Column #37, March 1998 by Jon Williams:

Goin' GUI

Graphical User Interfaces (GUIs) have been a PC standard for some time and, with the cost of intelligent graphic displays on the decline, GUIs are making their way into embedded applications. This month's project is a very simple pumping station simulator using the G12864 as the display. Since the circuitry is so simple, I assembled the project on Parallax's BASIC Stamp Activity Board (BSAC).

I've known Scott Edward for a few years now, and I've got to say one thing: the guy solves problems. A little over a year ago, I was considering a project using a graphical (versus character-based) LCD. It didn't take much research to convince myself that the project wasn't really practical, considering the Stamp's limited resources. Not long thereafter, I was chatting with Scott and asked him if he'd ever considered a backpack for graphic LCDs. I could sense Scott grinning as he said, "Be patient, Jon – I'm working on something you'll like." He wasn't kidding.

Scott's latest serial marvel is the G12864 Serial Graphics LCD. The module combines a 128x64-pixel backlit LCD with a serial interface. The G12864 can simultaneously display text (four lines of 16 characters) and graphics. Serial commands even allow you to draw lines with code. Since the G12864 comes with an excellent reference and code samples, our focus will be applying the combined text/graphics capability in a project. Let's get started.

Goin' GUI

All projects require planning, and this one is no exception. In fact, the use of a graphics display and the creation of those graphics requires even more planning and work. Here's an overview:

- Plan the project.
- Develop the graphics.
- Download graphics to the G12864.
- Write and test code.

The Project

As I just stated, our project is a very simple pumping station simulator. Please keep in mind that the purpose of this project is to demonstrate some of the capabilities of the G12864. Actual pumping station controls are much more sophisticated than what we'll present here.

Our project simulates a flow sensor by reading a potentiometer with the RCTIME function. The input is scaled to a maximum flow of about 1600 units. The program takes the flow reading and turns on up to four pumps to meet the demand. The pumps are capable of supplying 100, 200, 400, and 800 units, respectively, for a maximum combined output of 1500 units. If the flow demand exceeds the combined pump output by 25 units, the next pumping level is selected. If the total demand exceeds 1520 units, our station is shut down and must be reset.

For the display, I wanted a graphic that showed each of the four pumps, their status, and the flow demand. Pump status is displayed graphically – sort of (more on this later). The flow demand is output as a four-digit number.

While the G12864 is capable of displaying graphic images, it does not have the ability to display "sprites" (portions of a graphic page). So how am I going to display my pump status graphically? With text, that's how! Okay, okay, I know this sounds a bit confusing. Let me elaborate.

The G12864 contains 16 pages of non-volatile flash memory, with each page holding a 128x64 bit graphic image. The first three pages of memory are special. Pages zero and one contain the font used for text displays. The image on page two is called the splash screen. Configuration switch 6 on the G12864 allows the splash screen to be displayed at

power-up. We'll set this switch on to automatically display our pump station graphic. Each character on the font pages is eight pixels wide by 16 pixels tall, and all the characters up to ASCII 127 are predefined. This leaves room (on page 1) for up to 32 custom characters (ASCII codes 128 to 159). Since text and graphics can be combined on the G12864, custom characters will be used to indicate pump status. I designed my graphic so that each pump occupies a 16x16 pixel region. This means that two characters are needed to display each pump state: off, on, or error.

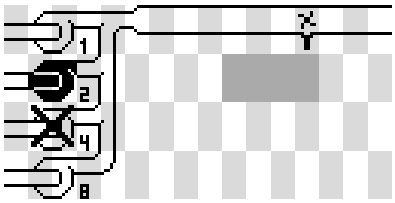
Developing G12864 Graphics

As with text editors, everyone has their favorite graphics program. Mine happens to be Paint Shop Pro (PSP) from Jasc, Inc. (see sources). PSP is one of the most popular shareware programs available today. I use it daily in my professional and personal projects – it's that good. In the following description, PSP key commands are indicated in parentheses. If you're using another graphics program, please refer to its documentation.

I start by creating a 16-color image (File | New) that is 128 pixels wide by 64 pixels tall. Since I'm going to combine text and graphics, I paint a checkerboard pattern to show me where the characters will fall. Each checker is eight pixels wide by 16 pixels tall (the same size as a character). I used white and light cyan as my checker colors. Use what you want, but keep these background colors very light. This will make the conversion to a two-color graphic – necessary for the G12864 – easier. I saved (File | Save As) this template as G12864.BMP.

