Chapter V

Design Project

This part of the study intends to discuss the theories and principles behind the creation of the project. It aims to explain in detail the technical side of its operation including its components. The project is divided into two major components, the hardware and the software.

The hardware components make up the mechanical and electronics part – the tangible side of the system. The primary components that make human interaction possible – making artificial intelligence expands its capability through extending its control to the outside world.

One aim of the proponents is to make the user of the machine be able to operate it without much dependency to the computer. And with the aid of voice recognition technology, basic commands may be executed through speech using the microphone.

The software components on the other side play important roles because without the software components, the hardware components are useless. These are the components that control the actuators, process data from sensors and implement artificial intelligence to analyze complex chess moves, making the machine play chess on its own.

The software components use cutting edge software technology, the Microsoft .NET Framework 2.0, a modern application architecture which features safe execution of applications, stability and fully organized programming classes and libraries for more efficient, rapid and productive application development. The K-S5 Windows Application is tailored using the Microsoft Visual C# .NET, one of the latest programming languages from Microsoft Technologies which is based under the .NET Framework 2.0. The main application was built using an integrated development environment, the Microsoft Visual C# 2005 Express Edition, an IDE that provides a powerful editor and many tools for creating and debugging C# applications.

5.1 The Hardware

The hardware components are divided into four subsections namely: actuators, chess set, circuits and project container assembly.

5.1.1 Actuators

The actuators form an X, Y and Z components, making-up the mechanical part that makes the machine navigate a three-dimensional space where the chess set is located. Each of the X, Y and Z components has a stepping motor which control the main movement of each part, home sensors which are mechanical switches that are used as starting reference for positioning the head of each actuator, gears, belts and other parts that resembles the moving parts.

Each X, Y and Z components are closely identical to each other except that Z component handles the chess pieces, in picking or capturing them. It uses a 12V electromagnet mounted at the lower part that magnetizes the metal on the head of each chess pieces, making it able to pick or move a certain piece. Two pairs of laser diode and light dependent resistors (LDR) were used to sense the presence or position of a piece before the actuator picks it up.

For economical purposes, the proponents considered recycling as an option for making the actuators; they used improvised materials from old dot-matrix printers and made some modifications instead of buying all new materials to reduce the over-all cost of the project.

5.1.2 Chess Set

The chess set made up the board and chess pieces. The chessboard is made up of 28.6X28.6 centimeters celluloid plastic material, inside it are 8X8 alternate color sticker squares that measures 3.2X3.2 centimeters. A pair of extra two rows (2X8) with the same material and square sizes where place beside the chess board for captured pieces so that the machine will be able to rearrange the chess set in case the human player wanted a rematch.

Beneath each square are hall-effect sensors (UGN-3120) which can determine the presence of magnetic flux induces from each permanent magnet planted under each chess piece. The signals coming from the array of hall-effect

transducers beneath the chess board are fed into the chess board circuit which will be discussed in the circuit section.

Each chess pieces are made up of wood painted black and white, and as mentioned earlier, under each piece has small permanent magnet. To simplify task, the proponents decided to buy commercial chess pieces and modify them to meet the specific requirement.

5.1.3 The Circuits

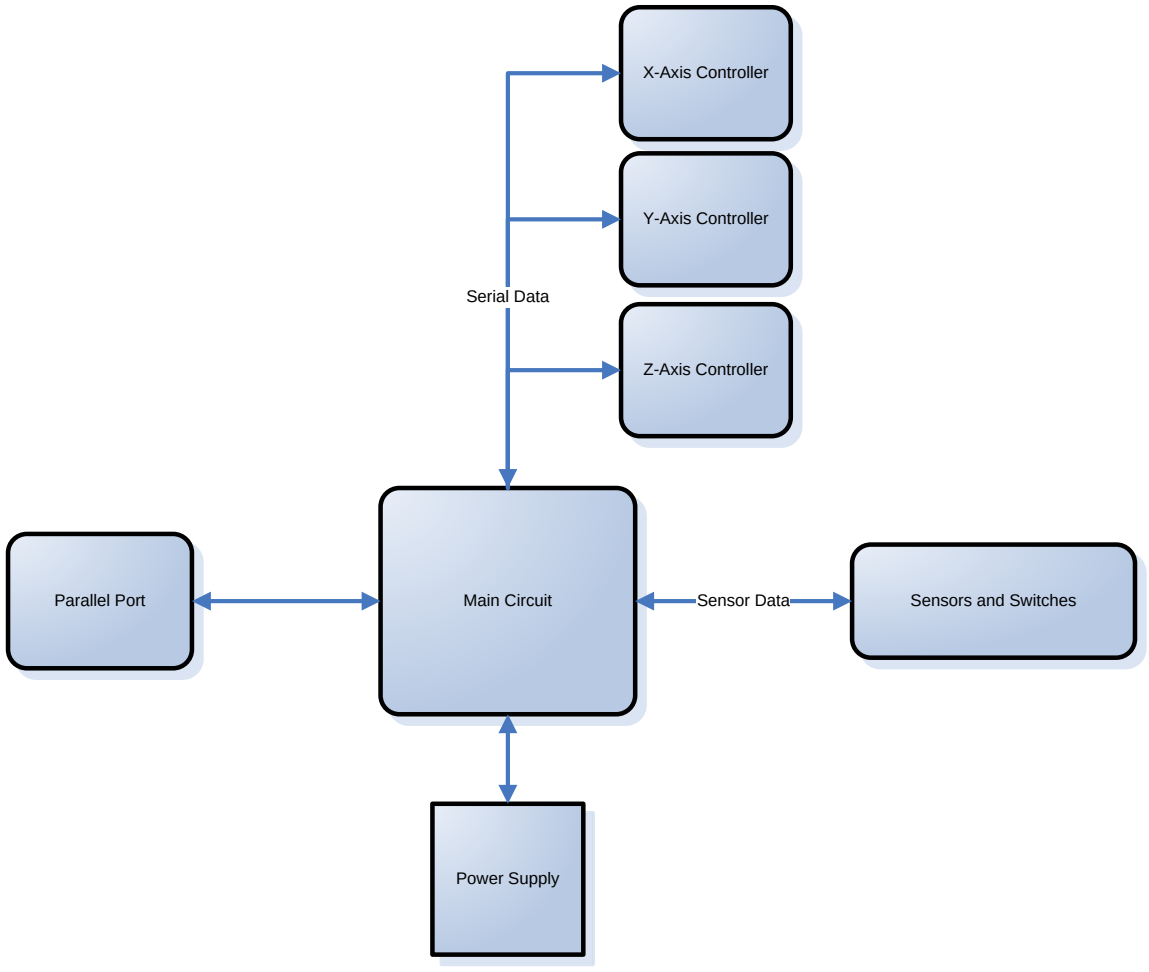


Figure 5.1 Simplified Circuit Diagram

The circuits of the project were divided into different subsections based on each functions and role for the project. These sections are the Main Circuit for multiplexing, expanding of input/output controls and handling miscellaneous I/O components (move indicated and power supply status), the X-Axis Controller, Y-Axis Controller and Z-Axis Controller which control the actuators and other components that are linked to them, Power Supply, and finally, the Sensors circuits which handle the sensor and other inputs from the project's hardware.

The X-Axis Controller Circuit (please see Appendix B for the schematic diagram)

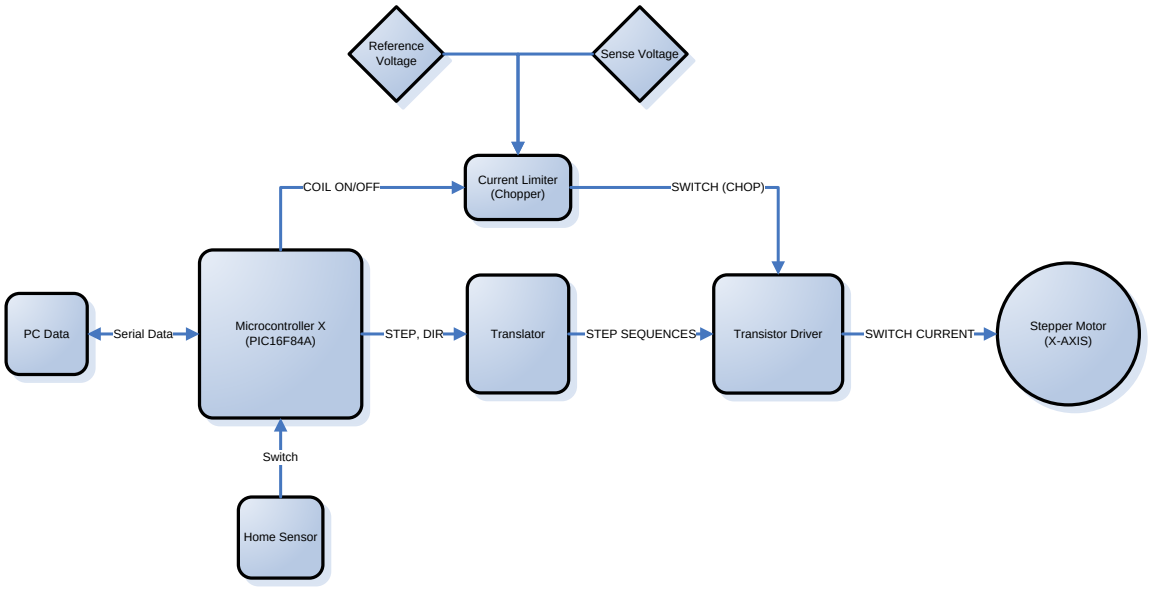


Figure 5.2 The X-Axis Circuit Diagram

Microcontroller X (PIC16F84A)

The PIC16F84A is an 8-bit Enhanced Flash/EEPROM microcontroller which has 13 I/O pins that are user-configured on a pin-to-pin basis. This microcontroller is the primary component responsible for controlling the stepper motor which makes it possible for the movement of the actuators. Running from an 11.059MHz external clock source, the microcontroller controls the motor using the bit 4 of PORTB for motor step (STEP) and bit 5 of PORTB for the direction (DIR). Applying a positive edge for the bit 4 steps the motor once base on the direction of bit 5. A logic high on bit 5 makes the motor step clockwise and logic low for counter-clockwise. Also, the microcontroller has the ability to turn on or turn off the power of the motor coil for low power consumption.

The PIC16F84A controls the actuator and other components related to the motor based on commands received from the personal computer in the form of serial data. The microcontroller uses the RB0/INT interrupt source that detects positive clock edge in RB0 coming from main system from the personal computer, the RB1 is for the input data and RB2 for the output data to the PC.

The source code for X-Axis microcontroller:

```
      PROCESSOR 16F84A
      INCLUDE <P16F84A.INC>
      __config  _HS_OSC & _WDT_OFF & _PWRTE_ON

BUFF1       EQU 0EH
BUFF2 EQU 0FH
BUFF3 EQU 10H
FLAG  EQU 11H
COUNT EQU 12H
```

```
SPC         EQU 13H
SPD         EQU 14H
ACC         EQU 15H
SPCTMP      EQU 16H
DEACC EQU 17H

      ORG 0
      GOTO INIT         ;goto initialization routine

      ORG 4
      GOTO ISR          ;interrupt routine

INIT
      ;########## reg init
      CLRF BUFF1
      CLRF BUFF2
      CLRF BUFF3

      MOVLW .8
      MOVWF COUNT

      MOVLW B'00000111'
      MOVWF FLAG              ;reciv 3 bytes

      MOVLW 0FCH              ;SPEED MULTIPLIER
      MOVWF SPC

      MOVLW 015H              ;SPEED
      MOVWF SPD
      ;########## end of reg init

      MOVLW B'10010000' ;enable RB0/INT interrupt
      MOVWF INTCON

      BCF STATUS,RP0          ;select BANK1
      CLRF PORTB

      BSF STATUS,RP0
      MOVLW B'10000011'
      MOVWF TRISB
      MOVLW B'11111'
      MOVWF TRISA

      MOVLW B'11111111' ;PGT on RB0/INT
      MOVWF OPTION_REG
      BCF STATUS,RP0          ;select BANK0

      CLRF PORTB
      BCF PORTB,6             ;COIL OFF
      BSF PORTB,2             ;ready to reciv data
MAIN
      NOP
      GOTO MAIN

;################################
;################################
ISR
```

```
      ;MOVLW 2
      ;ADDWF PORTB,1

      ;CHECK WHERE TO PUT DATA
      BTFSC FLAG,0
      GOTO GET_BYTE1
      BTFSC FLAG,1
      GOTO GET_BYTE2
      GOTO GET_BYTE3

;###############################



;###############################
;FIRST BYTE OF DATA TO BE RECIV
GET_BYTE1
      RLF BUFF1
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF1
      BSF BUFF1,0
      BTFSS PORTB,1
      BCF BUFF1,0

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .8
      MOVWF COUNT
      BCF FLAG,0
      GOTO ISR_END

;SECOND BYTE OF DATA
GET_BYTE2
      RLF BUFF2
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF2
      BSF BUFF2,0
      BTFSS PORTB,1
      BCF BUFF2,0

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .8
      MOVWF COUNT
      BCF FLAG,1
      GOTO ISR_END

;THIRD BYTE OF DATA
GET_BYTE3
      RLF BUFF3
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF3
      BSF BUFF3,0
      BTFSS PORTB,1
      BCF BUFF3,0

      DECFSZ COUNT
      GOTO ISR_END

      ;end of recieving 3 bytes of data
```

```
      BCF PORTB,2             ;set PIC to busy
      MOVLW .8
      MOVWF COUNT
      MOVLW B'00000111' ;RE-INIT FLAG
      MOVWF FLAG

      BTFSC BUFF3,1           ;SET DIRECTION
      BCF PORTB,5             ;HOME DIR
      BTFSS BUFF3,1
      BSF PORTB,5             ;AWAY DIR

      BTFSC BUFF3,4
      GOTO SET_ONLY           ;without stepping

      BSF PORTB,6             ;AUTO TURN ON COIL B4 DOING ANY STEP
      BTFSS BUFF3,2           ;TEST STOP AT HOME
      GOTO START_STEP         ;NUMBERED STEPS
      GOTO STEP_HOME          ;stop at home sensor
;################################



;################################
SET_ONLY
      BTFSS BUFF3,5
      BCF PORTB,6             ;TURN OFF COIL
      BTFSC BUFF3,5
      BSF PORTB,6             ;TURN ON COIL

      BTFSS BUFF3,0           ;TEST CHANGE SPEED
      GOTO NO_CH_SPEED

      ;CHANGE THE SPEED
      MOVF BUFF1,0
      MOVWF SPD
      MOVF BUFF2,0
      MOVWF SPC

NO_CH_SPEED
      BSF PORTB,2             ;READY TO RECIEVE AGAIN
      GOTO ISR_END
;################################



;################################
STEP_HOME
      BTFSC BUFF3,6
      CALL ACC_STEP_HOME

REG_STEP_HOME
      BTFSS PORTB,7
      GOTO AFTER_STEP

      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY
```

```
      GOTO REG_STEP_HOME
;################################



;################################
      ;setting output
START_STEP
      BTFSC BUFF3,6
      CALL ACC_STEP           ;ACCEL STEPS
REG_STEP
      DECFSZ BUFF1            ;LEAST SIGNIFICANT BYTE
      GOTO STEP_MOTOR         ;steps the motor several times, based on
data stored in DataIn
      DECFSZ BUFF2            ;MOST SIGNIFICANT BYTE
      GOTO REG_STEP

      BTFSC BUFF3,6
      CALL DEACC_STEP         ;DEACCEL MOTOR
      GOTO AFTER_STEP         ;ends stepping when DataIn is zero, and
start waiting for steps again

STEP_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY
      GOTO REG_STEP
      ;CLRF DataIn
;################################



;ACCEL SUBROUTINE, 1 STEP DISCRIPANCY
;################################
ACC_STEP
      SWAPF BUFF2,0           ;SWAP NIBBLES
      ANDLW B'00001111' ;GET ONLY LOWER NIBBLE
      MOVWF ACC               ;W TO ACC
      BTFSS BUFF3,7           ;COMPLETE THE <4:0> BITS FOR ACC
      BCF ACC,4
      BTFSC BUFF3,7
      BSF ACC,4

      ;CLEAR THE ACC STEPS FROM BUFF2
      MOVLW B'00001111'
      ANDWF BUFF2,1

      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

      MOVF ACC,0
      MOVWF DEACC
      ADDWF ACC,0             ;MULTIPLY BY TWO THEN STORE TO W

      ADDWF SPCTMP,1          ;W+SPCTMP ->SPCTMP

      ;ASSUMING ALL THINGS HAVE BEEN SET UP
```

```
      ;START THE ACCEL ACC TIMES, 2 DELAY UNITS
START_ACC
      DECF SPCTMP,1                 ;ACC THE MOTOR BY DECREASING STEP
DELAY
      DECF SPCTMP,1

      DECFSZ ACC              ;DECREMENT ACC
      GOTO ACC_MOTOR          ;STEP THE MOTOR
      RETURN                        ;START STEPPING MOTOR WITHOUT ACC


ACC_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_ACC
;################################



;################################
DEACC_STEP
      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

START_DEACC
      INCF SPCTMP,1           ;INCREASE STEP DELAY
      INCF SPCTMP,1

      DECFSZ DEACC            ;COUNTS DEACC TIMES
      GOTO DEACC_MOTOR
      RETURN


DEACC_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_DEACC
;################################



;ACCEL SUBROUTINE, 1 STEP DISCRIPANCY
;################################
ACC_STEP_HOME
      SWAPF BUFF2,0           ;SWAP NIBBLES
      ANDLW B'00001111' ;GET ONLY LOWER NIBBLE
      MOVWF ACC               ;W TO ACC
      BTFSS BUFF3,7           ;COMPLETE THE <4:0> BITS FOR ACC
      BCF ACC,4
      BTFSC BUFF3,7
```

```
      BSF ACC,4

      ;CLEAR THE ACC STEPS FROM BUFF2
      MOVLW B'00001111'
      ANDWF BUFF2,1

      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

      MOVF ACC,0
      ADDWF ACC,0             ;MULTIPLY BY TWO THEN STORE TO W

      ADDWF SPCTMP,1          ;W+SPCTMP ->SPCTMP

      ;ASSUMING ALL THINGS HAVE BEEN SET UP
      ;START THE ACCEL ACC TIMES, 2 DELAY UNITS
START_ACC_HOME
      DECF SPCTMP,1           ;ACC THE MOTOR BY DECREASING STEP DELAY
      DECF SPCTMP,1

      DECFSZ ACC              ;DECREMENT ACC
      GOTO ACC_MOTOR_HOME     ;STEP THE MOTOR
      RETURN                        ;START STEPPING MOTOR WITHOUT ACC


ACC_MOTOR_HOME
      BTFSS PORTB,7
      GOTO AFTER_STEP

      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_ACC_HOME
;################################




;################################
AFTER_STEP
      ;COIL ON/OFF
      BTFSC BUFF3,0
      BSF PORTB,6             ;COIL ON
      BTFSS BUFF3,0
      BCF PORTB,6

      BSF PORTB,2             ;PIC ready to reciv data again
      GOTO ISR_END
;################################



;################################
ISR_END
```

```
      BCF INTCON,1            ;RBO/INT interrupt did not occur (RETURN
TO MAIN)
      RETFIE
;################################



;################################
DELAY
      MOVF SPC,0
      ;MOVLW 0FCH
      MOVWF 0CH
      MOVF SPD,0
      ;MOVLW 15H  ;15H
D1    MOVWF 0DH
D2    DECFSZ 0DH
      GOTO D2
      DECFSZ 0CH
      GOTO D1
      RETURN
;################################



;################################
DELAY_ACC
      MOVF SPCTMP,0
      ;MOVLW 0FCH
      MOVWF 0CH
      MOVF SPD,0
      ;MOVLW 15H  ;15H
D1_ACC      MOVWF 0DH
D2_ACC      DECFSZ 0DH
      GOTO D2_ACC
      DECFSZ 0CH
      GOTO D1_ACC
      RETURN

      END
```

