



Column #47, March 1999 by Lon Glazner:

A Powerful Graphic Liquid Crystal Display

There's nothing quite like the beauty of a graphic LCD. Even the simplest electronic design can become a wonder of modern engineering when a "cool" graphic floats across a back-lit graphic LCD.

I remember spending hours-upon-hours working for a client to develop a pretty in-depth communication protocol. In the end, I was firm in my belief that the client would heap showers of praise upon my head. After all, I had solved some tricky problems in a short period of time, and with some pretty innovative techniques. When I showed up and delivered the design, my expectations were summarily dashed. It turned out that another consultant had been developing a user-interface based on a graphic LCD for the same design. And you should have seen this thing! Big, bright, and beautiful. I mean, sure, my design was still an integral part of the whole system, but the LCD interacted with the customer, it was beautiful, and it received ample praise.

So what's the point of this experience? I believe that any time you have an opportunity to interface an electronic design with the end user, looks count for something. All the design work in the world, accompanied with flow charts and schematics, may count for less than a good visual display. This is especially true if you are presenting your design to a non-

Column #47: A Powerful Liquid Crystal Display

technical group such as a marketing department. Don't get me wrong, I'm not saying that a nifty display replaces good engineering. I'm saying sometimes a nifty display IS good engineering.

So what's the best way to create a graphic LCD solution that functions with your BASIC Stamp design? Generally speaking, graphic LCDs require a lot of memory, and quite a few I/O lines. This is not good news in the Stamp world.

Luckily, Scott Edwards Electronics, Inc., has made the graphic LCD interface astoundingly easy. Most BASIC Stamp users are familiar with the Scott Edwards Electronics, Inc, serial LCD. One of Scott's newer products is the G12032 Mini Serial LCD. This is the LCD that I'm going to use in this design. This graphic LCD is similar in many ways to Scott Edward's other popular serial LCDs. One of the really exciting features of the G12032 LCD is the software provided for use with it. You can use this software to load bit maps from your PC into the G12032's non-volatile memory. We will use this same software to load bit maps from your PC into the RAMPack B, and then transfer the bit maps to the G12032 LCD. Each bit map represents a screen to be displayed on the graphic LCD.

There are quite a few features of the G12032 LCD that we will not be using in this design. These features include the previously mentioned on-board non-volatile memory for storing up to eight screens, multiple font sizes, and a large number of built-in commands. I would recommend downloading and reading the G12032 LCD data sheet from "www.seetron.com."

While the G12032 LCD does have non-volatile memory for storing display screens, you may wish to store more than the eight screens possible with the on-board memory. For simplicity, we will use the RAMPack B (RPB) serial RAM module to provide the extra memory. This device — by Solutions Cubed — can be used to store up to 17 graphic display screens for use with the G12032 LCD. If you were to upgrade the RPB to 32Kx8 RAM, then 67 screens could be stored. These devices work particularly well together because they both make use of serial communication. With a BASIC Stamp 2 (BS2) to act as a traffic director, we will download bit maps from the PC into the RPB, and then use the RPB to send blocks of data to the G12032 LCD for display.

Problem Statement

The goal of this design is to provide a simple graphic LCD display for a BS2. A minimum number of I/O pins will be used to interface to the LCD display and provide memory for the storage of graphics. The graphic files to be displayed will be generated in Microsoft Paint, and stored in the RPB. A DIP switch will be used to simulate user inputs. While all of this is going on, the BS2 will be used to control data flow, and monitor the user input. Some simple circuitry is required to convert signal levels between all of the modules in this design. This will be the first task of this design.

