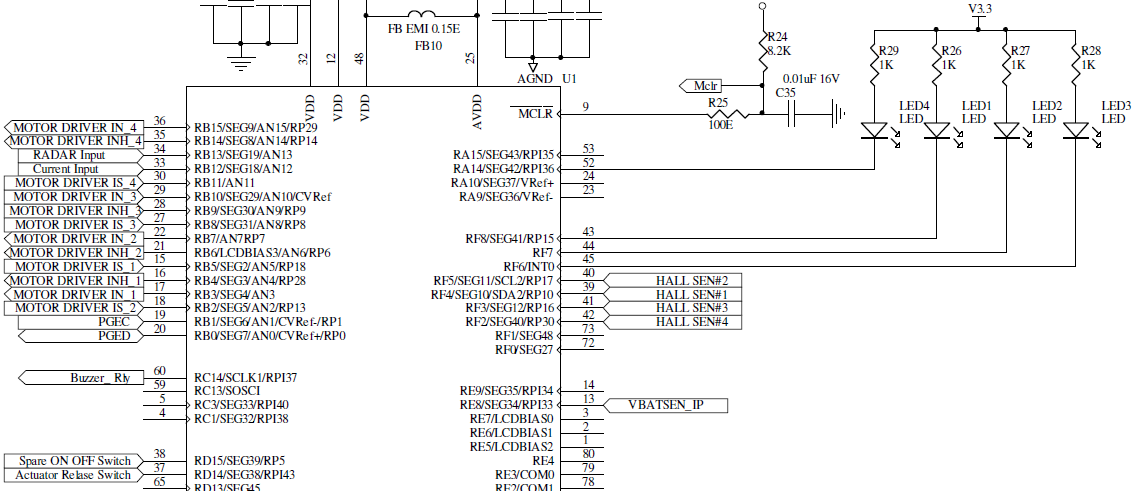
**Autonomous Tractor Safety System**

**Objective 5.0** To find and display the DC motor direction using LEDs in ATSS PCB board

****

**TRISF** DATA DIRECTION REGISTER F: 0x0000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRISF15 | TRISF14 | TRISF13 | TRISF12 | TRISF11 | TRISF10 | TRISF9 | TRISF8 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
|  |  |  |  |  |  |  | 0 |
|  |  |  |  |  |  |  | Output |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | TRISF0 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
| 0 | 0 |  |  |  |  |  |  |
| Output | Output |  |  |  |  |  |  |

**TRISA** DATA DIRECTION REGISTER A: 0x0000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRISA15 | TRISA14 | TRISA13 | TRISA12 | TRISA11 | TRISA10 | TRISA9 | TRISA8 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
|  | 0 |  |  |  |  |  |  |
|  | Output |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**TRISB** DATA DIRECTION REGISTER B: 0x3927

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRISB15 | TRISB14 | TRISB13 | TRISB12 | TRISB11 | TRISB10 | TRISB9 | TRISB8 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| OUTPUT | OUTPUT | INPUT | INPUT | INPUT | OUTPUT | OUTPUT | INPUT |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| OUTPUT | OUTPUT | INPUT | OUTPUT | OUTPUT | INPUT | INPUT | INPUT |

**ANSB** DATA DIRECTION REGISTER B: 0x0000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ANSB15 | ANSB14 | ANSB13 | ANSB12 | ANSB11 | ANSB10 | ANSB9 | ANSB8 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ANSB7 | ANSB6 | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL | DIGITAL |

**LATF** OUTPUT LATCH REGISTER F: 0x01C0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LATF15 | LATF14 | LATF13 | LATF12 | LATF11 | LATF10 | LATF9 | LATF8 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
|  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  | LED1 off |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | LATF0 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
| 1 | 1 |  |  |  |  |  |  |
| LED2 off | LED3 off |  |  |  |  |  |  |

