FTC #5037 Source Code 2013: Block Party

Generated by Doxygen 1.8.5

Tue Dec 10 2013 11:09:03

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 abs_drive.h File Reference

it allows the robot to drive forword and backward

Functions

- void **abs_drive** (e_drive_direction dir, e_move_stopping_method dist_method, int dist, int speed, bool stop_at_end)

2.1.1 Detailed Description

it allows the robot to drive forword and backward

Parameters

dir	Tells the robot what direction to go
dist_method	tells the robot how it should know when to stop
dist	tells the robot how far to go
speed	tells the robot how fast to go
stop_at_end	tells the robot if it should stop when it gets to were it needs to go or not

Returns

returns nothing

Copyright

Definition in file abs_drive.h.

2.2 abs_drive.h

```
00001
```

```
#ifndef ABS_DRIVE_H
#define ABS_DRIVE_H

void abs_drive(e_drive_direction dir, e_move_stopping_method dist_method, int dist, int speed, bool stop_at_end)
{
    HTANGresetAccumulatedAngle(angle_sensor);
    int i = 0;

    nMotorEncoder(right_motor)= 0;
    g_rel_heading = 0;

    //------------------------
    // time stopping method
    //------------------------
    if(dist_method == E_TIME)
    {
        ClearTimer(T1);
        while(time1[T1] < dist)
        {
            abs_gyro_drive(speed,dir);
        }
    }
    //------------------------
    // encoder stopping method
    //------------------------
    else if(dist_method == E_DEGREES)
    {
        while(i<5)
        {
            if(abs(nMotorEncoder(right_motor)) > distance_to_encoder_derees(dist)) i++;
            abs_gyro_drive(speed,dir);
        }
    }
    //------------------------
    // IR stopping method
    //------------------------
    else if(dist_method == E_IR_DETECT)
    {
        if(dir == FORWARD)
        {
            while(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (150*INT_ANGLE_SENSOR_CIRCUMFERENCE))
            {
                if(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (100*INT_ANGLE_SENSOR_CIRCUMFERENCE))
                {
                    if(!((g_bearing_ac2 >= dist - 1) || (g_bearing_ac2 == 0))) break;
                }
                else
                {
                    if(!((g_bearing_ac2 >= dist) || (g_bearing_ac2 == 0))) break;
                }
                abs_gyro_drive(speed,dir);
            }
            //g_screen_state = S_TIME_SHOW;
            g_debug_time_1 = nPgmTime;
        }
        else if(dir == BACKWARD)
        {
            while(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (150*INT_ANGLE_SENSOR_CIRCUMFERENCE))
            {
                if(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (100*INT_ANGLE_SENSOR_CIRCUMFERENCE))
                {
                    if(!((g_bearing_ac1 <= dist + 1) || (g_bearing_ac1 == 0))) break;
                }
                else
                {
                    if(!((g_bearing_ac1 <= dist) || (g_bearing_ac1 == 0))) break;
                }
                abs_gyro_drive(speed,dir);
            }
            //g_screen_state = S_TIME_SHOW;
            g_debug_time_1 = nPgmTime;
        }
    }
    //------------------------
    // IR stopping method 2
    //------------------------
    else if(dist_method == E_IR_DETECT2)
    {
        if(dir == FORWARD)
        {
            while(g_ir_bearing2 > dist)
            {
                abs_gyro_drive(speed,dir);
            }
        }
        else
        {
            while(g_ir_bearing2 < dist)
            {
                abs_gyro_drive(speed,dir);
            }
        }
    }
    //------------------------
    // accelermeoter sensor stopping method
    //------------------------
    else if(dist_method == E_TILT)
    {
        int j = 0;
        g_sensor_reference_drive = true;
        while(j<30)
        {
            abs_gyro_drive(speed,dir);
            if(g_accelermoeter_average > dist) j++;
        }
        g_sensor_reference_drive = false;
    }
    //------------------------
    // angle sensor stopping method
    //------------------------
    else
    {
        HTANGresetAccumulatedAngle(angle_sensor);
        while(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (dist*
INT_ANGLE_SENSOR_CIRCUMFERENCE))
        {
            abs_gyro_drive(speed,dir);
        }
    }
    if(stop_at_end)
    {
        motor[left_motor] = 0;
        motor[right_motor] = 0;
    }
    g_debug_time_2 = nPgmTime;
}

#endif /* !ABS_DRIVE_H */
```

2.3 abs_end_r1.h File Reference

stop point function to end on ramp 1

Functions

- void **abs_end_r1** (int delay, int lift_speed)

2.3.1 Detailed Description

stop point function to end on ramp 1

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_end_r1.h.

2.4 abs_end_r1.h

```
#ifndef ABS_END_R1_H
#define ABS_END_R1_H

void abs_end_r1(int delay, int lift_speed)
{
    wait1Msec(delay);
    servo[abdd] = g_abdd_down;
    abs_drive(FORWARD, E_ANGLE, g_to_turn_dist, 50, true);
    wait1Msec(200);
    abs_turn(COUNTERCLOCKWISE, POINT, TURN, 75, 60);
    wait1Msec(200);
    abs_drive(FORWARD, E_ANGLE, 85, 50, true);
    motor[block_lift_motor] = lift_speed;
    motor[block_lift_motor2] = lift_speed;
    abs_turn(COUNTERCLOCKWISE, POINT, TURN, 90, 60);
    motor[block_lift_motor] = 0;
    motor[block_lift_motor2] = 0;
    if(g_auto_grabber_selection_ramp_options == SUB_SELECTION_RAMP_STOP) abs_drive(FORWARD, E_ANGLE, 80, 50, true);
    else abs_drive(FORWARD, E_ANGLE, 130, 50, true);
}

#endif /* !ABS_S1_END_R1_H */
```

2.5 abs_end_r2.h File Reference

stop point function to end on ramp 2

Functions

- void **abs_end_r2** (int delay, int lift_speed)

2.5.1 Detailed Description

stop point function to end on ramp 2

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_end_r2.h.

2.6 abs_end_r2.h

```
#ifndef ABS_END_R2_H
#define ABS_END_R2_H

void abs_end_r2(int delay, int lift_speed)
{
    wait1Msec(delay);
    servo[abdd] = g_abdd_down;
    abs_drive(BACKWARD, E_ANGLE, g_to_turn_dist, 50, true);
    wait1Msec(200);
    abs_turn(COUNTERCLOCKWISE, POINT, TURN, 90, 60);
    wait1Msec(200);
    abs_drive(FORWARD, E_ANGLE, 87, 50, true);
    wait1Msec(500);
    motor[block_lift_motor] = lift_speed;
    motor[block_lift_motor2] = lift_speed;
    abs_turn(CLOCKWISE, POINT, TURN, 84, 50);
    motor[block_lift_motor] = 0;
    motor[block_lift_motor2] = 0;
    if(g_auto_grabber_selection_ramp_options == SUB_SELECTION_RAMP_STOP) abs_drive(FORWARD, E_ANGLE, 80, 50, true);
    else abs_drive(FORWARD, E_ANGLE, 130, 50, true);
}

#endif /* !ABS_S1_END_R2_H */
```

2.7 abs_gyro_cal.h File Reference

A header file that allows you to calculate the input comming from the gyro.

Functions

- float **abs_gyro_cal** (long caltime)

2.7.1 Detailed Description

A header file that allows you to calculate the input comming from the gyro.

Parameters

caltime	Tells the robot how long to calibrate the gyro

Returns

The drift

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_gyro_cal.h.

2.8 abs_gyro_cal.h

```

#ifndef ABS_GYRO_CAL_H
#define ABS_GYRO_CAL_H

float abs_gyro_cal(long caltime)
{
    long highest = -1000, lowest = 10000;
    float average = 0;
    g_start_time = nPgmTime;
    long samples=0;
    long data;
    while (nPgmTime < g_start_time+(caltime*1000))      // loop for the requested number of seconds
    {
        samples +=1;                    // count the number of iterations for averaging
        data = HTGYROreadRot(HTGYRO);           // get a new reading from the GYRO
        average += (float)data;             // add in the new value to the average
        if (highest < data) highest = data;     // adjust the highest value if necessary
            if (lowest> data) lowest = data;    // likewise for the lowest value
    }
    //g_gyro_noise=abs(highest-lowest);         // save the spread in the data for diagnostic display
    g_gyro_noise=abs(highest-lowest);
    return average/samples;                 // and return the average drift
}

#endif /* !ABS_GYRO_CAL_H */
```

2.9 abs_gyro_drive.h File Reference

handles the speed control for the motors based on the gyro

Functions

- void **abs_gyro_drive** (int speed, e_drive_direction dir)

2.9.1 Detailed Description

handles the speed control for the motors based on the gyro

Parameters

speed	tells the robot how fast to go
dir	Tells the robot what direction to go

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_gyro_drive.h.

2.10 abs_gyro_drive.h

```

#ifndef ABS_GYRO_DRIVE_H
#define ABS_GYRO_DRIVE_H

void abs_gyro_drive(int speed,e_drive_direction dir)
{
    int error = 0 - g_rel_heading;

    if(dir == FORWARD)
    {
        motor[left_motor] = speed + (error*g_gyro_adjust);
        motor[right_motor] = speed - (error*g_gyro_adjust);
    }
    else
    {
        motor[left_motor] = -(abs(speed) - (error*g_gyro_adjust));
        motor[right_motor] = -(abs(speed) + (error*g_gyro_adjust));
    }
}
#endif /* !ABS_GYRO_DRIVE_H */
```

2.11 abs_initialize.h File Reference

A header file that handles the initialization when we start the game.

Functions

- void **initialize** ()

2.11.1 Detailed Description

A header file that handles the initialization when we start the game.

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_initialize.h.

2.12 abs_initialize.h

```

#ifndef ABS_INITIALIZE_H
#define ABS_INITIALIZE_H

void initialize()
{
    StartTask(abs_screen);
    disableDiagnosticsDisplay();
    servoChangeRate[abdd] = 3;
    servo[roger_slide] = 127;
    servo[abdd] = g_abdd_down;
    servo[grabber_left] = GRABBER_LEFT_CLOSE;
    servo[grabber_right] = GRABBER_RIGHT_CLOSE;
    memset(g_intput_array,0,6);
    selection_program();
    PlaySoundFile("! Click.rso");
    g_drift = abs_gyro_cal(g_gyro_cal_time);

    if(!HTACreadAllAxes(HTAC, g_x_axis, g_y_axis, g_z_axis)) g_error = ERR_ACCELERMOETER;
    if(g_gyro_noise>10) g_error = ERR_GYRO_CAL;
    if(HTSMUXreadPowerStatus(SENSOR_MUX)) g_error = ERR_SENSOR_MUX;
    if(HTSMUXreadPowerStatus(GYRO_MUX)) g_error = ERR_GYRO_MUX;

    if(g_error != 0)
    {
        g_screen_state = S_ERROR;
        while(true)
        {
            g_gyro_true = true;
            PlayTone (250,25);
            wait1Msec(500);
            if(nNxtButtonPressed == kEnterButton && g_error == ERR_SENSOR_MUX)break;
            if(nNxtButtonPressed == kEnterButton && g_error == ERR_ACCELERMOETER)break;
        }
    }

    g_screen_state = S_READY;
    StartTask(abs_sensors);
    HTANGresetAccumulatedAngle(angle_sensor);

    waitForStart();
    eraseDisplay();
    g_start_time = nPgmTime;
    g_screen_state = S_DELAY_WAIT;
    wait1Msec(g_start_delay*1000);
    eraseDisplay();
    g_screen_state = S_GYRO_SHOW;
}

#endif /* !ABS_INITIALIZE_H */
```

2.13 abs_joystick_drive.h File Reference

The header file that handles the joystick motor control.

Functions

- void **abs_joystick_drive** (e_joystick_method joy_type)

2.13.1 Detailed Description

The header file that handles the joystick motor control.

Parameters

joy_type	Tells the robot if it should drive on a linear scale or a parabolic scale

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_joystick_drive.h.

2.14 abs_joystick_drive.h

```

#ifndef ABS_JOYSTICK_DRIVE_H
#define ABS_JOYSTICK_DRIVE_H

void abs_joystick_drive(e_joystick_method joy_type)
{
    //------------------------------
    // robot lift
    //------------------------------
    if(joy1Btn(4) || joy2Btn(4)) motor[sky_hook]=g_robot_lift_up;
    else if(joy1Btn(2))motor[sky_hook]=g_robot_lift_down;
    else motor[sky_hook] = 0;

    //------------------------------
    // drive motor controls
    //------------------------------

    int speed1;
    int speed2;

    int j1 = abs(joystick.joy1_y1);
    int j2 = abs(joystick.joy1_y2);

    if(joy_type == LINEAR)
    {
        speed1 = j1*100/127;
        speed2 = j2*100/127;
    }
    else
    {
        speed1 = ((j1*j1) * 100/(128*128));
        speed2 = ((j2*j2) * 100/(128*128));
    }

    if(joy1Btn(7) || joy1Btn(8))
    {
        speed1 = speed1/6;
        speed2 = speed2/6;
    }
    else if(joy1Btn(5) || joy1Btn(6)){}
    else
    {
        speed1 = speed1/3;
        speed2 = speed2/3;
    }

    if(speed1<10) speed1 = 0;
    if(speed2<10) speed2 = 0;

    if(joystick.joy1_y1<0) speed1 = -speed1;
    if(joystick.joy1_y2<0) speed2 = -speed2;

    motor[right_motor] = speed2;
    motor[left_motor] = speed1;
}

#endif /* !ABS_JOYSTICK_DRIVE_H */
```

2.15 abs_joystick_gunner.h File Reference

The header file that handles the joystick motor control.

Functions

- task **abs_joystick_gunner** ()

2.15.1 Detailed Description

The header file that handles the joystick motor control.

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_joystick_gunner.h.

