AVR063: LCD Driver for the STK®504

Features

- **Software Driver for Alphanumeric Characters**
- **Liquid Crystal Display (LCD) Contrast Control**
- **Interrupt Controlled Updating**
- **Conversion of ASCII to LCD Segment Control Codes (SCC)**
- **Interfaces the STK504 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 alphanumerical ones capable of displaying 10 – 80 characters. The standard alphanumerical LCD modules are easy to interface, but are fairly expensive because of the built-in drivers/controllers handling 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 with a built-in LCD driver to drive the LCD glass directly. This eliminates the need for a driver integrated in the LCD module, and thus reducing cost by up to as much as a factor of 10, since an LCD glass has a much lower cost than an LCD module.

The ATmega3290 Flash microcontroller from Atmel has an integrated LCD driver capable of controlling up to 160 segments. The highly efficient core and the very low power consumption of this device make it ideal for battery-powered applications. The STK504 is a hardware expansion board for STK500 that add support for 100 pin AVR LCD devices.

Figure 1-1. The STK504 LCD

8-bit AVR® Microcontrollers

Application Note

Rev. 8009A-AVR-04/06

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 ATmega3290 display driver, and a description of the LCD glass used in the application example is discussed. Further, this section contains a summary of the ATmega3290 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. Unenergized, 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 [(1)], 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 method: the static driving methods 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 in number of back planes are caused by a decrease in transparency due to increasing number of back-planes.

Note: 1. The LCD crystals are more twisted when more polarized due to higher voltage over the LCD segment.

2.2 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. Ghosting is also depending on Duty and Bias (explained below) and it may be required 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 or less prevents ghosting.

2.3 Segments Drivers and Common Terminals

Each LCD segment has two terminals. One is connected to a segment driver 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 describes common for a group of LCD segments.

To be able to activate only one segment though one (common) terminal that is shared between multiple segments, the driving waveforms are encoded in a way so that the segment can be activated individually. If only one common terminal is used, that is if each segment driver is only driving one segment, the segment driver of the segments that should not be energized are of opposite phase of the segment drivers that are energizing segments. This result in that there is a maximal voltage drop over the segment that should be energized, while no or only a small voltage drop over the segments that should not be energized. Figure 2-1 and Figure 2-2 shows the energized and the non-energized segment and their driving waveforms, 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

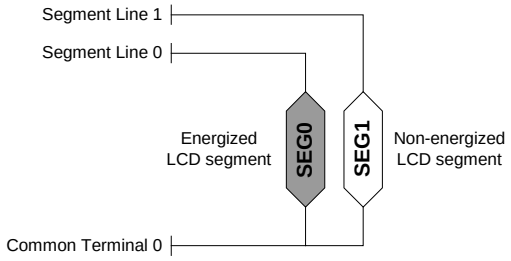
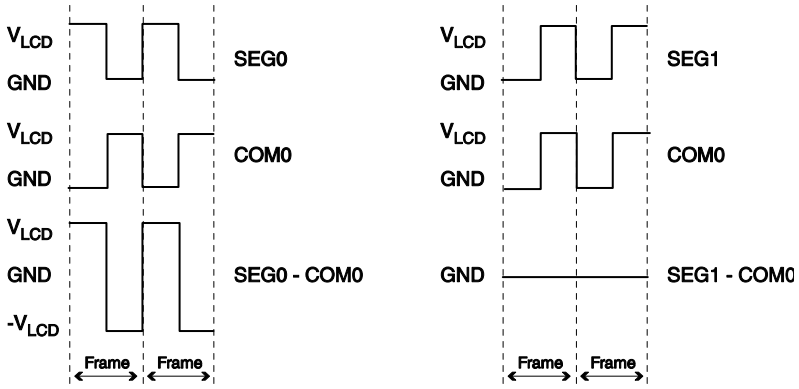


Figure 2-2. Driving Waveforms for Two LCD Segments Connected to the Same Common Terminal