Interface Circuitry

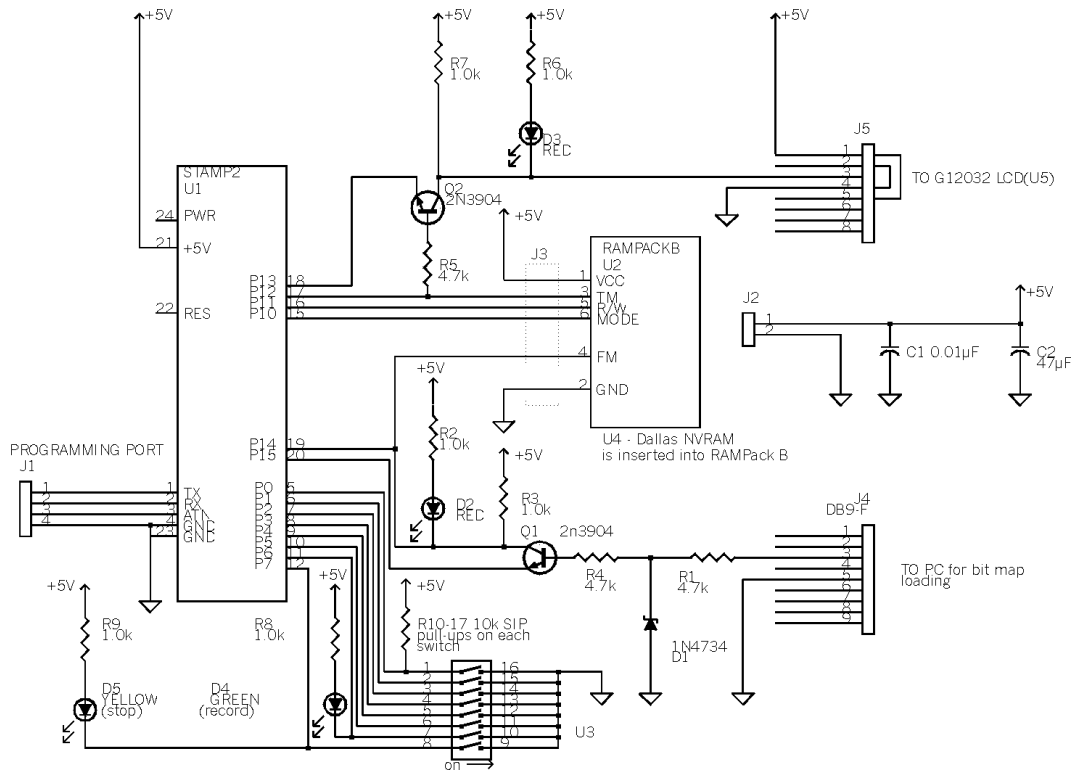
Several data format conversions are required to allow these modules to communicate freely. The BS2 must have access to the serial communication lines to send configuration data and commands to both the RPB and the G12032 LCD.

The RPB needs to share its “From Master” (FM) line with the BS2, and the serial data source. In this case, the serial data source is a PC, and the data is the actual bit map to be stored in the RPB. On top of these requirements, the serial data from the PC is received in RS232 (+12Vdc = logic low, and -12Vdc = logic high) format, and must be converted to a standard logic level (0Vdc = low, 5Vdc = high) format for the RPB. This is easily done via Q1, R4, R1, and D1 seen in Figure 47.1.

P15 (FMEN in the software listing) of the BS2 must be low (0Vdc) to enable data to travel from the PC to the RPB. Additionally, P14 (FMBS2) should be made an input. With P15 (FMEN) low and P14 (FMBS2) set as an input, any data from the PC will be received in roughly TTL format at the FM pin of the RPB. If the BS2 needs to send serial commands to the RPB, it simply sets P15 (FMEN) high, which effectively removes Q1 from the circuit. With P15 (FMBS2) set high, commands can be sent to the RPB via P14 (FMBS2) in standard TTL format. While the description may seem somewhat complicated, this method of line sharing and format converting is quite simple and works well.

A similar technique of line sharing is used to send data from the RPB to the G12032 LCD. Again, the G12032 LCD requires data in an inverted format. In other words, 0Vdc is regarded as a logic high. Since the RPB handles data in a non-inverted format, Q2 is used to invert the data headed to the G12032 LCD. At some points in the BS2 program, the BS2 will require information from the RPB on the “To Master” (TM) line. At these times, you do not want that information to be passed to G12032 LCD.

Figure 47.1: Interface Circuitry



To remove the G12032 LCD from the communication path, the BS2 must set the P13 (TMEN) high. This effectively removes Q2 from the circuit. If the BS2 needs to send data to the G12032, it can send data via P12 (TMBS2) as long as the RPB is not attempting to send data. If the RPB is not sending data its TM pin is configured as an input, and pulled high with a resistor. This allows the TM line to be shared.

