# *Model Motor* CONTROLLER

## *Driving a 12V Motor through an ST7 series Timer with PWM Function*

## Introduction

There are many ways in which model motors, say for slotcar racing or model railways can be controlled. Electronics and Beyond has covered a few in the past, but this article looks at a different approach using a microcontroller. It is based on the ST7 series of chips from ST Microelectronics and suitable software control generates pulse width modulated waves PWM at its output. This signal is tuneable in frequency and its duty cycle. As an example this project uses is based on controlling a motor using the the ST72251G2 device to control a model motor.

## What is Pulse Width Modulation

Pulse Width Modulation enables us to generate a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers in the chip. As the heading suggests you can vary the width of the pulse within the cycle.

The pulse width modulation uses the complete output compare 1 function and the OC2R register.

## Mark-to-Space Ratio and Duty Factor

Figure 1 below shows two pulse waveforms which have the same time period T, and hence the same pulse repetition frequency, but yet are clearly different from one another. The difference between them can be expressed in two different ways:

## Mark-to-Space Ratio

If the duration of each positive voltage is known as a mark and each period for which there is either no pulse or there is a negative pulse is known as a space, then the mark-to-space ratio is the ratio of mark time/space time.

For a square waveform, as shown in the first diagram of Figure 1 above, the mark-to-space ratio is unity. For the second waveform in Figure 1 the mark-to-space ratio is given by (6-5)/(15-6) or 1/9.

## Duty Factor

The duty factor of a pulse waveform is the ratio (mark time)/(mark time + space time). For a square waveform the duty factor is equal to 1/2, and for the second waveform in Figure 1, the duty factor is given by (6-5)/(15-5) or 1/10.

The average, mean, or d.c. value of a pulse waveform is equal to its amplitude, or peak value times the duty factor. Average Output Voltage Vav = Vpk x Duty Factor So in the example above if Vpk = 12V Output would be 6V and 1.2V respectively

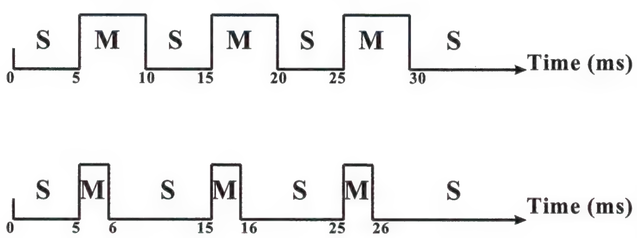



**Figure 1. Two pulse waveforms with different duty factors.**

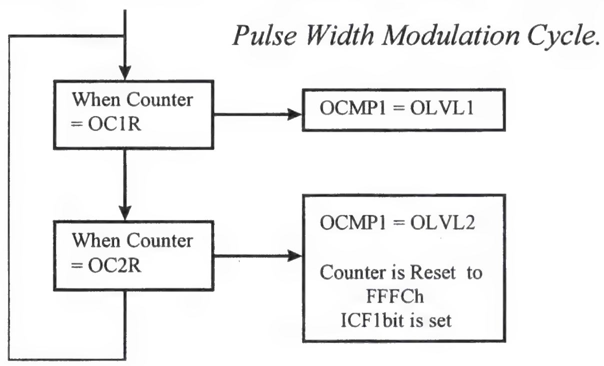



**Figure 2.**

## Loadinng up Procedure

To use the ST72251 microcontroller in pulse width modulation mode first of all:

1. Load the OC2R register with the value corresponding to the period of the signal.
2. Load the OC1R register with the value corresponding to the length of the pulse if (OLVL1=0 and OLVL2=1).
3. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with OC1R register.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with OC2R register.
4. Select the following in the CR2 register:
   - Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   - Set the PWM bit.
   - Select the timer clock (CC1-CC0).

If OLVL1=1 and OLVL2=0 the length of the pulse is the difference between the OC2R and OC1R registers. The OC iR register value required for a specific timing application can be calculated using the following formula:

$$\text{OCiR Value} = \frac{t \cdot f_{CPU}}{t_{PRESC}}$$

Where:
- t = Desired output compare period (seconds)
- f CPU = Internal clock frequency
- t PRESC = Timer clock prescaler

The Output Compare 2 event causes the counter to be initialised to FFFCh.