**LATA** OUTPUT LATCH REGISTER A: 0x4000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LATAF15 | LATA14 | LATA13 | LATA12 | LATA11 | LATA10 | LATA9 | LATA8 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
|  | 1 |  |  |  |  |  |  |
|  | LED4 off |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 |
| I/O | I/O | I/O | I/O | I/O | I/O | I/O | I/O |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**INTCON2** INTERRUPT CONTROL REGISTER 2: 0x0000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ALTIVT | DISI | - | - | - | - | - | - |
| ALT INT VECTOR TABLE EN | DISI INSTR STATUS |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | - | INT4EP | INT3EP | INT2EP | INT1EP | INT0EP |
|  |  |  | EXT INT4 POLARITY | EXT INT3 POLARITY | EXT INT2 POLARITY | EXT INT1 POLARITY |  |
|  |  |  | 0 | 0 | 0 | 0 |  |
|  |  |  | +VE EDGE | +VE EDGE | +VE EDGE | +VE EDGE |  |

**IFS1 INTERRUPT FLAG STATUS REGISTER 1**: 0x0000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| U2TXIF | U2RXIF | INT2IF | T5IF | T4IF | OC4IF | OC3IF | DMA2IF |
| UART2 Tx REQUEST | UART2 Rx REQUEST | EXT INT2 REQUEST | TMR4 INT REQUEST | TMR4 INT REQUEST | O/P CMP CH4 INT | O/P CMP CH3 INT | DMA CH2 INT |
|  |  | 0 |  |  |  |  |  |
|  |  | NO INT |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | IC7IF | - | INT1IF | CNIF | CMIF | MI2C1IF | S12C1IF |
|  | I/P CH 7 CAPTURE |  | EXT INT1 REQUEST | I/P CHNG NOTIF INT | CMP INT | MASTER I2C1 INT | SLAVE I2C1 INT |
|  |  |  | 0 |  |  |  |  |
|  |  |  | NO INT |  |  |  |  |

**IFS3 INTERRUPT FLAG STATUS REGISTER 3**: 0x0000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | RTCIF | DMA5IF | - | - | - | - | - |
|  | RTC INT REQ | DMA CH5 INT |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | INT4IF | INT3IF | - | - | MI2C2IF | SI2C2IF | - |
|  | EXT INT4 REQUEST | EXT INT3 REQUEST |  |  | MASTER I2C2 INT | SLAVE I2C2 INT |  |
|  | 0 | 0 |  |  |  |  |  |
|  | NO INT | NO INT |  |  |  |  |  |

**IEC1** INTERRUPT ENABLE CONTROL REGISTER 1: 0x2010

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| U2TXIE | U2RXIE | INT2IE | T5IE | T4IE | OC4IE | OC3IE | DMA2IE |
| UART2 Tx REQUEST | UART2 Rx REQUEST | EXT INT2 REQUEST | TMR4 INT REQUEST | TMR4 INT REQUEST | O/P CMP CH4 INT | O/P CMP CH3 INT | DMA CH2 INT |
|  |  | 1 |  |  |  |  |  |
|  |  | ENABLE |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | IC7IE | - | INT1IE | CNIE | CMIE | MI2C1IE | S12C1IE |
|  | I/P CH 7 CAPTURE |  | EXT INT1 REQUEST | I/P CHNG NOTIF INT | CMP INT | MASTER I2C1 INT | SLAVE I2C1 INT |
|  |  |  | 1 |  |  |  |  |
|  |  |  | ENABLE |  |  |  |  |

**IEC3** INTERRUPT ENABLE CONTROL REGISTER 3: 0x0060

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | RTCIE | DMA5IE | - | - | - | - | - |
|  | RTC INT REQ | DMA CH5 INT |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | INT4IE | INT3IE | - | - | MI2C2IE | SI2C2IE | - |
|  | EXT INT4 REQUEST | EXT INT3 REQUEST |  |  | MASTER I2C2 INT | SLAVE I2C2 INT |  |
|  | 1 | 1 |  |  |  |  |  |
|  | ENABLE | ENABLE |  |  |  |  |  |

**IPC5** INTERRUPT PRIORITY CONTROL REGISTER 5: 0x0004

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | - | - | - | IC7IP2 | IC7IP1 | IC7IP0 |
|  |  |  |  |  | I/P CAPTURE CH7 PRIORITY | | |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | - | - | - | INT1IP2 | INT1IP1 | INT1IP0 |
|  |  |  |  |  | EXT INT1 PRIORITY | | |
|  |  |  |  |  | 1 | 0 | 0 |
|  |  |  |  |  | 4 | | |

