Using LCDproc with Technolgic Systems X86 SBC

Technologic SYSTEMS

Introduction:

Official downloads and documentation for LCDproc can be found at http://lcdproc.omnipotent.net/. This is a supplementary document that quickly walks through a basic installation on a Technologic Systems' Single Board Computer (SBC). This document also demonstrates a basic use of the commands for LCDd with a simple walk-through with the use of telnet. This document does not go into network programming. The lcdproc package does contain some good examples of some C and Perl source code that display various information about the system and displaying that information on the LCD display.

LCDproc is consists of two parts: LCDd (the daemon that interacts with the LCD display), and user clients (which tell the daemon what to display on the LCD display). LCDd must be installed and running on the SBC. Because LCDd listens on a network interface, clients can be written in any language that can send data through networking sockets. This includes c, Perl, VB, Java, bash scripts pumping output to a program such as netcat, or a user entering commands by hand through a telnet session. As you can deduce, the client need not reside on the SBC or host itself.

All parts of this document assume that you have a working knowledge of Linux. This document also assumes that you are using a LCD display purchases from Technologic Systems that is connected properly to the LCD port a 3x00 or 5x00 series SBC from Technologic Systems. See your SBC manual for more information connecting a LCD to the LCD port of your SBC.

Part 1 is a quick installation procedure. Part 2 walks you through some simple commands that display data on the LCD. It demonstrates the flexibility that LCDproc has by including no code... just you, the user, entering commands via a telnet session. Part 3 lists all the available commands and their syntax. Functions and their explanation text are copied from servers/client_functions.c

For detailed information on the various commands and their meanings, see lcdproc source files themselves, particularly,
servers/client_functions.c, servers/client_functions.h, servers/widget.c, and servers/render.c

Quick install procedures

1. download lcdproc 0.4.3 and unpack it
2. download and apply the TS Patch for lcdproc 0.4.3
3. ./configure –enable-drivers=hd44780 && make
4. copy LCDd.conf to the SBC /etc and servers/LCDd to the SBC /usr/bin
5. Read the LCDd.conf file, editing it your preferences (for instance, change the IP it is bound to to access it from the network, or change Foreground=yes to Foreground=no to make it a background process)
6. start up LCDd (/usr/bin/LCDd)

Quick Demo of Commands:

This assumes LCDd is running, bound to IP 127.0.0.1 on port 13666.

From the board, type in:
telnet localhost 13666
after it connects, type in :
hello
you should receive: **connect LCDproc 0.4.3 protocol 0.3 lcd wid 24 hgt 4 cellwid 5 cellhgt 8**

Type in the following:

```
client_set -name Example
screen_add myscreen
screen_set myscreen -duration 10 -priority 2
widget_add myscreen simplewidget string
widget_set myscreen simplewidget 1 1 “Testing 1 2 3    !”
```
View the LCD
```
widget_set myscreen simplewidget 1 2 “Testing 1 2 3    !”
```
View the LCD
```
widget_add myscreen scrolling scroller
widget_set myscreen scrolling 7 1 14 1 v 5 “1234567890abcd”
```
View the LCD
```
widget_set myscreen scrolling 7 1 14 1 m 3 “Welcome...I
Hope you have a pleasant day!       :)       “
```
View the LCD
and quit your telnet session

List of client commands

test_func

- Debugging only.. prints out a list of arguments it receives

hello

- The client must say "hello" before doing anything else. It returns a string of info about the server to the client
- *usage*: hello

client_set

- sets info about the client, such as its name
- *usage*: client_set -name <id>

client_add_key

- Tells the server the client would like to accept keypresses of a particular type
- *usage*: client_add_key <keylist>

client_del_key

- Tells the server the client would NOT like to accept keypresses of a particular type
- *usage*: client_del_key <keylist>

screen_add

- Tells the server the client would like to accept keypresses of a particular type when the given screen is active on the display
- *usage*: screen_add_key <screenid> <keylist>

screen_del

- Tells the server the client would NOT like to accept keypresses of a particular type when the given screen is active on the display
- *usage*: screen_del_key <screenid> <keylist>

screen_set

- Configures info about a particular screen, such as its name, priority, or duration
- *usage*: screen_set <id> [-priority <int>] [-name <name>] [-duration <int>] [-wid <width>] [-hgt <height>] [-heartbeat <type>]

widget_add

- Adds a widget to a screen, but doesn't give it a value
- usage: widget_add <screenid> <widgetid> <widgettype> [-in <id>]
  - WidgetTypes can be one of the following:
    - title
    - hbar
    - string
    - vbar
    - scroller
    - frame
    - num

  - Read server/widget.h and server/widget.c for detailed information

widget_del
- Removes a widget from a screen, and forgets about it
- *usage*: widget_del <screenid> <widgetid>

widget_set
- Configures information about a widget, such as its size, shape, contents, position, speed, etc...
- *usage*: widget_set <screenid> <widgetid> <widget-SPECIFIC-data>
- The following lists widget type's parameters
  - WID_STRING: takes “ x y text”
  - WID_HBAR: hbar takes “x y length “
  - WID_VBAR: Vbar takes "x y length”
  - WID_ICON: Icon takes "x y binary_data"
  - WID_TITLE: title takes "text"
  - WID_SCROLLER: Scroller takes "left top right bottom direction speed text"
    - direction can be m (marquee), h(horizontal), v(vertical)
  - WID_FRAME: Frame takes "left top right bottom wid hgt direction speed"
  - WID_NUM: Num takes "x num"

menu_add
- Adds a menu to the client; handled by the server...
- *usage*: menu_add ...?

menu_del
- Removes a client's menu and all contents from the server
- *usage*: menu_del ...?

menu_set
- Sets info about a menu, but not its items. For example, should the menu be top-level, or buried somewhere?
- *usage*: menu_set ...?

menu_add_item
- Adds an item to a menu
- *usage*: menu_add_item ...?

menu_del_item
- Deletes an item from a menu
- *usage*: menu_del_item ...?

- **menu_set_item**
  Sets the info about a menu item. For example, text displayed, widget type, value, etc...
- *usage*: menu_set_item ...?

backlight
- Toggles the backlight, if enabled.
- *usage*: backlight <on|off|toggle|blink|flash>

output
- Sets the state of the output port (such as on MtxOrb LCDs)
- *usage*: output <on|off|int>

noop
- Does nothing, returns "noop complete" message This is useful for shell scripts or programs that want to talk with LCDproc and not get deadlocked. Send a noop after each command and look for the "noop complete" message.

info
- *usage*: info

sleep
- *usage*: sleep <seconds>