Wingman

Projekt: Wingman
Projektrapport

Version 1.0

Jens Borgelin
Christoffer Martinsson
Jonas Nilsson
Martin Hjertstedt
Amir Bosnjakovic

Innehållsförteckning

1. Projektrapport Wingman

1.1 Sammanfattning

Projektet har varit relativt framgångsrikt även om det finns detaljer som kunde förbättras. Vissa kravspecifikationer har inte kunnat uppfyllas. Produktens skjutmekanism fungerar bättre än väntat och ger roboten ett slagkraftigare vapen att göra mål med. Detta leder förhoppningsvis till vinst. Ursprungstanken med utfällbara vingar lades ned pga. för svag konstruktion.

1.2 Introduktion

Projektet består av två delar, en mekanisk och en elektronisk del som tillsammans ska bilda en ”hockey robot”. Den mekaniska och elektroniska konstruktionen är framtagen med hjälp av gruppmedlemmarnas tidigare erfarenheter. All elektronik är tillverkade som moduler för att få bästa möjliga flexibilitet.

1.3 Elektronik

Elektroniken består av fyra delar, sändarenhet, mottagarenhet, drivenhet och spelarenhet *(se fig1)*. All datakommunikation bygger på PPM (Pulse Position Modulation) standard.

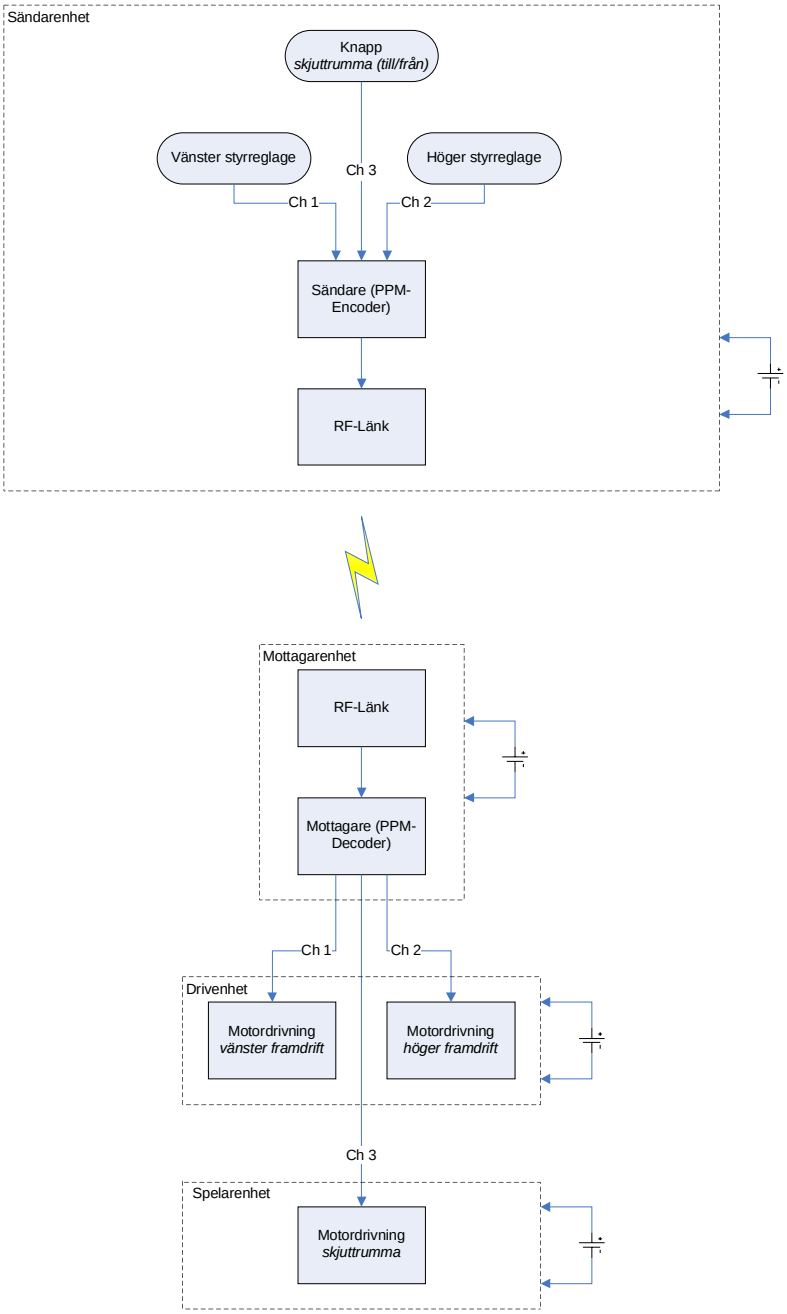


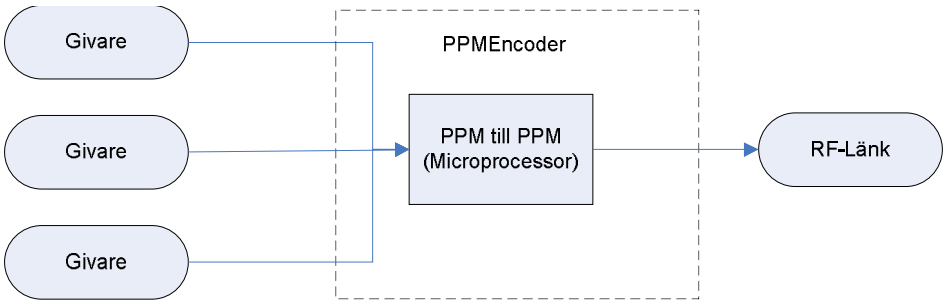
fig 1

Sändarenhet (PPMEncoder).

Spec:

- Låg kostnad
- Flexibel
- PPM-standard

Överblick:



PPMEncoder:
Microprocessorn ,som i detta fallet är en PIC18F2320, A/D omvandlar in-signalerna från varje kanals givare och genererar ett pulståg som består av en puls för varje kanal samt en sync-puls (se *fig 2*). Pulslängden för varje kanal är propotionell mot signalens spänningsnivå, 0V motsvarar en millisekund och 5V motsvarar två millisekunder. Se *bilaga 1* för schema och *bilaga 5* för programkod.

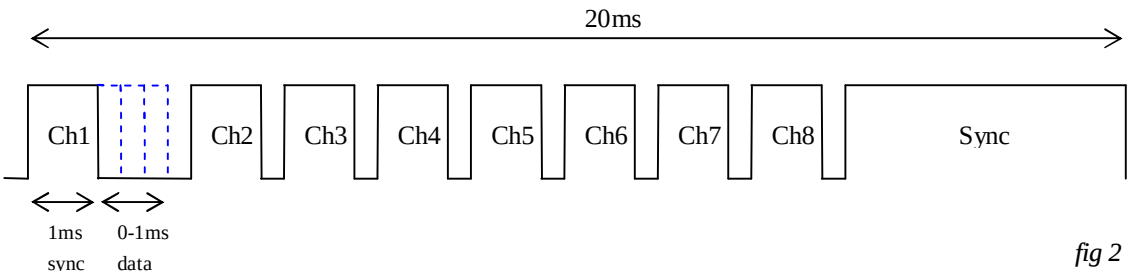
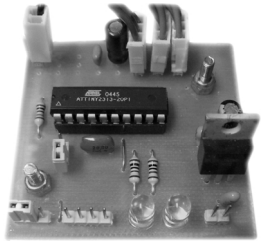


fig 2

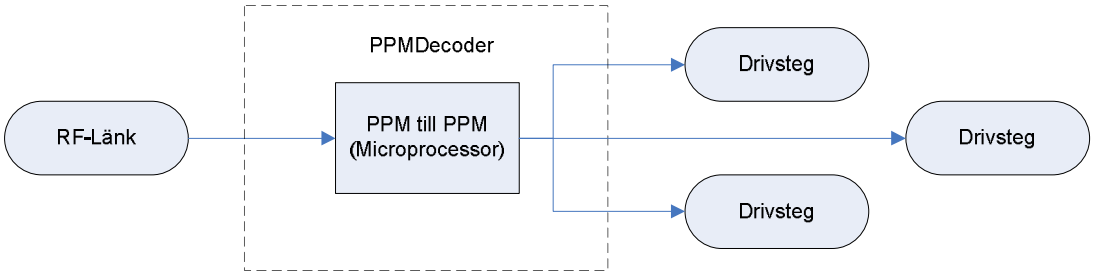
Mottagarenhet (PPMDecoder).



Spec:

- Låg kostnad
- Flexibel
- PPM-standard

Överblick:



PPMDecoder:
Genom att mäta de PPM-kodade pulslängder som decodern får via RF-Länken kan "bra" och "dåliga" signaler sorteras ut. Bra signaler är de som är mellen en och två millisekunder långa, allt annat är dåliga signaler. Därefter genereras en ny korrekt enkanals PPM signal baserat på den filtrerade in-signalen ut till respektive inkopplade drivsteg (se *fig 3*). Enheten är byggd för att klara av upp till åtta kanaler för bästa flexibilitet och processorn är därför valt till en Atmel Attiny2313 på grund av att den klarar av 20 MIPS vilket krävs för denna applikation. Programkod kan ses i *bilaga 6*. Schema och kretskortslayout kan ses i *bilaga 2*.

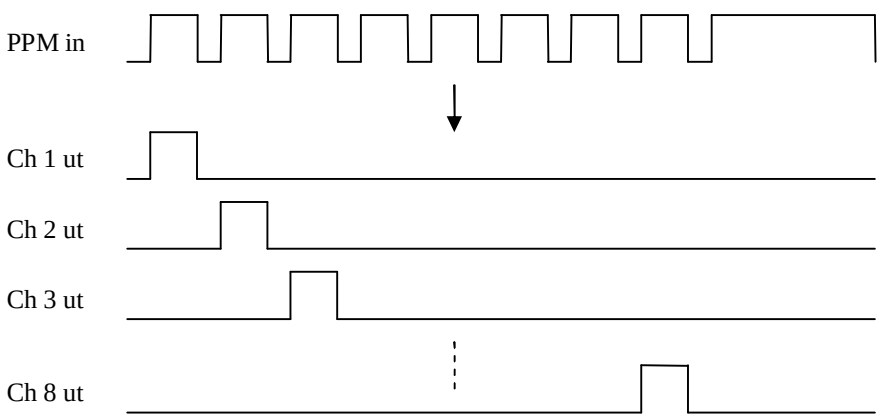
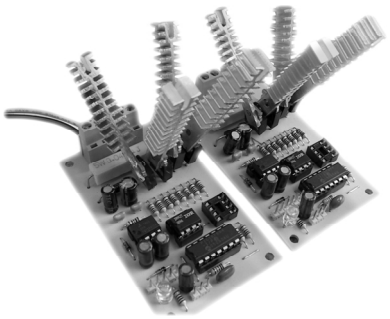


fig 3

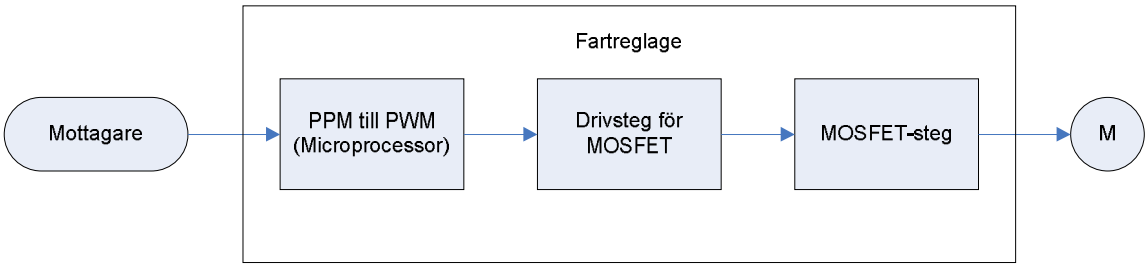
Drivenhet (MainMotorDriver).

Spec:



- 12V,10A
- Fram och back
- Effektiv
- Låg kostnad
- Flexibel
- Reglerbar hastighet

Överblick:



PPM till PWM

Signalen från mottagar-decodern kommer i form av en enkanals PPM signal. Denna består av en puls med en pulsbredd på mellan en och två millisekunder och upprepas varje tjugonde millisekund (se *fig4*).

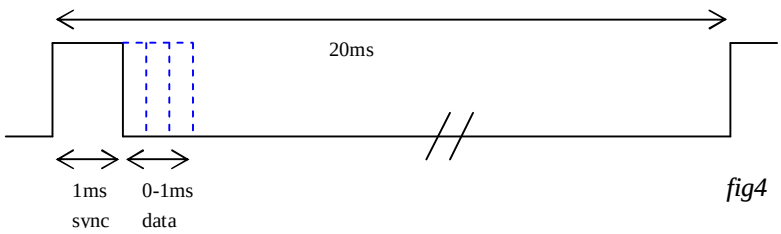


fig4

Genom att mäta pulsbredden på signalen fås det värde som bestämmer om motorn ska gå i fram eller back-drift och vilken hastighet motorn skall hålla.

