SLCD Application Note

In System Programming Example for:
SLCD+, SLCD43, and SLCD6 (f/w versions 2.7.0 and later)
SLCD5, SLCD5+ (f/w versions 1.2 and later)

Reach Technology, Inc.
sales@reachtech.com
(503) 675-6464
April 19, 2011

1 Overview

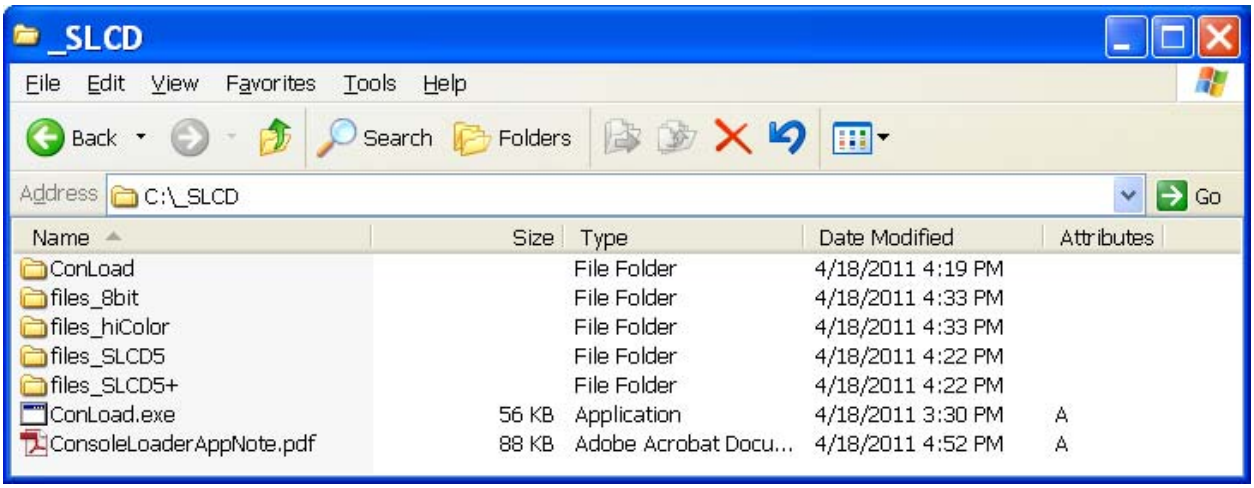
This Application Note describes a 'C' program demonstrating In-System-Programming of SLCD family bitmap and macro data via a PC COM port. The 'C' code is written to be easily ported to run on a target system's "host" processor to reprogram the SLCD. The program is implemented as a Windows Console Application that is executed from a Command Prompt window under Windows XP (or later) and uses command line arguments to specify the *PC COM port, the name of a ".bin" (or ".fwu") file (produced by the BMPLoad.exe program separately distributed by Reach), and whether or not the file contains firmware. Distributed with this Application Note is source code for the program, a M/S Visual Studio 2003 project for recompiling it, and small ".bin" files that can be used to verify operation of the program.

Firmware updates are also supported for the SLCD+, SLCD6, and SLCD43 controllers.

*Note: for newer PC's that have USB ports instead of COM ports, use a USB-Serial Adapter based on the FTDI chipset and install the appropriate drivers from here. DigiKey has some adapters you can order here.

2 Contents

This Application Note is distributed as a ".zip" archive file; open the file with Windows Explorer and copy its contents to a folder on your hard drive (for example, C:_slcd). This folder should now look something like:



"ConLoad" contains the source code and a M/S Visual Studio 2003 "solution" for examining the program, recompiling it, debugging it, etc.

"ConLoad.exe" is a Windows Console Application program, built from the aforementioned source code and solution in the Release configuration.

"ConsoleLoaderAppNote.pdf" is this Application Note.

