SLCD Technical Note TN-100

Programming the SLCD bitmap / macro flash memory in an embedded system

December 3, 2004

© Copyright Reach Technology Inc. 2004
All Rights Reserved

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

1. Introduction

The SLCD controller contains user programmable flash memory that stores bitmaps and macros for the Graphical User Interface (GUI). This is referred to as “external” flash since the SLCD microcontroller itself contains flash memory. Reach provides the Microsoft Windows program “BMPload.exe” to download bitmaps and macro files into this memory via the SLCD serial port. Once deployed in an embedded system, there needs to be a mechanism to update the installed bitmaps and macros. This technical note describes the protocol necessary to program the SLCD external flash memory.

The BMPload program version 1.5.4 and above allows the user to either update the SLCD flash or save the equivalent memory data to a binary file. This file can be used by the embedded host processor to update the SLCD external flash.

Note that the external flash memory needs a specially formatted version of the macro and bitmap files. This formatting is performed by the BMPload program. Partial updating or in-place modification not supported. The flash must be cleared and have the complete binary file written to it for correct operation.

The sequence of commands necessary to program the external flash is as follows:

1. Erase external flash
2. Enter special programming mode
3. Send programming data with escape codes inserted as needed
4. Exit special programming mode

This note assumes the reader is familiar with the SLCD Controller Manual, the general SLCD serial command syntax and operation, and ‘C’ programming language data representation.

2. Protocol

2.1. *Erase External Flash*

The external flash memory must be erased before new data can be programmed. The erase command is SLCD software version dependent. For software version 2.1.1 and above, the command is as follows. There is a delay between the first response string and the second while the memory is erased. This delay is typically 11 seconds; there is no maximum specified by the part manufacturer but it would be safe to assume a maximum of 22 seconds.

```
Command:
      xmc 0xFEEB<return>

First Response:
      Erasing chip...<return>
Second Response (if chip erase successful):
      Erased chip    <return>
Second Response (if chip erase fails):
      Erased failed  <return>
```

2.2. *Enter External Flash Programming Mode*

The flash is programmed on a byte-by-byte basis. A programming mode supporting a special protocol is used to allow binary programming within the standard SLCD command structure. The programming mode is entered by the "xmp" command. Note that the command allows programming to start at any address in flash, however you must start at address 0 (see example) for correct operation.

```
Command:
      xmp <hiAddr><loAddr><debug><return>

Arguments:
      <hiAddr>   = 'C' style 4 hex digit number, e.g. 0x0000
      <loAddr>   = 'C' style 4 hex digit number, e.g. 0x0000
                 programming starts at the 32 bit address
                 (hiAddr<<16) | loAddr
      <debug>    = 0 do not provide debug messages
                 = 1 provide debug messaged
Successful Response:
      p><return>
Failure Response: (if programming address illegal)
      ><return>

Example:
      xmp 0x0000 0x0000 0<return>
```

2.3. *Sending data*

Once the special programming mode has been entered as indicated by the "p><return> prompt, the binary data is sent in a sequence of data packets. Each packet contains one or more binary data bytes followed by a <return>. An escape code is used for control and compatibility with the basic SLCD protocol. The mapping of raw binary data to escape code sequence is as follows:

Character to be sent	**'C' style**	**Escape sequence (2 bytes)**	**Escape sequence, 'C' style**
0x20	' '	0x20 0x20	" "
0x00	'\0'	0x20 0x30	" 0"
0x0A	'\n'	0x20 0x4E	" N"
0x0D	'\r'	0x20 0x52	" R"
0x08	'\b'	0x20 0x42	" B"
(exit programming mode)		0x20 0x51	" Q"

As the binary data is received the flash memory address to be written next is automatically incremented. The <return> at the end of each packet is not written to the memory. If more data is sent than there is room in the memory, the failure response will be sent to the host.

```
Data packet example: (9 bytes total)
      0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08<return>
Successful Response:
      p><return>
Failure Response:
      <return>Cannot write past end of external memory!<return>
      p><return>
```

2.4. *Exiting Programming mode*

To exit programming mode, send the 2 byte escape code " Q" (space followed by the letter Q) equivalent to hex `0x20 0x51` followed by <return>.