## Model Motor control

Note: After a write instruction to the OCiHR register, the output compare function is inhibited until the OC iLR register is also written. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared. Therefore the Input Capture 1 function is inhibited but the Input Capture 2 is available. The OCF1 and OCF2 bits cannot be set by hard-ware in PWM mode therefore the Output Compare interrupt is inhibited. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.

## Voltage Setting (Duty Cycle)

The output power supply voltage is controlled by the duty cycle of the generated PWM signal. The OC1R register in the chip controls each duty cycle percentage of the timer PWM mode. Timer A Output Control Register 1, and 2 are both 16 bit registers, with 8 bits set in a low register and the other 8 bits set in the high register. For the purpose of this project the Timer A Output Control High Registers were set to 00 hex at all times. This was because an accurate output could be achieved by only using the low registers. The low output control of register 2 was also set to a constant value of FF hex throughout the project. This was set to a constant level in order to set a maximum pulse period equal to that of FF hex. Timer A Output Control register 1 was used throughout the project to input different values. These values were defined as "bytes in memory" at the beginning of the program.

For waveform 1: Duty Factor = (Mark Time) / (Mark Time + Space Time)

$$= \frac{60 \times 10^{-6}}{60 \times 10^{-6} + 200 \times 10^{-6}}$$

$$= \frac{60 \times 10^{-6}}{260 \times 10^{-6}}$$

$$= 0.231$$

For waveform 2: Duty Factor = (Mark Time) / (Mark Time + Space Time)

$$= \frac{200 \times 10^{-6}}{200 \times 10^{-6} + 70 \times 10^{-6}}$$

$$= \frac{200 \times 10^{-6}}{270 \times 10^{-6}}$$

$$= 0.741$$

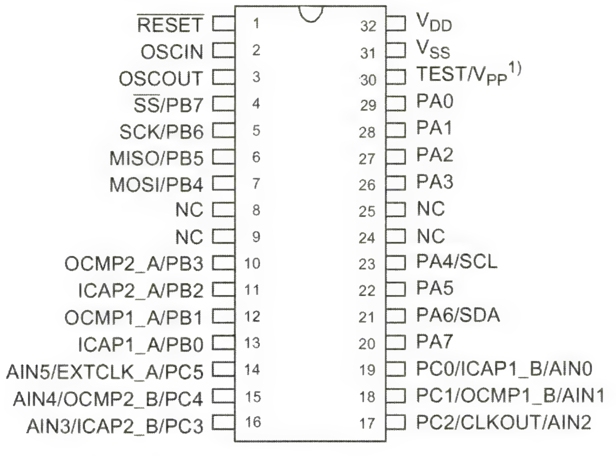



**Figure 2a. ST72251 SO28 Package**

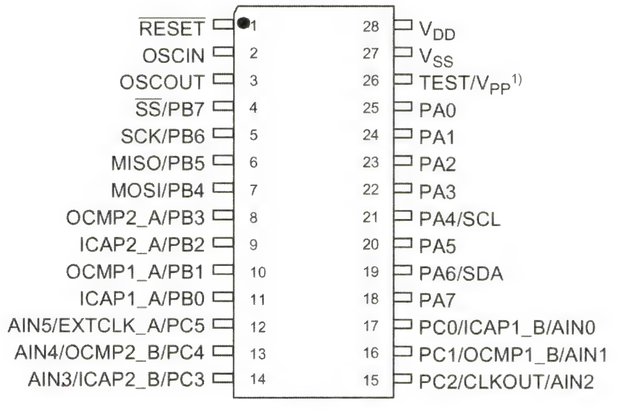
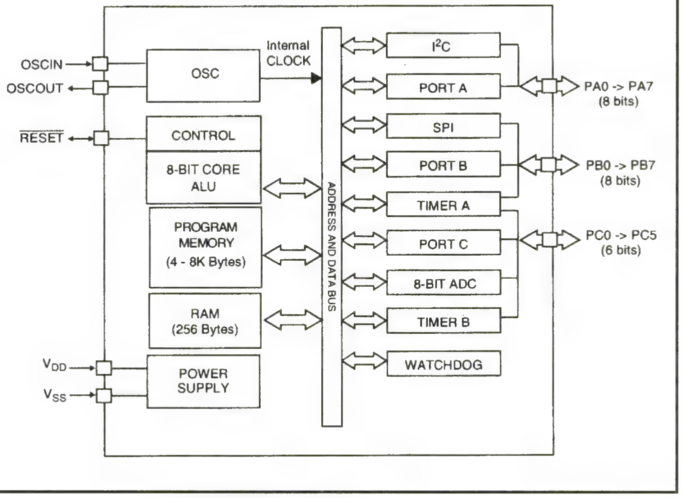



