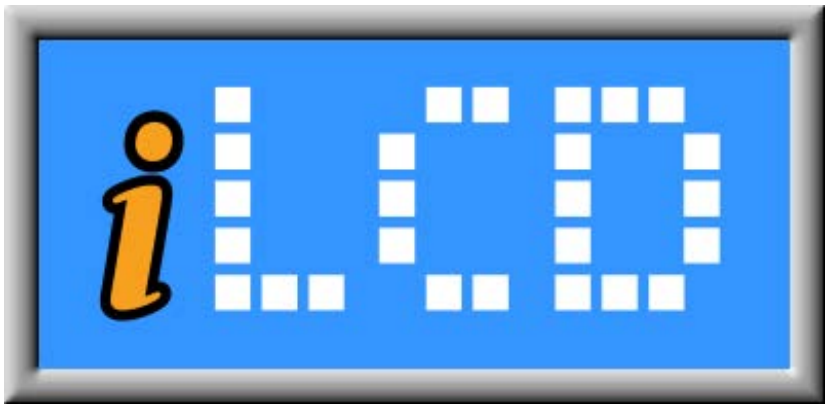


Next Generation Intelligent LCDs

I²C Application Note

Version 1.1
Document Date: November 23, 2012

demmel products gmbh ■ Weidelstr. 51 ■ A-1100 Vienna/Austria ■ Phone +43-1-6894700-0 ■ Fax+43-1-6894700-40 ■ Email: office@demmel.com ■ Web: www.demmel.com

Preface

This document describes the I²C specific communication with an iLCD controller only. Please read the "iLCD Commands" documentation available from http://www.demmel.com/download/ilcd/ilcd-commands.pdf to learn about how to send commands to iLCD panels first.

Controlling the iLCD Controller via I²C

In a typical master-slave configuration (the iLCD controller is a pure slave device) the slave has no possibility to talk without being asked for data. As there is the necessity to signal unexpected data existence, an extra pin named ALERT is added to the standard SCL (Clock) and SDA (Data) pins, which is pulled low (default setup) as long as the iLCD controller contains data to be read. By watching the state of this extra pin (usually via an interrupt) your application can react immediately to read any existing data. A different approach is to poll the iLCD controller periodically and to read data when the status read flags existence of data.

The iLCD's I²C communication protocol follows the rules of the Philips documents http://www.demmel.com/download/ilcd/i2c/i2c_bus_specification.pdf (describing the basics of I²C) and http://www.demmel.com/download/ilcd/i2c/smbus_specification.pdf (SMBus protocol, the proposed bus protocol is used for all commands). Please read the necessary parts of these two documents to learn more about how data is sent/read to/from an iLCD controller in advance.

The I²C part of the iLCD controller is carried out as a fast-mode 7-bit address slave device with up to 400 kHz clock frequency. Any 7-bit address can be set via the setup program.

Note:

DPC10xx iLCD controller: Please ensure your iLCD controller firmware has a version greater or equal version 2.04 for using the I²C functionality, earlier versions showed misbehavior on the I²C functionality under certain circumstances.

DPC20xx and DPC30xx iLCD controller: Please ensure your iLCD controller firmware has a version greater or equal version 1.13 for using the I²C functionality, earlier versions do not support I²C yet.

A sample source (written in standard C for 8051 micro-controllers) code showing how to talk with the iLCD controller via I²C can be found on http://www.demmel.com/download/ilcd/i2c_sample_code.zip

The output buffer of the iLCD panel has a size of 64 byte. Please make sure that you don't queue more byte, else the data is lost.

The iLCD panel can use clock stretching if the given clock speed from the master is too high.

The following I²C commands are implemented in any iLCD controller:

Read Status

The "Read Status" command is carried out as a "Read Byte Protocol" due to the SMBus specification chapter 5.5.5 where "Command Code" has the value Hex 01. The data byte read from the iLCD controller has the following meaning:

Bit 0: If set, some data bytes are available for reading
Bit 1: Indicates receive data overrun, that means the iLCD controller could not process all data (never happens if the command response is evaluated) - automatically reset after status has been read.
Bit 2: Indicates transmit data overrun, that means the iLCD controller could not send all data because they were not retrieved in time - automatically reset after status has been read.

