Autonomous Vehicular Trailer

EE 452 Final Project

Spring 2012

Conor Cahill, Steven Hopkins, and Adam Morse

Abstract

Today's vehicles with cruise control typically lock the speed to a fixed set point. Problems arise when the speed of the lead traffic varies, causing the drivers to have to manually adjust the set speed. A simple addition of a distance sensor opens up the possibility to follow the leading vehicle and maintain a fixed safe distance. This project implements one such method of following a lead vehicle using an ultrasonic distance sensor to measure the distance. A microcontroller plus code contain the algorithm necessary to adjust the power of the motors to maintain the proper distance. In the future, this algorithm could be expanded to take other variables into account, such as requested speed and collision avoidance.

Table of Contents

Introduction

An Autonomous Vehicular Trailer is a vehicle designed to be a trailer that follows behind a lead vehicle adjusting its velocity as necessary to maintain its relative position with the lead vehicle. This trailer is autonomous in that there are no direct communications/controls from the lead vehicle to the trailer. The trailer itself senses the location of the lead vehicle and adjusts its own velocity to automatically follow behind the lead vehicle.

We will construct the trailer using a kit-based tracked vehicle with two DC motors (one for each track) connected to a motor control unit that is, in turn, controlled by our PIC18 microcontroller. In addition we plan to use an ultrasonic sensor to determine the distance between the trailer and the lead vehicle and LED indicators to represent “hazard” lights.

The trailer will only adjust itself to movement in a straight line direction. It will not turn to follow the lead vehicle should the lead vehicle turn. If the trailer does not sense the lead vehicle, or loses track of the lead vehicle, it will stop and turn on its hazard lights and await the return of the lead vehicle.

The software control will continuously monitor the distance between the lead vehicle and the trailer and will use this information to also calculate the relative velocity and relative acceleration. In addition separate wheel sensors will be used to monitor the absolute velocity of the trailer. The absolute velocity of the trailer will be used to define a target distance range for the trailer (getting higher as the velocity increases to allow for additional response distance).

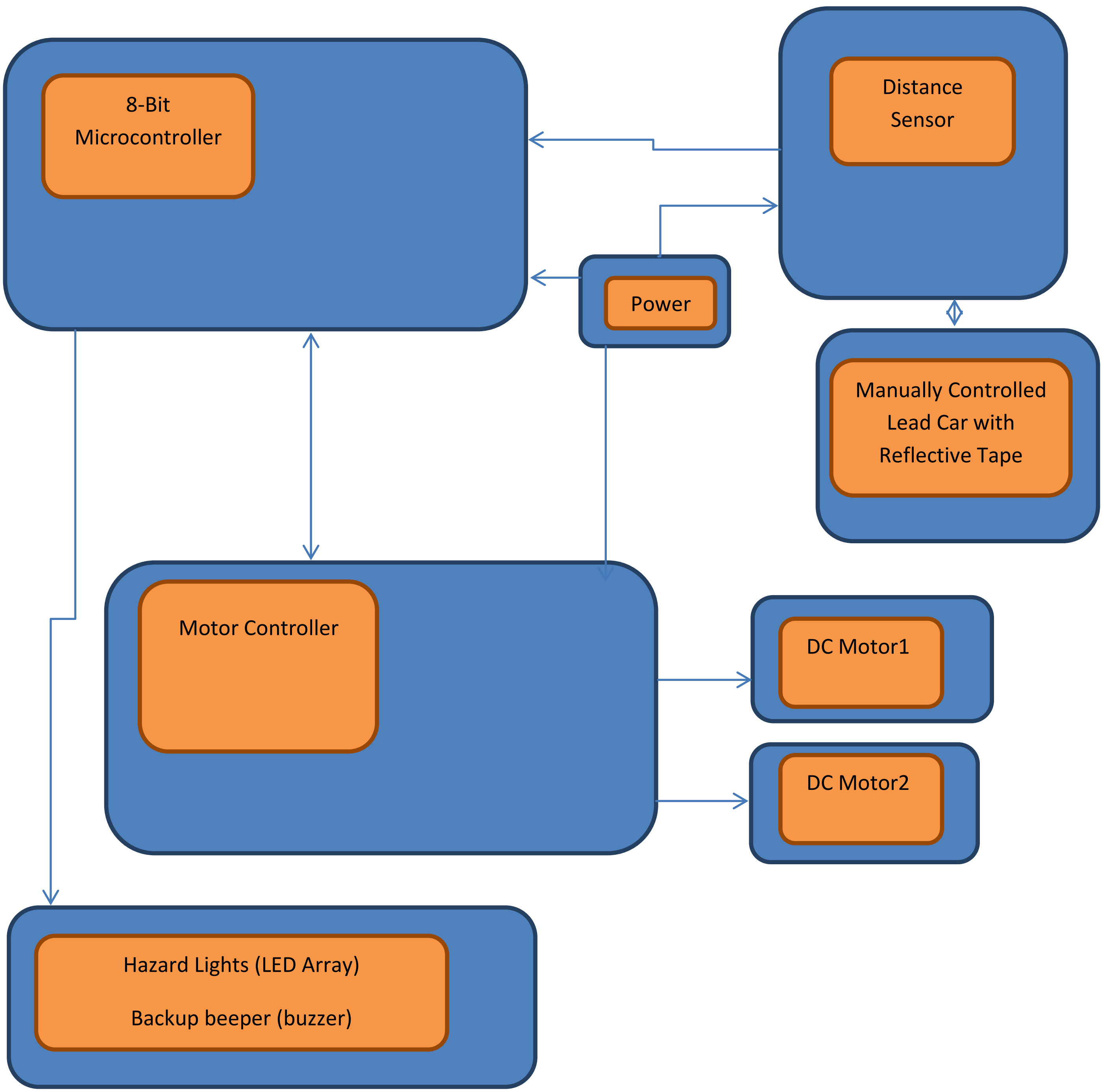
The distance, relative velocity and relative acceleration will be used to control the power provided to the motors in order to control the acceleration of the trailer in an efficient manner. We plan to use a proportional-integral-derivative (PID) algorithm to determine the amount of power to send to the motors in order to maintain the appropriate distance from the lead vehicle. The motors will also be used to provide braking force when necessary in order to maintain the appropriate separation distance when the lead vehicle slows abruptly.

We plan to support both positive and negative velocity so that the lead vehicle can move forward and/or backward and the trailer will maintain the appropriate separation distance.

Technical Approach

Block Diagram

The block diagram provides a high-level, graphic illustration of the components of the Autonomous Trailer. The arrows connecting the blocks illustrate a flow of information. The discrete elements of the system are the object to be followed (Lead Car or similar provided by user), an ultrasonic distance sensor, a microcontroller, a motor control board, the motors that drive the vehicle and the hazard lights/buzzer. Power is provided to the distance sensor, the microcontroller and the motor controller.

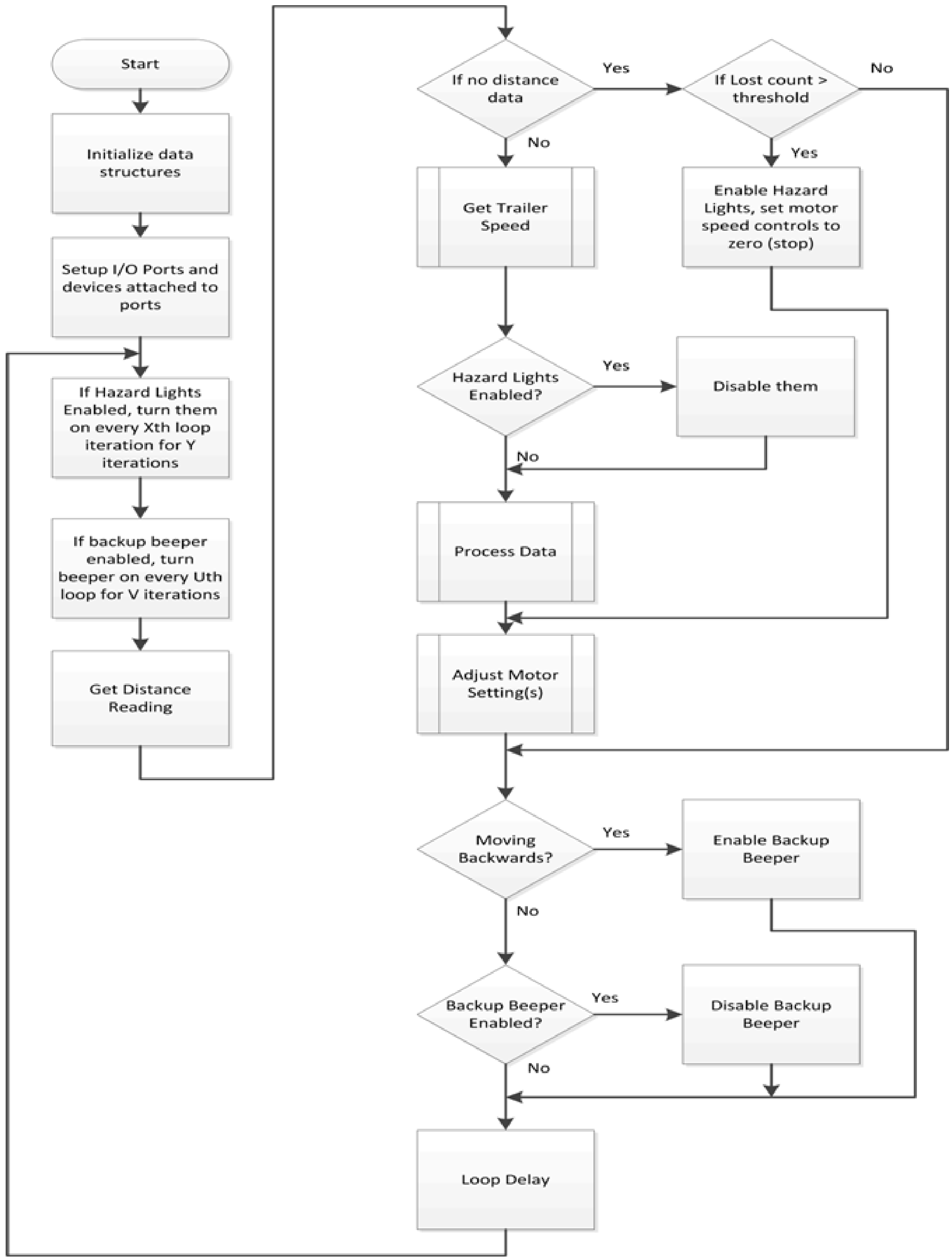


Software Systems

Algorithm

We have broken our algorithm and functionality into several pieces of functionality tied together by our main function. The algorithm for each component is discussed separately.

Main Loop



Sensor Control, Pseudo-code

```
; Algorithm for initializing and reading ultrasonic distance sensor
;
; Parameters:  None
;
; Outputs:     WREG and ADRESL will contain a regularly updated 8-bit value representing distance
; in cm (1 bit per cm)
;
; Global Data: Oscillator will be configured to 16MHz
;
; Persistent Data: When bits 0 or 1 ADRESH are set, this indicates a distance sensor error, and
; error code WREG=0xFF
;                                    is given
;
; Temporary Data:
;
; Algorithm
;**********************************************************************
;
; Initialize Sensor
       initSensor() {
              ;Set TRISA (pin RA2) to input and ANSELA (pin RA2) to configure analog

              TRISA <-- set bit 2
              ANSELA <-- set bit 2

              ;Configure ADCON0, ADCON1,ADCON2

              ;ADCON0 (comments move right to left from LSB, ignoring unused bits)
              ;ADON<0> -- 1 -- initialize ADC
              ;CHS<4,0> -- 00010 -- Channel select AN2
              ADCON0 <-- B'00001001'

              ;ADCON1
              ;PVCFG <1,0> -- 00 -- positive voltage reference set to MCU internal 5V
              ;NVCFG <1,0> -- 00 -- negative voltage reference set to MCU internal 0V
              ADCON1 <-- B'00000000'

              ;ADCON2
              ;ADCS<2,0> -- 010 -- Assume TAD>=1.6microsecond (from lecture),
              ;and MCU 16MHz device frequency, therefore use Fosc/32 as the ADC clock
              ;source (giving a period of 2microsecond, p299)
              ;ACQT<2,0> -- 100 -- If we assume maximum temp of 50C, utilizing TACQ
              ;formula, there is a still good deal of variation depending on the sensor
              ;impedance (unknown at this point). But based on a 2.5kohm impedance, the
              ;text (p577) calculates TACQ=~13microsec.  To be conservative, we can use
              ;8*TAD
              ADFM -- 1 -- right justify results in ADRESL/ADRESH
              ADCON2 <-- B'10100010'
       }
```

```
;Read Sensor Value
        readSensor() {
                ; Begin acquisition/conversion sampling loop
                ;
                ; Start conversion
                ADCON0 <-- GO ; set GO bit
                ;
                ; Test if conversion is finished before proceeding
                If ADCON0 <GO/DONE> = 1
                (
                        Return to test again with branch loop
                }
                ;
                ; Test for system error by testing ADRESH value
                ; If bits 0 or 1 are set, then WREG must give error code 0xFF
                ; If there is no error, place ADRESL into WREG as returned value

                If ADRESH <0> = 1 {  ; poll for errors in the first bit of ADRESH
                        WREG <-- 0xFF ;        set error code
                }
                Else {
                        WREG <-- ADRESL
                }
                If ADRESH <1> = 1 {  ; poll for errors in the first bit of ADRESH
                WREG <-- 0xFF ;        set error code
                }
                Else {
                WREG <-- ADRESL
                }
                ; Delay 2TAD before returning to readSensor start

        }
```

Motor Control, Pseudo-code

```
;
; MotorControl.txt - pseudocode for MotorControl functions
;
MotorInit  - called to initialize the Motor Control system
  ;
  ; We are using a PWMs to control each motor and need to
  ; configure them for our use and for our motors
  ;
  Set the period Register for Timer 2 to be 66 (which gets us a 15Khz PWM freq)
  Configure CCP1 and CCP2 on Port C for output
  Enable Timer2 and set it's pre-scaler to 4 (so our input is (16M/4)/4 or 1Mhz)
  Associate Timer2 with PWM1 and PWM2
  Enable PWM Mode on CCP1 and CCP2
  Configure pins 7-3 on PortC for output
  Set the motor to off

end

MotorUpdate  - called to update the motor settings

  ;
  ; Parameters:       Arg1 - the desired motor power percentage (0-100)
  ;                          Arg2 - the desired direction (1=reverse, 0=forward)
  ;
  Set the motor duty cycle to the value from Arg1 on Motor 1 and Motor 2
  If the desired direction is forward
    set MotorSettings to Forward
    clear Reverse status indicator
  else
    set MotorSettings to Reverse
    set Revers status indicator
  endif
  write the new MotorSettings to PortC
end
MotorEmergencyStop - called when we've lost track of Lead Vehicle
  Set motor duty cycle to zero for both Motor 1 and Motor 2
  Set MotorSettings to OFF
  write the new MotorSettings to PortC
end
```

Process Data, Pseudo-code

```
;
; ProcessData.txt - algorithms for ProcessData subroutines
;

ProcessDataInit - called at start of program and to reset ProcessData
  Initialize ProcessData variables to zero
end

ProcessDataLost - called when lead vehicle is lost
  Call ProcessDataInit to initialize variables
end

ProcessData
  Save parameter to SensorDist
  Get our previous return values and set new return values to same values

  if the motors are stopped
    if the lead vehicle is beyond target range
      save distance to target (SensorDist-TargetHigh) in DistOff
      set direction to forward
    else if the lead vehicle is below target range
      save distance to target (TargetLow-SensorDist) in DistOff
      set direction to reverse
    else (the vehicle is within target range)
         set NewMotorPower to zero
    endif

    if we are adjusting power (beyond or below)
      ;
      ; now to adjust power
      ;
      If we are within the low power adjustment range (< 5cm)
        set NewMotorPower to (1*DistOff) (e.g. 1%/cm)
      else
        set NewMotorPower to (2*DistOff) (e.g. 2%/cm)
      endif

      if NewMotorPower is greater than max allowed (20%)
        set NewMotorPower to 20%
      endif

    endif

  else (the trailer is moving)

    ;
    ; calculate some needed values for our operations/comparisons
    ;
    set DistChange to how far we've moved since last call
    set DistHigher flag to tru if our new distance is greater than prev dist
    set DistZero   flag if our new distance is exactly the same
    set DistOff    to the difference between our current distance and target distance
    set TgtBelow   if we are below target distnce
    set TgtExact   if we are exactly the target distance
    set CloserTgt  if we are closer to target than we were before

    if we are not on target or distance has changed

      if moving forward

        if we are too close (TgtBelow == true)
          set ReducePower flag
        esle if we aren't closer than we were before
          set ReducePower flag
             else if we aren't catching up to correct position quickly enough
          set IncreasePower flag
             else
          set PowerOK flag
             endif
```

```
         else  (we are moving backward)

        if we are too close (TgtBelow == true)
          set IncreasePower flag
        else if we aren't getting closer than we were before
          set ReducePower flag
        else if we aren't catching up to correct position quickly enough
          set IncreasePower flag
        else
          set PowerOK flag
        endif
      endif

      if IncreasingPower
        If we are within the low power adjustment range (< 5cm)
          set power increment to (1*DistOff) (e.g. 1%/cm)
        else
          set power increment to (2*DistOff) (e.g. 2%/cm)
        endif

        if power increment is greater than max allowed (20%)
          set power increment to 20%
        endif

               add power increment to NewMotorPower

      else if Reducing Power


        If we are within the low power adjustment range (< 5cm)
          set power decrement to (1*DistOff) (e.g. 1%/cm)
        else
          set power decrement to (2*DistOff) (e.g. 2%/cm)
        endif

        if power decrement is greater than max allowed (20%)
          set power decrement to 20%
        endif

        if power decrement is > existing power setting
          if current power is < 5%
            set NewMotorPower to zero
          else
            set NewMotorPower to PrevMotorPower/2
          endif
        else
          subtract power increment from NewMotorPower
        endif

      else (power must be just right)

        ; nothign to do, current setting is OK
      endif

    endif /* reverse */

  endif /* Moving */

  if NewMotorPower > 100 (out of range high)
    set NewMotorPower to 100
  endif

  save direction flag
  save motor power
  save distance

end
```

Source Code

Main.asm

```
;*****************************************************************************
;                                                                            *
;    Filename:         Main.asm                                              *
;    Date:             2/23/2012                                             *
;    Authors:          Adam Morse, Steve Hopkins, Conor P. Cahill            *
;    Company:          UND, EE452, Spring 2012                               *
;                                                                            *
;-----------------------------------------------------------------------------
; PROCESSOR DECLARATION
;-----------------------------------------------------------------------------

        LIST      P=PIC18F46K22          ; list directive to define processor
        #INCLUDE <P18F46K22.INC>         ; processor specific variable definitions
        #include "AutonomousTrailer.inc" ; Definitions for our project
        #include "Utilities.inc"

        ;
        ; get the processor configuration setup
        ;
        #include "ConfigProcessor.inc"

        ;
        ; external function references
        ;
        extern BackupBeeperInit, BackupBeeperEnable, BackupBeeperDisable, BackupBeeperUpdate
        extern HazardsInit, HazardsEnable, HazardsDisable, HazardsUpdate
        extern MotorInit, MotorUpdate, MotorEmergencyStop
        extern SensorInit, SensorRead
        extern ProcessData, ProcessDataLost, ProcessDataInit
        extern Delay_MS

;
; Local definitions
;
MSTAT_BIT_LOST               equ     D'0'
MSTAT_BIT_BKWRD                      equ     D'1'
MSTAT_BIT_NEED_BKWRD   equ     D'2'


;
; Reset Vector
;

RES_VECT  ORG     0x0000            ; processor reset vector
          GOTO    START             ; go to beginning of program

;
; High & Low priority interrupt vectors (just return for now)
;

ISRH      ORG     0x0008
          RETFIE

ISRL      ORG     0x0018
          RETFIE


;-----------------------------------------------------------------------------
; MAIN PROGRAM
;-----------------------------------------------------------------------------

START

               ;
               ; set clock to 16 mhz
               ;
               movf    OSCCON,W,A
```

```
                andlw   b'10001111'                     ; clear IRCF bits
                iorlw   b'01110000'                     ; set IRCF to 111 = 16 MHz
                movwf   OSCCON,A

                ;
                ; clear/initialize our registers
                ;
                clrf    MainStatus

                ;
                ; Initialize all of the components
                ;
                Call SensorInit
                Call MotorInit
                Call HazardsInit
                Call BackupBeeperInit
                Call ProcessDataInit

                ;
                ; Slight delay to let everything quiesce before starting processing
                ;
                movlw D'200'
                movwf Arg1,A
                Call Delay_MS

MainLoop:

                ;
                ; Read the distance sensor (no parameters to pass in)
                ;
                Call SensorRead

                ;
                ; Save the results
                ;
                movff   Result1,MainSensorReading

                ;
                ; if there isn't an error, skip the error handling code
                ;
                movlw   SNSR_LOSTVEHICLE
                cpfseq  MainSensorReading,A
                bra     MainSensorGood

                ;
                ; If we are already lost
                ;
                btfsc   MainStatus,MSTAT_BIT_LOST,A
                bra     MainAlertDone

                ;
                ; Set the lost status bit
                ;
                bsf     MainStatus,MSTAT_BIT_LOST,A

                ;
                ; We've lost our lead vehicle, so we need to enable the hazards lights and stop the
                ; trailer
                ;
                Call    HazardsEnable

                Call    MotorEmergencyStop

                ;
                ; tell ProcessData we lost the vehicle
                ;
                Call ProcessDataLost

MainAlertDone:

                ;
```

```
                ; Update the hazard lights (blinks them on/off periodically) (no args/return)
                ;
                Call HazardsUpdate

                ;
                ; skip to the end of the main loop
                ;
                bra MainLoopDone

MainSensorGood:

                ;
                ; if we aren't in "lost" mode
                ;
                btfss   MainStatus,MSTAT_BIT_LOST,A
                bra     MainProcess

                ;
                ; Turn off hazards (motor will clear after process data)
                ;
                Call HazardsDisable

                ;
                ; clear "lost" status
                ;
                bcf     MainStatus,MSTAT_BIT_LOST,A

MainProcess:
                ;
                ; Go process the sensor data and calculate the motor
                ;
                movff   MainSensorReading, Arg1
                Call ProcessData

                ;
                ; remember if ProcessData set the reverse direction
                ;
                bcf             MainStatus,MSTAT_BIT_NEED_BKWRD,A
                tstfsz  Result2,A
                bsf             MainStatus,MSTAT_BIT_NEED_BKWRD,A

                ;
                ; Update the motor settings with results from ProcessData
                ;
                movff   Result1, Arg1
                movff   Result2, Arg2
                Call MotorUpdate

                ;
                ; if the backward bit isn't set skip backwards code
                ;
                btfsc   MainStatus,MSTAT_BIT_NEED_BKWRD,A
                bra     MainMovingForward

                ;
                ; If backward status is alread set, skip setup code
                ;
                btfss   MainStatus,MSTAT_BIT_BKWRD,A
                bra     MainBackward_Process

                ;
                ; set backward status
                ;
                bsf     MainStatus,MSTAT_BIT_BKWRD,A

                ;
                ; enable the backward beeper
                ;
                call BackupBeeperEnable

                bra MainLoopDone
```

```
MainBackward_Process:

               ;
               ; Update the backward beeper
               ;
               Call BackupBeeperUpdate

MainMovingForward:
               ;
               ; If we were not moving backwards, skip past cleanup code
               ;
               btfsc   MainStatus,MSTAT_BIT_BKWRD,A
               bra     MainLoopDone

               ;
               ; Turn off backup beeper
               ;
               Call    BackupBeeperDisable

               ;
               ; clear backup status
               ;
               bcf     MainStatus,MSTAT_BIT_BKWRD,A

MainLoopDone:

               ;
               ; Delay 50ms between loop iterations
               ;
               movlw D'50'
               movwf Arg1,A
               Call Delay_MS

               goto MainLoop

FINISHED:
               GOTO $           ; stop here...


               END
```