LEDs D2 and D3 could be omitted from the interface circuit. I've found that visual indicators of program flow can be extremely helpful for troubleshooting electronic designs. Whenever possible, and particularly with prototypes, I try to include some form of visual indicator. These indicators can be removed upon completion of the design to limit component count and slightly reduce manufacturing costs.

Table 47.1 helps to show the state that each individual control line should be in to allow data to follow the desired path in the communication circuit. Additional control lines for the FIFO mode in the RPB should also be addressed, and are discussed in the code examples. You may also review the February '99 Stamp Applications article "Storing Data With The RAMPack B," for a more in-depth discussion of using the FIFO buffer mode in the RPB.

Figure 47.1: Communication Direction Truth Table

<u>Comm. Direction</u>	<u>P15 (FMEN)</u>	<u>P14 (FMBS2)</u>	<u>P13 (TMEN)</u>	<u>P12 (TMBS2)</u>
PC to RAMPack	low	input	n/a	n/a
BS2 to RAMPack	high	output	n/a	n/a
RAMPack to BS2	high	output-high	high	input
RAMPack to G12032	high	output-high	low	input
BS2 to G12032	high	output-high	low	output

System Functional Description

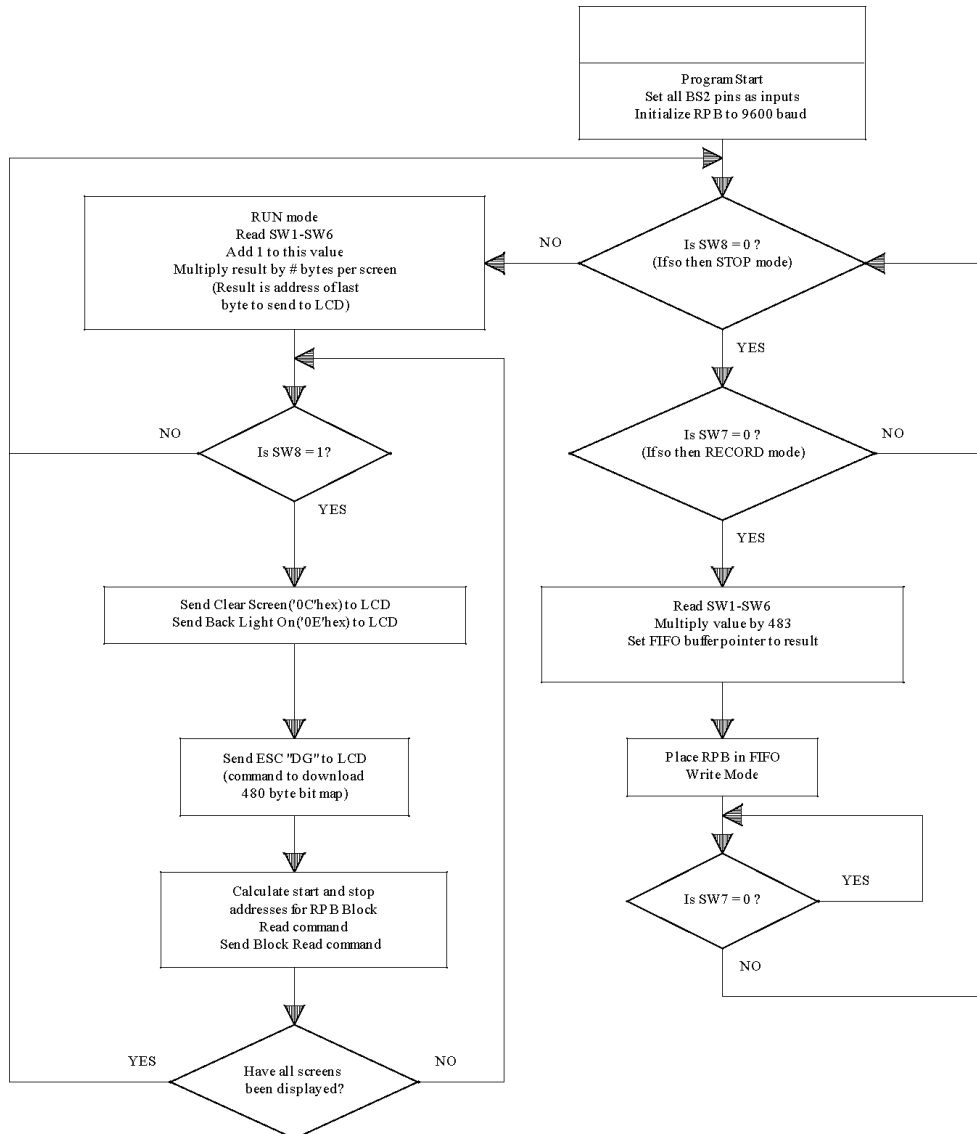
With the interface circuitry defined, we can now begin to ponder the general system functionality. From the start, we knew that this design would store bit maps received from the PC in the RPB. This same data was to be transferred to the G12032 LCD for display. But some user interface needs to be implemented into the design to allow for ease of use.

