
AVR065: LCD Driver for the STK502

Features

- Software Driver for Alphanumeric Characters
- Liquid Crystal Display (LCD) Contrast Control
- Interrupt Controlled Updating
- Conversion of ASCII to LCD Segment Control Codes (SCC)
- Interfacing the STK502 LCD Display

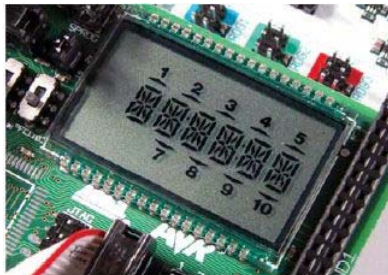
1 Introduction

In applications where user interaction is required it is often useful to be able to display information to the user. A simple interface could be status LEDs, whereas more complex interaction can benefit from a display capable of displaying letters, number, words or even sentences. Liquid Crystal Displays (LCD) are often used for displaying messages. LCD modules can either be graphical ones, which can be used to display graphics and text, or alphanumeric ones capable of displaying 10 – 80 characters. The standard alphanumeric LCD modules are easy to interface but are fairly expensive. These are expensive because they have built-in drivers/controllers, which handle the generation of the characters/graphics on the LCD glass.

The LCD glass is the glass plate in which the liquid crystal is contained. To reduce the cost of an application where a display is required one can choose to use a MCU that have a built-in LCD driver. The MCU can then drive the LCD glass directly, eliminating the need for the driver integrated in the LCD module. The cost of the display can be reduced by as much as a factor of 10, since an LCD glass has a much lower cost than an LCD module.

The ATmega169(P) Flash microcontroller from Atmel has an integrated LCD driver capable of controlling up to 100 segments. The highly efficient core and the very low power consumption of this device make it ideal for battery-powered applications requiring a human interface.

Figure 1-1. The STK502 LCD.



8-bit **AVR**[®]
Microcontrollers

Application Note





2 Theory of Operation

This section provides a basic overview of common features and an introduction to the terminology used in relation to LCD glass.

In addition, a summary of the ATmega169(P) display driver, and a description of the LCD glass used in the application example is discussed. Further, this section contains a summary of the ATmega169(P) LCD features and a description of the LCD glass used in the application example.

2.1 LCD Glass Explained

The Liquid Crystal Display is based on a display technology that uses rod-shaped molecules (liquid crystals) that flow like liquid and bend light. Un-energized, the crystals direct light through two polarizing filters, allowing a natural background color to show. When energized, they redirect the light to be absorbed in one of the polarizers, causing the dark appearance. The more the molecules are twisted, i.e. higher voltage over the LCD segment, the better the contrast and viewing angle.

The LCD must be driven by alternating current (AC). Direct current (DC) will cause electrophoresis effects in the liquid crystal and will degrade the display. There are two AC driving methods: the static driving method and the multiplex driving method.

In the static driving method, the LCD is driven with two square waveforms. The static driving method is the most basic method by which good display quality can be obtained. However, it is not suitable for liquid displays with many segments because one liquid crystal driver circuit is required per segment.

In the multiplex (MUX) driving method, there are a wide variety of drive waveforms and bias levels (explained below) depending on the driver manufacturers. The MUX level depends on the number of backplanes (also called common lines or common terminals). For example, a triple display (1/3 MUX) has three backplanes.

The two most common types of LCDs are: "Twisted Nematic" (TN) and "Super Twist Nematic" (STN). The TN is used for LCD glass with less than 16 back-planes, while STN LCDs are capable of having as much as 240 back planes. The limitations are due to the decrease in transparency as more backplanes are added. More details about the differences between TN and STN LCDs can be found in <http://en.wikipedia.org/wiki/LCD>.

2.1.1 LCD Frame Rate

The number of times the LCD segments are energized per second is called the LCD frame rate. The frame rate should be kept above 30 Hz to avoid that the human eye perceives the segments as flickering.

If a high frame rate is used, ghosting can occur. Ghosting is when LCD segments are not properly turned off. This effect also depends on Duty and Bias (explained below), and it may be necessary to adapt the frame rate to the actual Duty and Bias used. In general, ghosting can be avoided by using sufficiently low frame rates; A frame rate of 100 Hz prevents ghosting.

2.1.2 Segments Drivers and Common Terminals

Each LCD segment has two terminals. One is connected to a segment driver, and the other is connected to a common terminal. By applying an alternating current across

the segment driver and the common terminal, the liquid crystal is polarized (energized) and becomes visible. The common terminal is, as the term implies, common for a group of LCD segments.

To be able to activate individual segments through one (common) terminal that is shared between multiple segments, the driving waveforms must be encoded. If only one common terminal is used, i.e., if each segment driver only drives one segment, the segment drivers of the segments that should not be energized should be of opposite phase of the segment drivers that are energizing segments. This results in a maximal voltage drop over the segment that should be energized, and no or a small voltage drop over the segments that should not be energized. Figure 2-1 and Figure 2-2 show the energized and the non-energized segment and their driving waveforms, respectively, for segments where the LCD drivers are only driving one segment each (static Duty).

