Embedding it better...

µTasker Document

- **Segment LCD user's guide**

Table of Contents

1. Introduction

Some processors include an SLCD (Segment LCD) controller. This document discusses the basic operation of these controllers and how the µTasker project supports their use and simulation.

2. SLCD Modules and SLCD Controllers

SLCD modules are simple and very low power LCD displays. They can have a large number of segments in the form of dots (pixels) or symbols (parts of 7 segment characters or special images) and a large number of pins to enable these segments to be controlled.

SLCD controllers tend to have a high number of pins to handle large GLCD modules. These pins can be configured as back-plane or front-plane pins and generate the required multiplexing and waveforms needed to continuously refresh the SLCD.

Since SLCDS are often used in very low power applications the SLCD controllers tend to be able to operate autonomously when the processor is in low power modes (eg. WAIT, SLEEP or STOP modes). Special features such as automatic blink fields may be supported which may also be able to operate autonomously when the processor is not running.

Rather than have one connection for each segment, the segments are typically multiplexed so that less control lines are needed to control the same number of individual segments.

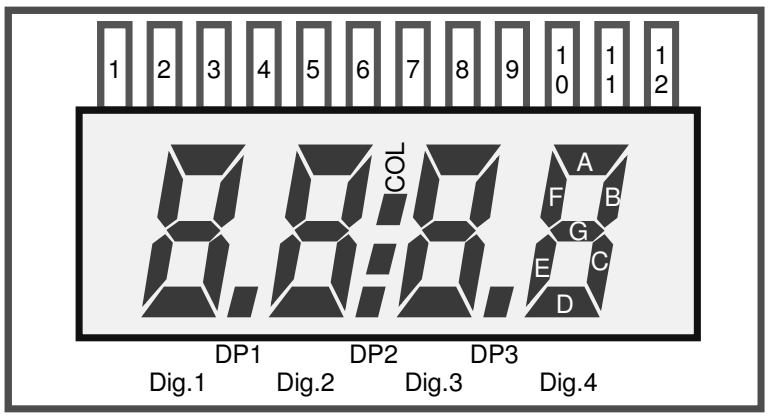
To illustrate this, the Luminex LCD-S401M16KR, as used on the Freescale FRDM-KL46Z is used as reference. The following is an excerpt from its data sheet showing its characteristics and how its 12 pins are connected to its elements:

CHARACTERISTICS	
FLUID TYPE	TN
BACKGROUND COLOR	GRAY
VIEWING DIRECTION	6:00
DISPLAY MODE	REFLECTIVE
OPERATING VOLTAGE	3.0V A.C.
OPERATING TEMP.	0°C TO 50°C
STORAGE TEMP.	−20°C TO 70°C
DRIVING METHOD	1/4 DUTY, 1/3 BIAS
CONNECTION TYPE	12 PINS

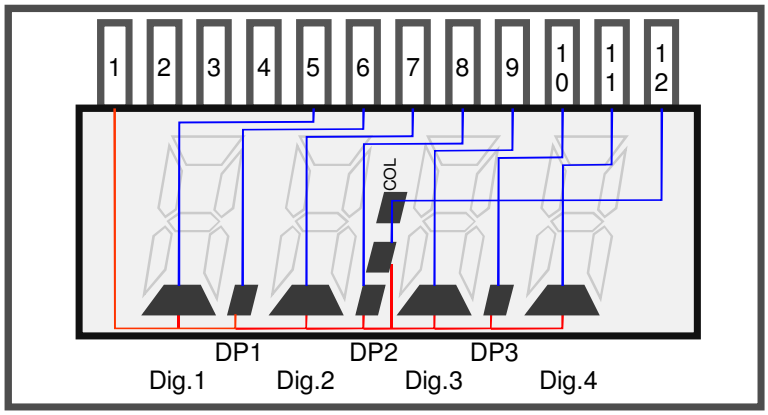
PIN	1	2	3	4	5	6	7
COM0	COM0	/	/	/	1D	DP1	2D
COM1	/	COM1	/	/	1E	1C	2E
COM2	/	/	COM2	/	1G	1B	2G
COM3	/	/	/	COM3	1F	1A	2F
PIN	8	9	10	11	12		
COM0	DP2	3D	DP3	4D	COL		
COM1	2C	3E	3C	4E	4C		
COM2	2B	3G	3B	4G	4B		
COM3	2A	3F	3A	4F	4A		

Characteristics and connection details of Luminex LCD-S401M16KR

And the display's elements are illustrated below according to their references in the connection table in the original data sheet:



This display has 12 pins, which are all connected to a driving pin on the SLCD controller. There are 4 common pins and so the driving mode is ¼ duty, since each of the common pins are driven for ¼ of the cycle period. Each of the common lines is connected to 8 LCD segments. Taking COM0 as an example the 8 segments that are controlled by it are show in the following diagram.



This means that when the COM0 line is being driven up to 8 segments can be turned on, depending on the state of the 8 segment pins. The 8 segment pins (5..12) thus control these 8 segments during this driving phase and can turn all on, all off, or an combination in between.