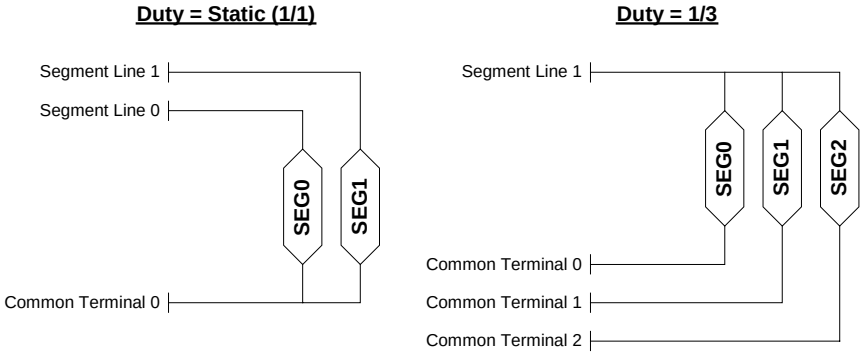
The left side of Figure 2-2 shows the active segment's driving waveforms and the right side of the figure shows the in-active segments driving waveforms.

The way of energizing the segments are 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.

2.4 Duty Cycle or Duty Ratio

The Duty Cycle or Duty Ratio is a number used to describe how long time 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/(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 Static and 1/3 Duty



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/(number of voltage levels-1). The more segments driven by each driver[1] the higher number of voltage levels are required. As per the definition of Duty Ratio, there is a direct relation between the Bias required and the Duty Ratio used.

Note: 1.The number of segments driven by a single segment line is depending on the number of back-planes in the LCD glass.

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

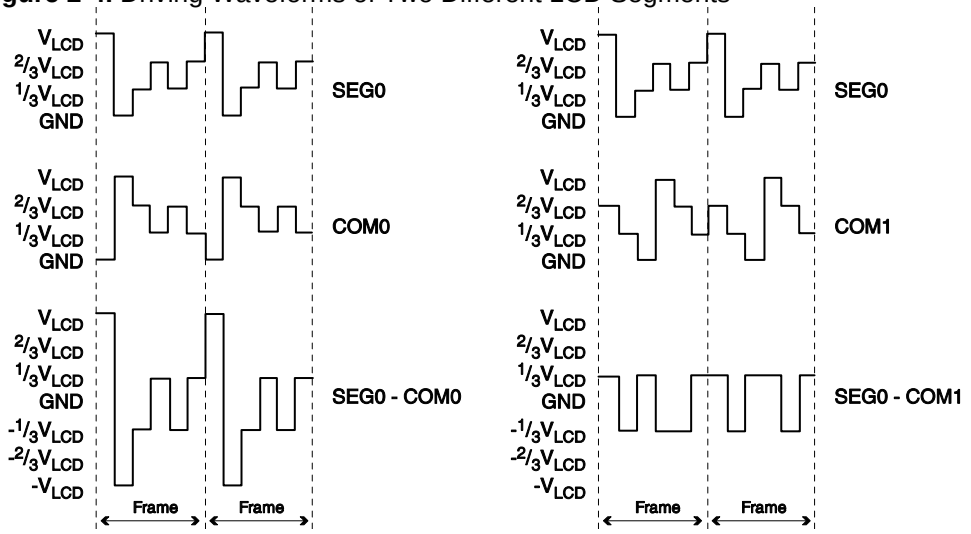


Figure 2-4 shows driving waveforms of two different LCD segments connected to two different common terminals. The segment line is shared. The left side figure is illustrating the active segment; the right side figure is illustrating the inactive segment. The segment represented by the right side figure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

2.5 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(1), or below the visual threshold voltage.

Note: 1. Saturation voltage is the voltage level where the crystals are fully polarized.

2.6 LCD Features of the ATmega3290

The ATmega3290 can with its four back-plane lines and 40 segment lines drive up to 160 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 40 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 ATmega3290 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 ATmega3290 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.

2.7 The STK504 LCD Glass

The LCD software driver described in this document is made for the STK504, which is an add-on module for the STK500 development board. A brief description of the LCD on the STK504 is provided here, further details on the STK500 and STK504 can be found in their respective User Guides.