The Y-Axis Controller Circuit (see Appendix B for the schematic diagram)

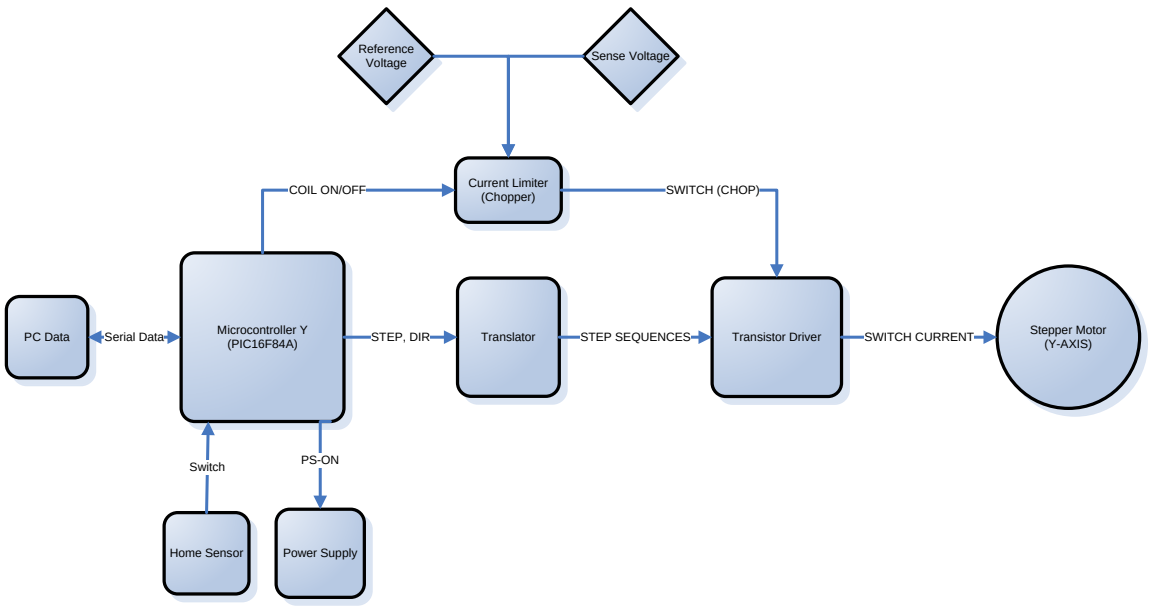


Figure 5.3 The Y-Axis Circuit Diagram

This circuit is much similar to the X-Axis Controller Circuit, based also on a PIC16F84A microcontroller except for some small variations in components used and extra function of the microcontroller – managing the whole circuit's power supply. And because of the motor used by the controller is slightly different from the X-Axis, using the advantage of the chopper circuit, a little adjustment to the reference voltage through the trimmer resistor was made. Delay routines from the microcontrollers instruction code varies a bit faster or slower because of the difference between the motors used. Additional parameter for turning on and off for the power supply is also added.

The microcontroller program code:

```
      PROCESSOR 16F84A
      INCLUDE <P16F84A.INC>
      __config  _HS_OSC & _WDT_OFF & _PWRTE_ON

BUFF1       EQU 0EH
BUFF2 EQU 0FH
BUFF3 EQU 10H
FLAG  EQU 11H
COUNT EQU 12H
PDELAY3     EQU 13H
SPC         EQU 14H
SPD         EQU 15H
ACC         EQU 16H
SPCTMP      EQU 17H
DEACC EQU 18H

      ORG 0
      GOTO INIT         ;goto initialization routine

      ORG 4
      GOTO ISR          ;interrupt routine

INIT
      CALL REG_INIT

      MOVLW B'10010000' ;enable RB0/INT interrupt
      MOVWF INTCON

      BCF STATUS,RP0          ;select BANK1
      CLRF PORTB

      BSF STATUS,RP0
      MOVLW B'10000011'
      MOVWF TRISB
      MOVLW B'11010'
      MOVWF TRISA

      MOVLW B'11111111' ;PGT on RB0/INT
      MOVWF OPTION_REG
      BCF STATUS,RP0          ;select BANK0

      CLRF PORTB
      BCF PORTB,6             ;COIL OFF
      BSF PORTB,2             ;ready to reciv data

      BSF PORTA,0             ;PS-ON OFF
      BCF PORTA,2             ;OFF LED INDICATOR


MAIN
      MOVLW 0FFH
      MOVWF PDELAY3
PWRD3
```

```
      MOVLW 0FFH
      MOVWF 0CH
PWRD1
      MOVLW 0FFH  ;15H
      MOVWF 0DH
PWRD2
      DECFSZ 0DH
      GOTO PWRD2
      DECFSZ 0CH
      GOTO PWRD1
      DECFSZ PDELAY3
      GOTO PWRD3

      BTFSS PORTA,1           ;TEST AUTO-POWER DOWN JUMPER
      GOTO MAIN
      BSF PORTA,0             ;TURN OFF POWER

      GOTO MAIN
;################################



;################################
REG_INIT
      ;########## reg init
      CLRF BUFF1
      CLRF BUFF2
      CLRF BUFF3

      MOVLW .8
      MOVWF COUNT

      MOVLW B'00000111'
      MOVWF FLAG              ;reciv 3 bytes

      MOVLW 0FCH              ;SPEED MULTIPLIER
      MOVWF SPC

      MOVLW 015H              ;SPEED
      MOVWF SPD
      ;########## end of reg init
      RETURN
;################################



;################################
ISR
      ;MOVLW 2
      ;ADDWF PORTB,1

      ;CHECK WHERE TO PUT DATA
      BTFSC FLAG,0
      GOTO GET_BYTE1
      BTFSC FLAG,1
      GOTO GET_BYTE2
```

```
      GOTO GET_BYTE3

;###############################



;###############################
;FIRST BYTE OF DATA TO BE RECIV
GET_BYTE1
      RLF BUFF1
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF1
      BSF BUFF1,0
      BTFSS PORTB,1
      BCF BUFF1,0

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .8
      MOVWF COUNT
      BCF FLAG,0
      GOTO ISR_END

;SECOND BYTE OF DATA
GET_BYTE2
      RLF BUFF2
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF2
      BSF BUFF2,0
      BTFSS PORTB,1
      BCF BUFF2,0

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .8
      MOVWF COUNT
      BCF FLAG,1
      GOTO ISR_END

;THIRD BYTE OF DATA
GET_BYTE3
      RLF BUFF3
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF3
      BSF BUFF3,0
      BTFSS PORTB,1
      BCF BUFF3,0

      DECFSZ COUNT
      GOTO ISR_END

;///end of recieving 3 bytes of data
      BCF PORTB,2             ;set PIC to busy
      MOVLW .8
      MOVWF COUNT
      MOVLW B'00000111' ;RE-INIT FLAG
      MOVWF FLAG

      MOVLW 0FFH
      MOVWF PDELAY3           ;RESET POWER DOWN COUNTER
```

```
      BTFSC BUFF3,1           ;SET DIRECTION
      BSF PORTB,5             ;HOME DIR
      BTFSS BUFF3,1
      BCF PORTB,5             ;AWAY DIR

      BTFSC BUFF3,4
      GOTO SET_ONLY           ;without stepping

      BSF PORTB,6             ;AUTO TURN ON COIL B4 DOING ANY STEP
      BTFSS BUFF3,2           ;TEST STOP AT HOME
      GOTO START_STEP         ;NUMBERED STEPS
      GOTO STEP_HOME          ;stop at home sensor
;###############################



;###############################
SET_ONLY
      BTFSS BUFF3,5
      BCF PORTB,6             ;TURN OFF COIL
      BTFSC BUFF3,5
      BSF PORTB,6             ;TURN ON COIL

      BTFSS BUFF3,6
      BCF PORTA,0             ;TURN ON MAIN POWER
      BTFSC BUFF3,6
      BSF PORTA,0             ;TURN OFF MAIN POWER

      BTFSS BUFF3,7
      BCF PORTA,2             ;TURN OFF LED INDICATOR
      BTFSC BUFF3,7
      BSF PORTA,2             ;TURN ON LED INDICATOR

      BTFSS BUFF3,0           ;TEST SPEED CHANGE
      GOTO NO_CH_SPEED

      ;CHANGE THE SPEED
      MOVF BUFF1,0
      MOVWF SPD
      MOVF BUFF2,0
      MOVWF SPC

NO_CH_SPEED
      BSF PORTB,2             ;READY STATE AGAIN
      GOTO ISR_END
;###############################



;###############################
STEP_HOME
      BTFSC BUFF3,6
      CALL ACC_STEP_HOME

REG_STEP_HOME
      BTFSS PORTB,7
```

```
      GOTO AFTER_STEP

      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY
      GOTO REG_STEP_HOME
;###############################



;###############################
      ;setting output
START_STEP
      BTFSC BUFF3,6
      CALL ACC_STEP           ;ACCEL MOTOR

REG_STEP
      DECFSZ BUFF1
      GOTO STEP_MOTOR         ;steps the motor several times, based on
data stored in DataIn
      DECFSZ BUFF2
      GOTO REG_STEP

      BTFSC BUFF3,6
      CALL DEACC_STEP         ;DEACCEL MOTOR
      GOTO AFTER_STEP         ;ends stepping when DataIn is zero, and
start waiting for steps again

STEP_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY
      GOTO REG_STEP
      ;CLRF DataIn
;###############################



;ACCEL SUBROUTINE, 1 STEP DISCRIPANCY
;###############################
ACC_STEP
      SWAPF BUFF2,0           ;SWAP NIBBLES
      ANDLW B'00001111' ;GET ONLY LOWER NIBBLE
      MOVWF ACC               ;W TO ACC
      BTFSS BUFF3,7           ;COMPLETE THE <4:0> BITS FOR ACC
      BCF ACC,4
      BTFSC BUFF3,7
      BSF ACC,4

      ;CLEAR THE ACC STEPS FROM BUFF2
      MOVLW B'00001111'
      ANDWF BUFF2,1

      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

      MOVF ACC,0
```

```
      MOVWF DEACC
      ADDWF ACC,0             ;MULTIPLY BY TWO THEN STORE TO W

      ADDWF SPCTMP,1          ;W+SPCTMP ->SPCTMP

      ;ASSUMING ALL THINGS HAVE BEEN SET UP
      ;START THE ACCEL ACC TIMES, 2 DELAY UNITS
START_ACC
      DECF SPCTMP,1                ;ACC THE MOTOR BY DECREASING STEP
DELAY
      DECF SPCTMP,1

      DECFSZ ACC              ;DECREMENT ACC
      GOTO ACC_MOTOR          ;STEP THE MOTOR
      RETURN                        ;START STEPPING MOTOR WITHOUT ACC


ACC_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_ACC
;################################



;################################
DEACC_STEP
      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

START_DEACC
      INCF SPCTMP,1           ;INCREASE STEP DELAY
      INCF SPCTMP,1

      DECFSZ DEACC            ;COUNTS DEACC TIMES
      GOTO DEACC_MOTOR
      RETURN


DEACC_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_DEACC
;################################



;ACCEL SUBROUTINE, 1 STEP DISCRIPANCY
;################################
ACC_STEP_HOME
      SWAPF BUFF2,0           ;SWAP NIBBLES
```

```
      ANDLW B'00001111' ;GET ONLY LOWER NIBBLE
      MOVWF ACC               ;W TO ACC
      BTFSS BUFF3,7           ;COMPLETE THE <4:0> BITS FOR ACC
      BCF ACC,4
      BTFSC BUFF3,7
      BSF ACC,4

      ;CLEAR THE ACC STEPS FROM BUFF2
      MOVLW B'00001111'
      ANDWF BUFF2,1

      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

      MOVF ACC,0
      ADDWF ACC,0             ;MULTIPLY BY TWO THEN STORE TO W

      ADDWF SPCTMP,1          ;W+SPCTMP ->SPCTMP

      ;ASSUMING ALL THINGS HAVE BEEN SET UP
      ;START THE ACCEL ACC TIMES, 2 DELAY UNITS
START_ACC_HOME
      DECF SPCTMP,1           ;ACC THE MOTOR BY DECREASING STEP DELAY
      DECF SPCTMP,1

      DECFSZ ACC              ;DECREMENT ACC
      GOTO ACC_MOTOR_HOME     ;STEP THE MOTOR
      RETURN                        ;START STEPPING MOTOR WITHOUT ACC


ACC_MOTOR_HOME
      BTFSS PORTB,7
      GOTO AFTER_STEP

      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_ACC_HOME
;################################




;################################
AFTER_STEP
      ;COIL ON/OFF
      BTFSC BUFF3,0
      BSF PORTB,6             ;COIL ON
      BTFSS BUFF3,0
      BCF PORTB,6

      BSF PORTB,2             ;PIC ready to reciv data again
      GOTO ISR_END
;################################
```

```
;################################
ISR_END
      BCF INTCON,1            ;RBO/INT interrupt did not occur (RETURN
TO MAIN)
      RETFIE
;################################



;################################

DELAY
      MOVF SPC,0
      ;MOVLW 0FCH
      MOVWF 0CH
      ;MOVLW 15H  ;15H
      MOVF SPD,0
D1    MOVWF 0DH
D2    DECFSZ 0DH
      GOTO D2
      DECFSZ 0CH
      GOTO D1
      RETURN
;################################



;################################
DELAY_ACC
      MOVF SPCTMP,0
      ;MOVLW 0FCH
      MOVWF 0CH
      MOVF SPD,0
      ;MOVLW 15H  ;15H
D1_ACC      MOVWF 0DH
D2_ACC      DECFSZ 0DH
      GOTO D2_ACC
      DECFSZ 0CH
      GOTO D1_ACC
      RETURN


      END
```

The Z-Axis Controller Circuit (please see Appendix B for the schematic diagram)

