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
```

```
00022 #ifndef ABS_DRIVE_H
00023 #define ABS_DRIVE_H
00024
00025 void abs_drive(e_drive_direction dir, e_move_stopping_method
      dist_method, int dist, int speed, bool stop_at_end)
00026 {
00027     HTANGresetAccumulatedAngle(angle_sensor);
00028     int i = 0;
00029
00030     nMotorEncoder(right_motor)= 0;
00031     g_rel_heading = 0;
00032
00033     //------------------------
00034     // time stopping method
00035     //------------------------
00036     if(dist_method == E_TIME)
00037     {
00038         ClearTimer(T1);
00039         while(time1[T1] < dist)
00040         {
00041             abs_gyro_drive(speed,dir);
00042         }
00043     }
00044     //------------------------
00045     // encoder stopping method
00046     //------------------------
00047     else if(dist_method == E_DEGREES)
00048     {
00049         while(i<5)
00050         {
00051             if(abs(nMotorEncoder(right_motor)) > distance_to_encoder_derees(dist)
      ) i++;
00052             abs_gyro_drive(speed,dir);
00053         }
00054     }
00055     //------------------------
00056     // IR stopping method
00057     //------------------------
00058     else if(dist_method == E_IR_DETECT)
00059     {
00060         if(dir == FORWARD)
00061         {
00062             while(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (150*
      INT_ANGLE_SENSOR_CIRCUMFERENCE))
00063             {
00064                 if(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (100*
      INT_ANGLE_SENSOR_CIRCUMFERENCE))
00065                 {
00066                     if(!((g_bearing_ac2 >= dist - 1) || (
      g_bearing_ac2 == 0))) break;
00067                 }
00068                 else
00069                 {
00070                     if(!((g_bearing_ac2 >= dist) || (g_bearing_ac2 == 0))) break;
00071                 }
00072                 abs_gyro_drive(speed,dir);
00073             }
00074             //g_screen_state = S_TIME_SHOW;
00075             g_debug_time_1 = nPgmTime;
00076         }
00077         else if(dir == BACKWARD)
00078         {
00079             while(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (150*
      INT_ANGLE_SENSOR_CIRCUMFERENCE))
00080             {
00081                 if(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (100*
      INT_ANGLE_SENSOR_CIRCUMFERENCE))
00082                 {
00083                     if(!((g_bearing_ac1 <= dist + 1) || (
      g_bearing_ac1 == 0))) break;
00084                 }
00085                 else
00086                 {
00087                     if(!((g_bearing_ac1 <= dist) || (g_bearing_ac1 == 0))) break;
00088                 }
00089                 abs_gyro_drive(speed,dir);
00090             }
00091             //g_screen_state = S_TIME_SHOW;
00092             g_debug_time_1 = nPgmTime;
00093         }
00094     }
```

```
00095     //------------------------
00096     // IR stopping method 2
00097     //------------------------
00098     else if(dist_method == E_IR_DETECT2)
00099     {
00100         if(dir == FORWARD)
00101         {
00102             while(g_ir_bearing2 > dist)
00103             {
00104                 abs_gyro_drive(speed,dir);
00105             }
00106         }
00107         else
00108         {
00109             while(g_ir_bearing2 < dist)
00110             {
00111                 abs_gyro_drive(speed,dir);
00112             }
00113         }
00114     }
00115     //------------------------
00116     // accelermeoter sensor stopping method
00117     //------------------------
00118     else if(dist_method == E_TILT)
00119     {
00120         int j = 0;
00121         g_sensor_reference_drive = true;
00122         while(j<30)
00123         {
00124             abs_gyro_drive(speed,dir);
00125             if(g_accelermoeter_average > dist) j++;
00126         }
00127         g_sensor_reference_drive = false;
00128     }
00129     //------------------------
00130     // angle sensor stopping method
00131     //------------------------
00132     else
00133     {
00134         HTANGresetAccumulatedAngle(angle_sensor);
00135         while(abs(HTANGreadAccumulatedAngle(angle_sensor)) < (dist*
      INT_ANGLE_SENSOR_CIRCUMFERENCE))
00136         {
00137             abs_gyro_drive(speed,dir);
00138         }
00139     }
00140     if(stop_at_end)
00141     {
00142         motor[left_motor] = 0;
00143         motor[right_motor] = 0;
00144     }
00145     g_debug_time_2 = nPgmTime;
00146 }
00147
00148 #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

Definition in file abs_end_r1.h.

