AC 2009-779: BIT-MAPPED GRAPHICS ON A BUDGET USING THE FREESCALE S10 MICROCONTROLLER

Christopher Carroll, University of Minnesota, Duluth

CHRISTOPHER R. CARROLL earned academic degrees at Georgia Tech and Caltech. He is Associate Professor of Electrical and Computer Engineering at the University of Minnesota Duluth. His interests are digital systems and microprocessor applications, especially as they relate to educational environments.

Bit-mapped Graphics on a Budget Using the Freescale S10 Microcontroller

Abstract

Graphics displays are handy output devices in microcontroller systems. This paper describes a simple and efficient graphics display using a bit-mapped approach, generated by the Freescale S12 microcontroller, implemented on Wytec's Dragon development board. The display uses an inexpensive CRT monitor or standard television monitor for output, and requires only a few passive components added to the Dragon development board.

This graphics display uses hardware that has been developed earlier, and that has been disclosed and discussed in earlier ASEE papers[1,2]. This paper describes software that implements the bit-mapped graphics output, and presents some lab experiments that use this graphics display, appropriate for assignments in introductory microcontroller classes.

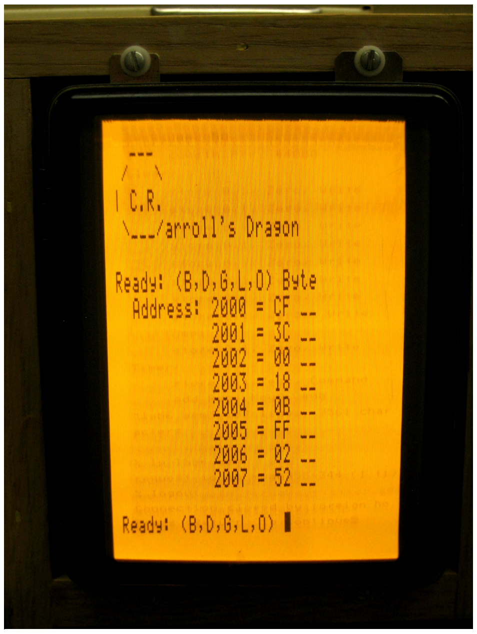
The critical task in generating a bit-mapped graphics display is getting the bits out of memory to the display's video signal fast enough to produce adequate resolution on the display. The Serial Peripheral Interface (SPI) unit on the S12 microcontroller is especially suited to high-speed bit stream output. With the bus clock on the Dragon development board at 24 MHz, the SPI can emit streams of bits from memory at up to 12 megabits per second, resulting in a resolution along a scanline of the display of approximately 600 pixels. The resolution along the axis of the display perpendicular to the scanlines is limited to the number of scanlines in the image, which is 239 scanlines in this design using an inexpensive television-type CRT display. In order to reduce the memory demands on the limited resources of the Dragon board, the bit rate emitted by the SPI is reduced to a quarter of the maximum rate, resulting in a display resolution of 239 x 128 pixels. Although this does not compare well with commercial high-resolution graphics displays, it is certainly useful in an educational environment, and is entirely adequate for teaching students how to use bit-mapped graphics.

This paper describes required software and its interaction with the S12 microcontroller SPI unit to produce a bit-mapped graphics display on a standard television or inexpensive CRT display. The paper also investigates several experiments using this graphics display that are possible in an introductory microcontroller course lab setting.

Lab Station

The lab station on which this bit-mapped graphics display is based has been described in earlier ASEE papers[1,2]. It consists of a Wytec Dragon development board for the Freescale S12 microcontroller[3,4], plus some additional hardware and software to implement an alphanumeric matrix keyboard input device and interface to a standard low-cost CRT monochrome display for output. As described in those earlier papers, the CRT display was used originally just to provide character output for display of alphanumeric characters. Figure 1 shows a typical display produced by the CRT output in the original lab station design.