En millisekund motsvarar 100% backdrift, en och en halv millisekund motsvarar 0% fram/backdrift och två millisekunder motsvarar 100% framdrift. Microprocessorn som utför mätningar och beräkningar är i detta fallet en PIC16F684. Denna har inbyggt hårvarustöd för H-bryggor vilket underlättar styrningen av MOSFET-setget. Se *bilaga 7* för programkod.

MOSFET-steget:

För att kunna driva motorn både i fram och back-riktning behöver man kunna polvända spänningen över motorn. Detta kan göras på olika sätt, t.ex med hjälp av ett relä som polvänder eller genom att använda två stycken batteri där det ena batteriet driver i framriktning och de andra i backriktning. Ett annat och i detta fallet mer lämpligt sätt är att använda en så kallad H-brygga (se *fig5*). H-bryggan kan byggas upp antingen av två stycken P-kanal MOSFET och två stycken N-kanal MOSFET eller med fyra stycken N-kanal MOSFET. Drivningen av transistorerna blir enklare när det övre paret är P-kanal men på grund av transistorns uppbyggnad har P-kanal högre inre resistans än N-kanal. För att få så lite effektförluster i transistorerna som möjligt väljs därför fyra N-kanal MOSFET för att bygga upp H-bryggan. Nackdelen med att sätta N-kanal som övre par är att gate-spänningen måste vara minst 17V för att det övre transistorparet ska driva.

Detta eftersom referensen till gate-ingången ligger på 12V och gs(gate to source)-spänningen måste vara runt 5V. Problemet löses genom att skapa en spänningsdubblare som då ger 24V för att driva de övre transistorparet.

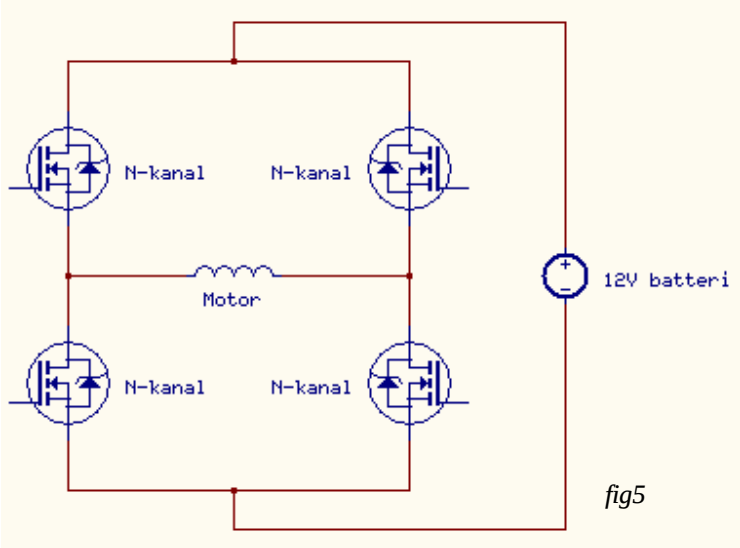


fig5

Drivning av H-bryggan:
Eftersom PIC-processorn endast ger 5V på utgångarna och våra övre MOSFET:ar behöver en gs-spännig på minst 17V krävs ett drivsteg mellan PIC-processorn och transistorerna. MOSFET-drivare IR4427 *(fig6)* används i detta fall för att driva de övre MOSFET paret med 24V från spännigsdublaren. För att förbättra effektiviteten kan med fördel det undre paret styras med 12V via en MOSFET-drivare men för att hålla kostnaderna ner styrs dessa direkt från PIC-processorns 5V:s utgångar.

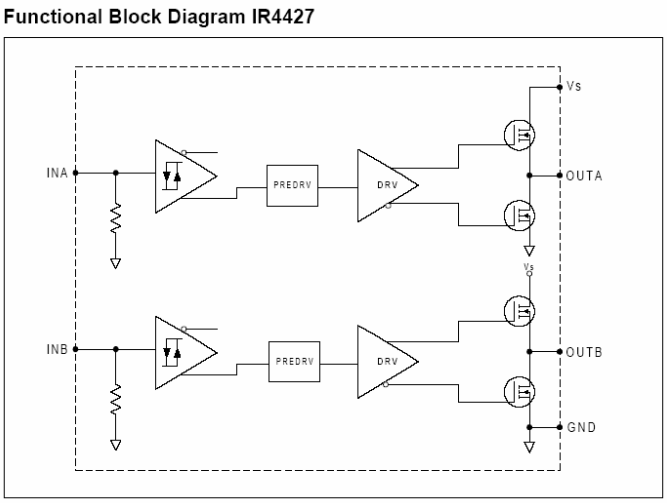
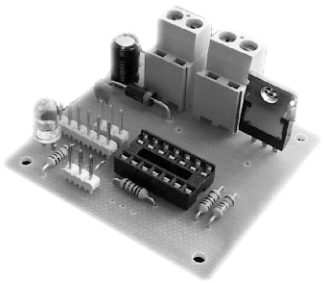


fig6

Se *Bilaga 3* för komplett kopplingsscema och kretskortslayout.

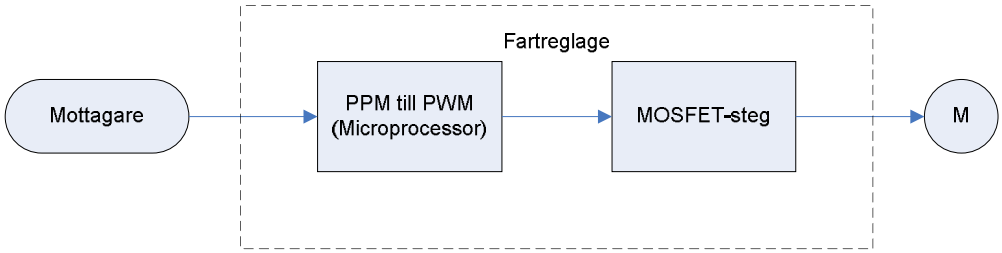
Spelarenhet (SecMotorDriver).

Spec:



- 12V,10A
- Endast framdrift
- Effektiv
- Låg kostnad
- Flexibel
- Reglerbar hastighet

Överblick:



PPM till PWM:
Signalen från mottagaren bearbetas på liknande sätt som Drivenheten. Skillnaden är att nu utnyttjas hela data-pulslängden som drivning frammåt. En millisekund motsvarar 0% framdrift och två millisekunder motsvarar 100% framdrift. Även här använd PIC16F684 som microprocessor. Programkod kan ses i *bilaga 8*.

MOSFET-steget:
Eftersom motorn skall drivas i endast en riktning behövs ingen H-brygga utan konstruktionen klarar sig med endast en N-kanal MOSFET. För att skydda transistorn kopplas även en frihjulsdiod in över motorn. För schema och kretskortlayout, se *bilaga 4*.

1.4 Genomförande

Projektet har genomförts relativt problemfritt och enligt uppgjord tidplan. Chassit konstruerades först enkelt av en metallstomme. Styrmotorernas upphängnings fäste skruvades fast i chassit. En metallåda för elektroniken är nedsänkt i chassit. En upphängnings-anordning till skjutmekanismen konstruerades. Slutligen tillverkades en ram och skal som täckte hela roboten för att skydda utsatta delar, så som elektronik och hjul. Se fig. 7. och fig. 8.

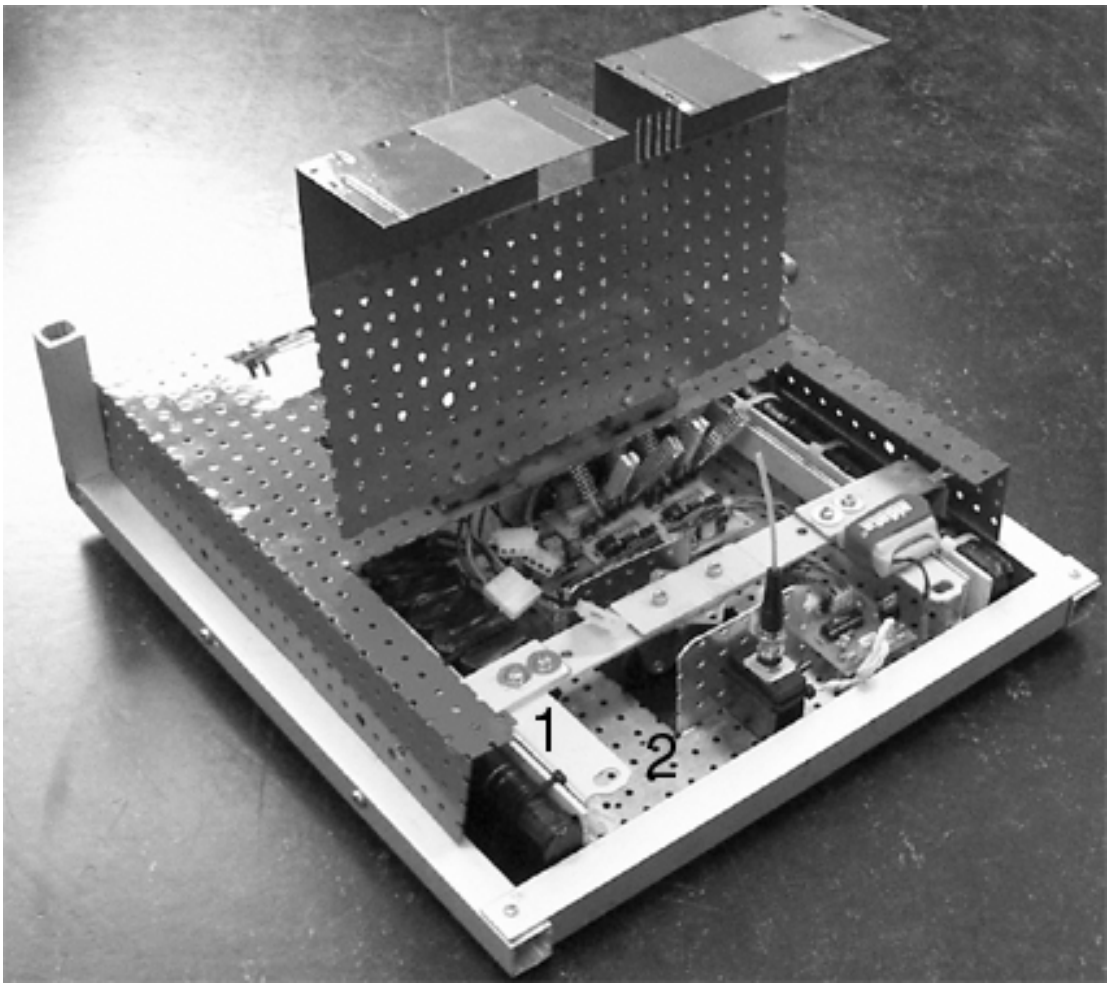


Fig. 7. Robotens baksida.
1. Chassi
2. Metallåda för elektroniken.

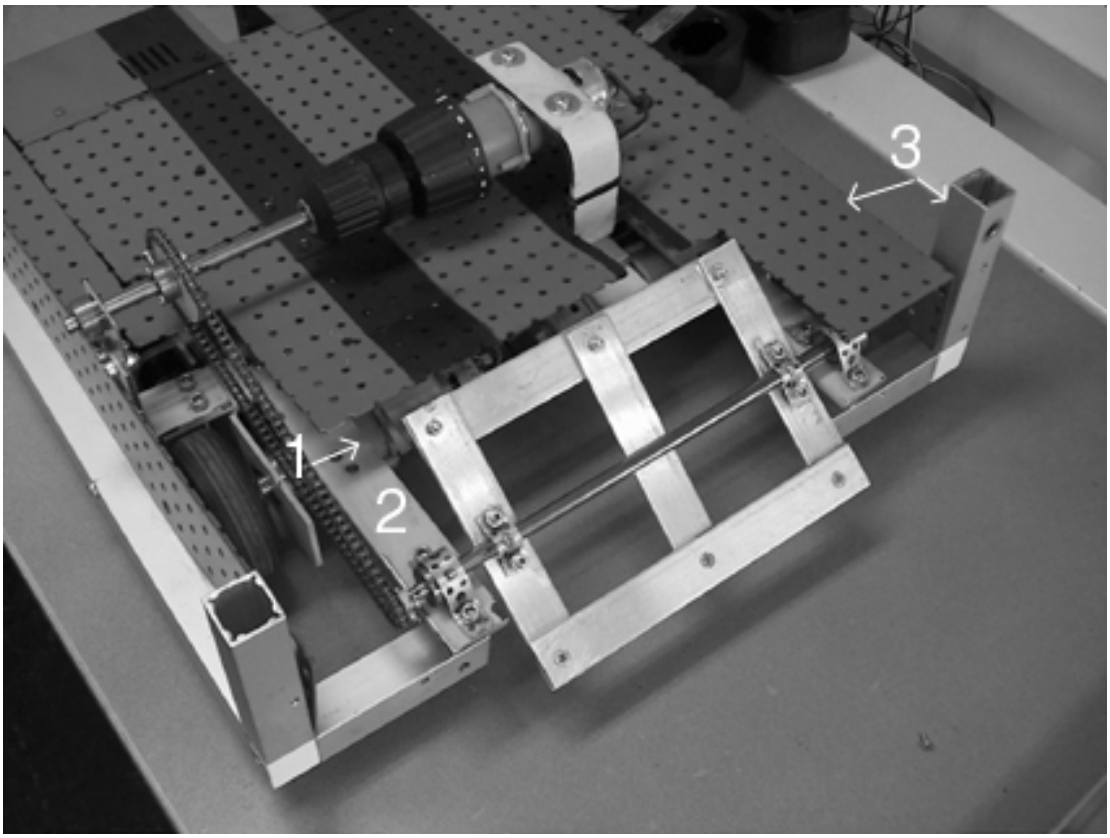


Fig. 8. Robotens framsida

1. Styrmotorernas upphängnings fäste.
2. Upphängnings anordning till skjutmekanismen.
3. Ram och skal

1.5 Jämförelse av kravspecifikaton och testresultat

1.5.1 Funktionalitetskrav 1

Prototypen ska kunna röra sig lätt och smidigt i alla riktningar på spelplanen.