With the template saved, I make a copy (Shift-D) and create my pump station master graphic. There's not a heck of a lot I can tell you here; if you're not particularly good at graphics design, you may want to enlist the help of a friend who is. Start with a sketch on paper or a white board. As you make progress, make a copy (Shift-D) so that you're not forced to start all over from a mistake. Once you've got your master graphic complete to your liking, save it (File | Save As). My project master is called PUMPSTA0.BMP. This is shown in Figure 37.1.

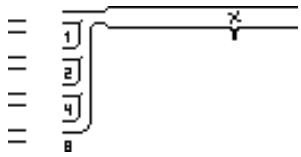
Figure 37.1: Project master drawing



You may be a bit puzzled by my master as it shows two pumps off, one on, and one with an error. Again, this is my master graphic and won't be downloaded to the G12864. Notice that I also defined where my flow value will be displayed by changing the appropriate checkers to yellow.

I made a copy of this image and cut out all of the pumps (background color is white; make selection then press Delete). Now I convert to a two-color image (Colors | Decrease Color Depth | 2 Colors). With light background colors, I used the Gray Values/Weighted/Nearest Color method. The resulting image is saved as PUMPSTA1.BMP (Figure 37.2 below).

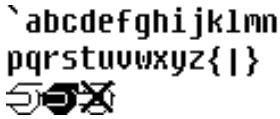
Figure 37.2: Project master drawing without pumps in two-color



When converting photos (like Scott's cat in the demo) use the Error Diffusion method. You may need to experiment with the brightness and contrast of the image to get the desired result.

With our background image complete, the last step in the graphics process is to add the pump graphics to font page 1. Start by making back-up files of the font pages that come with the G12864 (ALPHA0.BMP and ALPHA1.BMP). I made another copy of the master graphic, reduced it to two colors, then copied (select the area, then Ctrl-C) the pumps and pasted (Ctrl-E) the font page. The custom characters will have ASCII codes of 128 to 133. Be very careful not to disturb the other characters – you could get unexpected results when attempting to display text. With the pump images in place, the new font page is saved as ALPHA1PS.BMP (Figure 37.3).

Figure 37.3: Custom characters will have ASCII codes 128 to 133



With the graphics complete, we need to download them to the G12864. If you're using Windows 3.x, you can use BMPX.EXE, a DOS command-line utility supplied with the G12864. Start by removing power from the G12864 and setting configuration switch 5 (Protect/Write) and switch 6 (Blank/Screen 2) to ON. Connect the G12864 to your PC (Com1 or Com2) with a nine-pin serial cable and power it up. To download the updated font page at 9600 baud, you'd use the following syntax:

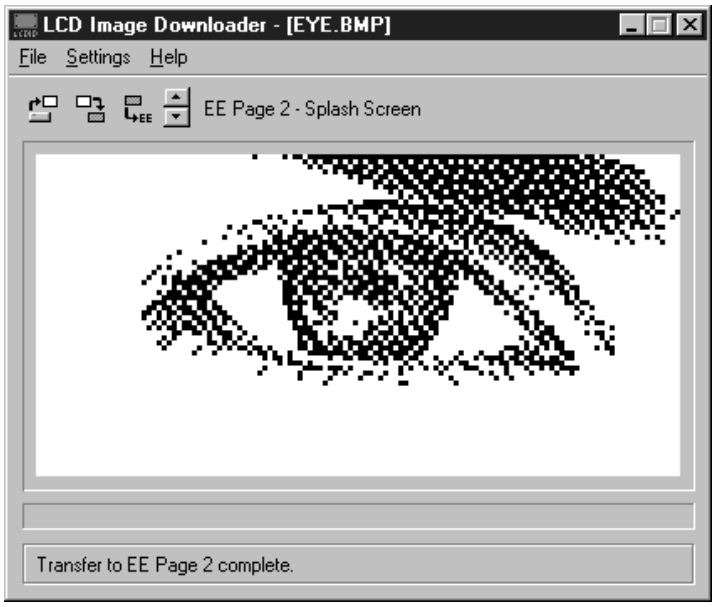
BMPX COM1 9600 ALPHA1PS.BMP

If everything is connected properly, the image will show up in the display. When the download is complete, you will be prompted to save the image to the EE memory. If you respond "Yes" (Y, then Enter), you will be asked for a page (0 to 15). Press 1, then Enter. Since page 1 is a font page, you will be asked to confirm the save to EEPROM. Press Y, then Enter, and the page will be written to memory. Repeat this process with PUMPSTA1.BMP and save it to page 2 (the splash screen).

