
<td>001h</td>
</tr>
<tr>
<td>0°C</td>
<td>0 0000 0000</td>
<td>000h</td>
</tr>
<tr>
<td>-0.5°C</td>
<td>1 1111 1111</td>
<td>1FFh</td>
</tr>
<tr>
<td>-25°C</td>
<td>1 1100 1110</td>
<td>1CEh</td>
</tr>
<tr>
<td>-55°C</td>
<td>1 1001 0010</td>
<td>192h</td>
</tr>
</tbody>
</table>
4.2 XC22xx Configuration

- XC22xx is configured in Master Mode with 400kbaud.
- Pin P0.6 is used as SDA pin.
- Pin P0.5 is used as SCL pin.
- USIC1 CH1 is used for I2C protocol.

![Interfacing the XC22xx Device with LM75A](image.png)

Figure 4   Interfacing the XC22xx Device with LM75A
5 IIC Configuration using DAvE

Note: Although USIC1 CH 1 is used as IIC Protocol in this example, other USIC channels can also be used. The following steps describe the configuration.

Select USIC1 Bubble from XC22xx GUI

![Diagram showing USIC1 Module](image-url)

Figure 5 DAvE Showing USIC1 Module
Configuration of USIC1

1. Select IIC from USIC1 Channel 1 Protocol Selection

Figure 6  IIC Protocol Selection from USIC1 Channel 1
Select CH 1 bubble (Part of USIC1)
Configuration of IIC General Tab
1. Select the 'Enable module; the module is supplied with the input clock' checkbox.
2. Select SCL pin (P0.5) from Pin Selection.
3. Select SDA pin (P0.6) from Pin Selection.
4. Configure Baud rate to 400 kbaud

Figure 8 U1C1 IIC General Page
Configuration of FIFO

1. Set 'No. of FIFO Buffer entries' to 4.
2. Set 'Tx FIFO Data Pointer' to 0.
3. Set 'No. of FIFO Buffer entries' to 2
4. Set 'Rx FIFO Data Pointer' to 4

Figure 9  U1C1 IIC FIFO Page
Configuration of Functions Tab

1. Select U1C1_IIC_vInit.
2. Select U1C1_IIC_uwGetStatus.
3. Select U1C1_IIC_vFillTxFIFO.
4. Select U1C1_IIC_uwGetRxFIFOData.
5. Select U1C1_IIC_ublsRxFIFObusy.
6. Select U1C1_IIC_ublsTxFIFObusy.
7. Select U1C1_IIC_ublsRxFIFOempty.
8. Select U1C1_IIC_ublsTxFIFOempty.
9. Select U1C1_IIC_ubGetRxFIFOFillingLevel.
10. Select U1C1_IIC_vFlushRxFIFO.
11. Select U1C1_IIC_vFlushTxFIFO.

Figure 10  U1C1 Function Page
6 Example C Code for the XC2265N Device

The following example C code, written for the XC2265N device, shows how to read the temperature from the temperature sensor with IIC as a protocol using USIC. Only temperature data is read in this example, using FIFO.

Tools Used
- DAve v2.1r22
- DAve_XC22xxN_Series_v0.2.dip (Latest Dip)
- EASY KIT XC22xxX / XE166 V3.1
- TASKING Tools for C166-VX v2.4r1
- Universal Debug Engine, Release: 2.04.19 (Universal access device 2)