2.4 abs_end_r1.h

```
00001
00014 #ifndef ABS_END_R1_H
00015 #define ABS_END_R1_H
00016
00017 void abs_end_r1(int delay, int lift_speed)
00018 {
00019     wait1Msec(delay);
00020     servo[abdd] = g_abdd_down;
00021     abs_drive(FORWARD, E_ANGLE, g_to_turn_dist, 50, true);
00022     wait1Msec(200);
00023     abs_turn(COUNTERCLOCKWISE, POINT, TURN, 75, 60);
00024     wait1Msec(200);
00025     abs_drive(FORWARD, E_ANGLE, 85, 50, true);
00026     motor[block_lift_motor] = lift_speed;
00027     motor[block_lift_motor2] = lift_speed;
00028     abs_turn(COUNTERCLOCKWISE, POINT, TURN, 90, 60);
00029     motor[block_lift_motor] = 0;
00030     motor[block_lift_motor2] = 0;
00031     if(g_auto_grabber_selection_ramp_options == SUB_SELECTION_RAMP_STOP) abs_drive(
      FORWARD, E_ANGLE, 80, 50, true);
00032     else abs_drive(FORWARD, E_ANGLE, 130, 50, true);
00033 }
00034
00035 #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

```
00001
00014 #ifndef ABS_END_R2_H
00015 #define ABS_END_R2_H
00016
00017 void abs_end_r2(int delay, int lift_speed)
00018 {
00019     wait1Msec(delay);
00020     servo[abdd] = g_abdd_down;
00021     abs_drive(BACKWARD, E_ANGLE, g_to_turn_dist, 50, true);
00022     wait1Msec(200);
00023     abs_turn(COUNTERCLOCKWISE, POINT, TURN, 90, 60);
00024     wait1Msec(200);
00025     abs_drive(FORWARD, E_ANGLE, 87, 50, true);
00026     wait1Msec(500);
00027     motor[block_lift_motor] = lift_speed;
00028     motor[block_lift_motor2] = lift_speed;
00029     abs_turn(CLOCKWISE, POINT, TURN, 84, 50);
00030     motor[block_lift_motor] = 0;
00031     motor[block_lift_motor2] = 0;
00032     if(g_auto_grabber_selection_ramp_options == SUB_SELECTION_RAMP_STOP) abs_drive(
      FORWARD, E_ANGLE, 80, 50, true);
00033     else abs_drive(FORWARD, E_ANGLE, 130, 50, true);
00034 }
00035
00036 #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

Definition in file abs_gyro_cal.h.

2.8 abs_gyro_cal.h

```
00001
00014 #ifndef ABS_GYRO_CAL_H
00015 #define ABS_GYRO_CAL_H
00016
00017 float abs_gyro_cal(long caltime)
00018 {
00019     long highest = -1000, lowest = 10000;
00020     float average = 0;
00021     g_start_time = nPgmTime;
00022     long samples=0;
00023     long data;
00024     while (nPgmTime < g_start_time+(caltime*1000))      // loop for the requested number of seconds
00025     {
00026         samples +=1;                    // count the number of iterations for averaging
00027         data = HTGYROreadRot(HTGYRO);           // get a new reading from the GYRO
00028         average += (float)data;             // add in the new value to the average
00029         if (highest < data) highest = data;     // adjust the highest value if necessary
00030             if (lowest> data) lowest = data;    // likewise for the lowest value
00031     }
00032     //g_gyro_noise=abs(highest-lowest);         // save the spread in the data for diagnostic display
00033     g_gyro_noise=abs(highest-lowest);
00034     return average/samples;                 // and return the average drift
00035 }
00036
00037 #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

Definition in file abs_gyro_drive.h.

2.10 abs_gyro_drive.h