Since there are 4 phases (4 COM pins), each controlling 8 segments during that phase, a total of 32 segments can be controlled – this corresponds to the number of segments that the SLCD physically has (not that the COL segment is made up of two points which cannot be controlled individually). The other 3 COM drive combinations are not shown in a diagram here but can be easily read from the original data sheet table.

In order to configure the SLCD driver to match this display 4 of its LCD pins are configured as COM drive pins and 8 as segment drive pins. The COM drive pins are configured to drive in different phases (eg. COM0 in phase 1, COM1 in phase 2, COM2 in phase 3 and COM3 in phase 4, where each of the phases are repeated according to ¼ duty driving mode). The duty mode and any other specific configuration are also set in the SLCD controller.

Each segment drive line each can control an individual segment in each of the 4 phases of the operation. Which segment that is depends on the connection in the SLCD itself (see connection table in this example) and how the SLCD pins are physically connected to the SLCD module. The designed can however quite easily make a list of which control bits in the segment drive registers correspond to which segment in the SLCD and then use this information to set text, numbers or patterns required by the application software. The SLCD

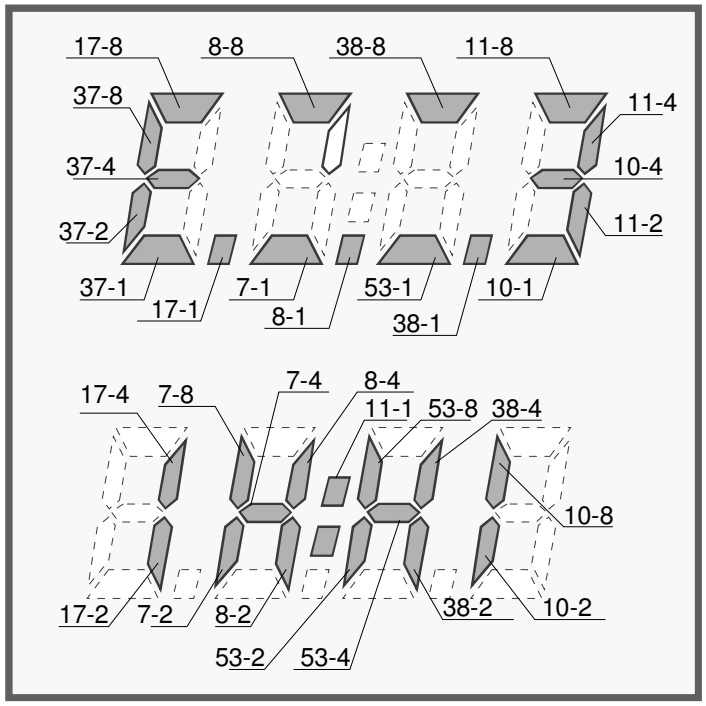
controller performs the multiplexing display driving autonomously based on the register states.

In the case of the Freescale FRDM-KL46Z the SLCD pin connections are as follows:

Pin 1 (COM 0) = LCD 40
Pin 2 (COM 1) = LCD 52
Pin 3 (COM 2) = LCD 19
Pin 4 (COM 3) = LCD 18
Pin 5 (1D/1E/1G/1F) = LCD 37
Pin 6 (DP1/1C/1B/1A) = LCD 17
Pin 7 (2D/2E/2G/2F) = LCD 7
Pin 8 (DP2/2C/2B/2A) = LCD 8
Pin 9 (3D/3E/3G/3F) = LCD 53
Pin 10 (DP3/3C/3B/3A) = LCD 38
Pin 11 (4D/4E/4G/4F) = LCD 10
Pin 12 (COL/4C/4B/4A) = LCD 11

The following chapter shows how an SLCD controller is configured to match this SLCD. The Kinetis SLCD driver is used as example but the principle of the operation should be essentially valid for all types.

