Stepper Motor Controller

Features

- **High-Speed Stepper Motor Controller**
- **Interrupt Driven**
- **Compact Code (Only 10 Bytes Interrupt Routine)**
- **Very High Speed**
- **Low Computing Requirement**

Introduction

This application note describes how to implement a compact-size and high-speed interrupt driven stepper motor controller. Stepper motors are typically used in applications like camera zoom/film feeder, fax machines, printers, copying machines, paper feeders/sorters and disk drives.

The high performance of the AVR® microcontroller embedded in FPSLIC devices enables the designer to implement high-speed stepper motor applications with low computing requirements of the controller. The Assembly code with the Stepper Motor Controller can be found in the FPSLIC Software section of the Atmel web site (http://www.atmel.com), under the **3045.asm** archive.

Theory of Operation

A DC stepper motor translates current pulses into motor rotation. A typical motor contains four winding coils. The coils are often labeled red, yellow/white, red/white and yellow, but may have other colors. Applying voltage to these coils forces the motor to step one step.

In normal operation, two winding coils are activated at the same time. The stepper motor moves clockwise one step per change in winding activated. If the sequence is applied in reverse order, the motor will run counterclockwise.

The speed of rotation is controlled by the frequency of the pulses. Every time a pulse is applied to the stepper motor the motor will rotate a fixed distance. A typical step rotation is 1.8 degrees. With 1.8 degree rotation in each step will a complete rotation of the motor (360 degrees) require 200 steps.

By changing the interval of the timer interrupts, the speed of the motor can be regulated, and by counting the number of steps, the rotation angle can be controlled.

FPSLIC™

Programmable SLI

AT94K

AT94S

Application Note

Rev. 3045B–FPSLI–04/02

Figure 1. Stepper Motor Step Sequence

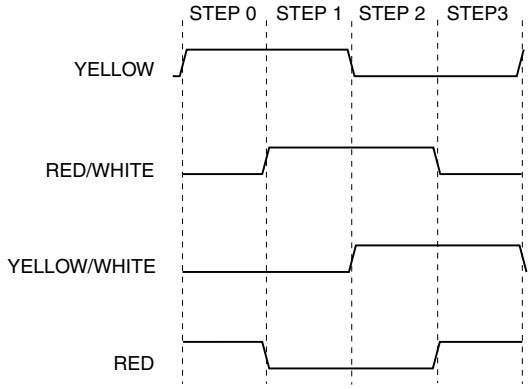


Table 1 shows the hexadecimal values to be output to the stepper motor to perform each step.

Table 1. Stepper Motor Values

Step	Yellow	Red/ White	Yellow/ White	Red	Hex Value
0	1	0	0	1	9
1	1	1	0	0	C
2	0	1	1	0	6
3	0	0	1	1	3

Software Description

The software uses a 16-bit timer with capture function to generate interrupt every 100 ms. When the interrupt is executed, a new step value is output to PORTD.

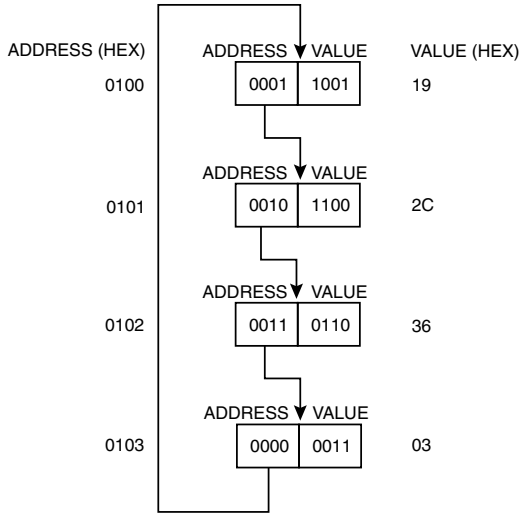
Values for the stepper motor are stored in SRAM program memory. At startup, the values are copied to SRAM data memory to achieve faster access and maximum speed performance.

In this implementation, the interrupt routine takes 7 cycles + 4 cycles to enter and 4 cycles to exit the interrupt. This totals 15 cycles. With a clock speed of 8 MHz, the stepper motor control takes less than 2 µs. If interrupt is required every 100 ms, the stepper motor handling takes only 0.002% of the processing power in the CPU.

In this example the values for the stepper motor are stored at RAM address 0100 (hex). The upper byte of the RAM address is constant and only the low nibble of the low byte is used to access the address information, see Figure 2.

The lower nibble (4 bits) of the variables is the actual value to control the stepper motor, the upper nibble holds the address of the next value.

Figure 2. Stepper Motor Addresses and Values



By using this method, maximum speed can be achieved, combined with a minimum of processor resources.