**IPC7** INTERRUPT PRIORITY CONTROL REGISTER 7: 0x0040

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | U2TXIP2 | U2TXIP1 | U2TXIP0 | - | U2RXIP2 | U2RXIP1 | U2RXIP0 |
|  | UART2 TX PRIORITY | | |  | UART2 RX PRIORITY | | |
|  |  |  |  |  |  |  |  |
|  |  | | |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | INT2IP2 | INT2IP1 | INT2IP0 | - | T5IP2 | T5IP1 | T5IP0 |
|  | EXT INT2 PRIORITY | | |  | TMR5 INT PRIORITY | | |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 4 | | |  |  |  |  |

**IPC13** INTERRUPT PRIORITY CONTROL REGISTER 13: 0x0440

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | - | - | - | INT4IP2 | INT4IP1 | INT4IP0 |
|  |  |  |  |  | EXT INT4 PRIORITY | | |
|  |  |  |  |  | 1 | 0 | 0 |
|  |  | | |  | 4 | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | INT3IP2 | INT3IP1 | INT3IP0 | - | - | - | - |
|  | EXT INT3 PRIORITY | | |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 4 | | |  |  |  |  |

**RPINR0** PERIPHERAL PIN SELECT INPUT REGISTER 0: 0x2700

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | INT1R5 | INT1R4 | INT1R3 | INT1R2 | INT1R1 | INT1R0 |
|  |  | ASSIGN EXT INT1 CORRESPONDING TO PINS IN RP0 | | | | | |
|  |  | 1 | 0 | 0 | 1 | 1 | 1 |
|  |  | PIN 39 IS HALL SENSOR1 | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | - | - | - | - | - | - |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**RPINR1** PERIPHERAL PIN SELECT INPUT REGISTER 1: 0x2928

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | INT3R5 | INT3R4 | INT3R3 | INT3R2 | INT3R1 | INT3R0 |
|  |  | ASSIGN EXT INT3 CORRESPONDING TO PINS IN RP1 | | | | | |
|  |  | 1 | 0 | 1 | 0 | 0 | 1 |
|  |  | PIN 41 IS HALL SENSOR3 | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | INT2R5 | INT2R4 | INT2R3 | INT2R2 | INT2R1 | INT2R0 |
|  |  | ASSIGN EXT INT2 CORRESPONDING TO PINS IN RP1 | | | | | |
|  |  | 1 | 0 | 1 | 0 | 0 | 0 |
|  |  | PIN 40 IS HALL SENSOR2 | | | | | |

**RPINR2** PERIPHERAL PIN SELECT INPUT REGISTER 2: 0x002A

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | - | - | - | - | - | - |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | INT4R5 | INT4R4 | INT4R3 | INT4R2 | INT4R1 | INT4R0 |
|  |  | ASSIGN EXT INT2 CORRESPONDING TO PINS IN RP1 | | | | | |
|  |  | 1 | 0 | 1 | 0 | 1 | 0 |
|  |  | PIN 42 IS HALL SENSOR4 | | | | | |

**Code Prg05.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Source code file: Prg05.c

Author, version, date: PSS ver 1.0 01.11.24

Program function: Motor direction

Simulation: PIC24FJ128GA308 MCU, MPLAB X IDE ver 6.05, XC16 ver 2.10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

SW1 switch on safety box for running (LED ON) the clutch (LED1) and brake (LED3) motors or stopping the motors (LED OFF)

SW2 switch near driver for running the clutch (LED2) and brake (LED4) motors in a clockwise (LED ON) or anticlockwise direction (LED OFF)

LED1: ON - clutch motor is running, OFF - clutch motor is not running

LED2: ON - clutch motor is running CLOCKWISE, OFF - clutch motor is running ANTICLOCKWISE

LED3: ON - brake motor is running, OFF - brake motor is not running

LED4: ON - brake motor is running CLOCKWISE, OFF - brake motor is running ANTICLOCKWISE