```
00001
00016 #ifndef ABS_GYRO_DRIVE_H
00017 #define ABS_GYRO_DRIVE_H
00018
00019 void abs_gyro_drive(int speed,e_drive_direction dir)
00020 {
00021     int error = 0 - g_rel_heading;
00022
00023     if(dir == FORWARD)
00024     {
00025         motor[left_motor] = speed + (error*g_gyro_adjust);
00026         motor[right_motor] = speed - (error*g_gyro_adjust);
00027     }
00028     else
00029     {
00030         motor[left_motor] = -(abs(speed) - (error*g_gyro_adjust));
00031         motor[right_motor] = -(abs(speed) + (error*g_gyro_adjust));
00032     }
00033 }
00034 #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

Definition in file abs_initialize.h.

2.12 abs_initialize.h

```
00001
00014 #ifndef ABS_INITIALIZE_H
00015 #define ABS_INITIALIZE_H
00016
00017 void initialize()
00018 {
00019     StartTask(abs_screen);
00020     disableDiagnosticsDisplay();
00021     servoChangeRate[abdd] = 3;
00022     servo[roger_slide] = 127;
00023     servo[abdd] = g_abdd_down;
00024     servo[grabber_left] = GRABBER_LEFT_CLOSE;
00025     servo[grabber_right] = GRABBER_RIGHT_CLOSE;
00026     memset(g_intput_array,0,6);
00027     selection_program();
00028     PlaySoundFile("! Click.rso");
00029     g_drift = abs_gyro_cal(g_gyro_cal_time);
00030
00031     if(!HTACreadAllAxes(HTAC, g_x_axis, g_y_axis, g_z_axis)) g_error =
      ERR_ACCELERMOETER;
00032     if(g_gyro_noise>10) g_error = ERR_GYRO_CAL;
00033     if(HTSMUXreadPowerStatus(SENSOR_MUX)) g_error = ERR_SENSOR_MUX;
00034     if(HTSMUXreadPowerStatus(GYRO_MUX)) g_error = ERR_GYRO_MUX;
00035
00036     if(g_error != 0)
00037     {
00038         g_screen_state = S_ERROR;
00039         while(true)
00040         {
00041             g_gyro_true = true;
00042             PlayTone (250,25);
00043             wait1Msec(500);
00044             if(nNxtButtonPressed == kEnterButton && g_error == ERR_SENSOR_MUX)break;
00045             if(nNxtButtonPressed == kEnterButton && g_error == ERR_ACCELERMOETER)break;
00046         }
00047     }
00048
00049     g_screen_state = S_READY;
00050     StartTask(abs_sensors);
00051     HTANGresetAccumulatedAngle(angle_sensor);
00052
00053     waitForStart();
00054     eraseDisplay();
00055     g_start_time = nPgmTime;
00056     g_screen_state = S_DELAY_WAIT;
00057     wait1Msec(g_start_delay*1000);
00058     eraseDisplay();
00059     g_screen_state = S_GYRO_SHOW;
00060 }
00061
00062 #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

```
00001
00014 #ifndef ABS_JOYSTICK_DRIVE_H
00015 #define ABS_JOYSTICK_DRIVE_H
00016
00017 void abs_joystick_drive(e_joystick_method joy_type)
00018 {
00019     //------------------------------
00020     // robot lift
00021     //------------------------------
00022     if(joy1Btn(4) || joy2Btn(4)) motor[sky_hook]=g_robot_lift_up;
00023     else if(joy1Btn(2))motor[sky_hook]=g_robot_lift_down;
00024     else motor[sky_hook] = 0;
00025
00026     //------------------------------
00027     // drive motor controls
00028     //------------------------------
00029
00030     int speed1;
00031     int speed2;
00032
00033     int j1 = abs(joystick.joy1_y1);
00034     int j2 = abs(joystick.joy1_y2);
00035
00036     if(joy_type == LINEAR)
00037     {
00038         speed1 = j1*100/127;
00039         speed2 = j2*100/127;
00040     }
00041     else
00042     {
00043         speed1 = ((j1*j1) * 100/(128*128));
00044         speed2 = ((j2*j2) * 100/(128*128));
00045     }
00046
```