While BMPX.EXE works fine from a DOS window in Windows95, I prefer graphical programs. Since Scott very generously supplied the source code for BMPX, I translated it and created a Win95 utility called LCDID (Figure 37.4). You can download LCDID.ZIP from my FTP directory (see sources).

After the images are downloaded and written, return configuration switch 5 to OFF (Protect), set switch 1 to ON (Demo), then cycle the power on the G12864. In demo mode, all 16 pages are displayed sequentially. This will let you check your new font page and splash page. Once you're satisfied, return switch 1 to OFF (Run) and cycle the power again.

Figure 37.4: LCDID Win95 utility allows easy download of images to the G12864



The Code

With the real hard work done, it's time to write our program (refer to Program Listing 37.1). The program is very simple – as far as BS2 programs go – but there are a couple of things worth pointing out.

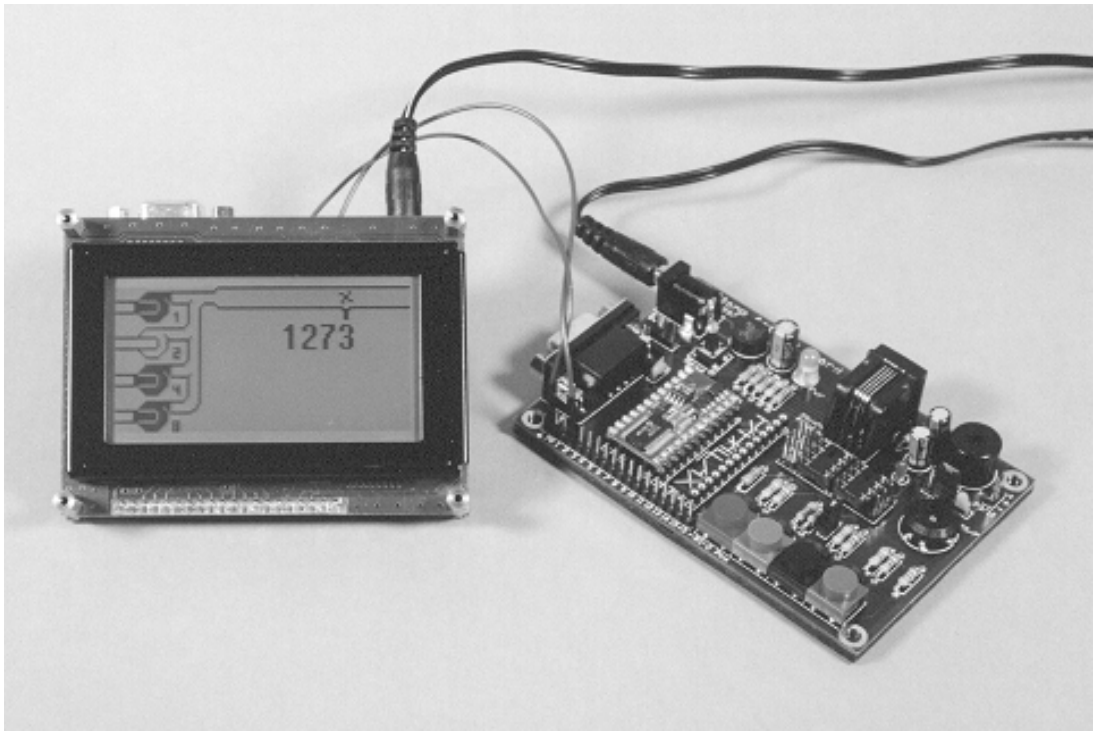
The G12864 uses ASCII character 16 (Ctrl-P) to position the text cursor. There are two ways to use the positioning command: text and binary. Using the text method, you would follow the positioning code with the ASCII text of the desired screen position (i.e., "25 "). The last character of the ASCII screen is a space to terminate the positioning mode. In an embedded application like this one, the binary method is preferred. Simply add the desired cursor position (0 to 63) to 64 and send the byte after the positioning command. A constant, FloPos, is defined as 64+25. This will be sent after the positioning command to put the flow reading at character position 25.

Since my station only has four pumps, a nibble-sized variable is used to store the status of the pumps (0 is OFF, 1 is ON). I overlaid bit-sized variables to make displaying pump status easy.

The program starts by waiting for one second. This gives the G12864 time to initialize before sending commands. After the delay, we clear the text screen and update the display with the subroutine called ShoFlo. This routine starts by sending the new flow value. The DEC4 parameter in the serial output command causes the flow to be displayed with four characters (using leading zeros).

