M16C/62: D-A CONVERTER

1.0: Abstract (Author: Darren Ting)

The following article will discuss the functions of D-A Converter (DAC) for the M16C/62 microcontroller (MCU). Attached is a sample assembly program that demonstrates the digital to analog conversion. A discussion on the sample firmware will be detailed.

1.1: M16C/62: Introduction

The M16C/62 is a 16-bit microcontroller based on the M16C/60 Series CPU core technology that is geared toward a wide range of consumer applications such as security systems, PCs, pagers, industrial, and communications.

The M16C family features an impressive list of features including LCD, USB, 10-bit A/D converter, UARTs, Timers, DMA controllers, D/A converter, and large on-chip ROM and RAM and FLASH.

1.2: Hardware: MSA0654

MSA0654 board (figure 1) comes with a serial cable with RS232C driver that connects 9-pin D-SUB to 2.54mm pitch 10-pin. The software package includes (1) Compiler: KNC30WA (2) Control software: KD30 (3) Target board Flash Memory programming software

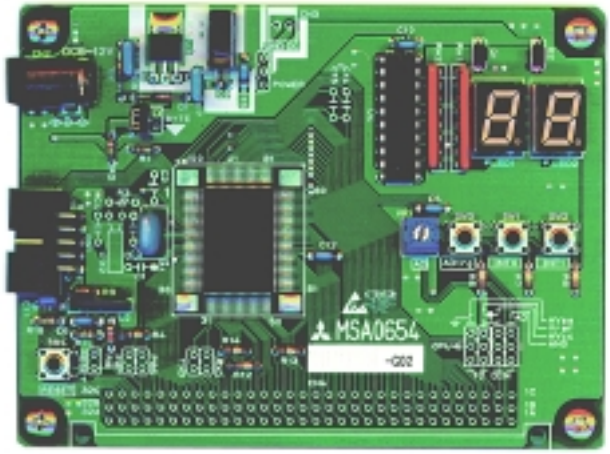


Figure 1—MSA0654 board

2.1: DAC: Introduction

D-A conversion takes a digital value and sends the output voltage that is proportional to the input value. M16C/62 contains two independent 8-bit, R-2R type D-A converters (see figure 2 for R-2R circuit diagram). DA0 register output goes to P9$_3$ and DA1 register output goes to P9$_4$.

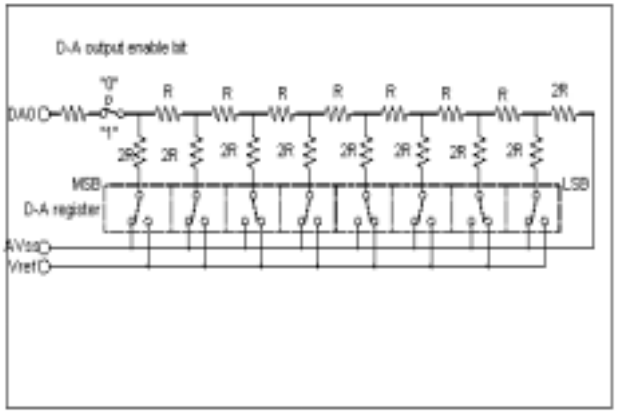


Figure 2—D-A converter equivalent circuit

In the D-A control register, bit 0 is DA0 output enable bit (DA0E) and bit1 is DA1 output enable bit (DA1E). Bit 2 to bit 7 are not assigned. Set bit to enable output and clear bit to disable output.

2.2: DAC: Description

D-A conversion is performed when a value is written to the corresponding D-A register. Bits 0 and 1(D-A output enable bits) of the D-A control register decide if the result of conversion is to be output.

Output analog voltage is determined by a set value in the D-A register.

$V = V_{REF} \times n / 256$ (n = 0 to 255)
V: output analog voltage
V_{REF}: reference voltage
n: set decimal value

3.0: Firmware: Implementation

The attached firmware is a sample assembly program that demonstrates the DAC function. This program generates an isosceles triangular output waveform.