2.16 abs_joystick_gunner.h

```

#ifndef ABS_JOYSTICK_GUNNER_H
#define ABS_JOYSTICK_GUNNER_H

task abs_joystick_gunner()
{
    while(true)
    {
        //-----------------------------
        // flag motor control
        //-----------------------------
        g_misc = joystick.joy2_TopHat;
        switch(joystick.joy2_TopHat)
        {
        case -1:
            motor[jolly_roger]= 0;
            break;
        case 0:
            motor[jolly_roger] = g_flag_speed_up;
            break;
        case 2:
            motor[jolly_roger] = g_flag_speed_right;
            break;
        case 6:
            motor[jolly_roger] = g_flag_speed_left;
            break;
        case 4:
            motor[jolly_roger] = g_flag_speed_down;
            break;
        }

        //-----------------------------
        // roger slide
        //-----------------------------
        if(joystick.joy2_y2>10) servo[roger_slide] = 255;
        else if(joystick.joy2_y2<-10) servo[roger_slide] = 0;
        else servo[roger_slide] = 127;

        //-----------------------------
        // robot kill switch
        //-----------------------------
        if((joy1Btn(9))&&(joy2Btn(9))&&(joy1Btn(10))&&(joy2Btn(10))) g_program_done = true;

        //-----------------------------
        // block lift
        //-----------------------------
        if(joystick.joy2_y1>10)
        {
            motor[block_lift_motor] = g_block_speed_up;
            motor[block_lift_motor2] = g_block_speed_up;
        }
        else if(joystick.joy2_y1<-10)
        {
            if(joy2Btn(11))
            {
                motor[block_lift_motor] = -100;
                motor[block_lift_motor2] = -100;
            }
            else
            {
                motor[block_lift_motor] = g_block_speed_down;
                motor[block_lift_motor2] = g_block_speed_down;
            }
        }

        else
        {
            motor[block_lift_motor] = 0;
            motor[block_lift_motor2] = 0;
        }

        //-----------------------------
        // block grabber
        //-----------------------------

        if(joy2Btn(1)) //grabber_position = GRABBER_OPEN;
        {
            servo[grabber_left] = GRABBER_LEFT_OPEN;
            servo[grabber_right] = GRABBER_RIGHT_OPEN;
        }
        if(joy2Btn(2)) //grabber_position = GRABBER_MID;
        {
            servo[grabber_left] = GRABBER_LEFT_MID;
            servo[grabber_right] = GRABBER_RIGHT_MID;
        }
        if(joy2Btn(3)) //grabber_position = GRABBER_CLOSE;
        {
            servo[grabber_left] = GRABBER_LEFT_CLOSE;
            servo[grabber_right] = GRABBER_RIGHT_CLOSE;
        }

        if(joy2Btn(5)) servo[grabber_left] = GRABBER_LEFT_OPEN;
        else if(joy2Btn(7)) servo[grabber_left] = GRABBER_LEFT_CLOSE;
        if(joy2Btn(6)) servo[grabber_right] = GRABBER_RIGHT_OPEN;
        else if(joy2Btn(8)) servo[grabber_right] = GRABBER_RIGHT_CLOSE;
    }
}

#endif /* !ABS_JOYSTICK_DRIVE_H */
```

2.17 abs_motor.h File Reference

Allows the robot to move attachments in auto.c.

Functions

- void **abs_motor** (e_motor_move move_type)

2.17.1 Detailed Description

Allows the robot to move attachments in auto.c.

Parameters

move_type	lets the robot know what attchment to move
power	tells the robot how much power it should use on the attachment

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot! FTC Team 5037

Definition in file abs_motor.h.

2.18 abs_motor.h

```

#ifndef ABS_MOTOR_H
#define ABS_MOTOR_H

void abs_motor(e_motor_move move_type)
{
    //------------------------------
    // roger slide
    //------------------------------
    if(move_type == ROGGER_SLIDE) servo[roger_slide] = 0;

    //------------------------------
    // ABDD
    //------------------------------
    if(move_type == ABDD)
    {
        servo[abdd] = g_abdd_up;
        wait10Msec(70);
        servo[abdd] = g_abdd_down;
    }
}

#endif /* !ABS_MOTOR_H */
```

2.19 abs_move_utils.h File Reference

abunch of things that help move the robot

Macros

- #define distance_to_angle_derees(X) ((float)(X∗360/ANGLE_SENSOR_CIRCUMFERENCE))
- #define distance_to_encoder_derees(X) (X∗360/DRIVE_WHEELS_CIRCUMFERENCE)

Enumerations

- enum e_direction { CLOCKWISE, COUNTERCLOCKWISE }
- enum e_drive_direction { FORWARD, BACKWARD }
- enum e_move_stopping_method {
  E_TILT, E_TIME, E_DISTANCE, E_DEGREES,
  E_IR_DETECT, E_IR_DETECT2, E_ANGLE, E_LIGHT }
- enum e_turn_method { SWING, POINT }
- enum e_turn_stopping_method { TURN, TURN_TO }
- enum e_motor_move { ABDD, LIFT, GRABBER, ROGGER_SLIDE }

2.19.1 Detailed Description

abunch of things that help move the robot

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_move_utils.h.

2.19.2 Macro Definition Documentation

2.19.2.1 #define distance_to_angle_derees(*X*) ((float)(X∗360/ANGLE_SENSOR_CIRCUMFERENCE))

converts *X* to degrees

Definition at line 123 of file abs_move_utils.h.

2.19.2.2 #define distance_to_encoder_derees(X) (X∗360/DRIVE_WHEELS_CIRCUMFERENCE)

converts *X* to degrees

Definition at line 129 of file abs_move_utils.h.

2.19.3 Enumeration Type Documentation

2.19.3.1 enum e_direction

This enum is used to let the robot know to turn clockwise or counterclickwise

Tells the robot to drive backwords or forwards onto the ramp

Enumerator

CLOCKWISE turn clockwise

COUNTERCLOCKWISE turn counterclockwise

Definition at line 24 of file abs_move_utils.h.

2.19.3.2 enum e_drive_direction

Tells the robot what direction to drive

Enumerator

FORWARD Drive forward

BACKWARD Drive Backward

Definition at line 37 of file abs_move_utils.h.

2.19.3.3 enum e_motor_move

Tells the robot what motor to move

Enumerator

ABDD Move the ABDD and put in a block

LIFT Move the block lifter

GRABBER close the block grabber

ROGGER_SLIDE Slide the flag liffter back

Definition at line 111 of file abs_move_utils.h.

2.19.3.4 enum e_move_stopping_method

Tells the robot what direction to drive

Enumerator

E_TILT Drive until the robot tilts a certen amount spesified in dist

E_TIME Drive for a set amount of time spesified in dist

E_DISTANCE Drive a certain amount of centameters spesified in dist

E_DEGREES Drive for a certain amount of degrees spesified in dist

E_IR_DETECT Drive until the robot detects the IR becon using the first IR sensor spesified in dist

E_IR_DETECT2 Drive until the robot detects the IR becon using the second IR sensor spesified in dist

E_ANGLE Drive for a certain amount of degrees spesified in dist

E_LIGHT Drive until the light sensor detects the lighting condition specified in dist

Definition at line 62 of file abs_move_utils.h.

2.19.3.5 enum e_turn_method

Tells the robot what type of turn it should do

Enumerator

SWING Perform a swing turn

POINT Perform a point turn

Definition at line 81 of file abs_move_utils.h.

2.19.3.6 enum e_turn_stopping_method

Tells the robot if to should to a certen amount of degreese or just turn

Enumerator

TURN Turn a swing turn

TURN_TO Turn a point turn

Definition at line 94 of file abs_move_utils.h.

2.20 abs_move_utils.h

```
00001
00014 #ifndef ABS_MOVE_UTILS_H
00015 #define ABS_MOVE_UTILS_H
00016
00024 typedef enum
00025 {
00026     CLOCKWISE,
00027     COUNTERCLOCKWISE
00028 } e_direction;
00029
00037 typedef enum
00038 {
00039     FORWARD,
00040     BACKWARD
00041 } e_drive_direction;
00042
00062 typedef enum
00063 {
00064     E_TILT,
00065     E_TIME,
00066     E_DISTANCE,
00067     E_DEGREES,
00068     E_IR_DETECT,
00069     E_IR_DETECT2,
00070     E_ANGLE,
  E_LIGHT
} e_move_stopping_method; //will make a method with a tilt sensor(wheel in the middle of the robot

typedef enum
{
    SWING,
    POINT
} e_turn_method;

typedef enum
{
    TURN,
    TURN_TO
} e_turn_stopping_method;

typedef enum
{
    ABDD,
    LIFT,
    GRABBER,
    ROGGER_SLIDE
} e_motor_move;

#define distance_to_angle_derees(X) ((float)(X*360/ANGLE_SENSOR_CIRCUMFERENCE))

#define distance_to_encoder_derees(X) (X*360/DRIVE_WHEELS_CIRCUMFERENCE)

#endif /* !ABS_TURN_UTILS */
```

2.21 abs_s1_mission_execute.h File Reference

runs the missions from the starting point S1

Functions

- void **abs_s1_mission_execute** ()

2.21.1 Detailed Description

runs the missions from the starting point S1

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_s1_mission_execute.h.

2.22 abs_s1_mission_execute.h

```
#ifndef ABS_S1_MISSION_EXECUTE_H
#define ABS_S1_MISSION_EXECUTE_H

void abs_s1_mission_execute()
{
    switch(g_mission_number)
    {
    case 0:
        g_screen_state = S_ANGLE_SHOW;
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/600, 50, true);
        break;

    case 1:
        g_screen_state = S_ANGLE_SHOW;
        abs_drive(FORWARD, E_IR_DETECT, 7, 40, true);
        PlayTone(200,20);
        wait1Msec(1000);
        //if(g_IR_angle_dist_complete == true) g_end_point = 12;
        if(g_end_point == 2)
        {
            if(HTANGreadAccumulatedAngle(angle_sensor)<(62*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate1_to_turn_dist;
            else if(HTANGreadAccumulatedAngle(angle_sensor)<(100*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate2_to_turn_dist;
            else if(HTANGreadAccumulatedAngle(angle_sensor)<(137*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate3_to_turn_dist;
            else if(HTANGreadAccumulatedAngle(angle_sensor)<(162*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate4_to_turn_dist;
        }
        else if(g_end_point == 3)
        {
            if(HTANGreadAccumulatedAngle(angle_sensor)<(62*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_backwards_crate1_to_turn_dist;
            else if(HTANGreadAccumulatedAngle(angle_sensor)<(100*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_backwards_crate2_to_turn_dist;
            else if(HTANGreadAccumulatedAngle(angle_sensor)<(137*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_backwards_crate3_to_turn_dist;
            else if(HTANGreadAccumulatedAngle(angle_sensor)<(162*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_backwards_crate4_to_turn_dist;
        }
        wait1Msec(500);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 2:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate4_to_turn_dist;
        else g_to_turn_dist = g_forward_crate4_to_turn_dist;
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/150, 50, true);
        wait1Msec(2000);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 3:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate3_to_turn_dist;
        else g_to_turn_dist = g_forward_crate3_to_turn_dist;
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/125, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 4:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate2_to_turn_dist;
        else g_to_turn_dist = g_forward_crate2_to_turn_dist;
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/75, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 5:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate1_to_turn_dist;
        else g_to_turn_dist = g_forward_crate1_to_turn_dist;
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/50, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 6:
        abs_turn(COUNTERCLOCKWISE, POINT, TURN, 75, 60);
        wait1Msec(200);
        abs_drive(FORWARD, E_ANGLE, 190, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN, 75, 60);
        abs_drive(FORWARD, E_ANGLE, 80, 50, true);
        break;

    case 7:
        abs_turn(COUNTERCLOCKWISE, POINT, TURN, 98, 60);
        wait1Msec(200);
        abs_drive(FORWARD, E_ANGLE, 87, 50, true);
        motor[block_lift_motor] = 40;
        motor[block_lift_motor2] = 40;
        abs_turn(CLOCKWISE, POINT, TURN, 103, 60);
        motor[block_lift_motor] = 0;
        motor[block_lift_motor2] = 0;
        abs_drive(FORWARD, E_ANGLE, 80, 50, true);
        break;

    case 140:
        int dist = 30;
        bool done = false;
        while(done == false)
        {
            int ac_start_time = nPgmTime;
            int i = 0;
            while((g_accelermoeter_sensor < dist+5) && (g_accelermoeter_sensor > dist-5) && ((ac_start_time - nPgmTime)<500))
            {
                i++;
                PlayTone(20,20);
                wait1Msec(1);
            }
            if(i > 490) done = true;
            PlayTone(20,20);
        }
        break;
    }
    wait1Msec(g_end_delay*1000);
    switch(g_end_point)
    {
    case 1:
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        abs_stop_robot();
        break;
    case 2:
        abs_end_r1(2000,40);
        break;
    case 3:
        abs_end_r2(2000,40);
        break;
    }
}

#endif /* !ABS_S1_MISSION_EXICUTE_H */
```

2.23 abs_s2_mission_execute.h File Reference

runs the missions from the starting point S2

Functions

- void **abs_s2_mission_execute** ()

2.23.1 Detailed Description

runs the missions from the starting point S2

Parameters

None	n/a

Returns

returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_s2_mission_execute.h.

2.24 abs_s2_mission_execute.h

```

#ifndef ABS_S2_MISSION_EXECUTE_H
#define ABS_S2_MISSION_EXECUTE_H

void abs_s2_mission_execute()
{
    switch(g_mission_number)
    {
    case 0:
        g_screen_state = S_ANGLE_SHOW;
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/600, 50, true);
        break;

    case 1:
        abs_drive(BACKWARD, E_IR_DETECT, 3, 40, true);
        PlayTone(200,20);
        //if(g_IR_angle_dist_complete == true) g_end_point = 12;
        if(g_end_point == 2)
        {
            if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(62* INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate4_to_turn_dist;
            else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(100* INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate3_to_turn_dist;
            else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(137* INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate2_to_turn_dist;
            else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(162* INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_forward_crate1_to_turn_dist;
        }
        else if(g_end_point == 3)
        {
            if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(62* INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_backwards_crate4_to_turn_dist;
            else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(100* INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_backwards_crate3_to_turn_dist;
            else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(137* INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
g_backwards_crate2_to_turn_dist;
            else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(162*INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist = g_backwards_crate1_to_turn_dist;
        }
        wait1Msec(1000);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/6, 50, true);
        wait1Msec(500);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 2:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate4_to_turn_dist;
        else g_to_turn_dist = g_forward_crate4_to_turn_dist;
        abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/40, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 3:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate3_to_turn_dist;
        else g_to_turn_dist = g_forward_crate3_to_turn_dist;
        abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/65, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 4:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate2_to_turn_dist;
        else g_to_turn_dist = g_forward_crate2_to_turn_dist;
        abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/115, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 5:
        if(g_end_point == 3)g_to_turn_dist = g_backwards_crate1_to_turn_dist;
        else g_to_turn_dist = g_forward_crate1_to_turn_dist;
        abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/140, 50, true);
        wait1Msec(2000);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        break;

    case 6:
        abs_turn(CLOCKWISE, POINT, TURN, 75, 60);
        wait1Msec(200);
        abs_drive(FORWARD, E_ANGLE, 190, 50, true);
        abs_turn(COUNTERCLOCKWISE, POINT, TURN, 75, 60);
        abs_drive(FORWARD, E_ANGLE, 80, 50, true);
        break;

    case 7:
        abs_turn(COUNTERCLOCKWISE, POINT, TURN, 98, 60);
        wait1Msec(200);
        abs_drive(FORWARD, E_ANGLE, 87, 50, true);
        motor[block_lift_motor] = 40;
        motor[block_lift_motor2] = 40;
        abs_turn(CLOCKWISE, POINT, TURN, 103, 60);
        motor[block_lift_motor] = 0;
        motor[block_lift_motor2] = 0;
        abs_drive(FORWARD, E_ANGLE, 80, 50, true);
        break;

    case 140:
        int dist = 30;
        bool done = false;
        while(done == false)
        {
            int ac_start_time = nPgmTime;
            int i = 0;
            while((g_accelermoeter_sensor < dist+5) && (g_accelermoeter_sensor > dist-5) && ((ac_start_time - nPgmTime)<500))
            {
                i++;
                PlayTone(20,20);
                wait1Msec(1);
            }
            if(i > 490) done = true;
            PlayTone(20,20);
        }
        break;
    }
    wait1Msec(g_end_delay*1000);
    switch(g_end_point)
    {
    case 1:
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        abs_stop_robot();
        break;
    case 2:
        abs_end_r1(2000,40);
        break;
    case 3:
        abs_end_r2(2000,40);
        break;
    }
}

#endif /* !ABS_S2_MISSION_EXECUTE_H */
```

2.25 abs_s3_mission_execute.h File Reference

runs the missions from the starting point S3

Functions

- void **abs_s3_mission_execute** ()

2.25.1 Detailed Description

runs the missions from the starting point S3

Parameters

None	n/a

Returns

returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_s3_mission_execute.h.