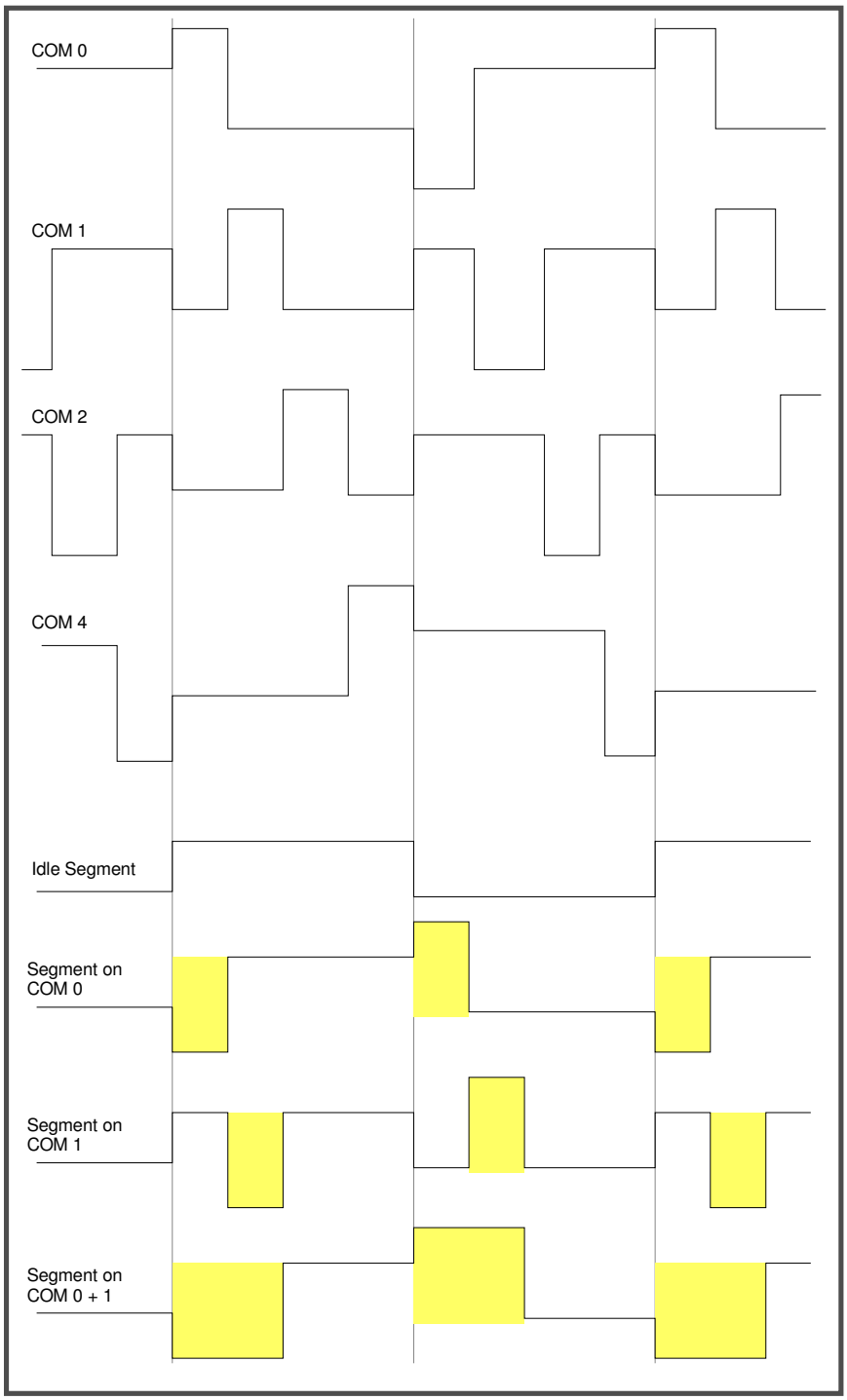
For completeness the complete list of segment control bits for the reference LCD used on the FRDM-KL46Z is given below.



Example: Segment control 8-8 is the bit 0x08 (phase controlled by COM 3) in the register LCD_WF8, which corresponds to the 67th control bit in the control block. To enable it the following can be used: `SET_SLCD(8, 0x08)`, to clear it `CLEAR_SLCD(8, 0x08)` and to toggle it `TOGGLE_SLCD(8, 0x08)`.

The following illustrations show typical drive signals. The SLCD module has no power supply and it is seen that the drive signals have an AC nature. The COM lines drive in one phase of the cycle but phase inverted in each period. The segment drive lines either drive an mid-level signal when no segment is to be enabled or else an out-of-phase signal in the respective COM phase when its corresponding segment is to be on.

The duty cycle period is configurable and typically in the 100Hz region.



3. GLCD Programmer's Model

Due to the fact that SLCD controllers take over all of the active control of SLCD modules the SLCD interface tends to be very simple from a programmer's perspective.

The controller will tend to be disabled at reset and so the programmer needs to configure it to suit the SLCD in use:

- Controller configuration to operate from the correct clock and in the correct mode
- Front-plane and Back-plane signal default configuration

Typically all segments will initially be off and the SLCD will display no segments once the initialisation is complete and then the programmer can individually control segments to indicate states (eg. Segment images or text), or to group together to form numbers (eg. seven segment numbers) or characters (eg. dot matix segments).