Kravet uppfyllt med hjälp av ledat bakhjul och en motor till vardera framhjul.

1.5.2 Funktionalitetskrav 2

Elektroniken ska ha lite värmeförluster.

Kravet uppfyllt genom noggrann eftertanke på konstruktionen.

1.5.3 Funktionalitetskrav 3

Prototypen ska kunna styras manuellt.

Kravet uppfyllt med hjälp av en trådlös kontroll.

1.5.4 Funktionalitetskrav 4

Elektroniken ska bestå av steglös drivning.

Kravet uppfyllt med hjälp av en H-brygga.

1.5.5 Funktionalitetskrav 5

Prototypen skall ha bra fäste i hjulen.

Kravet uppfyllt, då däcken ger bra fäste på klinkers.

1.5.6 Funktionalitetskrav 6

Elektroniken skall vara störningstålig.

Kravet uppfyllt med hjälp av filter och skärmning.

1.5.7 Funktionalitetskrav 7

Prototypen skall vara hållbar och kunna ta emot en krock med motståndarnas robot utan att ta skada.

Kravet uppfyllt genom en mycket hållbar stomme.

1.5.8 Funktionalitetskrav 8

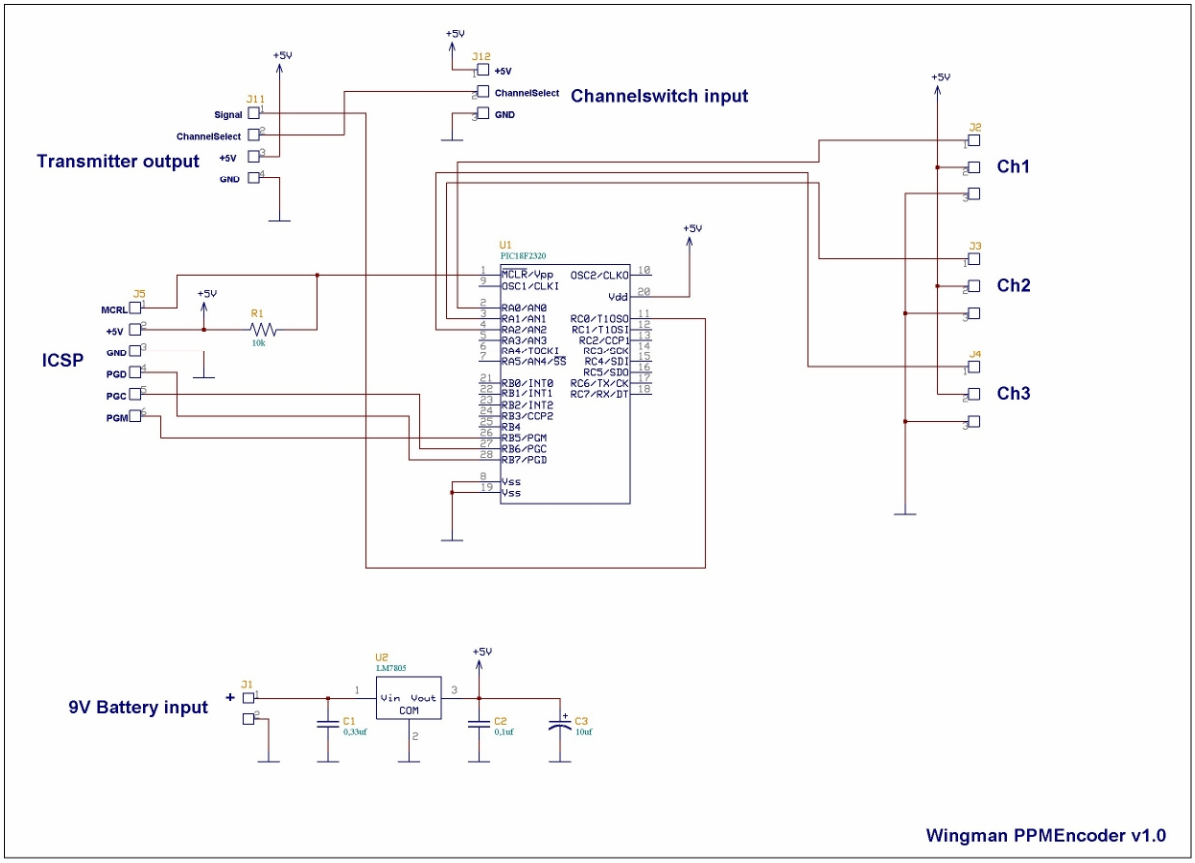
Mekaniken skall kunna skjuta ett skott från mitten av bilen till en punkt, minst 2 meter framför roboten.

Kravet uppfyllt, den skjuter mycket längre.

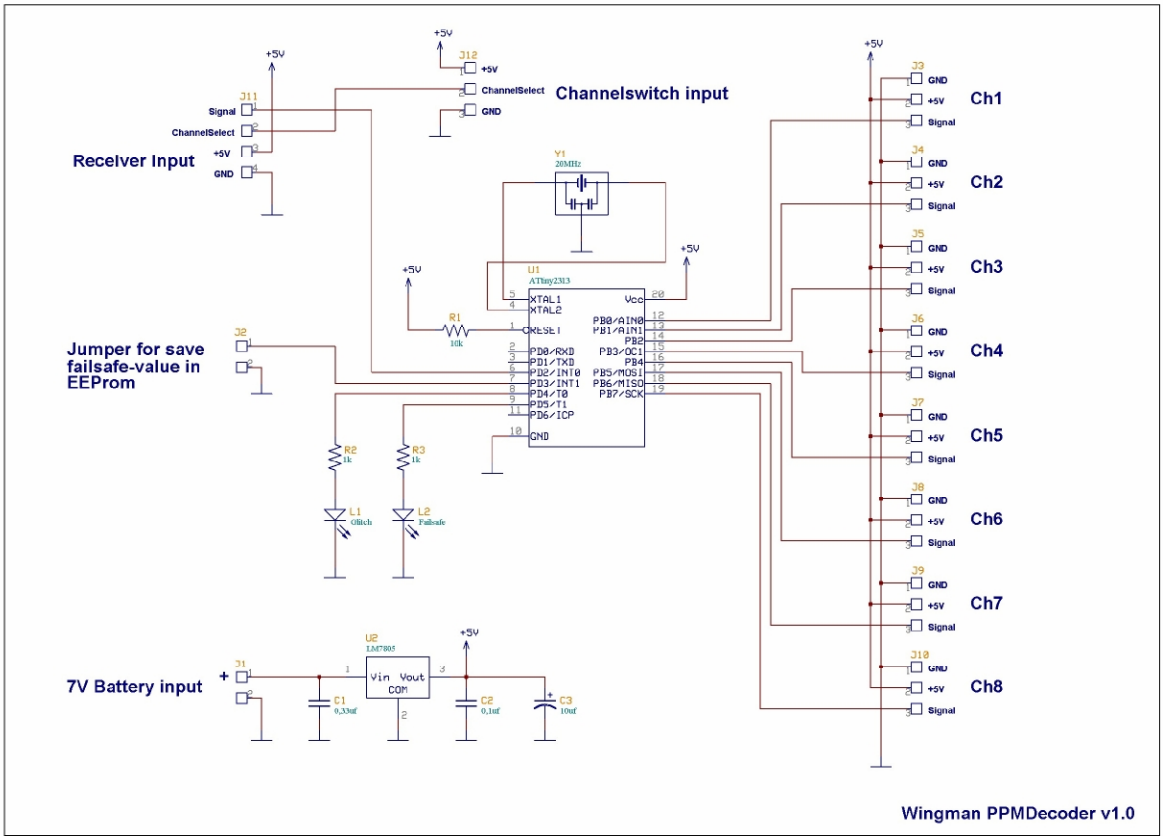
1.6 Slutsats

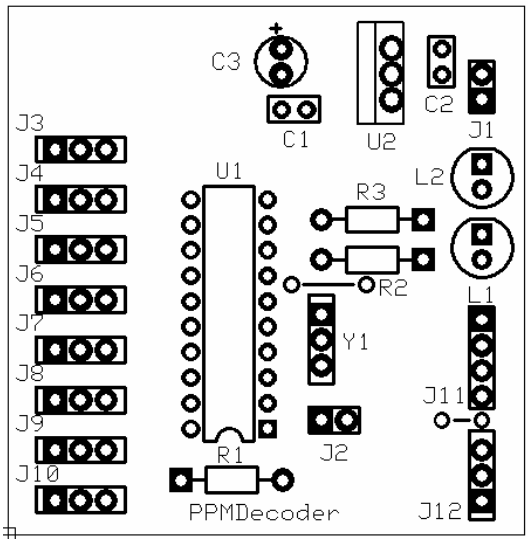
Projektet har utförts med stor framgång. Problem som uppstått har lösts, genom att problemen analyserats ordentligt för att på så sätt komma fram till bästa möjliga lösning. För att komma fram till skjutmekanismen har ett otaligt antal modeller och sidoprojekt uppkommit. Trots medelmåttig hårdvara och verktyg till vårt förfogande fick vi en modell som var relativt pålitlig.

Bilaga 1 – Schema över Sändarenhet (PPMEncoder)