The lab station as originally implemented allowed each station to function independently, without the need for a dedicated personal computer at each lab station. Instead, the stations emulated alphanumeric terminals and were networked via RS-232 terminal lines to one central multi-user personal computer running Linux. Students developed programs and assembled them on this multi-user computer using the alphanumeric input/output on the lab stations provided by the additions to the Dragon development board, and then downloaded the resulting code into the lab stations for execution and testing on the individual lab stations. This same environment supports new capabilities with the addition of the bit-mapped graphics feature.



```
  ---
 /   \
| C.R.
 \___/arroll's Dragon

Ready: (B,D,G,L,O) Byte
  Address: 2000 = CF __
           2001 = 3C __
           2002 = 00 __
           2003 = 18 __
           2004 = 0B __
           2005 = FF __
           2006 = 02 __
           2007 = 52 __

Ready: (B,D,G,L,O) ▌
```

Figure 1. Original alphanumeric display

In this paper, additional software expands the output capabilities of the CRT display to include bit-mapped graphics. No additional hardware beyond that employed in the original lab station is required. The graphics display provided by this expansion supplies additional features that can be used in lab assignments, and enables students to gain experience working with a bit-mapped graphics display.

Bit-Mapped Graphics

Bit-mapped graphics displays are characterized by a one-to-one correspondence between bits in the microcontroller memory and pixels displayed on the screen. In this case, just one bit of memory is associated with each pixel, providing just a simple on/off control of pixels on the screen by toggling the values of bits in memory. Some bit-mapped displays associate several bits of memory with each pixel, allowing color and/or intensity variation on a pixel-by-pixel basis, but the goal here is low cost and simplicity. Bit-mapped displays require allocation of a block of bits in memory corresponding to the number of pixels on the screen. For large displays, this can be a significant amount of memory, sometimes leading display designers to choose other graphics techniques. However, at the screen resolution described here, the amount of memory required is manageable.

The CRT display in the lab station is oriented with the long dimension vertically. This means that the scanlines composing the image run vertically, with the electron beam starting at the bottom of the screen and sweeping upwards for each scanline. Standard analog-television-type timing is approximated here, with scanlines generated at 15.3 KHz, and 256 scanlines per frame, or full image. A small fraction of the scanlines are "wasted" during the retrace time involved in returning the electron beam to the left side of the screen, leaving in this case 239 scanlines available for producing an image. The screen is refreshed 60 times per second, to match the power line frequency, thus minimizing graphic effects on the display caused by ripple in the CRT power supply. Of the 65 microseconds spent tracing out each scanline, a small fraction is wasted returning the electron beam to the bottom of the screen to prepare for the next scanline, resulting in about 50 microseconds of usable image time per scanline when overscan of the screen is accommodated. Thus, the horizontal (short dimension) resolution of the graphics screen here is limited to 239 pixels, i.e. the number of scanlines in the image. The vertical (long dimension) resolution depends upon how quickly the electron beam can be toggled during each scanline.

The biggest challenge in producing a bit-mapped display on a raster-scanned CRT device is emitting the bits representing pixels from memory to the electron gun in the CRT fast enough to achieve adequate resolution along the scanline. Fortunately, in the S12 microcontroller the Serial Peripheral Interface (SPI) is particularly well-suited to spewing a stream of bits out of the microcontroller at high speed. With a 24 MHz clock running the S12 processor on the Dragon board, the SPI can emit bits at a rate of up to 12 Mbits per second, resulting in about 600 pixels produced during the 50 microsecond image time along a scanline. In order to stay within the constraints of memory size on the Dragon board (12K bytes RAM, minus memory needed for user programs, stack, and data) a design decision was made to limit the bit rate of the SPI to 3 Mbits per second, resulting in a resolution along the scanline of 128 pixels. Thus, the bitmap image produced by the display in this application is composed of an array of 239 x 128 pixels. Figure 2 shows a photograph of a graphics image produced using this system.