\*/

#include <xc.h>

#include <libpic30.h>

#define LED1 LATFbits.LATF8

#define LED2 LATFbits.LATF7

#define LED3 LATFbits.LATF6

#define LED4 LATAbits.LATA14

#define spareOnOffSwitch LATDbits.LATD15

#define actuatorReleaseSwitch LATDbits.LATD14

#define INH1 LATBbits.LATB4 // Motor driver inhibit pins

#define INH2 LATBbits.LATB6

#define INH3 LATBbits.LATB9

#define INH4 LATBbits.LATB14

#define IN1 LATBbits.LATB3 // Motor input pins

#define IN2 LATBbits.LATB7

#define IN3 LATBbits.LATB10

#define IN4 LATBbits.LATB15

#define IS1 LATBbits.LATB5 // Motor current sensor pins

#define IS2 LATBbits.LATB2

#define IS3 LATBbits.LATB8

#define IS4 LATBbits.LATB11

#define ON 1

#define OFF 0

#define BEGINNING 0 // Fully disengaged

#define MIDDLE -1 // Fifty percent engaged

#define END 1 // Fully engaged

#define ON 1

#define OFF 0

#define CLOCKWISE 1

#define ANTICLOCKWISE 0

#define hallSensor1 LATFbits.LATF4

#define hallSensor2 LATFbits.LATF5

#define hallSensor3 LATFbits.LATF3

#define hallSensor4 LATFbits.LATF2

uint8\_t spareOnOffSwitchNow**;**

uint8\_t actuatorReleaseSwitchNow**;**

uint8\_t motorsState**;**

uint8\_t motorsStateNow**;**

uint8\_t clutchMotorState**;**

uint8\_t clutchMotorStateNow**;**

uint8\_t clutchMotorDirectionNow**;**

uint8\_t clutchMotorPositionNow**;**

uint8\_t bit0**;**

uint8\_t bit1**;**

uint8\_t bits10**;**

uint8\_t bits32**;**

uint8\_t bits3210**;**

uint8\_t brakeMotorState**;**

uint8\_t brakeMotorStateNow**;**

uint8\_t motorPosition**;**

uint8\_t brakeMotorDirectionNow**;**

uint8\_t brakeMotorPositionNow**;**

uint8\_t motorDirection**;**

uint16\_t motorDegrees**;**

uint8\_t motorPulsesCount**;**

uint8\_t motorPulsesPerRevolution **=** 280**;**

uint8\_t INH1Now**;**

uint8\_t INH2Now**;**

uint8\_t INH3Now**;**

uint8\_t INH4Now**;**

uint8\_t IN1Now**;**

uint8\_t IN2Now**;**

uint8\_t IN3Now**;**

uint8\_t IN4Now**;**

uint8\_t extInt1**;**

uint8\_t extInt2**;**

uint8\_t extInt3**;**

uint8\_t extInt4**;**

void initializeMotors**(**void**);**

uint8\_t turnOnMotors**(**void**);**

uint8\_t getClutchMotorState**(**void**);**

uint8\_t getBrakeMotorState**(**void**);**

uint8\_t getMotorDirection**();**

uint8\_t getMotorPosition**();**

void engageClutchMotor**(**void**);**

void disEngageClutchMotor**(**void**);**

void engageBrakeMotor**(**void**);**

void disEngageBrakeMotor**(**void**);**

int main**()**

**{**

initializeMotors**();**

turnOnMotors**();**

**while** **(**1**)**

**{**

clutchMotorStateNow **=** getClutchMotorState**();**

**if** **(**clutchMotorStateNow**)** clutchMotorDirectionNow **=** getMotorDirection**();**

brakeMotorStateNow **=** getBrakeMotorState**();**

**if** **(**brakeMotorStateNow**)** brakeMotorDirectionNow **=** getMotorDirection**();**

**if** **(**clutchMotorStateNow**)**

LATFbits**.**LATF6 **=** 0**;** // LED3 RF6 ON, clutch motor is running

**else**

LATFbits**.**LATF6 **=** 1**;** // LED3 RF6 OFF , clutch motor is not running

**if** **(**clutchMotorDirectionNow**)**

LATFbits**.**LATF7 **=** 0**;** // LED2 RF7 ON, cluctch motor is turning clockwise