```
00047     if(joy1Btn(7) || joy1Btn(8))
00048     {
00049         speed1 = speed1/6;
00050         speed2 = speed2/6;
00051     }
00052     else if(joy1Btn(5) || joy1Btn(6)){}
00053     else
00054     {
00055         speed1 = speed1/3;
00056         speed2 = speed2/3;
00057     }
00058
00059     if(speed1<10) speed1 = 0;
00060     if(speed2<10) speed2 = 0;
00061
00062     if(joystick.joy1_y1<0) speed1 = -speed1;
00063     if(joystick.joy1_y2<0) speed2 = -speed2;
00064
00065     motor[right_motor] = speed2;
00066     motor[left_motor] = speed1;
00067 }
00068
00069 #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

```
00001
00014 #ifndef ABS_JOYSTICK_GUNNER_H
00015 #define ABS_JOYSTICK_GUNNER_H
00016
00017 task abs_joystick_gunner()
00018 {
00019     while(true)
```

```
00020     {
00021         //------------------------------
00022         // flag motor control
00023         //------------------------------
00024         g_misc = joystick.joy2_TopHat;
00025         switch(joystick.joy2_TopHat)
00026         {
00027         case -1:
00028             motor[jolly_roger]= 0;
00029             break;
00030         case 0:
00031             motor[jolly_roger] = g_flag_speed_up;
00032             break;
00033         case 2:
00034             motor[jolly_roger] = g_flag_speed_right;
00035             break;
00036         case 6:
00037             motor[jolly_roger] = g_flag_speed_left;
00038             break;
00039         case 4:
00040             motor[jolly_roger] = g_flag_speed_down;
00041             break;
00042         }
00043 
00044         //------------------------------
00045         // roger slide
00046         //------------------------------
00047         if(joystick.joy2_y2>10) servo[roger_slide] = 255;
00048         else if(joystick.joy2_y2<-10) servo[roger_slide] = 0;
00049         else servo[roger_slide] = 127;
00050 
00051         //------------------------------
00052         // robot kill switch
00053         //------------------------------
00054         if((joy1Btn(9))&&(joy2Btn(9))&&(joy1Btn(10))&&(joy2Btn(10))) g_program_done = true;
00055 
00056         //------------------------------
00057         // block lift
00058         //------------------------------
00059         if(joystick.joy2_y1>10)
00060         {
00061             motor[block_lift_motor] = g_block_speed_up;
00062             motor[block_lift_motor2] = g_block_speed_up;
00063         }
00064         else if(joystick.joy2_y1<-10)
00065         {
00066             if(joy2Btn(11))
00067             {
00068                 motor[block_lift_motor] = -100;
00069                 motor[block_lift_motor2] = -100;
00070             }
00071             else
00072             {
00073                 motor[block_lift_motor] = g_block_speed_down;
00074                 motor[block_lift_motor2] = g_block_speed_down;
00075             }
00076         }
00077 
00078         else
00079         {
00080             motor[block_lift_motor] = 0;
00081             motor[block_lift_motor2] = 0;
00082         }
00083 
00084         //------------------------------
00085         // block grabber
00086         //------------------------------
00087 
00088         if(joy2Btn(1)) //grabber_position = GRABBER_OPEN;
00089         {
00090             servo[grabber_left] = GRABBER_LEFT_OPEN;
00091             servo[grabber_right] = GRABBER_RIGHT_OPEN;
00092         }
00093         if(joy2Btn(2)) //grabber_position = GRABBER_MID;
00094         {
00095             servo[grabber_left] = GRABBER_LEFT_MID;
00096             servo[grabber_right] = GRABBER_RIGHT_MID;
00097         }
00098         if(joy2Btn(3)) //grabber_position = GRABBER_CLOSE;
00099         {
00100             servo[grabber_left] = GRABBER_LEFT_CLOSE;
```

```
00101             servo[grabber_right] = GRABBER_RIGHT_CLOSE;
00102         }
00103
00104         if(joy2Btn(5)) servo[grabber_left] = GRABBER_LEFT_OPEN;
00105         else if(joy2Btn(7)) servo[grabber_left] = GRABBER_LEFT_CLOSE;
00106         if(joy2Btn(6)) servo[grabber_right] = GRABBER_RIGHT_OPEN;
00107         else if(joy2Btn(8)) servo[grabber_right] = GRABBER_RIGHT_CLOSE;
00108     }
00109 }
00110
00111 #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