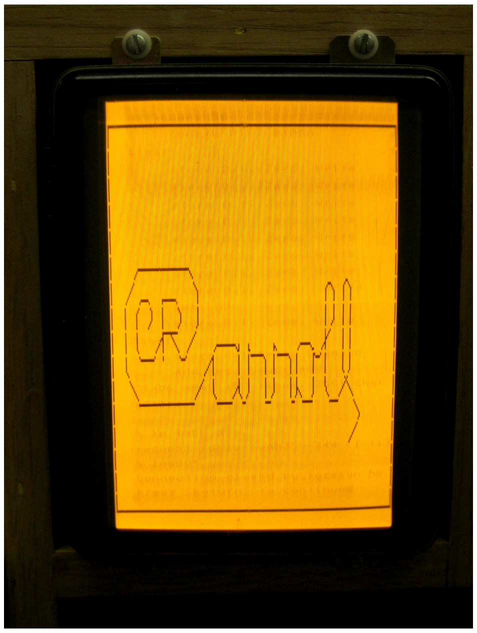


Figure 2. Bit-mapped graphics display

Software

The software for producing the 239 x 128 pixel bit-mapped graphics image on a standard CRT screen is built around two interrupt routines, each caused by an event generated by an output compare unit in the timing section of the S12 microcontroller. The S12 timer/counter is scaled down to count at 3 MHz rather than the full-speed 24 MHz in order to generate interrupts slowly enough to match the required 60 Hz rate of the frame synchronization.

Figure 3 shows the S12 assembly code for the interrupt routine responding to 60 Hz interrupts from output compare 0. This code first establishes the time of the next output compare 0 interrupt (lines 1-3), clears the flag bit for this output compare (line 4), initializes some variables for beginning the display of the bit-map part of memory (lines 5-6), and returns.

```
IntOC0:   ldd  $0050
          addd #$c400
          std  $0050
          bclr $004e,$fe
          clr  SLines
          movw #$2800,DspPtr
          rti
```

Figure 3: 60 Hz interrupt routine

```
IntOc1:   nop
          nop
          ldd  $0052
          addd #$0c4
          std  $0052
          bclr $004e,$fd
          inc  SLines
          ldab SLines
          cmpb #$10
          tpa
          lsra
          lsra
          anda #$01
          staa $0041
          ldaa #$10
Waste:    dbne a,Waste
          bset $0041,$02
          cmpb #$11
          blo  OC1out
          ldaa #$48
OC1X:     dbne a,OC1X
          ldx  #$10
OC1all:   ldy  DspPtr
          ldab 0,y
          iny
          sty  DspPtr
          bitb $00db
          stab $00dd
          ldab #$0e
OC1w3:    dbne b,OC1w3
          nop
          nop
          nop
          nop
          nop
          nop
          nop
          dbne x,OC1all
OC1out:   rti
```

Figure 4. 15.3KHz interrupt routine.

Figure 4 shows the S12 assembly code for the more interesting interrupt generated by output compare 1, which emits 128 bits of memory through the SPI for every scanline on the bit-map display. This interrupt occurs at a rate of 15.3 KHz, creating a scanline on each interrupt. The "nop" instructions throughout the code are present to adjust detailed timing of the routine's execution, and have no functional effect otherwise. The routine begins by establishing the time for the next output compare 1 interrupt and clearing the interrupt flag for the device, just as in Figure 3. The code then counts scanlines, and turns off the CRT's synchronization signals produced by output compares 0 and 1 at appropriate times to generate the scanline sync and frame sync pulses needed by the CRT display. After the 11th (base sixteen) scanline, image generation begins in the OC1all loop, which extends to the end of the interrupt service routine. In that loop, bytes are extracted from memory (identified by DspPtr) and emitted through the SPI by storing to address $00dd. The "bitb" instruction is there just to touch the SPI flag register so that the SPI flag gets cleared before storing the next byte for display. Time is wasted before and after this image generation in order to center the image on the displayed scanlines.

Three signals connect from the Dragon board to the CRT display. One is the video signal, conveying bits from the SPI that control the electron beam in the CRT, eventually producing the pixels of the image. A second signal is the "frame sync" signal produced by the output compare 0 unit at 60 Hz, to cause the side-to-side retrace of the screen in preparation for the next frame. The third signal is the "scanline sync" signal produced by the output compare 1 unit at 15.3 KHz, to cause the vertical retrace of the screen in preparation for the next scanline. These three signals keep the CRT scanning and image generation synchronized with the information being emitted by the SPI through the software in the interrupt service routines.

One defect in the graphics display presented here is caused by the operation of the SPI unit in the S12 microcontroller. After emitting each byte of information serially to the video CRT signal, the SPI needs a short "recovery" time before the next byte begins emerging. This short time is visible as a very short light spot every eight pixels along the scanlines. This spot is smaller than a pixel, and is not too objectionable, but it is visible, especially in a block of pixels that are all turned dark. This small imperfection in the display is the price paid for the overall simplicity of the design.