Get Data Size to Read

The "Get Data Size to Read" command is carried out as a "Read Byte Protocol" due to the SMBus specification chapter 5.5.5 where "Command Code" has the value Hex 02. The data byte read from the iLCD controller describes the number of bytes available in the iLCD's output buffer for reading.

Get Info

The "Get Info" command is carried out as a "Read Word Protocol" due to the SMBus specification chapter 5.5.5 where "Command Code" has the value Hex 03. The data word read from the iLCD controller is a combination of "Read Status" (first byte read = low data byte) and "Get Data Size to Read" (second byte read = high data byte). The meaning of the two bytes is the same as described above.

Block Read

The "Block Read" command is used to read data according the section "Data Sent By The iLCD Controller" in the "iLCD Commands" documentation. This data is normally sent by the iLCD automatically when communicating via a serial port, but have to be retrieved via a command due to the nature of the I²C protocol.

The "Block Read" command is carried out as a "Block Read Protocol" due to the SMBUS specification chapter 5.5.7. The "Command Code" used for "Block Read" has the value Hex 04.

If "Byte Count" is greater than the number of available bytes stored in the output buffer of the iLCD controller, the dummy byte hex FF is send after the last available character and the device returns to receive mode then.

There is no need to read the complete iLCD's output buffer in one chunk, multiple "Block Reads" can be done as long there is some data in the output buffer; the number of available bytes can be retrieved via the "Get Data Size to Read" or "Get Info" command at any time.

Block Write

The "Block Write" command is used to send data to the iLCD controller. Any data as described starting from section "Command Description" in the "iLCD Commands" documentation can be sent via the "Block Write Command".

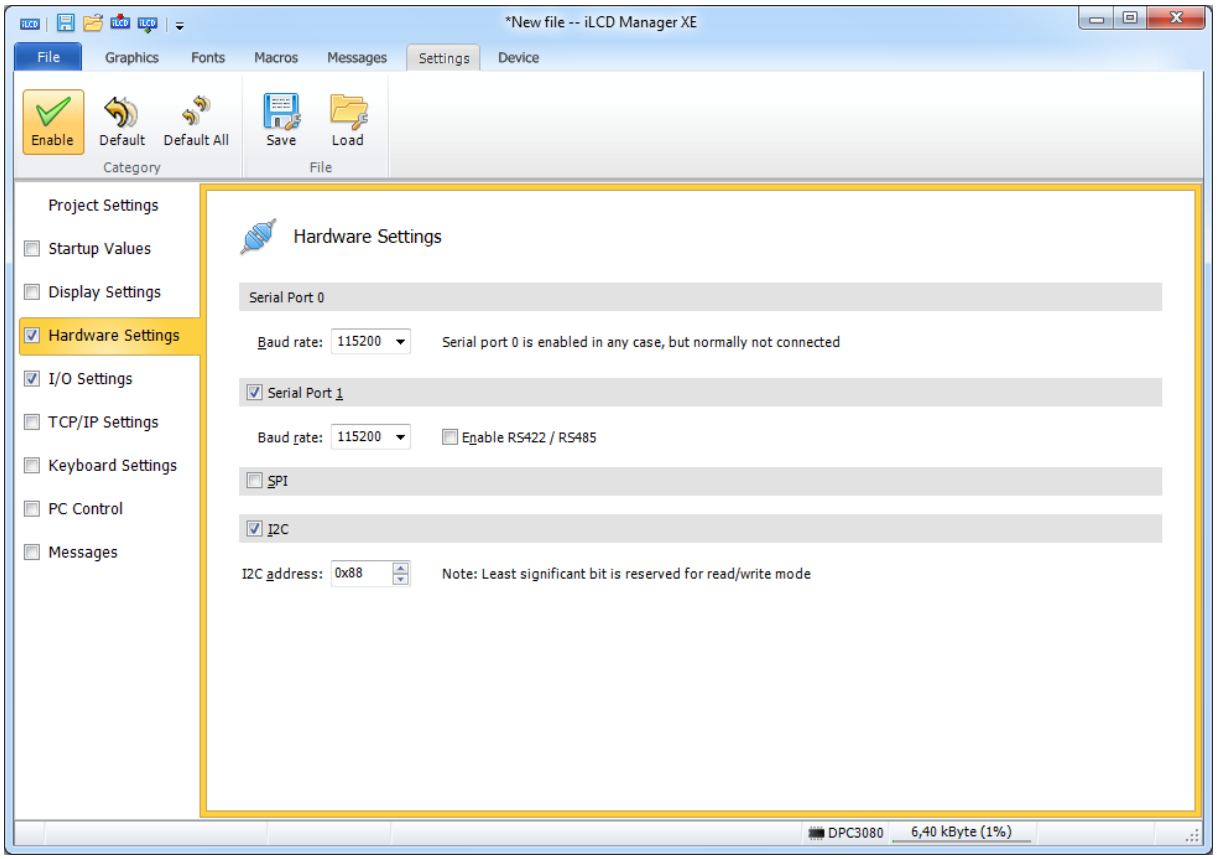
Please note that iLCD commands must not be sent within one "Block Write" command, they can be split up in as many parts as required. After an iLCD command has been processed, the ACK/NACK (Hex 06/15) is stored to the iLCD controller's output buffer and must be read via a "Block Read" command then.