3.1. Kinetis FRDM-KL46Z SLCD Configuration Example

The following code configures the SLCD and its pins, as well as setting the driving phase of each COM line. This configuration follows on from the introduction to the specific SLCD module used on this board:

```
// Configure the SLCD mode but don't enable it yet
//
LCD_GCR = (LCD_GCR_RVEN | (0x08000000 & LCD_GCR_RVTRIM_MASK) | LCD_GCR_CPSEL |
LCD_GCR_LADJ_MASK | LCD_GCR_VSUPPLY | LCD_GCR_SOURCE | LCD_GCR_LCLK_1 | LCD_GCR_DUTY_4BP);

// Configure the pins to be used as COM lines
//
LCD_BPENL = (SLCD_PIN_19 | SLCD_PIN_18);
LCD_BPENH = (SLCD_PIN_H_52 | SLCD_PIN_H_40);

// Configure the pins to be used as segment drive lines (plus the COM lines)
//
LCD_PENL = ((SLCD_PIN_17 | SLCD_PIN_11 | SLCD_PIN_10 | SLCD_PIN_8 | SLCD_PIN_7) |
            (SLCD_PIN_19 | SLCD_PIN_18));
LCD_PENH = ((SLCD_PIN_H_53 | SLCD_PIN_H_38 | SLCD_PIN_H_37) | (SLCD_PIN_H_52 |
             SLCD_PIN_H_40));

// Set the 4 COM line phases
//
WRITE_SLCD(43TO40, 0x00000001); // 40 to phase 1 (COM 0)
WRITE_SLCD(55TO52, 0x00000002); // 52 to phase 2 (COM 1)
WRITE_SLCD(19TO16, 0x04080000); // 19 to phase 3 (COM 2) and 18 to phase 3 (COM 3)

// Enable the SLCD operation
//
LCD_GCR = (LCD_GCR_LCDEN | LCD_GCR_RVEN | (0x08000000 & LCD_GCR_RVTRIM_MASK) | LCD_GCR_CPSEL |
LCD_GCR_LADJ_MASK | LCD_GCR_VSUPPLY | LCD_GCR_SOURCE | LCD_GCR_LCLK_1 | LCD_GCR_DUTY_4BP)
```

To turn on and off the segment COL the following two command enable and disable the control on the SLCD module's pin 12 (LCD 11)

```
SET_SLCD(11TO8, 0x01000000);
CLEAR_SLCD(11TO8, 0x01000000);
```

The equivalent control of the segment 4C (also controlled by the same pin) shows that the phase that the output is enable in makes the difference.

```
SET_SLCD(11TO8, 0x02000000);
CLEAR_SLCD(11TO8, 0x02000000);
```

If both of these segments are to be enabled the output is simply drive in more than one of the phases

```
SET_SLCD(11TO8, 0x03000000);
```

Obviously the maximum number of segments that can be controlled by this pin is 4, where the following would enable COL, 4C, 4B and 4A – each in their corresponding drive phase [0x1 is phase 1, 0x02 is phase 2, 0x04 is phase 3 and 0x08 is phase 4].

```
SET_SLCD(11TO8, 0x03000000);
```

3.2. Kinetis K40 GLCD Configuration Example

The Kinetis K40 contains an SLCD controller and is used as illustration of the initialization and control in the µTasker project.

Two SLCD types are discussed:

- 28 segment SLCD on the *TWRPI-SLCD* module which can be used together with the TWR-K40X256 tower processor board
- 306 segment SLCD on the K40 *KWIKSTIK*

During initialisation, the macro `CONFIGURE_SLCD()` is called. This macro is configured in `app_hw_kinetis.h` to suit the display types. The defines for the two SLCD types are shown below:

TWRPI-SLCD

```
#define CONFIGURE_SLCD() MCG_C1 |= MCG_C1_IRCLKEN; \
                         POWER_UP(3, SIM_SCGC3_SLCD); \
                         LCD_GCR = (LCD_GCR_VSUPPLY_VLL3 | LCD_GCR_SOURCE | LCD_GCR_LCLK_4 |
LCD_GCR_DUTY_4BP | LCD_GCR_ALTDIV_NONE); \
                         LCD_PENL = 0x0070f00f; \
                         LCD_PENH = 0x00000000; \
                         LCD_BPENL = 0x0000000f; \
                         LCD_BPENH = 0x00000000; \
                         fnClearSLCD(); \
                         WRITE_SLCD(3TO0, 0x08040201); \
                         LCD_GCR = (LCD_GCR_LCDEN | LCD_GCR_VSUPPLY_VLL3 | LCD_GCR_SOURCE |
LCD_GCR_LCLK_4 | LCD_GCR_DUTY_4BP | LCD_GCR_ALTDIV_NONE);
```

KWIKSTIK

```
#define CONFIGURE_SLCD()  MCG_C1 |= MCG_C1_IRCLKEN; \
                          POWER_UP(3, SIM_SCGC3_SLCD); \
                          LCD_GCR = (LCD_GCR_CPSEL | LCD_GCR_RVEN | LCD_GCR_RVTRIM_MASK |
LCD_GCR_LADJ_MASK | LCD_GCR_LCLK_0 | LCD_GCR_VSUPPLY_VLL3_EXT | LCD_GCR_SOURCE |
LCD_GCR_DUTY_8BP | LCD_GCR_ALTDIV_NONE); \
                          LCD_PENL = 0xfffffffe; \
                          LCD_PENH = 0x0000ffff; \
                          LCD_BPENL = 0x00000000; \
                          LCD_BPENH = 0x0000ff00; \
                          fnClearSLCD(); \
                          WRITE_SLCD(43TO40, 0x08040201); \
                          WRITE_SLCD(47TO44, 0x80402010); \
                          LCD_GCR = (LCD_GCR_LCDEN | LCD_GCR_CPSEL | LCD_GCR_RVEN |
LCD_GCR_RVTRIM_MASK | LCD_GCR_LADJ_MASK | LCD_GCR_LCLK_0 | LCD_GCR_VSUPPLY_VLL3_EXT |
LCD_GCR_SOURCE | LCD_GCR_DUTY_8BP | LCD_GCR_ALTDIV_NONE);
```

The configuration sequence involves the following steps:

- The 32kHz internal RC oscillator is enabled since it can be used as a clock source to the controller, which requires a clock in the 30.0kHz to 39.063kHz range.
- The SLCD module is powered up.

