Implementing a Sewing Machine Controller with an MC9RS08KA2 Microcontroller

by: Ulises Corrales Salgado
Applications Engineer
RTAC Americas

1 Introduction

This application note explains how to control motor speed using an MC9RS08KA microcontroller.

Many machines or devices are made entirely with mechanical parts, making maintenance a problem. Electronic devices are easier to maintain and cheaper to produce than mechanical devices.

This document explains how to design a low-cost, digital motor-speed control using an accelerometer and an MC9RS08KA2 microcontroller. This option can replace the pulleys and gears on ordinary mechanical devices, reduce cost, and increase functionality.

1.1 Description

Sewing machines can be classified into three types, depending on their build. The first type is made with mechanical parts and is difficult to use because, to control the motor’s speed, you must use both feet to move the pedal (Figure 1).
Introduction

The second type has a pedal made with gauges that function like a variable resistor. The resistor values vary depending on the information of the gauge. When you press the gauge, the motor speeds up. When you release it, it slows down (Figure 2).

The third type of sewing machine has an accelerometer sensor\(^1\) in its pedal. The accelerometer measures the tilt of the pedal. With this, you can speed up or speed down the motor (Figure 3).

The accelerometer measures acceleration and angles. In this application, it measures angles from 0° to 75°.

Figure 4 shows the blocks that make up the three types of sewing machines. The yellow blocks (also outlined with thicker lines) indicate the application note’s focus.

See 2.1, “Solution Benefits,” for the advantages of our solution.

1.2 Design Requirements

To create the accelerometer controller, you need:

- One MC9RS08KA2 microcontroller
- One MMA7260QT accelerometer sensor
- The power stage from the MCU to the DC motor, depending on the motor characteristics
2 Solution

Our solution depends on the accelerometer sensor’s data. The main program reads the analog-to-digital converter (ADC). After the KA2 acquires the data, the microcontroller processes it and generates a pulse-width modulation (PWM), which speeds up or slows the motor. The accelerometer sensor is located in the foot pedal. The motor speed depends on the degree of tilt (Figure 5).

NOTE

This device was made using a DC motor. This device will not work using an AC motor. The power-stage block varies depending on the motor characteristics. Also, the MC9RS08KA2 microcontroller does not have an ADC module, so an RC circuit and software obtains the ADC values.
The accelerometer demo board was used in this application note. The accelerometer demo board has an MC9S08QG that configures the sensitivity sensor and also keeps the accelerometer working in active mode. There is no documentation regarding this board except for a schematic (Appendix A, “Firmware”). You can solve this problem by using hardware such as in Figure 7.

**NOTE**
The MCU connection inside the dashed square are part of the KA2 evaluation board. You should make the DC motor connections and accelerometer connections.

**Figure 7. Sewing Machine Schematic Diagram**

Implementing a Sewing Machine Controller with an MC9RS08KA2 Microcontroller, Rev. 1

Freescale Semiconductor
2.1 Solution Benefits

Device benefits:

- Smaller foot pedal
- Pedal is not temperature-sensitive. If the temperature varies, the ADC reading is the same.
- The foot pedal is cheaper than a gauge pedal.
- Detects tilt changes more precisely, allowing better control over motor speed.

3 Detailed Description

The MCU firmware first configures the microcontroller (this routine configures the bus clock and disables the COP). After that, the modulo timer is configured. Next, the discharge capacitor is configured.

Data table — This part of the code is used to get values between 0 and 255. The values the SensorReading variable obtains are between 40 and 75. These values cannot be used in the PWM routine because the range values are too short, so adjustments are made to these values.

Lookup table — This part of the code does a page calculation to obtain the data from the data table. In this project, the ADC reads 35 values approximately between 0° to 75°. The values must be extrapolated between 0 and 255 because the lowest value the ADC obtains (0° of tilt) is 40, this value has to extrapolate to 255. You can only obtain the extrapolation values by subtracting seven from the previous value. The maximum extrapolation value is 255.

For example, if you want to use eight values rather than all the values the ADC obtained, extrapolate new values and enter them in the data table in the firmware project. Table 1 and Table 2 show the extrapolation values made in the project and the example extrapolation values obtained.