Figure 2-1: Two LCD Segments Connected to One Common Terminal

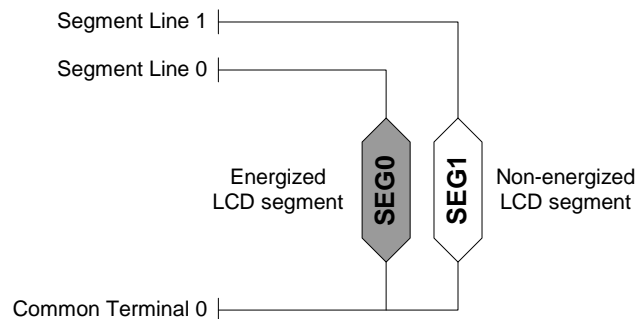
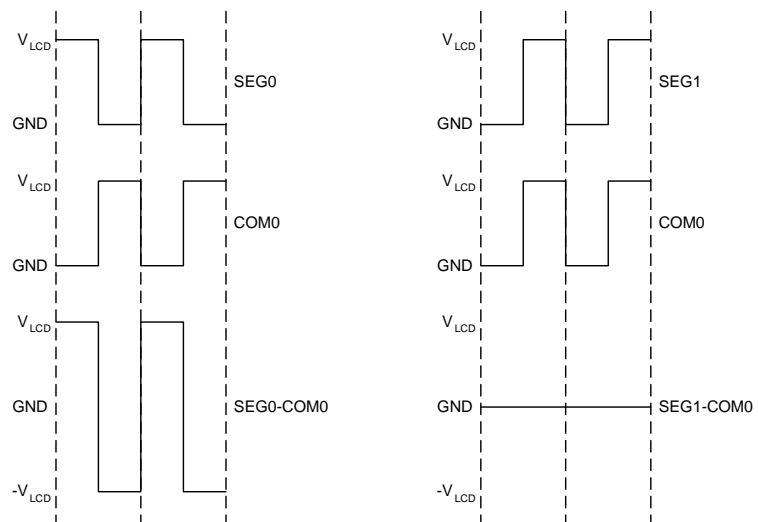


Figure 2-2: Driving Waveforms for Two LCD Segments Connected to the Same Common Terminal



The left side of Figure 2-2 shows the active segment's driving waveforms and the right side of Figure 2-2 shows the inactive segments' driving waveforms.

The energizing of segments becomes more complex if each LCD driver is driving more than one segment. This happens when multiple back-planes are present. In this

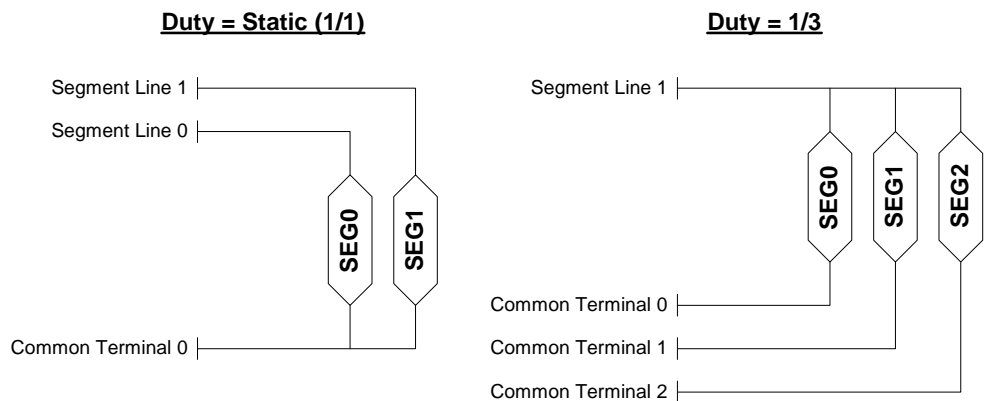


case, the so-called Duty Cycle of the driving waveforms is 1/2 or lower. The Duty Cycle is described in more details below.

2.1.3 Duty Cycle or Duty Ratio

The Duty Cycle or Duty Ratio is a number used to describe for how long each segment is activated during each frame. When each segment driver is only driving one segment the Duty Ratio is “static”. If the drivers are driving more than one segment each the Duty Ratio is given as $1/(\text{segments driven by each LCD driver})$. The number of segments driven by each LCD driver is equal to the number of common terminals. The Duty Ratio is therefore depending on the number of common terminals in a given LCD glass. Figure 2-3 illustrates the relation between the number of back-planes and the Duty Ratio used when controlling the LCD.