ProcessData.asm

```
;
; ProcessData.asm - Functions used to process the sensor data and calculate motor
;  configuration settings
;
        LIST      P=PIC18F46K22           ; list directive to define processor
        #INCLUDE <P18F46K22.INC>          ; processor specific variable definitions
        #INCLUDE "AutonomousTrailer.inc" ; Definitions for our project

        ;
        ; Define entry points that will be accessible from outside this file
        ;
        global  ProcessData, ProcessDataInit, ProcessDataLost

;
; Local definitions
;
TARGET_DIST                   equu     D'50'   ; target follwing distance
TARGET_DIST_HIGH              equ     D'55'   ; high allowed range
TARGET_DIST_LOW               equ     D'45'   ; low allowed range

MAX_POWERINC                  equ     D'20'   ; maximum power change in one operation
DIST_LOW_POWER_INC            equ     D'5'    ; low pwer increases for DistOff < this

;
; Bit positions used to store data in PD_Status
;
PDSTAT_DIR                    equ             D'0'
PDSTAT_DIST_ZERO              equ             D'1'
PDSTAT_CLOSER_TGT             equ             D'2'
PDSTAT_TGT_BELOW              equ             D'3'
PDSTAT_TGT_ZERO               equ             D'4'
PDSTAT_SENSOR_HIGHER          equ             D'5'

        ;
        ; Start of the code section
        ;
        Code

;
; Function:    ProcessDataInit
;
; Purpose:             To Initialize the registers used by ProcessData
;
; Parameters:  none
;
; Returns:             Nothing of any value
;
; Notes:
;
ProcessDataInit:


        ;
        ; Initialize ProcessData variables
        ;
        clrf    PD_PrevMPower
        clrf    PD_Status

        return;

;
; Function:    ProcessDataLost
;
; Purpose:             To reset things when we've lost the LV
;
; Parameters:  none
;
; Returns:             Nothing of any value
;
```

```
; Notes:               Just call ProcessDataInit to do the work.
;
ProcessDataLost:

       call ProcessDataInit

       return;

;
; Function:    ProcessData
;
; Purpose:             To process the sensor distance data an determine a motor
;                              control power setting that will maintain the desired
;                              following distance
;
; Parameters:  arg1 - the distance to the lead vehicle in centimeters
;
; Returns:             Result1 - a pwer setting from 0 to 100% for the motor control
;                              Result2 - a direction bit (1=reverse, 0 = forward)
;
; Notes:
;
;       Our goal is to follow along 50cm behind the lead vehicle (LV) and we allow
;   a 10cm range (+/- 5CM) that we consider to be within range.   As the distance
;   increases we increase the power to the motor in order to catch up and get
;       withiin the appropriate distance.   When we start catching up, we start to
;       decrease the power so that we ease into the desired range rather than having
;       an abrupt power change once we're in range.
;
;       when the vehicle does go out of the desired range, we try to bring it back to
;       the exact midpoint of the range and then allow the range to take place
;
;       In order to prevent radical jerky changes in motor power, we will limit the
;       power change per operation to 20%
;
;
;
ProcessData:

               ;
               ; save the parameter (distance sensor reading)
               ;
               movff   Arg1, PD_SensorDist

               ;
               ; get the current settings for the direction and motor power
               ;
               movff   PD_PrevDir, PD_NewDir
               movff   PD_PrevMPower, PD_NewMPower

               ;
               ; if the motors aren't currently off (e.g. we are not stopped)
               ;       go to check moving
               ;
               tstfsz  PD_PrevMPower
               bra             PD_CheckMoving

               ;
               ; if the lead vehicle is beyond the allowinable range
               ;
               movlw   TARGET_DIST_HIGH
               cpfsgt  PD_SensorDist
               bra             PD_StopCheckSD_LT

               ;
               ; calculate the distance off target and set direction to forward
               ;
               movlw   TARGET_DIST
               subwf   PD_SensorDist,W,A
               movwf   PD_DistOff,A
```

```
                bcf             PD_Status,PDSTAT_DIR,A
                movlw   PD_DIRECT_FWD
                movwf   PD_NewDir,A

                bra             PD_StopAdjustPower

PD_StopCheckSD_LT:

                ;
                ; if the lead vehicle is below the allowinable range
                ;
                movlw   TARGET_DIST_LOW
                cpfslt  PD_SensorDist
                bra             PD_StopCheckSD_EQ

                ;
                ; calculate the distance off target and set direction to reverse
                ;       DistOff = TargetDistance - SensorDistance
                ;
                movlw   TARGET_DIST
                movwf   PD_DistOff,A
                movf    PD_SensorDist,W,A
                subwf   PD_DistOff,F,A

                bsf             PD_Status,PDSTAT_DIR,A
                movlw   PD_DIRECT_REV
                movwf   PD_NewDir,A

                bra             PD_StopAdjustPower

PD_StopCheckSD_EQ:

                ;
                ; Else we're within range and should stay stopped
                ;
                clrf    PD_NewMPower

                bra             PD_Done

PD_StopAdjustPower:

                ;
                ; Time to adjust our power based on the makeup distance
                ;
                ; if it's a small distance increment,
                ;       increment power 1% per cm
                ; othersie
                ;       incrment power 2% per cm
                ;   and check to make sure we don't exceed max single iteration increment
                ;
                movlw   DIST_LOW_POWER_INC
                cpfslt  PD_DistOff,A
                bra             PD_StopAdjustHighPower

                ;
                ; just use the distance for the lower power options (so 1cm=1%)
                ;
                movff   PD_DistOff,PD_NewMPower
                bra             PD_StopAdjustPowerDone

PD_StopAdjustHighPower:
                ;
                ; for the high pwer options, shift PD_DistOff 1 position to left (*2)
                ; so we set it to 1cm=2%.
                ;
                RLNCF   PD_DistOff,F,A

                ;
                ; if the result is greater than max increment, set it to max increment

                movlw   MAX_POWERINC
```

```
                cpfslt  PD_DistOff,A
                movwf   PD_DistOff,A

                ;
                ; remember the new motor setting
                ;
                movff   PD_DistOff, PD_NewMPower

PD_StopAdjustPowerDone:

                ;
                ; OK, we're done with dealing with the stopped trailer
                ;
                bra PD_Done

PD_CheckMoving:

                ;
                ; we know that we are moving and therefore we have to calculate if and by how much
                ; we should alter the power settings on the motors.   To determine this we need to
                ; calculate the following values:
                ;
                ;               PD_DistChange                 - how far we moved since the last change
                ;               PD_Status(higher)             - true if our new dist is > our prev dist
                ;               PD_Status(zero)                       - true if we've stayed exactly the
                                                                      ;  same distance
                ;               PD_DistOff                            - how far we are off the target
                                                                      ;  distance
                ;               PD_Status(TgtBelow)           - true if we're below the target distance
                ;               PD_Status(TgtExact)           - true if we're exactly on target
                ;               PD_Status(CloserTGT)   - true if we are closer to target distance
                ;

                ;
                ; calculate how far we've moved in the last iteration (absolute value)
                ;
                ; If SensorDistance < PrevDistance
                ;
                movf    PD_PrevDist,W,A
                cpfslt  PD_SensorDist
                bra             PD_MV_SD_GT

                ;
                ; diff = PrevDist - SensorDistance
                ;
                clrf    STATUS,A                      ; clear borrow bits
                subfwb  PD_SensorDist,W,A

                ;
                ; SensorDistance is not higher than PrevDistance, so clear flag
                ;
                bcf             PD_Status, PDSTAT_SENSOR_HIGHER

                bra             PD_MV_SD_Done

PD_MV_SD_GT:

                ;
                ; diff = SensorDistance - PrevDistance
                ;
                subwf   PD_SensorDist,W,A

                ;
                ; if distance same (wreg == 0)
                tstfsz  WREG,A
                bra     PD_MV_SD_SetHigher

                ;
                ; set zero bit and clear higher bit
                ;
                bcf             PD_Status,PDSTAT_SENSOR_HIGHER
```

```
                bsf             PD_Status,PDSTAT_DIST_ZERO,A

                bra             PD_MV_SD_Done

PD_MV_SD_SetHigher

                ;
                ; set higher flag and clear zero flag
                ;
                bsf             PD_Status,PDSTAT_SENSOR_HIGHER
                bcf             PD_Status,PDSTAT_DIST_ZERO,A

PD_MV_SD_Done:

                ;
                ; Remember our distance change
                ;
                movwf   PD_DistChange,A


                ;
                ; calculate how far we are off the target distance
                ; remember if we are below/above/same as target distance
                ;
                ; If SensorDistance < TargetDistance
                ;
                movlw   TARGET_DIST
                cpfslt  PD_SensorDist
                bra             PD_MV_TGT_GT

                ;
                ; diff = TargetDistance - SensorDistance
                ; PD_Status(closer) = TRUE
                ;
                clrf    STATUS,A                        ; clear borrow bits
                subfwb  PD_SensorDist,W,A
                bsf             PD_Status,PDSTAT_TGT_BELOW,A
                bra             PD_MV_TGT_Done

PD_MV_TGT_GT:

                ;
                ; we aren't < Tgt, so clear below flag
                ;
                bcf             PD_Status,PDSTAT_TGT_BELOW,A

                ;
                ; diff = SensorDistance - TargetDistance
                ;
                subwf   PD_SensorDist,W,A

                ;
                ; set zero flag
                ; if diff is not zero
                ;       clear zero flag
                ;
                bsf             PD_Status,PDSTAT_TGT_ZERO,A
                tstfsz  WREG,A
                bcf             PD_Status,PDSTAT_TGT_ZERO,A


PD_MV_TGT_Done:

                ;
                ; Remember our distance change
                ;
                movwf   PD_DistOff,A


                ;
                ; Now to determin if we are closer to the LV or not
```

```
                ;

                ;
                ; start by clearing closer flag
                ;
                bcf     PD_Status,PDSTAT_CLOSER_TGT

                ;
                ; if    both TgtBelow and SensorHiger are true
                ;    or both TgtBelow and SensorHiger are false - we are closer
                ;
                btfss   PD_Status,PDSTAT_TGT_BELOW,A
                bra             PD_MV_CL_CheckNotHigh

                btfss   PD_Status,PDSTAT_SENSOR_HIGHER,A
                bra             PD_MV_CL_Done

                bra             PD_MV_CL_SetCloser

PD_MV_CL_CheckNotHigh:

                btfsc   PD_Status,PDSTAT_SENSOR_HIGHER,A
                bra             PD_MV_CL_Done

                ;
                ; fall through to set closer flag
                ;

PD_MV_CL_SetCloser:
                ;
                ; we are closer so set closer flag
                ;
                bsf     PD_Status,PDSTAT_CLOSER_TGT

PD_MV_CL_Done:



                ;
                ; we now have all the inputs we need, so lets go figure out any motor power
                ; and/or direction changes that we need
                ;

                ;
                ; if we are exactly on target AND distance hasn't changed
                ;
                btfss   PD_Status,PDSTAT_TGT_ZERO,A
                bra             PD_MV_CheckForward
                btfss   PD_Status,PDSTAT_DIST_ZERO,A
                bra             PD_MV_CheckForward

                ;
                ; there's no change to be made....
                ;
                bra             PD_Done


PD_MV_CheckForward:

                ;
                ; First take a look at what we need to do if we are moving forward
                ;

                ;
                ; If we are moving backward, go to CheckReverse
                ;
                btfsc   PD_Status,PDSTAT_DIR,A
                bra             PD_MV_CheckReverse

                ;
```

```
                ; else if we're at SensorDistance < TargetDistance we're too close, go reduce power
                ;
                btfsc   PD_Status,PDSTAT_TGT_BELOW,A
                bra             PD_MV_ReducePower

                ;
                ; else if we are gaining on the LV, go immediately to reduce power
                ;
                btfss   PD_Status,PDSTAT_CLOSER_TGT,A
                bra             PD_MV_ReducePower


                ;
                ; if we are NOT catching up/closing the distance
                ;
                ;   e.g. if( (Diff < 16) && (SensorDist-(Diff*8)) > TargetDistance )
                ;
                ;    Catching Up is defined as a distance difference >= 16 or when if we
                ;    keep going at the current power for 8 more iterations we would be
                ;    <= the target distance
                ;
                ; If we're already gaining 16cm since last iteration (unlikely)
                ;      No need to increase power so skip it
                ;
                movlw   D'16'                       ; limit to < 16 since we're gonna multiply by 8
                cpfslt  PD_DistChange,A
                bra             PD_MV_OKPower

                ;
                ; if SensorDist-(DistChange*8) < TargetDist, we can skip changing as well
                ;
                movf    PD_DistChange,W,A
                bcf             STATUS,C,A
                rlcf    WREG,W,A                    ; WREG = WREG * 2 (three times == WREG * 8)
                bcf             STATUS,C,A
                rlcf    WREG,W,A
                bcf             STATUS,C,A
                rlcf    WREG,W,A
                subwf   PD_SensorDist,W,A
                movwf   PD_Tmp,A
                movlw   TARGET_DIST
                cpfsgt  PD_Tmp
                bra             PD_MV_OKPower

                ;
                ; go increase the power
                ;
                bra             PD_MV_IncreasePower


PD_MV_CheckReverse:

                ;
                ; Ok, if we got here we're going backwards so check to see if we need
                ; to increase or decrease power
                ;


                ;
                ; if we're at SensorDistance < TargetDistance we're too close, go increase power
                ;
                btfsc   PD_Status,PDSTAT_TGT_BELOW,A
                bra             PD_MV_IncreasePower

                ;
                ; else if we aren't getting closer to the TargetDistance, we need to go decrease power
                ;
                btfss   PD_Status,PDSTAT_CLOSER_TGT,A
                bra             PD_MV_ReducePower
```

```
        ;
        ; if we are NOT catching up/closing the distance
        ;
        ;   e.g. if( (Diff < 16) && (SensorDist-(Diff*8)) > TargetDistance )
        ;
        ;    Catching Up is defined as a distance difference >= 16 or when if we
        ;    keep going at the current power for 8 more iterations we would be
        ;    <= the target distance
        ;
        ; If we're already gaining 16cm since last iteration (unlikely)
        ;       No need to increase power so skip it
        ;
        movlw   D'16'                   ; limit to < 16 since we're gonna multiply by 8
        cpfslt  PD_DistChange,A
        bra             PD_MV_OKPower

        ;
        ; if SensorDist-(DistChange*8) < TargetDist, we can skip changing as well
        ;
        movf    PD_DistChange,W,A
        bcf             STATUS,C,A
        rlcf    WREG,W,A                ; WREG = WREG * 2 (three times == WREG * 8)
        bcf             STATUS,C,A
        rlcf    WREG,W,A
        bcf             STATUS,C,A
        rlcf    WREG,W,A
        subwf   PD_SensorDist,W,A
        movwf   PD_Tmp,A
        movlw   TARGET_DIST
        cpfsgt  PD_Tmp
        bra             PD_MV_OKPower

        ;
        ; go reduce the power
        ;
        bra             PD_MV_ReducePower



PD_MV_IncreasePower:

        ;
        ; so now we know we need to increase power...
        ;

        ;
        ; start with a 1%/cm power change
        ;
        movff   PD_DistOff,PD_PowerChange

        ;
        ; if we are beyond the low pwer range
        ;
        movlw   DIST_LOW_POWER_INC
        cpfsgt  PD_DistOff,A
        bra             PD_MV_IncCheckMax

        ;
        ; for the high pwer options, shift PD_PowerChange 1 position to left (*2)
        ; so we set it to 1cm=2%.
        ;
        RLNCF   PD_PowerChange,F,A

PD_MV_IncCheckMax:
        ;
        ; if the result is greater than max increment, set it to max increment
        ;
        movlw   MAX_POWERINC
        cpfslt  PD_PowerChange,A
        movwf   PD_PowerChange,A
```

```
                ;
                ; add the pwer change into the motor power
                ;
                movf    PD_PowerChange,W,A
                addwf   PD_NewMPower,F,A

                bra             PD_Done


PD_MV_ReducePower:

                ;
                ; so now we know we need to reduce power...
                ;


                ;
                ; start with a 1%/cm power change
                ;
                movff   PD_DistOff,PD_PowerChange


                ;
                ; if we are beyond the low pwer range
                ;
                movlw   DIST_LOW_POWER_INC
                cpfsgt  PD_DistOff,A
                bra             PD_MV_RedCheckMax


                ;
                ; for the high pwer options, shift PD_PowerChange 1 position to left (*2)
                ; so we set it to 1cm=2%.
                ;
                bcf             STATUS,C,A
                rlcf    PD_PowerChange,F,A

PD_MV_RedCheckMax:
                ;
                ; if the result is greater than max increment, set it to max increment
                ;
                movlw   MAX_POWERINC
                cpfslt  PD_PowerChange,A
                movwf   PD_PowerChange,A


                ;
                ; if the decrement is larger than the current power setting
                ;
                movf    PD_PowerChange,W,A
                cpfslt  PD_NewMPower,A
                bra             PD_MV_RedSubtract


                ;
                ; if the current power is <= 5, set it to zero
                ;
                movlw   D'5'
                cpfsgt  PD_NewMPower,A
                bra             PD_MV_RedSetZero


                ;
                ; Otherwise Set power to half of its existing value
                ;
                bcf             STATUS,C,A
                rrcf    PD_NewMPower,F,A
                bra             PD_MV_RedDone

PD_MV_RedSetZero:

                ;
                ; set power to zero
                ;
                clrf    PD_NewMPower;
                bra             PD_MV_RedDone
```

```
PD_MV_RedSubtract:

                ;
                ; add the pwer change into the motor power
                ;
                movf    PD_PowerChange,W,A
                subwf   PD_NewMPower,F,A

PD_MV_RedDone:

                bra             PD_Done

PD_MV_OKPower:
                ;
                ; Nothing to do - power is already OK
                ;
                bra             PD_Done


PD_Done:

                ;
                ; if NewMPower is greater than 100, limit it to 100
                ;
                movlw   D'100'
                cpfslt  PD_NewMPower,A
                movwf   PD_NewMPower,A


                ;
                ; save the old distance value
                ;
                movff   PD_SensorDist,PD_PrevDist

                ;
                ; save the new settings for power & direction and
                ; place them in the return registers
                ;
                movff   PD_NewMPower,PD_PrevMPower
                movff   PD_NewMPower,Result1

                movff   PD_NewDir,PD_PrevDir
                movff   PD_NewDir,Result2


                return

                end
```

MotorControl.asm

```
;
; MotorControl.asm - definition of functions used to setup/control the motors
;
               LIST      P=PIC18F46K22          ; list directive to define processor
               #INCLUDE <P18F46K22.INC>         ; processor specific variable definitions
               #INCLUDE "AutonomousTrailer.inc" ; Definitions for our project

;
; Local definitions
;

MSTAT_REVERSE  equ 0x01

MOTOR_SET_FWD  equ b'11010000'
MOTOR_SET_REV  equ b'10101000'
MOTOR_SET_OFF  equ b'01010000'

               ;
               ; Define entry points that will be accessible from outside this file
               ;
               global  MotorInit, MotorUpdate, MotorEmergencyStop

               ;
               ; Start of the code section
               ;
               Code

;
; Function:    MotorInit
;
; Purpose:            Initialize the Motor Controls
;
; Parameters:  None
;
; Returns:            Nothing of any value
;
; Notes:
;
MotorInit:

               ;
               ; Configure the Timer 2 period register for PWM mode.  We have a 16Hz clock and
               ; desire a 15Khz PWM frequency so: 16M/4/4/15K - 1 = 66
               ;
               movlw d'66'
               movwf PR2,A

               ;
               ; configure CCP1 and CCP2 pin on PORT C for output
               ;
               bcf TRISC,CCP1,A
               bcf TRISC,CCP2_PORTC,A

               ;
               ; Clear Timer2 count and configure Timer2 as follows:
               ;
               ;   b7='0'      - unimplemented
               ;   b6-3='0000' - no ouput post scaling
               ;   b2='1'      - enable timer
               ;   b1-0='01'   - prescaler is 4 (so Timer = (FOSC/4)/4 = 1Mhz)
               ;
               clrf TMR2,A
               movlw b'00000101'
               movwf T2CON,A

               ;
               ; Associate Timer2 with PWM1 and PWM2
               ;
               movlb   0x0f
```

```
                movf    CCPTMRS0,W,A
                andlw   b'11100100'     ; both C2TSEL and C1TSEL set to 0
                movwf   CCPTMRS0,BANKED
                movlb   0x00

                ;
                ; Enable PWM Mode on CCP1 and CCP2
                ;
                ;    b7-6='00'  - unimplmented
                ;    b5-4='00'  - LOW order bits of PWM Duty Cycle
                ;    b3-0='1100' - Enable PWM Mode
                movlw 0x0C
                movwf CCP1CON,A
                movwf CCP2CON,A

                ;
                ; Configure pins 7-3 on PortC for use as output (motor control)
                ;
                ; We are using:
                ;
                ;              RC7 - EN (enable bit on motor control board)
                ;              RC6 - IN1_M1  - Motor 1 input 1
                ;              RC5 - IN2_M1  - Motor 1 input 2
                ;              RC4 - IN1_M2  - Motor 2 input 1
                ;              RC3 - IN2_M2  - Motor 2 input 2
                ;
                movf TRISC,W,A
                andlw b'00000111'; clear TRISC bits<7:3> to set pins as output
            movwf  TRISC,A

                ;
                ; Start with the motor off
                ;
            movlw      MOTOR_SET_OFF
                movwf   PORTC,A

                return

;
; Function:     MotorUpdate
;
; Purpose:              Update the motor control settings based on a desired increase
;                               or decrease in power as indicated in Arg1
;
; Parameters:  Arg1 - the desired motor power percentage (0-100)
;                               Arg2 - the desired direction (1=reverse, 0=forward)
;
; Returns:              nothing of any value
;
; Notes:
;
MotorUpdate:

                ;
                ; Notes:
                ;       From the spec, CPRxH doesn't need to be set -- it's read-only in PWM mode
                ;       The low order 2 bits of duty cycle settings are in CCPxCON b5-4.
                ;

                ;
                ; default settings are for the motor to be turned off -- this will get changed
                ; as we parse the input (most likely)
                ;
            movlw MOTOR_SET_OFF
            movwf MotorSettings, A

                ;
                ; get the passed in power percentage
                ;
                movf    Arg1,W,A
```

```
                movwf   CCPR1L,A
                movwf   CCPR1H,A
                movwf   CCPR2L,A
                movwf   CCPR2H,A

                ;
                ; if direction is forward
                ;
                tstfsz  Arg2,A
                bra             MotorReverse

                ;
                ; Set motor settings to indicate forward
                ;
            movlw       MOTOR_SET_FWD

                ;
                ; clear the reverse bit in the status
                ;
                bcf             MotorStatus, MSTAT_REVERSE

                bra             MotorSetup

MotorReverse:

                ;
                ; Set motor settings to indicate reverse
                ;
            movlw MOTOR_SET_REV

                ;
                ; set the status to indicate reverse
                ;
                bsf             MotorStatus, MSTAT_REVERSE

MotorSetup:

                ;
                ; remember motor settings
                ;
            movwf MotorSettings, A

                ;
                ; Pass the motor settings out the port
                ;
                movff MotorSettings, PORTC

                return


;
; Function:     MotorEmergencyStop
;
; Purpose:      Initialize the Motor Controls
;
; Parameters:   None
;
; Returns:              Nothing of any value
;
; Notes:                Just set the duty cycle to zero
;
MotorEmergencyStop:

                ;
                ; set duty cycle to zero
                ;
                clrf    CCPR1L,A
                clrf    CCPR1H,A
                clrf    CCPR2L,A
                clrf    CCPR2H,A
```

```
;
; turn the motors off
;
movlw MOTOR_SET_OFF
movwf MotorSettings, A
movff MotorSettings, PORTC

return


end
```

Sensor.asm

```
;
; Sensors.asm - definitions functions used to configure and/or read the sensors
;
                LIST      P=PIC18F46K22          ; list directive to define processor
                #INCLUDE <P18F46K22.INC>         ; processor specific variable definitions
                #INCLUDE "AutonomousTrailer.inc" ; Definitions for our project

                ;
                ; Define entry points that will be accessible from outside this file
                ;
                global  SensorInit, SensorRead

                ;
                ; Start of the code section
                ;
                Code

;
; Function:     SensorInit
;
; Purpose:              To initialize the sensor configuration
;
; Parameters:   None
;
; Returns:              Nothing of any value
;
; Notes:                None
;
SensorInit:

                ;
                ; Configure portA for input and set first bit for analog input
                ;
                clrf    PORTA,A
                clrf    ANSELA,A
                bsf             ANSELA,ANSA2
                bsf     TRISA,RA2

                ;
                ; Configure ADCON2:
                ;
                ;   b7='1'     - Right justified output
                ;   b6='0'     - unimplemented
                ;   b5-3='100' - 8 TAD
                ;   b2-0='020' - Conversion clock= FOSC/32
                ;
                movlw   B'10100010'
                movwf   ADCON2,A

                ;
                ; Configure ADCON1:
                ;
                ;   b7='0'      - use CCP5 trigger
                ;   b6-4='000'  - unimplemented
                ;   b3-2='00'   - use internal AVdd for A/D Vref+
                ;   b1-0='00'   - use internal Avss for A/D Vref-
                ;
                movlw   B'00000000'
                movwf   ADCON1,A ;Configure ADCON1, Internal ref

                ;
                ; Configure ADCON0:
                ;
                ;   b7='0'       - unimplemented
                ;   b6-2='00010' - use AN2 for analog input
                ;   b1='0'       - No conversion in progress yet
                ;   b0='1'       - ADC is enabled
                ;
                movlw   B'00001001'
```

```
                movwf   ADCON0,A ;Configure ADCON0, AN2, ADC ON

                return

;
; Function:     SensorRead
;
; Purpose:              To read the distance sensor
;
; Parameters:   None
;
; Returns:              The low order 8 bits of the distance to the lead
;                               vehicle or 0xFF if there was an error or the vehicle
;                               is too far ahead
;
; Notes:                We do not delay within the sensor function.  Instead
;                               we depend upon a delay in the main loop between each
;                               iteration
SensorRead:

                ;
                ; enable ADC conversion
                ;
                BSF             ADCON0,GO ; start conversion

                ;
                ; Loop until the conversion is done
                ;
adcpoll:
                BTFSC   ADCON0,DONE
                BRA     adcpoll

                ;
                ; Move the results into WREG and Result1
                ;
                movf    ADRESL, W, A
                movwf   Result1,A

                ;
                ; if the high order register is not zero, we have an error
                ; Either the trailer is too far ahead or there was actuall an
                ; error in the sensor.  In both cases we flag an error
                ;
                tstfsz  ADRESH,A
                bra             SensorRead_SetError

                ;
                ; check to see if we are less than 20CM from the vehicle (below
                ; the range for our sensor)
                ;
                movlw   D'20'
                cpfslt  ADRESL,A
                bra             SensorRead_Done

                ;
                ; fall through to set error code when < 20CM

SensorRead_SetError:

                ;
                ; set errror code in Result1
                ;
                movlw   SNSR_LOSTVEHICLE
                movwf   Result1,A
                bra     SensorRead_Done

SensorRead_Done:

                return
        end
```