NOTE

The ADC values are always the same, but you can use the values you want. The ADC values you do not choose are obtained by the extrapolated value of its predecessor. In the example, the values between 40 and 44 are extrapolated to the 255 value.

<table>
<thead>
<tr>
<th>Value Obtained by ADC</th>
<th>Degree of Tilt (Approximately)</th>
<th>Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>41</td>
<td>2.2</td>
<td>248</td>
</tr>
<tr>
<td>42</td>
<td>4.4</td>
<td>241</td>
</tr>
<tr>
<td>43</td>
<td>6.6</td>
<td>234</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>71</td>
<td>68.2</td>
<td>38</td>
</tr>
</tbody>
</table>
PWM — The motor speed is controlled in this section of code. There are two main routines: PWMLoopOn and PWMLoopOff.

- The PWMLoopOn routine activates the PTA4 pin, which causes the motor to spin.
- The PWMLoopOff deactivates the PTA4 pin, causing the motor to stop.

Table 1. Values from the Project (continued)

<table>
<thead>
<tr>
<th>Values Obtained by ADC</th>
<th>Degree of Tilt (Approximately)</th>
<th>Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>70.4</td>
<td>31</td>
</tr>
<tr>
<td>73</td>
<td>72.6</td>
<td>24</td>
</tr>
<tr>
<td>74</td>
<td>74.8</td>
<td>17</td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2. Example Values

<table>
<thead>
<tr>
<th>Desired ADC values</th>
<th>Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>45</td>
<td>220</td>
</tr>
<tr>
<td>50</td>
<td>185</td>
</tr>
<tr>
<td>55</td>
<td>150</td>
</tr>
<tr>
<td>60</td>
<td>115</td>
</tr>
<tr>
<td>65</td>
<td>80</td>
</tr>
<tr>
<td>70</td>
<td>45</td>
</tr>
<tr>
<td>75</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 8. Extrapolated Values from the Data Table
The PWM value determines the highest speed a motor can reach and, in this case, the ConvertedValue value.

Configure MTIM — In this section, the MTIM module is configured, and the ACMP module is disabled. A subtraction is executed in this section: 255 – ConvertedValue. The result is saved in the complement variable. This variable is used to turn off the motor in the PWM routine.

User constants — Reserved memory sections that cannot be modified.

User variables — Include values, erase, etc., that can be changed during the program.

Macro definitions — Here is where the internal clock source (ICS) is configured.

Configures port A — This section configures the port A pins. In this program PTA5, PTA4, and PTA1 are configured as outputs.

To understand the project code better, see the program-flow diagram (Figure 9).

---

**Figure 9. Program Flow**
4 How to Download the Program to Flash Memory

To download this project:
1. Open the CodeWarrior™ version 5.1 development tool.
2. Open AppNote.mcp file.
3. Select the option SofTec RS08.
4. Click on the make button.
5. Click on the debug button.
6. Select the DEMO9RS08KA2 board.
7. Click OK.

5 Conclusion

Electronic devices are easier to maintain and support than mechanical or gauge devices. You have better control with an electronic device than a mechanical one. This application note explained how to control a sewing machine’s motor speed with an MC9RS08KA microcontroller.

6 References


7 Glossary

AC — Alternating current
ACMP — Analog comparator
ADC — Analog-to-digital converter
COP — Computer operating properly (watchdog)
DC — Direct current
MCU — Microcontroller unit
MTIM — Modulo timer
PWM — Pulse-width modulation
RC — Resistor capacitor
Appendix A
Firmware

;*******************************************************************
;* MAIN.ASM                                                       *
;*******************************************************************
;* Copyright (c) 2007 Freescale Semiconductor                     *
;* http://www.freescale.com/                                      *
;*******************************************************************
;* Implementing a Sewing machine controller with a MC9RS08KA2     *
;* microcontroller. The speed motor depends of tilt pedal.        *
;*******************************************************************
;* Ulises Corrales Salgado                                       *
;* Applications Engineer                                          *
;* RTAC Americas                                                 *
;*******************************************************************

; Include derivative-specific definitions
   INCLUDE 'derivative.inc'