2.26 abs_s3_mission_execute.h

```

#ifndef ABS_S3_MISSION_EXECUTE_H
#define ABS_S3_MISSION_EXECUTE_H

```

```
void abs_s3_mission_execute()
{
    switch(g_mission_number)
    {
    case 1:
        break;

    case 2:
        abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 40, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/100, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3)g_to_turn_dist = 145;
        else g_to_turn_dist = g_forward_crate4_to_turn_dist;
        break;

    case 3:
        abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 40, 35);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/75, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3)g_to_turn_dist = 120;
        else g_to_turn_dist = g_forward_crate3_to_turn_dist;
        break;

    case 4:
        abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/33, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 39, 50);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/25, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3)g_to_turn_dist = 70;
        else g_to_turn_dist = g_forward_crate2_to_turn_dist;
        break;

    case 5:
        abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 35, 60);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3) g_to_turn_dist = 45;
        else if(g_end_point == 2) g_to_turn_dist = g_forward_crate1_to_turn_dist+5;
        break;

    case 6:
        abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 35, 60);
        abs_drive(BACKWARD, E_ANGLE, g_to_turn_dist, 50, true);
        abs_turn(COUNTERCLOCKWISE, POINT, TURN, 90, 60);
        abs_drive(FORWARD, E_ANGLE, 180, 50, true);
        break;

    case 7:
        break;
    }
    wait1Msec(g_end_delay*1000);
    switch(g_end_point)
    {
    case 1:
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        abs_stop_robot();
        break;
    case 2:
        abs_end_r1(2000,40);
        break;
    case 3:
        abs_end_r2(2000,40);
        break;
    }
}

#endif /* !ABS_S3_MISSION_EXECUTE_H */
```

2.27 abs_s4_mission_execute.h File Reference

runs the missions from the starting point S3

Functions

- void **abs_s4_mission_execute** ()

2.27.1 Detailed Description

runs the missions from the starting point S3

Parameters

None	n/a

Returns

returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_s4_mission_execute.h.

2.28 abs_s4_mission_execute.h

```

#ifndef ABS_S4_MISSION_EXECUTE_H
#define ABS_S4_MISSION_EXECUTE_H

void abs_s4_mission_execute()
{
    switch(g_mission_number)
    {
    case 1:
        break;

    case 2:
        abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 128, 60);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3) g_to_turn_dist = g_forward_crate1_to_turn_dist+5;
        else if(g_end_point == 2) g_to_turn_dist = 45;
        break;

    case 3:
        abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/29, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 135, 60);
        abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/23, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3)g_to_turn_dist = g_forward_crate2_to_turn_dist;
        else g_to_turn_dist = 70;
        break;

    case 4:
        abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/29, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 135, 60);
        abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/75, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3)g_to_turn_dist = g_forward_crate3_to_turn_dist;
        else g_to_turn_dist = 120;
        break;

    case 5:
        abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/29, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 135, 60);
        abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/100, 50, true);
        servo[abdd] = g_abdd_up;
        wait1Msec(2000);
        servo[abdd] = g_abdd_down;
        if(g_end_point == 3)g_to_turn_dist = g_forward_crate4_to_turn_dist;
        else g_to_turn_dist = 145;
        break;

    case 6:
        abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
        abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN_TO, 120, 60);
        abs_drive(FORWARD, E_ANGLE, g_to_turn_dist, 50, true);
        abs_turn(CLOCKWISE, POINT, TURN, 90, 60);
        abs_drive(FORWARD, E_ANGLE, 180, 50, true);
        break;

    case 7:
        break;
    }
    wait1Msec(g_end_delay*1000);
    switch(g_end_point)
    {
    case 1:
        wait1Msec(2000);
        servo[abdd] = g_abdd_up;
        abs_stop_robot();
        break;
    case 2:
        abs_end_r1(2000,40);
        break;
    case 3:
        abs_end_r2(2000,40);
        break;
    }
}

#endif /* !ABS_S4_MISSION_EXECUTE_H */
```

2.29 abs_screen.h File Reference

adds a way to put things on the screen

Functions

- task **abs_screen** ()

2.29.1 Detailed Description

adds a way to put things on the screen

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_screen.h.

2.30 abs_screen.h

```

#ifndef ABS_SCREEN_H
#define ABS_SCREEN_H

task abs_screen ()
{
    while(true)
    {
        nxtDisplayBigTextLine(7, "              ");
        switch(g_screen_state)
        {
        case S_CLEAR:
            nxtDisplayBigTextLine(1, "              ");
            nxtDisplayBigTextLine(3, "              ");
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_MISSION:
            nxtDisplayBigTextLine(1, "Misson ","2%d", g_mission_number);
            //nxtDisplayBigTextLine(3, "%2d", g_mission_number);
            nxtDisplayBigTextLine(3, g_mission_names1[g_mission_number]);
            nxtDisplayBigTextLine(5, g_mission_names2[g_mission_number]);
            break;
        case S_DELAY:
            if(g_auto_selection_point == SELECTION_START_DELAY) nxtDisplayBigTextLine(
1, "Start   ");
            else nxtDisplayBigTextLine(1, "Mission ");
            nxtDisplayBigTextLine(3, "Delay");
            nxtDisplayBigTextLine(5, "%2d", g_delay);
            break;
        case S_CAL_TIME:
            nxtDisplayBigTextLine(1, "CalTime");
            nxtDisplayBigTextLine(3, "%2d", g_gyro_cal_time);
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_GYRO_CAL:
            nxtDisplayTextLine(1, "Calibrating");
            nxtDisplayBigTextLine(3, "%2d", (g_gyro_cal_time*1000)-(nPgmTime-g_start_time));
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_READY:
            nxtDisplayBigTextLine(1, "Program");
            nxtDisplayBigTextLine(3, "Ready");
            if(g_auto_grabber_selection_ramp_options == SUB_SELECTION_RAMP_STOP)
nxtDisplayBigTextLine(5, "%1d%1d%1d%1d%1d N", g_start_point, g_start_delay, g_mission_number, g_end_delay,
g_end_point);
            else nxtDisplayBigTextLine(5, "%1d%1d%1d%1d%1d Y", g_start_point, g_start_delay,
g_mission_number, g_end_delay, g_end_point);
            break;
        case S_DELAY_WAIT:
            nxtDisplayBigTextLine(1, "Delay");
            nxtDisplayBigTextLine(3, "%2d", (g_start_delay*1000)-(nPgmTime-g_start_time));
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_GYRO_SHOW:
            nxtDisplayBigTextLine(1, "GyroValue");
            nxtDisplayBigTextLine(3, "%2d", g_const_heading);
            nxtDisplayBigTextLine(5, "%2d", g_rel_heading);
            break;
        case S_IR_SHOW:
            nxtDisplayBigTextLine(1, "IR Value");
            nxtDisplayBigTextLine(3, "%2d  %2d", g_bearing_ac1, g_bearing_ac2);
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_AC_SHOW:
            nxtDisplayBigTextLine(1, "ac Value");
            nxtDisplayBigTextLine(3, "%2d  %2d", g_accelermoeter_sensor, g_misc);
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_ERROR:
            nxtDisplayBigTextLine(1, "ERROR");
            nxtDisplayBigTextLine(3, g_error_list1[g_error]);
            nxtDisplayBigTextLine(5, g_error_list2[g_error]);
            break;
        case S_SMOKE_TEST:
            nxtDisplayBigTextLine(1, "%2d", g_smoke_test_num);
            nxtDisplayBigTextLine(3, g_smoke_test1[g_smoke_test_num]);
            nxtDisplayBigTextLine(5, g_smoke_test2[g_smoke_test_num]);
            break;
        case S_SMOKE_RUN1:
            nxtDisplayBigTextLine(1, g_smoke_test1[g_smoke_test_num]);
            nxtDisplayBigTextLine(3, "%2d", g_test_value);
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_SMOKE_RUN2:
            nxtDisplayBigTextLine(1, g_smoke_test1[g_smoke_test_num]);
            nxtDisplayBigTextLine(3, "%2d  %2d", g_sensor_value, g_sensor_value2);
            nxtDisplayBigTextLine(5, g_sensor_list[g_sensor_num]);
            break;
        case S_SCREEN_CALL:
            nxtDisplayBigTextLine(1, g_smoke_test1[g_smoke_test_num]);
            nxtDisplayBigTextLine(3, "%2d", g_test_value);
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_MISC_SHOW:
            nxtDisplayBigTextLine(1, "misc Value");
            nxtDisplayBigTextLine(3, "%2d", g_misc);
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
        case S_ANGLE_SHOW:
            nxtDisplayBigTextLine(1, "angle Value");
            nxtDisplayBigTextLine(3, "%2d", HTANGreadAccumulatedAngle(angle_sensor));
            nxtDisplayBigTextLine(5, g_mission_names1[0]);
            break;
        case S_STARTING_POINT:
            nxtDisplayBigTextLine(1, "startPnt");
            nxtDisplayBigTextLine(3, g_starting_names1[g_start_point]);
            nxtDisplayBigTextLine(5, g_starting_names2[g_start_point]);
            break;
        case S_ENDING_POINT:
            nxtDisplayBigTextLine(1, "endPoint");
            nxtDisplayBigTextLine(3, g_ending_names1[g_end_point]);
            nxtDisplayBigTextLine(5, g_ending_names2[g_end_point]);
            break;
        case S_SELECTION_SUB_GRABBERS:
            nxtDisplayBigTextLine(1, "Grabbers");
            nxtDisplayBigTextLine(3, "inOrOut?");
            if(g_auto_grabber_selections == SUB_SELECTION_GRABBERS_IN) nxtDisplayBigTextLine(5, g_basic_word_list [1]);
            else if(g_auto_grabber_selections == SUB_SELECTION_GRABBERS_OUT) nxtDisplayBigTextLine(5, g_basic_word_list [2]);
            break;
        case S_SELECTION_SUB_RAMP:
            nxtDisplayBigTextLine(1, "Ramp    ");
            nxtDisplayBigTextLine(3, "continu?");
            if(g_auto_grabber_selection_ramp_options == SUB_SELECTION_RAMP_CONTINUED) nxtDisplayBigTextLine(5, g_basic_word_list [3]);
            else if(g_auto_grabber_selection_ramp_options == SUB_SELECTION_RAMP_STOP) nxtDisplayBigTextLine(5, g_basic_word_list [4]);
            break;
        case S_TIME_SHOW:
            nxtDisplayBigTextLine(1, "T1    T2");
            nxtDisplayBigTextLine(3, "%2d", g_debug_time_1);
            nxtDisplayBigTextLine(5, "%2d", g_debug_time_2);
            break;
        case S_MISSION_SHOW:
            nxtDisplayBigTextLine(1, "numbers");
            nxtDisplayBigTextLine(3, " %1d%1d%1d%1d%1d", g_intput_array[1],g_intput_array[2],g_intput_array[3],g_intput_array[4],g_intput_array[5]);
            nxtDisplayTextLine(5, "%1d-%1d-%1d-%1d", g_start_point, g_start_delay, g_mission_number, g_end_delay, g_end_point);
            break;
        case S_SELECTION_TYPE:
            nxtDisplayBigTextLine(1, "Selecton");
            nxtDisplayBigTextLine(3, "Type:   ");
            if(selection_type == SELECTION_TYPE_CUSTOM) nxtDisplayBigTextLine(5, " custom  ");
            else if(selection_type == SELECTION_TYPE_NUMBER) nxtDisplayBigTextLine(5, "number  ");
            else if(selection_type == SELECTION_TYPE_QUICK) nxtDisplayBigTextLine(5, " quick    ");
            break;
        case S_NUMBER_SELECTION:
            nxtDisplayBigTextLine(1, "Mission");
            nxtDisplayBigTextLine(3, " %1d%1d%1d%1d%1d", g_intput_array[1],g_intput_array[2],g_intput_array[3],g_intput_array[4],g_intput_array[5]);
            switch(g_graph_selection_tab)
            {
            case 1: nxtDisplayBigTextLine(5, " ^      "); break;
            case 2: nxtDisplayBigTextLine(5, "  ^     "); break;
            case 3: nxtDisplayBigTextLine(5, "   ^    "); break;
            case 4: nxtDisplayBigTextLine(5, "    ^   "); break;
            case 5: nxtDisplayBigTextLine(5, "     ^  "); break;
            }
            break;
        case S_QUICK_SELECTION:
            nxtDisplayBigTextLine(1, "Mission");
            nxtDisplayBigTextLine(3, g_quick_names1[g_quick_mission]);
            nxtDisplayBigTextLine(5, g_quick_names2[g_quick_mission]);
            break;
        }
    }
}

#endif
```

2.31 abs_selection_custom.h File Reference

handles the custom selection options for auto mission selection

Functions

- void **abs_selection_custom** ()

2.31.1 Detailed Description

handles the custom selection options for auto mission selection

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_selection_custom.h.