**Figure 2b. ST72251 SO28 Package**

**Figure 3. Block diagram of ST225lG2**



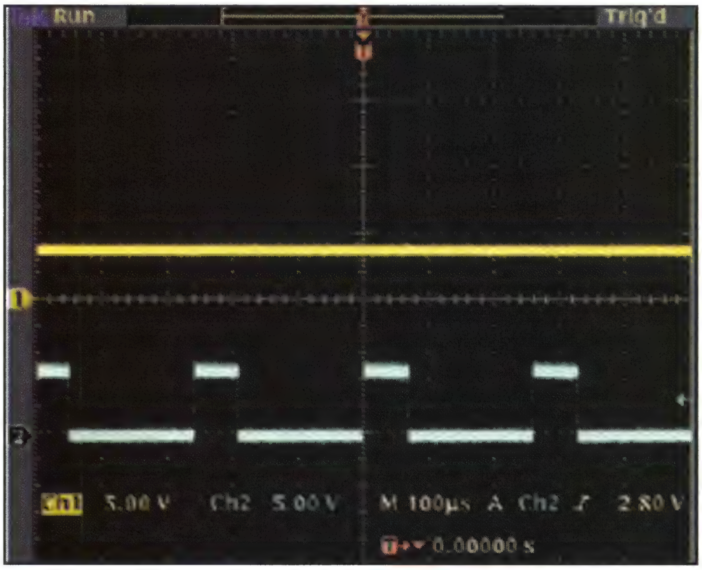

The waveforms shown here give the output PWM voltage that was recorded from the Output Compare 1 of Timer A. The project used five different settings altogether which ranged from an off state, to 100%. The two waveforms that are shown here are that of the 25% setting, and the 75% setting.

If each square going across the oscilloscope screen represents 100? seconds, and hence every mark within that square represents 20? seconds the following can be concluded:

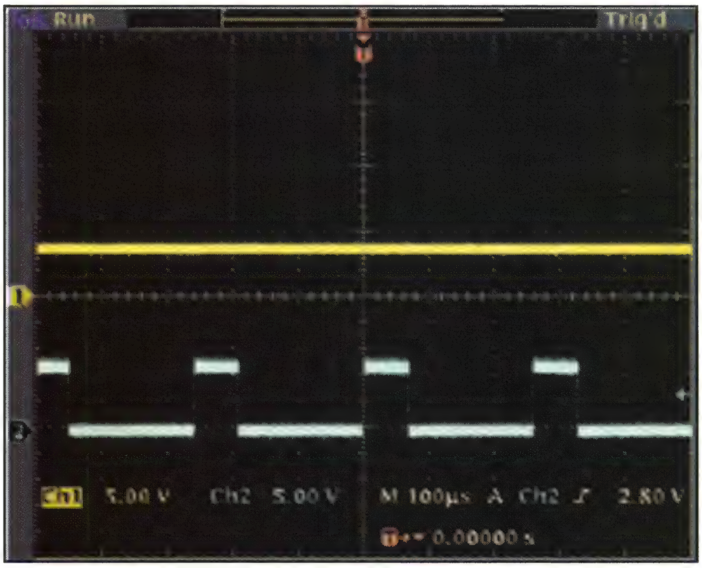
If we multiply the amplitude of the waveform which is 5 volts by the duty factor, we will get the overall voltage we require:

Average Voltage = 5 volts x 0.231 = 1.155, which is approximately 25%.

If we multiply the amplitude




**Waveform 1. 25% Duty Cycle.**




**Waveform 2.**

of the waveform which is 5 volts by the duty factor, we will get the overall voltage we require:

Average Voltage = 5 volts x 0.741 = 3.71 volts, which is approximately 75%.

## Simple Amplifier Circuit

The above shows that the ST7 microcontroller, like many other microcontrollers is only capable of supplying a maximum of 5 volts. However in order to control model cars or railways a voltage range of between 0 and 12 volts is required. This can be achieved by inserting the output OCMP1_A of the ST7 device into an amplifier circuit such as the one shown below in Figure 2.