Let's define three modes of operation in this design: RUN, RECORD, and STOP. In RUN mode, the BS2 will send blocks of data, each block representing a graphic screen, to the G12032 LCD. RECORD mode will be used when data from a PC is to be stored in the RPB. Data will be stored as blocks, with each block representing a bit map or screen. The STOP mode will be used to exit RUN mode and enter RECORD mode.

The BS2 will be interfaced to an eight-switch DIP switch. The switch will connect to P0-P7 of the BS2 as shown in Figure 47.1. Switches 7 and 8 will be used to select between RUN, RECORD, and STOP modes, while switches 1-6 will be used to select addresses to start recording at, or the number of screens to display, depending upon which mode switches 7 and 8 are set for.

Figure 47.2 displays a functional flow chart that the BS2 program is based on. It's appropriate here to describe how the memory is parsed from the RPB into the G12032 LCD. Each bit map consists of 480 bytes of data. There are three additional bytes of data associated with each bit map sent to the RPB. This means that for every bit map or screen downloaded to the RPB, 483 bytes will be stored.

Figure 47.2: System Functional Flow Chart



The additional three data bytes, consisting of ESC, “D,” and “G,” are generated by the software that Scott Edwards Electronics provides to send bit maps to the G12032 LCD. They represent a command required to download a 480-byte bit map to the G12032 LCD. Also specific to this command is a 200ms pause period that should occur between sending ESC “DG” command and the sending of the 480 bytes of bit map data. With the RPB, we will be using the Block Read command, as well as implementing FIFO Write operations.

It’s not possible to have the RPB pause during a Block Read. For that reason, the RPB can not implement the command required by the G12032 LCD appropriately. So, here we are faced with a problem. If the RPB can not send the appropriate command, how can it be implemented? And luckily the solution is readily available. The RPB TM pin is a “pseudo open-collector” configuration. By this, I mean that when it is not transmitting a digital low (0Vdc), it is configured as an input and pulled high by a 1K ohm resistor. For this reason, the BS2 P12 (TMBS2) can act as an input to receive data from the RPB, and as an output to send data to the G12032 LCD.

The solution to our problem is realizable due to the “pseudo open-collector” TM pin and the interface circuitry utilized in this design. While in the RUN mode, the BS2 sends the ESC “DG” command prior to each bit map screen being sent by the RPB. These command bytes are sent via P12 (TMBS2) of the BS2. After a 250ms pause, the RPB Block Read command is requested by the BS2. The RPB then sends the 480 data bytes for each bit map stored to the G12032 LCD. A simple algorithm prevents the ESC “DG” that is stored in the RPB from being re-sent to the G12032 LCD during each Block Read. The only serious concern here is that the RPB can not be sending data to the BS2 or G12032 LCD while the BS2 is sending data to the G12032 LCD. By adding delays of appropriate length to the BS2 software, this can be assured.

A little discussion of the RPB Block Read command and FIFO mode may shed some light on how this whole design is implemented. When RECORD mode is entered, the BS2 places the RPB in FIFO Write mode. In this mode, all serial data received is stored sequentially in RAM. Bit maps are sent as a three-byte command, ESC “DG,” followed by the 480-byte bit map image. How these bit maps are generated is discussed in the next section. Suffice it to say that for every bit map downloaded to the RPB, 483 bytes of data will be stored. With a Block Read command, you can request a block of data to be returned from the RPB. In our case, we are always requesting blocks that are 480 bytes in length. Each block represents a bit map or screen of the LCD.