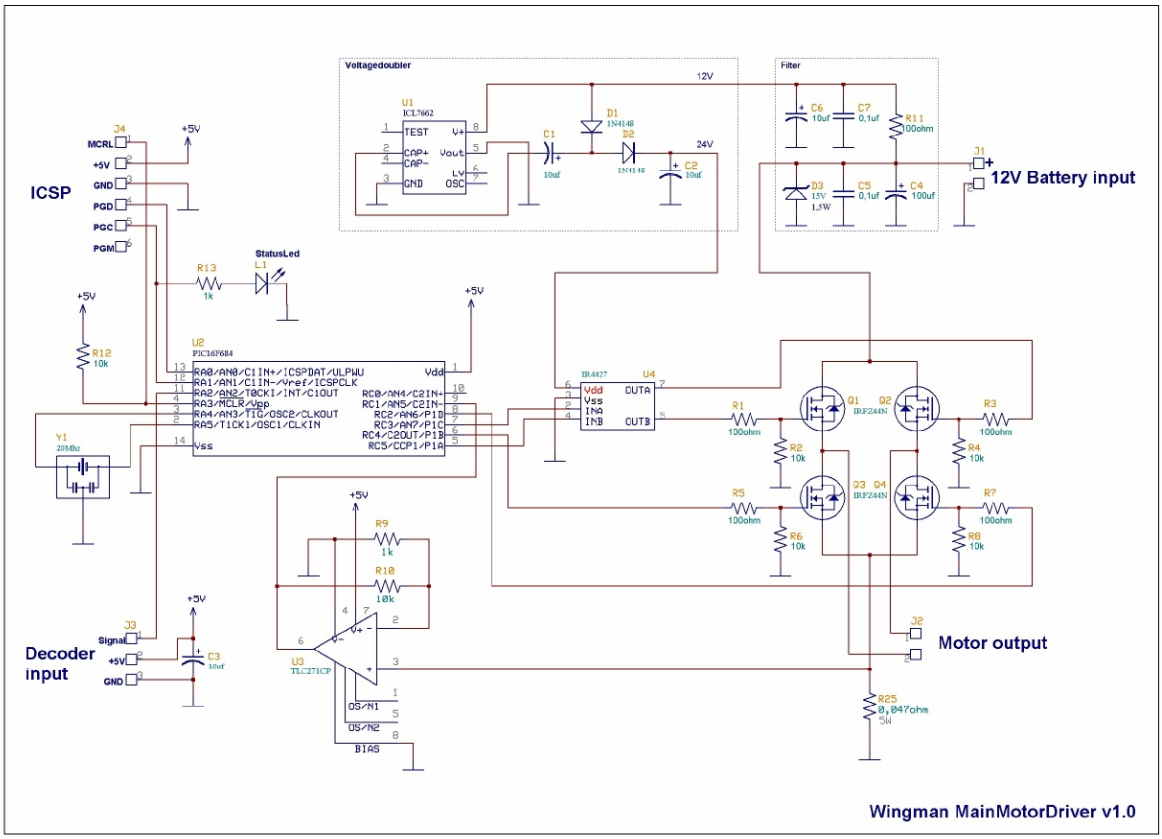


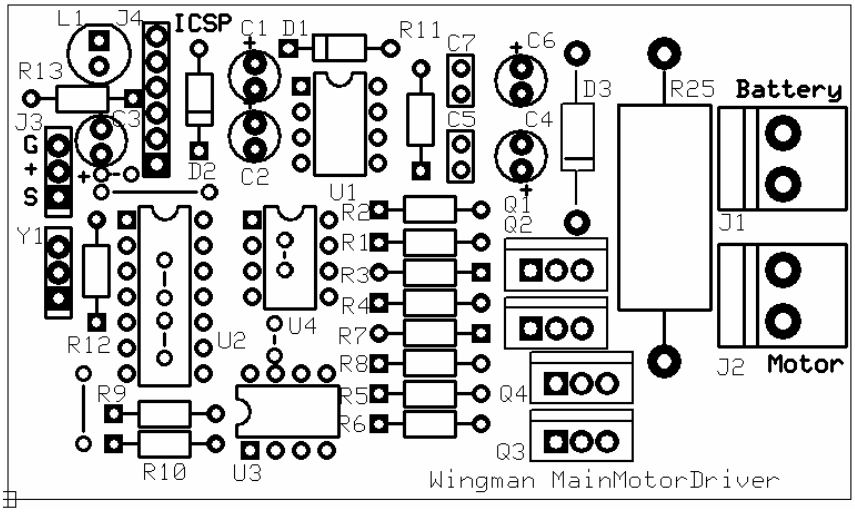
Bilaga 2 – Schema över Mottagarenhet (PPMDecoder)



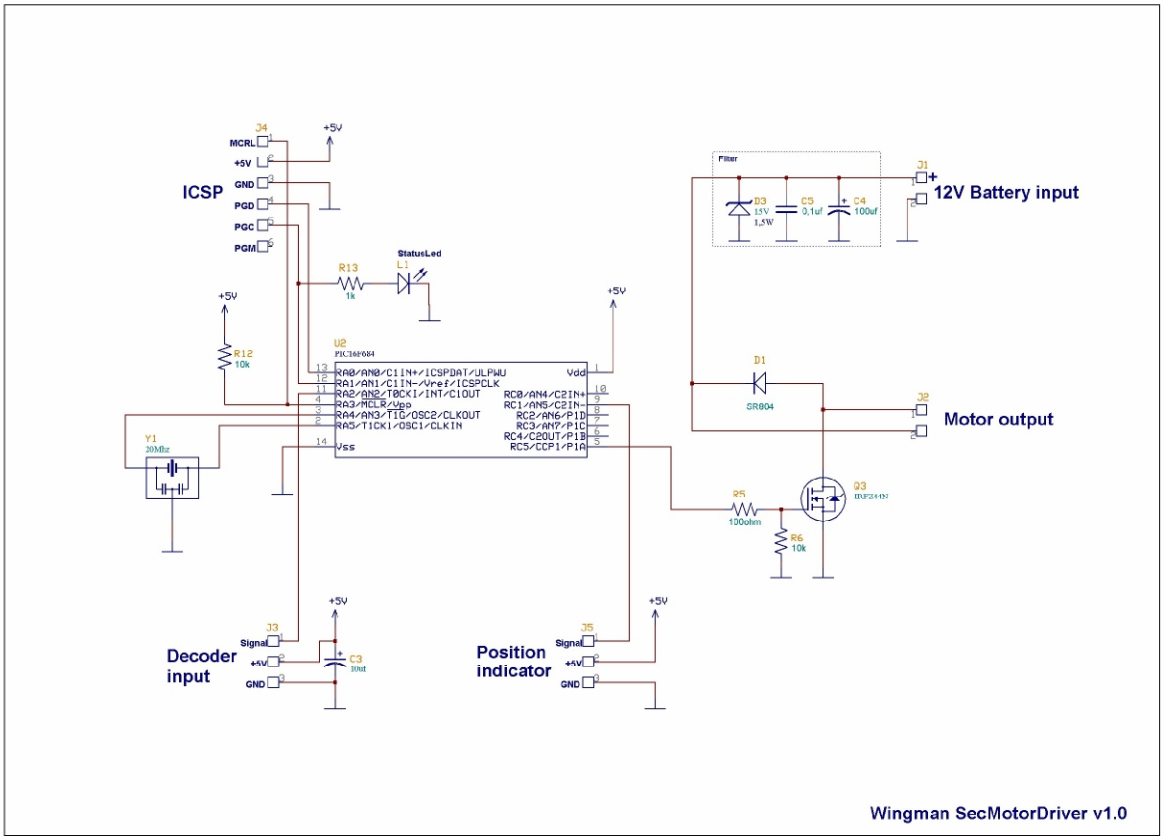


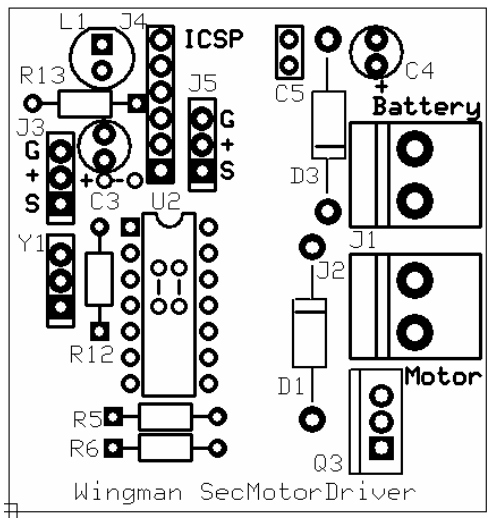
Bilaga 3 – Schema över Drivenhet (MainMotorDriver)





Bilaga 4 – Schema över Spelarenhet (SecMotorDriver)





Bilaga 5 – Programkod för Sändarenhet (PPMEncoder)

```
;*****************************************************************************************************
;
; Wingman PPMEncoder                                copyright 2005                          v1.0 (2005-04-22)
;
; Name: Christoffer Martinsson
; E-mail: cm@wsm.se
;
;
;*****************************************************************************************************

include P18F2320.INC

; Temp-registers
W_TEMP          EQU             0x20                    ; TempRegister for interrupt-routine
STATUS_TEMP     EQU             0x21                    ; TempRegister for interrupt-routine
TempH           EQU             0x25
TempL           EQU             0x26
Temp            EQU             0x27
DelayTemp       EQU             0x28
GetADValue      EQU             0x29

TXout           EQU             RC0

; Time-definitions
minPulseValue           =.4000;.20000
maxPulseValue           =.11000;.40000
ms1Value                =.5100;20000
ms05Value               =.2550;.10000

;*****************************************************************************************************
; Reset- and Interrupt-vectors
;*****************************************************************************************************
        ORG     0x0000
                        GOTO    Main                    ;Reset-vector
                        ORG             0x0008
                        GOTO    Interrupt               ;Interrupt-vector
                        ORG             0x0018
                        GOTO    Interrupt               ;Interrupt-vector
                        ORG             0x0028
;*****************************************************************************************************
; Macro
;*****************************************************************************************************
;-----------------------------------------------------------------------------------------------------
; StoreWregInTemp
;
; Description: Store Status and working-register in temp-registers
;
; Input: -
; ChangedReg: W_TEMP,STATUS_TEMP
; UsedReg: W,STATUS,W_TEMP,STATUS_TEMP
```

```
;-------------------------------------------------------------------------------------------------
StoreWregInTemp     MACRO
                    MOVWF   W_TEMP                  ;Copy W to TEMP register
                    SWAPF   STATUS,W                ;Swap status to be saved into W
                    CLRF    STATUS                  ;bank 0, regardless of current bank, Clears
IRP,RP1,RP0
                    MOVWF   STATUS_TEMP             ;Save status to bank zero STATUS_TEMP register
                    ENDM
;-------------------------------------------------------------------------------------------------
; RestoreWregFromTemp
;
; Description: Restore Status and working-register from temp-registers
;
; Input: -
; ChangedReg: W,STATUS
; UsedReg: W,STATUS,W_TEMP,STATUS_TEMP
;-------------------------------------------------------------------------------------------------
RestoreWregFromTemp MACRO
                    SWAPF   STATUS_TEMP,W           ;Swap STATUS_TEMP register into W (sets bank to original
state)
                    MOVWF   STATUS                  ;Move W into Status register
                    SWAPF   W_TEMP,F                ;Swap W_TEMP
                    SWAPF   W_TEMP,W                ;Swap W_TEMP into W
                    ENDM
;=================================================================================================
; Sub-routines
;=================================================================================================
;-------------------------------------------------------------------------------------------------
; delayMs
;
; Description: Delay for 1ms if W = 256.
;

; Input: W
; ChangedReg: DelayTemp
; UsedReg: W, DelayTemp
;-------------------------------------------------------------------------------------------------
delayMs             MOVWF   DelayTemp
                    INCF    DelayTemp,F
delayMs1  ;NOP
                    DECFSZ  DelayTemp,F
                    GOTO    delayMs1
                    RETURN
;-------------------------------------------------------------------------------------------------
; getAD
;
; Description: Peform A/D conversion and read data from AD.
;

; Input: W
; ChangedReg: DelayTemp
; UsedReg: W, DelayTemp
```

```
;-------------------------------------------------------------------------------------------------
getAD           CLRF    GetADValue
                BSF     ADCON0,1
getAD1          BTFSC   ADCON0,1
                GOTO    getAD1
                MOVF    ADRESH,W
                MOVWF   GetADValue
                RETURN


;=================================================================================================
; Interrupt routines
;=================================================================================================
Interrupt   StoreWregInTemp
;-------------------------------------------------------------------------------------------------
; Timer1_Int
;
; Description: Increment pulseIn counter-registers
;-------------------------------------------------------------------------------------------------
Timer0_IntBTFSS         PIR1,TMR1IF
                        GOTO            Int_Return
                        MOVLW           0xB1
                        MOVWF           TMR1H
                        MOVLW           0xDF
                        MOVWF           TMR1L

                        BCF             LATC,0

                        MOVLW           0x40
                        CALL            delayMs

                        ;Ch 1
                        MOVLW           0x03
                        ANDWF           ADCON0,F
                        CALL            getAD
                        BSF             LATC,0
                        MOVLW           0xFF
                        CALL            delayMs
                        MOVLW           0x08
                        CALL            delayMs
                        MOVF            GetADValue,W
                        CALL            delayMs
                        BCF             LATC,0

                        MOVLW           0x40
                        CALL            delayMs

                        ;Ch 2
                        MOVLW           0x03
                        ANDWF           ADCON0,F
                        MOVLW           0x04
                        ADDWF           ADCON0,F
```

```
CALL    getAD
BSF     LATC,0
MOVLW   0xFF
CALL    delayMs
MOVLW   0x08
CALL    delayMs
MOVF    GetADValue,W
CALL    delayMs
BCF     LATC,0

MOVLW   0x40
CALL    delayMs

;Ch 3
MOVLW   0x03
ANDWF   ADCON0,F
MOVLW   0x08
ADDWF   ADCON0,F
CALL    getAD
BSF     LATC,0
MOVLW   0xFF
CALL    delayMs

MOVF    GetADValue,W
CALL    delayMs
BCF     LATC,0

MOVLW   0x40
CALL    delayMs

;Ch 4
BSF     LATC,0
MOVLW   0xFF
CALL    delayMs

BCF     LATC,0

MOVLW   0x40
CALL    delayMs

;Ch 5
BSF     LATC,0
MOVLW   0xFF
CALL    delayMs

BCF     LATC,0

MOVLW   0x40
CALL    delayMs

;Ch 6
BSF     LATC,0
```

```
                        MOVLW       0xFF
                        CALL        delayMs

                        BCF         LATC,0

                        MOVLW       0x40
                        CALL        delayMs

                        ;Ch 7
                        BSF         LATC,0
                        MOVLW       0xFF
                        CALL        delayMs

                        BCF         LATC,0

                        MOVLW       0x40
                        CALL        delayMs

                        BSF         LATC,0

                        BCF         PIR1,TMR1IF
;-------------------------------------------------------------------------------------------------
Int_ReturnRestoreWregFromTemp
                        RETFIE

;=================================================================================================
; Main
;=================================================================================================
Main                    BCF         INTCON,GIE
                        MOVLW       b'01100010'         ; Internal OSC set to 4MHz
                        MOVWF       OSCCON

                        MOVLW       b'00001010'
                        MOVWF       ADCON1              ; A/D

                        MOVLW       b'00000000'
                        MOVWF       ADCON0

                        MOVLW       b'00100001'
                        MOVWF       ADCON2

                        MOVLW       b'00000001'
                        MOVWF       ADCON0


                        BCF         TRISC,TXout         ; Set "TXout"-pin to output
                        CLRF        LATC                ; Clear port c

                        CLRF        T0CON               ; Timer0 disble

                        MOVLW       b'10000101'         ; Timer1 enable and set interrupt to 20ms
                        MOVWF       T1CON
```

```
                        MOVLW       0xB1
                        MOVWF       TMR1H
                        MOVLW       0xDF
                        MOVWF       TMR1L

                        ;MOVLW      b'01000100'
                        ;MOVWF      T3CON

                        BSF         PIE1,TMR1IE         ; Timer1 interrupt enable
                        BSF         INTCON,PEIE
                        BSF         INTCON,GIE          ; Global interrup enable

;-----------------------------------------------------------------------------------------------
; MainLoop
;-----------------------------------------------------------------------------------------------
Mainloop
                        GOTO        Mainloop
                        END
;===============================================================================================
; END
;===============================================================================================
```