2.32 abs_selection_custom.h

```

#ifndef ABS_SELECTION_CUSTOM_H
#define ABS_SELECTION_CUSTOM_H

void abs_selection_custom()
{
    //---------------------------------------
    // Start point selection
    //---------------------------------------

    g_auto_selection_point = SELECTION_START_POINT;
    g_screen_state = S_STARTING_POINT;

    while(nNxtButtonPressed != kEnterButton)
    {
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(g_start_point < g_auto_starting_points) g_start_point++;
            else g_start_point = g_auto_starting_points;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(g_start_point > 0) g_start_point--;
            else g_start_point = 0;
        }
    }
    PlaySoundFile("! Click.rso");
    while(nNxtButtonPressed == kEnterButton){}
    eraseDisplay();

    //---------------------------------------
    // Start of start time selection
    //---------------------------------------

    g_auto_selection_point = SELECTION_START_DELAY;
    g_screen_state = S_DELAY;

    while(nNxtButtonPressed != kEnterButton)
    {
        g_delay = g_start_delay;
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(g_start_delay < 30) g_start_delay++;
            else g_start_delay = 30;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(g_start_delay > 0) g_start_delay--;
            else g_start_delay = 0;
        }
    }

    PlaySoundFile("! Click.rso");
    while(nNxtButtonPressed == kEnterButton){}

    //---------------------------------------
    // Start of mission selection
    //---------------------------------------

    g_auto_selection_point = SELECTION_MISSION_POINT;
    g_screen_state = S_MISSION;

    while(nNxtButtonPressed != kEnterButton)
    {
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(g_mission_number < g_auto_missions) g_mission_number++;
            else g_mission_number = g_auto_missions;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(g_mission_number > 0) g_mission_number--;
            else g_mission_number = 0;
        }
    }
    PlaySoundFile("! Click.rso");
    while(nNxtButtonPressed == kEnterButton){}
    eraseDisplay();

    //----------------------------------------
    // Start of time selection
    //----------------------------------------

    g_auto_selection_point = SELECTION_MISSION_DELAY;
    g_screen_state = S_DELAY;

    while(nNxtButtonPressed != kEnterButton)
    {
        g_delay = g_end_delay;
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(g_end_delay < 30) g_end_delay++;
            else g_end_delay = 30;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(g_end_delay > 0) g_end_delay--;
            else g_end_delay = 0;
        }
    }

    PlaySoundFile("! Click.rso");
    while(nNxtButtonPressed == kEnterButton){}

    //----------------------------------------
    // Start of end point selection
    //----------------------------------------

    g_screen_state = S_ENDING_POINT;
    g_auto_selection_point = SELECTION_END_POINT;

    while(nNxtButtonPressed != kEnterButton)
    {
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(g_end_point < g_auto_ending_points) g_end_point++;
            else g_end_point = g_auto_ending_points;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(g_end_point > 0) g_end_point--;
            else g_end_point = 0;
        }
    }
    PlaySoundFile("! Click.rso");
    while(nNxtButtonPressed == kEnterButton){}
    eraseDisplay();

    //----------------------------------------
    // Start of optional sub selection for grabbers on the ram
    //----------------------------------------

    if(false)//g_end_point == 2 || g_end_point == 3)
    {
        g_auto_selection_point = SELECTION_SUB_GRABBERS;
        g_screen_state = S_SELECTION_SUB_GRABBERS;

        int i = 1;
        while(nNxtButtonPressed != kEnterButton)
        {
            if(nNxtButtonPressed == kRightButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kRightButton){}
                if(i < 2)
                {
                    i++;
                    g_auto_grabber_selections = SUB_SELECTION_GRABBERS_OUT;
                }
                else
                {
                    g_end_delay = 2;
                    g_auto_grabber_selections = SUB_SELECTION_GRABBERS_OUT;
                }
            }
            if(nNxtButtonPressed == kLeftButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kLeftButton){}
                if(i > 1)
                {
                    i--;
                    g_auto_grabber_selections = SUB_SELECTION_GRABBERS_IN;
                }
                else
                {
                    i = 1;
                    g_auto_grabber_selections = SUB_SELECTION_GRABBERS_IN;
                }
            }
        }
        PlaySoundFile("! Click.rso");
        while(nNxtButtonPressed == kEnterButton){}
    }
}

#endif /* !ABS_SELECTION_CUSTOM_H */
```

2.33 abs_selection_number.h File Reference

handles the number selection options for auto mission selection

Functions

- void **abs_selection_number** ()

2.33.1 Detailed Description

handles the number selection options for auto mission selection

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_selection_number.h.

2.34 abs_selection_number.h

```

#ifndef ABS_SELECTION_NUMBER_H
#define ABS_SELECTION_NUMBER_H

void abs_selection_number()
{
    //---------------------------------------
    // number selection
    //---------------------------------------

    g_auto_selection_point = SELECTION_GRAPH_NUMBER_INPUT;
    g_screen_state = S_NUMBER_SELECTION;

    while(g_graph_selection_tab<5)
    {
        g_graph_selection_tab++;
        while(nNxtButtonPressed != kEnterButton)
        {
            if(nNxtButtonPressed == kRightButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kRightButton){}
                g_intput_array[g_graph_selection_tab] ++;
            }
            if(nNxtButtonPressed == kLeftButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kLeftButton){}
                g_intput_array[g_graph_selection_tab] --;
            }
        }
        while(nNxtButtonPressed == kEnterButton){}
        PlaySoundFile("! Click.rso");
    }
    g_start_point = g_intput_array[1];
    g_start_delay = g_intput_array[2];
    g_mission_number = g_intput_array[3];
    g_end_delay = g_intput_array[4];
    g_end_point = g_intput_array[5];

    g_screen_state = S_MISSION_SHOW;
}

#endif /* !ABS_SELECTION_NUMBER_H */
```

2.35 abs_selection_program.h File Reference

A header file that handles the begining selection for robot actions.

Functions

- void **selection_program** ()

2.35.1 Detailed Description

A header file that handles the begining selection for robot actions.

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_selection_program.h.

2.36 abs_selection_program.h

```

#ifndef ABS_SELECTION_PROGRAM_H
#define ABS_SELECTION_PROGRAM_H

void selection_program()
{
    while(nNxtButtonPressed == kEnterButton){}

    //---------------------------------------
    // number selection, quick selection, or custom selection
    //---------------------------------------

    g_auto_selection_point = SELECTION_SELECTION_TYPE;
    g_screen_state = S_SELECTION_TYPE;

    int j = 1;
    while(nNxtButtonPressed != kEnterButton)
    {
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(j < 3) j++;
            else j = 3;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(j > 1) j--;
            else j = 1;
        }
        switch(j)
        {
        case 1:
            selection_type = SELECTION_TYPE_CUSTOM;
            break;
        case 2:
            selection_type = SELECTION_TYPE_NUMBER;
            break;
            // in for future use
        case 3:
            selection_type = SELECTION_TYPE_QUICK;
            break;
        }
    }
    PlaySoundFile("! Click.rso");
    while(nNxtButtonPressed == kEnterButton){}
    eraseDisplay();

    //---------------------------------------
    // selection executes
    //---------------------------------------
    if(selection_type == SELECTION_TYPE_CUSTOM) abs_selection_custom();
    else if(selection_type == SELECTION_TYPE_NUMBER) abs_selection_number();
    else if(selection_type == SELECTION_TYPE_QUICK) abs_selection_quick();

    //---------------------------------------
    // Start of optional sub selection for ramp position
    //---------------------------------------

    if(g_end_point == 2 || g_end_point == 3)
    {
        g_auto_selection_point = SELECTION_SUB_RAMP;
        g_screen_state = S_SELECTION_SUB_RAMP;

        int i = 1;
        while(nNxtButtonPressed != kEnterButton)
        {
            if(nNxtButtonPressed == kRightButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kRightButton){}
                if(i < 2)
                {
                    i++;
                    g_auto_grabber_selection_ramp_options = SUB_SELECTION_RAMP_CONTINUED;
                }
            }
            if(nNxtButtonPressed == kLeftButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kLeftButton){}
                if(i > 1)
                {
                    i--;
                    g_auto_grabber_selections = SUB_SELECTION_RAMP_STOP;
                }
            }
        }

        PlaySoundFile("! Click.rso");
        while(nNxtButtonPressed == kEnterButton){}
    }

    //---------------------------------------
    // Start of gyro cal selection
    //---------------------------------------

    g_screen_state = S_CAL_TIME;

    while(nNxtButtonPressed != kEnterButton)
    {
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(g_gyro_cal_time < 30) g_gyro_cal_time++;
            else g_gyro_cal_time = 30;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(g_gyro_cal_time > 0) g_gyro_cal_time--;
            else g_gyro_cal_time = 0;
        }
    }
    PlaySoundFile("! Click.rso");
    while(nNxtButtonPressed == kEnterButton){}
    eraseDisplay();
    g_screen_state = S_GYRO_CAL;
}

#endif /* !ABS_SELECTION_PROGRAM_H */
```

2.37 abs_selection_quick.h File Reference

handles the quick selection options for auto mission selection

Functions

- void **abs_selection_quick** ()

2.37.1 Detailed Description

handles the quick selection options for auto mission selection

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_selection_quick.h.

2.38 abs_selection_quick.h

```
#ifndef ABS_SELECTION_QUICK_H
#define ABS_SELECTION_QUICK_H

void abs_selection_quick()
{
    //---------------------------------------
    // quick selection
    //---------------------------------------

    g_auto_selection_point = SELECTION_QUICK_INPUT;
    g_screen_state = S_QUICK_SELECTION;

    while(nNxtButtonPressed != kEnterButton)
    {
        if(nNxtButtonPressed == kRightButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kRightButton){}
            if(g_quick_mission < g_max_quick_missions)g_quick_mission++;
        }
        if(nNxtButtonPressed == kLeftButton)
        {
            PlaySoundFile("! Click.rso");
            while(nNxtButtonPressed == kLeftButton){}
            if(g_quick_mission > 1)g_quick_mission--;
        }
    }
    while(nNxtButtonPressed == kEnterButton){}
    PlaySoundFile("! Click.rso");

    switch(g_quick_mission)
    {
    case 1:
        g_start_point = 1;
        g_start_delay = 0;
        g_mission_number = 1;
        g_end_delay = 0;
        g_end_point = 2;
        break;
    case 2:
        g_start_point = 2;
        g_start_delay = 0;
        g_mission_number = 1;
        g_end_delay = 0;
        g_end_point = 3;
        break;
    }
}

#endif /* !ABS_SELECTION_NUMBER_H */
```

2.39 abs_sensors.h File Reference

A header file that handles the sensors.

Functions

- task **abs_sensors** ()

2.39.1 Detailed Description

A header file that handles the sensors.

Parameters

None	n/a

Returns

returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_sensors.h.

2.40 abs_sensors.h

```

#ifndef ABS_SENSOR_H
#define ABS_SENSOR_H

task abs_sensors()
{
    g_prev_time = nPgmTime;

    while(true)
    {
        //-------------------------
        // Light Sensor
        //-------------------------
//  g_light_sensor = SensorValue(lightSensor);
        //-------------------------
        // HiTechnic IR Sensor
        //-------------------------
        g_bearing_ac1 = HTIRS2readACDir(HTIRS2);               // Read the IR bearing from the sensor
        g_curr_dir1 = (float) g_bearing_ac1;

        HTIRS2readAllACStrength(HTIRS2, g_acs1[0], g_acs1[1], g_acs1[2], g_acs1[3], g_acs1[4]);
        //-----------------------------------
        // code for the peaks of IR sensor
        //-----------------------------------
        if (g_bearing_ac1!=0)                              // we have a valid IR signal
        {
            int maximum = -1;
            int peak = 0, offset=0;
            for (int i=0;i<5;i++)   // scan array to find the peak entry
            {   if (g_acs1[i]>maximum)
                {
                    peak = i;
                    maximum = g_acs1[i];
                }
            }
            offset=0;
            if ((peak < 4) && (peak>0) && (g_acs1[peak] != 0))  // we are not working with extreme value
            {
                if (g_acs1[peak-1]!=g_acs1[peak+1]) // if the values either side of the peak are identical then peak is peak
                {
                    if (g_acs1[peak-1]>g_acs1[peak+1])  // otherwise decide which side has higher signal
                    {
                        offset = -25*(1-(float)(g_acs1[peak]-g_acs1[peak-1])/       // calculate the bias away from the peak
                        max(g_acs1[peak], g_acs1[peak-1]));
                    }
                    else
                    {
                        offset = 25*(1-(float)(g_acs1[peak]-g_acs1[peak+1])/
                        max(g_acs1[peak], g_acs1[peak+1]));
                    }
                }
            }
            g_ir_bearing1 = (float)((peak-2)*50) + offset;     // direction is the total of the peak bias plus the adjacent bias
            //nxtDisplayBigTextLine(3, "%2d", g_ir_bearing1);
        }
        //-------------------------
        // HiTechnic IR Sensor 2
        //-------------------------
        g_bearing_ac2 = HTIRS2readACDir(HTIRS2_2);             // Read the IR bearing from the sensor
        g_curr_dir2 = (float) g_bearing_ac2;

        HTIRS2readAllACStrength(HTIRS2_2, g_acs2[0], g_acs2[1], g_acs2[2], g_acs2[3], g_acs2[4]);
        //------------------------------------
        // code for the peaks of IR sensor 2
        //------------------------------------
        if (g_bearing_ac2!=0)                              // we have a valid IR signal
        {
            int maximum = -1;
            int peak = 0, offset=0;
            for (int i=0;i<5;i++)   // scan array to find the peak entry
            {   if (g_acs2[i]>maximum)
                {
                    peak = i;
                    maximum = g_acs2[i];
                }
            }
            offset=0;
            if ((peak < 4) && (peak>0) && (g_acs2[peak] != 0))  // we are not working with extreme value
            {
                if (g_acs2[peak-1]!=g_acs2[peak+1]) // if the values either side of the peak are identical then peak is peak
                {
                    if (g_acs2[peak-1]>g_acs2[peak+1])  // otherwise decide which side has higher signal
                    {
                        offset = -25*(1-(float)(g_acs2[peak]-g_acs2[peak-1])/       // calculate the bias away from the peak
                        max(g_acs2[peak], g_acs2[peak-1]));
                    }
                    else
                    {
                        offset = 25*(1-(float)(g_acs2[peak]-g_acs2[peak+1])/
                        max(g_acs2[peak], g_acs2[peak+1]));
                    }
                }
            }
            g_ir_bearing2 = (float)((peak-2)*50) + offset;     // direction is the total of the peak bias plus the adjacent bias
            //nxtDisplayBigTextLine(3, "%2d", g_ir_bearing1);
        }
        //-------------------------
        // HiTechnic Gyro
        //-------------------------

        g_curr_time=nPgmTime;
        g_raw_gyro = HTGYROreadRot(HTGYRO);
        g_const_heading += (g_raw_gyro - g_drift) * (float)(g_curr_time-g_prev_time)/1000;
        g_rel_heading += (g_raw_gyro - g_drift) * (float)(g_curr_time-g_prev_time)/1000;
        g_prev_time = g_curr_time;

        g_recont_heading = g_const_heading % 360;
        if(g_recont_heading<0) g_recont_heading += 360;

        //-------------------------
        // HiTechnic accelermoeter
        //-------------------------

        HTACreadAllAxes(HTAC, g_x_axis, g_y_axis, g_z_axis);
        g_accelermoeter_sensor = g_x_axis;

        if(g_sensor_reference_drive == true)
        {
            g_accelermoeter_reads++;
            g_accelermoeter_array[g_accelermoeter_reads%50]=g_accelermoeter_sensor;
            for(int i=0;i<30;i++)
            {
                g_accelermoeter_total_value = g_accelermoeter_array[i];
            }
            g_accelermoeter_average = g_accelermoeter_total_value/50;
        }
        else
        {
            g_accelermoeter_reads = 0;
            g_accelermoeter_total_value = 0;
            g_accelermoeter_average = 0;
            memset(g_accelermoeter_array,0,30);
        }
        //-------------------------
        // HiTechnic angle sensor
        //-------------------------
        //if(g_reset_angle == true) HTANGresetAccumulatedAngle(HTANG);
        //else angle_sensor = HTANGreadAccumulatedAngle(HTANG);
    }
}
#endif
```