Downloading Bit Maps to the RPB

There are several steps to take prior to downloading a bit map from your PC to the RPB. For this implementation, we'll use Microsoft Paint. The first thing you'll want to do is set the parameters of the program to the dimensions of the G12032 LCD. In Microsoft Paint, under the Image pull-down menu, select Attributes. Select pixels (or pels) from the units selection. Then select black and white from the colors selection. When this is done, you can set the width of your bit map to 120, and the height to 32. I actually used clip art and sized it in another graphics program. This was then pasted into Paint for touching up.

Figure 47.3: Examples of Bitmaps

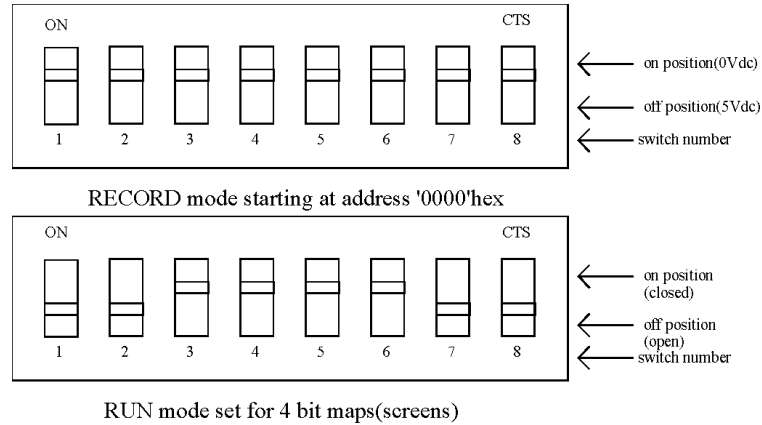


Figure 47.3 shows two examples of the kind of graphics that can be designed. These graphics took less than five minutes to complete and very few BS2 resources to display. One last step is required to store the bit maps in the RPB. You should download the file “dragdropbmp.zip” from “www.seetron.com” in the downloads directory. After unzipping this file set, you will find executable files named “Bmp2com1.exe” through “Bmp2com4.exe.” Simply drag the file that relates to the communication port you use for serial data transfer onto your desktop. For instance, I use communication port 2 for testing so I moved “Bmp2com2.exe” onto my desktop. This creates a shortcut to the file that you just moved.

At this point, you can place the system into RECORD mode. To do this, first place the system in STOP mode by setting switch 8 in the “on” position (0Vdc should be present at P7). Next, set switches 1-6 to the screen location that you want to record to. For simplicity, let's set ours to address zero. Next select RECORD mode by placing switch 7 into the “on” position (0Vdc should be present at P6). The RPB should now be in FIFO mode, and will be recording data.

See Figure 47.4 for DIP switch settings. If you use the schematic in Figure 47.1, the STOP mode is indicated by a lit yellow LED. The RECORD mode is indicated by both the yellow and green LEDs being lit.

Figure 47.4: DIP switch settings



To use the “Bmp2com*.exe” program, simply select your bit map from Windows Explorer and drag it onto the shortcut that you just created. A DOS window will open and tell you if the file was transferred correctly. When asked if you would like to save this image to EEPROM, answer no. Then close the DOS window. If the system is left in RECORD mode, you may continue to load bit maps by repeating the previous steps. Each bit map will be stored one after another in RAM by the RPB.