Bilaga 6 – Programkod för Mottagarenhet (PPMDecoder)

```
;***********************************************************************************************
;
; Wingman PPMDecoder                    copyright 2005                    v1.0 (2005-05-14)
;
; Name: Christoffer Martinsson
; E-mail: cm@wsm.se
;
; 8ch PPM decoder for RC-Receivers.
; Based on Atmel AT90S2313 or ATTiny2313 with 10 or 20Mhz crystal.
;
; This DSP is build on the idea of having one inputDecoding and one outputEncoding working simultanius.
; This to manage decode seven (or more) channels and not to be bound to use the "spare"-time at the end
; of the pulseFrame.
;
; PORTB,PB0-PB7 assign to output ch 1 to ch 7 (ch 8 reserved)
; PORTD,PD2 assign to input PPM-signal from RX
; PORTD,PD3 assign to StoreButton/Jumper
; PORTD,PD4 assign to red LED
; PORTD,PD5 assign to yellow LED
;
; The inputsignal is decoded into two separate buffert locations in the SRAM. The output-routine work
; with the buffert that the input-routine NOT currently working with. After a "good" frame is decoded,
; the buffertPointer is switched. This to ensure that a "good" pulseframe allways is available if a error
; should occur.
;
; * Up to 8 channels
; * 100step/ms resolution at 10Mhz, 200step/ms resolution at 20Mhz
; * Allways a "good" output-signal
; * Output is "Freezed" if error occurs
; * Failsafe sets predefined values if error not recovered in (recomended) 2sec (max 8sec/10Mhz, 4sec/20Mhz)
; * Failsafe-value easily changed by pressing a button
; * PulseValue filtered to a smooth average-value
; * Error/Glitch indicator
;
;***********************************************************************************************
; Processor-type
.EQU            AT90S2313       = 1
;.EQU           ATTiny2313      = 1

.NOLIST                                         ; Disable listfile generation
.ifdef AT90S2313
.INCLUDE  "2313def.inc"
.else
.INCLUDE  "tn2313def.inc"
.endif
.LIST                                           ; Reenable listfile generation

; General definitions
; Adjust this numbers to make it work propely with your application
```

```
.EQU    XTAL            =10             ; Used crystal (Mhz) (min 10Mhz)
.EQU    RecoveryNr      =1              ; Number of "good" pulseFrames to recover from error-state
.EQU    NrOfChannels    =7              ; Numbers of channels (max 8)
.EQU    FailSafeTime    =2              ; Time before enter failsafe (sec) (max 8sec/10Mhz, 4sec/20Mhz)

; 8bit register
.DEF    temp            =r16    ; Temporary register
.DEF    nextSRAMAddress =r12    ; Register for storing secondary SRAMAddress (buffert2)
.DEF    debounceFilter  =r15    ; Button-debounce register
.DEF    timeout         =r17    ; Timeout counter
.DEF    chAddressEE     =r18    ; EEPROM channelAddress
.DEF    chAddressIn     =r19    ; Input channelAddress
.DEF    chAddressOut    =r20    ; Output channelAddress
.DEF    chAddressFS     =r21    ; Failsafe channelAddress
.DEF    pulseError      =r22    ; PulseErrorCounter
.DEF    pulseFlag       =r23    ; Flag register

; "16bit" register
.DEF    frameCalcL      =r13    ; Register for calculate FrameTime
.DEF    frameCalcH      =r14
.DEF    tempL           =r24    ; Temporary register
.DEF    tempH           =r25
.DEF    pulseInL        =r26    ; InputPulseCounter
.DEF    pulseInH        =r27
.DEF    pulseOutL       =r28    ; OutputPulseCounter
.DEF    pulseOutH       =r29

; pulseFlag bit-definitions
.EQU    SYNC            =0      ; Sync found-Flag
.EQU    LEDredActive    =1      ; LEDred Active-Flag
.EQU    LEDyellowActive =2      ; LEDyellow Active-Flag
.EQU    BuffertToUse    =3      ; Buffert-Flag (0=buffert1, 1=buffert2)

; Output bit-definitions
.EQU    PPMSignal       =2      ; PD2 Input for PPM-signal
.EQU    StoreButton     =3      ; PD3 Input for StoreButton
.EQU    LEDred          =4      ; PD4 Output for LEDred
.EQU    LEDyellow       =5      ; PD5 Output for LEDyellow

; Memory definitions
.EQU    SRAMaddress1    =0x0060 ; Startlocation for buffert1 in SRAM
.EQU    SRAMAddress2    =0x0074 ; Startlocation for buffert2 in SRAM
.EQU    EEPROMaddress   =0x0000 ; Startlocation for failsafe-storage in EEPROM

; Time definitions (timer0)
.EQU    FSTimeoutTime   =(2*XTAL*FailSafeTime)      ; 1sec*FailSafeTime

; Time definitions (timer1)
.EQU    MinPulseTime    =(95*(XTAL/10))     ; ~0,95ms
.EQU    MaxPulseTime    =(220*(XTAL/10))    ; ~2,2ms
.EQU    MinSyncTime     =(400*(XTAL/10))    ; ~4ms
.EQU    MaxSyncTime     =(1400*(XTAL/10))   ; ~14ms
```

```
.EQU            FrameTime               =(1950*(XTAL/10))   ; ~20ms

.CSEG                                                       ; CODE segment
.ORG            0
;***************************************************************************************************
; Reset- and Interrupt-vectors
;***************************************************************************************************
                        RJMP    Main                ; Reset Handler
                        RJMP    Ex_Int0             ; External Interrupt0 Handler
                        RETI                        ; External Interrupt1 Handler
                        RETI                        ; Timer1 Capture Handler
                        RJMP    Timer_Int1          ; Timer1 CompareA Handler
                        RETI                        ; Timer1 Overflow Handler
                        RJMP    Timer_Int0          ; Timer0 Overflow Handler
                        RETI                        ; USART0 RX Complete Handler
                        RETI                        ; USART0,UDR Empty Handler
                        RETI                        ; USART0 TX Complete Handler
                        RETI                        ; Analog Comparator Handler

.ifdef ATTiny2313

                        RETI                        ; Pin Change Interrupt
                        RETI                        ; Timer1 Compare B Handler
                        RETI                        ; Timer0 Compare A Handler
                        RETI                        ; Timer0 Compare B Handler
                        RETI                        ; USI Start Handler
                        RETI                        ; USI Overflow Handler
                        RETI                        ; EEPROM Ready Handler
                        RETI                        ; Watchdog Overflow Handler
.endif
;***************************************************************************************************
; Macro
;***************************************************************************************************
;---------------------------------------------------------------------------------------------------
; Load16Temp
;
; Description: Load 16bit tempReg with 16bit @0-value
;
; Input: @0,@1
; ChangedReg: tempH,tempL
; UsedReg: tempH,tempL
;---------------------------------------------------------------------------------------------------
.MACRO          Load16Temp
                LDI                             tempH,HIGH(@0)
                LDI                             tempL,LOW(@0)
.ENDMACRO
;---------------------------------------------------------------------------------------------------
; Compare16Temp
;
; Description: Compare 16bit tempReg with 16bit @0,@1-reg
;
; Input: @0,@1
```

```
; ChangedReg: Carry
; UsedReg: tempH,tempL
;-----------------------------------------------------------------------------------------------
.MACRO          Compare16Temp
                CP                      tempL,@1
                CPC                     tempH,@0
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; Clear16Reg
;
; Description: Clear 16bit @0,@1-register
;
; Input: @0,@1
; ChangedReg: @0,@1
; UsedReg: -
;-----------------------------------------------------------------------------------------------
.MACRO          Clear16Reg
                CLR                     @0
                CLR                     @1
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; Add16Reg
;
; Description: Add 16bit @2,@3 to 16bit @0,@1
;
; Input: @0,@1
; ChangedReg: @0,@1,Carry
; UsedReg: -
;-----------------------------------------------------------------------------------------------
.MACRO          Add16Reg
                ADD                     @1,@3
                ADC                     @0,@2
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; Sub16Reg
;
; Description: Subtract 16bit @2,@3 from 16bit @0,@1
;
; Input: @0,@1
; ChangedReg: @0,@1,Carry
; UsedReg: -
;-----------------------------------------------------------------------------------------------
.MACRO          Sub16Reg
                SUB                     @1,@3
                SBC                     @0,@2
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; DivByTwo16Reg
;
; Description: Divide 16bit @0,@1 by two
;
; Input: @0,@1
```

```
; ChangedReg: @0,@1
; UsedReg: -
;-----------------------------------------------------------------------------------------------
.MACRO          DivByTwo16Reg
                CLC
                ROR             @0
                ROR             @1
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; Write16SRAM
;
; Description: Write data in 16bit @1,@2-reg to SRAM at position @0
;
;               (BuffertToUse-Flag: 0=buffert1, 1=buffert2)
;
; Input: @0,@1,@2
; ChangedReg: -
; UsedReg: ZL,pulseFlag
;-----------------------------------------------------------------------------------------------
.MACRO          Write16SRAM
                LDI             ZL,LOW(SRAMaddress1)
                SBRC            pulseFlag,BuffertToUse
                ADD             ZL,nextSRAMAddress
                ADD             ZL,@0
                ST              Z+,@1
                ST              Z,@2
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; Read16SRAM
;
; Description: Read data in SRAM at position @0 to 16bir @1,@2-reg
;
;       (BuffertToUse-Flag: 0=buffert1, 1=buffert2)
;
; Input: @0,@1,@2
; ChangedReg: @1,@2
; UsedReg: ZL,pulseFlag
;-----------------------------------------------------------------------------------------------
.MACRO          Read16SRAM
                LDI             ZL,LOW(SRAMaddress1)
                SBRC            pulseFlag,BuffertToUse
                ADD             ZL,nextSRAMAddress
                ADD             ZL,@0
                LD              @1,Z+
                LD              @2,Z
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; Write16EEPROM
;
; Description: Write data in 16bit @1,@2-reg to EEPROM at position @0
;
; Input: @0,@1,@2
```

```
; ChangedReg: -
; UsedReg: temp,EECR,EEARL,EEDR
;-----------------------------------------------------------------------------------------------------
.MACRO          Write16EEPROM
MWEEWait1:      SBIC            EECR,1
                RJMP            MWEEWait1
                LDI             temp,LOW(EEPROMaddress)
                ADD             temp,@0
                OUT             EEARL,temp
                OUT             EEDR,@1
                SBI             EECR,EEMWE
                SBI             EECR,EEWE

MWEEWait2:      SBIC            EECR,1
                RJMP            MWEEWait2
                LDI             temp,LOW(EEPROMaddress)
                ADD             temp,@0
                INC             temp
                OUT             EEARL,temp
                OUT             EEDR,@2
                SBI             EECR,EEMWE
                SBI             EECR,EEWE
.ENDMACRO
;-----------------------------------------------------------------------------------------------------
; Read16EEPROM
;
; Description: Read data in EEPROM at position @0 to 16bir @1,@2-reg
;
; Input: @0,@1,@2
; ChangedReg: @1,@2
; UsedReg: temp,EECR,EEARL,EEDR
;-----------------------------------------------------------------------------------------------------
.MACRO          Read16EEPROM
MREEWait1:      SBIC            EECR,1
                RJMP            MREEWait1
                LDI             temp,LOW(EEPROMaddress)
                ADD             temp,@0
                OUT             EEARL,temp
                SBI             EECR,EERE
                IN              @1,EEDR

MREEWait2:      SBIC            EECR,1
                RJMP            MREEWait2
                LDI             temp,LOW(EEPROMaddress)
                ADD             temp,@0
                INC             temp
                OUT             EEARL,temp
                SBI             EECR,EERE
                IN              @2,EEDR
.ENDMACRO
;-----------------------------------------------------------------------------------------------------
; StartTimer0
```

```
;
; Description: Start timer0. Prescale set to CK/1024
;
; Input: -
; ChangedReg: -
; UsedReg: temp,TCCR0
;-----------------------------------------------------------------------------------------------
.MACRO          StartTimer0
                LDI             temp,0x05
                OUT             TCCR0,temp              ; Prescale set to CK/1024
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; StopTimer0
;
; Description: Stop timer0
;
; Input: -
; ChangedReg: -
; UsedReg: temp,TCCR0
;-----------------------------------------------------------------------------------------------
.MACRO          StopTimer0
                CLR             temp
                OUT             TCCR0,temp
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; StartTimer1
;
; Description: Star timer1. Prescale set to CK
;
; Input: -
; ChangedReg: -
; UsedReg: temp,OCR1AH,OCR1AL,TCCR1B
;-----------------------------------------------------------------------------------------------
.MACRO          StartTimer1
                LDI             temp,HIGH(100)
                OUT             OCR1AH,temp
                LDI             temp,LOW(100)
                OUT             OCR1AL,temp
                LDI             temp,0x09
                OUT             TCCR1B,temp             ; Prescale set to CK
.ENDMACRO
;-----------------------------------------------------------------------------------------------
; StopTimer1
;
; Description: Stop timer1
;
; Input: -
; ChangedReg: -
; UsedReg: temp,TCCR1B
;-----------------------------------------------------------------------------------------------
.MACRO          StopTimer1
                CLR             temp
```

```
                OUT             TCCR1B,temp
.ENDMACRO
;***************************************************************************
; StoreTempOnStack
;
; Description: Store Status and temp-register on Stack
;
; Input: -
; ChangedReg: -
; UsedReg: temp,tempH,tempL,SREG
;***************************************************************************
.MACRO          StoreTempOnStack
                PUSH            tempH
                PUSH            tempL
                PUSH            temp
                IN              temp,SREG
                PUSH            temp
.ENDMACRO
;***************************************************************************
; RestoreTempFromStack
;
; Description: Restore Status and temp-register from Stack
;
; Input: -
; ChangedReg: temp,tempH,tempL,SREG
; UsedReg: temp,tempH,tempL,SREG
;***************************************************************************
.MACRO          RestoreTempFromStack
                POP             temp