```
00001
00015 #ifndef ABS_MOTOR_H
00016 #define ABS_MOTOR_H
00017
00018 void abs_motor(e_motor_move move_type)
00019 {
00020     //-----------------------------
00021     // roger slide
00022     //-----------------------------
00023     if(move_type == ROGGER_SLIDE) servo[roger_slide] = 0;
00024
00025     //-----------------------------
00026     // ABDD
00027     //-----------------------------
00028     if(move_type == ABDD)
00029     {
00030         servo[abdd] = g_abdd_up;
00031         wait10Msec(70);
```

```
00032         servo[abdd] = g_abdd_down;
00033     }
00034 }
00035
00036 #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,
```

```
00071   E_LIGHT
00072 } e_move_stopping_method; //will make a method with a tilt sensor(wheel in the middle
       of the robot
00073
00081 typedef enum
00082 {
00083     SWING,
00084     POINT
00085 } e_turn_method;
00086
00094 typedef enum
00095 {
00096     TURN,
00097     TURN_TO
00098 } e_turn_stopping_method;
00099
00111 typedef enum
00112 {
00113     ABDD,
00114     LIFT,
00115     GRABBER,
00116     ROGGER_SLIDE
00117 } e_motor_move;
00118
00123 #define distance_to_angle_derees(X) ((float)(X*360/ANGLE_SENSOR_CIRCUMFERENCE))
00124
00129 #define distance_to_encoder_derees(X) (X*360/DRIVE_WHEELS_CIRCUMFERENCE)
00130
00131 #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

```
00001
```

```
00014 #ifndef ABS_S1_MISSION_EXECUTE_H
00015 #define ABS_S1_MISSION_EXECUTE_H
00016
00017 void abs_s1_mission_execute()
00018 {
00019     switch(g_mission_number)
00020     {
00021     case 0:
00022         g_screen_state = S_ANGLE_SHOW;
00023         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/600, 50, true);
00024         break;
00025
00026     case 1:
00027         g_screen_state = S_ANGLE_SHOW;
00028         abs_drive(FORWARD, E_IR_DETECT, 7, 40, true);
00029         PlayTone(200,20);
00030         wait1Msec(1000);
00031         //if(g_IR_angle_dist_complete == true) g_end_point = 12;
00032         if(g_end_point == 2)
00033         {
00034             if(HTANGreadAccumulatedAngle(angle_sensor)<(62*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate1_to_turn_dist;
00035             else if(HTANGreadAccumulatedAngle(angle_sensor)<(100*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate2_to_turn_dist;
00036             else if(HTANGreadAccumulatedAngle(angle_sensor)<(137*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate3_to_turn_dist;
00037             else if(HTANGreadAccumulatedAngle(angle_sensor)<(162*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate4_to_turn_dist;
00038         }
00039         else if(g_end_point == 3)
00040         {
00041             if(HTANGreadAccumulatedAngle(angle_sensor)<(62*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_backwards_crate1_to_turn_dist;
00042             else if(HTANGreadAccumulatedAngle(angle_sensor)<(100*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_backwards_crate2_to_turn_dist;
00043             else if(HTANGreadAccumulatedAngle(angle_sensor)<(137*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_backwards_crate3_to_turn_dist;
00044             else if(HTANGreadAccumulatedAngle(angle_sensor)<(162*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_backwards_crate4_to_turn_dist;
00045         }
00046         wait1Msec(500);
00047         servo[abdd] = g_abdd_up;
00048         wait1Msec(2000);
00049         servo[abdd] = g_abdd_down;
00050         break;
00051
00052     case 2:
00053         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate4_to_turn_dist;
00054         else g_to_turn_dist = g_forward_crate4_to_turn_dist;
00055         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/150, 50, true);
00056         wait1Msec(2000);
00057         servo[abdd] = g_abdd_up;
00058         wait1Msec(2000);
00059         servo[abdd] = g_abdd_down;
00060         break;