**else**

LATFbits**.**LATF7 **=** 1**;**// LED2 RF7 OFF, clutch motor is running anticlockwise

**if** **(**brakeMotorStateNow**)**

LATFbits**.**LATF8 **=** 0**;** // LED1 RF8 ON, brake motor is running

**else**

LATFbits**.**LATF8 **=** 1**;** // LED1 RF8 OFF, brake motor is not running

**if** **(**brakeMotorDirectionNow**)**

LATAbits**.**LATA14 **=** 0**;** // LED4 RA14 ON, brake motor is turning clockwise

**else**

LATAbits**.**LATA14 **=** 1**;**// LED4 RA14 OFF, brake motor is turning anticlockwise

**}**

**}**

void initializeMotors**(**void**)**

**{**

ANSB **=** 0x0000**;** // Digital mode(D):0, Analog mode(A):1

TRISA **=** 0x0000**;** // Output mode(O):0, Input mode(I):1, RA14:LED4(O)

TRISB **=** 0x3923**;** // RB0:PGEC(I), RB1:PGED(I), RB2:IS2(I), RB3:IN1(O),

// RB4:INH1(O), RB5:IS1(I), RB6:INH2(O), RB7:IN2(O), RB8:IS3(I), RB9:INH3(O),

// RB10:IN3(O), RB11:IS4(I), RB12:CurrentSensorInput(I), RB13:RadarSensorInput(I),

// RB14:INH4(O), RB15:IN4(O)

TRISF **=** 0x003C**;** // RF2:HallSensor4(I), RF3:HallSensor3(I),

// RF4:HallSensor1(I), RF5:HallSensor2(I)

TRISD **=** 0xC110**;** // RD0:CAN CS(O), RD4:JetsonXavierTx(I),

// RD5:JetsonXavierRx(O), RD8:CAN SDI(I), RD9:CAN SDO(O), RD10:CAN SCL(I), RD11:CAN

// INT(O), RD14:spareOnOffSwitch(I), RD15:actuatorReleaseSwitch(I)

LATF **=** 0x01C0**;** // Set all LEDs LOW initially

LATA **=** 0x4000**;** // All the LEDS are in common cathode configuration

PORTA **=** PORTB **=** PORTD **=** 0x0000**;**

LATA **=** LATB **=** LATD **=** 0x0000**;**

RPINR0 **=** 0x2700**;** // PPS Peripheral Pin Select INT1(39:0x27)-hallSensor1(39)

RPINR1 **=** 0x2928**;**// INT3(41:0x29)-hallSensor3(41) INT2(40:0x28)-hallSensor2(40)

RPINR2 **=** 0x002A**;** // INT4(42:0x2A)-hallSensor4(42)

INTCON2 **=** 0x0000**;** // INT1-4 are +ve edge triggered

IFS1 **=** 0x0000**;** // Clear INT1,2 flags

IFS3 **=** 0x0000**;** // Clear INT3,4 flags

IEC1 **=** 0x2010**;** // Enable INT1,2

IEC3 **=** 0x0060**;** // Enable INT3,4

IPC5 **=** 0x0004**;** // Set INT1 priority as 4 (default)

IPC7 **=** 0x0040**;** // Set INT2 priority as 4 (default)

IPC13 **=** 0x0440**;** // Set INT3,4 priority as 4 (default)

LED1 **=** 1**;** // Set all LED OFF initially

LED2 **=** 1**;**

LED3 **=** 1**;**

LED4 **=** 1**;**

clutchMotorPositionNow **=** BEGINNING**;**

brakeMotorPositionNow **=** BEGINNING**;**

**}**

uint8\_t turnOnMotors**(**void**)**

**{**

spareOnOffSwitchNow **=** spareOnOffSwitch**;**

actuatorReleaseSwitchNow **=** actuatorReleaseSwitch**;**

**if** **(**spareOnOffSwitchNow**)** // RD15 Spare ON/OFF switch

**{**

**if** **(**actuatorReleaseSwitchNow**)** // RD14 actuatorReleaseSwitch

**{**

**while** **(**clutchMotorPositionNow **!=** END**)**

**{**

engageClutchMotor**();** // Motors run in clockwise direction

if (!IS1 & !IS2)