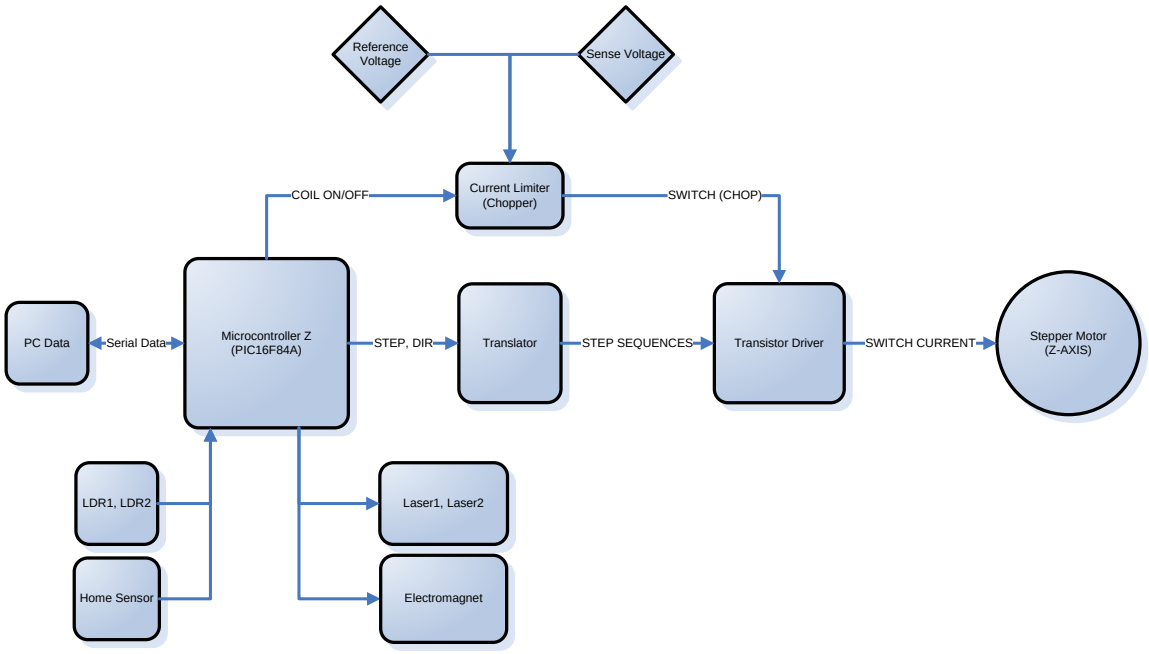


Figure 5.4 The Z-Axis Circuit Diagram

Z-Axis Microcontroller Program Code

```
      PROCESSOR 16F84A
      INCLUDE <P16F84A.INC>
      __config  _HS_OSC & _WDT_OFF & _PWRTE_ON

BUFF1       EQU 0EH
BUFF2 EQU 0FH
BUFF3 EQU 10H
FLAG  EQU 11H
COUNT EQU 12H
SPC         EQU 13H
SPD         EQU 14H
ACC         EQU 15H
SPCTMP      EQU 16H
DEACC EQU 17H

      ORG 0
      GOTO INIT         ;goto initialization routine

      ORG 4
      GOTO ISR          ;interrupt routine
```

```
INIT
      ;########## reg init
      CLRF BUFF1
      CLRF BUFF2
      CLRF BUFF3

      MOVLW .8
      MOVWF COUNT

      MOVLW B'00000111'
      MOVWF FLAG              ;reciv 3 bytes

      MOVLW 0FCH              ;SPEED MUL
      MOVWF SPC

      MOVLW 015H              ;SPEED
      MOVWF SPD
      ;########## end of reg init

      MOVLW B'10010000' ;enable RB0/INT interrupt
      MOVWF INTCON

      BCF STATUS,RP0          ;select BANK1
      CLRF PORTB

      BSF STATUS,RP0
      MOVLW B'10000011'
      MOVWF TRISB
      MOVLW B'00011'
      MOVWF TRISA

      MOVLW B'11111111' ;PGT on RB0/INT
      MOVWF OPTION_REG
      BCF STATUS,RP0          ;select BANK0

      CLRF PORTB
      CLRF PORTA

      BCF PORTB,6             ;COIL OFF
      BSF PORTB,2             ;ready to reciv data
MAIN
      NOP
      GOTO MAIN

;################################
;################################
ISR
      ;MOVLW 2
      ;ADDWF PORTB,1

      ;CHECK WHERE TO PUT DATA, RECIV OR WHAT TO DO
      BTFSC FLAG,7
      GOTO AFTER_REPORT ;PIECE REPORT

      BTFSC FLAG,4
      GOTO SEND_BYTE1
      BTFSC FLAG,5
```

```
      GOTO SEND_BYTE2

      BTFSC FLAG,0
      GOTO GET_BYTE1
      BTFSC FLAG,1
      GOTO GET_BYTE2
      GOTO GET_BYTE3
;################################



;################################
SEND_BYTE1
      BTFSC BUFF1,7           ;TEST MSB OF BUFF1
      BSF PORTB,2             ;SET DATA OUT
      BTFSS BUFF1,7           ;
      BCF PORTB,2             ;CLR DATA OUT
      RLF BUFF1

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .9                ;1 CLK FOR FINAL RECEPTION
      MOVWF COUNT
      BCF FLAG,4
      GOTO ISR_END
;################################



;################################
SEND_BYTE2
      BTFSC BUFF2,7           ;TEST MSB OF BUFF1
      BSF PORTB,2             ;SET DATA OUT
      BTFSS BUFF2,7           ;
      BCF PORTB,2             ;CLR DATA OUT
      RLF BUFF2

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .8
      MOVWF COUNT
      BCF FLAG,5
      BSF PORTB,2             ;READY TO RECIV AGAIN
      GOTO ISR_END
;################################



;################################
;FIRST BYTE OF DATA TO BE RECIV
GET_BYTE1
      RLF BUFF1
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF1
      BSF BUFF1,0
      BTFSS PORTB,1
```

```
      BCF BUFF1,0

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .8
      MOVWF COUNT
      BCF FLAG,0
      GOTO ISR_END

;SECOND BYTE OF DATA
GET_BYTE2
      RLF BUFF2
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF2
      BSF BUFF2,0
      BTFSS PORTB,1
      BCF BUFF2,0

      DECFSZ COUNT
      GOTO ISR_END
      MOVLW .8
      MOVWF COUNT
      BCF FLAG,1
      GOTO ISR_END

;THIRD BYTE OF DATA
GET_BYTE3
      RLF BUFF3
      BTFSC PORTB,1           ;input data from RB1 of PORTB to BUFF3
      BSF BUFF3,0
      BTFSS PORTB,1
      BCF BUFF3,0

      DECFSZ COUNT
      GOTO ISR_END

;///end of recieving 3 bytes of data
      BCF PORTB,2             ;set PIC to busy
      MOVLW .8
      MOVWF COUNT
      MOVLW B'00000111' ;RE-INIT FLAG
      MOVWF FLAG

      BTFSC BUFF3,1           ;SET DIRECTION
      BSF PORTB,5             ;HOME DIR
      BTFSS BUFF3,1
      BCF PORTB,5             ;AWAY DIR

      BTFSC BUFF3,4
      GOTO SET_ONLY           ;without stepping

      BTFSC BUFF3,5
      GOTO REPORT_PIECE ;REPORT IS PIECE IS PRESENT

      BSF PORTB,6             ;AUTO TURN ON COIL B4 STEP
      BTFSC BUFF3,2           ;TEST STOP AT HOME
      GOTO HOME         ;stop at home sensor?
```

```
      BTFSC BUFF3,3           ;TEST STOP AT LDR
      GOTO STEP_LDR
      GOTO START_STEP         ;NUMBERED STEPS
;################################



;################################
SET_ONLY

      BTFSS BUFF3,5
      BCF PORTB,6             ;TURN OFF COIL
      BTFSC BUFF3,5
      BSF PORTB,6             ;TURN ON COIL

      BTFSS BUFF3,6
      BCF PORTA,2             ;TURN OFF LASER
      BTFSC BUFF3,6
      BSF PORTA,2

      BTFSS BUFF3,7
      BCF PORTA,3             ;TURN OFF EM
      BTFSC BUFF3,7
      BSF PORTA,3             ;TURN ON EM

      BTFSS BUFF3,0           ;TEST CHANGE SPEED
      GOTO NO_CH_SPEED

      ;CHANGE THE SPEED
      MOVF BUFF1,0
      MOVWF SPD
      MOVF BUFF2,0
      MOVWF SPC

NO_CH_SPEED
      BSF PORTB,2
      GOTO ISR_END
;################################



;################################
HOME
      BTFSC BUFF3,3
      GOTO STEP_LDR
STEP_HOME
      BTFSC BUFF3,6
      CALL ACC_STEP_HOME

REG_STEP_HOME
      BTFSS PORTB,7
      GOTO AFTER_STEP

      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY
```

```
      GOTO REG_STEP_HOME
;################################



;################################
STEP_LDR
      BTFSC PORTA,0           ;TEST LDR0 (RA0)
      GOTO REPORT
      BTFSC PORTA,1           ;TEST LDR1 (RA1)
      GOTO REPORT

      ;BCF PORTB,4
      ;BSF PORTB,4
      ;CALL DELAY
      ;GOTO STEP_LDR
      DECFSZ BUFF1            ;DEC BYTE1
      GOTO STEP_MOTOR_LDR     ;START STEPPING
      DECFSZ BUFF2            ;DEC BYTE2
      GOTO STEP_LDR           ;CHECK AGAIN
      GOTO REPORT             ;CHECK REPORT EN

STEP_MOTOR_LDR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY
      GOTO STEP_LDR
;################################



;################################
REPORT
      BTFSS BUFF3,2
      GOTO AFTER_STEP
STEP_REPORT
      BSF FLAG,4
      BSF FLAG,5
      GOTO AFTER_STEP
;################################



;################################
REPORT_PIECE
      BCF PORTB,2             ;INITIAL: NO PIECE
      BTFSC PORTA,0
      BSF PORTB,2             ;THERE IS (RA0 BLOCKED)
      BTFSC PORTA,1
      BSF PORTB,2             ;THRE IS (RA1 BLOCKED)

      BSF FLAG,7
      GOTO ISR_END
;################################
```

```
;################################
AFTER_REPORT
      BCF FLAG,7
      BSF PORTB,2
      GOTO ISR_END
;################################




;################################
      ;setting output
START_STEP
      BTFSC BUFF3,6
      CALL ACC_STEP           ;ACCEL MOTOR

REG_STEP
      DECFSZ BUFF1
      GOTO STEP_MOTOR         ;steps the motor several times, based on
data stored in DataIn
      DECFSZ BUFF2
      GOTO REG_STEP

      BTFSC BUFF3,6
      CALL DEACC_STEP         ;DEACCEL MOTOR
      GOTO AFTER_STEP         ;ends stepping when DataIn is zero, and
start waiting for steps again

STEP_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY
      GOTO REG_STEP
;################################




;ACCEL SUBROUTINE, 1 STEP DISCRIPANCY
;################################
ACC_STEP
      SWAPF BUFF2,0           ;SWAP NIBBLES
      ANDLW B'00001111' ;GET ONLY LOWER NIBBLE
      MOVWF ACC               ;W TO ACC
      BTFSS BUFF3,7           ;COMPLETE THE <4:0> BITS FOR ACC
      BCF ACC,4
      BTFSC BUFF3,7
      BSF ACC,4

      ;CLEAR THE ACC STEPS FROM BUFF2
      MOVLW B'00001111'
      ANDWF BUFF2,1
```

```
      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

      MOVF ACC,0
      MOVWF DEACC
      ADDWF ACC,0             ;MULTIPLY BY TWO THEN STORE TO W

      ADDWF SPCTMP,1          ;W+SPCTMP ->SPCTMP

      ;ASSUMING ALL THINGS HAVE BEEN SET UP
      ;START THE ACCEL ACC TIMES, 2 DELAY UNITS
START_ACC
      DECF SPCTMP,1                ;ACC THE MOTOR BY DECREASING STEP
DELAY
      DECF SPCTMP,1

      DECFSZ ACC              ;DECREMENT ACC
      GOTO ACC_MOTOR          ;STEP THE MOTOR
      RETURN                        ;START STEPPING MOTOR WITHOUT ACC


ACC_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_ACC
;################################



;################################
DEACC_STEP
      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

START_DEACC
      INCF SPCTMP,1           ;INCREASE STEP DELAY
      INCF SPCTMP,1

      DECFSZ DEACC            ;COUNTS DEACC TIMES
      GOTO DEACC_MOTOR
      RETURN


DEACC_MOTOR
      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_DEACC
;################################
```

```
;ACCEL SUBROUTINE, 1 STEP DISCRIPANCY
;###############################
ACC_STEP_HOME
      SWAPF BUFF2,0           ;SWAP NIBBLES
      ANDLW B'00001111' ;GET ONLY LOWER NIBBLE
      MOVWF ACC               ;W TO ACC
      BTFSS BUFF3,7           ;COMPLETE THE <4:0> BITS FOR ACC
      BCF ACC,4
      BTFSC BUFF3,7
      BSF ACC,4

      ;CLEAR THE ACC STEPS FROM BUFF2
      MOVLW B'00001111'
      ANDWF BUFF2,1

      MOVF SPC,0              ;SPC -> SPCTMP
      MOVWF SPCTMP

      MOVF ACC,0
      ADDWF ACC,0             ;MULTIPLY BY TWO THEN STORE TO W

      ADDWF SPCTMP,1          ;W+SPCTMP ->SPCTMP

      ;ASSUMING ALL THINGS HAVE BEEN SET UP
      ;START THE ACCEL ACC TIMES, 2 DELAY UNITS
START_ACC_HOME
      DECF SPCTMP,1           ;ACC THE MOTOR BY DECREASING STEP DELAY
      DECF SPCTMP,1

      DECFSZ ACC              ;DECREMENT ACC
      GOTO ACC_MOTOR_HOME     ;STEP THE MOTOR
      RETURN                        ;START STEPPING MOTOR WITHOUT ACC


ACC_MOTOR_HOME
      BTFSS PORTB,7
      GOTO AFTER_STEP

      BCF PORTB,4
      BSF PORTB,4
      CALL DELAY_ACC

      GOTO START_ACC_HOME
;###############################




;###############################
AFTER_STEP
      ;COIL ON/OFF
      BTFSC BUFF3,0
      BSF PORTB,6             ;COIL ON
      BTFSS BUFF3,0
      BCF PORTB,6
```

```
      BSF PORTB,2             ;PIC ready to reciv data again
      GOTO ISR_END
;################################



;################################
ISR_END
      BCF INTCON,1            ;RBO/INT interrupt did not occur (RETURN
TO MAIN)
      RETFIE
;################################



;################################
DELAY
      MOVF SPC,0
      ;MOVLW 0FCH
      MOVWF 0CH
      ;MOVLW 15H  ;15H
      MOVF SPD,0
D1    MOVWF 0DH
D2    DECFSZ 0DH
      GOTO D2
      DECFSZ 0CH
      GOTO D1
      RETURN
;################################



;################################
DELAY_ACC
      MOVF SPCTMP,0
      ;MOVLW 0FCH
      MOVWF 0CH
      MOVF SPD,0
      ;MOVLW 15H  ;15H
D1_ACC      MOVWF 0DH
D2_ACC      DECFSZ 0DH
      GOTO D2_ACC
      DECFSZ 0CH
      GOTO D1_ACC
      RETURN

      END
```

The Translator Circuit

The 74LS191 is a synchronous, reversible up/down counter with a single clock for counting and direction control and the 74LS139 is a dual 1 of 4 Decoder. These integrated circuits and few other components serve as the translator for the driving circuit of the motor. From input data STEP and DIR from the PIC16F84A microcontroller, the STEP serves as the clock source for 74LS191 for 4-bit binary counting and DIR for the UP/DOWN count. The bit 0 and bit 1 from the output of the UP/DOWN counter then enters the 1 of 4 decoder which selects one of the four transistor drivers to be switch to allow the flow of current to the motor coil. The output from the decoder is considered as the step sequences for the motor which controls the stepping and direction by powering the coils in the appropriate sequence and interval.

The Current Limiter Circuit (Chopper)

There are several reasons for implementing current control. Some of the advantages are as follows:

- Avoid Overheating
- Accommodate a variety of Motors
- Increase Torque at higher speed
- Increase Top Speed

- Improve Power Efficiency

In its simplest configuration, a stepper motors draws it's rated current when connected to its rated voltage. The purpose of this 'rating' is to give the circuit designer guidance as to when the motor will over-heat or how much current can continuously run through the windings without overheating the motor. A motor should not over-heat (when appropriately mounted) if connected to its rated voltage. Over-heating should be avoided, since it adversely affects the performance of the motor and even presents serious safety issues.

However, the exact operating voltages of the motor used by the proponents were not available. And it may be desirable to drive the motor from a supply voltage higher than the rated voltage (with proper current limiting). The motors can always be driven with less than the rated voltage, but the motor will simply run at less than their full torque capability.

When the operating voltage is higher than the voltage rating of the motor, the simplest way of limiting the current through the motor windings, thus staying within the safe operating range of the motor, is with a current limiting resistor, using the L/R drivers (L=Inductance, R=Resistance). This resistor, placed in series with the winding diverts some current from the winding and dissipates it in the form of heat. This is the major problem that the proponents encountered will designing the appropriate driving circuit for the stepper motors.

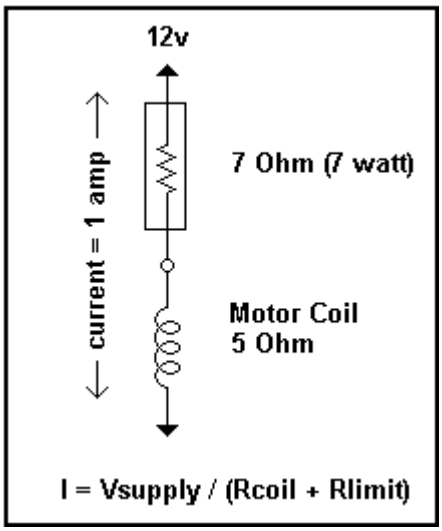


Figure 5.5 Power Resistor

The voltages and current in this simple circuit can be analyzed as a simple resistive voltage divider. Because of the small resistances of about 1 to 20 ohms, a special type of resistor is needed, called a 'power' resistor. Power is expressed as Watts = Volts * Amps, and a little calculation shows that the resistor needed can easily exceed 10 watts, which translates into a lot of heat. Another way to reduce the current is with the use of transistors instead of a resistor, but the same heat dissipation problem was encountered by the proponents. Linearly regulating the current flow within a circuit will always face a heat problem.

Fortunately there is a technique, known as ‘Chopping’, which can limit the current in the winding without generating excess heat using the comparator LM339. It is quite elegant and efficient. The essence of chopping is to switch the operating voltage on and off at a frequency higher than the operating range, and allow the motor itself to act as a filter, through the concept of Pulse-width

modulation, in which the 'duty cycle' (percentage of on time) determines the behavior of the motor or whatever is being controlled.

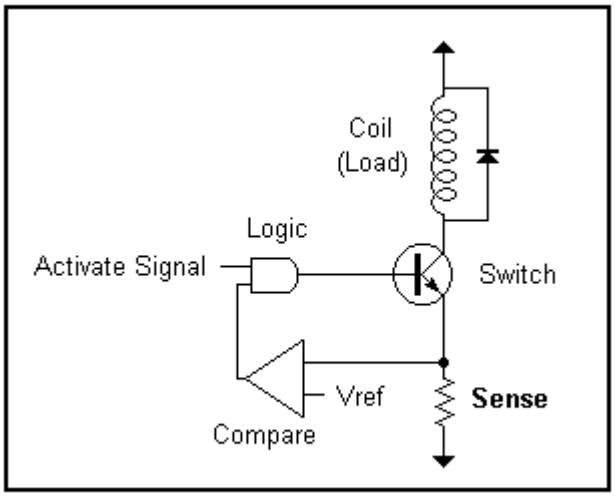


Figure 5.6 Conceptual Model of Chopper Circuit

When an incoming "activate" signal is received from the translator circuit, the load (stepper coil) is switched on. Current through the coil is developed as a voltage across a "sense" resistor, the Rsense (1ohm). This voltage is proportional to the amount of current in the coil and forms an important part of the feedback sensor. The value of the resistor Rsense is very low (1ohm) which equates to 1 amp, when the voltage at the sense resistor is 1 volt.) The sensed voltage is compared with a reference voltage, and when the sensed voltage becomes greater than the reference the logic switches the coil off. When the voltage drops below the reference voltage the coil is switched back on, unless the incoming activate signal is removed, in which case the coil is always off. Thus the feedback logic flips the switch on and off when the current is too high or

to low, maintaining a constant amount of current. The reference voltage is adjustable using a 150ohm potentiometer (R31) in series the resistor R22 (220 ohm) which allows the matching of current in the circuit to the motors rated current (see Appendix B).