The first part of the amplifier shows an RC network, which is used to filter the input signal to the amplifier. The operational amplifier used is the L272 op. amp. This is capable of supplying an output current of approximately 1 amp if the one side of the device is adequately soldered to a heat sink. However for supplying a motor we need to give the output a bit more current capacity, hence the two transistors inserted into the feedback path of the circuit in a Darlington configuration. The Darlington pair will supply a current gain approximately equal to the product of the individual betas of the transistors. The Darlington circuit is normally required when a high current or a high input impedance is needed. R3 and R4 of the circuit shown above in Figure 3 set the gain of the circuit in the feedback path, the gain set by the above amplifier is 2.5. Therefore if a voltage of 5 volts enters the amplifier the output will give the required 12 - 12.5 volts. In the diagrams above waveform 1, and waveform 2 the yellow (above) waveform shows the dc level that was output from the amplifier circuit. It should be noted that these are approximately 2.5 times greater than the average PWM input voltage.

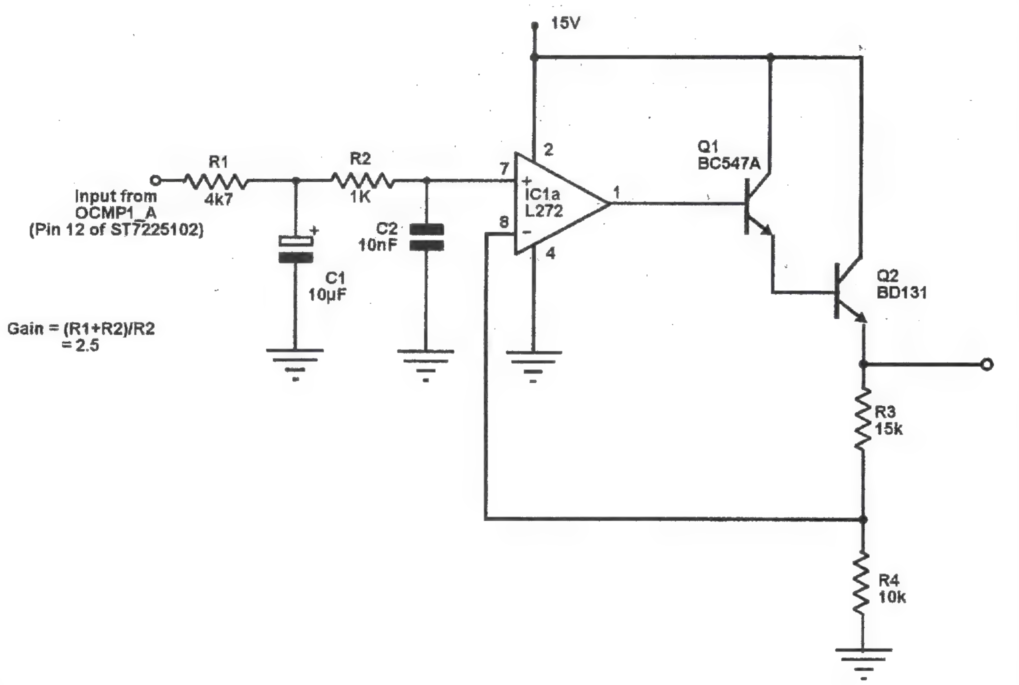



**Figure 3. Amplifier Circuit with a gain of 2.5.**

## Appendix - Software

```
;ST7 Setup File - Created by Application Builder
; ** Devicename is set by the project and tells the Assembler
; ** which device definition file to use.  To change device, use
; ** Project-Configure-Device
; Setup File Created with Device Set to ST72212G2

.Include $DeviceName   ; Include Device Definition File (ST72212G2.str)
segment 'Rom'     ; Code segment - Address defined in Definition file (.str)

Start:     ; Reset Label
           ; ********* Timer B Setup Code
      ; *Timer A Initial Value is always $FFFC
      ; *Timer A Control Register 1
      ld a, #$04; IC1 Disabled
      ld TACR1, a ; Forced Compare 1 Off Forced Compare 2 Off
      ; Interrupts : Input Capture Disabled

      ; *Timer A Control Register 2
      ld a, #$98    ; Output Compare 1 Disabled
      ld TACR2, a ; IC2 Disabled  Clock = Fcpu/2

      ; *Timer A Output Compare Register 1
      ld a, #$00    ; PWM On - Pulse Length
      ld TAOC1HR, a          ; High Byte
      ld a, #$32    ;SETS THE PULSE
      ld TAOC1LR, a          ; Low Byte

      ; *Timer A Output Compare Register 2
      ld a, #$00    ;PWM On - Pulse Period
      ld TAOC2HR, a          ; High Byte
      ld a, #$ff    ;SETS THE PERIOD
      ld TAOC2LR, a          ; Low Byte

      ; ********** Port A Setup Code
      ld a, #$00    ; Bit On = Output, Bit Off = Input
      ld PADDR, a; Port A Direction Register
      ld a, #$ff ; Option Selects Input or Output Type
      ld PAOR, a ; Port A Option Register

      ; ********** Port B Setup Code
      ld a, #$FF ; Bit On = Output, Bit Off = Input
      ld PBDDR, a ; Port B Direction Register
      ld a, #$FF ; OCMP1A,
      ld PBOR, a ; Port B Option Register
      ; ********** Port C Setup Code
      ld a, #$ff ; Bit On = Output, Bit Off = Input
      ld PCDDR, a ; Port C Direction Register
      ld a, #$ff ; OCMP1B,
      ld PCOR, a  ; Port C Option Register

      ; ********* Control Registers
      ld a, #$00 ; Main Clock Out Off
      ld MISCR, a  ; Normal Mode : Fcpu = 8000000Hz
      ld a, #$FF ; Stack Pointer Low Byte
      ld s, a ; Stack Pointer High Byte set by hardware
      ld a, #$7F ; Watchdog Control Register
