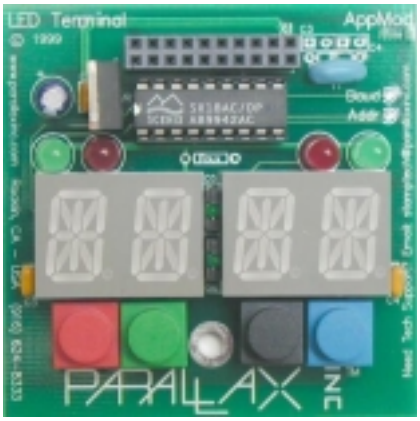
LED Terminal AppMod

General Description



Many applications require the user to interact directly with a host system. The LED Terminal AppMod provides an interface for direct interaction between the user and the host. It offers a four digit alphanumeric LED display, six individual LEDs, four pushbuttons, and a real-time clock. The host communicates with the LED Terminal AppMod via a single serial line. The user may employ any standard baud rate between 2400 baud and 38.4K baud inclusive for communications. The commands allow the user to display alphanumeric messages, read keypress information, and read the time.

Serial Protocol

Communication is always initiated by the host, the LED Terminal AppMod will not respond otherwise. The command syntax is structured to facilitate the allowed commands and prevent bus contention in the event that more than one AppMod is on the same serial line.

!LT0<CMD><x,y>

- Discrete LED control bytes (see Programming Notes)
- Command and any data that may be associated (Caps only)
- Unit ID (currently, only one LED Terminal per bus supported)
- AppMod Prefix (2 byte address denoting the type of AppMod addressed)
- Preface Character alerts units on the bus that this message is an AppMod command and serves to help the LED Terminal sync up with the host.

Commands

ASC Show ascii (text) data as received via serial interface. The following code snippet will display the ASCII text “TIME” on the display and turn off all of the discrete LEDs:

```
serout   6, N19K2 ,[“!LT0ASCTime”,$0,$0]
```

Please note that although it is specified that the word “Time” is to be displayed, the word “TIME” will be displayed. This is because only capitalized letters are supported by the display. The range of characters supported is: A-Z,0-9,*,/, ,\,+,-. The comma and the period are not supported.

BCD Show Binary Coded Decimal data as received via serial interface. The following code snippet will display the BCD number “0734” on the display and turn off all of the discrete LEDs.

```
serout   6, N19K2,["!LT0BCD",0,7,3,4,$0,$0]
```

Please note that numbers (0-9) will be displayed as such, whereas the numbers 10,11,12,13,14,15 will be displayed as hexadecimal numbers, i.e. A, B,C,D,E,F

BIN Receive binary data via serial interface and display it as decimal data. The following code snippet will display the contents of Num (a 16 bit word) on the display. Remember that the data must be sent lowbyte, then highbyte since serial protocol demands the data is eight bits wide.

```
Num     var     word
serout  6, N19K2,["!LT0BIN",Num.lowbyte,Num.highbyte,$0,$0]
```

Althought the data sent is in binary form , the LED Terminal will automatically convert the binary data to decimal so that it is easily readable by humans when it is displayed. Numbers supported are: **0-270F** (hexadecimal), **0-0010011100001111** (binary), **0-9999** (decimal).

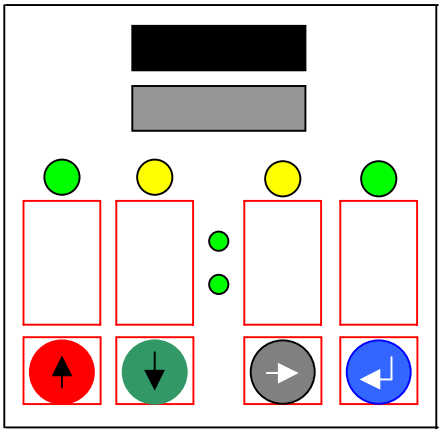
Mini-AppNote

The following program will demonstrate how you can use the ASC, BIN, and BCD commands work together on the LED Terminal AppMod using a BS2.

```
'TSTAT.bs2      Simple Thermostat Display Program

N19K2           con     32+$8000
DP2             con     $02             'Decimal point in second position
Temp            var     word

Start:          pause 900               'Wait for LED to power up and initialize
                Temp = 849                                      '84.9 degrees
                serout 6,N19K2,["!LT0ASC out",$0,$0]:  pause 400
                serout 6,N19K2,["!LT0ASCside",$0,$0]:  pause 400
                serout 6,N19K2,["!LT0ASCTemp",$0,$0]:          pause 800
                serout 6,N19K2,["!LT0BIN",Temp.lowbyte,Temp.highbyte,$02,$0]
                pause 1800

                Temp = 745                                      '74.5 degrees
                serout 6,N19K2,["!LT0ASC in ",$0,$0]:  pause 400
                serout 6,N19K2,["!LT0ASCside",$0,$0]:  pause 400
                serout 6,N19K2,["!LT0ASCTemp",$0,$0]:          pause 800
                serout 6,N19K2,["!LT0BIN",Temp.lowbyte,Temp.highbyte,DP2,$0]
                pause 1800
                                                                '74.2 degrees
                serout 6,N19K2,["!LT0ASCTemp",$0,$0]:          pause 400
                serout 6,N19K2,["!LT0ASC set",$0,$0]:  pause 400
                serout 6,N19K2,["!LT0ASC at ",$0,$0]:  pause 800
                serout 6,N19K2,["!LT0BCD",0,7,4,2,DP2,$0]
                pause 900
                goto Start
```