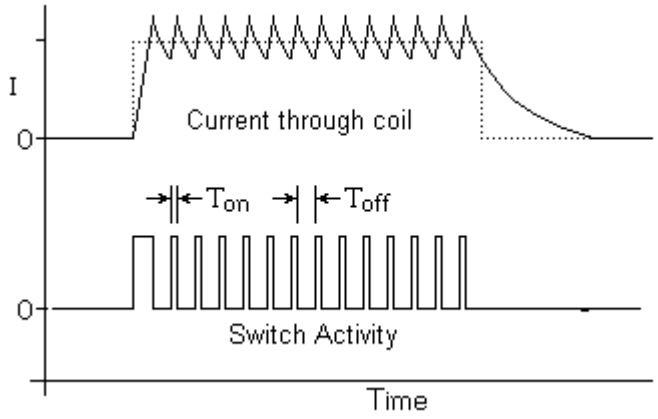


Figure 5.7 Waveforms of the Chopper Circuit

Looking at the waveforms of the chopper circuit, the current through the coil (top waveform) increase and then sawtooth about the desired current setting (as determined by the reference voltage.) The waveform indicates the switch activity logic turning the coil on and off at high frequency (typically 20 KHz)

Benefits in Using the Chopper Circuit

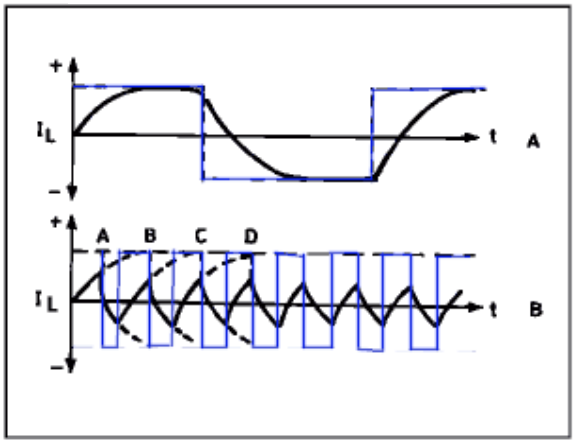
Accommodate a variety of Motors

The reference voltage in the feedback circuit is adjustable and provides a mechanism for regulating the amount of current in the coil(s) of the stepper

motor. The reference voltage is set through the potentiometer R31. As the motor turns faster the current in the motor drops and lose the current regulating effect. In addition to limiting the current, the chopping circuit actually helps maintain a constant amount of current, independent of load and the motor speed.

Increase Top Speed

The important reason for current limiting is to increase the top speed of the motor. The coils in the stepper are inductors - when a voltage is applied, it takes time for the current level to increase. The rise and fall time in reaction to an input step waveform is known as the 'slew rate'. It takes time for the current to 'slew' from one level to another. When the step frequency is low, this logarithmic slope doesn't affect the performance of the motor, since there is plenty of time for the current to reach full level relative to the step rate. However as the step rate approaches the time it takes for the current to ramp up, it begins to have an effect on the motor performance. The current does not have enough time to reach full value before the next step reverses the current flow. This is illustrated in the graph below.



At high step frequencies the winding current cannot reach full value because of the direction change.

Figure 5.8 Filtering effect of coil inductance

As shown above, when the step frequency is low (A) the current has time to reach full, ideal level. But, as the step frequency increases the current cannot reach full level, before it must reverse direction. The resultant waveform (B) is a wimpy, under-performing version of what is needed by the motor that controls the actuator that gets worse and worse as the step frequency increases. In terms of motor performance, the motor loses power the faster the step until eventually there's not enough power to move the actuator.

The solution to getting a better slew rate (a current waveform with sharper attack) is to increase the supply voltage (+12V). This decreases the time necessary to charge to full level, resulting in a 'squarer' looking waveform. However, now there's a problem which must be addressed – there would be too much current in the circuit. This current must be limited. A better slew rate is needed without over-heating the motor.

But the best performance is obtained with the chopping (Current Limited) circuit which can turn on the current with the sharpest attack and simply switch it on and off to maintain this level.

The Stepper Motor

Unipolar stepping motors were used for the actuators' means of movement. This permanent magnet and hybrid stepping motor with 5 or 6 wires is usually wired, as shown in the schematic in Figure 5.7, with a center tap on each of two windings.

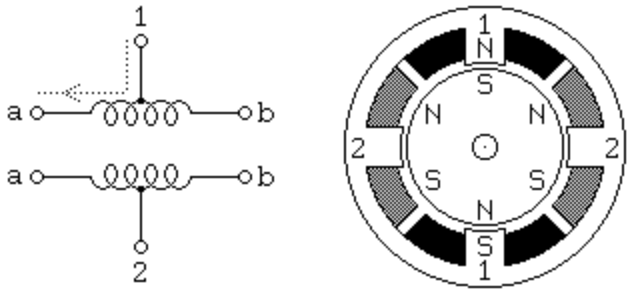


Figure 5.9 Unipolar Motors

The center taps of the windings are typically wired to the positive supply, and the two ends of each winding are alternately grounded to reverse the direction of the field provided by that winding. Motor winding number 1 is distributed between the top and bottom stator pole, while motor winding number 2 is distributed between the left and right motor poles. The rotor is a permanent magnet with 6 poles, 3 south and 3 north, arranged around its circumference.

Compared to the Bipolar stepping motors, Unipolar motors are more simple when it comes to drive circuitry since Bipolar motors requires an H-Bridge control circuit for each winding to reverse the polarity of each pair of motor poles. Yet when it comes to motor windings, bipolar motors are wired more simply since they don't have center taps (see Figure 5.8).

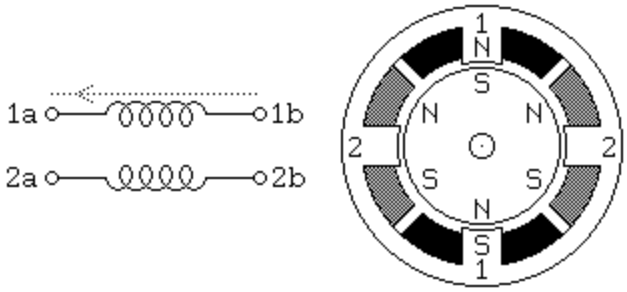


Figure 5.10 Bipolar Motors

Changing the Motor Coil Current

One major reason for increasing the current flowing through the coil of the motor is to achieve the desired torque of the motor. For setting the current flowing through the coil to a certain motor, reference voltage must be adjusted. Because of the reference voltage, when the desired current through the coil is reached, for example, 1A which equates to 1V (reference voltage) at 1Ω (sense resistor), the comparator will turn the coil off. When the current starts to drop below 1A, the comparator will then turn on the current through the coil, creating a cycle that turns on and off the current coil.

Because the reference voltage uses the basic principle of ohms law, the variable resistor concerned with the reference voltage is in series with a 200ohm resistor supplied by 5V.

For example, to set the motor current to 1A,

Because of the 1ohm sense resistor in series with the motor coil:

Desired Current = Sense Resistor Voltage = Voltage Reference

First, get the current flowing to the variable resistor:

$$I_{Tref} = \frac{(5 - \text{Desired Voltage})}{200\text{ohm}}$$

$$I_{Tref} = \frac{5V\text{-}1V}{200\Omega}$$

$$I_{Tref} = 0.02A$$

For the resistance of variable resistor,

$$R_{var} = \frac{\text{Desired Voltage}}{I_{Tref}}$$

$$R_{var} = \frac{1V}{0.02A}$$

$R_{var} = 50\Omega$, resistance at the variable resistor.

To compute for the power dissipated through the coil,

Vcoil = 12V – VRsense (Vref)
Vcoil = 12V – 1V
Vcoil = 11V

Power = Vcoil * Desired Current
Power = 11V * 1A
Power = 11W, approximate value

The Standard Parallel Port

The Parallel Port is the most commonly used port for interfacing projects and devices for research. This port allows the input of up to 9 bits or the output of 12 bits at any one given time, thus requiring minimal external circuitry to implement many simpler tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. This will be a serial RS-232 port and thus, is a totally incompatible port.

- 8 output pins accessed via the **DATA Port**
- 5 input pins (one inverted) accessed via the **STATUS Port**
- 4 input/output pins (three inverted) accessed via the **CONTROL Port**
- The remaining 8 pins are grounded

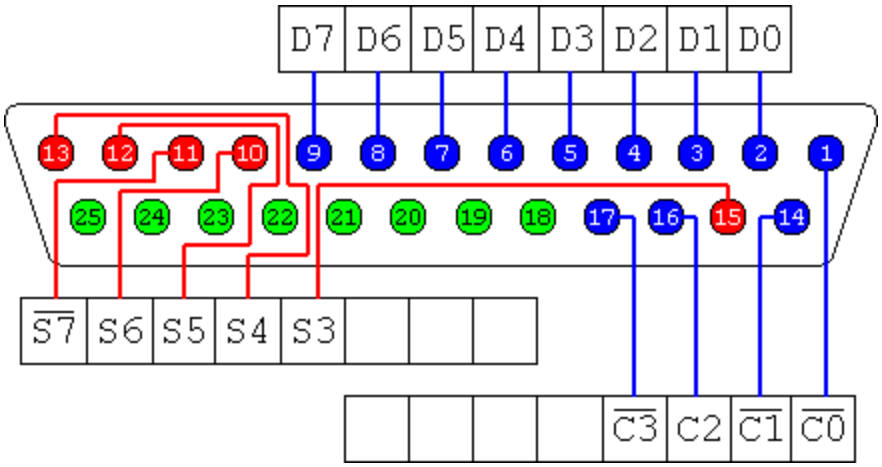


Figure 5.11 D-Type 25 Pin Connector

Hardware Properties

The D-Type 25 pin connector is the most common connector found on the Parallel Port of the computer, while the Centronics Connector is commonly found

on printers. The IEEE 1284 standard however specifies 3 different connectors for use with the Parallel Port. The first one, 1284 Type A is the D-Type 25 connector found on the back of most computers. The 2nd is the 1284 Type B which is the 36 pin Centronics Connector found on most printers. IEEE 1284 Type C however, is a 36 conductor connector like the Centronics, but smaller.

Using connectors from old printers, the proponents decided to implement the 36-pin Centronics (1284 Type B) connector to the project because of its abundance and availability.

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-Printer	In/Out	Control	Yes
18-25	19-30	Ground	Gnd		

Table 5.1. Pin Assignments of the D-Type 25 pin Parallel Port Connector.

The above table uses "n" in front of the signal name to denote that the signal is active low, e.g. nError. If the printer has occurred an error then this line is low. This line normally is high, should the printer be functioning correctly. The "Hardware Inverted" means the signal is inverted by the Parallel card's hardware. Such an example is the Busy line. If +5v (Logic 1) was applied to this pin and the status register read, it would return back a 0 in Bit 7 of the Status Register.

The output of the Parallel Port is normally TTL logic levels. The voltage levels are the easy part. The current you can sink and source varies from port to port. Most Parallel Ports implemented in ASIC, can sink and source around 12mA. However these are just some of the figures taken from Data sheets, Sink/Source 6mA, Source 12mA/Sink 20mA, Sink 16mA/Source 4mA, Sink/Source 12mA. As stated the values vary quite a bit. This is why the proponents decided to use the 74LS244 buffer, so the least current is drawn from the Parallel Port.

Software Registers - Standard Parallel Port (SPP)

Offset	**Name**	**Read/Write**	**Bit No.**	**Properties**
Base + 0	Data Port	Write (Note-1)	Bit 7	Data 7 (Pin 9)
			Bit 6	Data 6 (Pin 8)
			Bit 5	Data 5 (Pin 7)
			Bit 4	Data 4 (Pin 6)
			Bit 3	Data 3 (Pin 5)
			Bit 2	Data 2 (Pin 4)
			Bit 1	Data 1 (Pin 3)
			Bit 0	Data 0 (Pin 2)

Table 5.2 Data Port

Note 1: If the Port is bi-directional then Read and Write Operations can be performed on the Data Register.

The base address, usually called the Data Port or Data Register is simply used for outputting data on the Parallel Port's data lines (Pins 2-9). This register is normally a write only port. If you read from the port, you should get the last byte sent. However if your port is bi-directional, you can receive data on this address.

Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

Table 5.3 Status Port

The Status Port (base address + 1) is a read only port. Any data written to this port will be ignored. The Status Port is made up of 5 input lines (Pins 10,11,12,13 & 15), a IRQ status register and two reserved bits. Please note that Bit 7 (Busy) is a active low input. E.g. If bit 7 happens to show a logic 0, this means that there is +5v at pin 11. Likewise with Bit 2. (nIRQ) If this bit shows a '1' then an interrupt has not occurred.

Base + 2	Control	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable bi-directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

Table 5.4 Control Port

The Control Port (base address + 2), was intended as a write only port. When a printer is attached to the Parallel Port, four "controls" are used. These are Strobe, Auto Linefeed, Initialize and Select Printer, all of which are inverted except Initialize.

The printer would not send a signal to initialize the computer, nor would it tell the computer to use auto linefeed. However these four outputs can also be used for inputs. If the computer has placed a pin high (e.g. +5v) and your device wanted to take it low, you would effectively short out the port, causing a conflict on that pin. Therefore these lines are "open collector" outputs *(or open drain for CMOS devices).* This means that it has two states. A low state (0v) and a high impedance state (open circuit).

POWER SUPPLY

One major problem encountered by the proponents is designing the appropriate power supply. Although the project is power efficient has power

saving features, for proper and safety operation of the project, high current output power supply must be used. To prevent design complexity, time constraints, and because of the high current requirement of the project (stepper motors), the proponents decided to use an ATX power supply instead of making their own power supply for some logical reasons:

- Enabling the proponents focus on designing the primary circuits and components instead of allotting much time for making the necessary power supply design.
- Fully compatible and suitable for the project, regulated +5V DC for integrated/logic circuitry and +12V DC for motors and others components requiring higher supply voltage and current.
- High output current rating typically ~30A for +5V and ~15A for +12V
- Industry standard, efficient, and widely available.
- Safe. Features auto-power off which automatically shuts down itself when improperly operated.
- Easily replaceable in case of disaster or failure
- Compact, uses switching technology – eliminating the use of large transformer and capacitors (in contrast to producing high current output which requires large transformer and capacitors for traditional power supply design).

The ATX Power Connector

Pin	Signal	Wire Color
1	+3.3Vdc	Orange
2	+3.3Vdc	Orange
3	GND	Black
4	+5Vdc	Red
5	GND	Black
6	+5Vdc	Red
7	GND	Black
8	PWR-OK	Gray
9	+5Vdc VSB standby Voltage	Purple
10	+12Vdc	Yellow
11	+3.3Vdc	Orange {brown is 3.3Vdc sense]
12	-12Vdc	Blue
13	GND	Black
14	PS-ON	Green
15	GND	Black
16	GND	Black
17	GND	Black
18	-5Vdc	White
19	+5Vdc	Red
20	+5Vdc	Red

Table 5.5 ATX Power Connector Pins

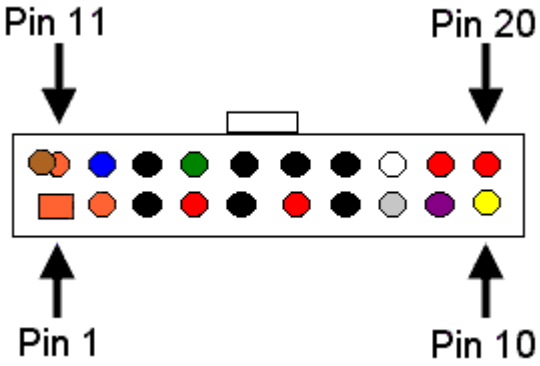


Figure 5.12 ATX Power Connector

The Sensor Circuit

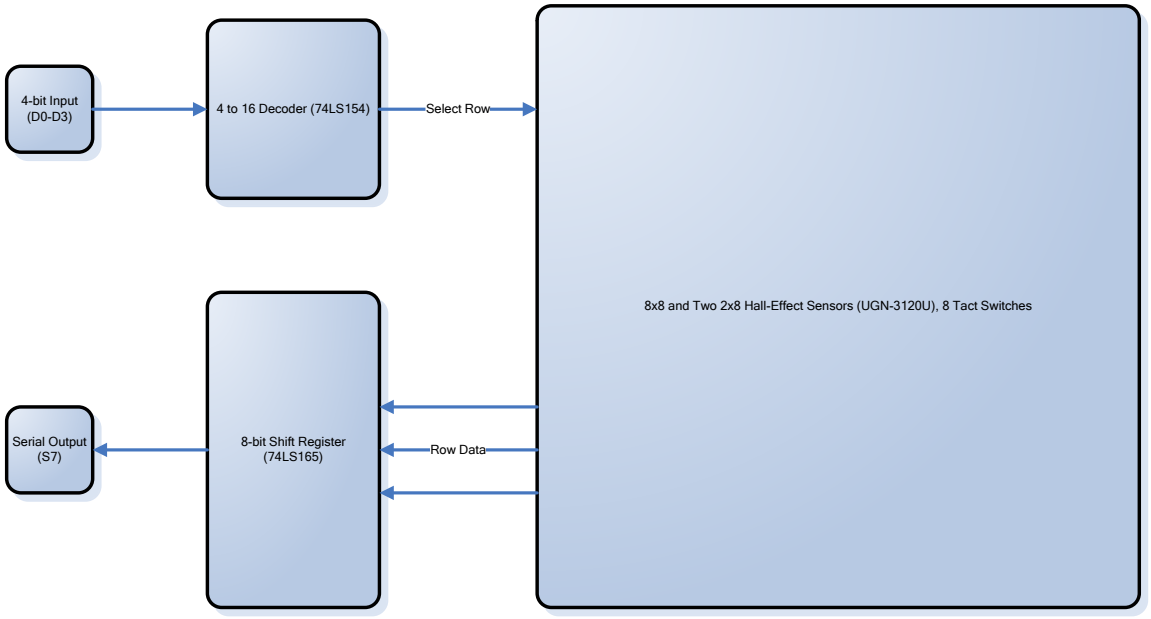


Figure 5.13 Sensor Board Circuit Diagram

Because of the large number of sensors used (96 hall-effect sensors, 8 tact switches and other I/O components) by the sensor board for detecting chess pieces and the parallel port has only a maximum 12 output lines, 8 for the Data Register, 4 for the Control Register and 5 input lines from the Status Register, the proponents used decoding techniques and parallel to serial conversion to be able read all data from the sensors. The sensor circuit may not be able read all data simultaneously, but because of the great processing speed of the computer (nanoseconds per instructions) it is able to gather all data seems like simultaneously.