- The basic configuration is set (this depends on the requirements of the SLCD module in respect to voltages, etc.) but the operation is not yet enabled. Note that the clock source matches the required clock speed and so no clock dividers are configured.
- The required frontplane and backplane signals are enabled (`LCD_PENx` and `LCD_BPENx`) which depends on the SLCD type, size and connection. *The K40's GPIO pins associated with the SLCD all default to SLCD use and no specific GPIO configuration is necessary to use them*.
- The routine `fnClearSLCD()` is called to ensure that all segments default to the off state. If this is not performed segments initial states could be random on the K40. The routine clears all segment register contents.
- The backplane signals are set active.
- Finally the operation is enabled.

When using different module types the initialisation may need to be modified to suit its characteristics, size and connection.

3.3. Kinetis K40 GLCD Control Example

The K40 SLCD controller contains a number of registers which map each individual segment on the SLCD to a bit. Depending on the connection and configuration the bits controlling each segment will be specific to the SLCD type.

The programmer simply needs to set a bit when the corresponding segment should be on and clear the bit when the segment is to be off. The fact that there is no standard coordination between the bits used and also what the symbols actually are there is no SLCD application interface defined. Actual control of the display tends to be relatively simple due to the fact that the segments seem to be controlled as individual port bits.

The following section introduces the µTasker SLCD simulator, where the K40 segment registers will be seen in more detail. In order to automatically interface with the simulator the following macros are used in the K40 project to modify segments:

```
#define WRITE_SLCD(reg, val);  // write segment values
#define SET_SLCD(reg, val);    // set specified bits
#define CLEAR_SLCD(reg, val);  // clear specified bits
#define TOGGLE_SLCD(reg, val); // toggle specified bits
```

There are a number of registers, each controlling up to 32 individual segments ('1' is on and'0' is off). The K40 has 16 such segment registers named `LCD_WF3TO0` to `LCD_WF63TO60`. The programmer must know which segment is mapped to which bit in which register to be able to control the display state. When the macros are used the simulator also automatically updates the resulting image accordingly. If register changes based on technique not using the macros (eg. direct pointer access, which may be more practical in some cases) is used, a final call to one of the macros will also synchronise the state of the simulated image to the register content.

An example of toggling the state of a particular segment would be:

```
TOGGLE_SLCD(3TO0, 0x1000);
```

In this case the 12th bit in the register LCD_WF3TO0 will be toggled and the SLCD display correspondingly updated in the simulator. On the real hardware the SLCD controller will of course also modify the control of the influenced segment accordingly.

4. SLCD Simulator

The µTasker simulator includes an SLCD simulation module which automatically display the content of the SLCD based on the setting of the segment registers. As detailed in the previous section, the use of the segment control macros ensures that these remain synchronised, whereby the changed segments are modified each time such a macro has been executed.

The simulator allows complete project code to be accurately tested within the simulation environment without the need to work on HW. It can also be used for simulating user-defined SLCD modules which are not yet available as hardware in order to accelerate project developments.

Due to the fact that there is no standard SLCD the simulator makes use of SLCD script files which are user definable. These script files are usually maintained in a directory called SLCD in the project simulation directory (eg. \Applications\uTaskerV1.4\Simulator\SLCD).

If the project is configured with a simulator file the simulation will be included. In addition some details about the size and colour of the SLCD is supplied. The following shows the configuration of the TWRPI-SLCD module:

```
#define SLCD_FILE  "SLCD\\TWR_K40.lcd" // SLCD simulation file in the simulator directory SLCD
//#define BIG_PIXEL                    // show SLCD double size
#define LCD_ON_COLOUR (COLORREF)RGB(210, 220, 210) // RGB colour of LCD when backlight is on
#define LCD_OFF_COLOUR (COLORREF)RGB(10, 10, 10)   // RGB colour of LCD when backlight is off
#define LCD_PIXEL_COLOUR   (COLORREF)RGB(0,0,0)    // RGB colour of LCD pixels
            #define GLCD_X  380
            #define GLCD_Y  90
```

The SLCD script file is TWR_K40.lcd. The LCD size is defined as being 380 pixels wide and 90 pixels high, whereby this refers to the simulated image and not directly to the display type being used since they are generally not pixel oriented.

If BIG_PIXEL is activated, the display size is doubled in the simulator.

In order to understand the script file content the simple SLCD will first be displayed:

This example shows 3 x 7-segment digits. The first '1' is a single segment, as are the ':', '°', '%', "AM", "PM" and freescale logo. This means that there are 28 individual segments, which are controlled by particular bits in the SLCD controller's registers LCD_WF15TO12 and LCD_WF23TO20.