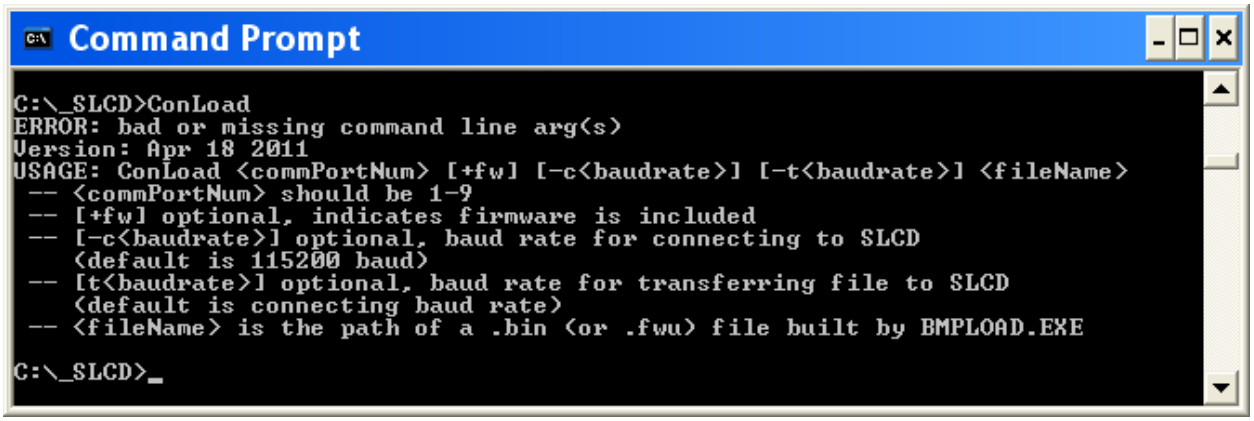
Folder “files_8bit” contains, “data_8bit.bin”, the files used with BMPload to build it, and “crc_8bit.txt”; these files are used for testing the program on SLCD6 or SLCD+ products running 8-bit color firmware.

Folder “files_hiColor” contains similar files for testing SLCD43, SLCD6, or SLCD+ products running high color firmware.

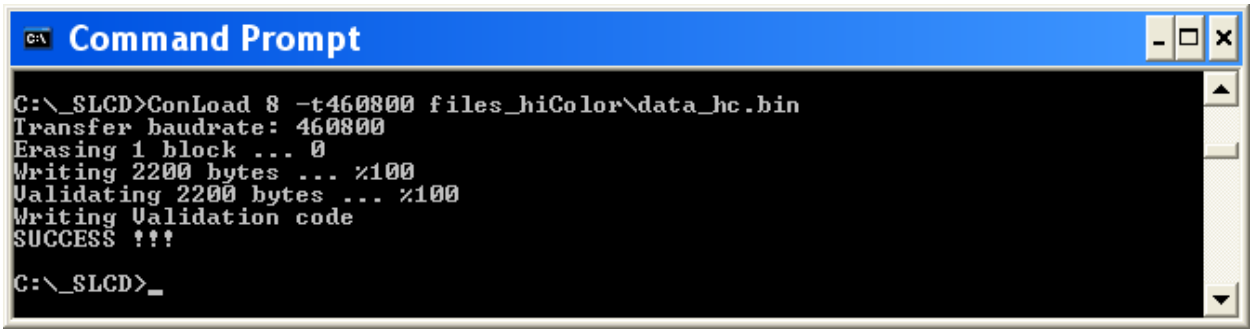
Folders “files_SLCD5” and “files_SLCD5+” contain similar files for testing SLCD5 or SLCD5+ products.