The 4-bit data coming from D0 to D3 of the Data Register of the parallel port serve as the 4-bit input data for the decoder, the 4 to 16 decoder (74LS154)

then selects 1 of each 13 rows of sensors (12 for hall-effect sensors, 1 for tact switches) using 13 of the 16 decoded output. Only 8 sensors or 1 row can be activated at a time. The 8-bit data coming from the select row from the array of sensors is then fed to the 8-bit Parallel-In/Serial-Out Shift Register (74LS165). Applying a negative edge (C0) to the Load pin of the shift register, the 8-bit data will be loaded and by applying a positive edge (D4) to the clock, the parallel to serial conversion will starts. The serial output will enter S7 of the Status Register which will be process by the main application.

5.2 The Software

The K-S5 Application is the one responsible for controlling the hardware components. Even though the control instructions for the hardware components are complicated and too technical for the user to know, the windows application features a user-friendly and easy to use environment. With few click on a button, using toolbar and menus, the user can fully operate the machine easily. It also features voice recognition and voice synthesis in which the user can command the machine through speech. Basic commands can be easily recognized because the Voice Recognition technology can be optimized to recognize words and phrases from a predefined set of words. It also features Voice Synthesis in which the program responses to certain commands or in cases the application or machine encountered an error in the form of spoken words that can be

recognized by the user. With the help of these technologies, the project offers different ways in which the user can interact with the machine.

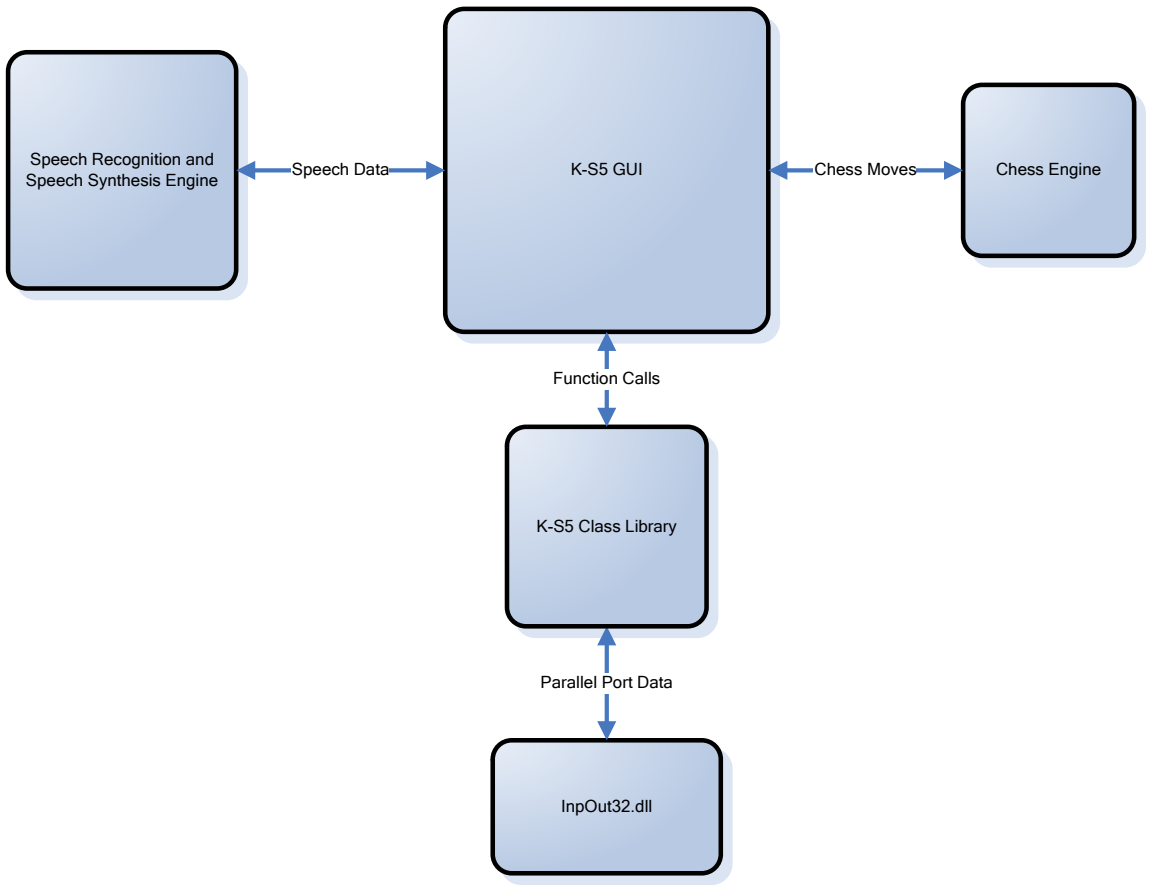


Figure 5.14 System block diagram of the K-S5 Application

The K-S5 Application is the user interface for the application, it is where the user can access commands, and gain control to the machine. To simplify the development and to have an organize way of controlling the machine through software instructions, the proponents developed a .NET class library in which all the basic instructions like controlling a certain axis of the actuator or controlling each electronic components can be access through function calls from the

library. In this case, it easy to develop other applications that controls the machine for different purpose not only to play chess. Authoring applications that controls the machine using different programming language supporting the .NET framework 2.0 is also possible because of the class library.

When the application calls a function from the library, the K-S5 class library communicates to the InpOut32.dll dynamic link library to be able to read and write data to the parallel port. Under the NT platform, application running at user mode is not allowed by the operating system to access the parallel port, at this situation, the proponents used the InpOut32.dll because it emulates a kernel mode application to gain access to the parallel port.

For the screenshots of the different software applications for the control and manipulation of the K-S5 see Appendices K, L and M.

5.2.1 Source Code

KS5.ChessEngine.dll
ChessEngine.cs

```
#region Using directives
using System;
using System.Diagnostics;
using System.IO;
using System.Threading;
#endregion

namespace KS5.ChessEngine
{
    public class Engine : IDisposable
    {
        private Process proc;
        private StreamWriter writer;
        private StreamReader reader;
```

```
private string line;
private string move;

public Engine()
{
    proc = new Process();

    proc.StartInfo.FileName = "ruffian/ruffian.exe";
    proc.StartInfo.WorkingDirectory = "ruffian";
    proc.StartInfo.Arguments = "";
    proc.StartInfo.UseShellExecute = false;
    proc.StartInfo.RedirectStandardError = true;
    proc.StartInfo.RedirectStandardInput = true;
    proc.StartInfo.RedirectStandardOutput = true;
    proc.StartInfo.CreateNoWindow = true;

    proc.Start();

    writer = proc.StandardInput;
    reader = proc.StandardOutput;
}

public string GetNextMove(string m)
{

    writer.WriteLine(m);
    writer.Flush();

    writer.WriteLine("stop");
    writer.Flush();

    writer.WriteLine("stop");
    writer.Flush();

    do
    {
        line = reader.ReadLine();

        if (line.StartsWith("bestmove"))
        {
            move = line.Substring(9, 4);
            writer.WriteLine(move);
            writer.Flush();
        }
    } while (!line.StartsWith("bestmove"));

    return move;
}

public string GetNextMove2()
{

    //writer.WriteLine(m);
    //writer.Flush();

    writer.WriteLine("stop");
    writer.Flush();
```

```
    writer.WriteLine("stop");
    writer.Flush();

    do
    {
        line = reader.ReadLine();

        if (line.StartsWith("bestmove"))
        {
            move = line.Substring(9, 4);
            writer.WriteLine(move);
            writer.Flush();
        }
    } while (!line.StartsWith("bestmove"));

    return move;
}

public void writeMove(string m)
{
    writer.WriteLine(m);
    writer.Flush();
}


public string GetNextMove()
{


    writer.WriteLine("stop");
    writer.Flush();

    do
    {

        line = reader.ReadLine();
        if (line.StartsWith("bestmove"))
        {
            move = line.Substring(9, 4);
            writer.WriteLine(move);
            writer.Flush();
        }

    } while (!line.StartsWith("bestmove"));

    return move;
}


public void initengine()
{
    writer.WriteLine("uci");
    writer.WriteLine("isready");
    writer.WriteLine("position startpos");
    writer.Flush();
}
```

```
        public void AbortMove()
        {
            writer.WriteLine("stop");
            writer.Flush();
        }


        public void Dispose()
        {
            writer.WriteLine("quit");
            writer.Flush();

            proc.Close();
        }


    }
}
```

KS5.ChessGame.dll

Game.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;

#endregion

namespace KS5.ChessGame
{
    public class Game
    {
        public bool whiteCastled, blackCastled, iswhitesTurn;
        public bool started, humanIsWhite;
        public Square[] mainboard, sbmachine, sbhuman

        public Game()
        {
            whiteCastled = false;
            blackCastled = false;
            iswhitesTurn = true;



            started = false;

            mainboard = new Square[64];
            sbhuman = new Square[16];
            sbmachine = new Square[16];

            for (int i = 0; i < 64; i++)
```

```
        {
            mainboard[i] = new Square();
        }

        for (int i = 0; i < 16; i++)
        {
            sbhuman[i] = new Square();
            sbmachine[i] = new Square();
        }


    }

    public void startGame(bool ishumanwhite)
    {
        string color1, color2;

        if (ishumanwhite)
        {
            humanIsWhite = true;
            color1 = "Black";
            color2 = "White";
        }
        else
        {
            humanIsWhite = false;
            color1 = "White";
            color2 = "Black";
        }

        for (int i = 0; i < 8; i++)
        {
            mainboard[6 * 8 + i].piece.Color = color1;
            mainboard[6 * 8 + i].piece.Name = "Pawn";
            mainboard[6 * 8 + i].isOccupied = true;
        }

        for (int i = 0; i < 8; i++)
        {
            mainboard[7 * 8 + i].piece.Color = color1;
            //mainboard[6 * 8 + i].name = "Pawn";
            mainboard[7 * 8 + i].isOccupied = true;
        }

        mainboard[56].piece.Name = "Rook";
        mainboard[57].piece.Name = "Knight";
        mainboard[58].piece.Name = "Bishop";

        if (humanIsWhite)
        {
            mainboard[59].piece.Name = "King";
            mainboard[60].piece.Name = "Queen";
        }
        else
        {
            mainboard[59].piece.Name = "Queen";
            mainboard[60].piece.Name = "King";
```

```
            }

            mainboard[61].piece.Name = "Bishop";
            mainboard[62].piece.Name = "Knight";
            mainboard[63].piece.Name = "Rook";


            /// next set
            ///
            for (int i = 0; i < 8; i++)
            {
                mainboard[8 + i].piece.Color = color2;
                mainboard[8 + i].piece.Name = "Pawn";
                mainboard[8 + i].isOccupied = true;
            }

            for (int i = 0; i < 8; i++)
            {
                mainboard[i].piece.Color = color2;
                //mainboard[6 * 8 + i].name = "Pawn";
                mainboard[i].isOccupied = true;
            }

            mainboard[0].piece.Name = "Rook";
            mainboard[1].piece.Name = "Knight";
            mainboard[2].piece.Name = "Bishop";

            if (humanIsWhite)
            {
                mainboard[3].piece.Name = "Queen";
                mainboard[4].piece.Name = "King";
            }
            else
            {
                mainboard[3].piece.Name = "King";
                mainboard[4].piece.Name = "Queen";
            }

            mainboard[5].piece.Name = "Bishop";
            mainboard[6].piece.Name = "Knight";
            mainboard[7].piece.Name = "Rook";

            started = true;
        }

        public void move(int from, int to)
        {
            mainboard[to].piece = mainboard[from].piece;
            mainboard[to].isOccupied = true;
            mainboard[from].isOccupied = false;
        }
    }
}
```

Piece.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;

#endregion

namespace KS5.ChessGame
{
    public class Piece
    {
        public string Color, Name;
        public bool enPassantMade;
        public int heightSteps;
        //public int index;

        public Piece()
        {
            Color = "";
            Name = "";
            enPassantMade = false;

            heightSteps = 0;
            //index = 0;
        }
    }
}
```

Square.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;

#endregion

namespace KS5.ChessGame
{
    public class Square
    {
        //public string name;
        //public int index;
        public bool isOccupied;
        public Piece piece;

        public Square()
        {
            piece = new Piece();
            isOccupied = false;
```

```
            //index = 0;
            //name = "";
        }
    }
}
```

KS5.Controller.dll
ChessParser.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;

#endregion

namespace KS5.Controller
{
    public class Chess
    {
        public Chess()
        {

        }

        public static int getMoveFromRow(string move)
        {
            return (8 - (int.Parse(move.Substring(1, 1))));
        }

        public static int getMoveToRow(string move)
        {
            return (8 - (int.Parse(move.Substring(3, 1))));
        }

        public static  int getMoveFromCol(string move)
        {
            return ((int)char.ConvertToUtf32(move, 0)-97);
        }

        public static int getMoveToCol(string move)
        {
            return ((int)char.ConvertToUtf32(move, 2)-97);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="str">example: "c1"</param>
        /// <returns>example: 0</returns>
        public static int getRow(string str)
        {
            int row = int.Parse(str.Substring(1, 1));
            if (char.IsNumber(str, 0))
            {
```

```
                if (row > 1)
                    return (row + 8);
                else
                {
                    if (row == 0)
                        return 9;
                    else
                        return 8;
                }
            }
            else
            {
                return (row - 1);
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="str">example: "d4"</param>
        /// <returns>example: 3</returns>
        public static int getCol(string str)
        {
            if (char.IsNumber(str, 0))
                return int.Parse(str.Substring(0, 1));
            else
                return getMoveFromCol(str);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="col">5</param>
        /// <param name="row">3</param>
        /// <returns>example: "f4"</returns>
        public static string getLocation(int col, int row)
        {
            if (row > 7)
            {
                if (row > 9)
                    row -= 8;
                else
                {
                    if (row == 8)
                        row = 1;
                    else
                        row = 0;
                }
                return (col.ToString() + (row).ToString());
            }
            else
            {
                return (char.ConvertFromUtf32(col + 97) + (row +
1).ToString());
            }
        }
```

```
    }
}
```

DataParser.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;

#endregion

namespace KS5.Controller
{
    public class Chess
    {
        public Chess()
        {

        }

        public static int getMoveFromRow(string move)
        {
            return (8 - (int.Parse(move.Substring(1, 1))));
        }

        public static int getMoveToRow(string move)
        {
            return (8 - (int.Parse(move.Substring(3, 1))));
        }

        public static  int getMoveFromCol(string move)
        {
            return ((int)char.ConvertToUtf32(move, 0)-97);
        }

        public static int getMoveToCol(string move)
        {
            return ((int)char.ConvertToUtf32(move, 2)-97);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="str">example: "c1"</param>
        /// <returns>example: 0</returns>
        public static int getRow(string str)
        {
            int row = int.Parse(str.Substring(1, 1));
            if (char.IsNumber(str, 0))
            {
                if (row > 1)
                    return (row + 8);
                else
```

```
                {
                    if (row == 0)
                        return 9;
                    else
                        return 8;
                }
            }
            else
            {
                return (row - 1);
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="str">example: "d4"</param>
        /// <returns>example: 3</returns>
        public static int getCol(string str)
        {
            if (char.IsNumber(str, 0))
                return int.Parse(str.Substring(0, 1));
            else
                return getMoveFromCol(str);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="col">5</param>
        /// <param name="row">3</param>
        /// <returns>example: "f4"</returns>
        public static string getLocation(int col, int row)
        {
            if (row > 7)
            {
                if (row > 9)
                    row -= 8;
                else
                {
                    if (row == 8)
                        row = 1;
                    else
                        row = 0;
                }
                return (col.ToString() + (row).ToString());
            }
            else
            {
                return (char.ConvertFromUtf32(col + 97) + (row +
1).ToString());
            }
        }

    }
}
```

KS5Properties.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;
using KS5.ChessGame;
#endregion

namespace KS5.Controller
{
    public class KS5Properties
    {
        const string nullByte = "00000000";


        public int d03, d5, d4, d6, d7;
        public int c0, c1, c2, c3;

        public int coilX, coilY, coilZ;
        public int EM, LASER, INDICATOR;
        public int currentAction;


        public int mbMarginX, mbMarginY;
        public int sbmMarginX, sbmMarginY;
        public int sbhMarginX, sbhMarginY;
        public int currentstepX, currentstepY, currentstepZ;
        public int maxStepX, maxStepY, maxStepZ;
        public int maxMBStepX, maxMBStepY, maxMBStepZ;
        public int EMMarginStep;
        public int pieceMarginStep;
        public int defaultZSteps;
        public int afterPickSteps, afterPickMarginSteps;
        public int sbAddSteps;

        public int ppawnHStep, pknightHStep, prookHStep, pbishopHStep,
pqueenHStep, pkingHStep;

        public int speedXLSB, speedMaxXMSB, speedYLSB, speedMaxYMSB,
speedZLSB, speedMaxZMSB;
        public int speedMinXMSB, speedMinYMSB, speedMinZMSB;
        public int accX, accY, accZ;

        public string move, move2;
        public bool success;
        public string errmsg;

        public string humanMove;

        public Game game;
        public KS5Properties()
        {
```

```
            //parallel port registers
            d03 = 0;
            d4 = 0;
            d5 = 0;
            d6 = 0;
            d7 = 0;

            c0 = 0;
            c1 = 2;
            c2 = 0;
            c3 = 0;


            //properties of x
            coilX = 0;              //initial state of coil
            speedXLSB = 100;        //speed
            speedMinXMSB = 45;      //slow speed
            speedMaxXMSB = 25;      //max speed2
            accX = 23;              //acceleration
            maxStepX = 1845;        //max range
            currentstepX = 0;


            //properties of y
            coilY = 0;
            speedYLSB = 100;
            speedMinYMSB = 45;
            speedMaxYMSB = 25;
            accY = 23;
            maxStepY = 1810;

            INDICATOR = 0;


            //properties of z
            coilZ = 0;
            speedZLSB = 100;
            speedMinZMSB = 35;
            speedMaxZMSB = 23;
            accZ = 15;
            maxStepZ = 1210;            //maximum steps z can go
            maxMBStepZ = 1250;
            defaultZSteps = 800;        //number of steps event with
piece without touching a piece
            afterPickSteps = 450;
            afterPickMarginSteps = 20;

            EM = 0;
            LASER = 0;



            //chessboard properties
            mbMarginX = 167;        //mainboard margin x
            mbMarginY = 623;        //mainboard margin y

            sbmMarginX = 165;         //sideboard margin x
```

```
            sbmMarginY = 160;          //sideboard margin y

            sbhMarginY = 620;
            sbhMarginX = 1540;

            currentstepX = 0;           //current step of x
            currentstepY = 0;           //current step of y
            currentstepZ = 0;           //current step of z

            maxMBStepX = 1226;
            maxMBStepY = 1215;


            //chess pieces properties
            EMMarginStep = 18;
            sbAddSteps = 50;

            game = new Game();
            humanMove = "";
        }
    }
}
```