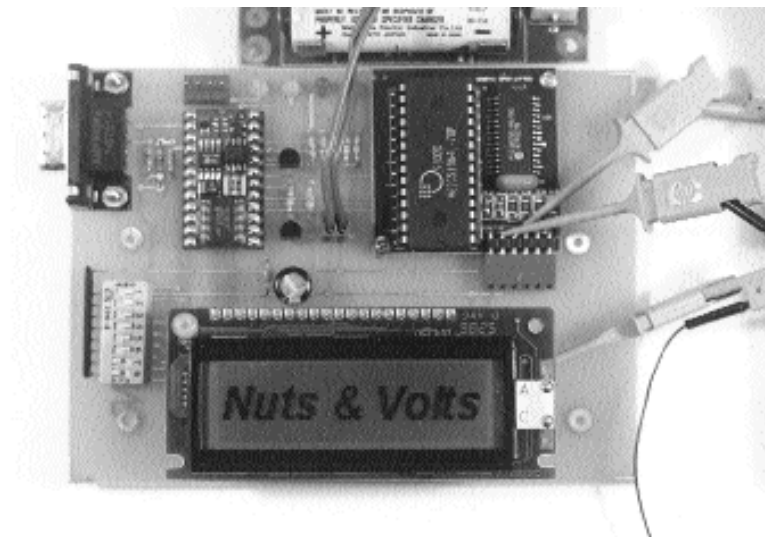
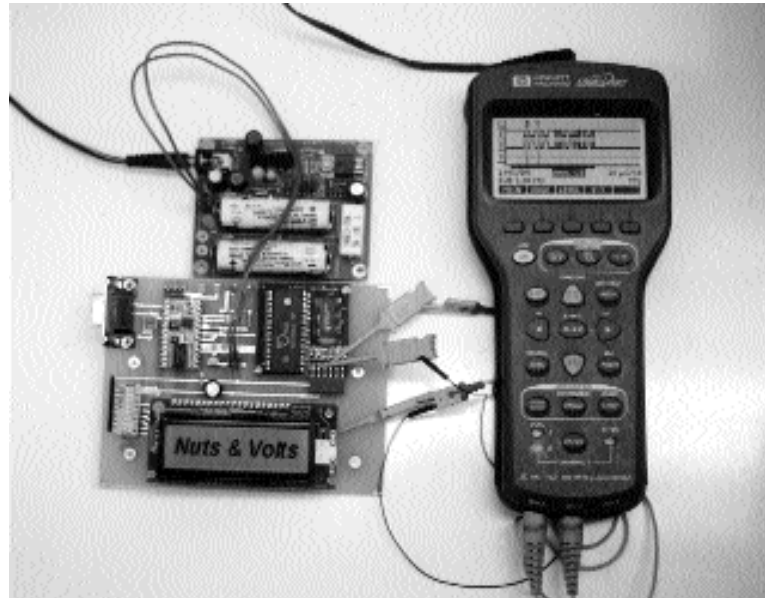
To enter RUN mode, place the system back into STOP mode and select the number of screens you want to display with DIP switches 1 through 6. Finally, exit STOP mode and the system should begin cycling through the graphics that are stored in the RPB. When selecting either the screen location to record at, or the number of screens to display, DIP switches 1 through 6 should be read in binary format and one should be added to the result. DIP switch 1 is the least significant bit. For instance, Figure 47.4 depicts RUN mode with the DIP switch settings for switches 1 through 6 at ‘000011’binary. This results in screens 1, 2, 3, and 4 being displayed.

Column #47: A Powerful Liquid Crystal Display

Figure 47.5: Bill of Materials for Graphic LCD Interface Design

<u>Number</u>	<u>Description</u>	<u>Part</u>	<u>Manufacturer</u>	<u>Distrib.</u>
C1	ceramic capacitor	0.01uF	Panasonic	Digi-Key
C2	electrolytic capacitor	47uF @35V	Panasonic	Digi-Key
D1	5.6V zener diode	1N4734	Lite-On/Vishay	Digi-Key
D2	red LED	P363-ND	Panasonic	Digi-Key
D3	red LED	P363-ND	Panasonic	Digi-Key
D4	green LED	P364-ND	Panasonic	Digi-Key
D5	yellow LED	P365-ND	Panasonic	Digi-Key
J1	BS2 program port	WM4202-ND	Waldom	Digi-Key
J2	+5V posts	WM4200-ND	Waldom	Digi-Key
J3	RPB socket	WM3004-ND	Waldom	Digi-Key
J4	DB9-Female	A2100-ND	Waldom	Digi-Key
J5	top entry 0.1" female	WM3206-ND	Waldom	Digi-Key
Q1	NPN transistor	2N3904	National	Digi-Key
Q2	NPN transistor	2N3904	National	Digi-Key
R1	resistor 1/8W 5%	4.7K ohm	Yageo	Digi-Key
R2	resistor 1/8W 5%	1.0K ohm	Yageo	Digi-Key
R3	resistor 1/8W 5%	1.0K ohm	Yageo	Digi-Key
R4	resistor 1/8W 5%	4.7K ohm	Yageo	Digi-Key
R5	resistor 1/8W 5%	4.7K ohm	Yageo	Digi-Key
R6	resistor 1/8W 5%	1.0K ohm	Yageo	Digi-Key
R7	resistor 1/8W 5%	1.0K ohm	Yageo	Digi-Key
R8	resistor 1/8W 5%	1.0K ohm	Yageo	Digi-Key
R9	resistor 1/8W 5%	1.0K ohm	Yageo	Digi-Key
R10	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
R11	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
R12	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
R13	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
R14	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
R15	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
R16	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
R17	1/9 of 10K SIP resistor	1/9 of Q9103-ND	Panasonic	Digi-Key
U1	BASIC Stamp 2	BASIC Stamp 2	Parallax	Digi-Key
U2	RPB connect to J3	RAMPack B	Solutions Cubed	Digi-Key
U3	8 switch DIP switch	CT1948MST-ND	CTS	Digi-Key
U4	8K NVRAM (optional)	DS1225	Dallas Semi.	Newark
U5	Graphic LCD module	G12032	Scott Ed. Elec.	