The "Block Write" command is carried out as a "Block Write Protocol" due to the SMBUS specification chapter 5.5.7. The "Command Code" used for "Block Write" has the value Hex 05.

I²C on iLCD in Real-Life

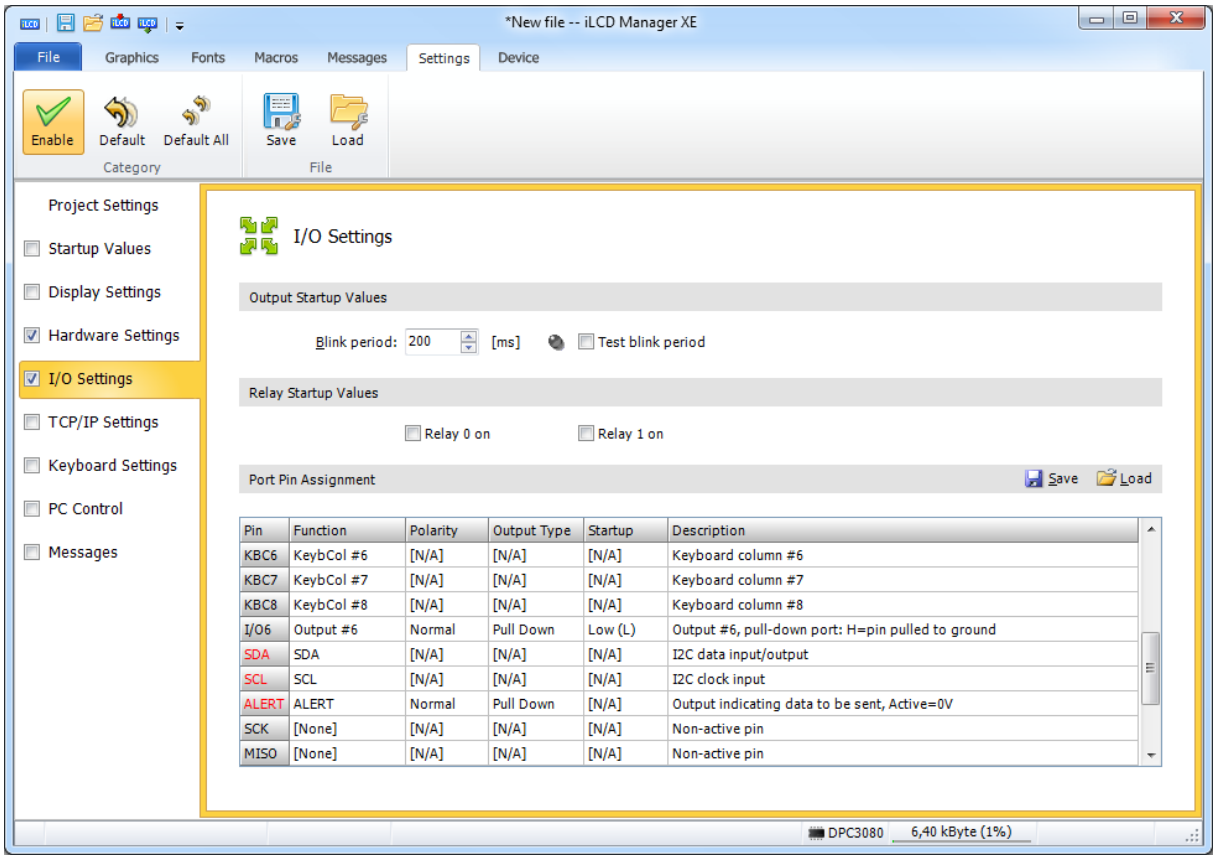
Setting Up the iLCD Panel for I²C Communication