3 Using the Program

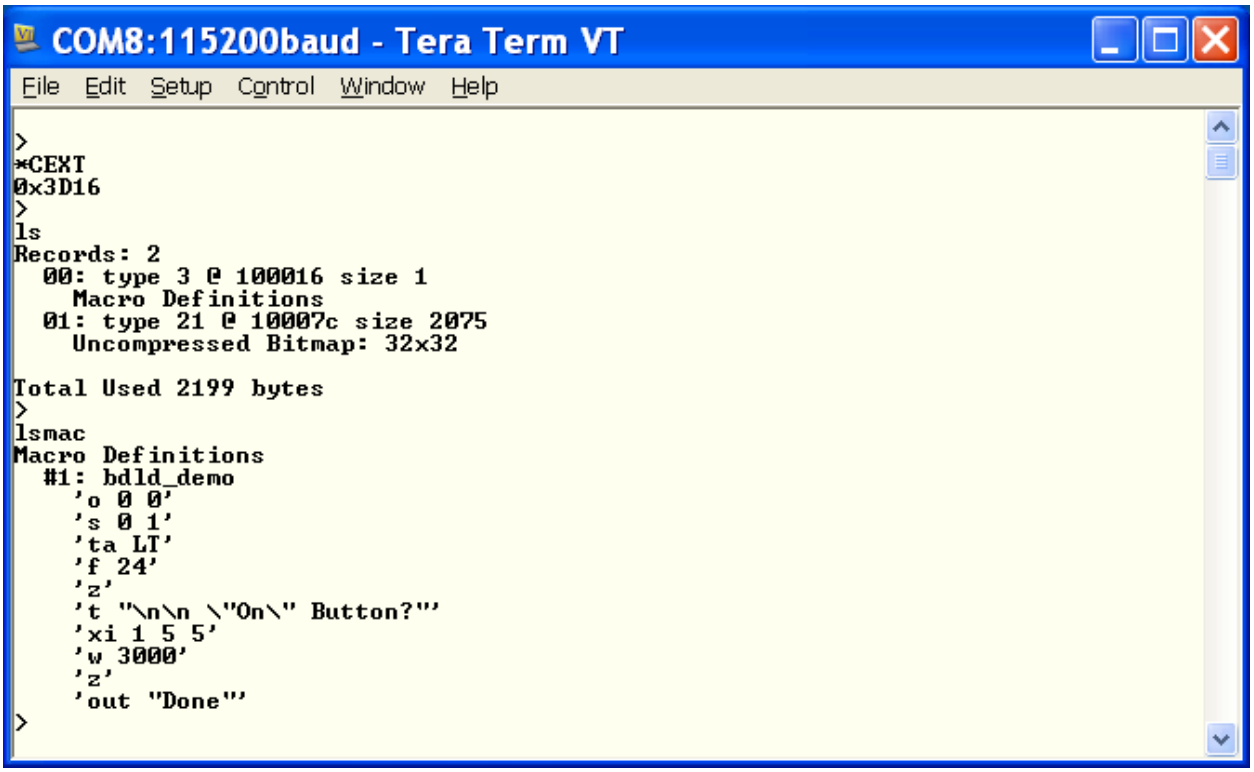
In order to verify the program runs on your PC, open a Command Prompt window, cd to the folder into which you copied this Application Note, and execute the program without any command line arguments. This will produce an ERROR message followed by a USAGE message, which identifies the program's expected command line arguments:



To demonstrate the program's functionality, connect an SLCD Display Module to one of the PC's COM ports (use USB-Serial Adapter if required). The program assumes it will have exclusive use of the COM port and that the SLCD is configured to run at 115200 baud. Be forewarned: the SLCD's existing bitmap, macro, and font data will be overwritten. Assuming your PC is using COM8 to control the SLCD (through a USB-Serial Adapter, which can run at 460800 baud) and the SLCD model is SLCD43 (uses high color firmware), you would run the program with the command line arguments shown below and see something like this:

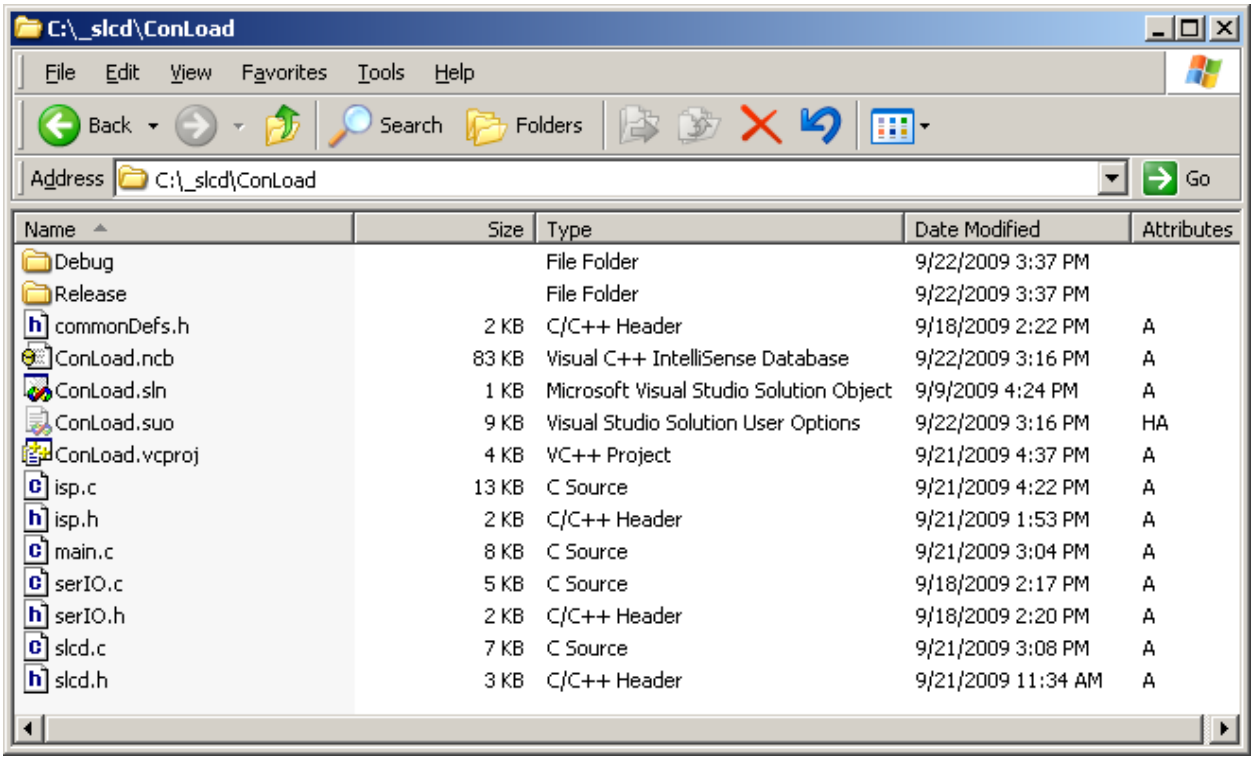


Next, verify the CRC of the data in the SLCD now matches that in the appropriate "crc_hc.txt" file: open the file “files_hiColor\crc_hc.txt” and note the CRC (0x3D16); then, open a terminal connection to SLCD and enter the command, "*CEXT" and press Enter. The result should match. Now, use the "ls" and "lsmac" commands to examine the SLCD contents. You should see something like this on your terminal:



4 Examining the Program

All the source code for the program is in the "ConLoad" folder. Viewing its contents with Windows Explorer, you should see something like:



Name	Size	Type	Date Modified	Attributes
Debug		File Folder	9/22/2009 3:37 PM	
Release		File Folder	9/22/2009 3:37 PM	
commonDefs.h	2 KB	C/C++ Header	9/18/2009 2:22 PM	A
ConLoad.ncb	83 KB	Visual C++ IntelliSense Database	9/22/2009 3:16 PM	A
ConLoad.sln	1 KB	Microsoft Visual Studio Solution Object	9/9/2009 4:24 PM	A
ConLoad.suo	9 KB	Visual Studio Solution User Options	9/22/2009 3:16 PM	HA
ConLoad.vcproj	4 KB	VC++ Project	9/21/2009 4:37 PM	A
isp.c	13 KB	C Source	9/21/2009 4:22 PM	A
isp.h	2 KB	C/C++ Header	9/21/2009 1:53 PM	A
main.c	8 KB	C Source	9/21/2009 3:04 PM	A
serIO.c	5 KB	C Source	9/18/2009 2:17 PM	A
serIO.h	2 KB	C/C++ Header	9/18/2009 2:20 PM	A
slcd.c	7 KB	C Source	9/21/2009 3:08 PM	A
slcd.h	3 KB	C/C++ Header	9/21/2009 11:34 AM	A

"Debug", "Release", "ConLoad.ncb", "ConLoad.sln", "ConLoad.suo", and "ConLoad.vcproj" are used by M/S Visual Studio 2003. The rest of the folder is source code for the program. It is heavily commented, so it can be used as a reference design.