2.41 abs_smoke_execute.h File Reference

executes commands sent in smoke test

Functions

- void **abs_smoke_execute** ()

2.41.1 Detailed Description

executes commands sent in smoke test

Parameters

None	n/a

Returns

Returns nothing

Copyright

Definition in file abs_smoke_execute.h.

2.42 abs_smoke_execute.h

```
#ifndef ABS_SMOKE_EXECUTE_H
#define ABS_SMOKE_EXECUTE_H

void abs_smoke_execute ()
{
    g_screen_state = S_SMOKE_RUN1;
    while(nNxtButtonPressed != kEnterButton)
    {
        switch(g_smoke_test_num)
        {
            //---------------------------------
            // Jolly Roger
            //---------------------------------
        case 1:
            if(nNxtButtonPressed == kLeftButton)
            {
                motor[jolly_roger] = g_flag_speed_down;
                g_test_value = g_flag_speed_down;
            }
            else if(nNxtButtonPressed == kRightButton)
            {
                motor[jolly_roger] = g_flag_speed_up;
                g_test_value = g_flag_speed_up;
            }
            else
            {
                g_test_value = 0;
                motor[jolly_roger] = 0;
            }
            break;
            //---------------------------------
            // Drive
            //---------------------------------
        case 2:
            if(nNxtButtonPressed == kLeftButton)
            {
                motor[right_motor] = 60;
                motor[left_motor] = 60;
                g_test_value = 60;
            }
            else if(nNxtButtonPressed == kRightButton)
            {
                motor[right_motor] = -60;
                motor[left_motor] = -60;
                g_test_value = -60;
            }
            else
            {
                g_test_value = 0;
                motor[right_motor] = 0;
                motor[left_motor] = 0;
            }
            break;
            //---------------------------------
            // sensors
            //---------------------------------
        case 3:
            g_screen_state = S_SMOKE_RUN2;
            if(nNxtButtonPressed == kLeftButton)
            {
                if(g_test_value > 1) g_test_value--;
                while(nNxtButtonPressed == kLeftButton) {}
            }
            if(nNxtButtonPressed == kRightButton)
            {
                if(g_test_value < g_sensor_max) g_test_value++;
                while(nNxtButtonPressed == kRightButton) {}
            }
            g_sensor_num = g_test_value;
            switch(g_sensor_num)
            {
            case ST_GYRO:
                g_sensor_value = g_rel_heading;
                break;
            case ST_IR:
                g_sensor_value = g_bearing_ac1;
                g_sensor_value2 = g_bearing_ac2;
                break;
            case ST_TILT:
                g_sensor_value = HTANGreadAccumulatedAngle(angle_sensor);
                break;
            case ST_ACCELEROMETER:
                g_sensor_value = g_accelermoeter_sensor;
                break;
            }
            break;
            //---------------------------------
            // Block lift
            //---------------------------------
        case 4:
            if(nNxtButtonPressed == kLeftButton)
            {
                motor[block_lift_motor] = g_robot_lift_down;
                motor[block_lift_motor2] = g_robot_lift_down;
                g_test_value = g_robot_lift_down;
            }
            else if(nNxtButtonPressed == kRightButton)
            {
                motor[block_lift_motor] = g_robot_lift_up;
                motor[block_lift_motor2] = g_robot_lift_up;
                g_test_value = g_robot_lift_up;
            }
            else
            {
                motor[block_lift_motor] = 0;
                motor[block_lift_motor2] = 0;
                g_test_value = 0;
            }
            break;
            //---------------------------------
            // grabbers
            //---------------------------------
        case 5:
            if(nNxtButtonPressed == kLeftButton)
            {
                if(g_test_value>1) g_test_value--;
                while(nNxtButtonPressed == kLeftButton){}
            }
            if(nNxtButtonPressed == kRightButton)
            {
                if(g_test_value<3) g_test_value++;
                while(nNxtButtonPressed == kRightButton){}
            }
            switch(g_test_value)
            {
            case 1:
            servo[grabber_left] = GRABBER_LEFT_OPEN;
            servo[grabber_right] = GRABBER_RIGHT_OPEN;
                break;
            case 2:
            servo[grabber_left] = GRABBER_LEFT_MID;
            servo[grabber_right] = GRABBER_RIGHT_MID;
                break;
            case 3:
            servo[grabber_left] = GRABBER_LEFT_CLOSE;
            servo[grabber_right] = GRABBER_RIGHT_CLOSE;
                break;
            }
        break;
        //---------------------------------
        // sky hook
        //---------------------------------
    case 6:
        if(nNxtButtonPressed == kLeftButton)
        {
            motor[sky_hook] = g_robot_lift_up;
            g_test_value = g_robot_lift_up;
        }
        else if(nNxtButtonPressed == kRightButton)
        {
            motor[sky_hook] = g_robot_lift_down;
            g_test_value = g_robot_lift_down;
        }
        else
        {
            motor[sky_hook] = 0;
            g_test_value = 0;
        }
        break;
        //---------------------------------
        // roger slide
        //---------------------------------
    case 7:
        if(nNxtButtonPressed == kLeftButton)
        {
            servo[roger_slide] = 255;
            g_test_value = 255;
        }
        else if(nNxtButtonPressed == kRightButton)
        {
            servo[roger_slide] = 0;
            g_test_value = 0;
        }
        else
        {
            servo[roger_slide] = 127;
            g_test_value = 127;
        }
        break;
    }
}
PlaySoundFile("! Click.rso");
}

#endif /* !ABS_SMOKE_EXECUTE_H */
```

2.43 abs_stop_robot.h File Reference

when called stops the robot from moving

Functions

- void **abs_stop_robot** ()

2.43.1 Detailed Description

when called stops the robot from moving

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot! FTC Team 5037

Definition in file abs_stop_robot.h.

2.44 abs_stop_robot.h

```

#ifndef ABS_STOP_ROBOT_H
#define ABS_STOP_ROBOT_H
```

```

void abs_stop_robot()
{
    motor[left_motor] = 0;
    motor[right_motor] = 0;
    motor[block_lift_motor] = 0;
    motor[block_lift_motor2] = 0;
    motor[sky_hook] = 0;
    servo[roger_slide] = 127;
}

#endif /* !ABS_STOP_ROBOT_H */
```

2.45 abs_tele_op_initialize.h File Reference

does some important stuff before we do the teleop program

Functions

- void **abs_tele_op_initialize** ()

2.45.1 Detailed Description

does some important stuff before we do the teleop program

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_tele_op_initialize.h.

2.46 abs_tele_op_initialize.h

```

#ifndef ABS_TELE_OP_INITIALIZE_H
#define ABS_TELE_OP_INITIALIZE_H

void abs_tele_op_initialize()
{
    if(joystick.joy1_TopHat == -1) g_joy1_enabled = true;
    if(joystick.joy2_TopHat == -1) g_joy2_enabled = true;

    servo[abdd] = g_abdd_down;

    StartTask(screen);
    g_screen_state = S_MISC_SHOW;
    getJoystickSettings(joystick);
}
#endif /* ABS_TELE_OP_INITIALIZE_H */
```

2.47 abs_teleop_utils.h File Reference

utils for teleop

Enumerations

- enum e_joystick_method { LINEAR, PARABOLIC }

2.47.1 Detailed Description

utils for teleop

Parameters

None	n/a

Returns

returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_teleop_utils.h.

2.47.2 Enumeration Type Documentation

2.47.2.1 enum e_joystick_method

tells the robot if it should drive linear or parabolic

Enumerator

LINEAR Drive linear

PARABOLIC Drive parabolic

Definition at line 23 of file abs_teleop_utils.h.

2.48 abs_teleop_utils.h

```
00001
00014 #ifndef ABS_TELEOP_UTILS_H
00015 #define ABS_TELEOP_UTILS_H
00016
00023 typedef enum
00024 {
00025     LINEAR,
00026     PARABOLIC
00027 } e_joystick_method;
00028
00029 #endif /* !ABS_TELEOP_UTILS */
```

2.49 abs_turn.h File Reference

The header file that alows you to do a point turn.

Functions

- void abs_turn (e_direction dir, e_turn_method turn_method, e_turn_stopping_method e_stop, int degree, int speed)

2.49.1 Detailed Description

The header file that alows you to do a point turn.

Parameters

degree	Tells the robot how much to turn
dir	Tells the robot what way to turn
speed	Tells the robot how fast to turn

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_turn.h.

2.49.2 Function Documentation

2.49.2.1 void abs_turn (e_direction *dir,* e_turn_method *turn_method,* e_turn_stopping_method *e_stop,* int *degree,* int *speed*)

macros

Definition at line 26 of file abs_turn.h.

References COUNTERCLOCKWISE, SWING, TURN, and TURN_TO.

2.50 abs_turn.h

```
00001
00018 #ifndef ABS_TURN_H
00019 #define ABS_TURN_H
00020
00023 //=======================================
00024 // point turn
00025 //=======================================
00026 void abs_turn(e_direction dir, e_turn_method turn_method,
      e_turn_stopping_method e_stop, int degree, int speed)
00027 {
00028     int i = 0;
00029     g_rel_heading = 0;
00030     int target = 0;
```

```

    if(e_stop == TURN_TO)
    {
        if(dir == COUNTERCLOCKWISE)
        {
            if(degree<g_recont_heading) target = -(g_recont_heading-degree);
            else target = -(360-(degree-g_recont_heading));
        }
        else
        {
            if(degree<g_recont_heading) target = 360-(g_recont_heading-degree);
            else target = degree-g_recont_heading;
        }
        abs_turn(dir, turn_method, TURN, target, speed);
        PlaySoundFile("! Click.rso");
    }
    else
    {
        //-------------------------
        // swing turn
        //-------------------------
        if(turn_method == SWING)
        {
            if(dir == COUNTERCLOCKWISE)
            {
                motor[right_motor] = speed;
                motor[left_motor] = 0;
            }
            else
            {
                motor[right_motor] = 0;
                motor[left_motor] = speed;
            }
        }

        //-------------------------
        // point turn
        //-------------------------
        else
        {
            if(dir == COUNTERCLOCKWISE)
            {
                motor[right_motor] = speed;
                motor[left_motor] = -speed;
            }
            else
            {
                motor[right_motor] = -speed;
                motor[left_motor] = speed;
            }
        }
    }
    //-------------------------
    // turn condition
    //-------------------------

    if(e_stop == TURN)
    {
        while(i < 5)
        {
            if (abs(g_rel_heading) > abs(degree)) i++;
            nxtDisplayCenteredBigTextLine(5, "%d", g_recont_heading);
        }
        motor[right_motor] = 0;
        motor[left_motor] = 0;
    }
}

#endif /* !ABS_TURN_H */
```

2.51 auto.c File Reference

The automatic program for the robot.

```
#include "joystickdriver.c"
#include "lib/xander/hitechnic-sensormux.h"
#include "lib/xander/hitechnic-irseeker-v2.h"
#include "lib/xander/hitechnic-gyro.h"
#include "lib/xander/hitechnic-angle.h"
#include "lib/xander/hitechnic-accelerometer.h"
#include "lib/global_varaibles.h"
#include "lib/abs_selection_number.h"
#include "lib/abs_selection_custom.h"
#include "lib/abs_selection_quick.h"
#include "lib/abs_selection_program.h"
#include "lib/abs_screen.h"
#include "lib/abs_gyro_cal.h"
#include "lib/math_utils.h"
#include "lib/abs_sensors.h"
#include "abs_move_utils.h"
#include "lib/abs_turn.h"
#include "abs_gyro_drive.h"
#include "lib/abs_drive.h"
#include "lib/abs_initialize.h"
#include "lib/abs_motor.h"
#include "lib/abs_stop_robot.h"
#include "lib/abs_end_r1.h"
#include "lib/abs_end_r2.h"
#include "lib/abs_s1_mission_execute.h"
#include "lib/abs_s2_mission_execute.h"
#include "lib/abs_s3_mission_execute.h"
#include "lib/abs_s4_mission_execute.h"
```

Functions

- task **main** ()

2.51.1 Detailed Description

The automatic program for the robot.

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file auto.c.