With the iLCD Manager XE go to the "Settings" page, check the "Hardware Settings" and "I2C" checkbox.



The iLCD Manager XE automatically sets the "I/O Settings". See the pins marked red: SDA, SCL and ALERT.

When using an iLCD panel with a DPC20xx or a DPC30xx iLCD controller, the ALERT function can be mapped to a different pin and/or it's polarity (normal/inverted) and output type (push/pull or pull down) can be changed as well.



Finally write data to the iLCD panel by using the "Write Project to iLCD" button.

Sending/Receiving Data via I²C

To clarify the I²C communication with iLCD via I²C, we've made some screenshots showing the pin states in detail.

All I²C data shown in the screen dumps below have been sent via our sample code running on a 8051 micro-controller. This sample code can be freely downloaded from the demmel products web site at http://www.demmel.com/download/ilcd/i2c_sample_code.zip.

All screenshots have been made with I²C address set to 0x88 and ALERT pin enabled via the iLCD setup software as mentioned above.

Sending a Block of Data

Screenshots 1, 2 and 3 show how to send the command "\iDT-Hi-\r\0" (drawing the text "-Hi-" with a trailing carriage return).

Screenshot 1 shows the details of the clock and data pin, the interpreted I²C shows the hex bytes sent to the iLCD, where "W:44" indicates address writing introducing a write operation (= bit 0 cleared) to the iLCD. Please note, that the address is interpreted in a different way as the iLCD setup software does (it does not let you enter a I²C address with bit 0 set), so shifting hex 44 one bit left gives the expected hex 88 address value set via the setup software. The "S" indicates the start , "A" the acknowledge, and "P" the stop state of I²C. The

second byte is the "Command Code", for "Block Write" it has the value Hex 05. The third byte indicates how many bytes you now want to send. Here Hex 09 for 9 bytes of the command "\iDT-Hi-\r\0".

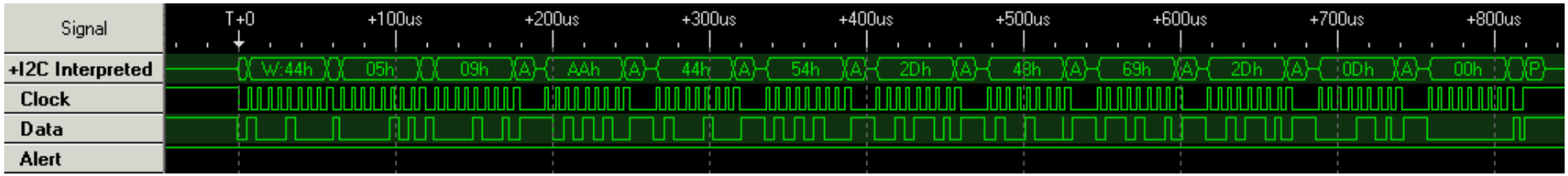
Screenshot 2 has the same contents as screenshot 1, but the magnification is lower to see the ALERT pin becoming active when the iLCD has processed the command and has data to be retrieved from (the <ACK>, Hex 06 value, answering any successfully processed command).