KS5Controller.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Threading;
#endregion

namespace KS5.Controller
{
    public class KS5Controller
    {
        const string nullByte = "00000000";
        public KS5Properties my;

        public KS5Controller()
        {
            my = new KS5Properties();
        }


        #region Transmitting a bit to microcontrollers

        private void HighX()
        {
            delay(1);
            Out(0x37A,my.c1 + 4);
            XPGT();
```

```
}

private void LowX()
{
    delay(1);
    Out(0x37A, 0 + my.c1);
    XPGT();
}

private void HighY()
{
    delay(1);
    Out(0x37A,my.c1 + 8);
    YPGT();
}

private void LowY()
{
    delay(1);
    Out(0x37A, 0 + my.c1);
    YPGT();
}

private void HighZ()
{
    delay(1);
    Out(0x37A, 1 + my.c1);
    ZPGT();
}

private void LowZ()
{
    delay(1);
    Out(0x37A, 0 +my.c1);
    ZPGT();
}

#endregion



protected string getStatus()
{
    return DataParser.BuildByte(Inp(0x379));
}

private void delay(int msec)
{
    System.Threading.Thread.Sleep(msec);
}


/// <summary>
/// basic machine functions
/// </summary>
/// <param name="str3"></param>
#region Basic Machine Functions....
```

```
        public void setSpeedX(int lsb, int msb)
        {
            string s_lsb = DataParser.BuildByte(lsb);
            string s_msb = DataParser.BuildByte(msb);

            sendDataX(s_lsb, s_msb, "00" + my.coilX.ToString() +
"10001");
        }

        public void setMaxSpeedX()
        {
            setSpeedX(my.speedXLSB, my.speedMaxXMSB);
        }

        public void setMinSpeedX()
        {
            setSpeedX(my.speedXLSB, my.speedMinXMSB);
        }

        public void setSpeedY(int lsb, int msb)
        {
            string s_lsb = DataParser.BuildByte(lsb);
            string s_msb = DataParser.BuildByte(msb);

            sendDataY(s_lsb, s_msb, my.INDICATOR.ToString() + "0" +
my.coilY.ToString() + "10001");
        }

        public void setMaxSpeedY()
        {
            setSpeedY(my.speedYLSB, my.speedMaxYMSB);
        }

        public void setMinSpeedY()
        {
            setSpeedY(my.speedYLSB, my.speedMinYMSB);
        }



        public void setSpeedZ(int lsb, int msb)
        {
            string s_lsb = DataParser.BuildByte(lsb);
            string s_msb = DataParser.BuildByte(msb);

            sendDataZ(s_lsb, s_msb, my.EM.ToString() +
my.LASER.ToString() + my.coilZ.ToString() + "10001");
        }

        public void setMaxSpeedZ()
        {
            setSpeedZ(my.speedZLSB, my.speedMaxZMSB);
        }

        public void setMinSpeedZ()
```

```
        {
            setSpeedZ(my.speedZLSB, my.speedMinZMSB);
        }

        public bool isHomeX()
        {
            if (my.currentstepX != 0)
                return false;
            else
                return true;
        }

        public bool isHomeY()
        {
            if (my.currentstepY != 0)
                return false;
            else
                return true;
        }

        public bool isHomeZ()
        {
            if (my.currentstepZ != 0)
                return false;
            else
                return true;
        }

        public void stepMotorX(int steps, int dir, int coil)
        {
            string sword = DataParser.BuildWord(Convert.ToString(steps
+ 257, 2));
            string tmp_byte2 = sword.Substring(0, 8);
            string tmp_byte1 = sword.Substring(8);
            sendDataX(tmp_byte1, tmp_byte2, "000000" + dir.ToString() +
coil.ToString());

            my.coilX = coil;

            if (dir == 1)
            {
                steps *= -1;
            }

            my.currentstepX  += steps;
        }


        public void stepMotorX(int steps, int dir, int coil, int acc)
        {
            int reg_steps = steps - (acc * 2);

            if (reg_steps < 0)
            {
                return;
            }
```

```
            string s_acc = DataParser.BuildByte(acc + 1);
            string s_word =
DataParser.BuildWord(Convert.ToString(reg_steps + 257, 2));
            string tmp_byte2 = s_word.Substring(0, 8);
            string tmp_byte1 = s_word.Substring(8);
            string tmp_byte3;

            tmp_byte2 = s_acc.Substring(4) + tmp_byte2.Substring(4);
            tmp_byte3 = s_acc.Substring(3, 1) + "10000" +
dir.ToString() + coil.ToString();

            sendDataX(tmp_byte1, tmp_byte2, tmp_byte3);
            my.coilX = coil;

            if (dir == 1)
            {
                steps *= -1;
            }

            my.currentstepX += steps;
        }



        public void stepMotorY(int steps, int dir, int coil)
        {
            string sword = DataParser.BuildWord(Convert.ToString(steps
+ 257, 2));
            string tmp_byte2 = sword.Substring(0, 8);
            string tmp_byte1 = sword.Substring(8);
            sendDataY(tmp_byte1, tmp_byte2, "000000" + dir.ToString() +
coil.ToString());

            my.coilY = coil;

            if (dir == 1)
            {
                steps *= -1;
            }

            my.currentstepY += steps;
        }


        public void stepMotorY(int steps, int dir, int coil, int acc)
        {
            int reg_steps = steps - (acc * 2);

            if (reg_steps < 0)
            {
                return;
            }

            string s_acc = DataParser.BuildByte(acc + 1);
            string s_word =
DataParser.BuildWord(Convert.ToString(reg_steps + 257, 2));
            string tmp_byte2 = s_word.Substring(0, 8);
```

```
            string tmp_byte1 = s_word.Substring(8);
            string tmp_byte3;

            tmp_byte2 = s_acc.Substring(4) + tmp_byte2.Substring(4);
            tmp_byte3 = s_acc.Substring(3, 1) + "10000" +
dir.ToString() + coil.ToString();

            sendDataY(tmp_byte1, tmp_byte2, tmp_byte3);
            my.coilY = coil;

            if (dir == 1)
            {
                steps *= -1;
            }
            my.currentstepY += steps;
        }



        public void stepMotorZ(int steps, int dir, int coil)
        {
            string sword = DataParser.BuildWord(Convert.ToString(steps
+ 257, 2));
            string tmp_byte2 = sword.Substring(0, 8);
            string tmp_byte1 = sword.Substring(8);
            sendDataZ(tmp_byte1, tmp_byte2, "000000" + dir.ToString() +
coil.ToString());

            my.coilZ = coil;

            if (dir == 0)
            {
                steps *= -1;
            }

            my.currentstepZ += steps;
        }


        public void stepMotorZ(int steps, int dir, int coil, int acc)
        {
            int reg_steps = steps - (acc * 2);

            if (reg_steps < 0)
            {
                return;
            }

            string s_acc = DataParser.BuildByte(acc + 1);
            string s_word =
DataParser.BuildWord(Convert.ToString(reg_steps + 257, 2));
            string tmp_byte2 = s_word.Substring(0, 8);
            string tmp_byte1 = s_word.Substring(8);
            string tmp_byte3;

            tmp_byte2 = s_acc.Substring(4) + tmp_byte2.Substring(4);
```

```
            tmp_byte3 = s_acc.Substring(3, 1) + "10000" +
dir.ToString() + coil.ToString();

            sendDataZ(tmp_byte1, tmp_byte2, tmp_byte3);
            my.coilZ = coil;
        }



        /// <summary>
        /// determines the presence of chess piece from the picker
        /// the laser should be turned on before calling this function
        /// </summary>
        /// <returns>1 if present, 0 none</returns>
        public bool isPiecePresent()
        {
            bool ret_val;
            sendDataZ(nullByte, nullByte, "00100000");

            delay(1);

            if (getStatus().Substring(2, 1) == "1")
            {
                ret_val = true;
            }
            else
            {
                ret_val = false;
            }

            ZPGT();
            return ret_val;
        }


        /// <summary>
        /// GETS THE NUMBER OF STEPS AFTER THE LDR ARE BLOCKED
        /// </summary>
        /// <returns></returns>

        public int getStepLDR()
        {

            string byte1 = "";
            string byte2 = "";

            for (int i = 0; i < 8; i++)
            {
                ZPGT();
                delay(1);
                byte1 += getStatus().Substring(2, 1);
            }

            for (int i = 0; i < 8; i++)
            {
                ZPGT();
                delay(1);
```

```
            byte2 += getStatus().Substring(2, 1);
        }

        ZPGT();
        return Convert.ToInt32(byte2 + byte1, 2);
    }

    /// <summary>
    /// determines if pic x is online
    /// </summary>
    /// <returns></returns>
    public bool isPICXOnline()
    {
        if (getStatus().Substring(4, 1) == "1")
            return true;
        else
            return false;
    }


    public bool isPICYOnline()
    {
        if (getStatus().Substring(3, 1) == "1")
            return true;
        else
            return false;
    }


    public bool isPICZOnline()
    {
        if (getStatus().Substring(2, 1) == "1")
        {
            return true;
        }
        else
        {
            return false;
        }
    }




    public string GetPortByte(int row)
    {
        string tmpStr = "";
        //byte tmpInput;


        //select row
        my.d03 = row;
        OutD();

        //load parallel data
        my.c1 = 0;
        OutC();
```

```
        my.c1 = 2;
        OutC();

        for (int i = 1; i <= 8; i++)
        {
            if (getStatus().Substring(0, 1) == "1")
                tmpStr += "0";
            else
                tmpStr += "1";

            my.d4 = 0;
            OutD();

            my.d4 = 16;
            OutD();
        }
        return tmpStr;
    }

    public bool isPresent(int col, int row)
    {
        string str = GetPortByte(row);
        if (str.Substring(col, 1) == "1")
            return true;
        else
            return false;
    }

    public bool isPresent(string str)
    {
        int col = Chess.getCol(str), row = Chess.getRow(str);
        return isPresent(col, row);
        //return isPresent(Chess.getCol(str), Chess.getRow(str));
    }

    #region low-level subroutines

    [DllImport("InpOut32.dll", EntryPoint = "Inp32")]
    public static extern byte Inp(int PortAddress);

    [DllImport("InpOut32.dll", EntryPoint = "Out32")]
    public static extern void Out(int PortAddress, int Value);

    protected void OutC()
    {
        Out(0x37A, my.c0 + my.c1 + my.c2 + my.c3);
    }

    protected void OutD()
    {
        Out(0x378, my.d03 + my.d4 + my.d5 + my.d6 + my.d7);
    }

    #endregion


    #region positive edge on microcontrollers
```

```
        private void ZPGT()
        {
            my.d7 = 0;
            Out(0x378,my.d03 +my.d4 +my.d5 +my.d6 + 0);
            my.d7 = 128;
            Out(0x378,my.d03 +my.d4 +my.d5 +my.d6 + 128);
        }

        private void YPGT()
        {
            my.d6 = 0;
            Out(0x378,my.d03 +my.d4 +my.d5 +my.d6 +my.d7);
            my.d6 = 64;
            Out(0x378,my.d03 +my.d4 +my.d5 +my.d6 +my.d7);
        }

        private void XPGT()
        {
            my.d5 = 0;
            Out(0x378, my.d03 +my.d4 +my.d5 +my.d6 +my.d7);
            my.d5 = 32;
            Out(0x378, my.d03 +my.d4 +my.d5 +my.d6 +my.d7);
        }

        #endregion


        /// <summary>
        /// TURNS ON THE X MOTOR COIL
        /// </summary>
        public void coilXON()
        {
            sendDataX(nullByte, nullByte, "00110000");
            my.coilX = 1;
        }


        /// <summary>
        /// TURNS OFF THE X MOTOR COIL
        /// </summary>
        public void coilXOFF()
        {
            sendDataX(nullByte, nullByte, "00010000");
            my.coilX = 0;
        }


        /// <summary>
        /// TURNS ON Y MOTOR COIL
        /// </summary>
        public void coilYON()
        {
            sendDataY(nullByte, nullByte, my.INDICATOR.ToString() +
"0110000");
            my.coilY  = 1;
        }
```

```
        /// <summary>
        /// TURNS OFF Y MOTOR COIL
        /// </summary>
        public void coilYOFF()
        {
            sendDataY(nullByte, nullByte,my.INDICATOR.ToString() +
"0010000");
            my.coilY = 0;
        }


        /// <summary>
        /// TURNS ON MOTOR Z COIL
        /// </summary>
        public void coilZON()
        {
            sendDataZ(nullByte, nullByte, my.EM.ToString() +
my.LASER.ToString() + "110000");
            my.coilZ = 1;
        }


        /// <summary>
        /// TURNS OFF MOTOR Z COIL
        /// </summary>
        public void coilZOFF()
        {
            sendDataZ(nullByte, nullByte, my.EM.ToString() +
my.LASER.ToString() + "010000");
            my.coilZ = 0;
        }



        /// <summary>
        /// STEPS MOTOR X TO HOME POSITION
        /// </summary>
        /// <param name="coil"></param>
        public void stepHomeX(int coil)
        {
            sendDataX(nullByte, nullByte, "0000011" + coil.ToString());
            my.coilX = coil;
            my.currentstepX = 0;
        }


        public void stepHomeX(int coil, int acc)
        {
            string s_acc = DataParser.BuildByte(acc + 1);
            string tmp_byte2;
            string tmp_byte3;

            tmp_byte2 = s_acc.Substring(4) + "0000";
            tmp_byte3 = s_acc.Substring(3, 1) + "100011" +
coil.ToString();
```

```
            sendDataX(nullByte, tmp_byte2, tmp_byte3);
            my.coilX = coil;
            my.currentstepX = 0;
        }


        /// <summary>
        /// STEPS MOTOR Y TO HOME POSITION
        /// </summary>
        /// <param name="coil"></param>
        public void stepHomeY(int coil)
        {
            sendDataY(nullByte, nullByte, "0000011" + coil.ToString());
            my.coilY = coil;
            my.currentstepY = 0;
        }

        public void stepHomeY(int coil, int acc)
        {
            string s_acc = DataParser.BuildByte(acc + 1);
            string tmp_byte2;
            string tmp_byte3;

            tmp_byte2 = s_acc.Substring(4) + "0000";
            tmp_byte3 = s_acc.Substring(3, 1) + "100011" +
coil.ToString();

            sendDataY(nullByte, tmp_byte2, tmp_byte3);
            my.coilY = coil;
            my.currentstepY = 0;
        }

        /// <summary>
        /// STEPS MOTOR Z TO HOME POSITION
        /// </summary>
        /// <param name="coil"></param>
        public void stepHomeZ(int coil)
        {
            sendDataZ(nullByte, nullByte, "0000011" + coil.ToString());
            my.coilZ = coil;
        }


        public void stepHomeZ(int coil, int acc)
        {
            string s_acc = DataParser.BuildByte(acc + 1);
            string tmp_byte2;
            string tmp_byte3;

            tmp_byte2 = s_acc.Substring(4) + "0000";
            tmp_byte3 = s_acc.Substring(3, 1) + "100011" +
coil.ToString();

            sendDataZ(nullByte, tmp_byte2, tmp_byte3);
            my.coilZ = coil;
        }
```

```
        /// <summary>
        /// STEPS MOTOR Z TIL THE LDR ARE BLOCKED, WITHOUT STEP
REPORTING
        /// </summary>
        /// <param name="steps"></param>
        /// <param name="coil"></param>
        public void stepLDRNoReport(int steps, int coil)
        {
            //setCoilZ(1);
            string sword = DataParser.BuildWord(Convert.ToString(steps
+ 257, 2));
            string tmp_byte2 = sword.Substring(0, 8);
            string tmp_byte1 = sword.Substring(8);
            sendDataZ(tmp_byte1, tmp_byte2, "0000110" +
coil.ToString());

            my.coilZ = coil;
        }


        /// <summary>
        /// STEPS MOTOR Z UNTIL THE LDR ARE BLOCKED, WITH STEP
REPORTING
        /// </summary>
        /// <param name="steps"></param>
        /// <param name="coil"></param>
        public void stepLDR(int steps, int coil)
        {
            string sword = DataParser.BuildWord(Convert.ToString(steps
+ 257, 2));
            string tmp_byte2 = sword.Substring(0, 8);
            string tmp_byte1 = sword.Substring(8);
            sendDataZ(tmp_byte1, tmp_byte2, "0000110" +
coil.ToString());

            my.coilZ = coil;
        }

        /// <summary>
        /// TURNS ON THE LED INDICATOR
        /// </summary>
        public void IndicatorON()
        {
            sendDataY(nullByte, nullByte, "10" + my.coilY.ToString() +
"10000");
            my.INDICATOR  = 1;
        }


        /// <summary>
        /// TURNS OFF THE LED INDICATOR
        /// </summary>
        public void IndicatorOFF()
        {
            sendDataY(nullByte, nullByte, "00" + my.coilY.ToString() +
"10000");
```

```
            my.INDICATOR  = 0;
        }


        /// <summary>
        /// TURNS ON THE ELECTROMAGNET
        /// </summary>
        public void EMON()
        {
            sendDataZ(nullByte, nullByte, "1" + my.LASER.ToString() +
my.coilZ.ToString() + "10000");
            my.EM = 1;
        }


        /// <summary>
        /// TURNS OFF THE ELECTROMAGNET
        /// </summary>
        public void EMOFF()
        {
            sendDataZ(nullByte, nullByte, "0" + my.LASER.ToString() +
my.coilZ.ToString() + "10000");
            my.EM = 0;
        }


        public void setPICYP(int ind, int coil)
        {
            sendDataY(nullByte, nullByte, ind.ToString() + "0" +
coil.ToString() + "10000");
            my.INDICATOR = ind;
            my.coilY = coil;
        }

        /// <summary>
        /// CONTROLS THE PERIPHERALS OF PIC Z (EM, LASER, COIL) AT ONCE
        /// </summary>
        /// <param name="em"></param>
        /// <param name="laser"></param>
        /// <param name="coil"></param>
        public void setPICZP(int em, int laser, int coil)
        {
            sendDataZ(nullByte, nullByte, em.ToString() +
laser.ToString() + coil.ToString() + "10000");
            my.EM = em;
            my.LASER = laser;
            my.coilZ = coil;
        }