The LCD glass mounted on the STK504 is illustrated in Figure 2-5. It consists of seven alphanumerical symbols, four numerical symbols and various graphical symbols.

Figure 2-5. The STK504 LCD Glass

The LCD glass has in total 160 segments – controlled through four back-planes and 40 segment lines. Since the ATmega3290 is capable of driving 160 segments, the device controls all segments. The software driver described in this document assumes that the LCD on the STK504 is connected in accordance with the standard configuration for the STK504.

The segments can be divided into three distinct groups

- Alphanumeric 14-segment Symbols
- Numeric 7-segment Symbols
- Graphical Symbols

Because of the position of the LCD segments in the IO memory map, these three groups are handled by separate functions. This will be explained in the implementation part of this application note.

3 Implementation

This section contains a description of where to find info about the physical connection between the ATmega3290 and the LCD display on the STK504 and how LCD segments are controlled. How to map ASCII characters to the LCD digits is also described.

3.1 Connections Between the LCD and the ATmega3290

The connections between the LCD display and the ATmega3290 are described in the STK504 User Guide found in the AVR Studio® On-line Help system. It is important to note that power to the LCD is supplied from the ATmega3290 – it does not have separate supply lines.

3.2 LCD Data Registers

The LCD segments are individually controlled through the bits of the 20 LCD Data Registers (LCDDR19:0). Each segment is uniquely controlled through setting or clearing its corresponding bit in LCDDR19:0. The AVR LCD module handles the encoding of the physical drive waveforms to the LCD.

3.3 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 and common lines must be well organized: Each LCD digit must be related in the same or similar way to the LCD Data Registers. This will simplify the translation from an ASCII character to LCD Segment Control Codes (SCC) described below. The SCC code 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.

3.4 LCD Segment Control Codes

Analyzing the segment mapping of the LCD display on the STK504, shows that it is not efficient to make one generic function to cover both the 7 and 14 segment characters. The segment mapping table is found in appendix B of the STK504 User Guide (in AVR Studio help). Writing to the 14 or 7 segment characters are thus covered by separate functions and will be explained separately.

3.4.1 14-Segment Characters

The odd digits are using the low nibbles of the associated LCD Data Registers and the even 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 STK504 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 14 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:

```
SCC = {bit 15:0} = {Nibble3:0}~{LCDDRx+15, LCDDRx+10, LCDDRx+5,
LCDDRx};
```

Table 3-1 shows how the Segment is for the first (leftmost) character on the LCD display.

Table 3-1. STK504 LCD Segment Mapping for character 1

	Bits								
Common Lines	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Registers**
COM3	2C	NB	2M	2D	**1C**	BEL	**1M**	**1D**	LCDDR15
COM2	2H	2N	2L	2E	**1H**	**1N**	**1L**	**1E**	LCDDR10
COM1	2B	2K	2J	2G	**1B**	**1K**	**1J**	**1G**	LCDDR5
COM0	TU	2A	2I	2F	MO	**1A**	**1I**	**1F**	LCDDR0

The SCC for Character 1 will be:

SCC = { bit 15:0 } ~ { LCDDR15[3:0], LCDDR10[3:0], LCDDR5[3:0], LCDDR0[3:0]};
SCC = { bit 15:0 } ~ { 1C, 0, 1M, 1D, 1H, 1N, 1L, 1E, 1B, 1K, 1J, 1G, 0, 1A, 1I, 1F };

Example: To display the letter ‘A’ in character position 1 segments 1A, 1B, 1C, 1E, 1F, 1G and 1H needs to be set. The correct SCC for the letter A is thus:

SCC_A = {bit 15:0 } = { 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1 } = {0x8995};

A complete listing of all predefined SCC codes can be found in the stk504_lcd.c source file.

The function is responsible for writing the correct bits without changing bits that are not part of the character.