                OUT             SREG,temp
                POP             temp
                POP             tempL
                POP             tempH
.ENDMACRO
;***************************************************************************
; SwitchBuffert
;
; Description: Switches buffertPointer in SRAM (BuffertToUse-Flag: 0=buffert1, 1=buffert2)
;
; Input: -
; ChangedReg: pulseFlag
; UsedReg: temp,pulseFlag
;***************************************************************************
.MACRO          SwitchBuffert
                LDI             temp,(1<<BuffertToUse)
                EOR             pulseFlag,temp          ; Switch Buffert in SRAM
.ENDMACRO
;***************************************************************************************************
; Interrupt routines
;***************************************************************************************************
;***************************************************************************
; Ex_Int0
```

```
;
; Description: Measure incoming pulse-value, filter it and then store it in SRAM.
;-----------------------------------------------------------------------------------------------------------
Ex_Int0:    StoreTempOnStack

ChkSync:  SBRC          pulseFlag,SYNC                          ; Check if sync found
          RJMP          ChkError
          Load16Temp    MinSyncTime
          Compare16Temp pulseInH,pulseInL                       ; Check if pulseIn > MinSyncTime
          BRCC          Error;                                  ; If not then jump to Error
          Load16Temp    MaxSyncTime
          Compare16Temp pulseInH,pulseInL                       ; Check if pulseIn < MaxSyncTime
          BRCS          Error                                   ; If not then jump to Error
          SBR           pulseFlag,(1<<SYNC)                     ; SYNC-Flag set
          CLR           chAddressIn                             ; Clear chAddressIn

ChkSyncExit:  Clear16Reg      pulseInH,pulseInL                 ; Clear pulseIn register
          RestoreTempFromStack
          RETI

ChkError:     Load16Temp      MinPulseTime
          Compare16Temp pulseInH,pulseInL                       ; Check if pulseIn > MinPulseTime
          BRCC          Error                                   ; If not then jump to Error
          Load16Temp    MaxPulseTime
          Compare16Temp pulseInH,pulseInL                       ; Check if pulseIn < MaxPulseTime
          BRCS          Error                                   ; If not then jump to Error

          CPI           pulseError,0                            ; Check if pulseError == 0
          BREQ          NoError                                 ; If true then jump to NoError
          DEC           pulseError                              ; Decrement pulseError

          INC           chAddressIn                             ; Increment pulseOut-address twice
          INC           chAddressIn
          CPI           chAddressIn,(NrOfChannels*2)            ; Check if chAddress == (NrOfChannels*2)
          BRNE          ChkErrExit                              ; If not then jump to ChkErrExit
          CBR           pulseFlag,(1<<SYNC)                     ; Clear SYNC-Flag
          Clear16Reg    frameCalcH,frameCalcL                   ; Clear FrameCalculation register

ChkErrExit:   Clear16Reg      pulseInH,pulseInL                 ; Clear pulseIn register
          RestoreTempFromStack
          RETI

Error:        CBR             pulseFlag,(1<<SYNC)               ; SYNC-Flagg cleared
          SBR           pulseFlag,(1<<LEDredActive)             ; LEDActive-Flag set
          LDI           pulseError,(RecoveryNr*NrOfChannels); pulseError = (RecoveryNr*NrOfChannels)

ErrorExit:    Clear16Reg      pulseInH,pulseInL                 ; Clear pulseIn register
          RestoreTempFromStack
          RETI

NoError:      CLR             timeout                           ; Clear timeout register
          CLR           chAddressFS                             ; Clear chAddressEE
```

```
                        StartTimer0                                     ; Start timer0

                        Read16SRAM      chAddressIn,tempH,tempL         ; Copy SRAM to temp
                        Add16Reg        pulseInH,pulseInL,tempH,tempL   ; Add past value to current value
                        DivByTwo16Reg   pulseInH,pulseInL               ; Divide value by two

                        Write16SRAM     chAddressIn,pulseInH,pulseInL   ; Copy temp to SRAM
                        Add16Reg        frameCalcH,frameCalcL,pulseInH,pulseInL; Add curren value to frameCalc register

                        INC             chAddressIn                     ; Increment pulseOut-address twice
                        INC             chAddressIn
                        CPI             chAddressIn,(NrOfChannels*2)    ; Check if chAddress == (NrOfChannels*2)
                        BRNE            NoErrorExit                     ; If not then jump to NoErrorExit
                        CBR             pulseFlag,(1<<SYNC)             ; Clear SYNC-Flag

                        Load16Temp      FrameTime
                        Sub16Reg        tempH,tempL,frameCalcH,frameCalcL       ; Calculate frame-time
                        Write16SRAM     chAddressIn,tempH,tempL         ; Write frame-time to SRAM

                        SwitchBuffert                                   ; Switch Buffert in SRAM
                        CBR             pulseFlag,(1<<LEDredActive)     ; LEDredActive-Flag cleared
                        CBR             pulseFlag,(1<<LEDyellowActive); LEDywllowActive-Flag cleared

                        Clear16Reg      frameCalcH,frameCalcL           ; Clear frameCalc register

NoErrorExit:            Clear16Reg      pulseInH,pulseInL               ; Clear pulseIn register
                        RestoreTempFromStack
                        RETI
;-----------------------------------------------------------------------------------------------------
; Timer_Int1
;
; Description: Generate output-pulses from SRAM to PORTB. Increment pulseIn and pulseOut counter-registers
;-----------------------------------------------------------------------------------------------------
Timer_Int1:             StoreTempOnStack

                        ADIW            pulseInL,1                      ; Increment pulseIn register
                        ADIW            pulseOutL,1                     ; Increment pulseOut register
                        CPI             chAddressOut,0                  ; Check if chAddressOut == 0
                        BRNE            ReadOPulse                      ; If true then jump to ReadOPulse
                        LDI             temp,0x01
                        OUT             PORTB,temp                      ; PORTB = 0x01

ReadOPulse:             SwitchBuffert                                   ; Switch Buffert in SRAM
                        Read16SRAM      chAddressOut,tempH,tempL        ; Copy SRAM to temp
                        SwitchBuffert                                   ; Switch Buffert in SRAM
                        Compare16Temp   pulseOutH,pulseOutL             ; Check if pulseOut < temp
                        BRCS            NextOPulse                      ; If not then jump to NextOPulse
                        RestoreTempFromStack
                        RETI

NextOPulse:             CPI             chAddressOut,(NrOfChannels*2); Check if chAddressOut ==
                        BREQ            ResetOPulse                     ; (NrOfChannels*2)
```

```
                    IN      temp,PORTB          ; If true then jump to ResetOPulse
                    LSL     temp
                    OUT     PORTB,temp          ; Roll PORTB left
                    INC     chAddressOut        ; Increment chAddressOut twice
                    INC     chAddressOut
                    Clear16Reg  pulseOutH,pulseOutL ; Clear pulseOut register
                    RestoreTempFromStack
                    RETI

ResetOPulse:        CLR     chAddressOut        ; Clear chAddressOut register
                    Clear16Reg  pulseOutH,pulseOutL ; Clear pulseOut register
                    RestoreTempFromStack
                    RETI
;-------------------------------------------------------------------------------------------------
; Timer_Int0
;
; Description: Failsafe "watchdog". Restore pulse-value from EEPROM to SRAM if timeout occurs.
;-------------------------------------------------------------------------------------------------
Timer_Int0:         StoreTempOnStack
                    CPI     timeout,FStimeoutTime   ; Check if timeout == FailsafeTime
                    BREQ    CopyPulseES         ; If true then jump to CopyPulseES
                    INC     timeout             ; Increment timeout
                    RestoreTempFromStack
                    RETI

CopyPulseES:        SBR     pulseFlag,(1<<LEDyellowActive); LEDActive-Flgg set
                    Read16EEPROM    chAddressFS,tempH,tempL ; Copy EEPROM to temp
                    Write16SRAM     chAddressFS,tempH,tempL ; Copy temp to SRAM
                    SwitchBuffert               ; Switch Buffert in SRAM
                    Write16SRAM     chAddressFS,tempH,tempL ; Copy temp to SRAM
                    SwitchBuffert               ; Switch Buffert in SRAM
                    INC     chAddressFS         ; Increment chAddressFS twice
                    INC     chAddressFS
                    CPI     chAddressFS,((NrOfChannels*2)+2); Check if chAddressFS == ((NrOfChannels*2)+2)
                    BRNE    NextPulseES         ; If not true then jump to NextPulseES
                    StopTimer0                  ; Stop timer0

NextPulseES:        RestoreTempFromStack
                    RETI

;=================================================================================================
; Main
;=================================================================================================
Main:               LDI     temp,LOW(RAMEND)
                    OUT     SPL,temp            ; Set StackPointeradress

                    CLI                         ; Disable GlobalInterrupt

                    ;Init I/O
                    LDI     temp,0xFF
                    OUT     DDRB,temp           ; All output on PORTB
                    SBI     DDRD,LEDred         ; All input except PIN"LEDred"
```

```
                SBI     DDRD,LEDyellow          ; and PIN"LEDyellow"
                SBI     PORTD,PPMSignal         ; Enable pullup on INT0 input
                SBI     PORTD,StoreButton       ; Enable pullup on StoreButton input

                LDI     temp,(1<<INT0)
                OUT     GIMSK,temp              ; Enable external interrupt 0
                LDI     temp,(1<<ISC01)
                OUT     MCUCR,temp              ; Trigg on falling edge
                LDI     temp,(1<<OCIE1A)|(1<<TOIE0)
                OUT     TIMSK,temp              ; Enable internal timer0
                                                ; and timer1 interrupt
                                                ; Reset register

                CLR     chAddressOut
                CLR     chAddressIn
                CLR     chAddressFS
                CLR     chAddressEE
                CLR     timeout
                CLR     pulseFlag
                CLR     ZH                      ; Clear Z-Pointer MSB
                Clear16Reg  pulseInH,pulseInL   ; Clear pulseIn-Reg
                Clear16Reg  pulseOutH,pulseOutL ; Clear pulseOut-Reg

                LDI     temp,(LOW(SRAMAddress2)-SRAMAddress1)
                MOV     nextSRAMAddress,temp    ; Set nextSRAMAddress

InitMemES:      Read16EEPROM  chAddressEE,tempH,tempL ; Copy pulse-value from EEPROM to SRAM
                Write16SRAM   chAddressEE,tempH,tempL
                SwitchBuffert                   ; Switch Buffert in SRAM
                Write16SRAM   chAddressEE,tempH,tempL
                SwitchBuffert                   ; Switch Buffert in SRAM
                INC     chAddressEE
                INC     chAddressEE
                CPI     chAddressEE,((NrOfChannels*2)+2);
                BRNE    InitMemES

                CBR     pulseFlag,(1<<SYNC)     ; Generate error
                SBR     pulseFlag,(1<<LEDredActive)
                LDI     pulseError,(RecoveryNr*NrOfChannels)

                SEI                             ; Enable GlobalInterrupt



                ;Start Timers
                StartTimer1                     ; Start timer1
                StartTimer0                     ; Start timer0
;-----------------------------------------------------------------------------------------------
; MainLoop
;-----------------------------------------------------------------------------------------------
MainLoop:
```

```
;Output yellow led-status
LEDy:           SBRC    pulseFlag,LEDyellowActive   ; Check if LEDyellowActive is set
                RJMP    LEDyON
LEDyOFF:        CBI     PORTD,LEDyellow             ; LEDyellow OFF (No Error)
                RJMP    LEDr
LEDyON:         SBI     PORTD,LEDyellow             ; LEDyellow ON (Error detected)

;Output red led-status
LEDr:           SBRC    pulseFlag,LEDredActive      ; Check if LEDredActive is set
                RJMP    LEDrON
LEDrOFF:        CBI     PORTD,LEDred                ; LEDred OFF (No Error)
                RJMP    CheckButton
LEDrON:         SBI     PORTD,LEDred                ; LEDred ON (Error detected)
                RJMP    MainLoop

;Save FailSafe-Value in EEPROM if button/jumper activated
CheckButton:    SBIC    PIND,StoreButton            ; Check if button pressed
                CLR     debounceFilter              ; Clear debounce register
                INC     debounceFilter              ; Incerment debounce register
                BRNE    MainLoop

BPressed:       CLI                                 ; Disable GlobalInterrupt
                SBI     PORTD,LEDyellow             ; Led ON (Work in progress)
                CLR     temp
                OUT     PORTB,temp                  ; PORTB = 0x00
                CLR     chAddressEE                 ; Clear chAddressEE register
                SwitchBuffert                       ; Switch Buffert in SRAM

CopyPulseSE:    Read16SRAM      chAddressEE,tempH,tempL     ; Copy SRAM to temp
                Write16EEPROM   chAddressEE,tempH,tempL     ; Copy temp to EEPROM
                INC     chAddressEE                 ; Increment chAddressEE twice
                INC     chAddressEE
                CPI     chAddressEE,((NrOfChannels*2)+2); Check if chAddressFE ==

                BRNE    CopyPulseSE                 ; ((NrOfChannels*2)+2)
                CBI     PORTD,LEDyellow             ; Led OFF (Work complete)
                SwitchBuffert                       ; Switch Buffert in SRAM

ButtonLoop:     SBIS    PIND,StoreButton            ; Check if button pressed
                RJMP    ButtonLoop
                SEI                                 ; Enable GlobalInterrupt
                RJMP    MainLoop

;=============================================================================================
; END
;=============================================================================================
```