Figure 2-3: LCD Segments Controlled by Using One or Three Common Terminals



2.1.4 Drive Bias

The Drive Bias (or just Bias) is related to the number of voltage levels used when driving the LCD. The Bias is defined as $1/(\text{number of voltage levels}-1)$. The more segments driven by each driver, the higher the number of voltage levels is required. The number of segments driven by a single segment line is depending on the number of back-planes in the LCD glass. As per the definition of Duty Ratio, there is a direct relation between the Bias required and the Duty Ratio used.

Table 2-1: Display Duty, Bias and Voltage Levels

Common Terminals	1	2	3	4	6	7	11	12
Duty	Static	1/2	1/3	1/4	1/6	1/7	1/11	1/12
Bias	1	1/2	1/3	1/3	1/3	1/4	1/4	1/5
Voltage Levels	2	3	4	4	4	5	5	6

If three common lines are used in an LCD it has 1/3 Duty Ratio (see Figure 2-3). This means that each segment driver controls up to three segments. To be able to control three segments from one segment driver the Bias needs to be 1/3. In other words it requires four different voltage levels to be able to control three segments using only one driver. Each of the LCD segments connected to a single segment line has different common terminals.

Figure 2-4 shows the driving waveforms of two LCD segments driven by the same segment line and connected to two different back-planes. The illustration shows that the driving waveform has four voltage levels, Bias of 1/3, which is sufficient to drive up to six back-planes. However since the waveform shows three cycles within one frame the Duty is 1/3 and indicating that the glass has three back-planes.

Figure 2-4: Driving Waveforms of Two Different LCD Segments

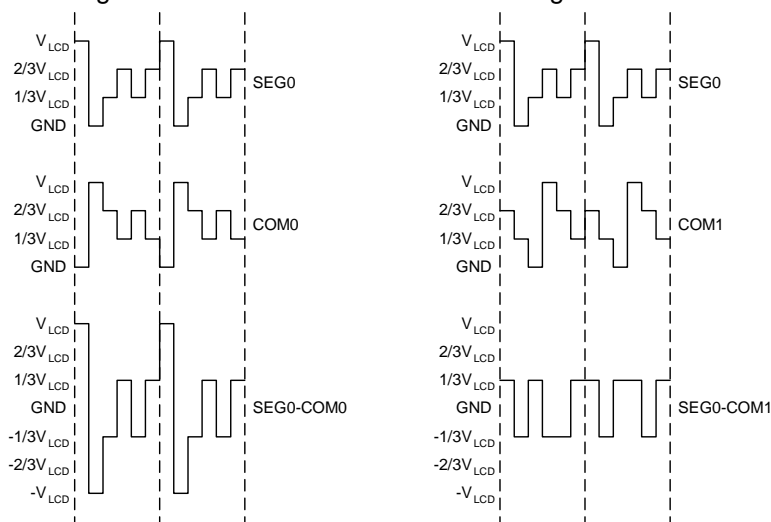


Figure 2-4 shows driving waveforms of two different LCD segments connected to two different common terminals. The segment line is shared. The left side figure is illustrating the active segment and the right side figure is illustrating the inactive segment. The segment represented by the right side figure is inactive because the LCD activation voltage threshold is not passed.

2.1.5 LCD Contrast

LCD contrast is a function of the RMS value of the back-plane minus segment line waveforms at that segment location. Waveforms can be generated such that, at any point in the LCD structure, the resulting RMS voltage is either above the saturation voltage or below the visual threshold voltage. Saturation voltage is the voltage level where the crystals are fully polarized.

2.2 LCD Features of the ATmega169(P)

The ATmega169(P) is the first AVR with integrated LCD drivers. The ATmega169(P) can with its four back-plane lines and 25 segment lines drive up to 100 LCD segments.

To provide high flexibility the LCD driving waveform is selectable. The Duty Ratio and Bias is programmable making it possible to interface LCD glass with one to four back planes and between 13 and 25 segment lines. The lines not used for driving the LCD can be used as general IO. The LCD contrast can be controlled by varying the driver voltage level between 2.6V and 3.35V – independent of VCC. These voltage levels only apply to the LCD pins while used by the LCD interface.

To be able to optimize for performance and current consumption the ATmega169(P) uses programmable frame rates and allow the use of “low power waveforms”. The low





power waveform ensures that the switching of the segment and common lines are kept at a minimum frequency. Further, the power saving modes of the ATmega169(P) allows the MCU to continue driving the LCD glass while reducing its current consumption to a minimum.

Due to the built-in LCD interrupt source, the software driver can be fully interrupt driven. This can be used to ensure that the timing related to updating the LCD is correct. It is therefore possible to avoid that partly updated LCD Data Registers are latched to the LCD lines.

2.3 The STK502 LCD Glass

The LCD software driver described in this document is made for the STK502, which is an add-on module for the STK500 development board. A brief description of the LCD on the STK502 is provided here, further details on the STK500 and STK502 can be found in their respective User Guide.

The LCD glass mounted on the STK502 is illustrated in Figure 2-5. It consists of seven alphanumerical symbols and various fixed symbols; Numbers from one to ten, a bell, a low-battery symbol and navigation arrows.