- "main.c" is where it all begins. This code handles the command line arguments and executes the program, using functions from the standard C Run Time Library and the other modules.
- "serIO.c" and "serIO.h" make up the module that handles opening (and closing) the PC COM port, configuring it, and sending and receiving serial data.
- "slcd.c" and "slcd.h" make up the module that handles sending SLCD commands and receiving SLCD responses.
- "isp.c" and "isp.h" make up the module that handles the basic operations of In-System-Programming.
- "commonDefs.h" is a 'C' header file containing common definitions used by the other modules.

5 Understanding the Program

The program is designed to support the following SLCD Controller Board models:

- SLCD43, SLCD6, SLCD+ *(collectively referred to in the code as SLCDx)*
- SLCD5
- SLCD5+

Reading through the code in main(), one can easily follow the basic flow of the program, which can be summarized as:

- Establish communications and determine SLCD model and firmware version
- Make sure the data in the ".BIN" file will fit in the SLCD's available Flash and is properly formatted for the SLCD model.
- Erase enough Flash sectors to hold the data
- Read the data from the .BIN file and write it into Flash, *except for the 1st 2 bytes, which need to remain erased until after the rest of the written data is validated.*
- Validate the data written to Flash -- see ISP_ValidateExtFlash() in isp.c for details:
    - Process it in chunks of upto 8K bytes (2^16 bits == 8K bytes), by comparing the 16-bit CRC of a chunk read from the .BIN file to the 16-Bit CRC of the corresponding chunk in Flash.
    - In the case of the 1st chunk read from the .BIN file, the 1st 2 bytes of the containing RAM buffer are replaced with 0xFF to match the 1st 2 unwritten (ie: erased) bytes of Flash, before the buffer is CRC'd.
    - Rewind the file and read in the 1st 2 bytes, and then write them to Flash (these form a magic value used to indicate the Flash data has been validated).
- If the .BIN file contained a firmware upgrade (SLCDx only), tell the SLCD to upgrade its firmware.
- If the model is not an SLCDx, tell it to reload the data from Flash into the active RAM copy.
- Declare success and quit.

6 Porting the Program

The program was developed to be easily ported to a target embedded system, assuming the standard C Runtime Library is available. Some parts of the implementation may need to be changed based on how well stdio is supported by the target system's development environment, but the bulk of the porting effort involves serial I/O.

The code in "serIO.c" and "serIO.h" will have to be changed to support the target system's serial port hardware. The program assumes that serial I/O responds to Xon/Xoff flow control when transmitting, that SERIO_GetChar() will timeout after SERIO_READ_TIMEOUT_MSEC if a char/byte is not available, and that SERIO_PutChar() and SERIO_PutString() block until transmission completes.

The code in "slcd.c" and "slcd.h" should require little or no change, as it uses the SERIO_xxx() functions to communicate with the SLCD. The constant, SLCD_MAXBYTES, is based on the maximum SLCD command size, and should not be changed.

The code in "isp.c" and "isp.h" should require little change, so long as the aforementioned serial I/O assumptions are met. If Xon/Xoff Tx Flow Control is not available, the implementation of ISP_WriteFileToExtFlash() will require a significant change. The existing implementation uses one "bdld" command to tell the SLCD to receive the whole binary data image, and then sends the image data a chunk at a time until it's all been sent. Without Xon/Xoff Tx Flow Control, it will be necessary to send a "bdld" command before each chunk and wait for it to complete, repeating until all the data has been sent.