; export symbols

   XDEF _Startup
   ABSENTRY _Startup

;*****************************************************************************
;*                            User Constants                                 *
;*****************************************************************************
Table_Data EQU $3E00

; Variable declarations
ACMP_ENABLE      SET   $92
ACMP_DISABLED    SET   $20
MTIM_INIT        SET   $50
MTIM_ENABLE      SET   $60
MTIM_STOP_RESET  SET   $30
MTIM_128_DIV     SET   $07
FREE_RUN         SET   $00
DEBUG_MODE       SET   $00
RUN_MODE         SET   $01

MODE:            EQU   DEBUG_MODE

;*****************************************************************************
;*                            User Variables                                 *
;*****************************************************************************
;
; variable/data section
;
ORG    RAMStart         ; Insert your data definition here

ConvertedValue: DS.B   1            ; This variable store converted value
Complement      DS.B   1
Temp_Page       DS.B   1            ; Temporal backup Page
SensorReading   DS.B   1
PcBuffer        DS.B   2            ; Temporal backup SPC

ORG    ROMStart

;*****************************************************************************
;*                      MACRO DEFINITIONS                                   *
;*****************************************************************************

TRIM_ICS: MACRO                   ; Macro used to configure the ICS with TRIM
MOV   #$FF,PAGESEL              ; change to last page
LDX   #$FA                      ; load the content which TRIM value is store
LDA   ,x                        ; read D[X]
CBEQA #$FF,No_Trim              ; Omit the 0xFF value if $3FFA location content

; the value
STA   ICSTRM ; Store TRIM value into ICSTRM register
No_Trim:
ENDM

ENTRY_SUB: MACRO     ; Macro for "stacking" SPC
SHA
Glossary

STA PcBuffer + 2*(\1)
SHA
SLA
STA PcBuffer + 2*(\1) +1
SLA
ENDM

NOP ; Needs to separate MACROS

EXIT_SUB: MACRO ; Macro for restore SPC
    SHA
    LDA PcBuffer + 2*(\1)
    SHA
    SLA
    LDA PcBuffer + 2*(\1) +1
    SLA
ENDM

;**********************************************************************
;*                      CONFIGURES PORT A                             *
;**********************************************************************
PortA:
    MOV #HIGH_6_13(PTAPE), PAGESEL
    MOV #$FE, MAP_ADDR_6(PTAPE) ; Enables internal Pulling device
    MOV #HIGH_6_13(PTAPUD), PAGESEL
    MOV #$04, MAP_ADDR_6(PTAPUD) ; Configures Internal pull up device in PTA ; 5
    MOV #$32, PTADD ; PTA5, PTA4 and PTA3 as outputs
    MOV #$00, PTAD

    RTS

;**********************************************************************
;*                      CONFIGURES SYSTEM CONTROL                    *
;**********************************************************************
Init_Config:
    IFNE MODE
MOV #HIGH_6_13(SOPT), PAGESEL
MOV #$01, MAP_ADDR_6(SOPT) ; Disables COP and enables RESET (PTA2) pin
ELSE
MOV #HIGH_6_13(SOPT), PAGESEL
MOV #$03, MAP_ADDR_6(SOPT) ; Disables COP, enables BKGD (PTA3) and RESET ; (PTA2) pins
ENDIF
CLR ICSC1 ; FLL is selected as Bus Clock
TRIM_ICS
CLR ICSC2
RTS

;****************************************************************************************************************************
;*          Analog Comparator Initial Configuration                       *
;****************************************************************************************************************************
ACMP_Conf:
MOV #ACMP_ENABLE,ACMPSC ; ACMP Enabled, ACMP+ pin active, Interrupt ;enabled, Rising edges detections
RTS

;****************************************************************************************************************************
;*            Modulus Timer Configuration for ADC                 *
;****************************************************************************************************************************
MTIM_ADC_Init:
MOV #MTIM_128_DIV,MTIMCLK ; Select bus clock as reference, Set prescaler with 64
MOV #FREE_RUN,MTIMMODE ; Configure Timer as free running
MOV #MTIM_STOP_RESET,MTIMSC
RTS