3.4.2 7-segment Characters

The write function for the 7 segment characters is implemented in a similar way, but since these characters only need 7 bits, they are stored in a single byte. The Segment Control Code is thus 8 bits wide. As for the 14 segment characters 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.

```
SCC = {bit 7:0} = {bit1:0}~{LCDDRx+15, LCDDRx+10, LCDDRx+5, LCDDRx};
```

Table 3-2. STK504 LCD Segment Mapping 7-segment characters

	Bits								
Common Lines	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Registers**
COM3	**9A**	**9B**	10A	10B	11A	11B	12A	12B	LCDDR19
COM2	**9F**	**9G**	10F	10G	11F	11G	12F	12G	LCDDR14
COM1	**9E**	**9C**	10E	10C	11E	11C	12E	12C	LCDDR9
COM0	**9D**	OUT	10D	P	11D	COL	12D	ICE	LCDDR4

Only two bit is stored in each LCDDR register, so to write a complete character, 4 registers need to be written.

The SCC is organized as follows:

SCC = { A, B, F, G, E, C, D, 0 };

Example: To display the character ‘A’ on the 7-segment character 9 (see LCD segment mapping in the STK504 user guide) the following segments need to be set: 9A, 9B, 9C, 9E, 9F, 9G. The correct SCC code is thus:

SCC_A = { 1, 1, 1, 1, 1, 1, 0, 0 } = 0xFC;

3.5 Symbols

In addition to the 7-segment and 14-segment characters there is number of graphical symbols that can be used. Since these are scattered around the IO map, the easiest way of handling these is by a separate function. The formats of these SCC codes are slightly different from the 7 and 14 segment functions.

	Bits								
Common Lines	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Registers**
COM3	9A	9B	10A	10B	11A	11B	12A	12B	LCDDR19
COM3	**AVR**	S7	S8	S9	7C	KEY	7M	7D	LCDDR18
COM3	6C	WRM	6M	6D	5C	8BC	5M	5D	LCDDR17

For symbols, the SCC code has the following format:

SCC = { bit:b[7..0] n[19..0], where b is the bit position within the LCDDR register and n is relative offset to LCDRR0.

Example: The SCC code to set the AVR symbol:

SCC_AVR = { bbb nnnnn } = { bit:7 offset:18} = { 111 10010 } = 0xF2;

The complete SCC code list can be found in the STK504_lcd.h source file.

4 Firmware Description

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.

4.1 LCD Driver Functions

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. Power reduction register PRLCD bit is cleared.
LCD_Update(global)	void	void	Transfers the content of the LCD data buffer to the LCDDRx registers. Forces update, instead of using the LCD interrupt routine to update the registers.
LCD_vect_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
LCD_PutChar(global)	unsigned char c, unsigned char digit	void	The ASCII character is passed in “c”. Argument is converted to a SCC code for the 14 segment characters, and displayed in the position indicated by “digit”. Note that the data is written to the LCD Display Buffer, not directly to the LCDDR registers. The actual data update is handled by the LCD_vect_interrupt function
LCD_PutDigit(global)	unsigned char c, unsigned char digit	void	The ASCII character is passed in “c”. Argument is converted to a SCC code for the 7 segment characters, and displayed in the position indicated by “digit”. Note that the data is written to the LCD Display Buffer, not directly to the LCDDR registers. The actual data update is handled by the LCD_vect_interrupt function
LCD_AllSegments(global)	unsigned char input	void	Clears all segments in the LCD display buffer if argument is zero, otherwise sets all segments.

Function Name	Arguments	Return	Description
LCD_SetSeg(global)	unsigned char symbol, unsigned char input	void	Clears the symbol indicated in the LCD display buffer if input is zero, otherwise sets the symbol.
LCD_BarSeg(global)	unsigned char value	void	Function to write to the eight segments inside the temp/battery symbol. Simplifies writing to these symbols, as the decoding is somewhat complex.

The variables listed in Global Variables Required When Using the LCD Software Driver 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 4-2.

Table 4-2. Global Variables Required When Using the LCD Software Driver

Variable Name	Description
LCD_status.updateRequired (bitfield, bool)	If TRUE the LCD_vect_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_vect_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 LcdPutChar/LcdPutDigit functions.
LCD_timer (Unsigned char)	Variable is decremented in the LCD_vect_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 depends 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.

5 Table of Contents

Atmel Corporation

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

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

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

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

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

8009A-AVR-04/06