2.5. *External memory size query*

A command is available to determine the size of the external memory in bytes. This can be useful to prevent programming overrun or to determine the memory size when several OEM size options are offered. Note 1K = 1024 bytes, so 512K is 524,288 bytes

```
Command:
      xma<return>

Response: (for units with 512K external flash)
      524288<return>
```

3. Example

The example presented here is the serial data exchange between the Host and the SLCD needed to write the file TN-100.bin into external flash. This is the binary file produced by BMPload Version 1.5.4 with a single bitmap 01_check.bmp and the macro file TN-100.mac. These files are available in the TN-100.zip file associated with this Technical Note.

In this example, the host sends 64 bytes of binary data at a time to the SLCD. Due to the escape code insertion, the actual number of bytes sent per packet is greater than 64.

The file TN-100.bin contains 218 bytes of the following data (in hex):

```
EB FE 02 00 03 16 00 10 00 01 00 00 00 01 37 00
10 00 A3 00 00 00 01 1D 00 10 00 03 00 7A 00 66
20 32 34 42 00 74 20 22 48 65 6C 6C 6F 22 20 31
30 30 20 31 30 30 00 20 00 20 00 04 00 01 00 5A
00 10 00 4B 00 10 00 00 00 00 00 02 00 00 52 00
10 00 00 00 00 00 FF FF FF 00 FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FC 00 00 3F FC 00 00 3F FC FF FF 3F FC FF
FD 3F FC FF F3 3F FC FF E7 3F FC FF CF 3F FC FF
9F 3F FC E7 1F 3F FC C2 3F 3F FC 80 3F 3F FC C0
7F 3F FC E0 7F 3F FC F0 FF 3F FC F8 FF 3F FC FD
FF 3F FC FF FF 3F FC FF FF 3F FC 00 00 3F FC 00
00 3F FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF
```

The serial data exchange used to program this file is as follows:

Host Sends:	**SLCD Sends:**
<return>	><return>
xmc 0xFEEB<return>	Erasing chip...<return> Erased chip <return> <return>
xmp 0x0 0x0 0<return>	p><return>
0xEB 0xFE 0x02 0x20 0x30 0x03 0x16 0x20 0x30 0x10 0x20 0x30 0x01 0x20 0x30 0x20 0x30 0x20 0x30 0x01 0x37 0x20 0x30 0x10 0x20 0x30 0xA3 0x20 0x30 0x20 0x30 0x20 0x30 0x01 0x1D 0x20 0x30 0x10 0x20 0x30 0x03 0x20 0x30 0x7A 0x20 0x30 0x66 0x20 0x20 0x32 0x34 0x42 0x20 0x30 0x74 0x20 0x20 0x22 0x48 0x65 0x6C 0x6C 0x6F 0x22 0x20 0x20 0x31 0x30 0x30 0x20 0x20 0x31 0x30 0x30 0x20 0x30 0x20 0x20 0x20 0x30 0x20 0x20 0x20 0x30 0x04 0x20 0x30 0x01 0x20 0x30 0x5A<return>	p><return>
0x20 0x30 0x10 0x20 0x30 0x4B 0x20 0x30 0x10 0x20 0x30 0x20 0x30 0x20 0x30 0x20 0x30 0x20 0x30 0x02 0x20 0x30 0x20 0x30 0x52 0x20 0x30 0x10 0x20 0x30 0x20 0x30 0x20 0x30 0x20 0x30 0x20 0x30 0xFF 0xFF 0xFF 0x20 0x30 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFC 0x20 0x30 0x20 0x30 0x3F 0xFC 0x20 0x30 0x20 0x30 0x3F 0xFC 0xFF 0xFF 0x3F 0xFC 0xFF<return>	p><return>
0xFD 0x3F 0xFC 0xFF 0xF3 0x3F 0xFC 0xFF 0xE7 0x3F 0xFC 0xFF 0xCF 0x3F 0xFC 0xFF 0x9F 0x3F 0xFC 0xE7 0x1F 0x3F 0xFC 0xC2 0x3F 0x3F 0xFC 0x80 0x3F 0x3F 0xFC 0xC0 0x7F 0x3F 0xFC 0xE0 0x7F 0x3F 0xFC 0xF0 0xFF 0x3F 0xFC 0xF8 0xFF 0x3F 0xFC 0xFD 0xFF 0x3F 0xFC 0xFF 0xFF 0x3F 0xFC 0xFF 0xFF 0x3F 0xFC 0x20 0x30 0x20 0x30 0x3F 0xFC 0x20 0x30<return>	p><return>
0x20 0x30 0x3F 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF<return>	p><return>
#20#51<return>	><return>