Figure 2-5: Layout of the STK502 LCD Glass

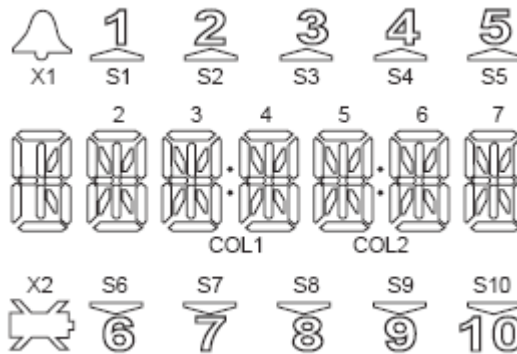
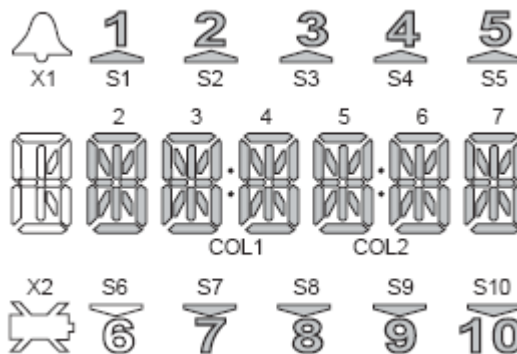


Figure 2-6: LCD Segments Connections on the STK502

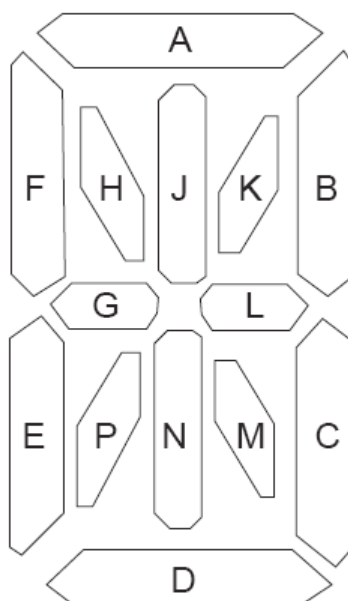


The LCD glass has in total 120 segments – controlled through four back-planes and 30 segment lines. Since the ATmega169(P) is capable of driving 100 segments some of the segments of the LCD glass is not connected in the standard configuration that

the STK502 comes with (see Figure 2-6). The software driver described in this document assumes that the LCD on the STK502 is connected in accordance with the standard configuration that the STK502 comes with.

The STK502 LCD has six similar segment groups, where each segment group is capable of displaying an alphanumeric character. One such group of segments capable of displaying one alphanumeric character is subsequently referred to as a LCD digit. It consists of 14 separate segments. Figure 2-7 shows an LCD digit and the letter used to refer to each of the segments within a LCD digit.

Figure 2-7: Segments and Reference Letters of the LCD Digits



According to the data sheet in the STK502 User Guide, the LCD must be operated at 1/4 Duty and 1/3 Bias. This is to be able to control all four back planes. It is recommended to supply the LCD with 3V.

3 Implementation

This section contains information about the functions included in this driver, and how to use them. The firmware can be downloaded from the Atmel website: <http://www.atmel.com/products/AVR/>. For compiler info and settings, device settings, target setup info and comprehensive source documentation please see the readme.html file included with the source.

3.1 Connections Between the LCD and the ATmega169(P)

The connections between the LCD display and the ATmega169(P) are described in the STK502 User Guide. It is important to note that power to the LCD is supplied from the ATmega169(P) – it does not have separate supply lines.



3.1.1 LCD Data Registers

The LCD segments are individually controlled through the bits of the 20 LCD Data Registers (LCDDR19:0). Not all bits in all 20 LCD Data Registers are used, the 7 highest bits in the highest register for each common terminal is not used in the ATmega169(P). Each segment is uniquely controlled through setting or clearing its corresponding bit in LDDR19:0. The AVR LCD module handles the encoding of the physical drive waveforms to the LCD.

3.1.2 Relation Between LCD Data Registers and LCD Segments

To be able to make an efficient LCD software driver, in terms of code density, the physical connections between the LCD digits and the segments lines and common lines must be well organized: Each LCD digit must be related in the same/similar way to the LCD Data Registers. This will simplify the translation from an ASCII character to LCD segment control codes described below. The Control Codes can be written to the LCDDR19:0 Registers. The data that is written to the LCDDR19:0 Registers are thus the direct control of the LCD segments through setting and clearing their corresponding bits in the LCD Data Registers.