Beeper.asm

```
;
; BackupBeeper.asm - Functions used to manage the backup Beeper
;
        LIST       P=PIC18F46K22           ; list directive to define processor
        #INCLUDE <P18F46K22.INC>           ; processor specific variable definitions
        #INCLUDE "AutonomousTrailer.inc" ; Definitions for our project

        ;
        ; Define entry points that will be accessible from outside this file
        ;
        global  BackupBeeperInit, BackupBeeperEnable, BackupBeeperDisable, BackupBeeperUpdate

;
; local definitions
;
BESTAT_ENABLED          equu    D'0'
BESTAT_ON               equ     D'1'

BE_CYCLE_ON_COUNT       equ     D'10'                 ; approx .5 seconds
BE_CYCLE_OFF_COUNT      equ     D'20'                 ; approx 1 second

BE_PWM_FREQ             equ     D'200'                ; 1300hz from (16M/4)/16/1300
BE_PWM_ON_DUTYC         equ     D'100'                ; moderate volume (25% of BW_PWM_FREQ)
BE_PWM_OFF_DUTYC        equ     D'0'

BE_PORT                 equ PORTA                     ; Beeper is connected to PORTA:5
BE_PIN                  equ RA5
BE_PORT_CTRL            equ TRISA

        ;
        ; Start of the code section
        ;
        Code

;
; Function:    BackupBeeperInit
;
; Purpose:            To initialize the backup Beeper
;
; Parameters:
;
; Returns:
;
; Notes:
;
BackupBeeperInit:

               ;
               ; The SFRs for the backup beeper are outside of the access bank, so we need to
               ; load the BSR (0xf) and use it for operations
               ;
               movlb   0x0F

               ;
               ; Configure the Timer 4 period register for PWM mode.  We have a 16Hz clock and
               ; desire a 1300hz PWM frequency so: 16M/4/16/1300 - 1 = 200
               ;
               movlw d'200'
               movwf PR4,BANKED

               ;
               ; configure RB5 pin on PORT B for output
               ;
               bcf TRISB,RB5,A

               ;
               ; Clear Timer4 count and configure Timer4 as follows:
               ;
               ;   b7='0'      - unimplemented
```

```
                ;   b6-3='0000' - no ouput post scaling
                ;   b2='1'      - enable timer
                ;   b1-0='01'   - prescaler is 4 (so Timer = (FOSC/4)/4 = 1Mhz)
                ;
                clrf TMR4,A
                movlw b'00000101'
                movwf T4CON,BANKED

                ;
                ; Associate Timer4 with PWM3
                ;
                movf    CCPTMRS0,W,A
                andlw   b'00111111'
                iorlw   b'01000000'            ; C3TSEL set to '01' = Timer4
                movwf   CCPTMRS0,BANKED

                ;
                ; Enable PWM Mode on CCPR3
                ;
                ;    b7-6='00'   - unimplmented
                ;    b5-4='00'   - LOW order bits of PWM Duty Cycle
                ;    b3-0='1100' - Enable PWM Mode
                ;
                movlw 0x0C
                movwf CCP3CON,BANKED

                ;
                ; clear the status word (indicateds disabled)
                ;
                clrf    BE_Status

                ;
                ; reset BSR
                ;
                movlb   0x00

                return

;
; Function:     BackupBeeperEnable
;
; Purpose:      To Enable the backup Beeper
;
; Parameters:   None
;
; Returns:      None
;
; Notes:
;
BackupBeeperEnable:

                ;
                ; set status to enabled
                ;
                bsf             BE_Status,BESTAT_ENABLED,A

                ;
                ; set the on indicator
                ;
                bsf             BE_Status,BESTAT_ON,A

                ;
                ; set the on cycle count-down timer
                ;
                movlw   BE_CYCLE_ON_COUNT
                movwf   BE_Counter,A

                ;
                ; The SFRs for the backup beeper are outside of the access bank, so we need to
                ; load the BSR (0xf) and use it for operations
                ;
```

```
                movlb 0xf

                ;
                ; turn on the output
                ;
                movlw   BE_PWM_ON_DUTYC
                movwf   CCPR3L,BANKED

                ;
                ; reset BSR
                ;
                movlb   0x00

                return

;
; Function:     BackupBeeperDisable
;
; Purpose:              To Enable the backup Beeper
;
; Parameters:   None
;
; Returns:              None
;
; Notes:
;
BackupBeeperDisable:

                ;
                ; The SFRs for the backup beeper are outside of the access bank, so we need to
                ; load the BSR (0xf) and use it for operations
                ;
                movlb 0xf

                ;
                ; turn off the PWM
                ;
                movlw   BE_PWM_OFF_DUTYC
                movwf   CCPR3L,BANKED

                ;
                ; turn off the beeper
                ;
                bcf             BE_PORT,BE_PIN

                ;
                ; set status to disabled
                ;
                bcf             BE_Status,BESTAT_ENABLED,A

                ;
                ; reset BSR
                ;
                movlb   0x00

                return

;
; Function:     BackupBeeperUpdate
;
; Purpose:              To update the BackupBeeper status (turn on/off) each iteration
;
; Parameters:   None
;
; Returns:      None
;
; Notes:
;
BackupBeeperUpdate:

                ;
```

```
                ; decrement the countdown counter and return if not zero
                ;
                decfsz          BE_Counter,F,A
                bra              BEU_Done

                ;
                ; if we get here, the counter has run out so time to change the
                ; state of the beeper
                ;

                ;
                ; The SFRs for the backup beeper are outside of the access bank, so we need to
                ; load the BSR (0xf) and use it for operations
                ;
                movlb 0xf

                ;
                ; if they are on
                ;
                btfss           BE_Status,BESTAT_ON,A
                bra             BEU_TurnEmOn

                ;
                ; turn off the PWM
                ;
                movlw   BE_PWM_OFF_DUTYC
                movwf   CCPR3L,BANKED

                ;
                ; set the off countdown counter
                ;
                movlw   BE_CYCLE_OFF_COUNT
                movwf   BE_Counter,A

                ;
                ; flag that they are off
                ;
                bcf             BE_Status,BESTAT_ON,A

                bra             BEU_Done

BEU_TurnEmOn:

                ;
                ; turn on the PWM
                ;
                movlw   BE_PWM_ON_DUTYC
                movwf   CCPR3L,BANKED

                ;
                ; set the on countdown counter
                ;
                movlw   BE_CYCLE_ON_COUNT
                movwf   BE_Counter,A

                ;
                ; flag that they are on
                ;
                bsf     BE_Status,BESTAT_ON,A

BEU_Done:

                ;
                ; reset BSR
                ;
                movlb   0x00

                return


                end
```

Hazards.asm

```
;
; Hazards.asm - Functions used to manage the Hazard LEDs
;
        LIST      P=PIC18F46K22          ; list directive to define processor
        #INCLUDE <P18F46K22.INC>         ; processor specific variable definitions
        #INCLUDE "AutonomousTrailer.inc" ; Definitions for our project

        ;
        ; Define entry points that will be accessible from outside this file
        ;
        global  HazardsInit, HazardsEnable, HazardsDisable, HazardsUpdate

;
; Local definitions
;
HZSTAT_ENABLED          equ D'0'
HZSTAT_ON               equ D'1'

HZ_CYCLE_ON_COUNT       equ D'15'                       ; approx .75 seconds
HZ_CYCLE_OFF_COUNT      equ D'20'                       ; approx 1 second

HZ_PORT                 equ PORTB                       ; Hazards are connected to PORTB:2
HZ_PIN                  equ RB2
HZ_PORT_CTRL            equ TRISB

        ;
        ; Start of the code section
        ;
        Code

;
; Function:    HazardsInit
;
; Purpose:     To initialize the backup beeper
;
; Parameters:
;
; Returns:
;
; Notes:               The hazards are controlled by a relay connected to
;                              digital port AN5
;
HazardsInit:

        ;
        ;       enable the port for output
        ;
        bcf     HZ_PORT_CTRL,HZ_PIN,A

        ;
        ; clear the status word (indicateds disabled)
        ;
        clrf    HZ_Status

        return

;
; Function:    HazardsEnable
;
; Purpose:     To Enable the backup beeper
;
; Parameters:  None
;
; Returns:     None
;
; Notes:
;
```

```
HazardsEnable:

        ;
        ; set status to enabled
        ;
        bsf             HZ_Status,HZSTAT_ENABLED,A

        ;
        ; set the lights on indicator
        ;
        bsf             HZ_Status,HZSTAT_ON,A
        ;
        ; set the on cycle count-down timer
        ;
        movlw   HZ_CYCLE_ON_COUNT
        movwf   HZ_Counter,A

        ;
        ; turn on the output pin
        ;
        bsf             HZ_PORT,HZ_PIN

        return

;
; Function:    HazardsDisable
;
; Purpose:     To Enable the backup beeper
;
; Parameters:  None
;
; Returns:     None
;
; Notes:
;
HazardsDisable:

        ;
        ; turn off the hazards
        ;
        bcf             HZ_PORT,HZ_PIN

        ;
        ; set status to disabled
        ;
        bcf             HZ_Status,HZSTAT_ENABLED,A

        return

;
; Function:    HazardsUpdate
;
; Purpose:     To update the Beeper status (turn on/off) each iteration
;
; Parameters:  None
;
; Returns:     None
;
; Notes:
;
HazardsUpdate:

        ;
        ; decrement the countdown counter and return if not zero
        ;
        decfsz          HZ_Counter,F,A
        bra              HZU_Done

        ;
        ; if we get here, the counter has run out so time to change the
        ; state of the hazards
```

```
        ;

        ;
        ; if they are on
        ;
        btfss           HZ_Status,HZSTAT_ON,A
        bra             HZU_TurnEmOn

        ;
        ; turn them off
        ;
        bcf             HZ_PORT,HZ_PIN

        ;
        ; set the off countdown counter
        ;
        movlw   HZ_CYCLE_OFF_COUNT
        movwf   HZ_Counter,A

        ;
        ; flag that they are off
        ;
        bcf             HZ_Status,HZSTAT_ON,A

        bra             HZU_Done

HZU_TurnEmOn:

        ;
        ; turn them on
        ;
        bsf             HZ_PORT,HZ_PIN

        ;
        ; set the on countdown counter
        ;
        movlw   HZ_CYCLE_ON_COUNT
        movwf   HZ_Counter,A

        ;
        ; flag that they are on
        ;
        bsf             HZ_Status,HZSTAT_ON,A



HZU_Done:

        return


        end
```

Utilities.asm

```
                LIST      P=PIC18F46K22          ; list directive to define processor
                #INCLUDE <P18F46K22.INC>         ; processor specific variable definitions
                #INCLUDE "AutonomousTrailer.inc" ; Definitions for our project

                ;
                ; Define entry points that will be accessible from outside this file
                ;
                global  Delay_S, Delay_MS

                ;
                ; Start of the code section
                ;
                Code

;
; Function:    Delay_S
;
; Purpose:             to delay exeuction for a specified number of seconds
;
; Parameters:  Arg1 - # of seconds to delay
;
; Returns:             nothing of any value
;
; Notes:               We use the thousandths of a second delay function to do the
;                              real work and just loop through successive calls of delays
;                              of 100 thousandths of a second
;

Delay_S:
                movff   Arg1, UtilDS_Outer

Delay_S_Outer:

                movlw   D'10'
                movwf   UtilDS_Inner,A

Delay_S_Inner:
                movlw   D'100'
                movwf   Arg1,A
                call    Delay_MS

                decfsz  UtilDS_Inner,F
                bra Delay_S_Inner

                decfsz  UtilDS_Outer,F
                bra Delay_S_Outer

                return

;
; Function:    Delay_MS
;
; Purpose:             to delay exeuction for a specified number of thousandths of
;                              a second (1-255) (so 10 = 1/100 second)
;
; Parameters:  Arg1 - # of hundredths of a second to delay
;
; Returns:             nothing of any value
;
; Notes:               We have a 16Mhz clock, so for each hundredth of a second
;                              we need to execute approximately 4K instructions.  Given
;                              that our loop counters are 8 bit numbers, we need 3 levels
;                              of loops.  The inner two levels create a 1/100 second delay
;                              while the outer loop works with the number of hundredths of
;                              a second parameter passed in.
;
;                              To execute 4K instructions, we have an inner loop that
;                              contains 17 instruction cycles (7 gotos, one DECFSZ
;                              and one branch).  This loop is executed 25 times which gives
```

```
;                               us 25*17-1 or 424 instructions.  The loop overhead in the
;                               middle loop consists of 20 instruction cycles and this loop is
;                               executed 9 times (424+21)*9-1 = 3995 instruction
;                               cycles.
;
;                               The outer loop will have 5 extra instruction cycles per
;                               loop execution, bringing the total to 4000 cycles per outer
;                               loop.
;
;                               The function function will have 5 instruction cycles of
;                               overhead (4 for the call/return and one for saving
;                               of WREG) so the delay will be slightly longer
;

Delay_MS:
               ;
               ; save Arg1 in our outer loop counter
               ;
               movfF   Arg1,UtilDMS_Outer

OuterLoop:

               movlw   D'9'
               movwf   UtilDMS_Middle,A

               ; with 103 iterations of the inner loop, we execute 2069 instructions
               ; per iteration of the middle loop

MiddleLoop:
               movlw   D'25'
               movwf   UtilDMS_Inner,A

InnerLoop:

               goto $+2
               goto $+2
               goto $+2
               goto $+2
               goto $+2
               goto $+2
               goto $+2
               decfsz  UtilDMS_Inner,F
               bra InnerLoop

               goto $+2
               goto $+2
               goto $+2
               goto $+2
               goto $+2
               goto $+2
               goto $+2
               nop
               decfsz UtilDMS_Middle, F
               bra MiddleLoop

               decfsz UtilDMS_Outer, F
               bra OuterLoop

               return

               end
```

AutonomousTrailer.inc

```
;
; Global defined values and register definitions for the Autonomous Trailer
; project.
;

;
; Sensor definitions
;
SNSR_LOSTVEHICLE       equ 0xFF

;
; ProcessData definitions
;
PD_DIRECT_FWD          equ D'0'
PD_DIRECT_REV          equ D'1'

;
; Motor control Definitions
;
MC_RES1_BIT_DIR               equ     0       ; set to 1 if we are moving backwards

;
; Function Call Interface Registers
;
Arg1           set 0x01
Arg2           set 0x02
Arg3           set 0x03
Result1        set 0x05
Result2        set 0x06
Result3        set 0x07

;
; Main Loop registers
;
MainStatus             set 0x10
MainSensorReading      set 0x11

;
; Sensor/Hazard/Beeper registers
;
HZ_Status              set 0x20
HZ_Counter             set 0x21
BE_Status              set 0x22
BE_Counter             set 0x23

;
; Motor Control Registers
;
MotorStatus    set 0x30
MotorSettings  set 0x31

;
; Utility Registers
;
UtilDS_Outer   set 0x40
UtilDS_Inner   set 0x41
UtilDMS_Outer  set 0x42
UtilDMS_Middle set 0x43
UtilDMS_Inner  set 0x44

;
; ProcessData Registers
;
PD_Status      set 0x50
PD_PrevMPower  set 0x51              ; the current motor setting
PD_NewMPower   set 0x52
PD_NewDir      set 0x53
PD_PrevDir     set 0x54
PD_DistChange  set 0x55
```

```
PD_DistOff     set 0x56
PD_Tmp         set 0x57
PD_SensorDist  set 0x58
PD_PowerChange set 0x59
PD_PrevDist    set 0x5A


;
; Timers
;
;       Timer2 - PWM control for moters
;

;
; I/O Port reservations
;
; Port A:
;
;       RA2 - used for ADC input port AN2
;
; Port B:
;
;       RB2 - Hazards
;       RB5 - CCP3/P3A - Beeper pwm
;
;
; Port C:
;
;       RC7 - EN
;       RC6 - IN1_M1
;       RC5     - IN2_M1
;       RC4 - IN1_M2
;       RC3     - IN2_M2
;
;       RC2 - CCP1/P1A - Motor 1 PWM
;       RC1     - CCP2/P2A - Motor 2 PWM
;
```

ConfigProcessor.inc

```
;
; Configuration bit settings (copied from template)
;
     ;Setup CONFIG11H
     CONFIG FOSC = INTIO7, PLLCFG = OFF, PRICLKEN = OFF, FCMEN = OFF, IESO = OFF
     ;Setup CONFIG2L
     CONFIG PWRTEN = OFF, BOREN = OFF, BORV = 190
     ;Setup CONFIG2H
     CONFIG WDTEN = OFF, WDTPS = 1
     ;Setup CONFIG3H
     CONFIG MCLRE = EXTMCLR, CCP2MX = PORTC1, CCP3MX = PORTB5, HFOFST = OFF, T3CMX = PORTB5, P2BMX =
PORTD2
     ;Setup CONFIG4L
     CONFIG STVREN = OFF, LVP = OFF, XINST = OFF
     ;Setup CONFIG5L
     CONFIG CP0 = OFF, CP1 = OFF, CP2=OFF, CP3=OFF
     ;Setup CONFIG5H
     CONFIG CPB = OFF, CPD = OFF
     ;Setup CONFIG6L
     CONFIG WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF
     ;Setup CONFIG6H
     CONFIG WRTB = OFF, WRTC = OFF, WRTD = OFF
     ;Setup CONFIG7L
     CONFIG EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF, EBTR3 = OFF
     ;Setup CONFIG7H
     CONFIG EBTRB = OFF
```

Utilities.inc

```
;
; External symbols and definitions used by Utlities.asm
;
; This should be included by code modules that want to invoke functions defined in Utilities.asm
;
       extern Delay_MS, Delay_S
```

Hardware Systems

Hardware Description

The hardware consists of a dual motor tracked chassis made by Dagu Electronics called the Rover 5. This kit came with 2 DC motors with 87:1 gear ratios to provide extra torque and optional wheel encoders. Other kits came with 4 motors, but the front 2 motors were disabled by removing one of the interconnecting gears in the transmission. The optional encoders were omitted because this project did not require absolute speed control. To maintain a fixed distance from the lead object, an ultrasonic distance sensor was used. The XL−MaxSonar−AE4, made by MaxBotics, emits a 44 kHz tone and internally measures the time between the transmission and the echo to determine the distance. Using this distance as the feedback mechanism to the MCU, we are able to control the speed of the motors to maintain a fixed distance of 50 cm to our target. The motor control consists of a motor driver chip by Freescale (MC33926) and the PWM function in the Microchip PIC18F46K22 MCU. The MCU is the heart of the system. It ties together the sensor readings, motor speed, back-up beeper, and hazard lights.

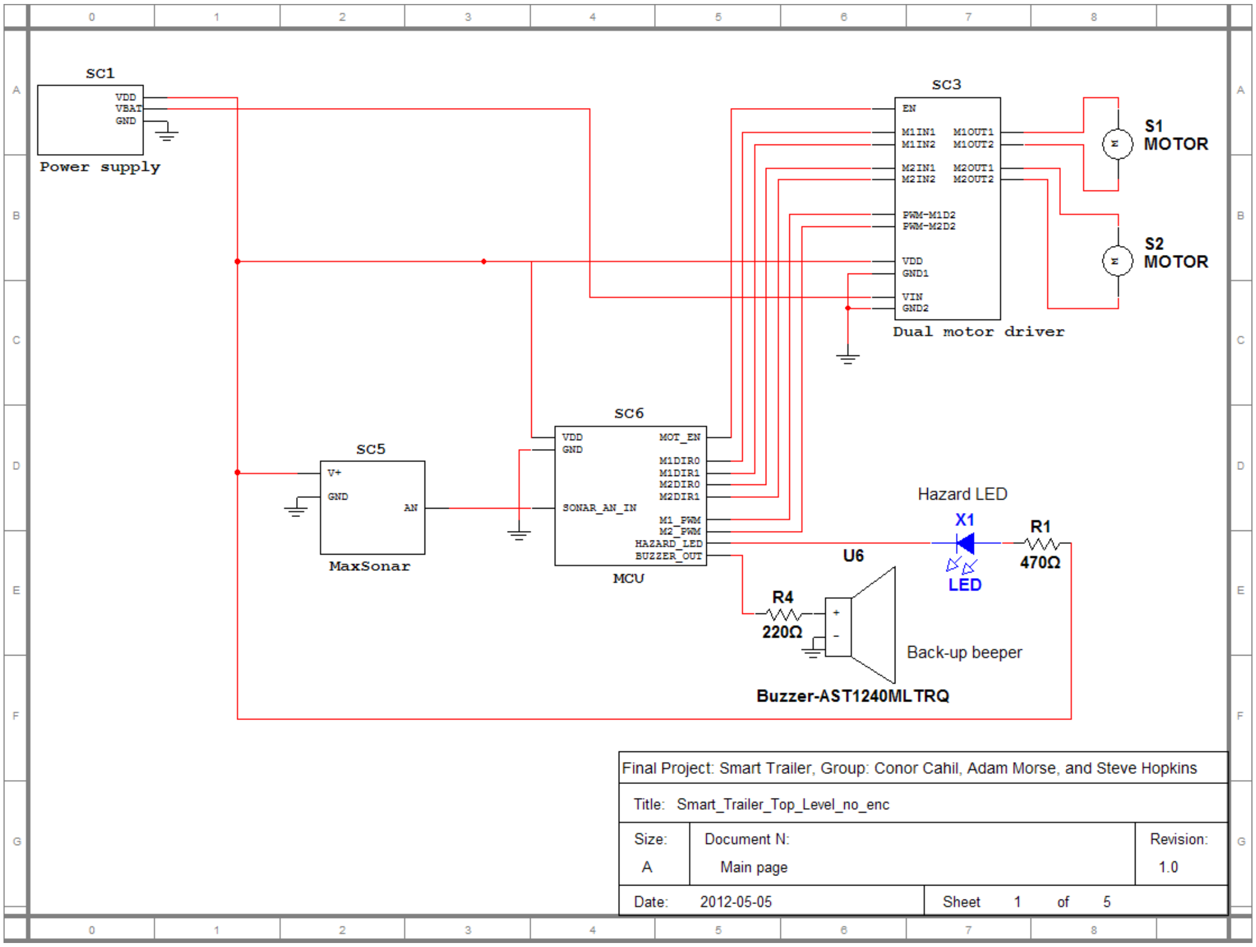
Power Circuits:

A single 9V battery is fed into an LM2940S-5.0 voltage regulator. It converts the 9 VDC to 5 VDC used to power the MCU and peripherals.

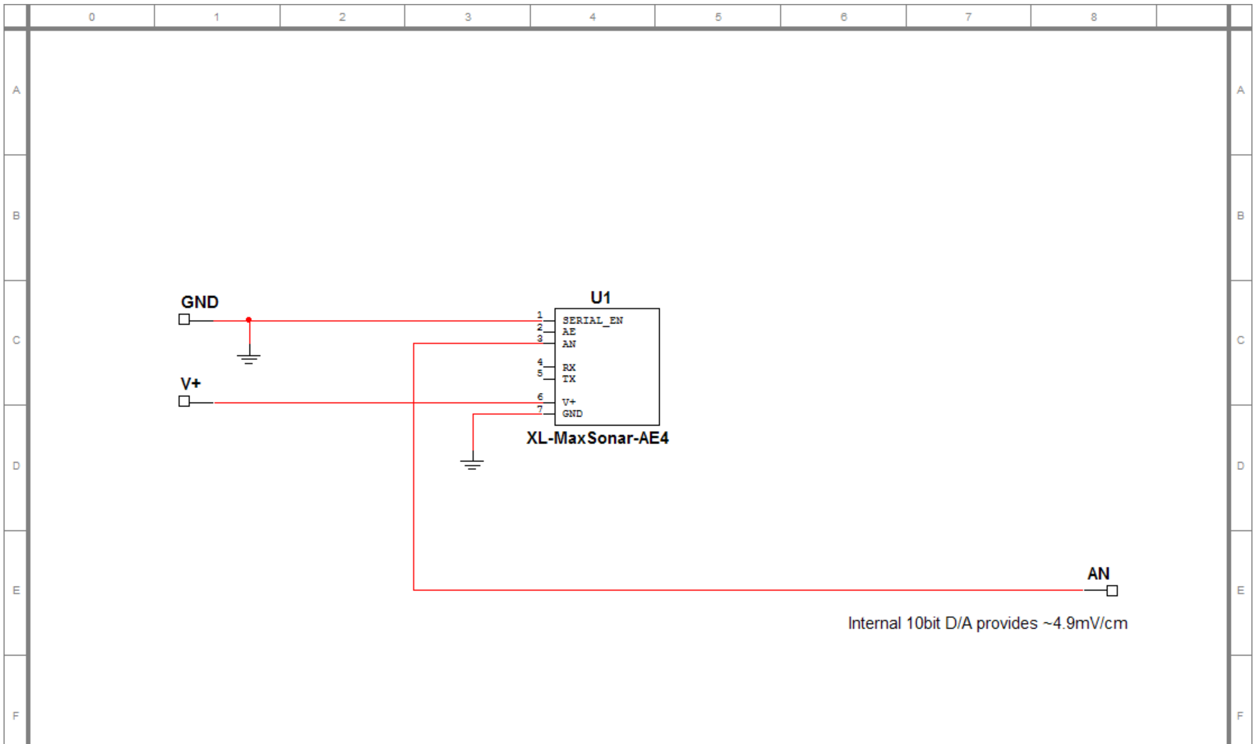
A separate battery pack with six 1.2 VDC NiMH batteries connected in series provides up to 7.2 VDC to the motor driver H-bridge FET switches to power the motors.