        /// <summary>
        /// SENDS 3 BYTES OF DATA TO PICX
        /// </summary>
        /// <param name="str1"></param>
        /// <param name="str2"></param>
        /// <param name="str3"></param>
        private void sendDataZ(string str1, string str2, string str3)
```

```
{
    for (int i = 0; i <= 7; i++)
    {
        if (str1.Substring(i, 1) == "1")
            HighZ();
        else
            LowZ();
    }

    for (int i = 0; i <= 7; i++)
    {
        if (str2.Substring(i, 1) == "1")
            HighZ();
        else
            LowZ();
    }

    for (int i = 0; i <= 7; i++)
    {
        if (str3.Substring(i, 1) == "1")
            HighZ();
        else
            LowZ();
    }

}


/// <summary>
/// SENDS 3 BYTES OF DATA TO PICY
/// </summary>
/// <param name="str1"></param>
/// <param name="str2"></param>
/// <param name="str3"></param>
private void sendDataY(string str1, string str2, string str3)
{

    for (int i = 0; i <= str1.Length - 1; i++)
    {
        if (str1.Substring(i, 1) == "1")
            HighY();
        else
            LowY();
    }


    for (int i = 0; i <= str2.Length - 1; i++)
    {
        if (str2.Substring(i, 1) == "1")
            HighY();
        else
            LowY();
    }

    for (int i = 0; i <= str3.Length - 1; i++)
    {
        if (str3.Substring(i, 1) == "1")
```

```
            HighY();
        else
            LowY();
    }
}


/// <summary>
/// SENDS 3 BYTES OF DATA TO PICX
/// </summary>
/// <param name="str1"></param>
/// <param name="str2"></param>
/// <param name="str3"></param>
private void sendDataX(string str1, string str2, string str3)
{

    for (int i = 0; i <= str1.Length - 1; i++)
    {
        if (str1.Substring(i, 1) == "1")
            HighX();
        else
            LowX();
    }

    for (int i = 0; i <= str2.Length - 1; i++)
    {
        if (str2.Substring(i, 1) == "1")
            HighX();
        else
            LowX();
    }

    for (int i = 0; i <= str3.Length - 1; i++)
    {
        if (str3.Substring(i, 1) == "1")
            HighX();
        else
            LowX();
    }
}


/// <summary>
/// TURN ON SYSTEM POWER
/// </summary>
public void powerON()
{
    sendDataY(nullByte, nullByte, "00010000");
}


/// <summary>
/// TURN OFF SYSTEM POWER
/// </summary>
public void powerOFF()
{
```

```
            sendDataY(nullByte, nullByte, "01010000");
        }


        public void laserON()
        {
            sendDataZ(nullByte, nullByte, my.EM.ToString() + "1" +
my.coilZ.ToString() + "10000");
            my.LASER = 1;
        }


        public void laserOFF()
        {
            sendDataZ(nullByte, nullByte, my.EM.ToString() + "0" +
my.coilZ.ToString() + "10000");
            my.LASER = 0;
        }

        /// <summary>
        /// determines if power is online
        /// </summary>
        /// <returns></returns>
        public bool isPowerOK()
        {
            if (DataParser.getBit(getStatus(), 6) == "1")
                return true;
            else
                return false;
        }


        #endregion
    }
}
```

KS5.GUI.exe

GUI.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using System.Threading;
using KS5.Controller;
using KS5.ChessGame;
using KS5.ChessEngine;
#endregion

namespace KS5.Main
{
```

```
    partial class GUI : Form
    {
        const int initKS5 = 1, movePiece = 2, pointSquare = 3,
getHumanMove = 4, capturePiece = 5;
        int timeCounter = 0;

        bool isKS5Init, KS5_busy;
        int current_action;
        KS5Controller KS5_machine;
        ChessEngine.Engine engine;
        string machineMove;
        string lastMove;

        public GUI()
        {
            InitializeComponent();

            KS5_machine = new KS5Controller();

            current_action = 0;
            isKS5Init = false;
            KS5_busy = false;
        }

        private void GUI_Load(object sender, EventArgs e)
        {
            if (!isKS5Init)
            {
                KS5_machine.my.currentAction = initKS5;
                bWKS5Handler.RunWorkerAsync(KS5_machine.my);
            }

            //MessageBox.Show(Chess.getLocation(0, 9));
        }

        private void bWKS5Handler_DoWork(object sender, DoWorkEventArgs
e)
        {
            // This method will run on a thread other than the UI
thread.
            // Be sure not to manipulate any Windows Forms controls
created
            // on the UI thread from this method.

            KS5Controller KS5C = new KS5Controller();
            KS5C.my = (KS5Properties)e.Argument;
            int xsteps = 0, ysteps = 0, xsteps2 = 0, ysteps2 = 0,
xstepsdest = 0, ystepsdest = 0;
            int letStepZ = 0, sbmarginStepZ = 0;
            int dirx = 0, diry = 0;
            int colFrom, colTo, rowFrom, rowTo;
            int captureStepX, captureStepY;

            if (KS5C.my.currentAction == initKS5)
            {
                if (KS5C.isPICYOnline())
                {
```

```
                    //initialize power
                    Thread.Sleep(250);
                    KS5C.powerON();

                    Thread.Sleep(2500);
                    KS5C.powerOFF();
                    Thread.Sleep(1000);

                    //if(

                    KS5C.powerON();
                    Thread.Sleep(1000);

                    //turn off all peripherals
                    KS5C.EMOFF();
                    KS5C.IndicatorOFF();
                    KS5C.laserOFF();

                    //set speed
                    KS5C.setMaxSpeedY();
                    KS5C.setMaxSpeedZ();
                    KS5C.setMaxSpeedX();

                    //step all motors to home position
                    KS5C.stepHomeZ(0, KS5C.my.accZ);
                    Thread.Sleep(500);
                    KS5C.stepHomeX(0, KS5C.my.accX);
                    KS5C.stepHomeY(0, KS5C.my.accY);

                    do
                    {
                        Thread.Sleep(100);
                    } while (!KS5C.isPICXOnline() ||
!KS5C.isPICYOnline() || !KS5C.isPICZOnline());

                }
                else
                {
                    //no power!!!
                }
            }
            else if (KS5C.my.currentAction == movePiece)
            {
                if (KS5C.isPresent(KS5C.my.move.Substring(2, 2)) ||
!KS5C.isPresent(KS5C.my.move.Substring(0, 2)))
                {
                    //MessageBox.Show("error!!!");
                }


                //initialize to starting position
                KS5C.setMaxSpeedX();
                KS5C.setMaxSpeedY();
                KS5C.setMaxSpeedZ();
```

```
                /// <summary>
                /// sends all motor first to home position
                /// </summary>
                KS5C.stepHomeZ(1, KS5C.my.accZ);    //step home z
                Thread.Sleep(500);                  //little delay for
not accidentally..
                KS5C.stepHomeX(1, KS5C.my.accX);
                KS5C.stepHomeY(1, KS5C.my.accY);    //step home y

                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICXOnline() || !KS5C.isPICYOnline()
|| !KS5C.isPICZOnline());
                //KS5C.setMaxSpeedX();
                ////////////////////////////////////////////////>>


                /// <summary>
                /// compute the steps required for pointing the piece
to get
                /// </summary>
                ///


                if (KS5C.isPresent(KS5C.my.move.Substring(2, 2)))
                {
                    sbmarginStepZ = KS5C.my.sbAddSteps;
                    ///if its a move from the mainboard
                    colFrom =
Chess.getMoveFromCol(KS5C.my.move.Substring(2,2));
                    rowFrom =
Chess.getMoveFromRow(KS5C.my.move.Substring(2,2));


                    ///first set of steps for locating the piece
                    ///
                    captureStepX = KS5C.my.mbMarginX +
(int)((KS5C.my.maxMBStepX / 8) * rowFrom);
                    captureStepY = KS5C.my.mbMarginY +
(int)((KS5C.my.maxMBStepY / 8) * colFrom);

                    KS5C.stepMotorX(xsteps, 0, 1, KS5C.my.accX);
                    KS5C.stepMotorY(ysteps, 0, 1, KS5C.my.accY);
                    KS5C.stepMotorZ(KS5C.my.defaultZSteps, 0, 1); //,
KS5C.my.accZ);
                    //KS5C.stepMotorZ(KS5C.my.maxStepZ, 0, 1,
KS5C.my.accZ);

                    do
                    {
                        Thread.Sleep(100);
                    } while (!KS5C.isPICXOnline() ||
!KS5C.isPICYOnline() || !KS5C.isPICZOnline());

                    if (KS5C.my.game.started)
                    {
```

```
                        for (int j = 0; j < 2; j++)
                        {
                            for (int i = 0; i < 8; i++)
                            {
                                if (!KS5C.my.game.sbmachine[j * 8 +
i].isOccupied)
                                {
                                    KS5C.my.game.sbmachine[j * 8 + i] =
KS5C.my.game.mainboard[Chess.getRow(KS5C.my.move.Substring(2, 2)) * 8 +
Chess.getCol(KS5C.my.move.Substring(2, 2))];
                                }
                            }
                        }
                    }
                }


                if (char.IsNumber(KS5C.my.move, 0))
                {
                    sbmarginStepZ = KS5C.my.sbAddSteps;

                    if (int.Parse(KS5C.my.move.Substring(1, 1)) > 1)
                    {
                        //sb machine
                        xsteps = KS5C.my.sbmMarginX +
(int)((KS5C.my.maxMBStepX / 8) * (int.Parse(KS5C.my.move.Substring(0,
1))));
                        ysteps = KS5C.my.sbmMarginY +
(int)((KS5C.my.maxMBStepY / 8) * (int.Parse(KS5C.my.move.Substring(1,
1)) - 2));
                    }
                    else
                    {
                        //sb human
                        xsteps = KS5C.my.sbhMarginX +
(int)((KS5C.my.maxMBStepX / 8) * (int.Parse(KS5C.my.move.Substring(1,
1))));
                        ysteps = KS5C.my.sbhMarginY +
(int)((KS5C.my.maxMBStepY / 8) * (int.Parse(KS5C.my.move.Substring(0,
1))));
                    }
                }
                else
                {
                    sbmarginStepZ = 0;
                    ///if its a move from the mainboard
                    colFrom = Chess.getMoveFromCol(KS5C.my.move);
                    rowFrom = Chess.getMoveFromRow(KS5C.my.move);


                    ///first set of steps for locating the piece
                    ///
                    xsteps = KS5C.my.mbMarginX +
(int)((KS5C.my.maxMBStepX / 8) * rowFrom);
                    ysteps = KS5C.my.mbMarginY +
(int)((KS5C.my.maxMBStepY / 8) * colFrom);
                }
```

```
                //KS5C.stepMotorX(KS5C.my.mbMarginX, 0, 1,
KS5C.my.accX);
                KS5C.stepMotorX(xsteps, 0, 1, KS5C.my.accX);
                KS5C.stepMotorY(ysteps, 0, 1, KS5C.my.accY);
                KS5C.stepMotorZ(KS5C.my.defaultZSteps, 0, 1); //,
KS5C.my.accZ);
                //KS5C.stepMotorZ(KS5C.my.maxStepZ, 0, 1,
KS5C.my.accZ);

                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICXOnline() || !KS5C.isPICYOnline()
|| !KS5C.isPICZOnline());


                ////////////////////////////////////////////////<<
                /// start picking the piece
                KS5C.laserON();
                KS5C.stepLDR(KS5C.my.maxMBStepZ - KS5C.my.defaultZSteps
+ sbmarginStepZ, 1);

                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICZOnline());

                KS5C.getStepLDR();

                /// turn on em and step a little for it to ...
                ///
                KS5C.EMON();
                KS5C.stepMotorZ(KS5C.my.EMMarginStep, 0, 1);

                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICZOnline());

                //KS5C.stepHomeZ(1);
                //KS5C.laserOFF();

                ///after picking the piece, step the motor up few steps
                ///
                KS5C.stepMotorZ(KS5C.my.afterPickSteps, 1, 1,
KS5C.my.accZ);

                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICZOnline());
                ////////////////////////////////////////////////>>
```

```
                if (char.IsNumber(KS5C.my.move, 2))
                {
                    //sbmarginStepZ = KS5C.my.sbAddSteps;

                    if (int.Parse(KS5C.my.move.Substring(3, 1)) > 1)
                    {
                        //sb machine
                        xsteps2 = KS5C.my.sbmMarginX +
(int)((KS5C.my.maxMBStepX / 8) * (int.Parse(KS5C.my.move.Substring(2,
1))));
                        ysteps2 = KS5C.my.sbmMarginY +
(int)((KS5C.my.maxMBStepY / 8) * (int.Parse(KS5C.my.move.Substring(3,
1)) - 2));
                    }
                    else
                    {
                        //sb human
                        xsteps2 = KS5C.my.sbhMarginX +
(int)((KS5C.my.maxMBStepX / 8) * (int.Parse(KS5C.my.move.Substring(3,
1))));
                        ysteps2 = KS5C.my.sbhMarginY +
(int)((KS5C.my.maxMBStepY / 8) * (int.Parse(KS5C.my.move.Substring(2,
1))));
                    }
                }
                else
                {
                    //sbmarginStepZ = 0;
                    ///if its a move from the mainboard

                    colTo = Chess.getMoveToCol(KS5C.my.move);
                    rowTo = Chess.getMoveToRow(KS5C.my.move);

                    ///second set of steps for locating the destination
of the piece
                    ///
                    xsteps2 = KS5C.my.mbMarginX +
(int)((KS5C.my.maxMBStepX / 8) * rowTo);
                    ysteps2 = KS5C.my.mbMarginY +
(int)((KS5C.my.maxMBStepY / 8) * colTo);
                }

                if (KS5C.my.currentstepX > xsteps2)
                {
                    dirx = 1;
                    xstepsdest = KS5C.my.currentstepX - xsteps2;
                    //KS5C.stepMotorX(xstepsdest, dirx, 1,
KS5C.my.accX);
                }

                if (KS5C.my.currentstepX < xsteps2)
                {
                    dirx = 0;
                    xstepsdest = xsteps2 - KS5C.my.currentstepX;
                    //KS5C.stepMotorX(xstepsdest, dirx, 1,
KS5C.my.accX);
```

```
                }

                if (KS5C.my.currentstepY > ysteps2)
                {
                    diry = 1;
                    ystepsdest = KS5C.my.currentstepY - ysteps2;
                }

                if (KS5C.my.currentstepY < ysteps2)
                {
                    diry = 0;
                    ystepsdest = ysteps2 - KS5C.my.currentstepY;
                }


                ///if dest steps for x are too little, use min speed
                ///
                if (KS5C.my.currentstepX != xsteps2)
                {
                    if (xstepsdest < (int)(KS5C.my.maxMBStepX / 8))
                    {
                        KS5C.setMinSpeedX();
                        KS5C.stepMotorX(xstepsdest, dirx, 1);
                    }
                    else
                    {
                        KS5C.stepMotorX(xstepsdest, dirx, 1,
KS5C.my.accX);
                    }
                }


                /// if dest steps for y are too little, use min speed
                ///
                if (KS5C.my.currentstepY != ysteps2)
                {
                    if (ystepsdest < (int)(KS5C.my.maxMBStepY / 8))
                    {
                        KS5C.setMinSpeedY();
                        KS5C.stepMotorY(ystepsdest, diry, 1);
                    }
                    else
                    {
                        KS5C.stepMotorY(ystepsdest, diry, 1,
KS5C.my.accY);
                    }
                }


                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICXOnline() || !KS5C.isPICYOnline());

                KS5C.setMaxSpeedX();
                KS5C.setMaxSpeedY();
```

```
                //KS5C.stepMotorZ(435, 0, 1); //, KS5C.my.accZ);
                letStepZ = KS5C.my.afterPickSteps -
KS5C.my.afterPickMarginSteps;

                if (char.IsNumber(KS5C.my.move, 0) &&
!char.IsNumber(KS5C.my.move, 2))
                    letStepZ -= KS5C.my.sbAddSteps;
                if (!char.IsNumber(KS5C.my.move, 0) &&
char.IsNumber(KS5C.my.move, 2))
                    letStepZ += KS5C.my.sbAddSteps;

                KS5C.stepMotorZ(letStepZ, 0, 1); //, KS5C.my.accZ);
                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICZOnline());

                ///put down the chess piece by turning off the em
                ///
                Thread.Sleep(250);
                KS5C.EMOFF();

                Thread.Sleep(250);
                int alignCount = 0;
                while (!KS5C.isPresent(KS5C.my.move.Substring(2, 2)) &&
(alignCount < 7))
                {
                    KS5C.stepMotorZ(300, 1, 1, KS5C.my.accZ);
                    do
                    {
                        Thread.Sleep(100);
                    } while (!KS5C.isPICZOnline());

                    KS5C.stepLDR(KS5C.my.maxMBStepZ, 1);
                    do
                    {
                        Thread.Sleep(100);
                    } while (!KS5C.isPICZOnline());

                    KS5C.getStepLDR();
                    KS5C.EMON();
                    KS5C.stepMotorZ(KS5C.my.EMMarginStep, 0, 1);

                    do
                    {
                        Thread.Sleep(100);
                    } while (!KS5C.isPICZOnline());

                    KS5C.stepMotorZ(300, 1, 1, KS5C.my.accZ);
                    do
                    {
                        Thread.Sleep(100);
                    } while (!KS5C.isPICZOnline());

                    KS5C.stepMotorZ(300 - KS5C.my.afterPickMarginSteps
- (10 * alignCount), 0, 1);
```

```
                    do
                    {
                        Thread.Sleep(100);
                    } while (!KS5C.isPICZOnline());

                    Thread.Sleep(100);
                    KS5C.EMOFF();
                    Thread.Sleep(250);
                    alignCount++;
                }


                /// the piece has been moved to the proper position
                /// step motor z up before stepping all motors to home
position
                KS5C.stepMotorZ(300, 1, 1, KS5C.my.accZ);

                //Thread.Sleep(1000);
                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICZOnline());

                KS5C.laserOFF();

                KS5C.stepHomeZ(0, KS5C.my.accZ);
                KS5C.stepHomeX(0, KS5C.my.accX);
                KS5C.stepHomeY(0, KS5C.my.accY);


                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICXOnline() || !KS5C.isPICYOnline());



            }

            else if (KS5C.my.currentAction == pointSquare)
            {
                //initialize to starting position
                KS5C.setMaxSpeedX();
                KS5C.setMaxSpeedY();
                KS5C.setMaxSpeedZ();



                /// <summary>
                /// sends all motor first to home position
                /// </summary>
                KS5C.stepHomeZ(1, KS5C.my.accZ);    //step home z
                Thread.Sleep(500);                  //little delay for
not accidentally..
                KS5C.stepHomeX(1, KS5C.my.accX);
                KS5C.stepHomeY(1, KS5C.my.accY);    //step home y
```

```
                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICXOnline() || !KS5C.isPICYOnline()
|| !KS5C.isPICZOnline());
                //KS5C.setMaxSpeedX();
                ////////////////////////////////////////////////>>


                /// <summary>
                /// compute the steps required for pointing the piece
to get
                /// </summary>
                ///

                if (char.IsNumber(KS5C.my.move, 0))
                {
                    sbmarginStepZ = KS5C.my.sbAddSteps;

                    if (int.Parse(KS5C.my.move.Substring(1, 1)) > 1)
                    {
                        //sb machine
                        xsteps = KS5C.my.sbmMarginX +
(int)((KS5C.my.maxMBStepX / 8) * (int.Parse(KS5C.my.move.Substring(0,
1))));
                        ysteps = KS5C.my.sbmMarginY +
(int)((KS5C.my.maxMBStepY / 8) * (int.Parse(KS5C.my.move.Substring(1,
1)) - 2));
                    }
                    else
                    {
                        //sb human
                        xsteps = KS5C.my.sbhMarginX +
(int)((KS5C.my.maxMBStepX / 8) * (int.Parse(KS5C.my.move.Substring(1,
1))));
                        ysteps = KS5C.my.sbhMarginY +
(int)((KS5C.my.maxMBStepY / 8) * (int.Parse(KS5C.my.move.Substring(0,
1))));
                    }
                }
                else
                {
                    sbmarginStepZ = 0;
                    ///if its a move from the mainboard
                    colFrom = Chess.getMoveFromCol(KS5C.my.move);
                    rowFrom = Chess.getMoveFromRow(KS5C.my.move);


                    ///first set of steps for locating the piece
                    ///
                    xsteps = KS5C.my.mbMarginX +
(int)((KS5C.my.maxMBStepX / 8) * rowFrom);
                    ysteps = KS5C.my.mbMarginY +
(int)((KS5C.my.maxMBStepY / 8) * colFrom);
                }
```

```
                //KS5C.stepMotorX(KS5C.my.mbMarginX, 0, 1,
KS5C.my.accX);
                KS5C.stepMotorX(xsteps, 0, 1, KS5C.my.accX);
                KS5C.stepMotorY(ysteps, 0, 1, KS5C.my.accY);
                KS5C.stepMotorZ(KS5C.my.defaultZSteps, 0, 1); //,
KS5C.my.accZ);
                //KS5C.stepMotorZ(KS5C.my.maxStepZ, 0, 1,
KS5C.my.accZ);

                do
                {
                    Thread.Sleep(100);
                } while (!KS5C.isPICXOnline() || !KS5C.isPICYOnline()
|| !KS5C.isPICZOnline());
            }
            else if (KS5C.my.currentAction == getHumanMove)
            {
                string[] rows = new string[8];
                bool validMoveMade = false;
                string moveFrom = "", moveTo = "", tmpMove = "";
                int fromCol = 0, fromRow = 0, toCol = 0, toRow = 0;
                string newRow = "", oldRow = "";

                for (int i = 0; i < 8; i++)
                {
                    rows[i] = KS5C.GetPortByte(i);
                }

                do
                {
                    Thread.Sleep(200);

                    ///main loop
                    ///
                    for (int i = 0; i < 8; i++)
                    {
                        newRow = KS5C.GetPortByte(i);
                        oldRow = rows[i];
                        if (oldRow != newRow && !validMoveMade)
                        {

                            //validMoveMade = true;
                            ///checking the changed row
                            ///
                            for (int j = 0; j < 8; j++)
                            {
                                /*
                                if(rows[i].Substring(j,1) !=
newRow.Substring(j,1))
                                {
                                    tmpMove = Chess.getLocation(j, i);
                                    KS5C.my.humanMove = moveFrom;
                                }
                                */

                                ///determining the column
                                ///
```

```
                                if (oldRow.Substring(j, 1) == "1" &&
newRow.Substring(j, 1) == "0")
                                {
                                    KS5C.IndicatorON();
                                    tmpMove = Chess.getLocation(j, i);

                                    ///update the row
                                    ///
                                    rows[i] = newRow;

                                    if (KS5C.my.game.iswhitesTurn)
                                    {
                                        if (KS5C.my.game.mainboard[i *
8 + j].piece.Color == "White")

                                        {
                                            moveFrom = tmpMove;
                                            fromCol = j;
                                            fromRow = i;
                                            //KS5C.my.humanMove =
"Source Move: " + moveFrom;
                                        }
                                        else
                                        {
                                            moveTo = tmpMove;
                                            toCol = j;
                                            toRow = i;
                                            //KS5C.my.humanMove =
"Destination Move: " + moveTo;
                                        }
                                    }
                                    else
                                    {
                                        if (KS5C.my.game.mainboard[i *
8 + j].piece.Color == "Black")

                                        {
                                            moveFrom = tmpMove;
                                            fromCol = j;
                                            fromRow = i;
                                        }
                                        else
                                        {
                                            moveTo = tmpMove;
                                            toCol = j;
                                            toRow = i;
                                        }
                                    }
                                }
                                else if (oldRow.Substring(j, 1) == "0"
&& newRow.Substring(j, 1) == "1")
                                {
                                    //final move
                                    KS5C.IndicatorOFF();

                                    tmpMove = Chess.getLocation(j, i);
                                    moveTo = tmpMove;
                                    KS5C.my.game.move(fromRow * 8 +
fromCol, i * 8 + j);
```

```
                                    KS5C.my.humanMove = moveFrom +
moveTo;
                                    validMoveMade = true;
                                    j = 8;
                                    i = 8;
                                }
                            }

                            //i = 8;
                        }
                    }
                } while (!validMoveMade);

            }

            e.Result = KS5C.my;

        }

        private void bWKS5Handler_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            KS5_machine.my = (KS5Properties)e.Result;

            if (KS5_machine.my.currentAction == getHumanMove)
            {
                //MessageBox.Show(KS5_machine.my.humanMove);
                if (KS5_machine.my.game.humanIsWhite)
                {
                    KS5_machine.my.game.iswhitesTurn = false;
                }
                else
                {
                    KS5_machine.my.game.iswhitesTurn = true;
                }

                //Thread.Sleep(500);

                //machineMove =
engine.GetNextMove(KS5_machine.my.humanMove);
                engine.writeMove(KS5_machine.my.humanMove);
                Thread.Sleep(1000);
                machineMove = engine.GetNextMove2();

                if
(KS5_machine.my.game.mainboard[Chess.getRow(machineMove.Substring(0,
2)) * 8 + Chess.getCol(machineMove.Substring(0, 2))].piece.Color ==
"White")
                {
                    machineMove = engine.GetNextMove();
                }

                lastMove = machineMove;
                if (KS5_machine.isPresent(machineMove.Substring(2, 2)))
                {
                    for (int j = 0; j < 2; j++)
```

```
                    {
                        for (int i = 0; i < 8; i++)
                        {
                            if (!KS5_machine.my.game.sbmachine[j * 8 +
i].isOccupied)
                            {
                                KS5_machine.my.game.sbmachine[j * 8 +
i] =
KS5_machine.my.game.mainboard[Chess.getRow(KS5_machine.my.move.Substrin
g(2, 2)) * 8 + Chess.getCol(KS5_machine.my.move.Substring(2, 2))];
                                KS5_machine.my.move =
machineMove.Substring(2, 2) + i.ToString() + (j + 2).ToString();
                                KS5_machine.my.currentAction =
capturePiece;

bWKS5Handler.RunWorkerAsync(KS5_machine.my);
                            }
                        }
                    }
                }
                else
                {
                    KS5_machine.my.currentAction = movePiece;
                    KS5_machine.my.move = machineMove;
                    bWKS5Handler.RunWorkerAsync(KS5_machine.my);
                }

                //engine.writeMove(KS5_machine.my.humanMove);

            }
            else if (KS5_machine.my.currentAction == movePiece)
            {
                if (KS5_machine.my.game.started)
                {
                    KS5_machine.my.currentAction = getHumanMove;
                    bWKS5Handler.RunWorkerAsync(KS5_machine.my);
                }
            }
            else if (KS5_machine.my.currentAction == capturePiece)
            {
                KS5_machine.my.currentAction = movePiece;
                KS5_machine.my.move = lastMove;
                bWKS5Handler.RunWorkerAsync(KS5_machine.my);
            }


            else
            {
                MessageBox.Show("Initialization Complete!");
            }


        }

        private void button1_Click(object sender, EventArgs e)
        {
            KS5_machine.my.currentAction = movePiece;
```

```
            KS5_machine.my.move = textBox1.Text;
            bWKS5Handler.RunWorkerAsync(KS5_machine.my);
        }

        private void button2_Click(object sender, EventArgs e)
        {

MessageBox.Show(KS5_machine.isPresent(textBox2.Text).ToString());
        }

        private void button3_Click(object sender, EventArgs e)
        {
            //game.humanIsWhite = true;
            //game.startGame(true);
            StartGame(true);
        }

        private void StartGame(bool HumanIsWhite)
        {
            KS5_machine.my.game.startGame(HumanIsWhite);

            /*
            if (HumanIsWhite)
            {

            }
            */
            engine = new Engine();
            engine.initengine();

            if (HumanIsWhite)
            {
                KS5_machine.my.currentAction = getHumanMove;
                bWKS5Handler.RunWorkerAsync(KS5_machine.my);
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            KS5_machine.my.currentAction = pointSquare;
            KS5_machine.my.move = textBox3.Text;
            bWKS5Handler.RunWorkerAsync(KS5_machine.my);
        }

        private void CheckStartingPosition()
        {

        }


    }
}
```