3.1.3 LCD Segment Control Codes

The similarities between connections of the six LCD digits available in the STK502 LCD software driver can be seen from bit mapping tables in the STK502 User Guide. The even digits are using the low nibbles of the associated LCD Data Registers and the odd are using the high nibbles. Four different LCD Data Registers are used for each LCD digit; these are all related to different back planes. The relation between the LCD Data Registers and the digits are organized so that the address offset between each LCD Data Register used is fixed; The interspacing between the addressed are in all cases 0x05. Finally, digit 2 and 3, 4 and 5, and 6 and 7 are in pairs respectively starting at LCDDR0, LCDDR1, and LCDDR2.

To be able to translate ASCII characters into the LCD segment control codes (SCC) required to set and clear the bits in the LCD Data Registers, the bit mapping tables in the STK502 User Guide are used. Since the tables can be reused for all LCD digits only one bit mapping table needs to be used.

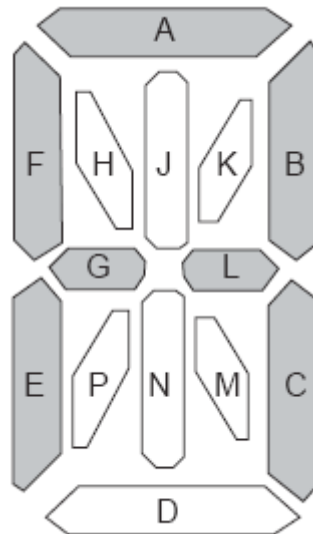
To control all segment lines of a LCD digit 16 bits are required. The LCD SCC is thus 16 bit wide, arranged so that each nibble of the LCD SCC is related to one LCD Data Register. The SCC can therefore be used as follows:

$$\text{SCC} = \{\text{bit } 16:0\} = \{\text{Nibble } 4:0\} \sim \{\text{LCDDR}_x+15, \text{LCDDR}_x+10, \text{LCDDR}_x+5, \text{LCDDR}_x\};$$

The relation between the SCC nibbles and the LCD Data Registers is conditioned by which LCD digit is accessed. Even LCD digits use the low nibble in the LCD Data Registers while the odd numbered LCD digits use the high nibble in the LCD Data Registers.

Consider an example where the letter "A" will be shown in an digit . The letter "A" will look like described in Figure 3-1.

Figure 3-1: The Letter “A” Displayed in a LCD Digit



As seen the letter “A” require activation of the LCD segments {A, B, C, E, F, G, L}. To describe this in the LCD SCC the bit maps from the STK502 is used. A bit map table as shown in Table 3-1, represent both even and odd LCD digits.

Table 3-1. LCD digit segment mapping into the LCD Data Registers

Register Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDDRx	K	-	-	A	K	-	-	A
LCDDRx+5	J	F	H	B	J	F	H	B
LCDDRx+10	L	E	G	C	L	E	G	C
LCDDRx+15	M	P	N	D	M	P	N	D

To determine the LCD SCC for the ASCII character “A” consider the following:

Nibbles	Segments	Char “A”
SCC = {nibble0}	= {{K, -, -, A},	= {{0, 0, 0, 1},
{nibble1}	{J, F, H, B},	{0, 1, 0, 1},
{nibble2}	{L, E, G, C},	{1, 1, 1, 1},
{nibble3}	{M, P, N, D}}	{0, 0, 0, 0}}

The SCC for character “A” can thus be determined to be:

$$SCC_A = \{0x0, 0xF, 0x5, 0x1\} = 0x0F51$$

Many standard ASCII characters and symbols are converted to LCD SCC format. The LCD SCC codes are stored in a flash table, which is used when translating the ASCII characters to LCD SCC at run-time.





Several legal ASCII symbols are not used in the Flash table, since some of these are not possible to display. These locations in the table can e.g., be used to define customer symbols for the LCD.

3.2 AVR Studio® LCD Plug-in

To evaluate/test the code described in this application note without the STK502, the LCD plug-in for AVR Studio 4 can be installed. The plug-in is downloaded with the AVR LCD Visualizer found together with AVR Studio 4.12 on Atmel AVR web <http://www.atmel.com/products/avr/>. This makes it possible to see what the LCD on STK502 looks like by opening the configuration file for the STK502 LCD in AVR Studio. The file is named Mega169.dis and is on a standard installation found in the folder: C:\Program Files\Atmel\AVR Tools\AvrLcd\Displays\Mega169\

The LCD visualizer can be used to design your own LCD display, hook it to any LCD registers and simulate/emulate the display in AVR Studio.

4 Firmware Description

This section contains detailed information about how to use the LCD driver and how it is implemented.

In addition to the descriptions found here, a main.c file is provided along with the LCD driver. The main.c file is an example of how to use the LCD driver and can be studied to understand the driver fully.

4.1.1 LCD Driver Functions

The functions found in the file LCD_driver.c are listed and commented in Table 4-1.