Having separate power sources insures that the power hungry and noisy motors do not interfere with the logic systems and cause brownout conditions or intermittent behavior in the MCU.

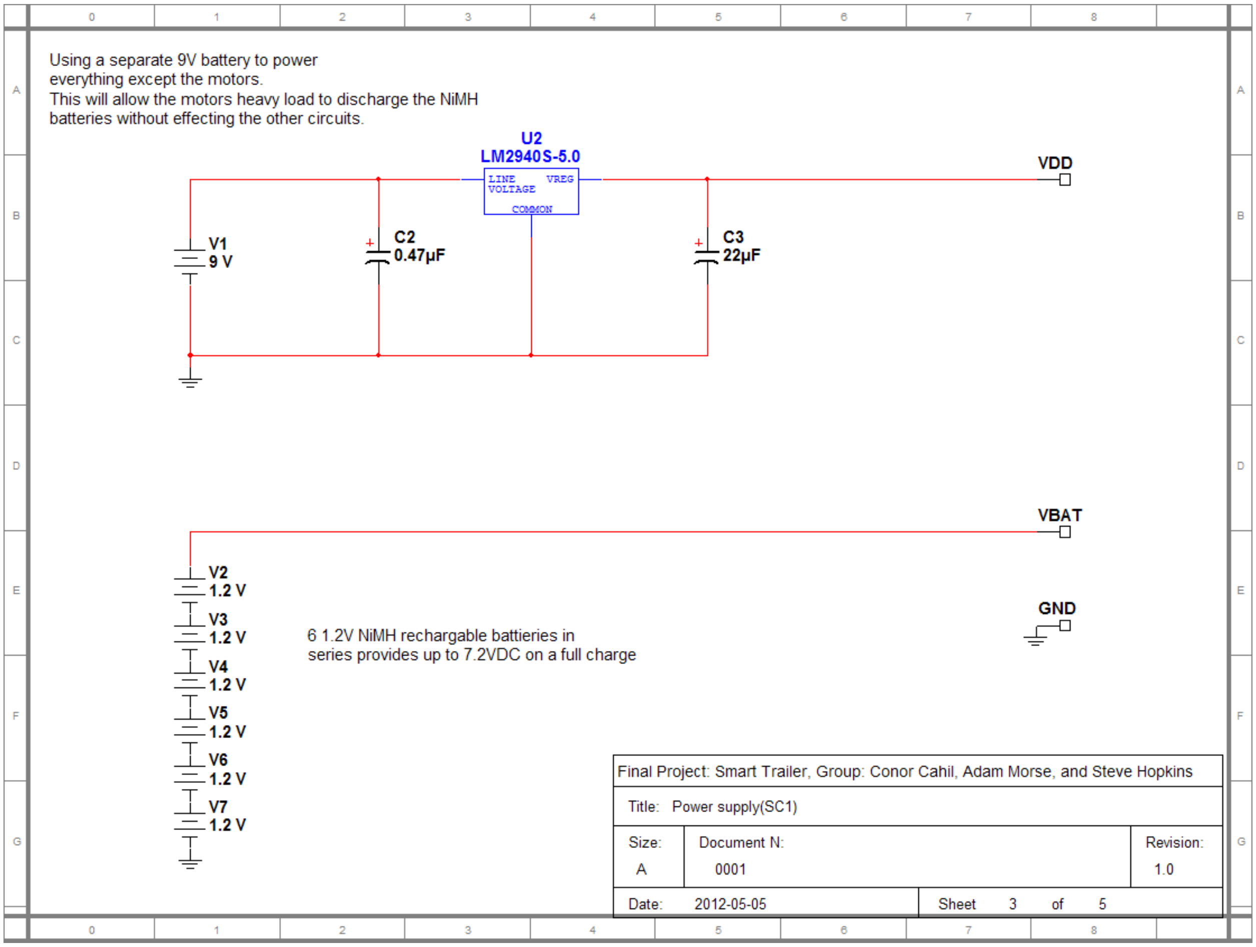
Schematics

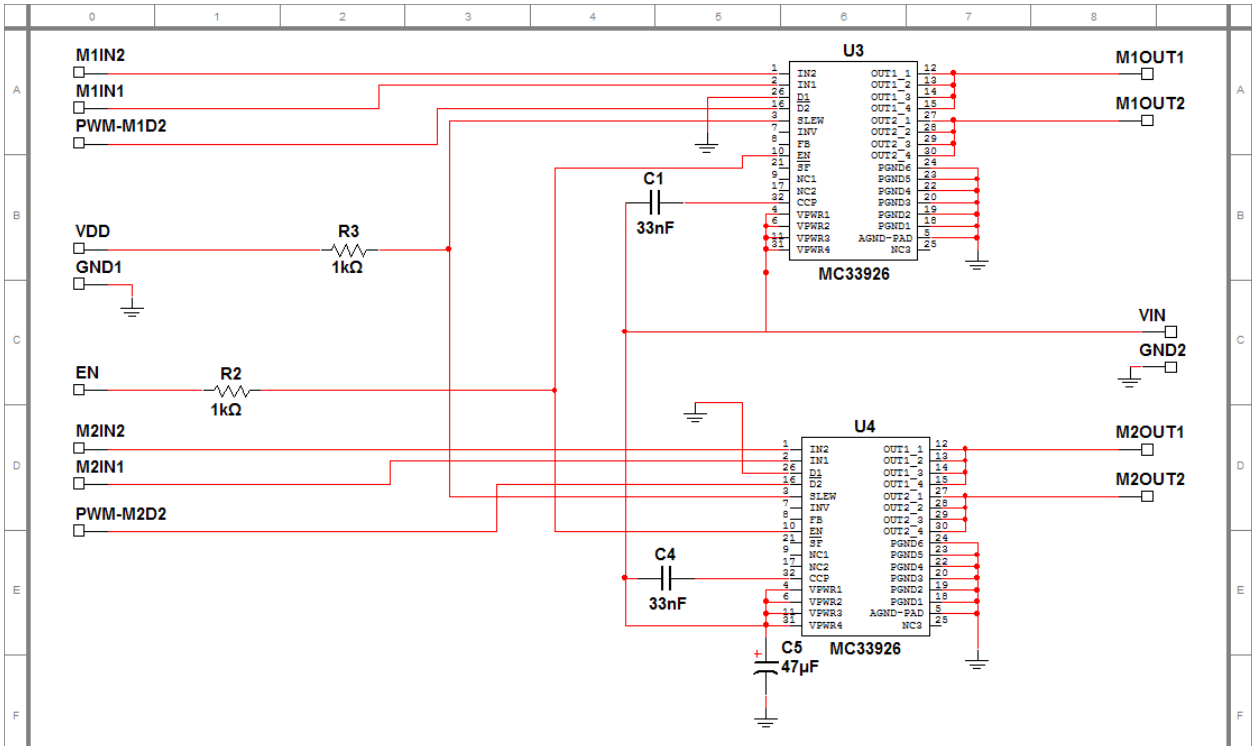


SC1
VDD
VBAT
GND
Power supply
SC3
EN
M1IN1
M1IN2
M1OUT1
M1OUT2
M2IN1
M2IN2
M2OUT1
M2OUT2
PWM-M1D2
PWM-M2D2
VDD
GND1
VIN
GND2
Dual motor driver
S1
MOTOR
S2
MOTOR
SC5
V+
GND
AN
MaxSonar
SC6
VDD
GND
SONAR_AN_IN
MOT_EN
M1DIR0
M1DIR1
M2DIR0
M2DIR1
M1_PWM
M2_PWM
HAZARD_LED
BUZZER_OUT
MCU
Hazard LED
X1
LED
R1
470Ω
U6
R4
220Ω
Back-up beeper
Buzzer-AST1240MLTRQ
Final Project: Smart Trailer, Group: Conor Cahil, Adam Morse, and Steve Hopkins
Title: Smart_Trailer_Top_Level_no_enc
Size: A
Document N: Main page
Revision: 1.0
Date: 2012-05-05
Sheet 1 of 5

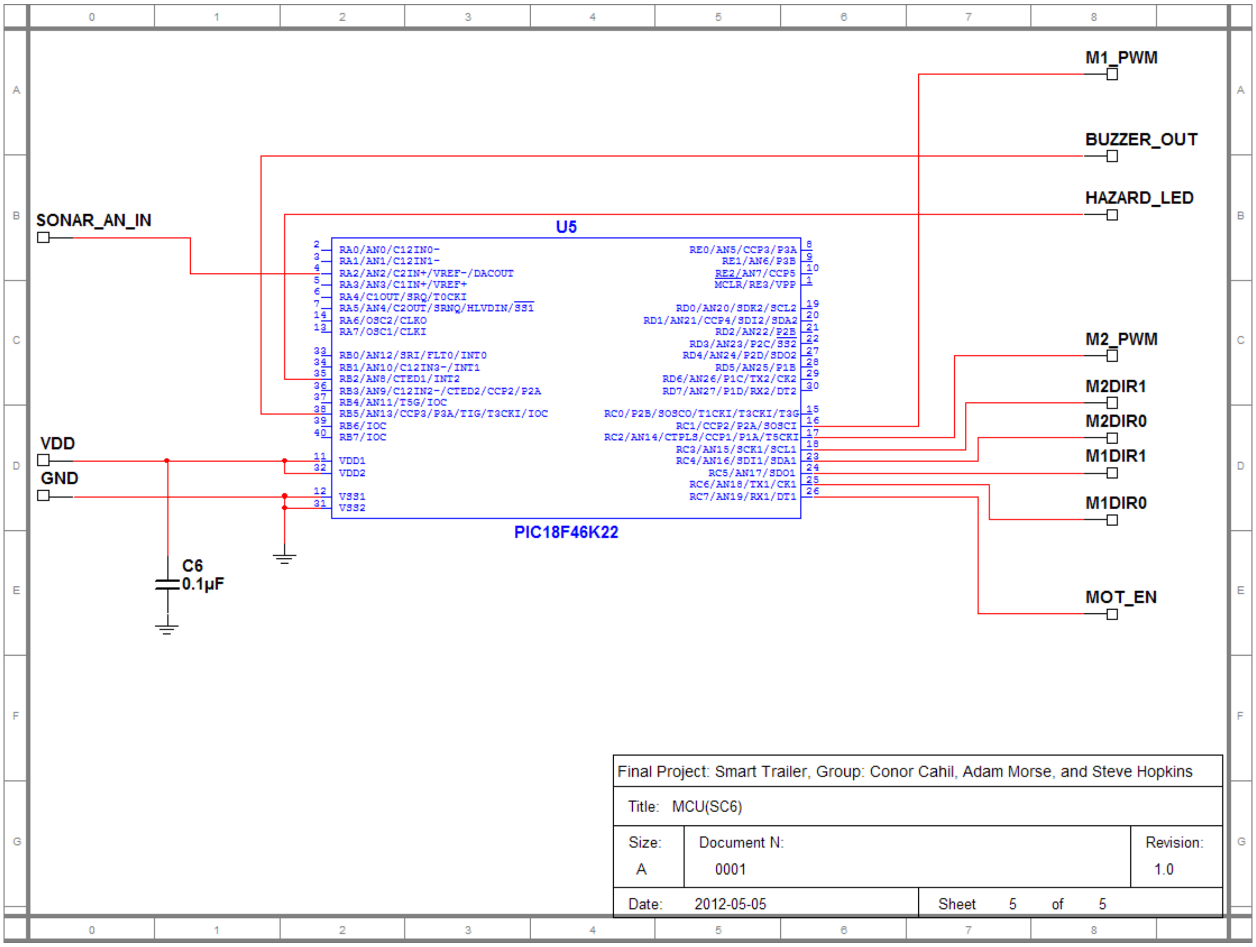


Final Project: Smart Trailer, Group: Conor Cahil, Adam Morse, and Steve Hopkins		
Title: MaxSonar(SC5)		
Size: A	Document N: 0001	Revision: 1.0
Date: 2012-05-05	Sheet 2 of 5	


Using a separate 9V battery to power everything except the motors.
This will allow the motors heavy load to discharge the NiMH batteries without effecting the other circuits.
U2
LM2940S-5.0
LINE VOLTAGE
VREG
COMMON
VDD
V1
9 V
C2
0.47µF
C3
22µF
VBAT
GND
V2
1.2 V
V3
1.2 V
V4
1.2 V
V5
1.2 V
V6
1.2 V
V7
1.2 V
6 1.2V NiMH rechargable battiers in series provides up to 7.2VDC on a full charge
Final Project: Smart Trailer, Group: Conor Cahil, Adam Morse, and Steve Hopkins
Title: Power supply(SC1)
Size: A
Document N: 0001
Revision: 1.0
Date: 2012-05-05
Sheet 3 of 5



Final Project: Smart Trailer, Group: Conor Cahil, Adam Morse, and Steve Hopkins		
Title: Dual motor driver(SC3)		
Size: A	Document N: 0001	Revision: 1.0
Date: 2012-05-05	Sheet 4 of 5	



M1_PWM
BUZZER_OUT
HAZARD_LED
SONAR_AN_IN
U5
M2_PWM
M2DIR1
M2DIR0
M1DIR1
M1DIR0
MOT_EN
VDD
GND
PIC18F46K22
C6
0.1µF
Final Project: Smart Trailer, Group: Conor Cahil, Adam Morse, and Steve Hopkins
Title: MCU(SC6)
Size: A
Document N: 0001
Revision: 1.0
Date: 2012-05-05
Sheet 5 of 5

Test Plan and Results

The objective in testing the Autonomous Trailer is to determine the success of both the hardware and software design. Each functional component of the system will have a test assigned to it, and will be evaluated based on an appropriate criteria. Some components cannot be adequately tested quantitatively and will be given a qualitative assessment. After testing each component as independently as possible, the system as a whole will be evaluated as far as it meets the design objectives.

1. **Distance Sensor**
2. **Data Processing**
3. **System Operation and Motor Control**

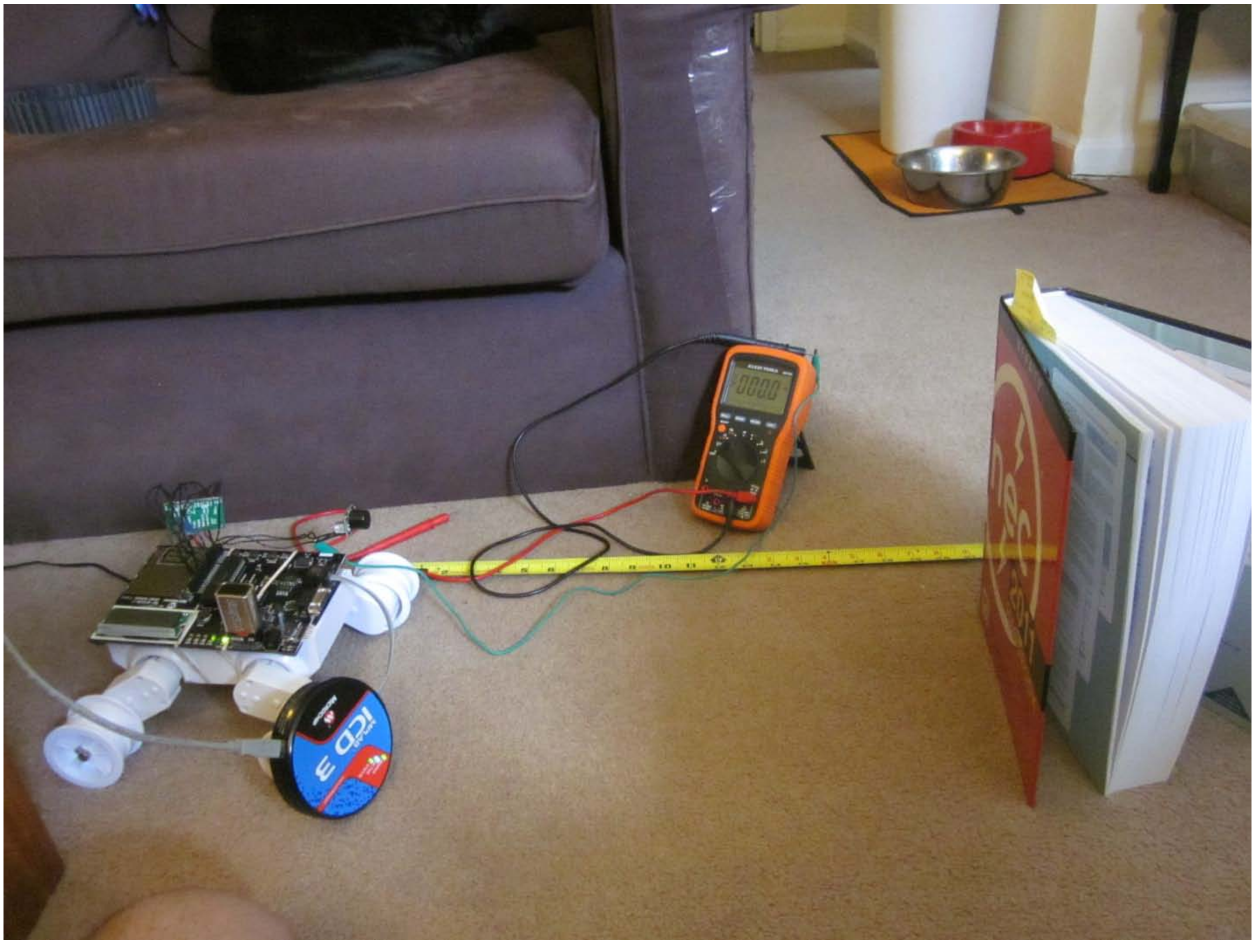
1. Distance Sensor

The distance sensor will be tested under three physical conditions: 20cm, 50cm and 100cm. 20cm indicates the smallest value that the sensor can reliably read and 100cm represents a high value for what we'd expect to measure. For our purposes, we also test the sensor at 50cm because that is our Trailer's defined set point. At each distance, with an obstruction appropriately placed, an analog and digital reading will be taken. Analog measurement will be taken directly from the sensor output with a digital multimeter; while digital measurements will be taken using the watch-window in the MPLAB development environment.

20cm:
Expected analog reading: 98mV (4.9mV/cm per datasheet)
Expected digital reading: 20
Observed analog reading: 108 mV
Observed digital reading: 0x16 or 22

50cm:
Expected analog reading: 245mV (4.9mV/cm per datasheet)
Expected digital reading: 50
Observed analog reading: 246mV
Observed digital reading:0x30 or 36

100cm:
Expected analog reading: 490mV (4.9mV/cm per datasheet)
Expected digital reading: 100
Observed analog reading: 491mV
Observed digital reading: 0x67 or 48



Conclusions:

The conclusions for the distance sensor are impressive. The inaccuracies of the voltmeter and the manually measured distances may account for the differences between expected and observed results. We were also impressed at how well our ADC translated the analog values to accurate digital ones. A future study might be made of the accuracy of ultrasonic sensors versus other types of sensors (eg infrared).

2. Data Processing

This software section of the system translates information from the distance sensor to a useable value for the motor control stage. It needs to compare the digital distance sensor values it receives to a pre-defined set-point (for our purposes, 50cm), and adjust the motor output value to force the vehicle to better approximate the set-point through its speed.

By simulating different distance situations, we can show the processing stages ability to adjust the speed of the vehicle either up or down. Some of the situations are as follows. When the distance sensor is continually out of range high, we want the processing stage to recognize this as an error, bring the vehicle to a stop and put the hazards on. When the distance is high, but not out of range, we want the processing unit to output a high incremented speed value. When the distance is low, we expect a lower incremented value. When the distance is right on (50cm), we want to maintain the trailer's current distance. Likewise we want to test that the processing unit signals a deceleration when the distance is below the set-point. All of these tests can be verified by using the watch window functionality of the MPLAB environment.

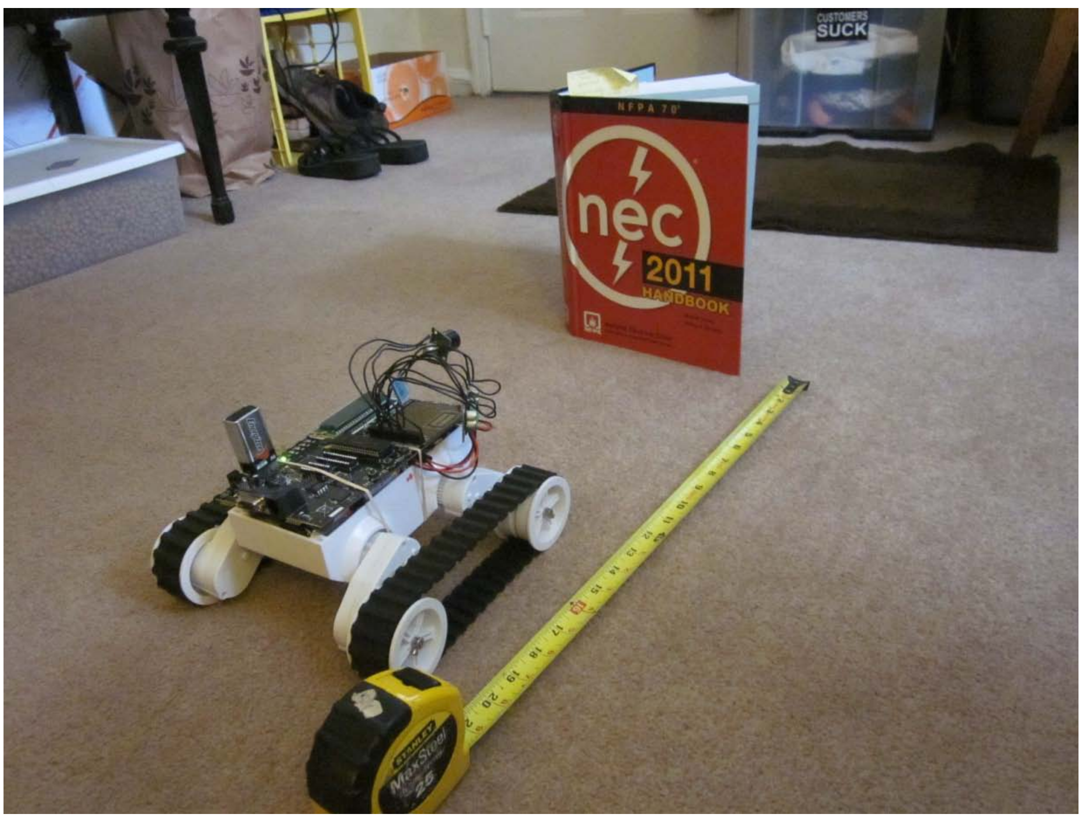
3. System Operation and Motor Control

For this section, we are interested in the system's ability as a whole to serve the defined objective. That objective is to maintain a defined following distance from a lead vehicle or simulated obstruction. This is a qualitative test.

For the first test (the Collision Test), we run the Autonomous Trailer with a simulated lead vehicle in front of it, maintaining the defined 50cm. When the lead vehicle comes to an abrupt stop, we want to verify that the Autonomous Trailer quickly comes to a stop as well. Our group tested with more primitive design than is currently under development (improvements were conceived based partly on these tests). In this case, the PWM duty-cycle for the motors was basically made identical to the value of the distance sensor.

1. Collision Test:

After running smoothly at its max speed, our Autonomous Vehicle came to a stop 30cm before the fixed obstruction.



Conclusion:

The reason for the 30cm stopping distance can be interpolated from our design. To be clear, this test was not run with the 50cm set-point that we devised for later designs. For this test, the distance values directly controlled the PWM duty-cycle. At 30cm the PWM duty-cycle was reduced to such a level that the motors could not overcome the resistance of the tracks and the friction of the floor. We know 30cm is less than 50% duty-cycle, and that appears to have been the threshold

It was surely a success for us that the vehicle did not crash into the obstruction. For our more sophisticated design, we sought to achieve a more proportional control of the vehicle and maintain a fixed set-point through incremental control. The added functionality of moving in reverse, hazard lights and a buzzer also improved on this early design.