The script file requires a monochrome bitmap of each of the segment types; the Freescale logo is a single bit map, as are the '°', '%', "AM", etc.; the 7-segment digits are a collection of 7 different bitmaps; the three 7-segment digits can share the same segment bitmaps since they are essentially built up of the same shapes. The complete script file is shown below:

```
// SLCD definition for the TWRPI-SLCD on the TWR-K40x256 board (350 x 90 pixels
used as base)
//

// bit map defines (these should be monochrome bitmaps) - BMP0 to BMP99 possible
//
BMP0  "freescale1.bmp"
BMP1  "digit1.bmp"
BMP2  "seven_seg_b_l.bmp"
BMP3  "seven_seg_t_l.bmp"
BMP4  "seven_seg_top.bmp"
BMP5  "seven_seg_b_r.bmp"
BMP6  "seven_seg_t_r.bmp"
BMP7  "seven_seg_mid.bmp"
BMP8  "seven_seg_bot.bmp"
BMP9  "degree.bmp"
BMP10 "percent.bmp"
BMP11 "am.bmp"
BMP12 "pm.bmp"
BMP13 "dp.bmp"

// backplane bits
//
bx                      // not used

// front plane bits
// segment number, BMP number, x-coordinate, y-coordinate
//
f123 0 20  15           // freescale logo

f179 1 105 5            // '1'

f120 3 125 5            // first 7-segment - top left
f122 2 125 5            // bottom left
f113 6 153 5            // top right
f114 5 153 5            // bottom right
f112 4 125 4            // top
f121 7 125 41           // middle
f115 8 125 78           // bottom

f107 13 180 26          // ':'

f104 3 205 5            // second 7-segment - top left
f106 2 205 5            // bottom left
f97 6 233 5             // top right
f98 5 233 5             // bottom right
f96 4 205 4             // top
f105 7 205 41           // middle
f99 8 205 78            // bottom

f176 3 255 5            // second 7-segment - top left
f178 2 255 5            // bottom left
f161 6 283 5            // top right
f162 5 283 5            // bottom right
f160 4 255 4            // top
f177 7 255 41           // middle
f163 8 255 78           // bottom

f168 9 310 7            // °
f169 10 335 35          // %
f170 11 330 53          // AM
f171 12 330 69          // PM
```

```
// end of file
```

Script file development is very simple and involves the following steps:

1. Each segment type is assigned a monochrome bitmap, which should be in the same directory as the script file itself. Each is given a unique reference `BMPx`.
2. In this version the backplane signals are not defined.
3. The segments are defined in the frontplane section by referencing each possible segment to its bitmap and its x, y coordinates.

Eg. `f123 0 20 15` means that the segment number 123 (counting from 0..123) is mapped to the bitmap number 0 (reference `BMP0`). It is drawn at SLCD location 20, 15.

Note that the segment number itself is SLCD specific and so this user must know which register bit actually controls each segment. The segment number 123 is in fact the 124th bit in the K40 segment control register set, which corresponds to `LCD_WF15TO12`, bit 27, controlled by `TOGGLE_SLCD(15TO12, 0x08000000);`

Note that segments can share bitmap references positioned at different locations. This is illustrated well by the 7-segment definition, where the three 7-segment groups are constricted by sharing the same bitmaps positioned at locations offset in the x-direction only.

The µTasker project for the Kinetis includes script files and bitmap sets for the two display mentioned earlier, whereby the KWIKSTIK display is rather more complicated than this simpler example. Users can define their won script files based on the example above to support different types or user-specific ones. The following practical tips simplify the process:

- It is recommended first to measure the physical display size to get an idea of dimensions. For example, a display of 3cm x 1cm could be represented by a simulated image size of 300 x 100 pixels, allowing pixel locations to calculated easily.
- Before all bitmaps are available it is advisable to use dummy bitmaps of approximate size of the ones finally to be used. These can simply be black images so that they can be viewed and then adjusted to their exact locations by tuning the BMP size itself and adjusting the x, y coordinates in the script file. Once the image sizes and locations are accurate their content can be optimised.
- The simulator includes a menu item "`SLCD | Show all segments`" which allows all existing segments to be displayed at the same time without the need to write any special application code to aid the script file development. To revert back to normal display the command "`SLCD | Release segments`" can be used.

Results of simulations are shown in the next chapter where working demonstrations are shown. Note that the behaviour in the simulator is the same as the behaviour on the HW target, which is the goal of the simulator, thus removing the necessity to develop projects on the hardware itself and correspondingly improve project development efficiency.

