LCD Memory Management

The T6963C controller for this LCD comes with approximately 8K of memory. It is divided into two user defined parts, one for graphics memory and one for character memory. It also has the option for user defined characters, but this feature is not utilized for this project. The graphics memory has a one to one relationship between set bits and on pixels. The character memory maps characters stored in an internal ROM to the corresponding location of the LCD.

The character width can be set to 8 bits or 6 bits. At 6 bits, characters are only 6 bits wide. However, the 6 bit setting uses significantly more memory because this also affects the graphics memory. On the 6 bit setting, only the lower 6 bits are significant because each graphics block is considered to be six bits long. Since my main concern was fitting as much data as possible on the screen, I chose the 6 bit option because it allows characters to take up less space.

The graphics memory is set to start at 0000h. For a 240 x 128 display, the number of bytes required to represent a single line across the screen in 6 bit mode is 240 / 6 = 40. Multiplying this by 128 lines results in 5120 bytes required for graphics memory. Therefore the character memory must start after this at 1400h. The character memory requires 40 bytes per 8 bit high line, resulting in a memory usage of 320 bytes. The total memory usage is 5440 bytes, which is well below the 8K memory maximum.

0000h	**&#65533;**	0027h
&#65533;	**Graphics**	**&#65533;**
13D8h	**&#65533;**	13FFh
1400h	**&#65533;**	1427h
&#65533;	**Character**	**&#65533;**
1658h		167Fh

Block diagram of LCD controller memory usage

To write the data to memory it is necessary to transform the data format. The initial data format is a series of 128 bit numbers, each of which represents a voltage. This must be transformed to a format where it can be written to memory. The following structure performs the transformation:

```
For mempointer = 13D8h to 167Fh      *loop across bottom of graphics memory
Mempointertmp = mempointer           *setup temporary variable
Voltage = memory(Z+)                 *get value to display
   For bit = 0 to 5                   *loop across 6 bits for each memory location
      For cnt = 0 to voltage
         Mempointertmp += -40         *shift pointer up one row for each voltage tick
      Next cnt
      Write bit to Mempointertmp     *write bit to position at Mempointer
   Next bit                           *ie: write 0th through 5th bits because each address has six bits
Next mempointer
```

This routine writes/erases the wave form data from memory to the LCD display.