Table 4-1. LCD Software Driver Functions

Function Name	Arguments	Return	Description
LCD_Init (global)	void	void	Initialize the LCD display Data Buffer, which is used to buffer the LCD Data Registers. The LCD frame rate and contrast is selected. The segment lines and common lines are configured.
LCD_WriteDigit (global)	unsigned char c, unsigned char digit	void	The ASCII character passed in the "c" argument is converted to a LCD Segment Control Code (SCC). The SCC is the data that maps the ASCII into LCD Segment Symbols. The SCC is copied into the LCD Display Data Buffer. (The actual update is handled by the LCD_SOF_interrupt routine.) Digit is the number of the digit that is desired accessed. The value of digit is in this implementation limited from 2 to 7.
LCD_AllSegments (global)	unsigned char input (used as bool)	void	Clear or sets all the segments of the LCD (updating the LCD Display Data Buffer only – the actual update is handled by the LCD_SOF_interrupt routine).

Function Name	Arguments	Return	Description
LCD_SOF_interrupt (local, interrupt service routine)	void	void	Latches the LCD Display Data Buffer to the LCD Data Registers. The latching is depending on the LCD_timer variable and the LCD_status.updateRequired variable.

4.1.2 Macros

The macros relevant when using the LCD software driver are listed in Table 4-2.

Table 4-2. Macros used with the LCD Software Driver

Macro Name	Arguments	Description
LCD_CONTRAST_LEVEL	level	Adjust the LCD contrast. Ranges from 0 to 15, where 15 gives highest contrast level.
LCD_SET_COLON	active	Display or hide the colons in the LCD. Valid input is [TRUE/FALSE]

4.1.3 Global Variables

The variables listed in are the global variables that are required to get control of the communication between the main LCD driver functions and the LCD interrupt function. Details on the functionality of the variables are provided in.

Table 4-3. Global Variables Required When Using the LCD Software Driver

Variable Name	Description
LCD_status.updateRequired (bitfield, bool)	If TRUE the LCD_SOF_interrupt routine will be allowed to latch the LCD Display Data Buffer to the LCD Data Registers. If FALSE the interrupt routine will not latch the LCD display data. This variable can thus be used to request or block the LCD Display Data Buffer latching: While updating the LCD display data buffer the variable should be set to FALSE, so that no latching is performed until the LCD display data buffer is fully updated.
LCD_status.updateComplete (bitfield, bool)	The variable is set to TRUE when the LCD Display Data Buffer has been latched. This is done in the LCD_SOF_interrupt routine. The variable can thus be used to test if the data written to the LCD display buffer has been latched after accessing the LCD display data buffer. It can be used to handle update timing since the LCD interrupt is occurring with fixed intervals. The variable can be used to control calls to the LCD_WriteDigit function.
LCD_timer (Unsigned char)	Variable is decremented in the LCD_SOF_interrupt routine. When the value becomes 0, the next latching of the LCD display data buffer will occur. The default timer seed is also reloaded at this time. The variable controls the duration of the interval between LCD updates. The LCD update also depend on the LCD_status.updateRequired variable. If the update cannot be performed because LCD_status.updateRequired is FALSE, the LCD update will be attempted during the next LCD SFO interrupt. The variable can be set to one from the main application if immediate updating of the LCD is desired.



4.2 Function Flowchart

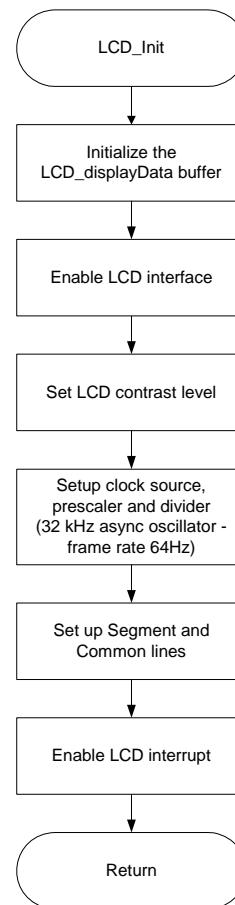
The driver consists of the functions described in text and flowchart below.

4.2.1 LCD_Init

The function used to initialize the LCD software driver is called LCD_Init.

The LCD_Init function first clears the LCDdisplayData buffer. This approach is chosen since it is more code size efficient than initializing it when the LCD_displayData buffer is defined. The AVR LCD module is enabled and configured to match the LCD mounted on the STK502; The contrast level is set, the clock source is set to the external 32 kHz asynchronous Oscillator and the LCD clock prescaler and divider are set to generate a frame rate of 64 Hz. Finally, the LCD Start Of Frame (SOF) interrupt is enabled. The global interrupt is not enabled – this will be most often handled in the main function.

