Appendix B4

Signature Generator

SIGNATURE GENERATOR

1. INTRODUCTION

The CL-GD546X incorporates SG (signature generator) logic to test the display memory and video logic. This feature makes it possible to capture a 16-bit signature for any pixel bit position of any video mode and the display memory contents. An error in the display memory, control logic, or pixel data manipulation logic produces a signature different from a known good signature produced with the same test conditions.

This is useful at final test time and allows the test technician to quickly and accurately check a test pattern, to avoid resorting to a time-consuming and error-prone visual inspection of the screen. The SG is used extensively in the manufacturing test.

The SG is captured for a specific bit within every pixel in the frame. There are two fields in the SG Control register that specify the byte within the pixel and the bit within the byte. Two additional bits in the SG Control register serve to reset and enable the SG, respectively.

Note that the signature is a function of the displayed pixels, not just the display data. If the display screen includes blinking attributes or a blinking cursor, the signature is different for those frames when the pixel is blinked off than when the pixel is blinked on. The SG is intended to be used when blinking is disabled.

2. REGISTER SR18: SIGNATURE GENERATOR CONTROL

Register SR18 controls the signature generator and is described in detail in Chapter 6, “Enhanced V-Port™ Registers”. Register SR18 definitions are summarized in Table B4-1.

Table B4-1. Register SR18: Signature Generator Control

SR18 Bits	Use	Note
SR18[7:6]	Byte select	0, 1, 2 only
SR18[5]	Enable data generator	Factory testing only
SR18[4:2]	Bit select	Within byte selected in [7:6]
SR18[1]	SG reset	
SR18[0]	SG enable/status	

Bits 7:6 and 4:2 are used together on the screen to select the bit within each pixel tested. The bit position is eight times SR18[7:6] plus SR[4:2]. The bit position must make sense in the context of the selected video mode. For example, selecting bit 22 for testing in an 8-bpp mode returns useless results.

Table B4-2. Bit Selection for Signature Generator Testing

SR18[4:2]	SR18 [7:6] = 00	SR18 [7:6] = 01	SR18 [7:6] = 10
000b	0	8	16
001b	1	9	17
010b	2	10	18
011b	3	11	19
100b	4	12	20
101b	5	13	21
110b	6	14	22
111b	7	15	23

SR18[1] is programmed to ‘1’ to reset the SG to an initial known state.

SR18[0] is programmed to ‘1’ (simultaneously with SR18[1] being programmed to ‘0’) to start the SG. The SG automatically delays until the next VSYNC and then begins accumulating a signature of the specified bit. The signature is accumulated for one frame and then SR18[0] is set to ‘0’. By monitoring this bit, the program can determine when the signature is complete. When the signature is complete, it can be read from SR19 and SR1A.

3. SAMPLE CODE

The following code example in ‘C’, illustrates the method a programmer might use to capture eight signatures for any 8-bpp screen. It is assumed that the screen is already displayed, and no blinking attributes (in Text mode) are displayed.

```
signature_capture () /* Capture signatures for 8 BPP mode */
  {
 unsigned int result,i,SR19,SR1A;
 unsigned int SIG [8];

for (i = 0;i <= 7; i++) {     /* cycle through eight data bits */
             outp (0x3c4,0x18);         /* point to SR18 */
    outp (0x3c5, (2 | (i<<2)));/* reset the SG* /
    outp (0x3c5, (i << 2));    /* turn off the reset bit */
    outp (0x3c4,0x18);         /* this really is necessary */
    outp (0x3c5, (1 | (i << 2)));  /* and start it up */
    result = inp (0x3c5);          /* read this once here */
     while ((result & 0x01) != 0) {  /* wait until signature is captured */
         outp (0x3c4,0x18);      /* this must be done here */
         result = inp (0x3c5);
         }
    outp (0x3c4,0x19);           /* point to low byte */
    SR19 = inp (0x3c5);          /* and read it */
    outp (0x3c4,0x1A);           /* point to byte */
    SR1A = inp (0x3c5);          /* and read it */
    SIG [i] = (SR1A << 8) + SR19;/* combine the two byte */
    }                            /* end of for... */
 }
```