      ld WDGCR, a ; Watchdog Disabled
      ;rim ; Global Interrupt Enable

      ;ST7 Main File - Created by Application Builder
      ; Author :  Tony Ashton
      ; Company : Kanda Systems
      ; Comment : Power Supply Using PWM
      ; ** includes all the setup code for this project.  The default file
      ; ** extension is .STS. Renaming this setup file will also update the
      ; ** Project so the assembler will always find it

      ; File Created with Device Set to ST72212G2
      .include $SetupFileName ; Set up file (.sts) is included here
      segment 'Ram0'  ; RAM Page Zero segment - Always at ADDRESS $80-$FF
```

```
;Define user variables here if required -
Example Syntax is
;COUNTER DS.B 1
;COUNTER2 DS.W 1
;PIN_NUMBER EQU 6

segment 'Rom'
    ; Code segment - Address defined in
    device file (.str)
pwm_00l dc.b $00     ;loads 00 into memory
                     location
pwm_00h dc.b $00     ;loads 00 into memory
                     location
pwm_25l dc.b $34     ;loads 03 into memory
                     location
pwm_25h dc.b $00     ;loads 00 into memory
                     location
pwm_50l dc.b $7a     ;loads 16 into memory
                     location
pwm_50h dc.b $00     ;loads 00 into memory
                     location
pwm_75l dc.b $bd     ;loads 24 into memory
                     location
pwm_75h dc.b $00     ;loads 00 into memory
                     location
pwm_100l dc.b $ff    ;loads 34 into memory
                     location
pwm_100h dc.b $00    ;loads 00 into memory
                     location
offl dc.b $01        ;loads 04 into memory
                     location
offh dc.b $00        ;loads 00 into memory
                     location

main:     ;Main Code here
ld a,#$FF            ;loads FF hex into the
                     accumulator
ld pbdr,A            ;loads FF hex into port a
                     direction register
ld a,pwm_00l         ;loads 00 into accumulator
ld x,pwm_00h         ;loads 00 into register x
call time            ;calls time sub routine
main_1
nop                  ;no operation
nop                  ;no operation
btjf padr,#0,off     ;bit test and jump to off
                     routine
btjf padr,#4,percent_50
                     ;bit test and jump to 50 %
btjf padr,#3,percent_25
                     ;bit test and jump to 25 %
btjf padr,#5,percent_75
                     ;bit test and jump to 75 %
btjf padr,#6,percent_100
                     ;bit test and jump to 100 %
call del   ;calls delay sub routine
jra main_1           ;returns to main_1

off:
*************NOVOLTAGE....OFF***************
t0
nop                  ;no operation
nop                  ;no operation
call del             ;calls delay sub routine
bres pbdr,#0         ;turns on led on bit 0 of
                     pbdr
btjf padr,#0,t0      ;debounce routine
ld a,offl            ;loads 01 into accumulator
ld x,offh            ;loads 00 into register x
bset pbdr,#0         ;turns led on bit 0 of pbdr
off
call time            ;calls time sub routine
jp main_1            ;jumps to main_1

percent_100:
**************100%SUB_ROUTINE*************
t1
nop                  ;no operation
nop                  ;no operation
call del             ;calls delay sub_routine
bres pbdr,#6         ;turns on led on bit 6 of
                     pbdr
btjf padr,#6,t1      ;debounce routine
ld a,pwm_100l        ;loads ff into accumulator
ld x,pwm_100h        ;loads 00 into register x
bset pbdr,#6         ;turns led on bit 4 of pbdr