For the second test, we are interested in the ability of our control system and motors to respond to the gradual changes of speed in the lead vehicle. We are interested in the ability of the Trailer to gradually increase in speed, gradually decrease in speed and to switch to reverse.

2. Direction and Speed Test

For this test, we observe the vehicle traveling slowly to maintain a distance and then accelerating to catch up with a speeding lead car. Clearly the vehicle's max speed is often less than that of the moving obstruction. This means that Autonomous Trailer is frequently operating at its peak speed.

One of the conditions we were concerned about was the vehicle's oscillation around a set-point. Because of the slow speed of the vehicle (10cm/s), overshooting the set-point appeared not to be a problem. Clearly the lower % PWM duty-cycle values (as low as 50%) were inhibited by the resistance of the tracks. We may incorporate this fact into future code.

A YouTube video of the vehicle in action has been uploaded to: http://www.youtube.com/watch?v=zsaG9s-oie4

Conclusions

The goal of this project was to prove the concept that you can follow a moving object with a microcontroller and a distance sensor to maintain a set distance. The complexity was in creating the algorithm and associated software. We each successfully built our own vehicles with all of the core features working.

To make this a more robust system, a bit more code polishing may be required. In the future, adding in an encoder to measure and track the absolute speed could be implemented to improve on the design.

References

PIC ® Microcontroller: An Introduction to Software & Hardware Interfacing, Huang, ISBN 1-4018-3967-3

www.pololu.com

Appendixes

Time Tracking

Time was allocated evenly between all three group members.

	Task	Percentage of allocated time	Total
Steve Hopkins Conor Cahill Adam Morse	Research and Planning	5%	3.75 hrs
	Algorithm Development and coding	70%	52.5 hrs
	Simulation and Testing	20%	15 hrs
	Documentation	5%	3.75 hrs
		Total group hours (25hr * 3 people)	75 hrs

User Manual

1. Safety Concerns

2. Setup

3. Using Your Autonomous Vehicle

4. Shutdown and Storage

5. Troubleshooting

6. System Components

Safety Concerns

Your new Autonomous Trailer is an easy-to-use, fun demonstration tool for simulating collision avoidance systems. Yet with all electrical systems, there are important safety considerations to keep in mind.

- Adult Supervision: the Autononmous Trailer should not be used without adult supervision
- Pinch Points: exercise caution when handling the robot, as the tracks can begin moving unexpectedly creating a pinch point between the wheels and the tracks
- Unintended Additions: make no modifications to the existing system in particular do not power the device by anything other than the supplied 9V battery
- Shock Hazard: the Autonomous Trailer operates at low voltages yet it still important not to contact exposed terminations and not to remove the 9V battery while the tracks are in motion

Setup

The Trailer is easy to set up straight out of the box. Its programming is done in advance and all the pieces come pre-assembled. The following elements are all that is involved:

- If you must operate the vehicle outside, only do so in a clean environment
- Avoid operating the vehicle in the rain or in dusty/dirty conditions
- Pick the vehicle up by the chassis and avoid touching PCB boards whenever possible
- Ensure that the microcontroller demoboard always is firmly attached to the chassis assembly (via the supplied rubber bands or other means)
- Place the car on your knee (or similar block) for setup such that the treads do not make contact with any surface
- Adjust the ultrasonic distance sensor so that it is level to the ground and points directly ahead of the vehicle
- Tuck all stray wires into the cab of the vehicle so they do not make contact with the tracks

- If a ‘lead car’ is to be used, place it a couple of feet in front of where the Autonomous Trailer will begin its course
- Remove all obstructions from the straight path along where the cars will travel

3. Using Your Autonomous Vehicle

- Attach the 9V battery to the demoboard
- Plug the ENABLE pin plugged into the demoboard at RC7
- After the tracks have begun to turn, place the vehicle on the floor behind the object that is to follow
- The Autonomous Trailer is designed to track 50cm behind the lead vehicle
- It will adjust its speed, and move forward and backwards, based on the movement of the lead vehicle

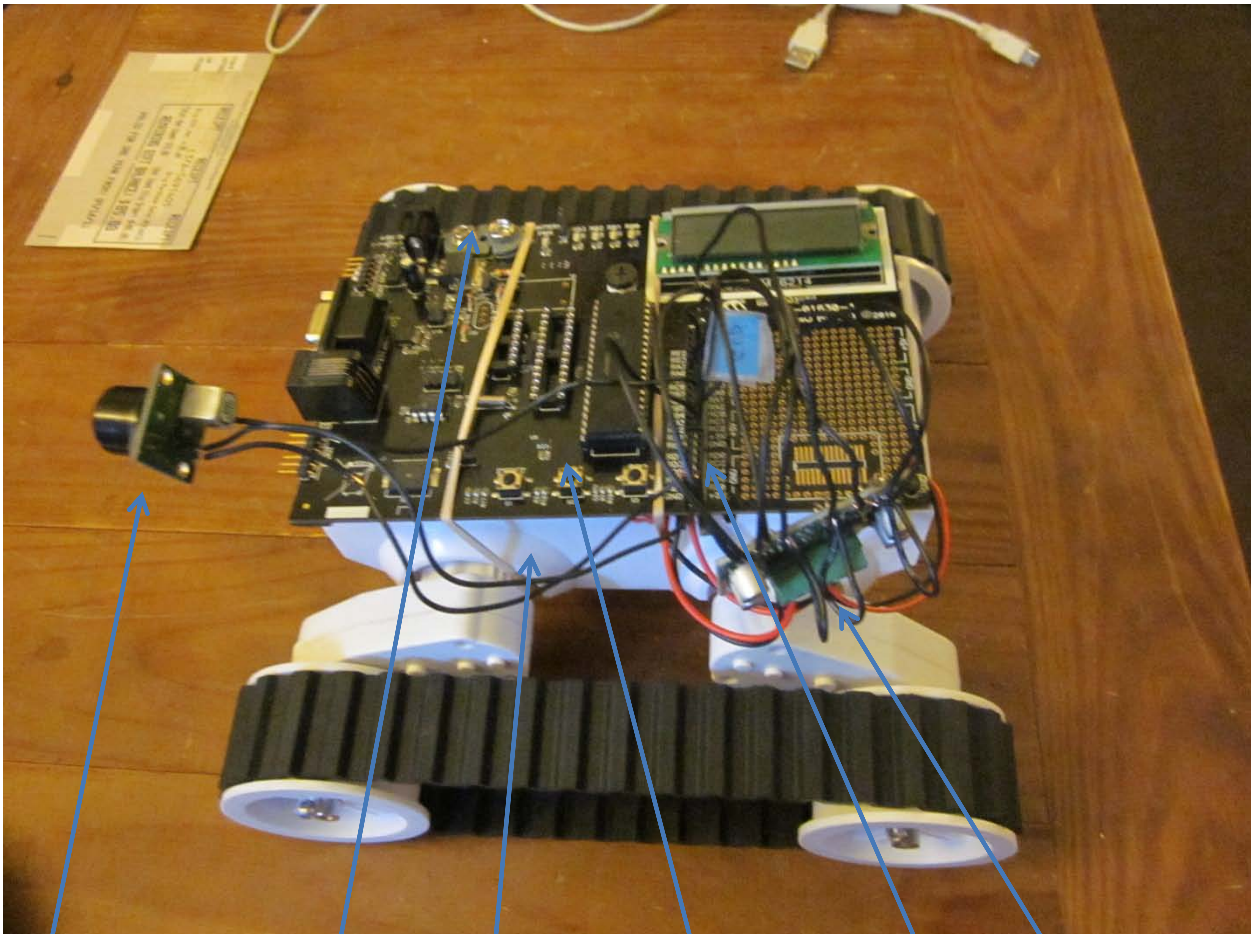
4. Shutdown and Storage

- To shut the device off, first lift it off the ground
- Remove the ENABLE pin at RC7
- Remove the 9V battery from the demoboard
- Remove the tracks from the vehicle to lessen the strain on the wheels and gearbox
- Store in a dry environment

5. Troubleshooting

- If the vehicle does not follow the target while in its operating range (45 cm to 55cm):
    - Adjust the distance sensor so that it points more directly at the object
    - Modify the lead vehicle so that it can be better recognized by the trailer (tape a vertical piece of cardboard on it, for instance)
    - Ensure that the lead vehicle does not exceed the Autonomous Trailer max speed of 10cm per second
    - Ensure that the distance sensor wires are firmly plugged into the demoboard
    - Confirm that wires are in the correct terminals (see Hardware Schematic)
- If the vehicle struggles to move:
    - Relocate to a surface with less friction
    - Ensure the all terminations between the motor controller, battery pack and demoboard are adequately connected

6. System Components



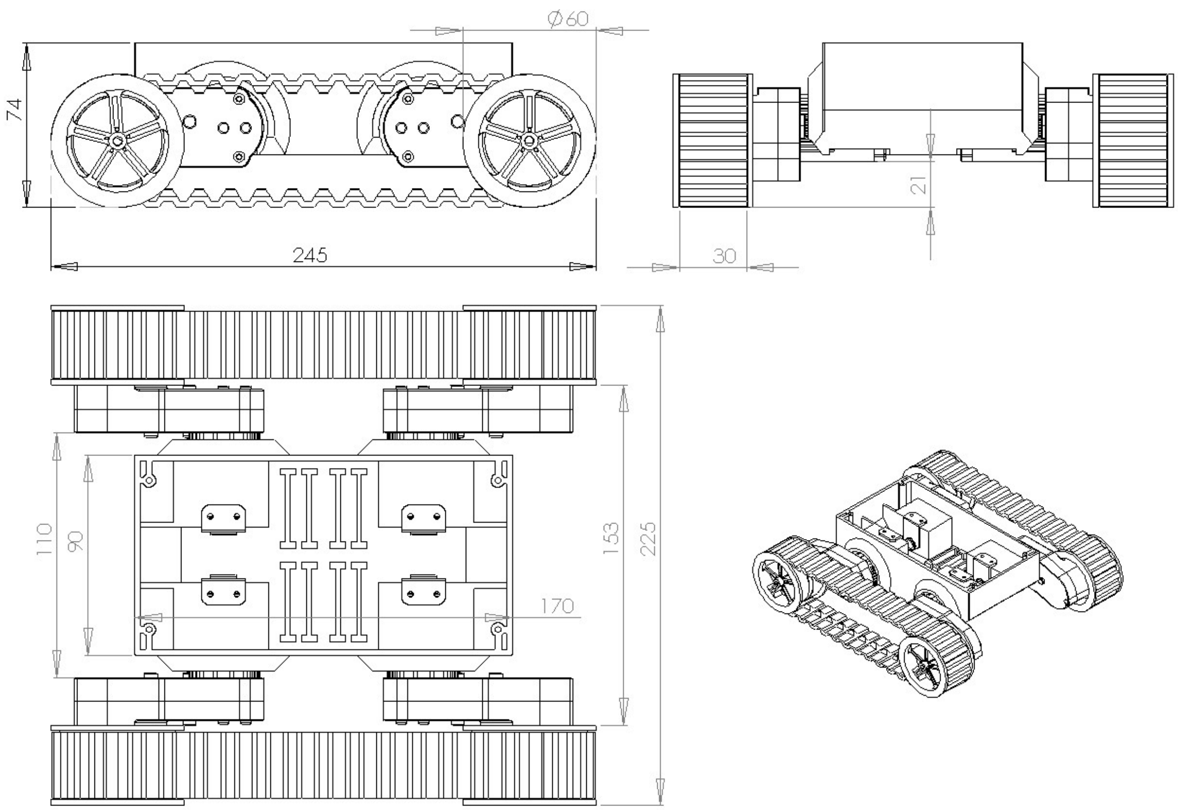
Datasheets

Rover 5

Rover 5 is a new breed of tracked robot chassis designed specifically for students and hobbyist. Unlike conventional tracked chassis's the clearance can be adjusted by rotating the gearboxes in 5-degree increments. "Stretchy" rubber treads maintain tension as the clearance is raised.
Each gearbox has an 87:1 ratio includes an optical quadrature encoder that gives 1000 pulses over 3 revolutions of the output shaft. The chassis can be upgraded to include four motors and encoders making it ideal for mecanum wheels.

Inside of the chassis are 4 noise suppression coils at the bottom and a battery holder that accepts 6x AA

batteries. It is recommended to use NiMh batteries as they last longer and have a higher current output than Alkaline batteries.
Video of the chassis in action can be seen here:
Video indoors autonomous: http://v.youku.com/v_show/id_XMjE5NzkwODA0.html
Video outdoors RC mode: http://v.youku.com/v_show/id_XMjIwMTkxODk2.html
Dimensions:



Specifications:
Motor rated voltage: 7.2V
Motor stall current: 2.5A
Output shaft stall torque: 10Kg/cm
Gearbox ratio: 86.8:1
Encoder type: Quadrature
Encoder resolution: 1000 state changes per 3 wheel rotations
Speed: 1Km/hr

Motor Driver: Freescale MC33926

Freescale Semiconductor
Advance Information

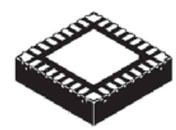
Document Number: MC33926
Rev. 8.0, 4/2009

RoHS

5.0 A Throttle Control H-Bridge

33926

AUTOMOTIVE THROTTLE H-BRIDGE ACTUATOR/ MOTOR EXCITER



Bottom View

PNB SUFFIX (Pb-FREE)
98ARL10579D
32-PIN PQFN

ORDERING INFORMATION		
Device	**Temperature Range (T_A)**	**Package**
MC33926PNB/R2	-40°C to 125°C	32 PQFN

The 33926 is a monolithic H-Bridge Power IC designed primarily for automotive electronic throttle control, but is applicable to any low-voltage DC servo motor control application within the current and voltage limits stated in this specification.

The 33926 is able to control inductive loads with currents up to 5.0 A peak. RMS current capability is subject to the degree of heatsinking provided to the device package. Internal peak-current limiting (regulation) is activated at load currents above 6.5 A ± 1.5 A. Output loads can be pulse width modulated (PWM'ed) at frequencies up to 20 kHz. A load current feedback feature provides a proportional (0.24% of the load current) current output suitable for monitoring by a microcontroller's A/D input. A Status Flag output reports under-voltage, over-current, and over-temperature fault conditions.

Two independent inputs provide polarity control of two half-bridge totem-pole outputs. Two independent disable inputs are provided to force the H-Bridge outputs to tri-state (high-impedance off-state). An inverted input changes the IN1 and IN2 inputs to LOW = true logic.

Features

- 8.0 to 28 V continuous operation (transient operation from 5.0 to 40 V)
- 225 mΩ maximum $R_{DS(ON)}$ @ 150°C (each H-Bridge MOSFET)
- 3.0 V and 5.0 V TTL / CMOS logic compatible inputs
- Over-current limiting (Regulation) via an internal constant-off-time PWM
- Output short-circuit protection (short to VPWR or ground)
- Temperature-dependent current-limit threshold reduction
- All Inputs have an internal source/sink to define the default (floating input) states
- Sleep mode with current draw < 50 µA (with inputs floating or set to match default logic states)
- Pb-free packaging designated by suffix code PNB

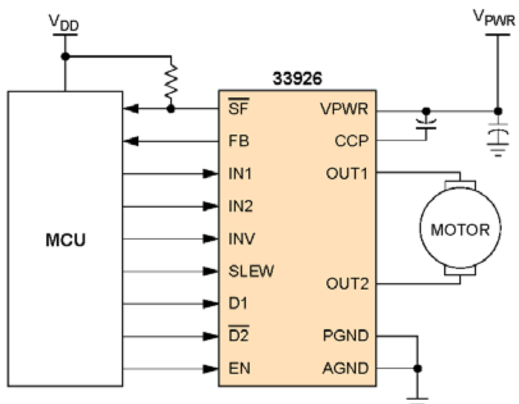


Figure 1. 33926 Simplified Application Diagram

* This document contains certain information on a new product.
Specifications and information herein are subject to change without notice.

freescale™ semiconductor

INTERNAL BLOCK DIAGRAM

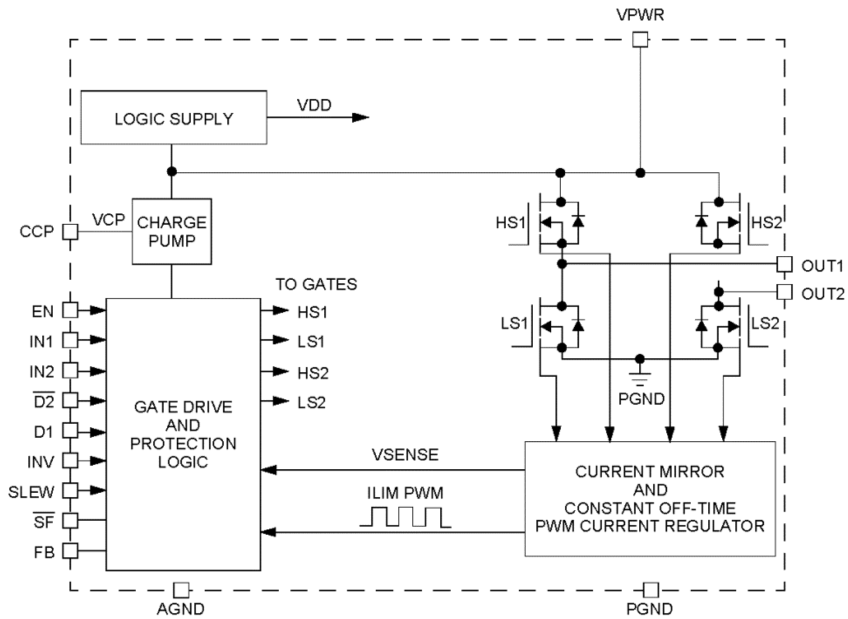


Figure 2. 33926 Simplified Internal Block Diagram

PIN CONNECTIONS

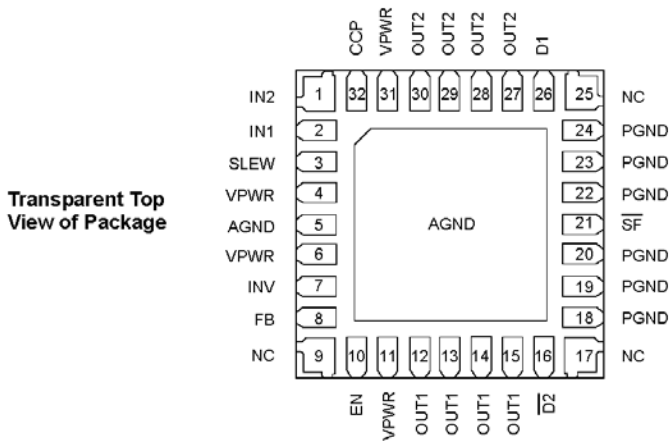


Figure 3. 33926 Pin Connections

Table 1. 33926 Pin Definitions

A functional description of each pin can be found in the Functional Description section beginning on page 12.

Pin	Pin Name	Pin Function	Formal Name	Definition
1	IN2	Logic Input	Input 2	Logic input control of OUT2; e.g., when IN2 is logic HIGH, OUT2 is set to V_{PWR}, and when IN2 is logic LOW, OUT2 is set to PGND. (Schmitt trigger input with ~80 μA source so default condition = OUT2 HIGH.)
2	IN1	Logic Input	Input 1	Logic input control of OUT1; e.g., when IN1 is logic HIGH, OUT1 is set to V_{PWR}, and when IN1 is logic LOW, OUT1 is set to PGND. (Schmitt trigger Input with ~80 μA source so default condition = OUT1 HIGH.)
3	SLEW	Logic Input	Slew Rate	Logic input to select fast or slow slew rate. (Schmitt trigger input with ~80 μA sink so default condition = slow.)
4, 6, 11, 31	VPWR	Power Input	Positive Power Supply	These pins must be connected together physically as close as possible and directly soldered down to a wide, thick, low resistance supply plane on the PCB.
5, Exposed Pad	AGND	Analog Ground	Analog Signal Ground	The low current analog signal ground must be connected to PGND via low impedance path (<<10 mΩ, 0 Hz to 20 kHz). Exposed copper pad is also the main heatsinking path for the device.
7	INV	Logic Input	Input Invert	Sets IN1 and IN2 to logic LOW = TRUE. (Schmitt trigger input with ~80 μA sink so default condition = non-inverted.)
8	FB	Analog Output	Feedback	Load current feedback output provides ground referenced 0.24% of H-Bridge high-side output current. (Tie pin to GND through a resistor if not used.)
9, 17, 25	NC		No Connect	No internal connection is made to this pin.
10	EN	Logic Input	Enable Input	When EN is logic HIGH, the device is operational. When EN is logic LOW, the device is placed in Sleep mode. (logic input with ~80 μA sink so default condition = Sleep mode.)

Table 1. 33926 Pin Definitions (continued)

A functional description of each pin can be found in the Functional Description section beginning on page 12.

Pin	Pin Name	Pin Function	Formal Name	Definition
12, 13, 14, 15	OUT1	Power Output	H-Bridge Output 1	Source of high-side MOSFET1 and drain of low-side MOSFET1.
16	$\overline{D2}$	Logic Input	Disable Input 2 (Active Low)	When $\overline{D2}$ is logic LOW, both OUT1 and OUT2 are tri-stated. (Schmitt trigger input with ~80 µA sink so default condition = disabled.)
18–20, 22–24	PGND	Power Ground	Power Ground	High-current power ground pins must be connected together physically as close as possible and directly soldered down to a wide, thick, low resistance ground plane on the PCB.
21	$\overline{SF}$	Logic Output - Open Drain	Status Flag (Active Low)	Open drain active LOW status flag output (requires an external pull-up resistor to V_{DD}. Maximum permissible load current < 0.5 mA. Maximum V_{CEsat} < 0.4 V @ 0.3 mA. Maximum permissible pullup voltage < 7.0 V.)
26	$\overline{D1}$	Logic Input	Disable Input 1 (Active High)	When D1 is logic HIGH, both OUT1 and OUT2 are tri-stated. Schmitt trigger input with ~80 µA source so default condition = disabled.
27, 28, 29, 30	OUT2	Power Output	H-Bridge Output 2	Source of high-side MOSFET2 and drain of low-side MOSFET2.
32	CCP	Analog Output	Charge Pump Capacitor	External reservoir capacitor connection for internal charge pump; connected to VPWR. Allowable values are 30 to 100 ηF. Note: This capacitor is required for the proper performance of the device.

ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

Table 2. Maximum Ratings

All voltages are with respect to ground unless otherwise noted. Exceeding these ratings may cause a malfunction or permanent damage to the device. These parameters are not production tested.

Ratings	Symbol	Value	Unit
ELECTRICAL RATINGS			
Power Supply Voltage			V
Normal Operation (Steady-state)	$V_{PWR(SS)}$	-0.3 to 28	
Transient Over-voltage [1]	$V_{PWR(t)}$	-0.3 to 40	
Logic Input Voltage [2]	V_{IN}	-0.3 to 7.0	V
$\overline{SF}$ Output [3]	$V_{\overline{SF}}$	-0.3 to 7.0	V
Continuous Output Current [4]	$I_{OUT(CONT)}$	5.0	A
ESD Voltage [5]			V
Human Body Model	V_{ESD1}	±2000	
Machine Model		±200	
Charge Device Model	V_{ESD2}		
Corner Pins (1,9,17,25)		±750	
All Other Pins		±500	
THERMAL RATINGS			
Storage Temperature	T_{STG}	-65 to 150	°C
Operating Temperature [6]			°C
Ambient	T_A	-40 to 125	
Junction	T_J	-40 to 150	

Notes

1. Device will survive repetitive transient overvoltage conditions for durations not to exceed 500 ms @ duty cycle not to exceed 10%. External protection is required to prevent device damage in case of a reverse battery condition.
2. Exceeding the maximum input voltage on IN1, IN2, EN, INV, SLEW, D1, or $\overline{D2}$ may cause a malfunction or permanent damage to the device.
3. Exceeding the pullup resistor voltage on the open drain $\overline{SF}$ pin may cause permanent damage to the device.
4. Continuous output current capability is dependent on sufficient package heatsinking to keep junction temperature ≤150°C.
5. ESD1 testing is performed in accordance with the Human Body Model (C_{ZAP} = 100 pF, R_{ZAP} = 1500 Ω), ESD2 testing is performed in accordance with the Machine Model (C_{ZAP} = 200 pF, R_{ZAP} = 0 Ω), and the Charge Device Model (CDM), Robotic (C_{ZAP} = 4.0pF).
6. The limiting factor is junction temperature, taking into account the power dissipation, thermal resistance, and heat sinking provided. Brief non-repetitive excursions of junction temperature above 150°C can be tolerated provided the duration does not exceed 30 seconds maximum. (Non-repetitive events are defined as not occurring more than once in 24 hours.)

Table 2. Maximum Ratings *(continued)*

All voltages are with respect to ground unless otherwise noted. Exceeding these ratings may cause a malfunction or permanent damage to the device. These parameters are not production tested.