The program is designed so that every 20us the voltage goes up 1/256 of V_{REF} (in this case, we set V_{REF} to 5v) from 0v for 255 times. At the end of the 255th increment (waveform going upwards), the waveform reaches its peak and the voltage starts to drop (waveform going downwards) 1/256 of 5v for

255 times until the voltage goes down to 0v again. The program is written in an infinite loop so that it displays output every 20us and produces a continuous waveform.

V_{REF} is 5v and n is set to be 255 (255 increments upward and 255 decrements downward so that the full range between 0v and 5v is used.

$V = V_{REF} X n / 256$

= (5v)(255 / 256)

= 4.98v

The time interval for each increment and decrements is set to be 20us.

T = (time interval)(# of increments + decrements)

= (20us)(255 + 255)

= 10.2ms

Thus, the triangular output waveform has a peak-to-peak amplitude of approximately 5v and period of 10.2ms (see figure 3 for the output waveform and verification).

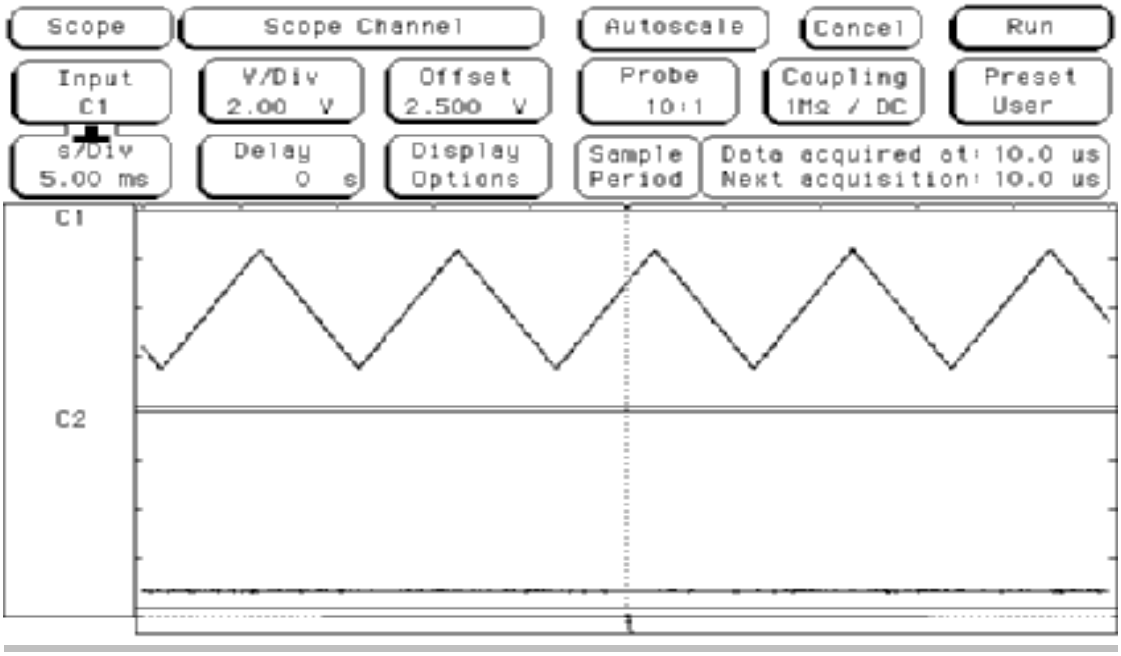
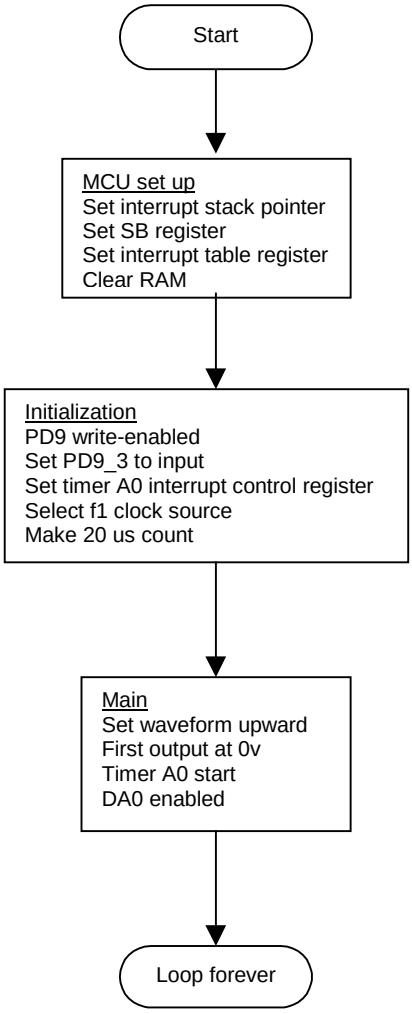
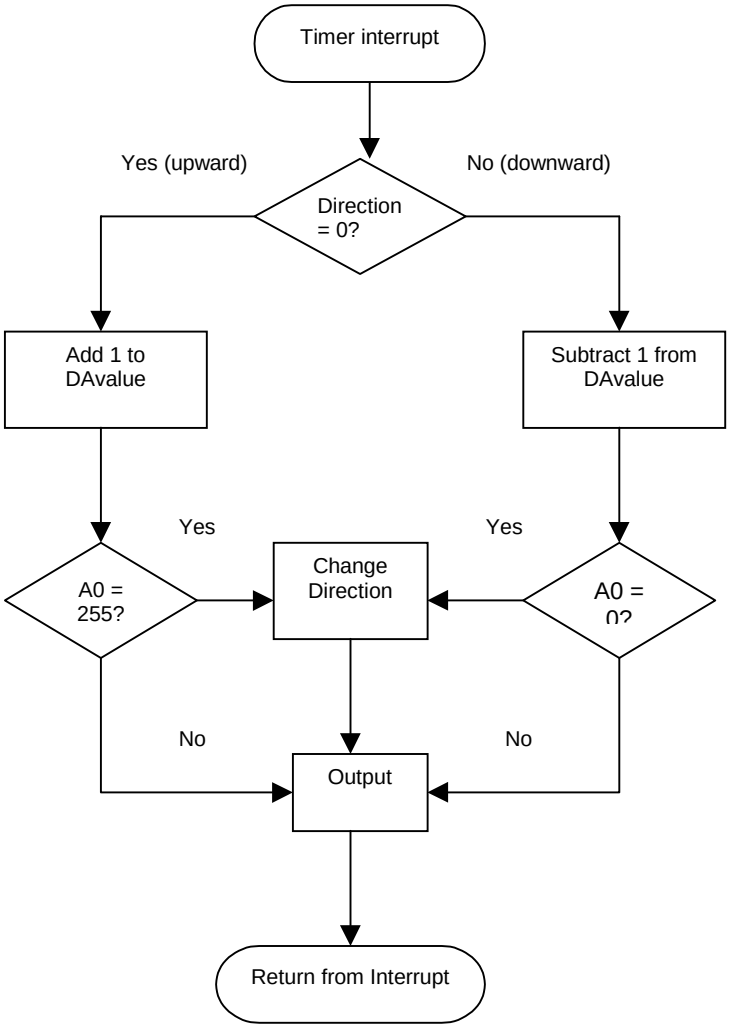