Resources

Table 2. CPU and Memory Usage

Function	Code Size	Cycles	Register Usage	Interrupt	Description
main	38 Words	-	R16, XL, XH, ZL, ZH	-	Initialization and example program
OC1A	10 Words	13 + Return	R16, XL, XH	Timer 1 Output Compare A	Output stepper motor value and calculate next value
TOTAL	48 Words	-	R16, XL, XH, ZL, ZH		

Table 3. Peripheral Usage

Peripheral	Description	Interrupts Enabled
4 I/O pins	Stepper motor output pins	
Timer 1	Generate timer interrupt for stepper motor frequency generation	Timer 1 Output Compare A

Code Listing

```
;***************************************************************************
;*
;* Title:               Stepper Motor Controller
;* Version:             1.0
;* Last updated:        03.19.2002
;* Target:              AT94K FPSLIC Family
;*
;* Support E-mail:      fpslic@atmel.com
;*
;* DESCRIPTION
;*      This application note describes how to implement a compact-size
;*      and high-speed interrupt driven stepper motor controller.
;*
;***************************************************************************

.include "at94kdef.inc"


;*************** Define Global Registers ************************
.def    rTemp   = r16


;*************** Define Constants ******************************
.equ    cValue  = 500


;***************************************************************************
;*
;*      PROGRAM START - EXECUTION STARTS HERE
;*
;***************************************************************************

.cseg

.org $0000
        rjmp    Main
.org TIM1_COMPAaddr
        rjmp    OC1A


;***************************************************************************
;* OC1A         - Timer1 Output compare A interrupt routine
;*
;* DESCRIPTION
;*
;*This interrupt routine loads new stepper motor value from the stepper
;*motor table in SRAM. The values in the table have two functions,
;*the lower nibble contains the value to output to the stepper motor.
;*The upper nibble holds the address of the next value. First the
```

```
;*step value is output to the port, next the address is moved to
;*the XL register.
;*
;* Number of words     :6 + return
;* Number of cycles    :7 + return
;* Low registers used  :None
;* High registers used :3 (temp,XL,XH)
;***************************************************************************

OC1A:
        in      rTemp, SREG
        push    rTemp
        ld      rTemp, X                ; Load rTemp with X Pointer Value
        mov     XL, rTemp               ; Move Value to X Pointer
        andi    rTemp, $0F              ; Mask Away Upper Nibble
        out     PORTD, rTemp            ; Output Lower Nibble to Stepper Motor
        swap    XL                      ; Swap Upper and Lower Nibble
        andi    XL, $0F                 ; Mask Away Upper Nibble, Address is
Ready
        pop     rTemp
        out     SREG, rTemp
        reti



;***************************************************************************
;*
;* Main Program
;*
;*This program initializes Timer 1 output compare interrupt to
;*occur with a interval defined with the c_value constant.
;*The stepper motor look-up table is loaded from the flash and stored
;*in SRAM address 0x0100 to achieve maximum speed.
;*
;***************************************************************************

MAIN:

        ldi     rTemp, high($0FFF)      ; Initialize Stack Pointer
        out     SPH, rTemp
        ldi     rTemp, low($0FFF)
        out     SPL, rTemp
        ldi     rTemp, $0F              ; Set PORTD(3..0) as Output
        out     DDRD, rTemp
        ldi     rTemp, $00
        out     PORTD, rTemp            ; Write Initial Value to PORTD
        ldi     rTemp, high(cValue)     ; Load Compare High Value
        out     OCR1AH, rTemp
        ldi     rTemp, low(cValue)      ; Load Compare Low Value
        out     OCR1AL, rTemp
        ldi     rTemp, $00
```

```
        out     TCNT1H, rTemp           ; Clear TC1 High Byte
        out     TCNT1L, rTemp           ; Clear TC1 Low Byte
        out     TCCR1A, rTemp           ; Clear TC1 Control Register A
        ldi     rTemp, $40
        out     TIFR, rTemp             ; Clear Pending Timer Interrupt
        out     TIMSK, rTemp            ; Enable TC1 Compare Interrupt

       ldi     ZH, high(step * 2)      ; Init Z Pointer to Step Table in Prog
Memory
        ldi     ZL, low(step * 2)
        ldi     XH, high($0100)         ; Init X Pointer to Data SRAM Location
        ldi     XL, low($0100)

        ldi     rTemp, $04              ; Load Counter Value
LOAD:   lpm                             ; Load Step Value from Prog Memory
        st      X+, r0                  ; Store Step Value in Data SRAM
        adiw    ZL, $01                 ; Increment Prog Memory Pointer
        dec     rTemp                   ; Decrement Counter
        brne    LOAD                    ; Continue until Table is Loaded

        ldi     XH, high($0100)         ; Init X Pointer to Data SRAM Location
        ldi     XL, low($0100)

        ldi     rTemp, $09
        out     TCCR1B, rTemp           ; Clear Timer on Compare Match, CK/1
        sei                             ; Enable Global Interrupts

LOOP:   rjmp    LOOP                    ; Do Something Else


STEP:   .db     $019, $02C, $036, $03   ; Stepper Motor Look-Up Table
```

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

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
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
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
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Atmel Nantes
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
Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Atmel Smart Card ICs
Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Atmel Heilbronn
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

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

Atmel Programmable SLI Hotline
(408) 436-4119

Atmel Programmable SLI e-mail
fpslic@atmel.com

FAQ
Available on web site

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

3045B–FPSLI–04/02 0M