Ratings	Symbol	Value	Unit
Peak Package Reflow Temperature During Reflow (7), (8)	T_{PPRT}	250	°C
Approximate Junction-to-Case Thermal Resistance (9)	$R_{\theta JC}$	<1.0	°C/W

Notes

7. Pin soldering temperature limit is for 10 seconds maximum duration. Not designed for immersion soldering. Exceeding these limits may cause malfunction or permanent damage to the device.
8. Freescale's Package Reflow capability meets Pb-free requirements for JEDEC standard J-STD-020C for Peak Package Reflow Temperature and Moisture Sensitivity Levels (MSL),
9. Exposed heatsink pad plus the power and ground pins comprise the main heat conduction paths. The actual $R_{\theta JB}$ (junction-to-PC board) values will vary depending on solder thickness and composition and copper trace thickness and area. Maximum current at maximum die temperature represents ~16 W of conduction loss heating in the diagonal pair of output MOSFETs. Therefore, the $R_{\theta JA}$ must be <5.0°C/W for maximum current at 70°C ambient. Module thermal design must be planned accordingly.

STATIC ELECTRICAL CHARACTERISTICS

Table 3. Static Electrical Characteristics

Characteristics noted under conditions 8.0 V ≤ V_{PWR} ≤ 28 V, -40°C ≤ T_A ≤ 125°C, GND = 0 V unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25°C under nominal conditions unless otherwise noted.

Characteristic	Symbol	Min	Typ	Max	Unit
POWER INPUTS (VPWR)					
Operating Voltage Range [10]					V
Steady-state	$V_{PWR(SS)}$	8.0	–	28	
Transient (t < 500 ms) [11]	$V_{PWR(t)}$	–	–	40	
Quasi-functional ($R_{DS(ON)}$ May Increase by 50%)	$V_{PWR(QF)}$	5.0	–	8.0	
Sleep State Supply Current [12] EN, $\overline{D2}$, INV, SLEW = Logic [0], IN1, IN2, D1 = Logic [1], and I_{OUT} = 0 A	$I_{PWR(SLEEP)}$	–	–	50	µA
Standby Supply Current (Part Enabled) I_{OUT} = 0 A, V_{EN} = 5.0 V	$I_{PWR(STANDBY)}$	–	–	20	mA
Under-voltage Lockout Thresholds					
$V_{PWR(FALLING)}$	$V_{UVLO(ACTIVE)}$	4.15	–	–	V
$V_{PWR(RISING)}$	$V_{UVLO(INACTIVE)}$	–	–	5.0	V
Hysteresis	$V_{UVLO(HYS)}$	150	200	350	mV
CHARGE PUMP					
Charge Pump Voltage (CP Capacitor = 33 nF)	V_{CP} - V_{PWR}				V
V_{PWR} = 5.0 V		3.5	–	–	
V_{PWR} = 28 V		–	–	12	
CONTROL INPUTS					
Operating Input Voltage (EN, IN1, IN2, D1, $\overline{D2}$, INV, SLEW)	V_I	–	–	5.5	V
Input Voltage (IN1, IN2, D1, $\overline{D2}$, INV, SLEW) [13]					
Logic Threshold HIGH	V_{IH}	2.0	–	–	V
Logic Threshold LOW	V_{IL}	–	–	1.0	V
Hysteresis	V_{HYS}	250	400	–	mV
Input Voltage (EN) Threshold	V_{TH}	1.0	–	2.0	V
Logic Input Currents, VPWR = 8.0 V	I_{IN}				µA
Inputs EN, $\overline{D2}$, INV, SLEW (internal pull-downs), V_{IH} = 5.0 V		20	80	200	
Inputs IN1, IN2, D1 (internal pull-ups), VIL = 0V		-200	-80	-20	

Notes

10. Device specifications are characterized over the range of 8.0 V ≤ V_{PWR} ≤ 28 V. Continuous operation above 28 V may degrade device reliability. Device is operational down to 5.0 V, but below 8.0 V the output resistance may increase by 50 percent.
11. Device will survive the transient overvoltage indicated for a maximum duration of 500 ms. Transient not to be repeated more than once every 10 seconds.
12. $I_{PWR(SLEEP)}$ is with Sleep mode activated and EN, $\overline{D2}$, INV, SLEW = logic [0], and IN1, IN2, D1 = logic [1] or with these inputs left floating.
13. SLEW input voltage hysteresis is guaranteed by design.

Table 3. Static Electrical Characteristics (continued)

Characteristics noted under conditions 8.0 V ≤ V_{PWR} ≤ 28 V, -40°C ≤ T_A ≤ 125°C, GND = 0 V unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25°C under nominal conditions unless otherwise noted.

Characteristic	Symbol	Min	Typ	Max	Unit
POWER OUTPUTS OUT1, OUT2					
Output-ON Resistance [15], I_{LOAD} = 3.0 A	$R_{DS(ON)}$				mΩ
V_{PWR} = 8.0 V, T_J = 25°C		–	120	–	
V_{PWR} = 8.0 V, T_J = 150°C		–	–	225	
V_{PWR} = 5.0 V, T_J = 150°C		–	–	325	
Output Current Regulation Threshold	I_{LIM}				A
T_J < T_{FB}		5.2	6.5	8.0	
T_J ≥ T_{FB} (Fold back Region - see Figure 9 and Figure 11) [14]		–	4.2	–	
High Side Short-circuit Detection Threshold (Short-circuit to GND) [14]	I_{SCH}	11	13	16	A
Low Side Short-circuit Detection Threshold (Short-circuit to VPWR) [14]	I_{SCL}	9.0	11	14	A
Output Leakage Current [16], Outputs off, V_{PWR} = 28 V	$I_{OUTLEAK}$				µA
V_{OUT} = V_{PWR}		–	–	100	
V_{OUT} = Ground		–60	–	–	
Output MOSFET Body Diode Forward Voltage Drop	V_F				V
I_{OUT} = 3.0 A		–	–	2.0	
Over-temperature Shutdown [14]					°C
Thermal Limit @ T_J	T_{LIM}	175	–	200	
Hysteresis @ T_J	T_{HYS}	–	12	–	
Current Foldback at T_J [14]	T_{FB}	165	–	185	°C
Current Foldback to Thermal Shutdown Separation [14]	T_{SEP}	10	–	15	°C
HIGH SIDE CURRENT SENSE FEEDBACK					
Feedback Current (pin FB sourcing current) [17]	I_{FB}				
I_{OUT} = 0 mA		0.0	–	50	µA
I_{OUT} = 300 mA		0.0	270	750	µA
I_{OUT} = 500 mA		0.35	0.775	1.56	mA
I_{OUT} = 1.5 A		2.86	3.57	4.28	mA
I_{OUT} = 3.0 A		5.71	7.14	8.57	mA
I_{OUT} = 6.0 A		11.43	14.29	17.15	mA
STATUS FLAG [18]					
Status Flag Leakage Current [19]	$I_{\overline{SF}LEAK}$				µA
$V_{\overline{SF}}$ = 5.0 V		–	–	5.0	
Status Flag SET Voltage [20]	$V_{\overline{SF}LOW}$				V
$I_{\overline{SF}}$ = 300 µA		–	–	0.4	

Notes

14. This parameter is guaranteed by design.
15. Output-ON resistance as measured from output to VPWR and from output to GND.
16. Outputs switched OFF via D1 or $\overline{D2}$.
17. Accuracy is better than 20% from 0.5 to 6.0 A. Recommended terminating resistor value: R_{FB} = 270 Ω.
18. Status Flag output is an open drain output requiring a pull-up resistor to logic V_{DD}.
19. Status Flag Leakage Current is measured with Status Flag HIGH and *not* SET.
20. Status Flag Set Voltage measured with Status Flag LOW and SET with $I_{\overline{FS}}$ = 300 µA. Maximum allowable sink current from this pin is < |500 µA|. Maximum allowable pull-up voltage < 7.0 V.

DYNAMIC ELECTRICAL CHARACTERISTICS

Table 4. Dynamic Electrical Characteristics

Characteristics noted under conditions 8.0 V ≤ V_{PWR} ≤ 28 V, -40°C ≤ T_A ≤ 125°C, GND = 0 V unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25°C under nominal conditions unless otherwise noted.

Characteristic	Symbol	Min	Typ	Max	Unit
TIMING CHARACTERISTICS					
PWM Frequency [21]	f_{PWM}	–	–	20	kHz
Maximum Switching Frequency During Current Limit Regulation [22]	f_{MAX}	–	–	20	kHz
Output ON Delay [23] V_{PWR} = 14 V	t_{DON}	 –	 –	 18	µs
Output OFF Delay [23] V_{PWR} = 14 V	t_{DOFF}	 –	 –	 12	µs
I_{LIM} Output Constant-OFF Time [24]	t_A	15	20.5	32	µs
I_{LIM} Blanking Time [25]	t_B	12	16.5	27	µs
Disable Delay Time [26]	$t_{DDISABLE}$	–	–	8.0	µs
Output Rise and Fall Time [27] SLEW = SLOW SLEW = FAST	t_F, t_R	 1.5 0.2	 3.0 –	 6.0 1.45	µs
Short-circuit/Over-temperature Turn-OFF (Latch-OFF) Time [28] [29]	t_{FAULT}	–	–	8.0	µs
Power-ON Delay Time [29]	t_{POD}	–	1.0	5.0	ms
Output MOSFET Body Diode Reverse Recovery Time [29]	t_{RR}	75	100	150	ns
Charge Pump Operating Frequency [29]	f_{CP}	–	7.0	–	MHz

Notes

21. The maximum PWM frequency is obtained when the device is set to Fast Slew Rate via the SLEW pin. PWM-ing when SLEW is set to SLOW should be limited to frequencies < 11 kHz in order to allow the internal high side driver circuitry time to fully enhance the high side MOSFETs.
22. The internal current limit circuitry produces a constant-OFF-time Pulse Width Modulation of the output current. The output load's inductance, capacitance, and resistance characteristics affect the total switching period (OFF-time + ON-time), and thus the PWM frequency during current limit.
23. Output Delay is the time duration from 1.5 V on the IN1 or IN2 input signal to the 20% or 80% point (dependent on the transition direction) of the OUT1 or OUT2 signal. If the output is transitioning HIGH-to-LOW, the delay is from 1.5 V on the input signal to the 80% point of the output response signal. If the output is transitioning LOW-to-HIGH, the delay is from 1.5 V on the input signal to the 20% point of the output response signal. See Figure 4, page 10.
24. The time during which the internal constant-OFF time PWM current regulation circuit has tri-stated the output bridge.
25. The time during which the current regulation threshold is ignored so that the short-circuit detection threshold comparators may have time to act.
26. Disable Delay Time measurement is defined in Figure 5, page 10.
27. Rise Time is from the 10% to the 90% level and Fall Time is from the 90% to the 10% level of the output signal with V_{PWR} = 14 V, R_{LOAD} = 3.0 ohm. See Figure 6, page 10.
28. Load currents ramping up to the current regulation threshold become limited at the I_{LIM} value (see Figure 7). The short-circuit currents possess a di/dt that ramps up to the I_{SCH} or I_{SCL} threshold during the I_{LIM} blanking time, registering as a short-circuit event detection and causing the shutdown circuitry to force the output into an immediate tri-state latch-OFF (see Figure 8). Operation in Current Limit mode may cause junction temperatures to rise. Junction temperatures above ~160°C will cause the output current limit threshold to "fold back", or decrease, until ~175°C is reached, after which the T_{LIM} thermal latch-OFF will occur. Permissible operation within this fold back region is limited to non-repetitive transient events of duration not to exceed 30 seconds (see Figure 9).
29. Parameter is guaranteed by design.

TIMING DIAGRAMS

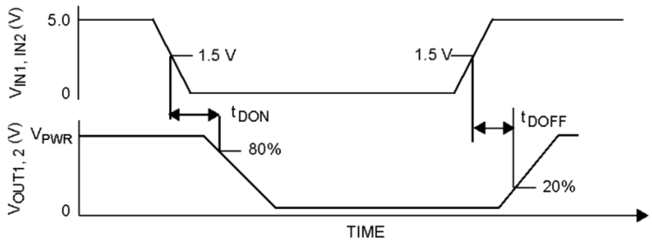


Figure 4. Output Delay Time

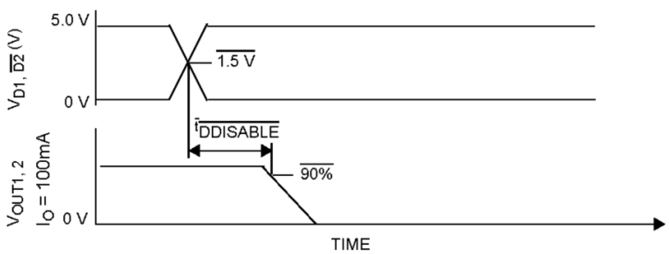


Figure 5. Disable Delay Time

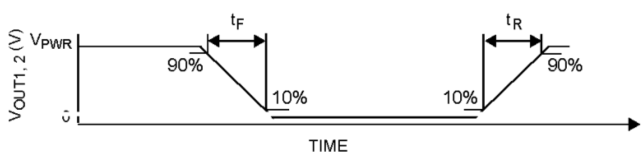


Figure 6. Output Switching Time

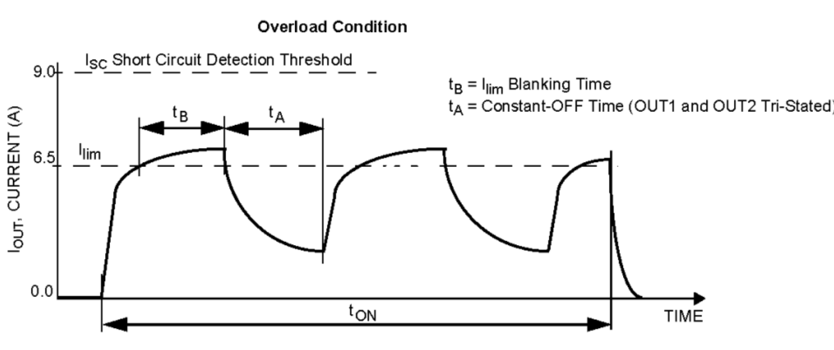


Figure 7. Current Limit Blanking Time and Constant-OFF Time

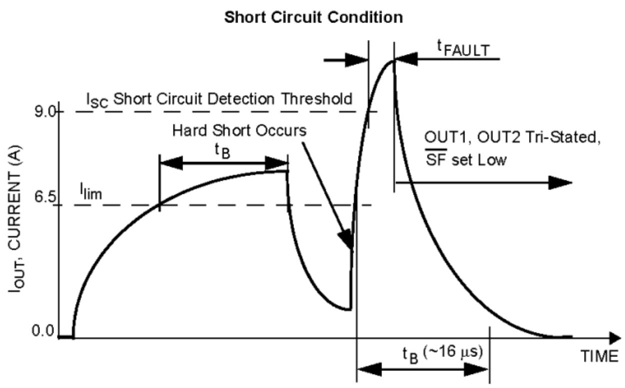


Figure 8. Short-circuit Detection Turn-OFF Time t_{FAULT}

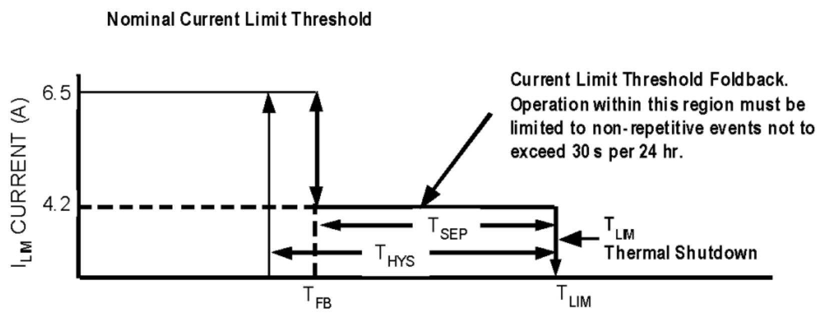


Figure 9. Output Current Limiting Foldback Region

FUNCTIONAL DESCRIPTION

INTRODUCTION

Numerous protection and operational features (speed, torque, direction, dynamic breaking, PWM control, and closed-loop control) make the 33926 a very attractive, cost-effective solution for controlling a broad range of small DC motors. The 33926 outputs are capable of supporting peak DC load currents of up to 5.0 A from a 28 V_{PWR} source. An internal charge pump and gate drive circuitry are provided that can support external PWM frequencies up to 20 kHz.

The 33926 has an analog feedback (current mirror) output pin (the FB pin) that provides a constant-current source ratioed to the active high side MOSFETs' current. This can be used to provide "real time" monitoring of output current to facilitate closed-loop operation for motor speed/torque control, or for the detection of open load conditions.

Two independent inputs, IN1 and IN2, provide control of the two totem-pole half-bridge outputs. An input invert, INV, changes IN1 and IN2 to LOW = true logic. Two different output slew rates are selectable via the SLEW input. Two independent disable inputs, D1 and $\overline{D2}$, provide the means to force the H-Bridge outputs to a high-impedance state (all H-Bridge switches OFF). An EN pin controls an enable function that allows the IC to be placed in a power-conserving Sleep mode.

The 33926 has Output Current Limiting (via Constant OFF-Time PWM Current Regulation), Output Short-circuit Detection with Latch-OFF, and Over-temperature Detection with Latch-OFF. Once the device is latched-OFF due to a fault condition, either of the Disable inputs (D1 or $\overline{D2}$), V_{PWR}, or EN must be "toggled" to clear the status flag.

Current limiting (Load Current Regulation) is accomplished by a constant-OFF time PWM method using current limit threshold triggering. The current limiting scheme is unique in that it incorporates a junction temperature-dependent current limit threshold. This means that the current limit threshold is "reduced to around 4.2 A" as the junction temperature increases above 160°C. When the temperature is above 175°C, over-temperature shutdown (latch-OFF) will occur. This combination of features allows the device to continue operating for short periods of time (<30 seconds) with unexpected loads, while still retaining adequate protection for both the device and the load.

FUNCTIONAL PIN DESCRIPTION

POWER GROUND AND ANALOG GROUND (PGND AND AGND)

The power and analog ground pins should be connected together with a very low-impedance connection.

POSITIVE POWER SUPPLY (VPWR)

VPWR pins are the power supply inputs to the device. All VPWR pins must be connected together on the printed circuit board with as short as possible traces, offering as low-impedance as possible between pins.

Transients on V_{PWR} which go below the Under-voltage Threshold will result in the protection activating. It is essential to use an input filter capacitor of sufficient size and low ESR to sustain a V_{PWR} greater than V_{UVLO} when the load is switched (See 33926 Typical Application Schematic on page 18).

STATUS FLAG ($\overline{SF}$)

This pin is the device fault status output. This output is an active LOW open drain structure requiring a pull-up resistor to V_{DD}. The maximum V_{DD} is <7.0 V. Refer to Table 5, Truth Table, page 16 for the SF Output status definition.

INPUT INVERT (INV)

The Input Invert Control pin sets IN1 and IN2 to LOW = TRUE. This is a Schmitt trigger input with ~80 μA sink; the default condition is non-inverted. If IN1 and IN2 are set so that the current is being commanded to flow through the load attached between OUT1 and OUT2, changing the logic level at INV will have the effect of reversing the direction of current commanded. Thus, the INV input may be used as a "forward/reverse" command input. If both IN1 and IN2 are the same logic level, then changing the logic level at INV will have the effect of changing the bridge's output from freewheeling high to freewheeling low or vice versa.

SLEW RATE (SLEW)

The SLEW pin is the logic input that selects fast or slow slew rate. Schmitt trigger input with ~80 μA sink so the default condition is SLOW. When SLEW is set to SLOW, PWM-ing should be limited to frequencies less than 11 kHz in order to allow the internal high-side driver circuitry time to fully enhance the high-side MOSFETs.

INPUT 1,2 AND DISABLE INPUT 1,2 (IN1, IN2, AND D1, $\overline{D2}$)

These pins are input control pins used to control the outputs. These pins are 3.0 V/5.0 V CMOS-compatible inputs with hysteresis. IN1 and IN2 independently control OUT1 and OUT2, respectively. D1 and $\overline{D2}$ are complementary inputs used to tri-state disable the H-Bridge outputs.

When either D1 or $\overline{D2}$ is SET (D1 = logic HIGH or $\overline{D2}$ = logic LOW) in the disable state, outputs OUT1 and OUT2 are both tri-state disabled; however, the rest of the device circuitry is fully operational and the supply $I_{PWR(STANDBY)}$ current is reduced to a few mA. Refer to Table 3. Static Electrical Characteristics, page 7.

H-BRIDGE OUTPUT (OUT1, OUT2)

These pins are the outputs of the H-Bridge with integrated freewheeling diodes. The bridge output is controlled using the IN1, IN2, D1, and $\overline{D2}$ inputs. The outputs have PWM current limiting above the I_{LIM} threshold. The outputs also have thermal shutdown (tri-state latch-OFF) with hysteresis as well as short-circuit latch-OFF protection.

A disable timer (time t_B) is incorporated to distinguish between load currents that are higher than the I_{LIM} threshold and short-circuit currents. This timer is activated at each output transition.

CHARGE PUMP CAPACITOR (CCP)

This pin is the charge pump output pin and connection for the external charge pump reservoir capacitor. The allowable value is from 30 nF to 100 nF. This capacitor must be connected from the CCP pin to the VPWR pin. The device cannot operate properly without the external reservoir capacitor.

ENABLE INPUT (EN)

The EN pin is used to place the device in a Sleep mode so as to consume very low currents. When the EN pin voltage is a logic LOW state, the device is in Sleep mode. The device is enabled and fully operational when the EN pin voltage is in logic HIGH. An internal pulldown resistor maintains the device in Sleep mode in the event EN is driven through a high-impedance I/O, or an unpowered microcontroller, or the EN input becomes disconnected.

FEEDBACK (FB)

The 33926 has a feedback output (FB) for "real time" monitoring of H-Bridge high-side output currents to facilitate closed-loop operation for motor speed and torque control.

The FB pin provides current sensing feedback of the H-Bridge high side drivers. When running in the forward or reverse direction, a ground-referenced 0.24% of load current is output to this pin. Through the use of an external resistor to ground, the proportional feedback current can be converted to a proportional voltage equivalent and the controlling microcontroller can "read" the current proportional voltage with its analog-to-digital converter (ADC). This is intended to provide the user with only first-order motor current feedback for motor torque control. The resistance range for the linear operation of the FB pin is $100 < R_{FB} < 300\ \Omega$.

If PWM-ing is implemented using the disable pin inputs (either D1 or $\overline{D2}$), a small filter capacitor (~1.0 μF) may be required in parallel with the R_{FB} resistor to ground for spike suppression.

FUNCTIONAL INTERNAL BLOCK DESCRIPTION

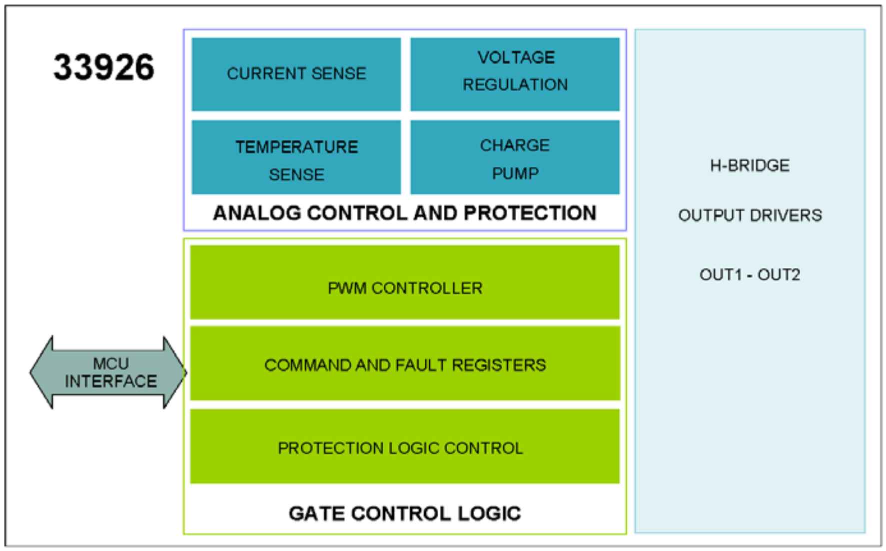


Figure 10. Functional Internal Block Diagram

ANALOG CONTROL AND PROTECTION CIRCUITRY:

The on-chip voltage regulator supplies 3.3 V to the internal logic. The charge pump provides gate drive for the H-Bridge MOSFETs. The current and temperature sense circuitry provides detection and protection for the output drivers. Output under-voltage protection shuts down the MOSFETS.

GATE CONTROL LOGIC:

The 33926 is a monolithic H-Bridge Power IC designed primarily for any low voltage DC servo motor control application within the current and voltage limits stated for the device. Two independent inputs provide polarity control of two half-bridge totem-pole outputs. Two independent disable inputs are provided to force the H-Bridge outputs to tri-state (high-impedance off-state).