2.52 auto.c

```
#pragma config(Hubs,  S1, HTMotor,  HTMotor,  HTMotor,  HTServo)
#pragma config(Sensor, S1,      ,               sensorI2CMuxController)
#pragma config(Sensor, S2,      GYRO_MUX,       sensorI2CCustom)
#pragma config(Sensor, S3,      SENSOR_MUX,     sensorI2CCustom)
#pragma config(Sensor, S4,      angle_sensor,   sensorI2CCustom)
#pragma config(Motor,  mtr_S1_C1_1,     block_lift_motor, tmotorTetrix, openLoop, encoder)
#pragma config(Motor,  mtr_S1_C1_2,     sky_hook,      tmotorTetrix, openLoop, reversed, encoder)
```

```
#pragma config(Motor,  mtr_S1_C2_1,     jolly_roger,   tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C2_2,     block_lift_motor2, tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C3_1,     right_motor,   tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C3_2,     left_motor,    tmotorTetrix, openLoop)
#pragma config(Servo,  srvo_S1_C4_1,    grabber_right,        tServoStandard)
#pragma config(Servo,  srvo_S1_C4_2,    grabber_left,         tServoStandard)
#pragma config(Servo,  srvo_S1_C4_3,    roger_slide,          tServoContinuousRotation)
#pragma config(Servo,  srvo_S1_C4_4,    light_sensor,         tServoStandard)
#pragma config(Servo,  srvo_S1_C4_5,    servo5,               tServoNone)
#pragma config(Servo,  srvo_S1_C4_6,    abdd,                 tServoStandard)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*Includes*/

//------------------------
// sensor/mux/joystick includes
//------------------------

#include "joystickdriver.c"
#include "lib/xander/hitechnic-sensormux.h"
#include "lib/xander/hitechnic-irseeker-v2.h"
#include "lib/xander/hitechnic-gyro.h"
#include "lib/xander/hitechnic-angle.h"
#include "lib/xander/hitechnic-accelerometer.h"

//------------------------
// custom functions includes
//------------------------

#include "lib/global_varaibles.h"
#include "lib/abs_selection_number.h"
#include "lib/abs_selection_custom.h"
#include "lib/abs_selection_quick.h"
#include "lib/abs_selection_program.h"
#include "lib/abs_screen.h"
#include "lib/abs_gyro_cal.h"
#include "lib/math_utils.h"
#include "lib/abs_sensors.h"
#include "abs_move_utils.h"
#include "lib/abs_turn.h"
#include "abs_gyro_drive.h"
#include "lib/abs_drive.h"
#include "lib/abs_initialize.h"
#include "lib/abs_motor.h"
#include "lib/abs_stop_robot.h"

//------------------------
// auto mission includes
//------------------------

#include "lib/abs_end_r1.h"
#include "lib/abs_end_r2.h"

#include "lib/abs_s1_mission_execute.h"
#include "lib/abs_s2_mission_execute.h"
#include "lib/abs_s3_mission_execute.h"
#include "lib/abs_s4_mission_execute.h"

//========================================
// Main program
//========================================

task main()
{
    initialize();
    while(true)
    {
        g_rel_heading = 0;
        switch(g_start_point)
        {
        case 1:
            abs_s1_mission_execute();
            break;
        case 2:
            abs_s2_mission_execute();
            break;
        case 3:
            abs_s3_mission_execute();
            break;
        case 4:
            abs_s4_mission_execute();
            break;
        }
    }
}
```

2.53 global_varaibles.h File Reference

varaibles that are global

Macros

- #define INT_ANGLE_SENSOR_CIRCUMFERENCE 18
- #define FLOAT_ANGLE_SENSOR_CIRCUMFERENCE 17.6
- #define DRIVE_WHEELS_CIRCUMFERENCE 26
- #define GRABBER_LEFT_OPEN 3
- #define GRABBER_RIGHT_OPEN 245
- #define GRABBER_LEFT_MID 60
- #define GRABBER_RIGHT_MID 180
- #define GRABBER_LEFT_CLOSE 120
- #define GRABBER_RIGHT_CLOSE 131
- #define ST_GYRO 1
- #define ST_IR 2
- #define ST_ACCELEROMETER 3
- #define ST_TILT 4
- #define S_CLEAR 0
- #define S_MISSION 1
- #define S_DELAY 2
- #define S_CAL_TIME 3
- #define S_GYRO_CAL 4
- #define S_READY 5
- #define S_DELAY_WAIT 6
- #define S_GYRO_SHOW 7
- #define S_ERROR 8
- #define S_SMOKE_TEST 9
- #define S_SMOKE_RUN1 10
- #define S_SMOKE_RUN2 11
- #define S_SMOKE_RUN3 12
- #define S_SCREEN_CALL 13
- #define S_IR_SHOW 14
- #define S_AC_SHOW 15
- #define S_MISC_SHOW 16
- #define S_STARTING_POINT 17
- #define S_ENDING_POINT 18
- #define S_SELECTION_SUB_GRABBERS 19
- #define S_ANGLE_SHOW 20
- #define S_TIME_SHOW 21
- #define S_SELECTION_TYPE 22
- #define S_NUMBER_SELECTION 23
- #define **S_SELECTION_SUB_RAMP** 24
- #define **S_MISSION_SHOW** 25

- #define **S_QUICK_SELECTION** 26
- #define ERR_NONE 0
- #define ERR_GYRO_CAL 1
- #define ERR_GYRO_MUX 2
- #define ERR_SENSOR_MUX 3
- #define ERR_JOYSTICKS 4
- #define ERR_ACCELERMOETER 5

Enumerations

- enum e_auto_selection_points {
SELECTION_START_POINT, SELECTION_START_DELAY, SELECTION_MISSION_POINT, SELECTION_MISSION_DELAY,
SELECTION_END_POINT, SELECTION_SUB_GRABBERS, SELECTION_GYRO_CAL, SELECTION_SELECTION_TYPE,
SELECTION_GRAPH_NUMBER_INPUT, **SELECTION_QUICK_INPUT**, **SELECTION_SUB_RAMP** }
- enum e_selection_types { SELECTION_TYPE_NUMBER, SELECTION_TYPE_CUSTOM, SELECTION_TYPE_QUICK }
- enum e_auto_sub_selection { SUB_SELECTION_GRABBERS_OUT, SUB_SELECTION_GRABBERS_IN }
- enum e_auto_sub_selection_ramp { SUB_SELECTION_RAMP_STOP, SUB_SELECTION_RAMP_CONTINUED }

Variables

- const tMUXSensor **HTIRS2** = msensor_S3_1
- const tMUXSensor **HTAC** = msensor_S3_2
- const tMUXSensor **HTGYRO** = msensor_S2_1
- const tMUXSensor **HTIRS2_2** = msensor_S3_3
- const tMUXSensor **LEGOLS** = msensor_S3_4
- bool **g_gyro_true** = false
- const int **g_block_speed_down** = -60
- const int **g_block_speed_up** = 100
- const int **g_robot_lift_down** = -40
- const int **g_robot_lift_up** = 100
- const int **g_flag_speed_down** = 90
- const int **g_flag_speed_right** = 20
- const int **g_flag_speed_up** = -90
- const int **g_flag_speed_left** = -20
- const int **g_abdd_up** = 10
- const int **g_abdd_down** = 235
- const int **g_gyro_adjust** = 10
- e_auto_selection_points **g_auto_selection_point** = SELECTION_START_POINT
- e_selection_types **selection_type** = SELECTION_TYPE_CUSTOM
- e_auto_sub_selection **g_auto_grabber_selections** = SUB_SELECTION_GRABBERS_IN
- e_auto_sub_selection_ramp **g_auto_grabber_selection_ramp_options** = SUB_SELECTION_RAMP_STOP
- int **g_to_turn_dist** = 0
- bool **g_IR_angle_dist_complete** = false
- const int **g_forward_crate1_to_turn_dist** = 135
- const int **g_forward_crate2_to_turn_dist** = 110
- const int **g_forward_crate3_to_turn_dist** = 60

- const int **g_forward_crate4_to_turn_dist** = 35
- const int **g_backwards_crate1_to_turn_dist** = 45
- const int **g_backwards_crate2_to_turn_dist** = 70
- const int **g_backwards_crate3_to_turn_dist** = 120
- const int **g_backwards_crate4_to_turn_dist** = 145
- int **g_smoke_test_num** = 1
- int **g_smoke_test_total** = 12
- int **g_smoke_run** = false
- int **g_test_value** = 0
- int **g_intput_array** [6]
- int **g_debug_time_1** = 0
- int **g_debug_time_2** = 0
- int **g_auto_ending_points** = 4
- int **g_travel_dist** = 0
- int **g_auto_starting_points** = 4
- int **g_auto_missions** = 10
- int **g_drive_heading** = 0
- int **g_ir_heading** = 5
- bool **g_program_done** = false
- bool **g_joy1_enabled** = false
- bool **g_joy2_enabled** = false
- int **g_selection_value** = 0
- int **g_end_point** = 1
- int **g_start_point** = 1
- int **g_mission_number** = 1
- int **g_delay** = 0
- int **g_end_delay** = 0
- int **g_start_delay** = 0
- int **g_gyro_cal_time** = 5
- int **g_gyro_noise** = 0
- long **g_start_time** = 0
- int **g_drift** = 0
- float **g_const_heading** = 0
- float **g_rel_heading** = 0
- long **g_curr_time** = 0
- long **g_prev_time** = 0
- int **g_raw_gyro** = 0
- int **g_recont_heading** = 0
- int g_light_sensor
- int g_bearing_ac1 = 0
- int g_bearing_ac2 = 0
- float g_ir_bearing1 = 0.0
- float g_ir_bearing2 = 0.0
- int **g_acs1** [5]
- int **g_acs2** [5]
- float **g_curr_dir1** = 0.0
- float **g_curr_dir2** = 0.0
- int **g_misc** = 0
- bool **g_reset_angle** = false
- int **g_accelermoeter_sensor** = 0
- int **g_x_axis** = 0

- int **g_y_axis** = 0
- int **g_z_axis** = 0
- const int **g_target_angle** = 110
- ubyte **g_accelermoeter_reads** = 0
- int **g_accelermoeter_array** [] = {0,1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}
- ubyte **g_accelermoeter_total_value** = 0
- int **g_accelermoeter_average** = 0
- int **g_sensor_num** = 1
- int **g_sensor_max** = 4
- int **g_sensor_value** = 0
- int **g_sensor_value2** = 0
- bool **g_sensor_reference_drive** = false
- string **g_sensor_list** []
- string **g_basic_word_list** []
- int **g_screen_state** = 1
- int **g_graph_selection_tab** = 0
- long **g_graph_selection_number** = 10000
- int **g_error** = 0
- string **g_starting_names1** []
- string **g_starting_names2** []
- string **g_ending_names1** []
- string **g_ending_names2** []
- string **g_mission_names1** []
- string **g_mission_names2** []
- string **g_quick_names1** []
- string **g_quick_names2** []
- int **g_quick_mission** = 1
- int **g_max_quick_missions** = 6
- string **g_error_list1** []
- string **g_error_list2** []
- string **g_smoke_test1** []
- string **g_smoke_test2** []

2.53.1 Detailed Description

varaibles that are global

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file global_varaibles.h.

2.53.2 Macro Definition Documentation

2.53.2.1 #define DRIVE_WHEELS_CIRCUMFERENCE 26

Tells the robot the cercumference of the drive wheels

Definition at line 57 of file global_varaibles.h.

2.53.2.2 #define ERR_ACCELERMOETER 5

Tells the robot that theres a error with the accelermoeter

Definition at line 480 of file global_varaibles.h.

2.53.2.3 #define ERR_GYRO_CAL 1

Tells the robot that theres a error with the gyro calibrate

Definition at line 476 of file global_varaibles.h.

2.53.2.4 #define ERR_GYRO_MUX 2

Tells the robot that theres a error with the gyro mux

Definition at line 477 of file global_varaibles.h.

2.53.2.5 #define ERR_JOYSTICKS 4

Tells the robot that theres a error with the joysticks

Definition at line 479 of file global_varaibles.h.

2.53.2.6 #define ERR_NONE 0

Tells the robot that theres no error

Definition at line 475 of file global_varaibles.h.

2.53.2.7 #define ERR_SENSOR_MUX 3

Tells the robot that theres a error with the sensor mux

Definition at line 478 of file global_varaibles.h.

2.53.2.8 #define FLOAT_ANGLE_SENSOR_CIRCUMFERENCE 17.6

Tells the robot the exact circumference of the angle sensors wheel

Definition at line 56 of file global_varaibles.h.

2.53.2.9 #define GRABBER_LEFT_CLOSE 120

tells the robot where the left block grabber needs to be to be closed

Definition at line 63 of file global_varaibles.h.

2.53.2.10 #define GRABBER_LEFT_MID 60

tells the robot where the left block grabber needs to be to be in the middle

Definition at line 61 of file global_varaibles.h.

2.53.2.11 #define GRABBER_LEFT_OPEN 3

tells the robot where the left block grabber needs to be to be open

Definition at line 59 of file global_varaibles.h.

2.53.2.12 #define GRABBER_RIGHT_CLOSE 131

tells the robot where the left block grabber needs to be to be closed

Definition at line 64 of file global_varaibles.h.

2.53.2.13 #define GRABBER_RIGHT_MID 180

tells the robot where the right block grabber needs to be to be in the middle

Definition at line 62 of file global_varaibles.h.

2.53.2.14 #define GRABBER_RIGHT_OPEN 245

tells the robot where the right block grabber needs to be to be open

Definition at line 60 of file global_varaibles.h.

2.53.2.15 #define INT_ANGLE_SENSOR_CIRCUMFERENCE 18

Tells the robot the circumference of the angle sensors wheel

Definition at line 55 of file global_varaibles.h.

2.53.2.16 #define S_AC_SHOW 15

Tells the robot the screen state number for this screen statestate

Definition at line 434 of file global_varaibles.h.

2.53.2.17 #define S_ANGLE_SHOW 20

Tells the robot the screen state number for this screen statestate

Definition at line 439 of file global_varaibles.h.

2.53.2.18 #define S_CAL_TIME 3

Tells the robot the screen state number for this screen statestate

Definition at line 422 of file global_varaibles.h.

2.53.2.19 #define S_CLEAR 0

Tells the robot the screen state number for this screen statestate

Definition at line 419 of file global_varaibles.h.

2.53.2.20 #define S_DELAY 2

Tells the robot the screen state number for this screen statestate

Definition at line 421 of file global_varaibles.h.

2.53.2.21 #define S_DELAY_WAIT 6

Tells the robot the screen state number for this screen statestate

Definition at line 425 of file global_varaibles.h.

2.53.2.22 #define S_ENDING_POINT 18

Tells the robot the screen state number for this screen statestate

Definition at line 437 of file global_varaibles.h.

2.53.2.23 #define S_ERROR 8

Tells the robot the screen state number for this screen statestate

Definition at line 427 of file global_varaibles.h.

2.53.2.24 #define S_GYRO_CAL 4

Tells the robot the screen state number for this screen statestate

Definition at line 423 of file global_varaibles.h.