Example Code

```c
//****************************************************************************
// @Function      void Test_vApplication(void) 
// @Description   This function Reads Temperature
//     NOTE: This function should be called in MAIN.C
// @Returnvalue   None
// @Parameters:   None
//****************************************************************************
uword uwFIFO_SendData[4];
uword uwFIFO_RecData[2] = {0};

void Test_vApplication(void)
{
    ubyte ubReadData = 0;
    ubyte ubFIFOLevel = 2;
    uwword uwSlAdr  = 0x90; // Address A0 - A2 = 0

    Doprintf("\r\n+--------------------------------------------------------------------------+");
    Doprintf("\r\nTemperature Sensor");
    Doprintf("\r\nUSIC U0C0 - ASC Communication for UART");
    Doprintf("\r\nBaud rate - 19,200 kbaud, Used pins for UART: Txd -> P7.3, Rxd -> P7.4");
    Doprintf("\r\nUSIC U1C1 - IIC Communication for Temperature Sensor");
    Doprintf("\r\nBaud rate - 400 kbaud, Used pins for IIC: SDA -> P0.6, SCL -> P0.5");
    Doprintf("\r\n+--------------------------------------------------------------------------");

    uwFIFO_SendData[0]= (((U1C1TDF_MStart << 8) & 0x0700) | ((uwSlAdr + U1C1IIC_READ) & 0x00FF)); // Read mode
    uwFIFO_SendData[1]= (((U1C1TDF_MRxAck0 << 8) & 0x0700)); // TDF_Ack0
    uwFIFO_SendData[2]= (((U1C1TDF_MRxAck1 << 8) & 0x0700)); // TDF_Ack1
```
uwFIFO_SendData[3]= ((U1C1TDF_MStop << 8) & 0x0700); // TDF_Stop
while(!U1C1_IIC_ubIsTxFIFOempty()); // Wait for Tx FIFO Empty
Dprintf("\r\nTxFlush Empty");
while(U1C1_IIC_ubIsTxFIFObusy()); // Wait for Tx FIFO Not Busy
Dprintf("\r\nTx FIFO Not Busy");
U1C1_IIC_vFlushTxFIFO(); // clear the transmit FIFO
Dprintf("\r\nClear the transmit FIFO");
U1C1_IIC_vFillTxFIFO(uwFIFO_SendData , 4);
sprintf(s, "\r\nProtocol Status[%x", U1C1_IIC_uwGetStatus());
Dprintf(s);
while (U1C1_IIC_ubGetRxFIFOFillingLevel() != 2); // Wait for Two Data's
while (ubReadData != ubFIFO0Level) // Read Two Data's
  {
    while (U1C1_IIC_ubIsRxFIFObusy()); // Wait for RxFIFO Not busy
    uwFIFO_RecData[ubReadData] = U1C1_IIC_uwGetRxFIFOData(); // Get the Received Data
    Dprintf("\r\nRec Data");
    ubReadData++;
  }
U1C1_IIC_vFlushRxFIFO();
while(!U1C1_IIC_ubIsRxFIFOempty()); // Wait for Tx FIFO Empty
Dprintf("\r\nRxFlush Empty");
sprintf(s, "\r\nRecData0[%x", (uwFIFO_RecData[0] & 0x00FF)); // Read 1st Byte
Dprintf(s);
sprintf(s, "\r\nRecData1[%x", ((uwFIFO_RecData[1] & 0x0080) >> 7)); // Read 2nd Byte (1-LSB bit)
Dprintf(s);
Dprintf("\r\n+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Communication END +-+-+-+-+-+-+-+-+-+-+");
// end of application}
6.1 Results

The Test_vApplication result, captured through Serial Port (HyperTerminal), is given below.

The received value can be converted to temperature by using the Temperature Data Format (see section 4.1.2).

![Figure 11 Result Captured through Serial Port](image-url)
7 Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td>First In, First Out</td>
</tr>
<tr>
<td>I2C</td>
<td>IIC - Inter-IIC Bus Protocol</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Bit</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Bit</td>
</tr>
<tr>
<td>SDA</td>
<td>Serial data line</td>
</tr>
<tr>
<td>SCL</td>
<td>Serial clock line</td>
</tr>
<tr>
<td>USIC</td>
<td>Universal Serial Interface Channel</td>
</tr>
</tbody>
</table>

8 Related Documents

- www.infineon.com
  - XC226xn Data Sheet (xc226xn_ds_v1.2_2009_12.pdf), Feb 2010, v0.2
  - StarterKits and Evaluation Boards Manuals.
  - XC22xxN-Series DIP files for DAvE – latest version

- www.national.com
  - LM75A Data sheet