H-BRIDGE OUTPUT DRIVERS: OUT1 AND OUT2

The H-Bridge is the power output stage. The current flow from OUT1 to OUT2 is reversible and under full control of the user by way of the Input Control Logic. The output stage is designed to produce full load control under all system conditions. All protective and control features are integrated into the control and protection blocks. The sensors for current and temperature are integrated directly into the output MOSFET for maximum accuracy and dependability.

FUNCTIONAL DEVICE OPERATION

OPERATIONAL MODES

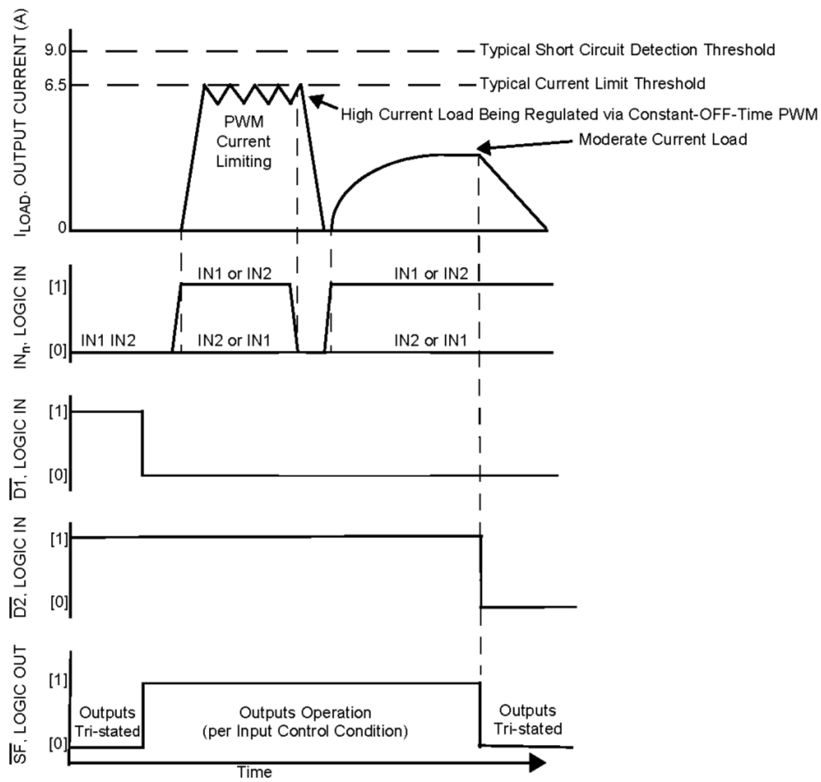


Figure 11. Operating States

LOGIC COMMANDS AND REGISTERS

Table 5. Truth Table

The tri-state conditions and the status flag are reset using D1 or $\overline{D2}$. The truth table uses the following notations: L = LOW, H = HIGH, X = HIGH or LOW, and Z = High Impedance. All output power transistors are switched off.

Device State	Input Conditions					Status	Outputs	
	EN	D1	$\overline{D2}$	IN1	IN2	$\overline{SF}$	OUT1	OUT2
Forward	H	L	H	H	L	H	H	L
Reverse	H	L	H	L	H	H	L	H
Free Wheeling Low	H	L	H	L	L	H	L	L
Free Wheeling High	H	L	H	H	H	H	H	H
Disable 1 (D1)	H	H	X	X	X	L	Z	Z
Disable 2 ($\overline{D2}$)	H	X	L	X	X	L	Z	Z
IN1 Disconnected	H	L	H	Z	X	H	H	X
IN2 Disconnected	H	L	H	X	Z	H	X	H
D1 Disconnected	H	Z	X	X	X	L	Z	Z
$\overline{D2}$ Disconnected	H	X	Z	X	X	L	Z	Z
Under-voltage Lockout (30)	H	X	X	X	X	L	Z	Z
Over-temperature (31)	H	X	X	X	X	L	Z	Z
Short-circuit (31)	H	X	X	X	X	L	Z	Z
Sleep Mode EN	L	X	X	X	X	H	Z	Z
EN Disconnected	Z	X	X	X	X	H	Z	Z

Notes

30. In the event of an under-voltage condition, the outputs tri-state and status flag is SET logic LOW. Upon under-voltage recovery, status flag is reset automatically or automatically cleared and the outputs are restored to their original operating condition.
31. When a short-circuit or over-temperature condition is detected, the power outputs are tri-state latched-OFF, independent of the input signals, and the status flag is latched to logic LOW. To reset from this condition requires the toggling of either D1, $\overline{D2}$, EN, or V_{PWR}.

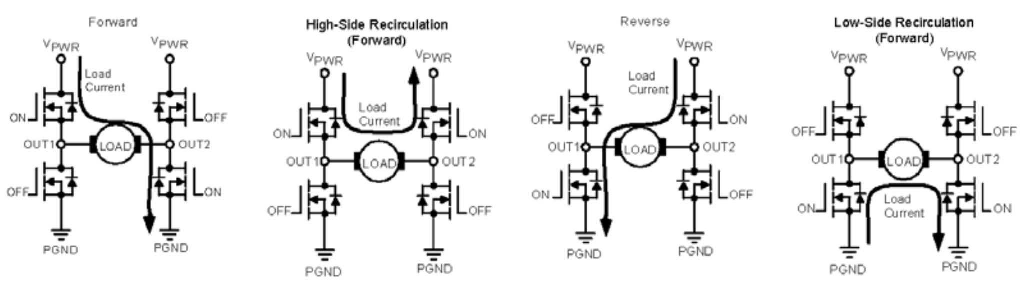


Figure 12. 33926 Power Stage Operation

PROTECTION AND DIAGNOSTIC FEATURES

SHORT-CIRCUIT PROTECTION

If an output short-circuit condition is detected, the power outputs tri-state (latch-OFF) independent of the input (IN1 and IN2) states, and the fault status output flag ($\overline{SF}$) is SET to a logic LOW. If the D1 input changes from a logic HIGH to logic LOW, or if the $\overline{D2}$ input changes from a logic LOW to logic HIGH, the output bridge will become operational again, and the fault status flag will be reset (cleared) to a logic HIGH state.

The output stage will always switch into the mode defined by the input pins (IN1, IN2, D1, and $\overline{D2}$), provided the device junction temperature is within the specified operating temperature range.

INTERNAL PWM CURRENT LIMITING

The maximum current flow under normal operating conditions should be less than 5.0 A. The instantaneous load currents will be limited to I_{LIM} via the internal PWM current limiting circuitry. When the I_{LIM} threshold current value is reached, the output stages are tri-stated for a fixed time (T_A) of 20 µs typical. Depending on the time constant associated with the load characteristics, the output current decreases during the tri-state duration until the next output ON cycle occurs.

The PWM current limit threshold value is dependent on the device junction temperature. When $-40°C < T_J < 160°C$, I_{LIM} is between the specified minimum/maximum values. When T_J exceeds 160 °C, the I_{LIM} threshold decreases to 4.2 A. Shortly above 175 °C the device over-temperature circuit will detect T_{LIM} and an over-temperature shutdown will occur. This feature implements a graceful degradation of operation before thermal shutdown occurs, thus allowing for intermittent unexpected mechanical loads on the motor's gear-reduction train to be handled.

Important Die temperature excursions above 150°C are permitted only for non-repetitive durations <30 seconds. Provision must be made at the system level to prevent prolonged operation in the current-foldback region.

OVER-TEMPERATURE SHUTDOWN AND HYSTERESIS

If an over-temperature condition occurs, the power outputs are tri-stated (latched-OFF), and the fault status flag ($\overline{SF}$) is SET to a logic LOW.

To reset from this condition, D1 must change from a logic HIGH to logic LOW, or $\overline{D2}$ must change from a logic LOW to logic HIGH. When reset, the output stage switches ON again, provided that the junction temperature is now below the over-temperature threshold limit minus the hysteresis.

Important Resetting from the fault condition will clear the fault status flag. Powering down and powering up the device will also reset the 33926 from the fault condition.

OUTPUT AVALANCHE PROTECTION

If VPWR were to become an open circuit, the outputs would likely tri-state simultaneously due to the disable logic. This could result in an unclamped inductive discharge. The VPWR input to the 33926 should not exceed 40 V during this transient condition, to prevent electrical overstress of the output drivers.This can be accomplished with a zener clamp or MOV, and/or an appropriately valued input capacitor with sufficiently low ESR (see Figure 13).

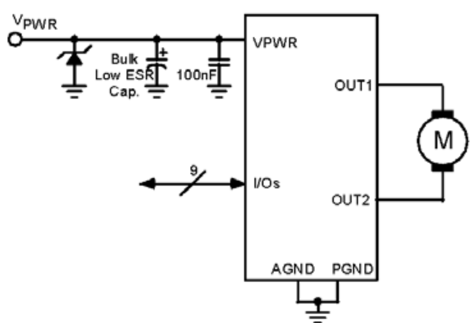


Figure 13. Avalanche Protection

TYPICAL APPLICATIONS

INTRODUCTION

A typical application schematic is shown in Figure 14. For precision high current applications in harsh, noisy environments, the V_{PWR} by-pass capacitor may need to be substantially larger.

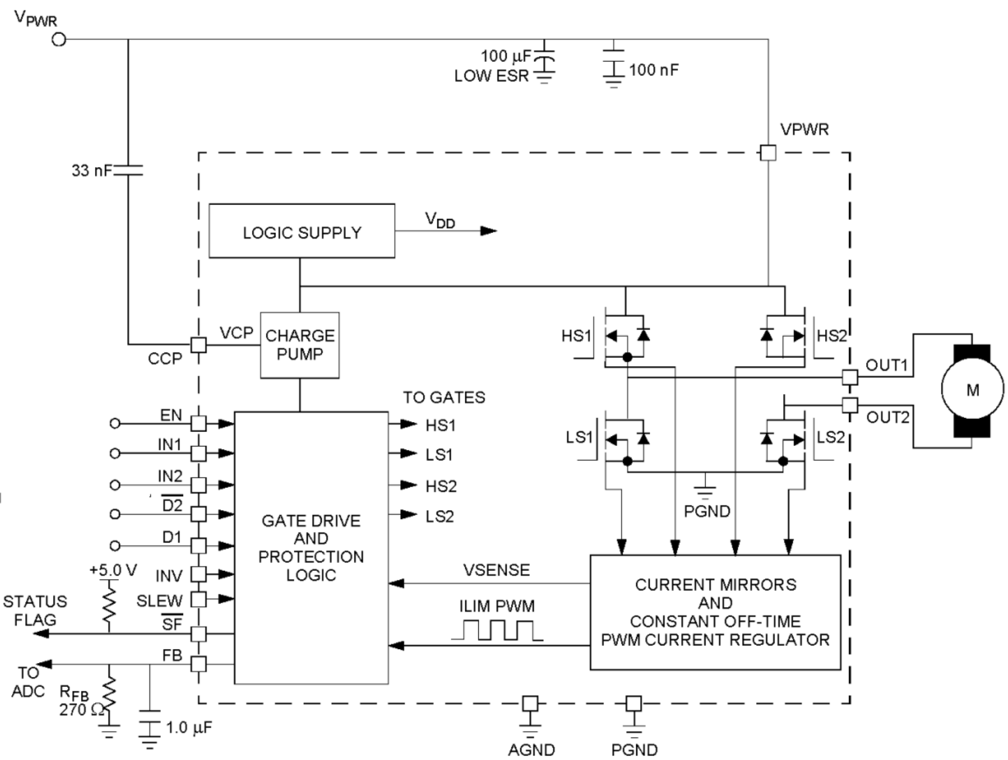
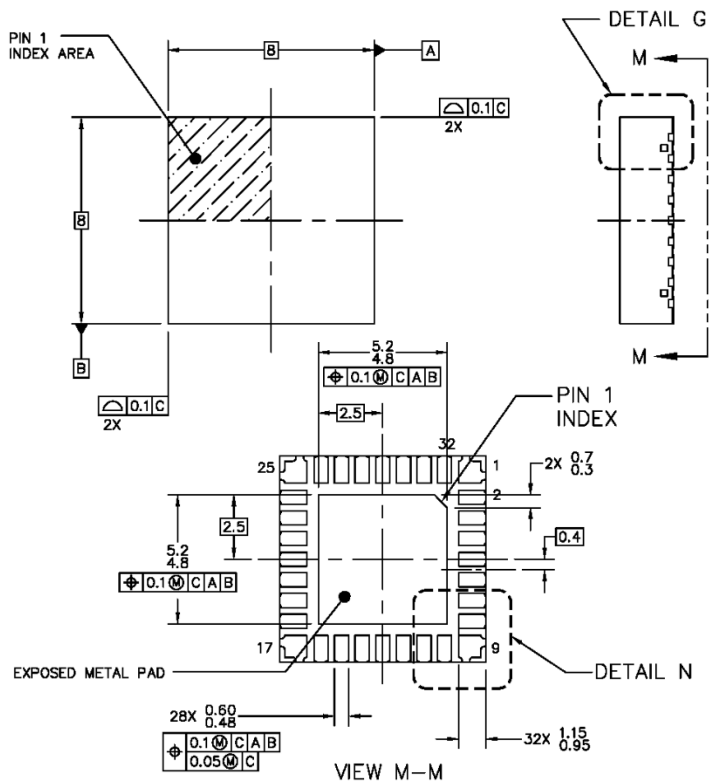


Figure 14. 33926 Typical Application Schematic

PACKAGING

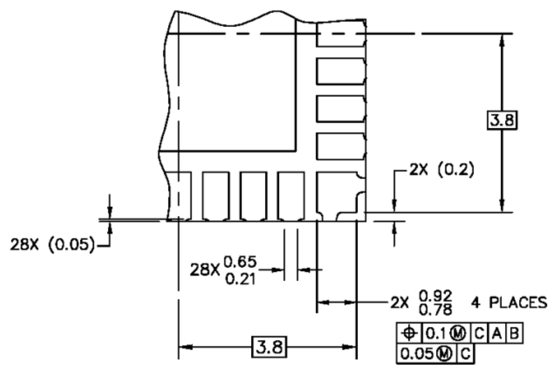
PACKAGE DIMENSIONS

For the most current package revision, visit **www.freescale.com** and perform a keyword search using the 98Axxxxxxxxx listed below.

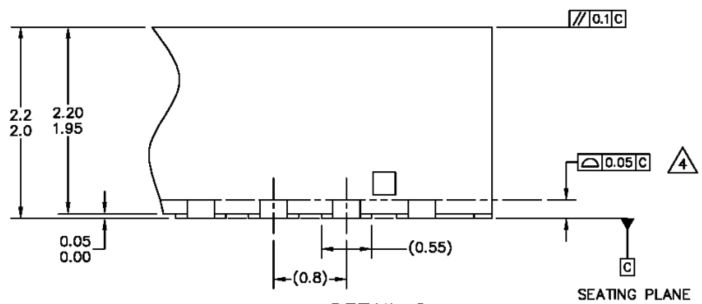


© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	**MECHANICAL OUTLINE**	PRINT VERSION NOT TO SCALE
TITLE: POWER QUAD FLAT NON–LEADED PACKAGE (PWR QFN), 32 TERMINAL, 0.8 PITCH(8X8X2.1)	DOCUMENT NO: 98ARL10579D	REV: C
	CASE NUMBER: 1536–04	31 OCT 2006
	STANDARD: JEDEC MO–251A ADDB–1	

PNB SUFFIX
98ARL10579D
32-PIN PQFN
ISSUE C



DETAIL N
CORNER CONFIGURATION



DETAIL G
VIEW ROTATED 90° CW

	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: POWER QUAD FLAT NON-LEADED PACKAGE (PWR QFN), 32 TERMINAL, 0.8 PITCH(8X8X2.1)	DOCUMENT NO: 98ARL10579D	REV: C
	CASE NUMBER: 1536-04	31 OCT 2006
	STANDARD: JEDEC MO-251A ADDB-1	

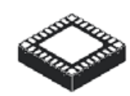
PNB SUFFIX
98ARL10579D
32-PIN PQFN
ISSUE C

ADDITIONAL DOCUMENTATION

THERMAL ADDENDUM (REV 2.0)

33926

32-PIN PQFN



PNB SUFFIX
98ARL10579D
32-PIN PQFN
8.0 mm x 8.0 mm

Note For package dimensions, refer to the 33926 data sheet.

Introduction

This thermal addendum is provided as a supplement to the 33926 technical datasheet. The addendum provides thermal performance information that may be critical in the design and development of system applications. All electrical, application, and packaging information is provided in the datasheet.

Packaging and Thermal Considerations

The 33926 is offered in a 32 pin PQFN, single die package. There is a single heat source (P), a single junction temperature (T_J), and thermal resistance ($R_{\theta JA}$).

$$\{T_J\} = [R_{\theta JA}] \cdot \{P\}$$

The stated values are solely for a thermal performance comparison of one package to another in a standardized environment. This methodology is not meant to, and will not predict the performance of a package in an application-specific environment. Stated values were obtained by measurement and simulation according to the standards listed below.

STANDARDS

Table 6. Thermal Performance Comparison

Thermal Resistance	[°C/W]
$R_{\theta JA}$ (1),(2)	28
$R_{\theta JB}$ (2),(3)	12
$R_{\theta JA}$ (1), (4)	80
$R_{\theta JC}$ (5)	1.0

Notes

1. Per JEDEC JESD51-2 at natural convection, still air condition.
2. 2s2p thermal test board per JEDEC JESD51-5 and JESD51-7.
3. Per JEDEC JESD51-8, with the board temperature on the center trace near the center lead.
4. Single layer thermal test board per JEDEC JESD51-3 and JESD51-5.
5. Thermal resistance between the die junction and the exposed pad surface; cold plate attached to the package bottom side, remaining surfaces insulated.

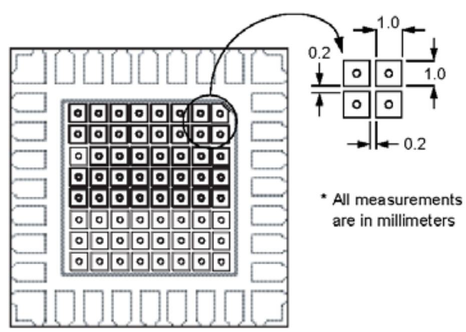
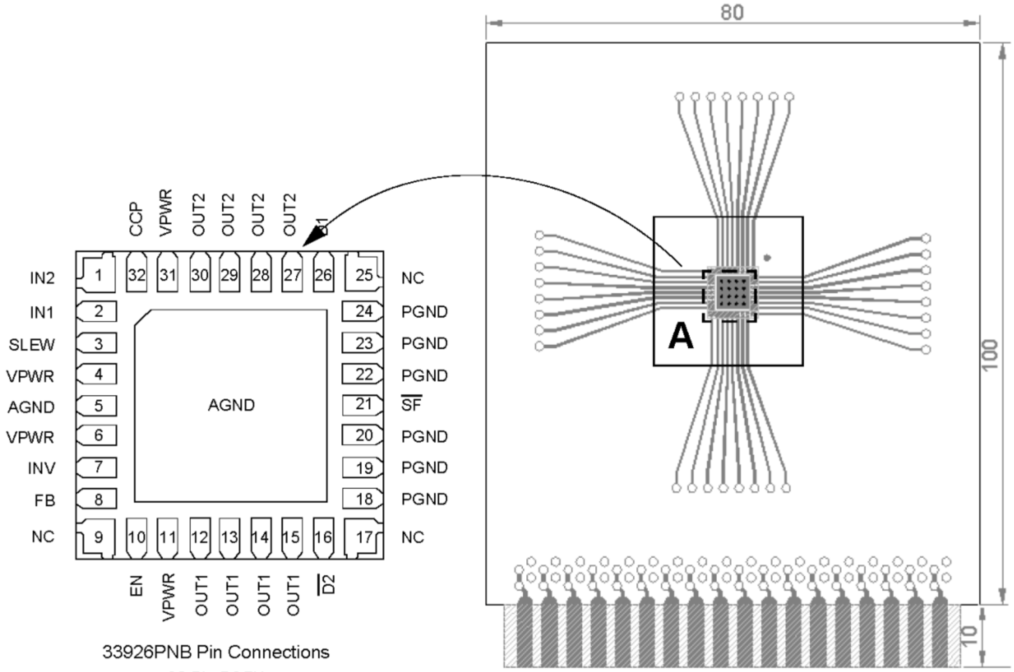


Figure 15. Surface Mount for Power PQFN with Exposed Pads



33926PNB Pin Connections

32-Pin PQFN
0.80 mm Pitch
8.0 mm x 8.0 mm Body

Figure 16. Thermal Test Board

Device on Thermal Test Board

Material:	Single layer printed circuit board FR4, 1.6 mm thickness Cu traces, 0.07 mm thickness
Outline:	80 mm x 100 mm board area, including edge connector for thermal testing
Area A:	Cu heat-spreading areas on board surface
Ambient Conditions:	Natural convection, still air

Table 7. Thermal Resistance Performance

A [mm^2]	$R_{\theta JA}$ [°C/W]
0	81
300	49
600	40

$R_{\theta JA}$ is the thermal resistance between die junction and ambient air.

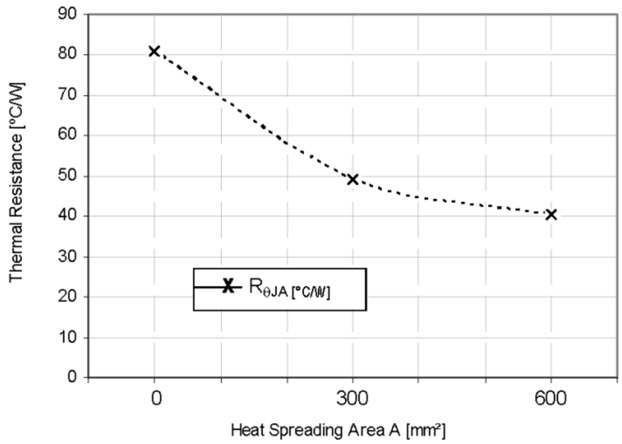


Figure 17. Device on Thermal Test Board $R_{\theta JA}$

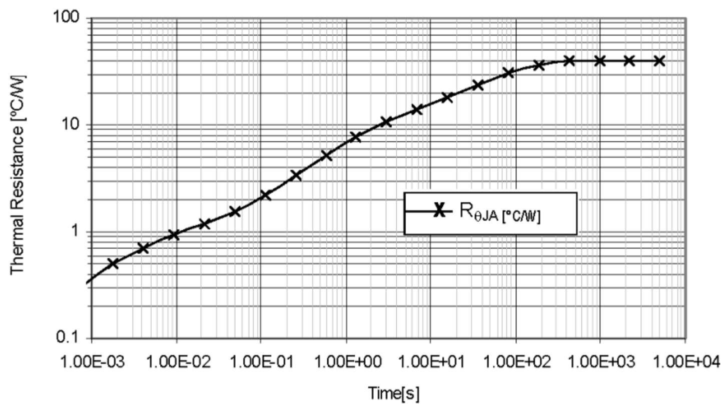


Figure 18. Transient Thermal Resistance $R_{\theta JA}$,
1.0 W Step response, Device on Thermal Test Board Area A = 600 (mm^2)

REVISION HISTORY

REVISION	DATE	DESCRIPTION
1.0	3/2006	• Updated formatting and technical content throughout entire document.
2.0	6/2007	• Updated formatting and technical content throughout entire document
3.0	10/2006	• Updated formatting and technical content throughout entire document
4.0	12/2006	• Updated formatting and technical content throughout entire document
5.0	2/2007	• Updated formatting and technical content throughout entire document
6.0	3/2007	• Changed Human Body Model, Charge Pump Voltage (CP Capacitor = 33 nF), No PWM and PWM = 20kHz, Slew Rate = Fast, Output Rise and Fall Time [27] • Added second paragraph to Positive Power Supply (VPWR) • Added “Low ESR” to 100µF on 33926 Typical Application Schematic
7.0	6/2007	• Changed status to Advance Information
8.0	4/2009	• Minor corrections and clarifications.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

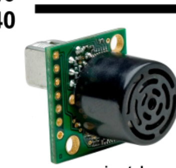
MC33926
Rev. 8.0
4/2009

freescale™
semiconductor

Ultrasonic Distance Sensor: MaxBotics XL-MaxSonar-AE4 (MB1340)

MB1240
MB1340

XL- MaxSonar®- EZ4™ (MB1240)
XL- MaxSonar®- AE4™ (MB1340)
Sonar Range Finder with High Power Output, Noise Rejection, Auto Calibration & Short-Range Narrow Detection Zone (Hardware gain of 250)



approximately actual size

The MB1240 & MB1340 have a new high power output along with real-time auto calibration for changing conditions (temperature, voltage or acoustic or electrical noise) that ensure you receive the most reliable (in air) ranging data for every reading taken. The MB1240 & MB1340 low power 3.3V – 5V operation provides very short to long-range detection and ranging, in a tiny and compact form factor. The MB1240 & MB1340 detect objects from 0-cm to 645-cm (25.1 feet) and provide sonar range information from 20-cm out to 645-cm with 1-cm resolution. Objects from 0-cm* to 20-cm typically range as 20-cm. (*Objects from 0-mm to 1-mm may not be detected.) The interface output formats included are pulse width output (MB1240), real-time analog voltage envelope (MB1340), analog voltage output, and serial digital output.*

Features

- High acoustic power output
- Real-time auto calibration and noise rejection for every ranging cycle
- Calibrated beam angle
- Continuously variable gain
- Object detection as close as 1-mm from the sensor
- 3.3V to 5V supply with very low average current draw
- Readings can occur up to every 100mS, (10-Hz rate)
- Free run operation can continually measure and output range information
- Triggered operation provides the range reading as desired
- All interfaces are active simultaneously
- Serial, 0 to Vcc, 9600Baud, 81N
- Analog, (Vcc/1024) / cm
- Pulse Width (MB1240)
  Real-time analog envelope (MB1340)
- Sensor operates at 42KHz

Benefits

- Acoustic and electrical noise resistance
- Reliable and stable range data
- Sensor dead zone virtually gone
- Low cost
- Quality controlled beam characteristics
- Very low power ranger, excellent for multiple sensor or battery based systems
- Ranging can be triggered externally or internally
- Sensor reports the range reading directly, frees up user processor
- Fast measurement cycle
- User can choose any of the sensor outputs
- No calibration requirement is perfect for when objects may be directly in front of the sensor during power up
- Small size allows for easy mounting

Applications and Uses

- UAV blimps, micro planes and some helicopters
- Bin level measurement
- Proximity zone detection
- People detection
- Robot ranging sensor
- Autonomous navigation
- Environments with acoustic and electrical noise
- Multi-sensor arrays
- Distance measuring
- Short range object detection
- Users who prefer to process the analog voltage envelope (MB1340)
- -40°C to +65°C operation (+85°C limited operation)

MB1240 & MB1340 Pin Out

Pin 1 - Leave open (or high) for serial output on the Pin 5 output. When Pin 1 is held low the Pin 5 output sends a pulse (instead of serial data), suitable for low noise chaining.