2.53.2.25 #define S_GYRO_SHOW 7

Tells the robot the screen state number for this screen statestate

Definition at line 426 of file global_varaibles.h.

2.53.2.26 #define S_IR_SHOW 14

Tells the robot the screen state number for this screen statestate

Definition at line 433 of file global_varaibles.h.

2.53.2.27 #define S_MISC_SHOW 16

Tells the robot the screen state number for this screen statestate

Definition at line 435 of file global_varaibles.h.

2.53.2.28 #define S_MISSION 1

Tells the robot the screen state number for this screen statestate

Definition at line 420 of file global_varaibles.h.

2.53.2.29 #define S_NUMBER_SELECTION 23

Tells the robot the screen state number for this screen statestate

Definition at line 442 of file global_varaibles.h.

2.53.2.30 #define S_READY 5

Tells the robot the screen state number for this screen statestate

Definition at line 424 of file global_varaibles.h.

2.53.2.31 #define S_SCREEN_CALL 13

Tells the robot the screen state number for this screen statestate

Definition at line 432 of file global_varaibles.h.

2.53.2.32 #define S_SELECTION_SUB_GRABBERS 19

Tells the robot the screen state number for this screen statestate

Definition at line 438 of file global_varaibles.h.

2.53.2.33 #define S_SELECTION_TYPE 22

Tells the robot the screen state number for this screen statestate

Definition at line 441 of file global_varaibles.h.

2.53.2.34 #define S_SMOKE_RUN1 10

Tells the robot the screen state number for this screen statestate

Definition at line 429 of file global_varaibles.h.

2.53.2.35 #define S_SMOKE_RUN2 11

Tells the robot the screen state number for this screen statestate

Definition at line 430 of file global_varaibles.h.

2.53.2.36 #define S_SMOKE_RUN3 12

Tells the robot the screen state number for this screen statestate

Definition at line 431 of file global_varaibles.h.

2.53.2.37 #define S_SMOKE_TEST 9

Tells the robot the screen state number for this screen statestate

Definition at line 428 of file global_varaibles.h.

2.53.2.38 #define S_STARTING_POINT 17

Tells the robot the screen state number for this screen statestate

Definition at line 436 of file global_varaibles.h.

2.53.2.39 #define S_TIME_SHOW 21

Tells the robot the screen state number for this screen statestate

Definition at line 440 of file global_varaibles.h.

2.53.2.40 #define ST_ACCELEROMETER 3

The reference value for the sensor in smoke test

Definition at line 344 of file global_varaibles.h.

2.53.2.41 #define ST_GYRO 1

The reference value for the sensor in smoke test

Definition at line 342 of file global_varaibles.h.

2.53.2.42 #define ST_IR 2

The reference value for the sensor in smoke test

Definition at line 343 of file global_varaibles.h.

2.53.2.43 #define ST_TILT 4

The reference value for the sensor in smoke test

Definition at line 345 of file global_varaibles.h.

2.53.3 Enumeration Type Documentation

2.53.3.1 enum e_auto_selection_points

Tells the robot what part it is in the selection program

Enumerator

SELECTION_START_POINT Tells the robot to go to this part in the selection program

SELECTION_START_DELAY Tells the robot to go to this part in the selection program

SELECTION_MISSION_POINT Tells the robot to go to this part in the selection program

SELECTION_MISSION_DELAY Tells the robot to go to this part in the selection program

SELECTION_END_POINT Tells the robot to go to this part in the selection program

SELECTION_SUB_GRABBERS Tells the robot to go to this part in the selection program

SELECTION_GYRO_CAL Tells the robot to go to this part in the selection program

SELECTION_SELECTION_TYPE Tells the robot to go to this part in the selection program

SELECTION_GRAPH_NUMBER_INPUT Tells the robot to go to this part in the selection program

Definition at line 134 of file global_varaibles.h.

2.53.3.2 enum e_auto_sub_selection

Enumerator

SUB_SELECTION_GRABBERS_OUT turn clockwise drive with the grabbers out

SUB_SELECTION_GRABBERS_IN turn counterclockwise drive with the grabbers in

Definition at line 183 of file global_varaibles.h.

2.53.3.3 enum e_auto_sub_selection_ramp

Tells the robot to drive onto the ramp and continue or stop

Enumerator

SUB_SELECTION_RAMP_STOP Stop on the ramp

SUB_SELECTION_RAMP_CONTINUED Continue on the ramp

Definition at line 199 of file global_varaibles.h.

2.53.3.4 enum e_selection_types

Lets the robot know how you wan to implement the auto program

Enumerator

SELECTION_TYPE_NUMBER Select a program by id

SELECTION_TYPE_CUSTOM Select one of the custom programs

SELECTION_TYPE_QUICK Select one of the most commenly used progams

Definition at line 163 of file global_varaibles.h.

2.53.4 Variable Documentation

2.53.4.1 string g_basic_word_list[ ]

Initial value:

```
= {
    "unknown ",
    "in      ",
    "out     ",
    "yes     ",
    "no      "}
```

Definition at line 356 of file global_varaibles.h.

2.53.4.2 g_bearing_ac1 = 0

the raw value from the first IR sensor

Definition at line 302 of file global_varaibles.h.

2.53.4.3 g_bearing_ac2 = 0

the raw value from the second IR sensor

Definition at line 303 of file global_varaibles.h.

2.53.4.4 string g_ending_names1[ ]

Initial value:

```
= {
    "        ",
    "Stop    ",
    "Ramp 1  ",
    "Ramp 2  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 543 of file global_varaibles.h.

2.53.4.5 string g_ending_names2[ ]

Initial value:

```
= {
    "        ",
    "        ",
    "        ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 571 of file global_varaibles.h.

2.53.4.6 string g_error_list1[ ]

Initial value:

```
= {
    "Unknown ",
    "GyroCal ",
    "Gyro    ",
    "Sensor  ",
    "joystick",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 679 of file global_varaibles.h.

2.53.4.7 string g_error_list2[ ]

Initial value:

```
= {
    "error   ",
    "Failure ",
    "Mux     ",
    "Mux     ",
    "fail    ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 704 of file global_varaibles.h.

2.53.4.8 g_ir_bearing1 = 0.0

the calibrated value from the first IR sensor

Definition at line 304 of file global_varaibles.h.

2.53.4.9 g_ir_bearing2 = 0.0

the calibrated value from the second IR sensor

Definition at line 305 of file global_varaibles.h.

2.53.4.10 g_light_sensor

Sensor variables

holds the value of the light sensor

Definition at line 301 of file global_varaibles.h.

2.53.4.11 string g_mission_names1[ ]

Initial value:

```
= {
    "        ",
    "IR crate",
    "crate 4 ",
    "crate 3 ",
    "crate 2 ",
    "crate 1 ",
    "defence ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 599 of file global_varaibles.h.

2.53.4.12 string g_mission_names2[ ]

Initial value:

```
= {
    "        ",
    "Test 1  ",
    "Test 2  ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "score 4 ",
    "score 3 ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 627 of file global_varaibles.h.

2.53.4.13 string g_quick_names1[ ]

Initial value:

```
= {
    "Unknown ",
    "S1 IR E1",
    "S1 IR E2",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  "}
```

Definition at line 655 of file global_varaibles.h.

2.53.4.14 string g_quick_names2[ ]

Initial value:

```
= {
    "Unknown ",
    "Test 1  ",
    "Test 2  ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  "}
```

Definition at line 664 of file global_varaibles.h.

2.53.4.15 string g_sensor_list[ ]

Initial value:

```
= {
    "unknown ",
    "gyro    ",
    "IR   IR2",
    "accel   ",
    "tilt    "}
```

Definition at line 349 of file global_varaibles.h.

2.53.4.16 string g_smoke_test1[ ]

Initial value:

```
= {
    "Unknown ",
    "Jolly   ",
    "Drive   ",
    "Sensor  ",
    "Block   ",
    "Grabbers",
    "sky hook",
    "roger   ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 732 of file global_varaibles.h.

2.53.4.17 string g_smoke_test2[ ]

Initial value:

```
= {
    "Unknown ",
    "Roger   ",
    "Train   ",
    "Sensor  ",
    "Lift    ",
    "        ",
    "        ",
    "slide   ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 760 of file global_varaibles.h.

2.53.4.18 string g_starting_names1[ ]

Initial value:

```
= {
    "        ",
    "S1      ",
    "S2      ",
    "S3      ",
    "S4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 487 of file global_varaibles.h.

2.53.4.19 string g_starting_names2[ ]

Initial value:

```
= {
    "        ",
    "        ",
    "Test 2  ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "}
```

Definition at line 515 of file global_varaibles.h.

2.54 global_varaibles.h

```
#pragma systemFile // treat as system file to eliminate warnings for unused variables

//
//============================================================
// Define sensor multiplexor connectivity and port allocations
//============================================================

const tMUXSensor HTIRS2 = msensor_S3_1;     // HiTechnic Infrared sensor
const tMUXSensor HTAC = msensor_S3_2;
const tMUXSensor HTGYRO = msensor_S2_1;    // HiTechnic GYRO sensor
const tMUXSensor HTIRS2_2 = msensor_S3_3;     // HiTechnic Infrared sensor 2
//const tMUXSensor HTANG = msensor_S3_3;
const tMUXSensor LEGOLS = msensor_S3_4;

bool g_gyro_true = false;

//=========================================================
// Robot constants
//=========================================================

#define INT_ANGLE_SENSOR_CIRCUMFERENCE 18
#define FLOAT_ANGLE_SENSOR_CIRCUMFERENCE 17.6
#define DRIVE_WHEELS_CIRCUMFERENCE 26

#define GRABBER_LEFT_OPEN 3
#define GRABBER_RIGHT_OPEN 245
#define GRABBER_LEFT_MID 60
#define GRABBER_RIGHT_MID 180
#define GRABBER_LEFT_CLOSE 120
#define GRABBER_RIGHT_CLOSE 131

const int g_block_speed_down = -60;
const int g_block_speed_up = 100;

const int g_robot_lift_down = -40;
const int g_robot_lift_up = 100;

const int g_flag_speed_down = 90;
const int g_flag_speed_right = 20;
const int g_flag_speed_up = -90;
const int g_flag_speed_left = -20;

const int g_abdd_up = 10;
const int g_abdd_down = 235;

const int g_gyro_adjust = 10;

//=========================================================
// auto selection points
//=========================================================
typedef enum
{
    SELECTION_START_POINT,
    SELECTION_START_DELAY,
    SELECTION_MISSION_POINT,
    SELECTION_MISSION_DELAY,
    SELECTION_END_POINT,
    SELECTION_SUB_GRABBERS,
    SELECTION_GYRO_CAL,
    SELECTION_SELECTION_TYPE,
    SELECTION_GRAPH_NUMBER_INPUT,
    SELECTION_QUICK_INPUT,
    SELECTION_SUB_RAMP
} e_auto_selection_points;

e_auto_selection_points g_auto_selection_point = SELECTION_START_POINT;

//=========================================================
// auto selection type options
//=========================================================
typedef enum
{
    SELECTION_TYPE_NUMBER,
    SELECTION_TYPE_CUSTOM,
    SELECTION_TYPE_QUICK
} e_selection_types;

```

```
e_selection_types selection_type = SELECTION_TYPE_CUSTOM;

//=========================================================
// auto sub selections
//=========================================================
typedef enum
{
    SUB_SELECTION_GRABBERS_OUT,
    SUB_SELECTION_GRABBERS_IN
} e_auto_sub_selection;

e_auto_sub_selection g_auto_grabber_selections = SUB_SELECTION_GRABBERS_IN;

typedef enum
{
    SUB_SELECTION_RAMP_STOP,
    SUB_SELECTION_RAMP_CONTINUED
} e_auto_sub_selection_ramp;

e_auto_sub_selection_ramp g_auto_grabber_selection_ramp_options = SUB_SELECTION_RAMP_STOP;

//=========================================================
// auto movements
//=========================================================
int g_to_turn_dist = 0;

bool g_IR_angle_dist_complete = false;

const int g_forward_crate1_to_turn_dist = 135;
const int g_forward_crate2_to_turn_dist = 110;
const int g_forward_crate3_to_turn_dist = 60;
const int g_forward_crate4_to_turn_dist = 35;

const int g_backwards_crate1_to_turn_dist = 45;
const int g_backwards_crate2_to_turn_dist = 70;
const int g_backwards_crate3_to_turn_dist = 120;
const int g_backwards_crate4_to_turn_dist = 145;

//=========================================================
// Smoke test varaibles
//=========================================================

int g_smoke_test_num = 1;
int g_smoke_test_total = 12;
int g_smoke_run = false;
int g_test_value = 0;

//=========================================================
// auto number input variable
//=========================================================

int g_intput_array[6];

//=========================================================
// Misc
//=========================================================

int g_debug_time_1 = 0;
int g_debug_time_2 = 0;

int g_auto_ending_points = 4;
int g_travel_dist = 0;
int g_auto_starting_points = 4;
int g_auto_missions = 10;
int g_drive_heading = 0;
int g_ir_heading = 5;
bool g_program_done = false;

bool g_joy1_enabled = false;
bool g_joy2_enabled = false;

int g_selection_value = 0;

//=============================================================
// Define user configurable parameters
//=============================================================
int g_end_point = 1;
int g_start_point = 1;
int g_mission_number = 1;
int g_delay = 0;
int g_end_delay = 0;
int g_start_delay = 0;
int g_gyro_cal_time = 5;

//==============================================================
// Gyro variables
//==============================================================
int g_gyro_noise = 0;
long g_start_time = 0;
int g_drift = 0;
float g_const_heading = 0;
float g_rel_heading = 0;
long g_curr_time = 0;
long g_prev_time = 0;
int g_raw_gyro = 0;
int g_recont_heading = 0; //this is the recalculated const gyro heading

int g_light_sensor;
int g_bearing_ac1 = 0;
int g_bearing_ac2 = 0;
float g_ir_bearing1 = 0.0;
float g_ir_bearing2 = 0.0;
int g_acs1[5];
int g_acs2[5];
float g_curr_dir1 = 0.0;
float g_curr_dir2 = 0.0;
int g_misc = 0;
bool g_reset_angle = false;