Figure 3—sample program output


Start
MCU set up
Set interrupt stack pointer
Set SB register
Set interrupt table register
Clear RAM
Initialization
PD9 write-enabled
Set PD9_3 to input
Set timer A0 interrupt control register
Select f1 clock source
Make 20 us count
Main
Set waveform upward
First output at 0v
Timer A0 start
DA0 enabled
Loop forever


Timer interrupt
Yes (upward)
No (downward)
Direction = 0?
Add 1 to DAvalue
Subtract 1 from DAvalue
Yes
Yes
A0 = 255?
Change Direction
A0 = 0?
No
No
Output
Return from Interrupt

```
;***********************************************************************
;*    Sample Program                                                   *
;*    Digital to Analog Converter                                      *
;*    By Darren Ting                                                   *
;*                                                                     *
;*    Note: This program was originally written using KDB30 debugger.  *
;*          The symbol definitions for KD30 are provided below.        *
;***********************************************************************
;
;     Copyright,1999
;     MITSUBISHI ELECTRIC CORPORATION AND
;     MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATIOON
;
;----- include of sfr file ------------------------------------

.list       off              ; don't draw up a program list
.include    sfr.inc          ; include of sfr file
.list       on               ; draw up a program list

;----- define of symbols for KDB30----------------------------------------

VramTOP          .equ  000400h          ; top address of internal RAM
VramEND          .equ  002BFFh          ; end address of internal RAM
VIstack          .equ  002C00h          ; stack pointer
VprogTOP         .equ  018000h          ; top address of program area
Vintbase         .equ  01FD00h          ; top address of variable vector table
Vvector          .equ  01FFDCh          ; top address of fixed vector table
SB_base          .equ  000380h          ; base address of SB relative

;***** DEFINE OF SYMBOLS FOR KD30 ***********************************************
;VramTOP          .equ  000400h          ; top address of internal RAM
;VramEND          .equ  002BFFh          ; end address of internal RAM
;VIstack          .equ  002C00h          ; stack pointer
;VprogTOP         .equ  0F0000h          ; top address of program area
;Vintbase         .equ  0FA000h          ; top address of variable vector table
;Vvector          .equ  0FFFDCh          ; top address of fixed vector table
;SB_base          .equ  000380h          ; base address of SB relative
;***********************************************************************

;----- keep of RAM area ---------------------------------------

.section    memory,data
.org        VramTOP

DAvalue:    .blkb 1                ; data for the DA output
direction:  .blkb 1                ; DA output waveform direction

.section    prog,code
.org        VprogTOP
.sb         SB_base                ; assigns provisional SB register value
.sbsym      DAvalue                ; place DAvalue in SB addressing mode
.sbsym      direction              ; place direction in SB addressing mode

;----- clear of RAM -------------------------------------------

reset:
      ldc         #VIstack,ISP           ; set Interrupt Stack Pointer
```

```
      ldc         #SB_base,SB             ; set SB register
      ldintb      #Vintbase               ; set Interrupt Table register

      mov.w #0,R0                         ; 0 clear
      mov.w #(VramEND+1-VramTOP)/2,R3     ; number of times
      mov.w #VramTOP,A1                   ; start address
      sstr.w

;----- initiallize ----------------------------------------------

      bset  PRC2                    ; P9 direction register write-enabled
      bclr  PD9_3                   ; set P9_3 direction register to input

      mov.b #00000001B,TA0IC        ; timerA0 interrupt control register set
      mov.b #0, TA0MR               ; f1 clock source
      mov.w #320,TA0                ; make 20us count down (200 for MSA0600)

      fset  I                       ; set flag

;----- main program ---------------------------------------------

main:
      mov.b #0,direction            ; upward

      mov.b #0,DAvalue              ; first output starts at 0v
      bset  TA0S                    ; timer A0 start
      bset  DA0E                    ; DA0 enabled

loop1:
      jmp   loop1

timer_int:                          ; 20us timer interrupt
      push.w A0                     ; push to stack
      mov.b  direction,A0           ; 0 = upward, 1 = downward
      cmp.b  #1,A0                  ; check direction
      jeq    downward

upward:
      add.b #1,DAvalue              ; 1 increment every 20us
      mov.b DAvalue,A0
      cmp.b #0ffh,A0                ; 255 increments?
      jne   output
      mov.b #1,direction            ; change direction to downward
      jmp   output

downward:
      sub.b #1,DAvalue              ; 1 decrement every 20us
      mov.b DAvalue,A0
      cmp.b #0,A0                   ; 255 decrements?
      jne   output
      mov.b #0,direction            ; change direction to upward

output:
      mov.b DAvalue, DA0            ; display output
      pop.w A0                      ; pull from stack
      reit
```

```
;----- vector table -------------------------------------------

.section    uninter,romdata
.org        Vintbase+(4*21); for TimerA0 interrupt
.lword      timer_int
.section    inter,romdata
.org        Vvector+(8*4)
.lword      reset

;----- program end --------------------------------------------

.end
```