;****************************************************************************************************************************
;*                   Discharge Capacitor                              *
;****************************************************************************************************************************
Discharge_Cap:
BSET 1,PTADD ; Configure PTA1 as Output
BCLR 1,PTAD ; Start Capacitor discharging
LDA #$FE ; Set delay time
Glossary

Waste_time:
    DBNZA Waste_time ; Wait until Delay = 0
    RTS

_Startup:

    BSR Init_Config
    BSR PortA

NextValue:
    BSR MTIM_ADC_Init
    BSR Discharge_Cap
    BSR ACMP_Conf ; Configure ACMP+ and ACMP-
    MOV #MTIM_ENABLE,MTIMSC ; Timer Counter Enabled

mainLoop:

    WAIT ; Wait for ACMP interrupt
    BSET 4,MTIMSC
    LDA MTIMCNT
    STA SensorReading ; Store counter value
    MOV #HIGH_6_13(SIP1), PAGESEL
    BRSET 3, MAP_ADDR_6(SIP1),PWM ; Branch if ACMP interrupt arrives
    BCLR 7,ACMPSC
    BRA NextValue

PWM:

    MOV #MTIM_STOP_RESET,MTIMSC ; Stop and reset counter
    MOV #ACMP_DISABLED, ACMPSC ; ACMP Disabled, Clear Interrupt flag

    LDA SensorReading
    CMP #75
    BHS NextValue
    JSR LookupTable

;**********************************************************************
;*                     Configure MTIM                                    *

Implementing a Sewing Machine Controller with an MC9RS08KA2 Microcontroller, Rev. 1

14

Freescale Semiconductor
Activecounter:

MOV #$00,MTIMCLK ; Enables interrupt, stops and resets timer counter
MOV #$01,MTIMMOD ; MTIM modulo with 1 counts before interrupt.
MOV #$70,MTIMSC ; Selects internal clock as reference bus clock (4 MHz) with prescaler 1

MOV #MTIM_STOP_RESET,MTIMSC ; Stop and reset counter
MOV #ACMP_DISABLED, ACMPSC ; ACMP disabled, Clear interrupt flag

LDA #$FF
SUB ConvertedValue
STA Complement

BCLR 4,MTIMSC ; MTIM counter is Active
BRA PWMLoopOn

PWMLoopOn:
BRSET 7,MTIMSC,PWM_Isr_D ; Branch if timer interrupt pending
BRA PWMLoopOn

PWM_Isr_D:
BSET 5,MTIMSC ; Reset MTIM Counter, Clear overflow flag
BRA PWM_Set_D

PWM_Set_D:
BSET 4,PTAD ; Turn on led 4
DBNZ ConvertedValue,PWMLoopOn

PWMLoopOff:
BRSET 7,MTIMSC,PWM_Isr_D1 ; Branch if timer interrupt pending
BRA PWMLoopOff

PWM_Isr_D1:
BSET 5,MTIMSC ; Reset MTIM Counter, Clear overflow flag
BRA PWM_Clear

PWM_Clear:
BCLR 4,PTAD ; Turn off led 4
Implementing a Sewing Machine Controller with an MC9RS08KA2 Microcontroller, Rev. 1

Glossary

DBNZ Complement, PWMLoopOff
BRA NextValue

LookupTable:
LDA SensorReading
ROLA ; Getting 2 MSB
ROLA
ROLA
AND #$03
ADD #(Table_Data>>6) ; Page Calculating
MOV #PAGESEL, Temp_Page ; Backup actual page
STA PAGESEL ; Page Change
LDA SensorReading
AND #$3F ; Extract 6 LSB
ADD #$C0 ; Index to paging window
TAX
LDA ,x ; Load table result
STA ConvertedValue ; Store result
MOV #Temp_Page, PAGESEL ; Back Page
RTS

;****************************************************************************
;*                      Startup Vector                                      *
;****************************************************************************
ORG $3FFD
JMP _Startup ; Reset

;****************************************************************************
;*                             Data Table                                      *
;****************************************************************************
ORG Table_Data
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 87,80,73,66,59,52,45,38,31,24,17,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
Appendix B  Accelerometer Demo Board Schematic