Figure 4-1. Flowchart of the LCD_Init function



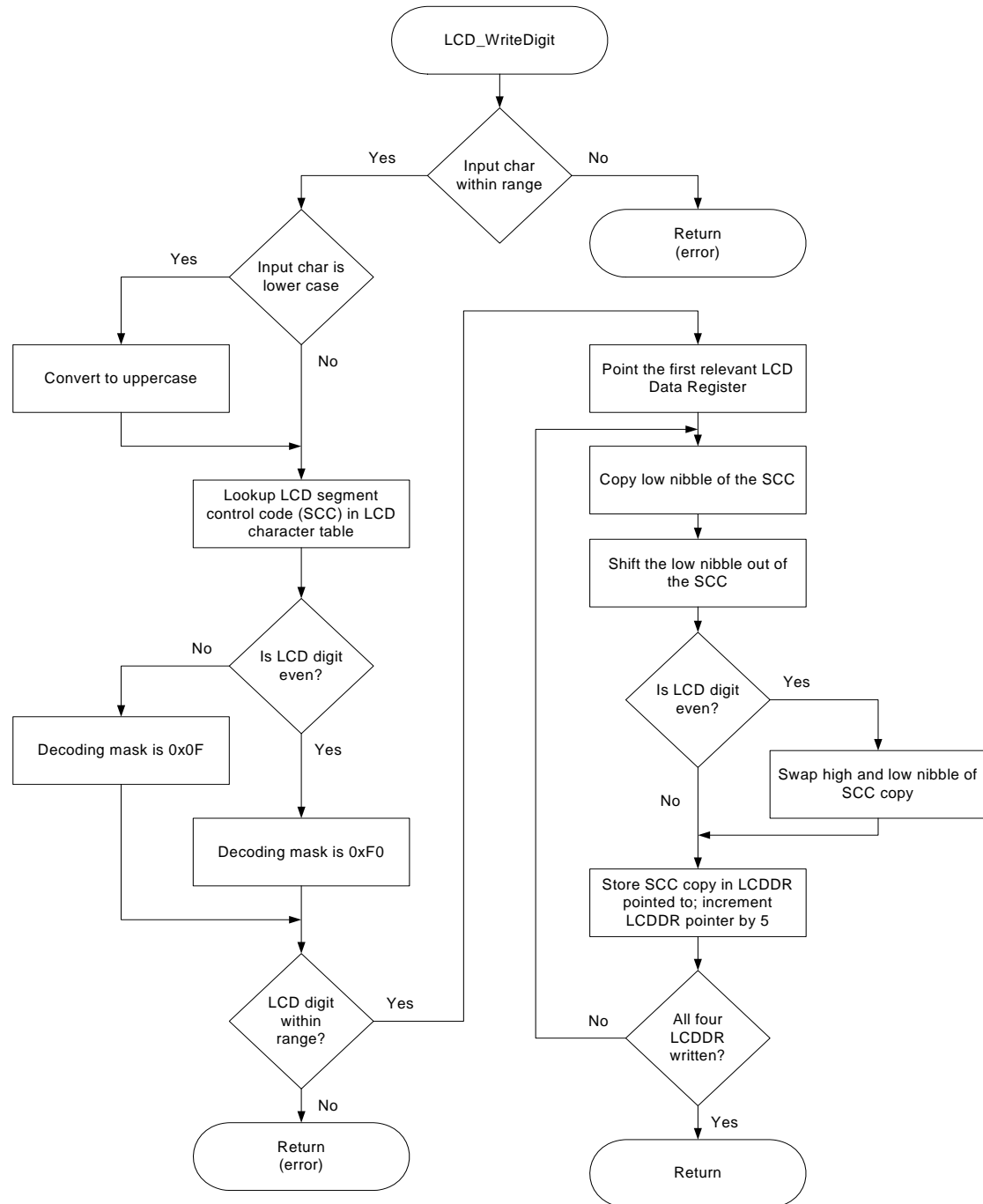
4.2.2 LCD_WriteDigit

The purpose of the LCD_WriteDigit function is to convert and copy an ASCII character into the LCD_displayData buffer. Note that the latching of the buffer into the LCD Data Registers is handled in the LCD SOF interrupt.

The character passed as a function argument is tested to verify that it is within the character range in the ASCII table. If the character is a lower case character, it is converted to upper case. The character minus the offset from NULL is used as a look up in a Flash table. The data retrieved from the table is the LCD Segment Control Code, which is a two byte value. Each of the four nibbles in the LCD SCC is then merged into the corresponding LCD Display Data Buffer elements related to the specific LCD digit that is being updated. Recall that the LCD Display Data Buffer is the information that is latched directly into the LCD Data Registers. The four nibbles of the LCD SCC are therefore merged into the corresponding LCD Data Registers through the buffer.