Screenshot 3 basically shows the same as screenshot 1 and 2, but the ALERT pin is already active before the command was sent due to data not retrieved from the iLCD controller yet. The I²C interpretation is shown in ASCII in this picture.

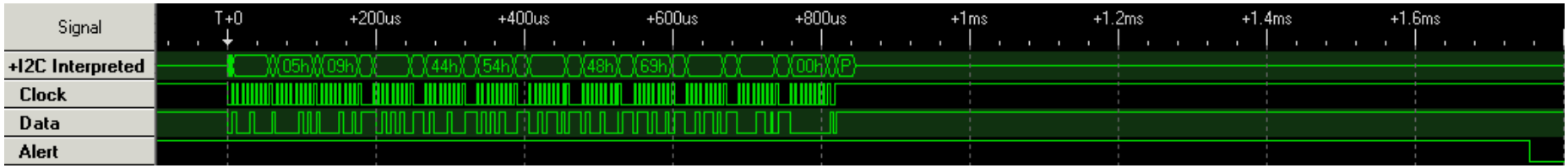
Reading a Block of Data

Screenshot 4 shows how to read back data from the iLCD (in this case the <ACK>). After sending the Read Block command (Hex 04) the address is sent again after a repeated start, but this time with bit 0 set indicating the following read command. The first byte read (hex 01) indicates the number of bytes to be sent from the iLCD panel, the next bytes (in our case only one byte) are the data read from the iLCD panel.

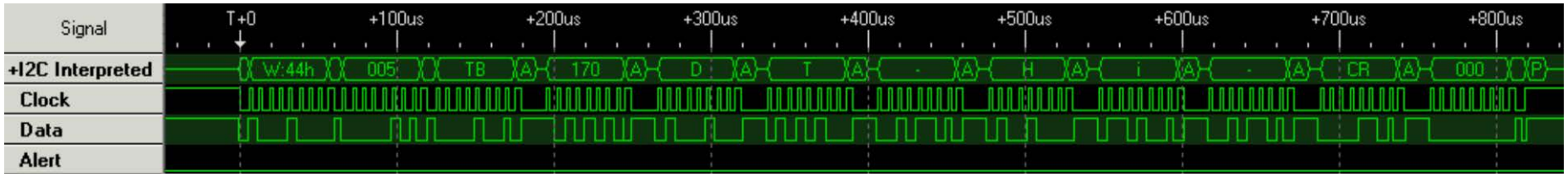
Screenshot 1: Sending Block of Data (Hex-Interpreted)



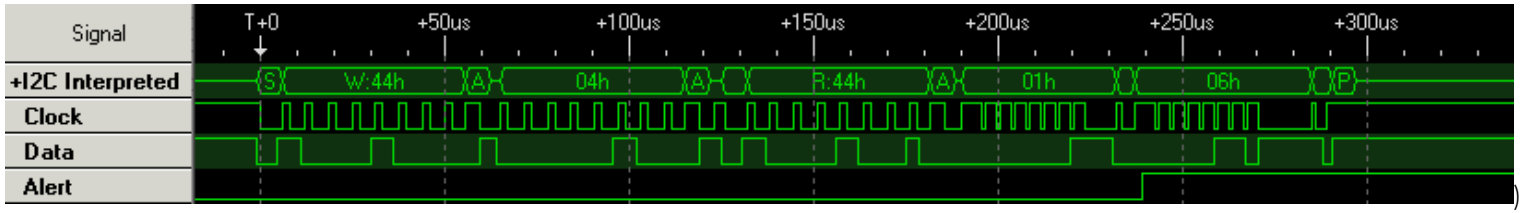
Screenshot 2: Sending Block of Data and Watching the ALERT pin



Screenshot 3: Sending Block of Data (Ascii-Interpreted)



Screenshot 4: Reading Block of Data (ACK



Revision History

Date	Rev. #	Revision Details
June 6, 2007	1.0	First issue
November 23, 2012	1.1	Description for iLCD Manager XE added

If you find any errors in this document, please contact demmel products at support@demmel.com