KS5.KS5Console.exe
KS5Console.cs

```
#region Using directives

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using KS5.Controller;
using System.Threading;
#endregion

namespace KS5.Main
{
    partial class KS5Console : Form
    {
        KS5Controller KS5C;

        public KS5Console()
        {
            InitializeComponent();
            KS5C = new KS5Controller();
        }

        private void KS5Console_Load(object sender, EventArgs e)
        {
            cboXDir.Text = "Home";
            cboYDir.Text = "Home";
            cboZDir.Text = "Home";
        }

        private void label8_Click(object sender, EventArgs e)
        {

        }


        private void button2_Click(object sender, EventArgs e)
        {
            int coil;
            if (chkX.Checked)
                coil = 0;
            else
                coil = 1;

            if (txtAccX.Text == "0")
            {
                KS5C.stepHomeX(coil);
            }
            else
            {
                KS5C.stepHomeX(coil, int.Parse(txtAccX.Text));
```

```
    }
}

private void button7_Click(object sender, EventArgs e)
{
    int coil;
    if (chkY.Checked)
        coil = 0;
    else
        coil = 1;
    if (txtAccY.Text == "0")
    {
        KS5C.stepHomeY(coil);
    }
    else
    {
        KS5C.stepHomeY(coil, int.Parse(txtAccY.Text));
    }
}

private void button15_Click(object sender, EventArgs e)
{
    int coil;
    if (chkZ.Checked)
        coil = 0;
    else
        coil = 1;
    if (txtAccZ.Text == "0")
    {
        KS5C.stepHomeZ(coil);
    }
    else
    {
        KS5C.stepHomeZ(coil, int.Parse(txtAccZ.Text));
    }
}

private void button24_Click(object sender, EventArgs e)
{
    KS5C.powerON();
}

private void button25_Click(object sender, EventArgs e)
{
    KS5C.powerOFF();
}

private void button9_Click(object sender, EventArgs e)
{
    KS5C.IndicatorON();
}

private void button10_Click(object sender, EventArgs e)
{
    KS5C.IndicatorOFF();
}
```

```
private void button23_Click(object sender, EventArgs e)
{
    if (KS5C.isPICXOnline())
        cboXState.Text = "Online";
    else
        cboXState.Text = "Offline";

    if (KS5C.isPICYOnline())
        cboYState.Text = "Online";
    else
        cboYState.Text = "Offline";

    if (KS5C.isPICZOnline())
        cboZState.Text = "Online";
    else
        cboZState.Text = "Offline";

    if (KS5C.isPowerOK())
        cboPower.Text = "Online";
    else
        cboPower.Text = "Offline";
}

private void button3_Click(object sender, EventArgs e)
{
    KS5C.coilXON();
}

private void button4_Click(object sender, EventArgs e)
{
    KS5C.coilXOFF();
}

private void button6_Click(object sender, EventArgs e)
{
    KS5C.coilYON();
}

private void button5_Click(object sender, EventArgs e)
{
    KS5C.coilYOFF();
}

private void button14_Click(object sender, EventArgs e)
{
    KS5C.coilZON();
}

private void button13_Click(object sender, EventArgs e)
{
    KS5C.coilZOFF();
}

private void button21_Click(object sender, EventArgs e)
{
    if (KS5C.isPiecePresent())
        cboPiece.Text = "Present";
```

```
            else
                cboPiece.Text = "None";
        }

        private void button22_Click(object sender, EventArgs e)
        {
            txtLDRSteps.Text = KS5C.getStepLDR().ToString();
        }

        private void button19_Click(object sender, EventArgs e)
        {
            KS5C.stepLDR(KS5C.my.maxStepZ, 1);
        }

        private void button17_Click(object sender, EventArgs e)
        {
            KS5C.laserON();
        }

        private void button18_Click(object sender, EventArgs e)
        {
            KS5C.laserOFF();
        }

        private void button12_Click(object sender, EventArgs e)
        {
            KS5C.EMON();
        }

        private void button11_Click(object sender, EventArgs e)
        {
            KS5C.EMOFF();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int dir, coil;
            if (cboXDir.Text == "Home")
                dir = 1;
            else
                dir = 0;

            if (chkX.Checked)
                coil = 0;
            else
                coil = 1;

            if (txtAccX.Text == "0")
            {
                KS5C.stepMotorX(int.Parse(txtXSteps.Text), dir, coil);
            }
            else
            {
                KS5C.stepMotorX(int.Parse(txtXSteps.Text), dir, coil,
int.Parse(txtAccX.Text));
            }
        }
```

```
        private void button8_Click(object sender, EventArgs e)
        {
            int dir, coil;
            if (cboYDir.Text == "Home")
                dir = 1;
            else
                dir = 0;

            if (chkY.Checked)
                coil = 0;
            else
                coil = 1;

            if (txtAccY.Text == "0")
            {
                KS5C.stepMotorY(int.Parse(txtYSteps.Text), dir, coil);
            }
            else
            {
                KS5C.stepMotorY(int.Parse(txtYSteps.Text), dir, coil,
int.Parse(txtAccY.Text));
            }
        }

        private void button16_Click(object sender, EventArgs e)
        {
            int dir, coil;

            if (cboZDir.Text == "Home")
                dir = 1;
            else
                dir = 0;

            if (chkZ.Checked)
                coil = 0;
            else
                coil = 1;

            if (txtAccZ.Text == "0")
            {
                KS5C.stepMotorZ(int.Parse(txtZSteps.Text), dir, coil);
            }
            else
            {
                KS5C.stepMotorZ(int.Parse(txtZSteps.Text), dir, coil,
int.Parse(txtAccZ.Text));
            }
        }

        private void cmdSpeedX_Click(object sender, EventArgs e)
        {
            KS5C.setSpeedX(int.Parse(txtSXLSB.Text),
int.Parse(txtSXMSB.Text));
        }

        private void cmdSpeedY_Click(object sender, EventArgs e)
```

```
        {
            KS5C.setSpeedY(int.Parse(txtSYLSB.Text),
int.Parse(txtSYMSB.Text));
        }


        private void cmdSpeedZ_Click(object sender, EventArgs e)
        {
            KS5C.setSpeedZ(int.Parse(txtSZLSB.Text),
int.Parse(txtSZMSB.Text));
        }

        private void button20_Click(object sender, EventArgs e)
        {

        }

        private void button28_Click(object sender, EventArgs e)
        {
            string str="";
            for (int i = 0; i < 8; i++)
            {
                str += KS5C.GetPortByte(i) + "\n";
                Thread.Sleep(10);
            }

            str += "\n";

            for (int i = 8; i < 10; i++)
            {
                str += KS5C.GetPortByte(i) + "\n";
                Thread.Sleep(10);
            }

            str += "\n";

            for (int i = 10; i < 12; i++)
            {
                str += KS5C.GetPortByte(i) + "\n";
                Thread.Sleep(10);
            }

            //MessageBox.Show(str);
            textBox1.Text = str;
        }


    }
}
```

Program.cs

```
#region Using directives

using System;
```

```
using System.Collections.Generic;
using System.Windows.Forms;

#endregion

namespace KS5.Main
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.EnableRTLMirroring();
            Application.Run(new KS5Console());
        }
    }
}
```

Software Requirements:

- Microsoft® Windows® 98
- Microsoft® Windows® 98 Second Edition
- Microsoft® Windows® Millennium Edition
- Microsoft® Windows NT® 4.0 Workstation with Service Pack 6.0a or later
- Microsoft® Windows NT® 4.0 Server with Service Pack 6.0a or later
- Microsoft® Windows® 2000 Professional
- Microsoft® Windows® 2000 Server
- Microsoft® Windows® 2000 Advanced Server
- Microsoft® Windows® 2000 Datacenter Server
- Microsoft® Windows® XP Home Edition
- Microsoft® Windows® XP Professional
- Microsoft® Windows® Server 2003 family

Note: On all these systems, Microsoft® Internet Explorer 5.01 or later and Microsoft® Windows® Installer 2.0 or later are also required.

Table 5.6 Hardware Requirements

	Minimum	**Recommended**
Processor	Pentium 1 (or similar processor)*	Pentium 3 or higher (or similar processor)
Memory (RAM)	32MB*	128MB or higher
Extra Disk Space	10MB*	50MB or higher

*Or the minimum required by the operating system, whichever is higher.

Hardware Accessories:

- Microphone
- Speaker
- Mouse