Lab Experiments

With a bit-mapped graphics display available, a variety of lab experiments comes to mind as possible lab assignments in a typical microcontroller class. These range in complexity from very straightforward tasks to sophisticated designs that exceed the capabilities of most students just learning to program microcontrollers. Graphics applications are a rewarding technique for improving students' programming skills because errors in programming are immediately visible as errors in the graphics produced by the software. This quality of bit-mapped graphics displays makes them particularly instructive to students learning to write software, as the effects of their programming statements are visible in the image produced by the display, providing valuable feedback to the novice programmer. A program's interaction with data in memory is a fundamental part of any program, and a bit-mapped display makes that interaction visible.

Perhaps the easiest application is a simple "etch-a-sketch" program. This program starts with a blank screen, and then just draws a line from a starting point in a direction indicated by a keypress on the alphanumeric keyboard. The image in Figure 2 above was created with such a program. This type of application acquaints students with the one-to-one correspondence between pixels on the screen and bits in memory, and provides a great example that requires dexterity in manipulating individual bits in memory. Debugging such a program is straightforward, as in most graphics programs, because the effect of program behavior (correct or incorrect) is immediately visible on the displayed image. This is an excellent place to start when introducing bit-mapped graphics displays.

Games are another great source of applications for graphics displays. An easy game that is easy to describe to students is the original "pong" game found in arcades years ago. All that is required is the creation of a "ball" (one pixel) and some paddles that can be moved via keyboard input. More interesting games are possible, using some sort of fixed graphic "playfield" and some simple graphic symbols that interact in interesting ways. The sky's the limit in creativity here. The nice thing about bit-mapped graphics is that there is no restriction on what images are presented. It's just a matter of turning on the right bits in memory.

More sophisticated applications might involve positioning text on the display, or even moving text around on the display. Character generator tables are already present in the EPROM software used to create the alphanumeric displays of the lab station (Figure 1) so all that's needed is a way to access those tables, extract the information, and store it into the right bits of memory to make characters appear on the screen. This takes a bit of effort in assembly code, but again, debugging is easy because errors in the code are very visible on the graphics display.

Graphics on the level of complexity with which students are familiar on their personal computers are beyond the reach of this low-resolution display. However, techniques for composing those more advanced graphic images can be explored even within the simplicity of the design described here. Extension to more marketable applications is left to the student!

Pedagogical Impact

A bit-mapped graphics display not only produces a handy output device for application programs, but also provides a tool that enhances the pedagogical effectiveness of laboratory exercises in a microcontroller class. The bit-mapped nature of the display physically shows the programmer the status of bits in memory in the block of memory that is being presented on the screen. Thus, the effect of various programming errors is immediately visible when the application is run. Even when properly constructed, the effect of different algorithms used to access the data can be visible in the displayed image as variations in speed of image generation or through other characteristics of the image. Using a bit-mapped graphics display in an application is appealing because of the glamorous result produced by the student programmer, and students respond by spending more time optimizing their algorithmic approach rather than being satisfied with the first solution that works. From an instructor's point of view it also provides feedback to the programmer that can be instructive in unanticipated ways.

Summary

Described here is a simple but workable bit-mapped graphics display using the commercial Wytec development board and its S12 microcontroller, expanded as described in earlier ASEE papers to include an alphanumeric keyboard and CRT display. The design requires no special hardware beyond that required in the original alphanumeric-based lab station. Only revised software is needed to implement the bit-mapped graphics feature. Having the bit-mapped graphics capability opens the door to a wide variety of possible graphic-based applications that can form the basis for lab experiments of varying complexity. Adding this graphics capability to the lab station expands the options available to lab designers, and provides a rich environment for educating students.

References

1. Carroll, C. R. (2008). Innovative 'HCS12 microcontroller lab station using limited lab resources. *Proceedings of the 2008 North Midwest Section Meeting of ASEE*. Platteville, WI.
2. Carroll, C. R. (2008). Innovative lab station using the Freescale 'HCS12 microcontroller and Dragon development board. *2008 ASEE Annual Conference Proceedings*. Pittsburgh, PA.
3. Cady, F. M. (2008). *Software and hardware engineering: Assembly and C programming for the Freescale HCS12 microcontroller*. Oxford University Press. New York, NY.
4. Pack, D.J. and S.F. Barrett. (2008). *Microcontroller theory and applications: HC12 and S12*. Pearson/Prentice Hall. Upper Saddle River, NJ.