//-----------------------------
// accelermoeter variables
//-----------------------------
int g_accelermoeter_sensor = 0;
int g_x_axis = 0;
int g_y_axis = 0;
int g_z_axis = 0;
const int g_target_angle = 110;
ubyte g_accelermoeter_reads = 0;
int g_accelermoeter_array [] = {0,1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30};
ubyte g_accelermoeter_total_value = 0;
int g_accelermoeter_average = 0;

int g_sensor_num = 1;
int g_sensor_max = 4;
int g_sensor_value = 0;
int g_sensor_value2 = 0;
#define ST_GYRO 1
#define ST_IR 2
#define ST_ACCELEROMETER 3
#define ST_TILT 4

bool g_sensor_reference_drive = false;

string g_sensor_list [] = {
    "unknown ",
    "gyro    ",
    "IR   IR2",
    "accel   ",
    "tilt    "};

string g_basic_word_list [] = {
    "unknown ",
    "in      ",
    "out     ",
    "yes     ",
    "no      "};

//==============================================================
// Define screen related variables
//==============================================================
#define S_CLEAR 0
#define S_MISSION 1
#define S_DELAY 2
#define S_CAL_TIME 3
#define S_GYRO_CAL 4
#define S_READY 5
#define S_DELAY_WAIT 6
#define S_GYRO_SHOW 7
#define S_ERROR 8
```

```
#define S_SMOKE_TEST 9
#define S_SMOKE_RUN1 10
#define S_SMOKE_RUN2 11
#define S_SMOKE_RUN3 12
#define S_SCREEN_CALL 13
#define S_IR_SHOW 14
#define S_AC_SHOW 15
#define S_MISC_SHOW 16
#define S_STARTING_POINT 17
#define S_ENDING_POINT 18
#define S_SELECTION_SUB_GRABBERS 19
#define S_ANGLE_SHOW 20
#define S_TIME_SHOW 21
#define S_SELECTION_TYPE 22
#define S_NUMBER_SELECTION 23
#define S_SELECTION_SUB_RAMP 24
#define S_MISSION_SHOW 25
#define S_QUICK_SELECTION 26

int g_screen_state = 1;

//==============================================================
// Define graph selection variables
//==============================================================

int g_graph_selection_tab = 0;
long g_graph_selection_number = 10000;

//==============================================================
// Define error numbers
//==============================================================
#define ERR_NONE 0
#define ERR_GYRO_CAL 1
#define ERR_GYRO_MUX 2
#define ERR_SENSOR_MUX 3
#define ERR_JOYSTICKS 4
#define ERR_ACCELERMOETER 5

int g_error = 0;

//==============================================================================
// Define the text to be displayed for each starting point line 1
//==============================================================================
string g_starting_names1 [] = {
    "        ",
    "S1      ",
    "S2      ",
    "S3      ",
    "S4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//==============================================================================
// Define the text to be displayed for each starting point line 2
//==============================================================================
string g_starting_names2 [] = {
    "        ",
    "        ",
    "Test 2  ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//===============================================================================
// Define the text to be displayed for each ending point line 1
//===============================================================================
string g_ending_names1 [] = {
    "        ",
    "Stop    ",
    "Ramp 1  ",
    "Ramp 2  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//===============================================================================
// Define the text to be displayed for each ending point line 2
//===============================================================================
string g_ending_names2 [] = {
    "        ",
    "        ",
    "        ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//===============================================================================
// Define the text to be displayed for each mission
//===============================================================================
string g_mission_names1 [] = {
    "        ",
    "IR crate",
    "crate 4 ",
    "crate 3 ",
    "crate 2 ",
    "crate 1 ",
    "defence ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//==============================================================================
// Define the text to be displayed on the second line for each mission
//==============================================================================
string g_mission_names2 [] = {
    "        ",
    "Test 1  ",
    "Test 2  ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "score 4 ",
    "score 3 ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//==============================================================================
// Define the text to be displayed for quick selection
//==============================================================================
string g_quick_names1 [] = {
    "Unknown ",
    "S1 IR E1",
    "S1 IR E2",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  "};

string g_quick_names2 [] = {
    "Unknown ",
    "Test 1  ",
    "Test 2  ",
    "Test 3  ",
    "Test 4  ",
    "Test 5  ",
    "Test 6  "};

int g_quick_mission = 1;
int g_max_quick_missions = 6;

//==============================================================================
// Define the text to be displayed for the errors
//==============================================================================
string g_error_list1 [] = {
    "Unknown ",
    "GyroCal ",
    "Gyro    ",
    "Sensor  ",
    "joystick",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

string g_error_list2 [] = {
    "error   ",
    "Failure ",
    "Mux     ",
    "Mux     ",
    "fail    ",
    "Test 5  ",
    "Test 6  ",
    "Test 7  ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//===========================================================================
// Define the text to be displayed for smoke test line 1
//===========================================================================
string g_smoke_test1 [] = {
    "Unknown ",
    "Jolly   ",
    "Drive   ",
    "Sensor  ",
    "Block   ",
    "Grabbers",
    "sky hook",
    "roger   ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};

//===========================================================================
// Define the text to be displayed for smoke test line 2
//===========================================================================
string g_smoke_test2 [] = {
    "Unknown ",
    "Roger   ",
    "Train   ",
    "Sensor  ",
    "Lift    ",
    "        ",
    "        ",
    "slide   ",
    "Test 8  ",
    "Test 9  ",
    "Test 10 ",
    "Test 11 ",
    "Test 12 ",
    "Test 13 ",
    "Test 14 ",
    "Test 15 ",
    "Test 16 ",
    "Test 17 ",
    "Test 18 ",
    "Test 19 ",
    "Test 20 ",
    "Test 21 ",
    "Test 22 "};
```

2.55 math_utils.h File Reference

a collection of math operations

Macros

- #define gyro_degrees(X) (X)
- #define product(X, Y) ((X) $*$ (Y))
- #define sum(X, Y) ((X) + (Y))
- #define min(X, Y) ((X) $<$ (Y) ? (X) : (Y))
- #define max(X, Y) ((X) $>$ (Y) ? (X) : (Y))

2.55.1 Detailed Description

a collection of math operations

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file math_utils.h.

2.55.2 Macro Definition Documentation

2.55.2.1 #define gyro_degrees(*X*) (X)

macros

```
converts gyro value @a X to degrees
```

Definition at line 23 of file math_utils.h.

2.55.2.2 #define max(*X*, *Y*) ((X) > (Y) ? (X) : (Y))

computes the maximum of *X* and *Y*

Definition at line 47 of file math_utils.h.

2.55.2.3 #define min(*X*, *Y*) ((X) < (Y) ? (X) : (Y))

computes the minimum of *X* and *Y*

Definition at line 41 of file math_utils.h.

2.55.2.4 #define product(*X*, *Y*) ((X) * (Y))

computes the product of *X* and *Y*

Definition at line 29 of file math_utils.h.

2.55.2.5 #define sum(*X*, *Y*) ((X) + (Y))

computes the sum of *X* and *Y*

Definition at line 35 of file math_utils.h.

2.56 math_utils.h

```
00001
00014 #ifndef MATH_UTILS_H
00015 #define MATH_UTILS_H
00016
00023 #define gyro_degrees(X)(X)
00024
00029 #define product(X, Y) ((X) * (Y))
00030
00035 #define sum(X, Y) ((X) + (Y))
00036
00041 #define min(X, Y) ((X) < (Y) ? (X) : (Y))
00042
00047 #define max(X, Y) ((X) > (Y) ? (X) : (Y))
00048
00049 //#define range(X, Y) ((X) > (Y) ? (X) : (Y))
00050
00051 #endif /* !MATH_UTILS_H */
```

2.57 smoke_test.c File Reference

The automatic program for the robot.

```
#include "JoystickDriver.c"
#include "lib/xander/hitechnic-sensormux.h"
#include "lib/xander/hitechnic-gyro.h"
#include "lib/xander/hitechnic-angle.h"
#include "lib/xander/hitechnic-irseeker-v2.h"
#include "drivers/hitechnic-accelerometer.h"
#include "lib/global_varaibles.h"
#include "lib/abs_screen.h"
#include "lib/math_utils.h"
#include "lib/abs_sensors.h"
#include "lib/abs_smoke_execute.h"
```

Functions

- task **main** ()

2.57.1 Detailed Description

The automatic program for the robot.

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file smoke_test.c.

2.58 smoke_test.c

```
#pragma config(Hubs,  S1, HTMotor,  HTMotor,  HTMotor,  HTServo)
#pragma config(Sensor, S1,     ,               sensorI2CMuxController)
#pragma config(Sensor, S2,     GYRO_MUX,       sensorI2CCustom)
#pragma config(Sensor, S3,     SENSOR_MUX,     sensorI2CCustom)
#pragma config(Sensor, S4,     angle_sensor,   sensorI2CCustom)
#pragma config(Motor,  mtr_S1_C1_1,     block_lift_motor, tmotorTetrix, openLoop, encoder)
#pragma config(Motor,  mtr_S1_C1_2,     sky_hook,      tmotorTetrix, openLoop, reversed, encoder)
#pragma config(Motor,  mtr_S1_C2_1,     jolly_roger,   tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C2_2,     block_lift_motor2, tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C3_1,     right_motor,   tmotorTetrix, openLoop, reversed)
#pragma config(Motor,  mtr_S1_C3_2,     left_motor,    tmotorTetrix, openLoop)
#pragma config(Servo,  srvo_S1_C4_1,    grabber_right,        tServoStandard)
#pragma config(Servo,  srvo_S1_C4_2,    grabber_left,         tServoStandard)
#pragma config(Servo,  srvo_S1_C4_3,    roger_slide,          tServoContinuousRotation)
#pragma config(Servo,  srvo_S1_C4_4,    light_sensor,         tServoStandard)
#pragma config(Servo,  srvo_S1_C4_5,    servo5,               tServoNone)
#pragma config(Servo,  srvo_S1_C4_6,    abdd,                 tServoStandard)
//*!!Code automatically generated by ’ROBOTC’ configuration wizard               !!*//

//-----------------------
// sensor/mux/joystick includes
//-----------------------

#include "JoystickDriver.c"
#include "lib/xander/hitechnic-sensormux.h"
#include "lib/xander/hitechnic-gyro.h"
#include "lib/xander/hitechnic-angle.h"
#include "lib/xander/hitechnic-irseeker-v2.h"
#include "drivers/hitechnic-accelerometer.h"

//-----------------------
// custom functions includes
//-----------------------
```

```
#include "lib/global_varaibles.h"
#include "lib/abs_screen.h"
#include "lib/math_utils.h"
#include "lib/abs_sensors.h"
#include "lib/abs_smoke_execute.h"

//==================================
// main program
//==================================
task main()
{
    StartTask(screen);
    StartTask(abs_sensors);
    g_test_value = 1;
    while(true)
    {
        while(nNxtButtonPressed == kEnterButton){}
        g_screen_state = S_SMOKE_TEST;
        //--------------------------------------
        // Start of mission selection
        //--------------------------------------

        while(nNxtButtonPressed != kEnterButton)
        {
            if(nNxtButtonPressed == kRightButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kRightButton){}
                if(g_smoke_test_num < g_smoke_test_total) g_smoke_test_num++;
                }
            if(nNxtButtonPressed == kLeftButton)
            {
                PlaySoundFile("! Click.rso");
                while(nNxtButtonPressed == kLeftButton){}
                if(g_smoke_test_num > 1) g_smoke_test_num--;
            }
        }
        PlaySoundFile("! Click.rso");
        while(nNxtButtonPressed == kEnterButton){}
        eraseDisplay();

        abs_smoke_execute();
    }
}
```

2.59 tele_op.c File Reference

The tele_op program for the robot.

```
#include "JoystickDriver.c"
#include "lib/xander/hitechnic-sensormux.h"
#include "drivers/hitechnic-accelerometer.h"
#include "lib/xander/hitechnic-irseeker-v2.h"
#include "lib/xander/hitechnic-gyro.h"
#include "lib/xander/hitechnic-angle.h"
#include "lib/global_varaibles.h"
#include "lib/abs_screen.h"
#include "lib/abs_teleop_utils.h"
#include "lib/abs_joystick_drive.h"
#include "lib/abs_joystick_gunner.h"
#include "lib/abs_tele_op_initialize.h"
```

Functions

- task **main** ()

2.59.1 Detailed Description

The tele_op program for the robot.

Parameters

None	n/a

Returns

Returns nothing

Copyright

Copyright 2013, Got Robot? FTC Team 5037

Definition in file tele_op.c.

2.60 tele_op.c

```
#pragma config(Hubs,  S1, HTMotor,  HTMotor,  HTMotor,  HTServo)
#pragma config(Sensor, S2,     GYRO_MUX,       sensorI2CCustom)
#pragma config(Sensor, S3,     SENSOR_MUX,     sensorI2CCustom)
#pragma config(Sensor, S4,     angle_sensor,   sensorI2CCustom)
#pragma config(Motor,  mtr_S1_C1_1,     block_lift_motor, tmotorTetrix, openLoop, encoder)
#pragma config(Motor,  mtr_S1_C1_2,     sky_hook,      tmotorTetrix, openLoop, reversed, encoder)
#pragma config(Motor,  mtr_S1_C2_1,     jolly_roger,   tmotorTetrix, openLoop, reversed)
#pragma config(Motor,  mtr_S1_C2_2,     block_lift_motor2, tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C3_1,     right_motor,   tmotorTetrix, openLoop, reversed)
#pragma config(Motor,  mtr_S1_C3_2,     left_motor,    tmotorTetrix, openLoop)
#pragma config(Servo,  srvo_S1_C4_1,    grabber_right,        tServoStandard)
#pragma config(Servo,  srvo_S1_C4_2,    grabber_left,         tServoStandard)
#pragma config(Servo,  srvo_S1_C4_3,    roger_slide,          tServoContinuousRotation)
#pragma config(Servo,  srvo_S1_C4_4,    light_sensor,         tServoStandard)
#pragma config(Servo,  srvo_S1_C4_6,    abdd,                 tServoStandard)
//*!!Code automatically generated by ’ROBOTC’ configuration wizard               !!*//

//-----------------------
// sensor/mux/joystick includes
//-----------------------

#include "JoystickDriver.c"
#include "lib/xander/hitechnic-sensormux.h"
#include "drivers/hitechnic-accelerometer.h"
#include "lib/xander/hitechnic-irseeker-v2.h"
#include "lib/xander/hitechnic-gyro.h"
#include "lib/xander/hitechnic-angle.h"

//-----------------------
// custom functions includes
//-----------------------

#include "lib/global_varaibles.h"
#include "lib/abs_screen.h"
#include "lib/abs_teleop_utils.h"
#include "lib/abs_joystick_drive.h"
#include "lib/abs_joystick_gunner.h"
#include "lib/abs_tele_op_initialize.h"
//========================================
// Main program
//========================================

task main ()
{
    abs_tele_op_initialize();
    StartTask(abs_joystick_gunner);
    while(g_program_done==false)
    {
        abs_joystick_drive(LINEAR);
    }
}
```