Figure 47.6: Finished Project



Column #47: A Powerful Liquid Crystal Display

In Closing

This design example is just one method of interfacing to the G12032 LCD. Because the G12032 LCD has on-board EEPROM, the use of the RPB as a serial data buffer may be overkill. Then again, if you want to do some limited display animation, or have a requirement for more than eight screens in your graphic display, you may want to take a closer look at this design. Furthermore, this design can be a road map to building all sorts of useful serial data buffers. Storing Global Positioning System (GPS) data, or receiving long serial data strings, can be easily accomplished with a BS2 and an RPB. The RPB comes with static RAM, so in some designs you might consider replacing the static RAM with non-volatile RAM. A non-volatile RAM part is listed as U4 in the bill of materials (BOM) for your reference.

All told, this design utilized only 10 bytes of RAM in the BS2, and six I/O lines to interface to the G12032 LCD and the RPB. Additional I/O lines were used to interface to a DIP switch, but these could be omitted based on other design requirements. There was also a minimal use of program space required to implement this design. All of these factors make room for modifications that could be useful in applications other than the one presented here.

There are plenty of real-world applications for this kind of system. Imagine a restaurant with a graphic LCD on every table. The display could cycle through meal specials, show the soup-of-the-day, and even advertise other local establishments. With a little more design work, the owner could update all of the table top displays through a single PC, on a weekly basis.

For your average BASIC Stamp guru, a pretty cool data logger and serial printer interface could be hammered out of this design. Well, after all is said and done, I hope some of you Stampers found at least a little something new here.

A Gerber file for PCB generation is included on the CD-ROM. This might be useful to any of you out there who might want to build this graphic LCD bit map recorder.

Column #47: A Powerful Liquid Crystal Display

```
'Program Listing 47.1 - LCD_399.BS2 Graphic LCD Code
'LCD_399.BS2 - This program is used to control a BS2 based
'device to record bit maps to the RAMPack B and then relay
'them to a Scott Edwards Electronics G12032 Graphics LCD.
'
'Variables used for RECORD mode
RecordAddress VAR Word 'Address to start recording at
RecHigh VAR RecordAddress.highbyte
RecLow VAR RecordAddress.lowbyte
StopAddress VAR Word 'Address where recording stopped
StopHigh VAR StopAddress.highbyte
StopLow VAR StopAddress.lowbyte
'
'Variables used for RUN mode
DisplayEnd VAR Word 'Final display address
DisEndHigh VAR DisplayEnd.highbyte
DisEndLow VAR DisplayEnd.lowbyte
BlockStart VAR Word 'Start address for current screen
StartHigh VAR BlockStart.highbyte
StartLow VAR BlockStart.lowbyte
BlockEnd VAR Word 'End address for current screen
EndHigh VAR BlockEnd.highbyte
EndLow VAR BlockEnd.lowbyte
'
'G12032 Graphic LCD constants
ClrLCD CON 12 'Clear LCD command
BLiteOn CON 14 'Turn on backlight LCD command
'
'I/O pin lables
FMEN CON 15 'FM enable
FMBS2 CON 14 'FM on BS2
TMEN CON 13 'TM enable
TMBS2 CON 12 'TM on BS2
R_W CON 11 'RPB read/write pin
MODE CON 10 'RPB FIFO mode select pin
'
'*****
Initialize:
    PAUSE 1000 'Allow time for power up
    HIGH MODE 'Normal operation mode
    HIGH R_W 'Default to Read mode
    HIGH FMEN 'Disable outside serial data
    HIGH TMEN 'Disable serial data to LCD
    PAUSE 100
    GOTO MainProgram
'*****
'RECORD MODE: The record subroutine records data from the serial
'input device in the RAMPack B. The DIP switch settings for switches
'1-6 determine where the recording starts at in RAM. While in
'record mode the RAMPack B is placed in FIFO mode. This causes all
```