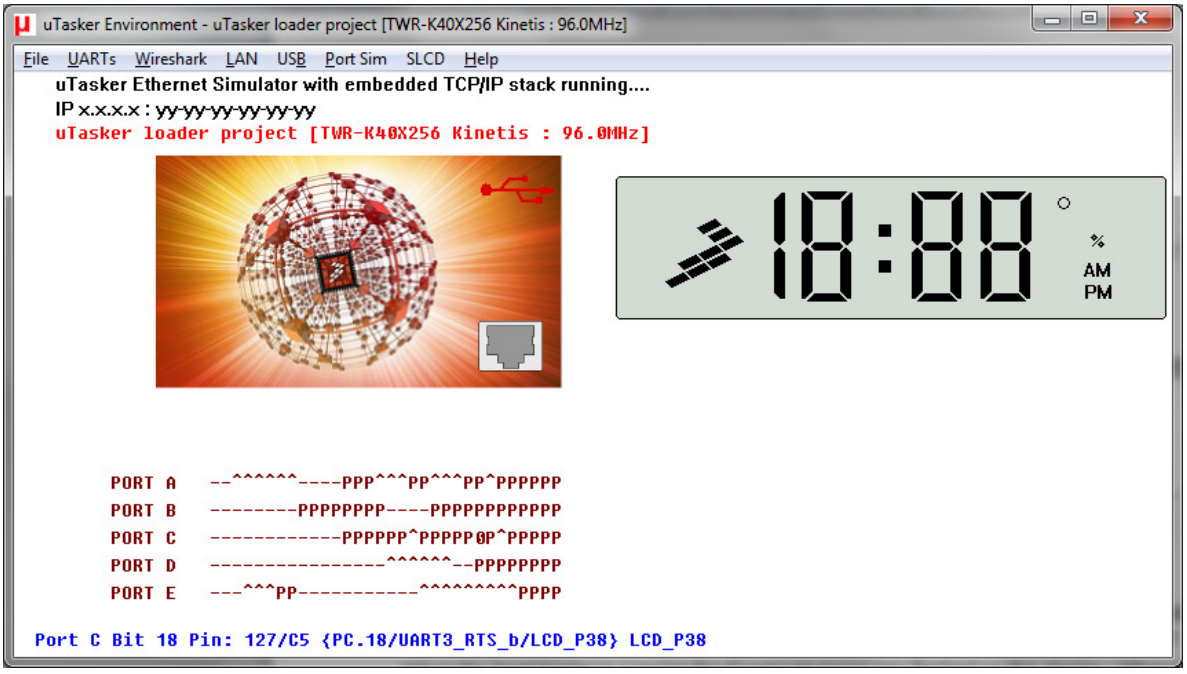
5. Practical Examples

Since the SLCD operation requires very little project code, control of the SLCD was added to the K40 USB boot loader to illustrate its operation on the hardware and in the simulator.

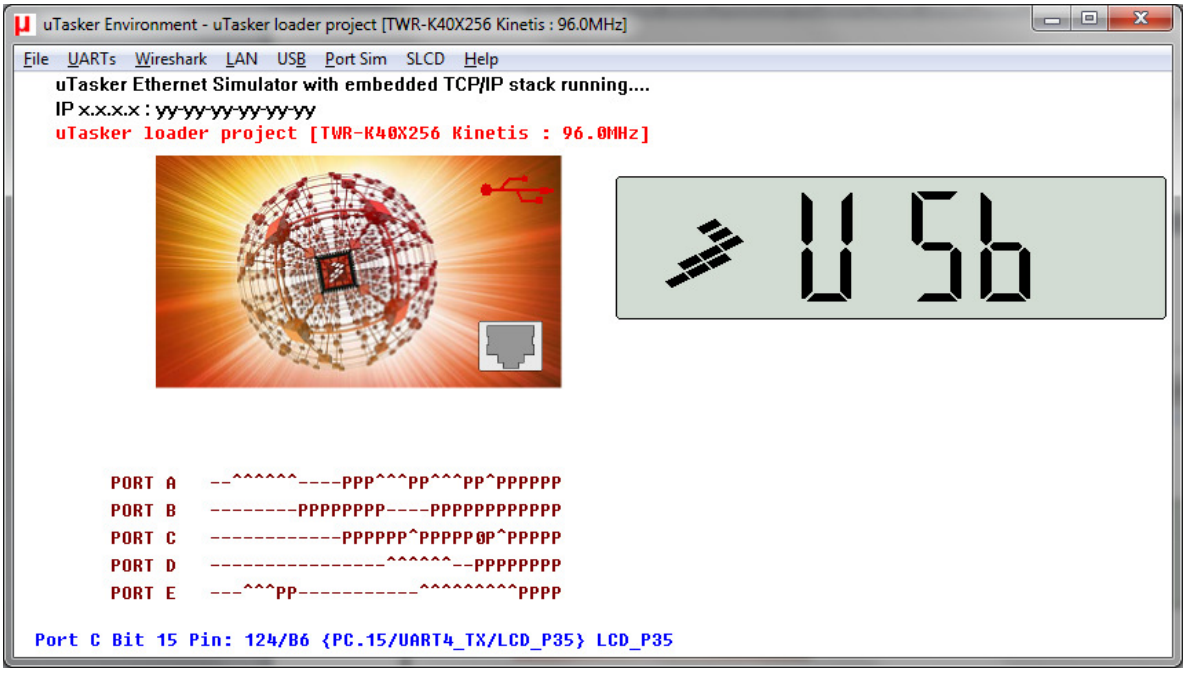
When the boot loader is running the Freescale logos are flashed on the display. When the USB connection is established the USB symbol is displayed on the KWIKSTIK SLCD and the letters USB are displayed on the tower board SLCD.