This program displays three different textual messages and the three different numbers associated with each text message. Please remember when programming that the commands ASC and BCD expect as parameters 6 bytes of data (4 for the display and 2 for the discrete LEDs) after the command. The BIN command expects to receive 4 bytes after the command (2 bytes for display data and 2 bytes for the discrete LEDs). Incomplete serial commands sent to the LED Terminal will be ignored.

NKY Show an ascii (text) prompt as received via serial interface and allow the user to enter a numeric value using the local pushbuttons. The following code snippet will flash the text, "TIME" on the display until the user hits any pushbutton. After the keypress, the LED Terminal will switch to data entry mode. In this mode, the display is redrawn with four underscore "_" characters, with the left most digit flashing. The user may then change the value of the digit that is flashing using the pushbuttons (red to increment and green to decrement) until the desired value is entered. Pressing the black pushbutton advances the flashing digit one place to the right so that all digits may be entered. Pressing the black pushbutton while the right most digit is flashing will result in a 'wrap-around effect' leaving the left most digit flashing. The blue pushbutton is used to complete the data enty mode. At any time the host may request the data entered with the NKR command detailed later in this document.



```
serout   6, N19K2,["!LT0NKY","Time",$0,$0]
```

NKR Request the value entered by the user (via the NKY command). The following code snippet will retrieve the value entered by the user (via the NKY command) and place the data in an array called "datain". Note: if no value has been entered by the user, datain will be loaded with the value **0000**.

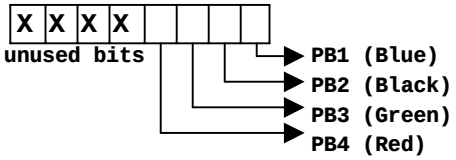
Mini-AppNote

The following program will demonstrate how you can use the NKY and NKR commands to work together on the LED Terminal AppMod using a BS2.

```
N19K2  con     32+$8000
datain var     byte(5)
TimeH  var     byte
TimeM  var     byte
Init:  serout  6,N19K2,["!LT0NKYTime",$0,$0]
Start: pause   15000
       serout  6,N19K2,["!LT0NKR"]
       serin   6,N19K2,5000,Start,[STR datain\4]
       pause   1000
       TimeH   = datain(0)*10+datain(1)
       TimeM   = datain(2)*10+datain(3)
       serout 6,N19K2,["!LT0RTI",$0,TimeH,TimeM,$0,$0,$0]
Done:  goto    Done
```

PBR Read the pushbuttons. The following code snippet will read a register on the LED Terminal AppMod that contains the last key(s) pressed since the last PBR command. Each time a PBR command is executed, the register containing the last key(s) pushed is cleared to zero. If no keys have been pressed and a PBR command is sent, the LED Terminal will respond with a zero.

```
N19K2  con 32+$8000
datain var byte
Start: serout 6,N19K2,["!LT0PBR"]
       serin  6,N19K2,[datain]
       debug  bin4 datain,cr
       pause  5000
       goto   Start
```



RTI Set the real-time clock to the time values sent by the host via the serial interface. The following code snippet will set the real-time clock to: day = 0, hour = 8, minute = 30, and second = 0.

```
N19K2  con 32+$8000
TimeD  con 0
TimeH  con 8
TimeM  con 30
TimeS  con 0

       serout  6,N19K2,["!LT0RTI",TimeD,TimeH,TimeM,TimeS,$0,$0]
```

RTH Hide the time value from the LED display (hrs:mins). The LED Terminal AppMod will display the time every second unless you 'hide' the time. The following snippet of code will hide the current time value (so that you may display something else).

```
N19K2  con 32+$8000

       serout  6,N19K2,["!LT0RTH"]
```

RTS Show the current real-time clock value on the LED display (hrs:mins). This is the default mode when the LED Terminal is powered up. The display will be updated with the local time every second.

```
N19K2  con 32+$8000

       serout  6,N19K2,["!LT0RTS"]
```

RTR Read the current value of the real-time clock. The following code snippet will retrieve the real-time clock value and store it in as array called, "datain" and format and echo the data to the debug screen.

```
N19K2  con 32+$8000
datain var byte(4)

       serout  6,N19K2,["!LT0RTR"]
       serin   6,N19K2,[STR datain\5]
       debug   "Day:  ",dec2 datain(0),cr
       debug   "Time: ", dec2 datain(1),":", dec2 datain(2)," "
       debug   dec2 datain(3), ".", dec1 datain(4), cr
```

The output format for the date is: dd
where dd is the day: 0-99
The output format for the time is: hh:mm ss.tt
where hh is the hour: 0-23
where mm is the minute: 0-59
where ss is the second: 0-59
where tt is the 0.1 second: 0-9
This code will cause the following to appear on the debug screen (your time may vary):

```
Day:  00
Time: 00:05 58.6
```