Pin 2 - MB1240 (PW) This pin outputs a pulse width representation of range. To calculate distance, use the scale factor of 58uS per cm.
MB1340 (AE) This pin outputs the analog voltage envelope of the acoustic wave form.

Pin 3 - (AN) This pin outputs analog voltage with a scaling factor of (Vcc/1024) per cm. A supply of 5V yields ~4.9mV/cm., and 3.3V yields ~3.2mV/cm. Hardware limits the maximum reported range on this output to ~700 cm at 5V and ~600 cm at 3.3V. The output is buffered and corresponds to the most recent range data.

Pin 4 - (RX) This pin is internally pulled high. The MB1240 & MB1340 will continually measure range and output if the pin is left unconnected or held high. If held low the MB1240 & MB1340 will stop ranging. Bring high 20uS or more for range reading.

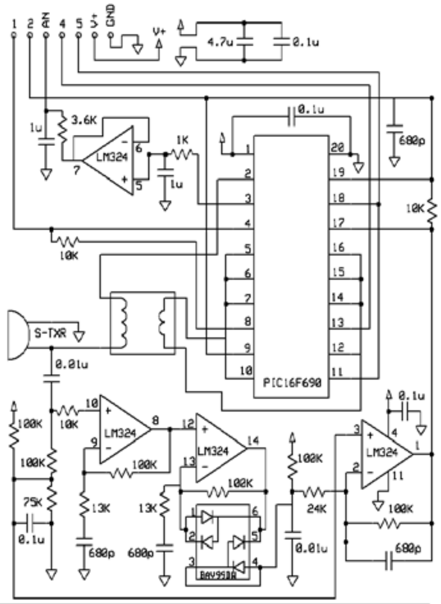
Pin 5 - (TX) When Pin 1 is open or held high, the Pin 5 output delivers asynchronous serial with an RS232 format, except voltages are 0-Vcc. The output is an ASCII capital "R", followed by three ASCII character digits representing the range in centimeters up to a maximum of 765, followed by a carriage return (ASCII 13). The baud rate is 9600, 8 bits, no parity, with one stop bit. Although the voltage of 0-Vcc is outside the RS232 standard, most RS232 devices have sufficient margin to read 0-Vcc serial data. If standard voltage level RS232 is desired, invert, and connect an RS232 converter such as a MAX232.
When Pin 1 is held low, the Pin 5 output sends a single pulse, suitable for low noise chaining (no serial data).

V+ Operates on 3.3V - 5V. The average (and peak) current draw for 3.3V operation is 2.1mA (50mA peak) and 5V operation is 3.4mA (100mA peak) respectively. Peak current is used during sonar pulse transmit.

GND Return for the DC power supply. GND (& V+) must be ripple and noise free for best operation.

MB1240 & MB1340 Circuit

The sensor functions using active components consisting of an LM324 and PIC16F690, together with a variety of other components. The schematic is shown to provide the user with detailed connection information.



MB1240 & MB1340 Real-time Operation

175mS after power-up, the XL-MaxSonar® is ready to begin ranging. If Pin-4 is left open or held high (20uS or greater), the sensor will take a range reading. The XL-MaxSonar® checks the Pin-4 at the end of every cycle. Range data can be acquired once every 99mS. Each 99mS period starts by Pin-4 being high or open, after which the XL-MaxSonar® calibrates and calculates for 20.5mS, and after which, thirteen 42KHz waves are sent.

Then for the MB1240, the pulse width (PW) Pin-2 is set high. When an object is detected the PW pin is set low. If no target is detected the PW pin will be held high for up to 44.4mS (i.e. 58uS * 765cm) (For the most accurate range data, use the PW output of the MB1240 product.)

For the MB1340 with analog envelop output, Pin-2 will show the real-time signal return information of the analog waveform.

For both parts, the remainder of the 99mS time (less 4.7mS) is spent adjusting the analog voltage to the correct level, (and allowing the high acoustic power to dissipate). During the last 4.7mS, the serial data is sent.

MB1240 & MB1340 Real-time Auto Calibration

Each time after the XL-MaxSonar® takes a range reading it calibrates itself. The sensor then uses this data to range objects. If the temperature, humidity, or applied voltage changes during sensor operation, the sensor will continue to function normally. The sensor does not apply compensation for the speed of sound change verses temperature to any range readings.

MB1240 & MB1340 Real-time Noise Rejection

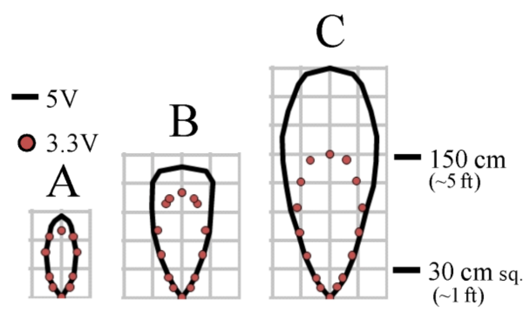
While the XL-MaxSonar® is designed to operate in the presence of noise, best operation is obtained when noise strength is low and desired signal strength is high. Hence, the user is encouraged to mount the sensor in such a way that minimizes outside acoustic noise pickup. In addition, keep the DC power to the sensor free of noise. This will let the sensor deal with noise issues outside of the users direct control (in general, the sensor will still function well even if these things are ignored). Users are encouraged to test the sensor in their application to verify usability.

For every ranging cycle, individual filtering for that specific cycle is applied. In general, noise from regularly occurring periodic noise sources such as motors, fans, vibration, etc., will not falsely be detected as an object. This holds true even if the periodic noise increases or decreases (such as might occur in engine throttling or an increase/decrease of wind movement over the sensor). Even so, it is possible for sharp non-periodic noise sources to cause false target detection. In addition, *(because of dynamic range and signal to noise physics,) as the noise level increases, at first only small targets might be missed, but if noise increases to very high levels, it is likely that even large targets will be missed.

*In high noise environments, if needed, use 5V power to keep acoustic signal power high. In addition, a high acoustic noise environment may use some of the dynamic range of the sensor. For applications with large targets, consider a part with ultra clutter rejection like the MB7092.

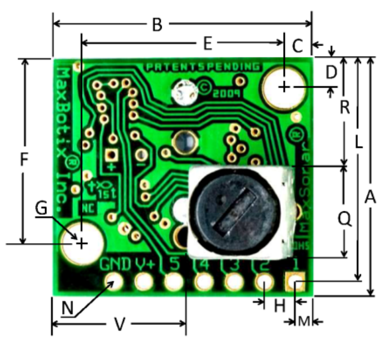
MB1240 & MB1340 Beam Characteristics

The MB1240 and MB1340 have a wide and long sensitive beam that offers excellent detection of objects and people. The MB1240 and MB1340 balances the detection of objects and people with minimal side-lobes. Sample results for measured beam patterns are shown to the right on a 30-cm grid. The detection pattern is shown for dowels of varying diameters that are place in front of the sensor;
(A) 6.1-mm (0.25-inch) diameter dowel,
(B) 2.54-cm (1-inch) diameter dowel,
(C) 8.89-cm (3.5-inch) diameter dowel,

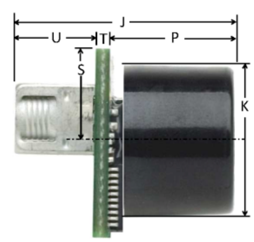


Beam characteristics are approximate

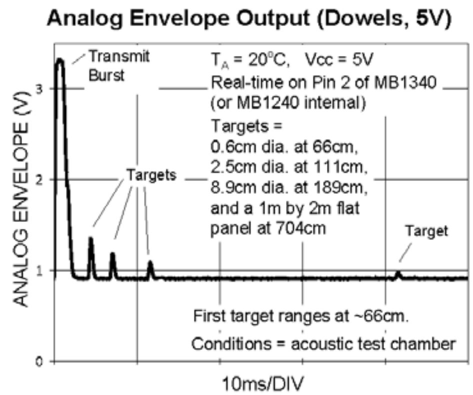
MB1240 & MB1340 Mechanical Dimensions

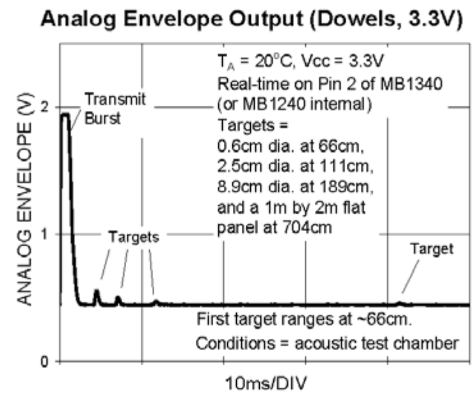


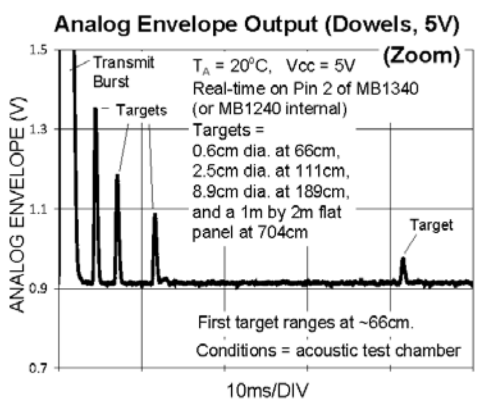
A	0.785"	19.9mm	L	0.735"	18.7mm
B	0.870"	22.1mm	M	0.065"	1.7mm
C	0.100"	2.54mm	N	0.038" dia.	1.0mm dia.
D	0.100"	2.54mm	P	0.537"	13.64mm
E	0.670"	17.0mm	Q	0.304"	7.72mm
F	0.610	15.5mm	R	0.351"	8.92mm
G	0.124" dia.	3.1mm dia.	S	0.413"	10.5mm
H	0.100"	2.54mm	T	0.063"	1.6mm
J	0.989"	25.11mm	U	0.368"	9.36mm
K	0.645"	16.4 mm	V	0.492"	12.5mm
values are nominal			Weight, 5.9 grams		

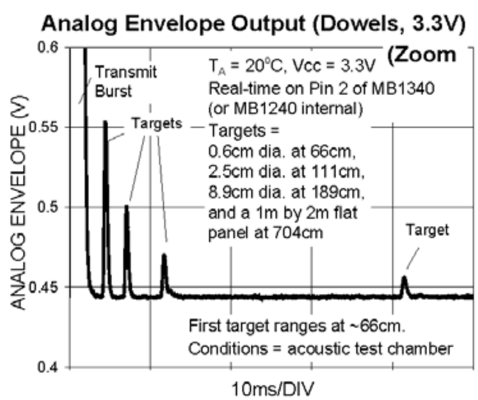


Typical Performance to Targets



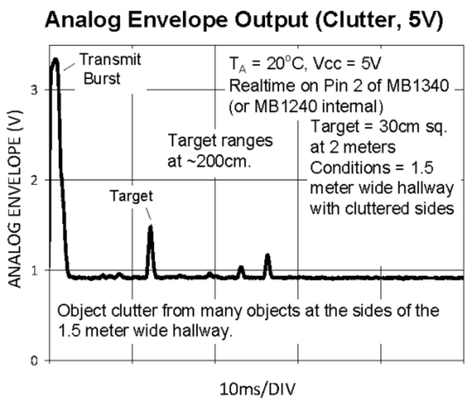


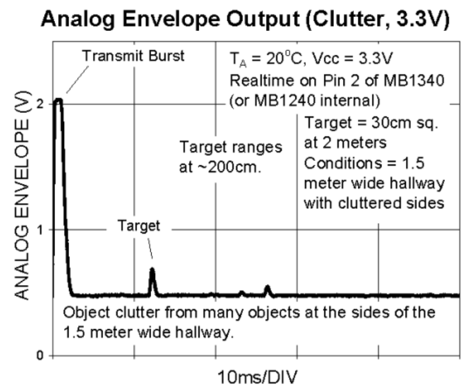




Email: info@maxbotix.com
Web: www.maxbotix.com

Typical Performance in Clutter





MCU: Microchip PIC18F46K22 (Partial)

MICROCHIP

PIC18(L)F2X/4XK22

28/40/44-Pin, Low-Power, High-Performance Microcontrollers with nanoWatt XLP Technology

High-Performance RISC CPU:

- C Compiler Optimized Architecture:
  - Optional extended instruction set designed to optimize re-entrant code
- Up to 1024 Bytes Data EEPROM
- Up to 64 Kbytes Linear Program Memory Addressing
- Up to 3896 Bytes Linear Data Memory Addressing
- Up to 16 MIPS Operation
- 16-bit Wide Instructions, 8-bit Wide Data Path
- Priority Levels for Interrupts
- 31-Level, Software Accessible Hardware Stack
- 8 x 8 Single-Cycle Hardware Multiplier

Flexible Oscillator Structure:

- Precision 16 MHz Internal Oscillator Block:
  - Factory calibrated to ± 1%
  - Selectable frequencies, 31 kHz to 16 MHz
  - 64 MHz performance available using PLL – no external components required
- Four Crystal modes up to 64 MHz
- Two External Clock modes up to 64 MHz
- 4X Phase Lock Loop (PLL)
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops
  - Two-Speed Oscillator Start-up

Analog Features:

- Analog-to-Digital Converter (ADC) module:
  - 10-bit resolution, up to 30 external channels
  - Auto-acquisition capability
  - Conversion available during Sleep
  - Fixed Voltage Reference (FVR) channel
  - Independent input multiplexing
- Analog Comparator module:
  - Two rail-to-rail analog comparators
  - Independent input multiplexing
- Digital-to-Analog Converter (DAC) module:
  - Fixed Voltage Reference (FVR) with 1.024V, 2.048V and 4.096V output levels
  - 5-bit rail-to-rail resistive DAC with positive and negative reference selection
- Charge Time Measurement Unit (CTMU) module:
  - Supports capacitive touch sensing for touch screens and capacitive switches

Extreme Low-Power Management with nanoWatt XLP:

- Sleep mode: 20 nA, typical
- Watchdog Timer: 300 nA, typical
- Timer1 Oscillator: 800 nA @ 32 kHz
- Peripheral Module Disable

Special Microcontroller Features:

- Full 5.5V Operation – PIC18FXXK22 devices
- 1.8V to 3.6V Operation – PIC18LFXXK22 devices
- Self-Programmable under Software Control
- High/Low-Voltage Detection (HLVD) module:
  - Programmable 16-Level
  - Interrupt on High/Low-Voltage Detection
- Programmable Brown-out Reset (BOR):
  - With software enable option
  - Configurable shutdown in Sleep
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- In-Circuit Serial Programming™ (ICSP™):
  - Single-Supply 3V
- In-Circuit Debug (ICD)

Peripheral Highlights:

- Up to 35 I/O Pins plus 1 Input-Only Pin:
  - High-Current Sink/Source 25 mA/25 mA
  - Three programmable external interrupts
  - Four programmable interrupt-on-change
  - Nine programmable weak pull-ups
  - Programmable slew rate
- SR Latch:
  - Multiple Set/Reset input options
- Two Capture/Compare/PWM (CCP) modules
- Three Enhanced CCP (ECCP) modules:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
  - PWM steering
- Two Master Synchronous Serial Port (MSSP) modules:
  - 3-wire SPI (supports all 4 modes)
  - I^2C™ Master and Slave modes with address mask

TABLE 1: PIC18(L)F2XK22 PIN SUMMARY

28-SSOP, SOIC 28-SPDIP	28-QFN, UQFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
2	27	RA0	AN0	C12IN0-										
3	28	RA1	AN1	C12IN1-										
4	1	RA2	AN2	C2IN+			V_{REF}- DACOUT							
5	2	RA3	AN3	C1IN+			V_{REF}+							
6	3	RA4		C1OUT		SRQ		CCP5			T0CKI			
7	4	RA5	AN4	C2OUT		SRNQ	HLVDIN			$\overline{SS}$1				
10	7	RA6												OSC2 CLKO
9	6	RA7												OSC1 CLKI
21	18	RB0	AN12			SRI		CCP4 FLT0		$\overline{SS}$2		INT0	Y	
22	19	RB1	AN10	C12IN3-				P1C		SCK2 SCL2		INT1	Y	
23	20	RB2	AN8		CTED1			P1B		SDI2 SDA2		INT2	Y	
24	21	RB3	AN9	C12IN2-	CTED2			CCP2 P2A(1)		SDO2			Y	
25	22	RB4	AN11					P1D			T5G	IOC	Y	
26	23	RB5	AN13					CCP3 P3A(4) P2B(3)			T1G T3CKI(2)	IOC	Y	
27	24	RB6							TX2/CK2			IOC	Y	PGC
28	25	RB7							RX2/DT2			IOC	Y	PGD
11	8	RC0						P2B(3)			SOSCO T1CKI T3CKI(2) T3G			
12	9	RC1						CCP2 P2A(1)			SOSCI			
13	10	RC2	AN14		CTPLS			CCP1 P1A			T5CKI			
14	11	RC3	AN15							SCK1 SCL1				
15	12	RC4	AN16							SDI1 SDA1				
16	13	RC5	AN17							SDO1				
17	14	RC6	AN18					CCP3 P3A(4)	TX1/CK1					
18	15	RC7	AN19					P3B	RX1/DT1					
1	26	RE3												$\overline{MCLR}$ V_{PP}
8	5													V_{SS}
19	16													V_{SS}
20	17													V_{DD}

Note 1: CCP2/P2A multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: P2B multiplexed in fuses.
4: CCP3/P3A multiplexed in fuses.

TABLE 2: PIC18(L)F4XK22 PIN SUMMARY

40-PDIP	40-UQFN	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
2	17	19	19	RA0	AN0	C12IN0-										
3	18	20	20	RA1	AN1	C12IN1-										
4	19	21	21	RA2	AN2	C2IN+			V_{REF-} DACOUT							
5	20	22	22	RA3	AN3	C1IN+			V_{REF+}							
6	21	23	23	RA4		C1OUT		SRQ					T0CKI			
7	22	24	24	RA5	AN4	C2OUT		SRNQ	HLVDIN			$\overline{SS1}$				
14	29	31	33	RA6												OSC2 CLKO
13	28	30	32	RA7												OSC1 CLKI
33	8	8	9	RB0	AN12			SRI		FLT0				INT0	Y	
34	9	9	10	RB1	AN10	C12IN3-								INT1	Y	
35	10	10	11	RB2	AN8		CTED1							INT2	Y	
36	11	11	12	RB3	AN9	C12IN2-	CTED2			CCP2 P2A$^{(1)}$					Y	
37	12	14	14	RB4	AN11								T5G	IOC	Y	
38	13	15	15	RB5	AN13					CCP3 P3A$^{(3)}$			T1G T3CKI$^{(2)}$	IOC	Y	
39	14	16	16	RB6										IOC	Y	PGC
40	15	17	17	RB7										IOC	Y	PGD
15	30	32	34	RC0						P2B$^{(4)}$			SOSCO T1CKI T3CKI$^{(2)}$ T3G			
16	31	35	35	RC1						CCP2$^{(1)}$ P2A			SOSCI			
17	32	36	36	RC2	AN14		CTPLS			CCP1 P1A			T5CKI			
18	33	37	37	RC3	AN15							SCK1 SCL1				
23	38	42	42	RC4	AN16							SDI1 SDA1				
24	39	43	43	RC5	AN17							SDO1				
25	40	44	44	RC6	AN18						TX1 CK1					
26	1	1	1	RC7	AN19						RX1 DT1					
19	34	38	38	RD0	AN20							SCK2 SCL2				
20	35	39	39	RD1	AN21					CCP4		SDI2 SDA2				
21	36	40	40	RD2	AN22					P2B$^{(4)}$						
22	37	41	41	RD3	AN23					P2C		$\overline{SS2}$				
27	2	2	2	RD4	AN24					P2D		SDO2				
28	3	3	3	RD5	AN25					P1B						
29	4	4	4	RD6	AN26					P1C	TX2 CK2					
30	5	5	5	RD7	AN27					P1D	RX2 DT2					
8	23	25	25	RE0	AN5					CCP3 P3A$^{(3)}$						

Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: CCP3/P3A multiplexed in fuses.
4: P2B multiplexed in fuses.

TABLE 2: PIC18(L)F4XK22 PIN SUMMARY (CONTINUED)

40-PDIP	40-UQFN	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
9	24	26	26	RE1	AN6					P3B						
10	25	27	27	RE2	AN7					CCP5						
1	16	18	18	RE3											Y	$\overline{MCLR}$ V_{PP}
11 32	7, 26	7 28	7,8 28, 29													V_{DD}
12 31	6, 27	6 29	6 30, 31													V_{SS}
—	—	12, 13 33, 34	13	NC												

Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: CCP3/P3A multiplexed in fuses.
4: P2B multiplexed in fuses.

Buzzer: Mallory AST1240MLTRQ

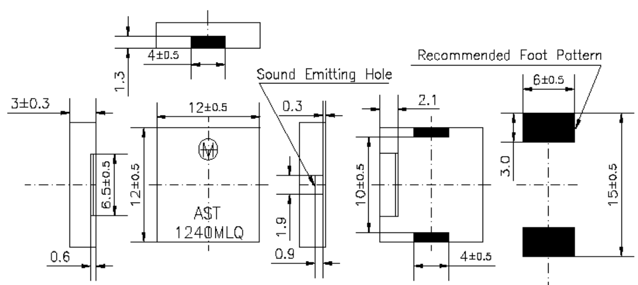
Mallory Sonalert Products Inc.	Part # **AST1240MLTRQ**
Sales Outline Drawing	Revision **A**

Specifications:

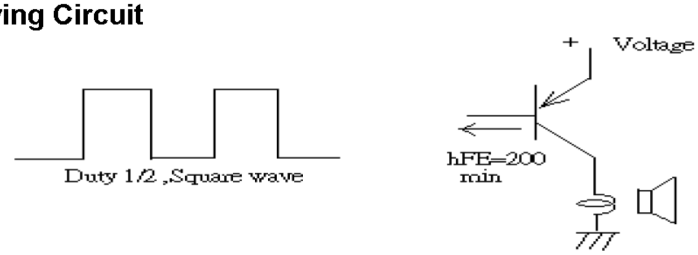
Resonant Frequency (Hz)	4000 ± 500
Operating Voltage (Vp-p)	25
Rated Voltage (Vp-p)	3
Current Consumption (mA/max)	5 at Rated Voltage
Sound Pressure Level Min. (dB)	75 at 10cm at Rated Voltage
Capacitance (pF)	16,000 ± 30% at 120 Hz
Operating Temperature (°C)	-20 ~ +70
Storage Temperature (°C)	-30 ~ +80
Housing	LCP Plastic Resin (Color: Black)
Type of Connection Terminals	SMD (Tin Plated Brass)
Condition by reflow soldering (°C)	245 ± 5°C / 1 min
Condition by hand soldering (°C)	350 ± 20°C / within 5 sec
Weight (Grams)	1
Options	For other options contact factory

PS: Vp-p = 1/2duty, square wave **RoHS Compliant**

Dimensions: (units: mm)



Recommend Driving Circuit



Packaging:

Parts are supplied on reels (1000 pcs per tape and reel).
Units do not come with wash label and are not appropriate for aqueous wash.

4411 South High School Road, Indianapolis, Indiana 46241 USA ● Phone (317) 612-1000 ● Fax (317) 612-1010 ● http://www.mallory-sonalert.com