When simulating, the complete SLCD display can be shown by executing the menu command "`SLCD | Show all segments`". To return to normal view the command "`SLCD | Release segments`" can be executed.

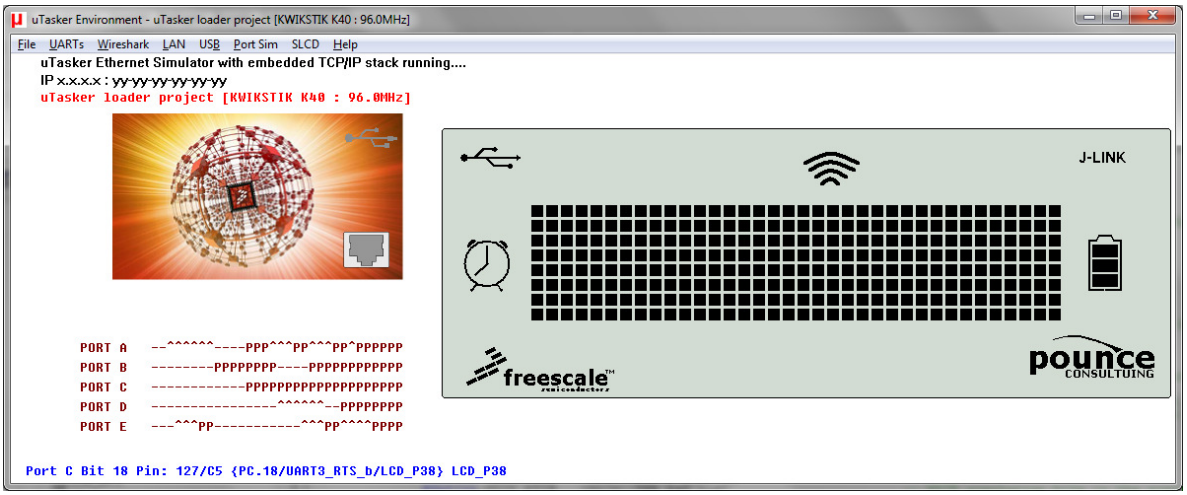
The following images show the simulated displays showing all SLCD symbols as well as the enumerated USB image state (this is the case when either the USB cable is connected to the corresponding processor module or else the enumeration sequence is successfully simulated in the µTasker simulator (by executing the menu "`USB | Enumeration`").



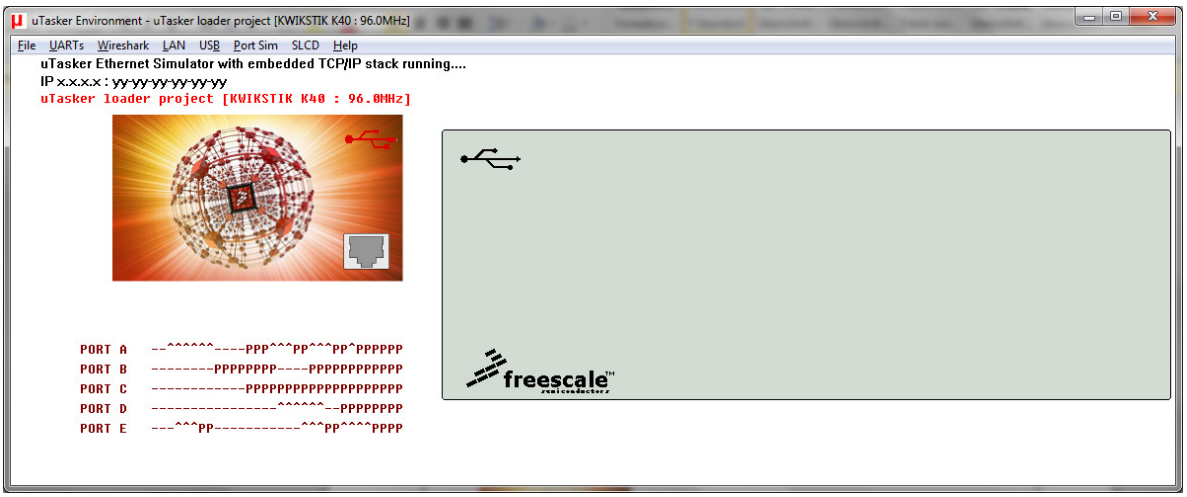
TWR-K40X256 simulation showing all SLCD segments on



TWR-K40X256 simulation showing enumerated USB



KWIKSTIK simulation showing all SLCD segments on



KWIKSTIK simulation showing enumerated USB

The operation of the real modules are show in the following short video:
http://www.youtube.com/watch?v=nm2DmZv1rj8

6. Conclusion

This document has given a brief introduction to the operation of SLCD modules and SLCD controllers.

The SLCD support in the µTasker project has been discussed based on simulation and practical examples of the SLCD operation on Kinetis KL46 and K40 modules.

Modifications:
- V0.0 22.07.2011 First draft – in progress
- V1.0 27.01.2014 FRDM-KL46Z added plus additional generic operation details - released