Mini-AppNote

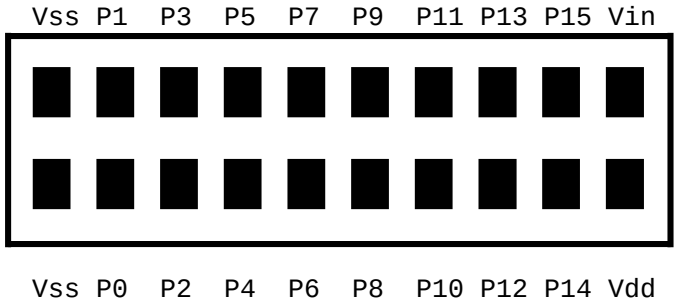
Here's an example of how NKY and the RTI functions can work together to initialize the LED Terminal's real-time clock.

```
N19K2  con     32+$8000
datain var     byte(5)
TimeH  var     byte
TimeM  var     byte

Init:  serout  6,N19K2,["!LT0NKYTime",$0,$0]
Start: pause   10000
       serout  6,N19K2,["!LT0NKR"]
       serin   6,N19K2,5000,Start,[STR datain\4]
       debug   hex datain(0)," ", hex datain(1)," "
       debug   hex datain(2)," ", hex datain(3),cr
       TimeH  = datain(0)*10+datain(1)
       TimeM  = datain(2)*10+datain(3)
       serout 6,N19K2,["!LT0RTI",$0,TimeH,TimeM,$0,$0,$0]
Done:  goto Done
```

Electrical Specifications

The LED Terminal AppMod requires 6 – 9 Volts DC and will draw 40-150 mA depending on how many LED segments are energized. Please use caution when touching or handling the LED Terminal Display while it is energized (or even if it was recently energized) as the on-board voltage regulator can get warm during normal operation.

As with all AppMods, the LED Terminal AppMod has a special stack-through 2x10 header. This header allows you to connect the LED Terminal to either the GrowBot, the BOE-Bot, the Stamp Activity Board, the Super Carrier Board, and of course, other AppMods. We've provided the pinout of the AppMod header below, as viewed from the top.



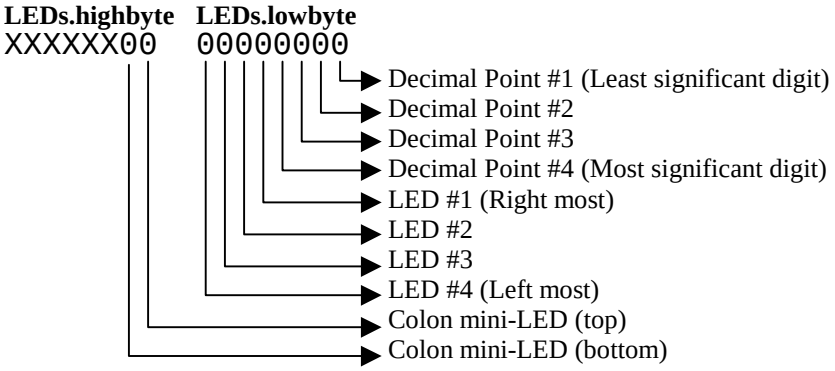
Programming Notes

1) Throughout the examples mentioned herein, I/O Pin 6 has been referenced as the serial pin for all serial communications. This is because, with respect to the AppMod header included on various products such as the GrowBot, the Board of Education, the BOE-Bot, and the SuperCarrier Board, I/O Pin 6 has been hardwired to serve this function.

2) Throughout the examples mentioned herein, the baudmode parameter has been, "N19K2". It is depicted this way since the value of N19K2 will be different depending on which stamp you use. Also note that all code snippets herein are for the BS2. Please refer to the AppMod diskette for examples of code for the BS2-SX and the BS1.

3) When sending numeric data (BCD or BIN) to the display, the LED Terminal AppMod will automatically blank leading zeroes based on where the decimal point is located.

4) You may have noticed that many of the code snippets here have a couple of ",$0"s trailing. This is to allow the user access to the decimal points and the discrete LEDs on-board. The following table correlates the values for the LED bytes with their physical positions.



The following code snippet will display a zero while cycling slowing through all of the LEDs, decimal points, and the LEDs that make up the colon:

```
LEDs    var     word
Init
        LEDs = $1
Start
        pause 1000
        serout 6,32+$8000,["!LT0BIN",$0,$0,LEDs.lowbyte,LEDs.highbyte]
        debug bin2 LEDs.highbyte," ",bin8 LEDs.lowbyte,cr
        LEDs = LEDs << 1
        if LEDs.bit10 = 1 then Init
        goto Start
```

5) If the host sends an NKY command to prompt the user to input a value, the LED Terminal AppMod will wait until the user does so. The host may poll the LED Terminal AppMod anytime, before, during, or after the user inputs the data. If the host system decides to not wait any longer for the user to input data, the host may send a command, (i.e. RTS) to end the NKY mode.