After displaying the flow, we calculate the status of the pumps. This is a simple matter of dividing the flow demand by 100 and storing this value in the variable called pumps. If the flow demand is greater than 25 units more than our current pump combination can supply, we move up to the next level. The status of each pump is displayed with our custom characters. Two characters are needed for each pump. This routine takes advantage of the overlaid variables to calculate the correct status character to display.

Figure 37.5: Finished graphic display project built on a BASIC Stamp Activity Board



Our flow sensor is actually an RC circuit and the BS2's RCTIME function is used to return the value. RCTIME is a bit different that the BS1's POT function in that the value is not scaled. The maximum value is dependent upon the components used and the circuit configuration. I ran a small test on my BSAC and found that RCTIME returned a maximum value of 6150. Since I wanted the flow range to span from zero to 1600, I had to multiply the RCTIME value by 0.26. This is accomplished with the */ (star-slash) operator and a value of $0043 (0.26 * 256). Refer to the Feb. '98 issue for details on using the */ operator.

When I checked the code, I found that it worked fine, except that the response was jerky if I moved the potentiometer too quickly. I remember a trick that Scott Edwards described some time ago about simple digital filtering. Basically, you take a portion of the new reading and add it to a portion of the previous reading. After some experimenting, I found that a 60/40 (old/new) ratio and a slight pause between readings gave me a response that seemed realistic.

Once again the */ operator was used to get the proportional values from each variable.

The last thing to do is check our flow for an error condition. If this happens, we transfer to an infinite loop called OvrFlo. The code in this loop will run until the Stamp is reset. A couple of small loops within this routine cause the pumps (error condition) and the word "Error" to flash.

Done. Simple program, not-so-simple task getting it all running, but a very worthwhile exercise in this age of graphical interfaces. Download the code and give it a try. Then change the graphics and roll your own. You'll find that the G12864 is very easy to use, and gives you tremendous versatility in user interface. Figure 37.5 is a photo of the completed project.

For those of you not using the BASIC Stamp Activity Board, connect the RC circuit shown in the BASIC Stamp Programming Manual to pin 7. And for those of you that develop interesting new font pages, please drop me a copy – I'd love to see them.

Beginner's Corner

In the event you haven't heard, Parallax sponsors a listserv for Stamps. The listserv is an excellent resource for Stamp programmers of all levels. Check out the Parallax website for details on subscribing under the "Discussion Group" link.

There have been several recent posts from beginners that have Stamps, but don't know what to do with them, or how to do it. This is not an uncommon problem – particularly for users that don't have a lot of electronics experience. What to do?

Shell out the $80 for Parallax's BASIC Stamp Activity Board. To those of you on a limited budget, the cost probably seems steep, but I can tell you that it will be money well spent. You'd spend much more than that in time and money trying to duplicate its circuitry and ease of use.