{

clutchMotorPositionNow **=** getMotorPosition**();**

clutchMotorDirectionNow **=** getMotorDirection**();**

**}**

**}**

**while** **(**brakeMotorPositionNow **!=** END**)**

**{**

engageBrakeMotor**();**

if (!IS3 & !IS4)

{

brakeMotorPositionNow **=** getMotorPosition**();**

brakeMotorDirectionNow **=** getMotorDirection**();**

}

**}**

**}**

**else**

**{**

**while** **(**clutchMotorPositionNow **!=** BEGINNING**)**

**{**

disEngageClutchMotor**();** // Motors run in anticlockwise direction

if (!IS1 & !IS2)

{

clutchMotorPositionNow **=** getMotorPosition**();**

clutchMotorDirectionNow **=** getMotorDirection**();**

**}**

**}**

**while** **(**brakeMotorPositionNow **!=** BEGINNING**)**

**{**

disEngageBrakeMotor**();**

if (!IS3 & !IS4)

{

brakeMotorPositionNow **=** getMotorPosition**();**

brakeMotorDirectionNow **=** getMotorDirection**();**

**}**

**}**

**}**

**}**

**else**

**{**

INH1 **=** 0**;** // Clutch motor in SLEEP inactive mode

INH2 **=** 0**;**

INH3 **=** 0**;** // Brake motor in SLEEP inactive mode

INH4 **=** 0**;**

clutchMotorStateNow **=** OFF**;**

brakeMotorStateNow **=** OFF**;**

**}**

**}**

uint8\_t getClutchMotorState**(**void**)**

**{**

INH1Now **=** INH1**;**

INH2Now **=** INH2**;** // RB4 is INH1, RB6 is INH2

IN1Now **=** IN1**;** // RB3 is IN1, RB7 is IN2

IN2Now **=** IN2**;**

**if** **((**INH1Now**)** **&** **(**INH2Now**))** // Low INHIBIT puts BT8982A in SLEEP mode, clutch motor

// is in an OFF state

**{**

**if** **(((**IN1Now **&** **(!**IN2Now**))** **|** **((!**IN1Now**)** **&** IN2Now**)))**// (IN1, IN2) = (0, 1) is

// CLOCKWISE, (IN1, IN2) = (1, 0) is ANTICLOCKWISE

clutchMotorState **=** ON**;**

**else**

clutchMotorState **=** OFF**;**

**}**

**return** clutchMotorState**;**

**}**

uint8\_t getBrakeMotorState**(**void**)**

**{**

INH3Now **=** INH3**;** // RB10 is INH3, RB15 is INH4

INH4Now **=** INH4**;**

IN3Now **=** IN3**;** // RB9 is IN3, RB14 is IN4

IN4Now **=** IN4**;**

**if** **((**INH3Now**)** **&** **(**INH4Now**))** // Low INHIBIT puts BT8982A in SLEEP mode, brake

// motor is in an OFF state

**{**

**if** **(((**IN3 **&** **(!**IN4**))** **|** **((!**IN3**)** **&** IN4**)))** // (IN3, IN4) = (0, 1) is CLOCKWISE,

// (IN3, IN4) = (1, 0) is ANTICLOCKWISE

brakeMotorState **=** ON**;**

**else**

brakeMotorState **=** OFF**;**

**}**

**return** brakeMotorState**;**

**}**

uint8\_t getMotorDirection**()**

**{**

**if** **((**extInt1 **&** extInt2**)** **|** **(**extInt3 **&** extInt4**))**

**{**

bit1 **=** hallSensor1**;**

bit0 **=** hallSensor2**;**

bits10 **=** **(**bit1 **<<** 1**)** **|** bit0**;**

bits3210 **=** **(**bits32 **<<** 2**)** **|** bits10**;**

**if** **(**bits3210 **==** 0b1101 **||** bits3210 **==** 0b0100 **||** bits3210 **==** 0b0010 **||** bits3210 **==** 0b1011**)** motorDirection **=** ANTICLOCKWISE**;**