Bilaga 7 – Programkod för Drivsteg (MainMotorDriver)

```
;*****************************************************************************************************
;
; Wingman MainMotorDriver                    copyright 2005                              v1.0 (2005-04-22)
;
; Name: Christoffer Martinsson
; E-mail: cm@wsm.se
;
;
;*****************************************************************************************************

include P16F684.INC

; Temp-registers
W_TEMP          EQU             0x20                    ; TempRegister for interrupt-routine
STATUS_TEMP     EQU             0x21                    ; TempRegister for interrupt-routine
DivRegH         EQU             0x22                    ; TempRegister for div-routine
DivRegL         EQU             0x23                    ; TempRegister for div-routine
SetPWM_Temp     EQU             0x24
TempH           EQU             0x25
TempL           EQU             0x26
Temp            EQU             0x27

; Time-definitions
minPulseValue           =.4000;.20000
maxPulseValue           =.11000;.40000
ms1Value                =.5100;20000
ms05Value               =.2550;.10000

;*****************************************************************************************************
; Reset- and Interrupt-vectors
;*****************************************************************************************************
        ORG     0x0000
                        GOTO    Main                    ;Reset-vector
                        ORG             0x0004
                        GOTO    Interrupt       ;Interrupt-vector
;*****************************************************************************************************
; Macro
;*****************************************************************************************************
;-----------------------------------------------------------------------------------------------------
; StoreWregInTemp
;
; Description: Store Status and working-register in temp-registers
;
; Input: -
; ChangedReg: W_TEMP,STATUS_TEMP
; UsedReg: W,STATUS,W_TEMP,STATUS_TEMP
;-----------------------------------------------------------------------------------------------------
StoreWregInTemp MACRO
                MOVWF   W_TEMP                  ;Copy W to TEMP register
                SWAPF   STATUS,W                ;Swap status to be saved into W
```

```
                CLRF    STATUS                  ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
                MOVWF   STATUS_TEMP             ;Save status to bank zero STATUS_TEMP register
                ENDM
;-----------------------------------------------------------------------------------------------
; RestoreWregFromTemp
;
; Description: Restore Status and working-register from temp-registers
;
; Input: -
; ChangedReg: W,STATUS
; UsedReg: W,STATUS,W_TEMP,STATUS_TEMP
;-----------------------------------------------------------------------------------------------
RestoreWregFromTemp     MACRO
                        SWAPF   STATUS_TEMP,W       ;Swap STATUS_TEMP register into W (sets bank to original
state)
                        MOVWF   STATUS              ;Move W into Status register
                        SWAPF   W_TEMP,F            ;Swap W_TEMP
                        SWAPF   W_TEMP,W            ;Swap W_TEMP into W
                        ENDM
;-----------------------------------------------------------------------------------------------
; Sub16L
;
; Description: Subtract literal from 16bit-register.    Reg = Reg - Number, WITH VALID CARRY
;
; Input: RegH, RegL, Number
; ChangedReg: C
; UsedReg: W
;-----------------------------------------------------------------------------------------------
Sub16L          MACRO   RegH, RegL, Number
                        MOVLW   low Number
                        SUBWF   RegL
                        MOVLW   high Number
                        ;BTFSS   STATUS,C
                        ;INCFSZ  high Number,W
                        SUBWF   RegH
                        ENDM
;-----------------------------------------------------------------------------------------------
; Sub16
;
; Description: Subtract literal from 16bit-register.    Reg = Reg - Number, WITH VALID CARRY
;
; Input: RegH, RegL, Number
; ChangedReg: C
; UsedReg: W
;-----------------------------------------------------------------------------------------------
Sub16           MACRO   Reg1H, Reg1L, Reg2H ,Reg2L
                        MOVLW   Reg2L
                        SUBWF   Reg1L
                        MOVLW   Reg2H
                        BTFSS   STATUS,C
                        INCFSZ  Reg2H,W
                        SUBWF   Reg1H
```

```
                        ENDM
;-------------------------------------------------------------------------------------------------
; Set16L
;
; Description: Sets 16bit-register to literal. Reg = Number
;
; Input: RegH, RegL, Number
; ChangedReg: C
; UsedReg: W
;-------------------------------------------------------------------------------------------------
Set16L          MACRO   RegH, RegL, Number
                        MOVLW   low Number
                        MOVWF    RegL
                        MOVLW   high Number
                        MOVWF   RegH
                        ENDM
;-------------------------------------------------------------------------------------------------
; Clear16
;
; Description: Clear 16bit-register.
;
; Input: RegH, RegL
; ChangedReg:
; UsedReg:
;-------------------------------------------------------------------------------------------------
Clear16         MACRO   RegH, RegL
                        CLRF    RegH
                        CLRF    RegL
                        ENDM
;-------------------------------------------------------------------------------------------------
; Comp16L
;
; Description: Compare one 16bit-registers to literal             if Reg1=Number then Z=1.
;
                        if Reg1<Number then C=1.
;
                        if Number<Reg1 then C=0.
; Input: RegH, RegL, Number
; ChangedReg: Z, C
; UsedReg: W
;-------------------------------------------------------------------------------------------------
Comp16L         MACRO   RegH, RegL, Number
                        MOVF    RegH,W
                        SUBLW   high Number

                        BTFSS   STATUS,Z
                        EXITM
                        MOVF    RegL,W
                        SUBLW   low Number
                        ENDM
;-------------------------------------------------------------------------------------------------
; SetPWM
```

```
;
; Description: Sets hardware PWM-output to 8bit register-value. PWM = Reg
;

; Input: Reg
; ChangedReg: CCP1CON(4,5), CCPR1L
; UsedReg: W, CCP1CON(4,5), CCPR1L, Reg, SetPWM_Temp
;-------------------------------------------------------------------------------------------------
SetPWM          MACRO   Reg
                BCF     STATUS,C
                MOVF    Reg,W
                ANDLW   b'00000011'
                MOVWF   SetPWM_Temp
                RLF     SetPWM_Temp,F
                RLF     SetPWM_Temp,F
                RLF     SetPWM_Temp,F
                RLF     SetPWM_Temp,F
                MOVF    CCP1CON,W
                ANDLW   b'11001111'
                IORWF   SetPWM_Temp,W
                MOVWF   CCP1CON
                MOVF    Reg,W
                ANDLW   b'11111100'
                MOVWF   SetPWM_Temp
                RRF     SetPWM_Temp,F
                RRF     SetPWM_Temp,F
                MOVF    SetPWM_Temp,W
                MOVWF   CCPR1L
                ENDM
;=================================================================================================
; Interrupt routines
;=================================================================================================
Interrupt   StoreWregInTemp
;-------------------------------------------------------------------------------------------------
; Timer1_Int
;
; Description: Increment pulseIn counter-registers
;-------------------------------------------------------------------------------------------------
Timer1_IntBCF           STATUS,RP0                      ; Bank 0
                BTFSS   PIR1,TMR1IF
                GOTO    Int_Return
                BCF     PIR1,TMR1IF
                BCF     T1CON,TMR1ON                    ; Timer1 off
;-------------------------------------------------------------------------------------------------
Int_ReturnRestoreWregFromTemp
                RETFIE

;=================================================================================================
; Sub-routines
;=================================================================================================
;-------------------------------------------------------------------------------------------------
; Div16byL
```

```
;
; Description: Divide a 16bit-register with a literal.          DivReg = DivReg / W.
;

; Input: DivRegH, DivRegL, W
; ChangedReg: Z, C, DivRegH, DivRegL
; UsedReg: W, DivRegH, DivRegL
;-------------------------------------------------------------------------------------------------
Div16byL  CALL      DivSkipHiShift
                    iDivRepeat      =8
                    WHILE           iDivRepeat
                    call            DivCode
                    iDivRepeat--
                    ENDW
                    RLF             DivRegL,F                   ; C << lo << C
                    BCF             STATUS,Z
                    RETURN                                      ; we are done!

DivCode              RLF            DivRegL,F                   ; C << lo << C
                    RLF             DivRegH,F                   ; C << hi << C
                    BTFSS           STATUS,C                    ; if Carry
                    GOTO            DivSkipHiShift
                    SUBWF           DivRegH,F                   ; hi-=w
                    BSF             STATUS,C                    ; ignore carry
                    RETURN                                      ; done
                                                                ; endif
DivSkipHiShift
                    SUBWF           DivRegH,F                   ; hi-=w
                    BTFSC           STATUS,C                    ; if carry set
                    RETURN                                      ; done
                    ADDWF           DivRegH,F                   ; hi+=w
                    BCF             STATUS,C                    ; clear carry
                    RETURN
;=================================================================================================
; Main
;=================================================================================================
Main                BCF             STATUS,RP0                  ; Bank 0
                    CLRF            PORTC                       ; Clear PORTC
                    CLRF            PORTA                       ; Clear PORTA
                    MOVLW           0x07
                    MOVWF           CMCON0                      ; Comparator OFF
                    CLRF            CCPR1L
                    MOVLW           b'01001100'
                    MOVWF           CCP1CON
                    MOVLW           b'10000001'
                    MOVWF           ADCON0

                    BSF             STATUS,RP0                  ; Bank 1
                    MOVLW           b'00001101'
                    MOVWF           TRISA
                    CLRF            TRISC
                    MOVLW           b'00000000'
```

```
                MOVWF   ANSEL                   ; All pins to digital I/O except AN0

                MOVLW   0x3F
                MOVWF   PR2
                MOVLW   b'00100000'
                MOVWF   ADCON1
                MOVLW   b'11001111'
                MOVWF   OPTION_REG

                BCF     STATUS,RP0              ; Bank 0
                BCF     T2CON,TMR2ON            ; PWM on
                BSF     INTCON,GIE              ; Global interrup enable
                BSF     T2CON,T2CKPS0

;-------------------------------------------------------------------------------
; MainLoop
;-------------------------------------------------------------------------------
Mainloop: BCF   STATUS,RP0                      ; Bank 0
                CLRWDT
;DECODE_PPM
PPM_Lo:         BTFSC   PORTA,2                 ; Wait for pulse to go low

                GOTO    PPM_Lo
PPM_Hi:         BTFSS   PORTA,2         ; Wait for pulse to go high (start measuring)
                GOTO    PPM_Hi
                Clear16 TMR1H,TMR1L             ; Clear Timer1 countervalue
                BSF     T1CON,TMR1ON    ; Start Timer1
PPM_Loop:       BTFSC   PORTA,2                 ; Wait for pulse to go low (stop measuring)
                GOTO    PPM_Loop
                BCF     T1CON,TMR1ON            ; Stop Timer1
                Sub16L  TMR1H,TMR1L,ms1Value    ; else counter-value - 1ms
                Comp16L TMR1H,TMR1L,ms05Value
                BTFSS   STATUS,C                ; If counter > 0,5ms
                GOTO    PWM_Fwr                 ; then goto PWM_forward
                GOTO    PWM_Rev                 ; else goto PWN_reverse

PWM_Fwr:        BCF     CCP1CON,P1M1            ; Set H-bridge in forward-mode
                Sub16L  TMR1H,TMR1L,ms05Value   ; counter-value - 0,5ms
                GOTO    PWM_SetVal              ; goto PWM_SetValue
PWM_Rev:        BSF     CCP1CON,P1M1            ; Set H-bridge in reverse-mode
                MOVLW   0xFF
                XORWF   TMR1H,F                 ; Invert counterH
                XORWF   TMR1L,F                 ; Invert counterL
                Sub16L  TMR1H,TMR1L,(0xFFFF-ms05Value)  ; Inverted counter-value - (65535-0,5ms)

PWM_SetVal:     MOVF    TMR1H,W                 ; Divide by 79...
                MOVWF   DivRegH                 ; Load div-routine with countervalue
                MOVF    TMR1L,W
                MOVWF   DivRegL                 ; Load div-routine with countervalue
                MOVLW   ((ms05Value/0xFF))      ; Load div-routine with divide-value
                CALL    Div16byL                ; Divide!
```

```
                        ;BTFSC          STATUS,C

                        SetPWM          DivRegL                 ;DivRegL
                        BSF             T2CON,TMR2ON            ; PWM on

                        GOTO            Mainloop
                        END
;=================================================================================================
; END
;=================================================================================================
```