The BSAC can be used with the Stamp 1 or Stamp 2 (but not at the same time!), and has switches, LEDs, a potentiometer, a speaker, and RC network for smoothing PWM signals, and a lot more. The BSAC is ideal for testing sub-circuits and code fragments (a demo disk is included). It even comes with a "wall-wart" power supply so you won't go through too many batteries experimenting. If you want to stop worrying about circuits and start learning to code, get a BSAC.

```
' Program Listing 37.1
' Nuts & Volts: Stamp Applications, March 1998

' ----[ Title ]---------------------------------------------------------
'
' File...... PUMPS.BS2
' Purpose... SEE G12864 Serial Graphics LCD Demo
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' WWW....... http://members.aol.com/jonwms
' Started... 24 JAN 98
' Updated... 15 FEB 98


' ----[ Program Description ]-------------------------------------------
'
' This program is a very simplistic pumping station demonstration that
' takes advantage of the graphics capabilities of the Scott Edwards
' Electronics G12864 Serial Graphics LCD. "Flow" demand is sensed by the
' BS2 (potentiometer input) and converted to a group of pumps. The flow
' and pump status is displayed on the G12864.
'
' Two custom screens are downloaded to the G12864: the pumping station
' graphic and an updated fonts screen. Graphics animation is accomplished
' by adding custom characters to the second fonts page and combining the
' custom characters with the pump station graphic.
'
' G12864 Configuration Switch Settings:
' 1 : Off      (Run)
' 2 : On       (9600 baud)
' 3 : On       (BL On)
' 4 : Off      (Esc)
' 5 : Off      (Protect EE - must be On to download custom graphics)
' 6 : On       (Screen 2)


' ----[ Revision History ]----------------------------------------------
'
' 25 JAN 98 : Version 1 complete
' 15 FEB 98 : Added filtering to smooth sensor input


' ----[ Constants ]-----------------------------------------------------
'
FloSnsr CON    7                    ' RCTIME input for *flow* (on BSAC)
HiFlow  CON    1520                 ' highest allowable flow

GxLCD   CON    2                    ' serial output on pin 2
N9600   CON    $4054                ' 9600-bps output
FloPos  CON    64+25                ' Print flow at position 25
```

```
' G12864 codes
ClrLCD  CON     12                      ' Clear LCD text screen
PosCmd  CON     16                      ' Position cursor

' Custom character addresses
Pmp0a   CON     128                     ' pump off, left
Pmp0b   CON     129                     ' pump off, right
Pmp1a   CON     130                     ' pump on, left
Pmp1b   CON     131                     ' pump on, right
ErrorA  CON     132                     ' pump with X, left
ErrorB  CON     133                     ' pump with X, right


' ----[ Variables ]------------------------------------------------------
'
rawFlow VAR     WORD                    ' raw flow from *sensor* (RCTIME)
oldFlow VAR     WORD                    ' last flow reading
newFlow VAR     WORD                    ' new flow reading
fCheck  VAR     BYTE                    ' overflow check
pumps   VAR     NIB                     ' pump status
pump1   VAR     pumps.Bit0
pump2   VAR     pumps.Bit1
pump3   VAR     pumps.Bit2
pump4   VAR     pumps.Bit3

x       VAR     BYTE                    ' loop counter

' ----[ EEPROM Data ]----------------------------------------------------
'

' ----[ Initialization ]-------------------------------------------------
'
Init:   PAUSE 1000                      ' let the G12864 initialize
        ' clear text; all pumps off
        SEROUT GxLCD,N9600,[ClrLCD]
        newFlow = 0
        GOSUB ShoFlo


' ----[ Main Code ]------------------------------------------------------
'
Main:   HIGH FloSnsr                    ' discharge RC cap
        PAUSE 5
        RCTIME FloSnsr,1,rawFlow        ' read the sensor
        rawFlow = rawFlow */ $0043      ' scale to 0 - 1600 (approx)
                                        ' (rawFlow * 0.26)

        ' filter by combining in 60/40 (old/raw) ratio
        newFlow = (oldFlow */ $009A)+(rawFlow */ $0066)
```

```
        IF newFlow > HiFlow THEN OvrFlo ' flow is too high -- shut down
        GOSUB ShoFlo                    ' update the display

        oldFlow = newFlow               ' save last flow reading
        PAUSE 500                       ' delay between readings
        GOTO Main                       ' do it all again


' ----[ Subroutines ]------------------------------------------------------
'
ShoFlo: ' show flow
        SEROUT GxLCD,N9600,[PosCmd,FloPos,DEC4 newFlow]

        ' calculate pumps
        pumps = newFlow / 100
        fCheck = newFlow // 100
        IF fCheck < 25 THEN ShPmps    ' if over by 25+, inc pump level
        pumps = pumps + 1

        ' show pumps
ShPmps: SEROUT GxLCD,N9600,[PosCmd, 65,(Pmp0a+(2*pump1)),(Pmp0b+(2*pump1))]
        SEROUT GxLCD,N9600,[PosCmd, 81,(Pmp0a+(2*pump2)),(Pmp0b+(2*pump2))]
        SEROUT GxLCD,N9600,[PosCmd, 97,(Pmp0a+(2*pump3)),(Pmp0b+(2*pump3))]
        SEROUT GxLCD,N9600,[PosCmd,113,(Pmp0a+(2*pump4)),(Pmp0b+(2*pump4))]
        RETURN


OvrFlo: pumps = %0000
        SEROUT GxLCD,N9600,[PosCmd,FloPos," Error"]
        FOR x = 1 TO 49 STEP 16
          SEROUT GxLCD,N9600,[PosCmd,(64+x),Pmp0a,Pmp0b]
        NEXT
        PAUSE 500
        SEROUT GxLCD,N9600,[PosCmd,FloPos, "      "]
        FOR x = 1 TO 49 STEP 16
          SEROUT GxLCD,N9600,[PosCmd,(64+x),ErrorA,ErrorB]
        NEXT
        PAUSE 500
        GOTO OvrFlo
```