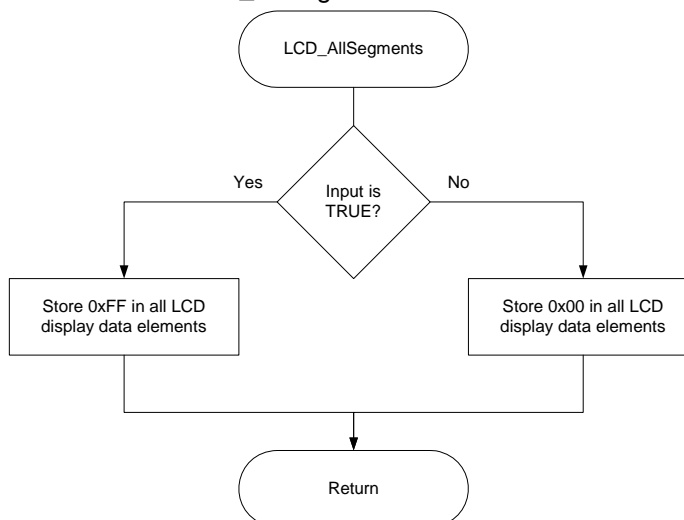
Figure 4-2. Flowchart of the LCD_WriteDigit function



4.2.3 LCD_AllSegments

Depending on the calling argument the LCD_AllSegments clear or set the entire LCD_displayData buffer. This will eventually clear or set the LCD segments.

Figure 4-3. Flowchart of the LCD_AllSegments function



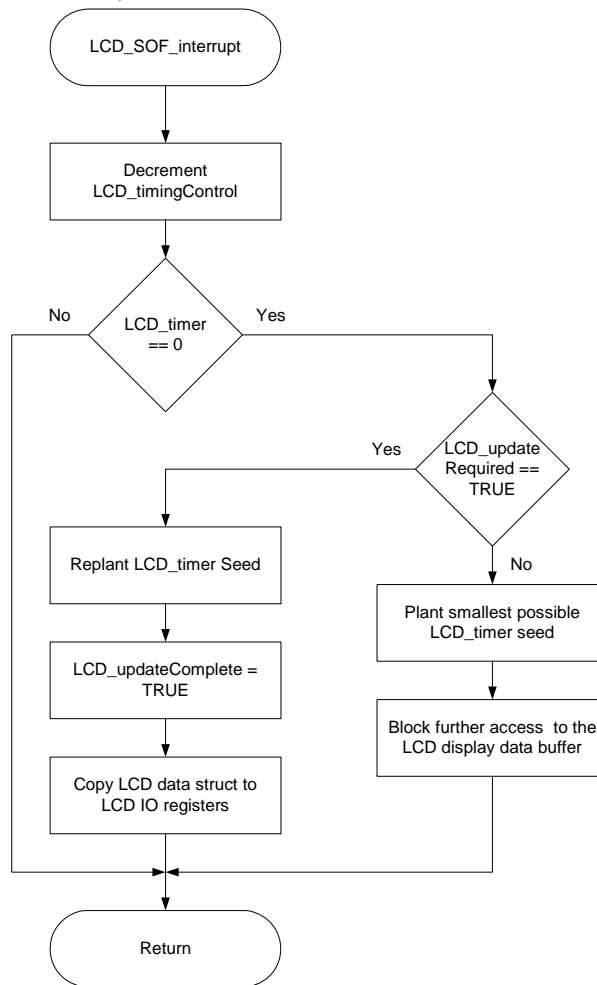
4.2.4 LCD_SOF_interrupt

Updating of the LCD is handled by the LCD_SOF_interrupt; It latches the data from the LCD_displayData Buffer to the LCD Data Registers when the interrupt occurs. Two variables are influencing the LCD update – the LCD_status.updateRequired Flag and the LCD_timer variable.

Every time the LCD_SOF_interrupt is executed the LCD_timer is decremented. Once it reaches zero the LCD_status.updateRequested Flag is tested. If it tests TRUE the LCD_timer is reloaded with the default timer seed and then the LCD_displayData is latched into the LCD Data Registers. When an LCD update is complete the LCD_status.updateComplete Flag is set to indicate that the LCD_displayData has been latched.

If however the LCD_timer has been decremented to zero, but the LCD update is blocked by the LCD_status.updateRequested being FALSE, the LCD_timer is loaded with the smallest possible timer seed (one) and the LCD_status.updateComplete Flag is set to FALSE. This will ensure that a new LCD update is attempted in the first consecutive LCD SOF interrupt. The reason that the LCD_status.updateComplete Flag is cleared is that this should be used in the main routine to test if the LCD_displayData Buffer can be updated. If the LCD_status.update Complete Flag is cleared the main routine should not initiate further updating of the LCD_displayData Buffer. Still, ongoing access to the LCD_displayData buffer should not be terminated while the LCD_status.updateComplete Flag is cleared.

Figure 4-4. Flowchart of the LCD Start Of Frame Interrupt Service Routine (the LCD_SOF_interrupt function)



4.3 Literature List

1. STK502 User Guide found at Atmel AVR web: <http://www.atmel.com/products/avr/>
2. ATmega169(P) Data Sheet found at Atmel AVR web site under application notes: http://www.atmel.com/dyn/products/datasheets.asp?family_id=607
3. LCD, H4042-DL DE5156/L data sheet, included in the STK502 User Guide.
4. LCD technology, <http://en.wikipedia.org/wiki/LCD>



Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.