Bilaga 8 – Programkod för Spelarenhet (SecMotorDriver)

```
;*************************************************************************************************
;
; Wingman SecMotorDriver                copyright 2005                v1.0 (2005-04-22)
;
; Name: Christoffer Martinsson
; E-mail: cm@wsm.se
;
;
;*************************************************************************************************

include P16F684.INC

; Temp-registers
W_TEMP          EQU             0x20            ; TempRegister for interrupt-routine
STATUS_TEMP     EQU             0x21            ; TempRegister for interrupt-routine
DivRegH         EQU             0x22            ; TempRegister for div-routine
DivRegL         EQU             0x23            ; TempRegister for div-routine
SetPWM_Temp     EQU             0x24
TempH           EQU             0x25
TempL           EQU             0x26
Temp            EQU             0x27

; Time-definitions
minPulseValue           =.4000;.20000
maxPulseValue           =.11000;.40000
ms1Value                =.5100;20000
ms05Value               =.2550;.10000

;*************************************************************************************************
; Reset- and Interrupt-vectors
;*************************************************************************************************
        ORG     0x0000
                        GOTO    Main            ;Reset-vector
                        ORG     0x0004
                        GOTO    Interrupt       ;Interrupt-vector
;*************************************************************************************************
; Macro
;*************************************************************************************************
;-------------------------------------------------------------------------------------------------
; StoreWregInTemp
;
; Description: Store Status and working-register in temp-registers
;
; Input: -
; ChangedReg: W_TEMP,STATUS_TEMP
; UsedReg: W,STATUS,W_TEMP,STATUS_TEMP
;-------------------------------------------------------------------------------------------------
StoreWregInTemp MACRO
                        MOVWF   W_TEMP                  ;Copy W to TEMP register
                        SWAPF   STATUS,W                ;Swap status to be saved into W
```

```
                        CLRF    STATUS                  ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
                        MOVWF   STATUS_TEMP             ;Save status to bank zero STATUS_TEMP register
                        ENDM
;-------------------------------------------------------------------------------
; RestoreWregFromTemp
;
; Description: Restore Status and working-register from temp-registers
;
; Input: -
; ChangedReg: W,STATUS
; UsedReg: W,STATUS,W_TEMP,STATUS_TEMP
;-------------------------------------------------------------------------------
RestoreWregFromTemp     MACRO
                        SWAPF   STATUS_TEMP,W       ;Swap STATUS_TEMP register into W (sets bank to original state)
                        MOVWF   STATUS              ;Move W into Status register
                        SWAPF   W_TEMP,F            ;Swap W_TEMP
                        SWAPF   W_TEMP,W            ;Swap W_TEMP into W
                        ENDM
;-------------------------------------------------------------------------------
; Sub16L
;
; Description: Subtract literal from 16bit-register.    Reg = Reg - Number, WITH VALID CARRY
;
; Input: RegH, RegL, Number
; ChangedReg: C
; UsedReg: W
;-------------------------------------------------------------------------------
Sub16L          MACRO   RegH, RegL, Number
                        MOVLW   low Number
                        SUBWF   RegL
                        MOVLW   high Number
                        ;BTFSS   STATUS,C
                        ;INCFSZ  high Number,W
                        SUBWF   RegH
                        ENDM
;-------------------------------------------------------------------------------
; Sub16
;
; Description: Subtract literal from 16bit-register.    Reg = Reg - Number, WITH VALID CARRY
;
; Input: RegH, RegL, Number
; ChangedReg: C
; UsedReg: W
;-------------------------------------------------------------------------------
Sub16           MACRO   Reg1H, Reg1L, Reg2H ,Reg2L
                        MOVLW   Reg2L
                        SUBWF   Reg1L
                        MOVLW   Reg2H
                        BTFSS   STATUS,C
                        INCFSZ  Reg2H,W
                        SUBWF   Reg1H
                        ENDM
```

```
;-------------------------------------------------------------------------------------------------
; Set16L
;
; Description: Sets 16bit-register to literal. Reg = Number
;
; Input: RegH, RegL, Number
; ChangedReg: C
; UsedReg: W
;-------------------------------------------------------------------------------------------------
Set16L          MACRO   RegH, RegL, Number
                        MOVLW   low Number
                        MOVWF    RegL
                        MOVLW   high Number
                        MOVWF   RegH
                        ENDM
;-------------------------------------------------------------------------------------------------
; Clear16
;
; Description: Clear 16bit-register.
;
; Input: RegH, RegL
; ChangedReg:
; UsedReg:
;-------------------------------------------------------------------------------------------------
Clear16         MACRO   RegH, RegL
                        CLRF    RegH
                        CLRF    RegL
                        ENDM
;-------------------------------------------------------------------------------------------------
; Comp16L
;
; Description: Compare one 16bit-registers to literal             if Reg1=Number then Z=1.
;
                        if Reg1<Number then C=1.
;
                        if Number<Reg1 then C=0.
; Input: RegH, RegL, Number
; ChangedReg: Z, C
; UsedReg: W
;-------------------------------------------------------------------------------------------------
Comp16L         MACRO   RegH, RegL, Number
                        MOVF    RegH,W
                        SUBLW   high Number

                        BTFSS   STATUS,Z
                        EXITM
                        MOVF    RegL,W
                        SUBLW   low Number
                        ENDM
;-------------------------------------------------------------------------------------------------
; SetPWM
;
```

```
; Description: Sets hardware PWM-output to 8bit register-value. PWM = Reg
;

; Input: Reg
; ChangedReg: CCP1CON(4,5), CCPR1L
; UsedReg: W, CCP1CON(4,5), CCPR1L, Reg, SetPWM_Temp
;-------------------------------------------------------------------------------------------------
SetPWM          MACRO   Reg
                        BCF     STATUS,C
                        MOVF    Reg,W
                        ANDLW   b'00000011'
                        MOVWF   SetPWM_Temp
                        RLF     SetPWM_Temp,F
                        RLF     SetPWM_Temp,F
                        RLF     SetPWM_Temp,F
                        RLF     SetPWM_Temp,F
                        MOVF    CCP1CON,W
                        ANDLW   b'11001111'
                        IORWF   SetPWM_Temp,W
                        MOVWF   CCP1CON
                        MOVF    Reg,W
                        ANDLW   b'11111100'
                        MOVWF   SetPWM_Temp
                        RRF     SetPWM_Temp,F
                        RRF     SetPWM_Temp,F
                        MOVF    SetPWM_Temp,W
                        MOVWF   CCPR1L
                        ENDM
;=================================================================================================
; Interrupt routines
;=================================================================================================
Interrupt       StoreWregInTemp
;-------------------------------------------------------------------------------------------------
; Timer1_Int
;
; Description: Increment pulseIn counter-registers
;-------------------------------------------------------------------------------------------------
Timer1_Int BCF          STATUS,RP0                      ; Bank 0
                        BTFSS   PIR1,TMR1IF
                        GOTO    Int_Return
                        BCF     PIR1,TMR1IF
                        BCF     T1CON,TMR1ON            ; Timer1 off
;-------------------------------------------------------------------------------------------------
Int_Return  RestoreWregFromTemp
                        RETFIE

;=================================================================================================
; Sub-routines
;=================================================================================================
;-------------------------------------------------------------------------------------------------
; Div16byL
;
```

```
; Description: Divide a 16bit-register with a literal.          DivReg = DivReg / W.
;

; Input: DivRegH, DivRegL, W
; ChangedReg: Z, C, DivRegH, DivRegL
; UsedReg: W, DivRegH, DivRegL
;-------------------------------------------------------------------------------------------------
Div16byL   CALL          DivSkipHiShift
                         iDivRepeat =8
                         WHILE          iDivRepeat
                         call           DivCode
                         iDivRepeat--
                         ENDW
                         RLF            DivRegL,F                ; C << lo << C
                         BCF            STATUS,Z
                         RETURN                                  ; we are done!

DivCode                  RLF            DivRegL,F                ; C << lo << C
                         RLF            DivRegH,F                ; C << hi << C
                         BTFSS          STATUS,C                 ; if Carry
                         GOTO           DivSkipHiShift
                         SUBWF          DivRegH,F                ; hi-=w
                         BSF            STATUS,C                 ; ignore carry
                         RETURN                                  ; done
                                                                 ; endif
DivSkipHiShift
                         SUBWF          DivRegH,F                ; hi-=w
                         BTFSC          STATUS,C                 ; if carry set
                         RETURN                                  ; done
                         ADDWF          DivRegH,F                ; hi+=w
                         BCF            STATUS,C                 ; clear carry
                         RETURN
;=================================================================================================
; Main
;=================================================================================================
Main                     BCF            STATUS,RP0               ; Bank 0
                         CLRF           PORTC                    ; Clear PORTC
                         CLRF           PORTA                    ; Clear PORTA
                         MOVLW          0x07
                         MOVWF          CMCON0                                                    ;
Comparator OFF
                         CLRF           CCPR1L
                         MOVLW          b'00001100'
                         MOVWF          CCP1CON
                         MOVLW          b'10000001'
                         MOVWF          ADCON0

                         BSF            STATUS,RP0               ; Bank 1
                         MOVLW          b'11011111'
                         MOVWF          TRISC
                         CLRF           ANSEL
                         MOVLW          0x3F
```

```
                        MOVWF       PR2
                        MOVLW       b'00100000'
                        MOVWF       ADCON1
                        MOVLW       b'11001111'
                        MOVWF       OPTION_REG

                        BCF         STATUS,RP0          ; Bank 0
                        BCF         T2CON,TMR2ON        ; PWM on
                        BSF         INTCON,GIE          ; Global interrup enable
                        BSF         T2CON,T2CKPS0

;-------------------------------------------------------------------------------------------
; MainLoop
;-------------------------------------------------------------------------------------------
Mainloop:   BCF         STATUS,RP0                      ; Bank 0
                        CLRWDT
;DECODE_PPM
PPM_Lo:                 BTFSC       PORTA,2             ; Wait for pulse to go low

                        GOTO        PPM_Lo
PPM_Hi:                 BTFSS       PORTA,2             ; Wait for pulse to go high (start measuring)
                        GOTO        PPM_Hi
                        Clear16     TMR1H,TMR1L         ; Clear Timer1 countervalue
                        BSF         T1CON,TMR1ON        ; Start Timer1
PPM_Loop:               BTFSC       PORTA,2             ; Wait for pulse to go low (stop measuring)
                        GOTO        PPM_Loop
                        BCF         T1CON,TMR1ON        ; Stop Timer1


PWM_SetVal:             MOVF        TMR1H,W             ; Divide by 79...
                        MOVWF       DivRegH             ; Load div-routine with countervalue
                        MOVF        TMR1L,W
                        MOVWF       DivRegL             ; Load div-routine with countervalue
                        MOVLW       (ms1Value/0xFF)     ; Load div-routine with divide-value
                        CALL        Div16byL            ; Divide!

                        SetPWM      DivRegL             ;DivRegL
                        BSF         T2CON,TMR2ON        ; PWM on

                        GOTO        Mainloop
                        END
;===========================================================================================
; END
;===========================================================================================
```