Column #47: A Powerful Liquid Crystal Display

```
'data sent in 8N1 9600 baud format to be stored sequentially in RAM.
'Mutiple screens can be stored in RAM with the software for storing
'bit maps at the "www.seetron.com" downloads website. Debug
'statements are used to indicate the start location of the recording
'and the final location recorded to.
'
RecordMode:
    RecordAddress = INL          'Find start address from DIP switch
    RecordAddress = RecordAddress&%00111111
    RecordAddress = RecordAddress * 483
    Debug "StartAddr = ", DEC RecordAddress,cr
    Debug "recording",cr
    SEROUT FMBS2,84,[$55,$05,RecHigh,RecLow]
    INPUT  FMBS2
    LOW    FMEN                'Enable outside serial data
    LOW    R_W                 'Place FIFO in write mode
    LOW    M0DE                'Place RPB in FIFO mode
KeepRecording:
    If IN6 = 0 Then KeepRecording 'Stay in FIFO until record ends
    HIGH   M0DE                'Exit FIFO mode
    HIGH   R_W                 'Default to FIFO Read mode
    HIGH   FMEN                'Disable outside serial data
    PAUSE  10
    SEROUT FMBS2,84,[$55,$07]   'Read end of FIFO pointer
    SERIN  TMBS2,84,[StopHigh,StopLow]
    Debug "StopAddr = ", DEC StopAddress,cr
    RETURN
'*****
'STOP MODE: The stop subroutine checks to see if the RECORD mode
'is requested. If it is not desired then this routine returns to
'the main menu.
'
StopMode:
    If NOT IN6 = 0 Then NoRecordMode
    GOSUB RecordMode
NoRecordMode:
    RETURN
'*****
'RUN MODE: The run subroutine makes use of the DIP switches 1-6
'in a manner similar to the RECORD mode. The DIP switches are read
'and from this value the number of screens to display is determined.
'The bit map data is sent to the G12032 LCD in 480 byte blocks.
'When the DisplayEnd variable is equal to the EndBlock variable
'then the RUN routine is exited. In between sending screens to the
'LCD this routine checks for a STOP Mode by testing switch 8. If
'a stop is requested then this routine is exited immediately.
'
RunMode:
    DisplayEnd = INL          'Read DIP switch
    DisplayEnd = DisplayEnd&$003F 'zero all but switches 1-6
    DisplayEnd = DisplayEnd + 1 '# screens = switch value + 1
```

Column #47: A Powerful Liquid Crystal Display

```
    DisplayEnd = DisplayEnd * 483 'Final address = # screens * 483
    BlockStart = 0
    LOW      TMEN                  'Enable serial data to LCD
FrameDelay:
    PAUSE    700                  'Allow time for data to get to LCD
    If IN7 = 1 Then ContinueFrame 'Test for STOP mode
    RETURN
ContinueFrame:
    SEROUT   TMBS2,84,[ClrLCD,BLiteOn]
    SEROUT   TMBS2,84,[$1B,"DG"]    'Send commands to G12032 LCD
    INPUT    TMBS2
    PAUSE    250
    BlockEnd = BlockStart + 483    'Find end of block address
    BlockStart = BlockStart + 3    'Bypass stored command bytes
    SEROUT   FMBS2,84,[$55,$02,StartHigh,StartLow,EndHigh,EndLow]
    BlockStart = BlockEnd          'Update start address
    If BlockStart = DisplayEnd then DisplayDone
    GOTO FrameDelay
DisplayDone:
    PAUSE    550                  'Allow time for data to get to LCD
    HIGH     TMEN                  'Disable serial data to LCD
    RETURN
*****
MainProgram:
    If NOT IN7 = 0 Then NoStopMode 'Test for STOP mode
    GOSUB    StopMode
    goto     MainProgram
NoStopMode:
    GOSUB    RunMode              'Run display
    goto     MainProgram
```