**if** **(**bits3210 **==** 0b1110 **||** bits3210 **==** 0b0111 **||** bits3210 **==** 0b0001 **||** bits3210 **==** 0b1000**)** motorDirection **=** CLOCKWISE**;**

bits32 **=** bits10**;**

**}**

**return** motorDirection**;**

**}**

uint8\_t getMotorPosition**()**

**{**

**if** **((**extInt1 **&** extInt2**)** **|** **(**extInt3 **&** extInt4**))**

**{**

bit1 **=** hallSensor3**;**

bit0 **=** hallSensor4**;**

bits10 **=** **(**bit1 **<<** 1**)** **|** bit0**;**

bits3210 **=** **(**bits32 **<<** 2**)** **|** bits10**;**

**if** **(**bits3210 **==** 0b1101 **||** bits3210 **==** 0b0100 **||** bits3210 **==** 0b0010 **||**

bits3210 **==** 0b1011**)** motorPulsesCount**--;**

**if** **(**bits3210 **==** 0b1110 **||** bits3210 **==** 0b0111 **||** bits3210 **==** 0b0001 **||**

bits3210 **==** 0b1000**)** motorPulsesCount**++;**

bits32 **=** bits10**;**

motorDegrees **=** motorPulsesCount **\*** 360 **/** motorPulsesPerRevolution**;** // degrees = // edgeCount \* 360 /(x \* N), x = type of encoding (X1, X2, X4), x = 1 for X1

**switch** **(**motorDegrees**)** // N is the pulse count for one revolution

**{**

**case** 0**:**

motorPosition **=** BEGINNING**;**

**break;**

**case** 180**:**

motorPosition **=** MIDDLE**;**

**break;**

**case** 360**:**

motorPosition **=** END**;**

**break;**

**}**

**}**

**return** motorPosition**;**

**}**

void engageClutchMotor**(**void**)**

**{**

INH1 **=** 1**;**

INH2 **=** 1**;**

IN1 **=** 1**;**

IN2 **=** 0**;** // Clockwise motor

**}**

void disEngageClutchMotor**(**void**)**

**{**

INH1 **=** 1**;**

INH2 **=** 1**;** // Counterclockwise motor

IN1 **=** 0**;**

IN2 **=** 1**;**

**}**

void engageBrakeMotor**(**void**)**

**{**

INH3 **=** 1**;**

INH4 **=** 1**;**

IN3 **=** 1**;**

IN4 **=** 0**;**

**}**

void disEngageBrakeMotor**(**void**)**

**{**

INH3 **=** 1**;**

INH4 **=** 1**;**

IN3 **=** 0**;**

IN4 **=** 1**;**

**}**

void \_\_attribute\_\_**((**interrupt**,** no\_auto\_psv**))** \_INT1Interrupt**(**void**)**

// External Interrupt 1 service sub routine

**{**

IFS1bits**.**INT1IF **=** 0**;** // Clear External Interrupt Flag INT1IF(0)

extInt1 **=** 1**;**

**}**

void \_\_attribute\_\_**((**interrupt**,** no\_auto\_psv**))** \_INT2Interrupt**(**void**)**

// External Interrupt 2 service sub routine

**{**

IFS1bits**.**INT2IF **=** 0**;** // Clear External Interrupt Flag INT2IF(0)

extInt2 **=** 1**;**

**}**

void \_\_attribute\_\_**((**interrupt**,** no\_auto\_psv**))** \_INT3Interrupt**(**void**)**

// External Interrupt 3 service sub routine

**{**

IFS3bits**.**INT3IF **=** 0**;** // Clear External Interrupt Flag INT3IF(0)

extInt3 **=** 1**;**

**}**

void \_\_attribute\_\_**((**interrupt**,** no\_auto\_psv**))** \_INT4Interrupt**(**void**)**

// External Interrupt 4 service sub routine

**{**

IFS3bits**.**INT4IF **=** 0**;** // Clear External Interrupt Flag INT4IF(0)

extInt4 **=** 1**;**

**}**