                     off
call time            ;calls time sub routine
jp main_1            ;jumps to main_1
percent_50:
****************50%SUB_ROUTINE***********
t2
nop                  ;no operation
nop                  ;no operation
call del             ;calls delay sub_routine
bres pbdr,#4         ;turns on led on bit 4 of
                     pbdr
btjf padr,#4,t2      ;debounce routine
ld a,pwm_50l         ;loads 7a into accumulator
ld x,pwm_50h         ;loads 00 into register x
bset pbdr,#4         ;turns led on bit 4 of pbdr
                     off
call time            ;calls time sub routine
jp main_1            ;jumps to main_1

percent_75:
**********75%SUB_ROUTINE*****************
t3
nop                  ;no operation
nop                  ;no operation
call del             ;calls delay sub_routine
bres pbdr,#5         ;turns on led on bit 5 of
                     pbdr
btjf padr,#5,t3      ;debounce routine
ld a,pwm_75l         ;loads bd into accumulator
ld x,pwm_75h         ;loads 00 into register x
bset pbdr,#5         ;turns led on bit 5 of pbdr
                     off
call time            ;calls time sub routine
jp main_1            ;jumps to main_1

percent_25:
************25%SUB_ROUTINE****************
t4
nop                  ;no operation
nop                  ;no operation
call del             ;calls delay sub_routine
bres pbdr,#3         ;turns on led on bit 6 of
                     pbdr

btjf padr,#3,t4      ;debounce routine
ld a,pwm_25l         ;loads 34 into accumulator
ld x,pwm_25h         ;loads 00 into x register
bset pbdr,#3         ;turns led on bit 6 of pbdr
                     off
call time            ;calls time sub routine
jp main_1            ;jumps to main_1

*************DELAYROUTINE******************
DEL:
push x               ;pushes x onto the stack
ld x,#$02            ;push x onto stack
DEL1:
dec x                ;decrements value of x
jrne del1            ;jumps to del1 if value not
                     equal
pop x                ;pops x from the stack
ret         ;return from del subroutine

time: ;*Timer B Output Compare Register 1
ld TAOC1HR, x      ; High Byte, loads current
value in x register into TAOC1HR
ld TAOC1LR, a      ; Low Byte, loads current
value in accumulator into TAOC1LR ret
; Interrupts in use, so .STV File called here
; This file contains shell Interrupt Service
Routines (ISR)
; and Interrupt Vector Table.  Add Your ISR
Code to this file

.include $VectorFileName  ; Interrupt and
Vector file (.stv)
end
; ST7 Vector File - Created by Application
Builder
; *********** Interrupt Service Routines
; File Created with Device Set to ST72212G2

segment 'Rom'     ; Code segment - Address
defined in Definition file (.str)

TimerA:  ; ******** TimerA Interrupt ISR Code
Here

TAOC1:  ; Test OCF1 Flag (TASR.6) iret
TAOC2:  ; Test OCF2 Flag (TASR.3) iret

; ******** End Timer A Interrupt

segment 'Vectit'    ; Vector segment - Address
defined in Definition file (.str)
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
TimA:   DC.W    TimerA
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
        DC.W    0
reset:  DC.W    Start

        end
```

***This software routine will be available on the magazine website: <electronicsandbeyond.com.>***

## PARTS LIST

**For Simple Amplifier Ciruict**

**RESISTORS:**
R1 4K7
R2 1K
R3 15K
R4 100K

**CAPACITORS**
C1 10$\mu$F
C2 10nF

**SEMICONDUCTORS**
TR1 BC547
TR2 BD131
IC1 LL272

**MICROCONTROLLER**
ST72251G2
ST Microelectronics