00061
00062     case 3:
00063         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate3_to_turn_dist;
00064         else g_to_turn_dist = g_forward_crate3_to_turn_dist;
00065         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/125, 50, true);
00066         servo[abdd] = g_abdd_up;
00067         wait1Msec(2000);
00068         servo[abdd] = g_abdd_down;
00069         break;
00070
00071     case 4:
00072         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate2_to_turn_dist;
00073         else g_to_turn_dist = g_forward_crate2_to_turn_dist;
00074         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/75, 50, true);
00075         servo[abdd] = g_abdd_up;
00076         wait1Msec(2000);
00077         servo[abdd] = g_abdd_down;
00078         break;
```

```
00079
00080     case 5:
00081         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate1_to_turn_dist;
00082         else g_to_turn_dist = g_forward_crate1_to_turn_dist;
00083         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/50, 50, true);
00084         servo[abdd] = g_abdd_up;
00085         wait1Msec(2000);
00086         servo[abdd] = g_abdd_down;
00087         break;
00088
00089     case 6:
00090         abs_turn(COUNTERCLOCKWISE, POINT, TURN, 75, 60);
00091         wait1Msec(200);
00092         abs_drive(FORWARD, E_ANGLE, 190, 50, true);
00093         abs_turn(CLOCKWISE, POINT, TURN, 75, 60);
00094         abs_drive(FORWARD, E_ANGLE, 80, 50, true);
00095         break;
00096
00097     case 7:
00098         abs_turn(COUNTERCLOCKWISE, POINT, TURN, 98, 60);
00099         wait1Msec(200);
00100         abs_drive(FORWARD, E_ANGLE, 87, 50, true);
00101         motor[block_lift_motor] = 40;
00102         motor[block_lift_motor2] = 40;
00103         abs_turn(CLOCKWISE, POINT, TURN, 103, 60);
00104         motor[block_lift_motor] = 0;
00105         motor[block_lift_motor2] = 0;
00106         abs_drive(FORWARD, E_ANGLE, 80, 50, true);
00107         break;
00108
00109     case 140:
00110         int dist = 30;
00111         bool done = false;
00112         while(done == false)
00113         {
00114             int ac_start_time = nPgmTime;
00115             int i = 0;
00116             while((g_accelermoeter_sensor < dist+5) && (g_accelermoeter_sensor > dist-5) && ((ac_start_time
      - nPgmTime)<500))
00117             {
00118                 i++;
00119                 PlayTone(20,20);
00120                 wait1Msec(1);
00121             }
00122             if(i > 490) done = true;
00123             PlayTone(20,20);
00124         }
00125         break;
00126     }
00127     wait1Msec(g_end_delay*1000);
00128     switch(g_end_point)
00129     {
00130     case 1:
00131         wait1Msec(2000);
00132         servo[abdd] = g_abdd_down;
00133         abs_stop_robot();
00134         break;
00135     case 2:
00136         abs_end_r1(2000,40);
00137         break;
00138     case 3:
00139         abs_end_r2(2000,40);
00140         break;
00141     }
00142 }
00143
00144 #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

```
00001
00014 #ifndef ABS_S2_MISSION_EXECUTE_H
00015 #define ABS_S2_MISSION_EXECUTE_H
00016
00017 void abs_s2_mission_execute()
00018 {
00019     switch(g_mission_number)
00020     {
00021     case 0:
00022         g_screen_state = S_ANGLE_SHOW;
00023         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/600, 50, true);
00024         break;
00025
00026     case 1:
00027         abs_drive(BACKWARD, E_IR_DETECT, 3, 40, true);
00028         PlayTone(200,20);
00029         //if(g_IR_angle_dist_complete == true) g_end_point = 12;
00030         if(g_end_point == 2)
00031         {
00032             if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(62*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate4_to_turn_dist;
00033             else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(100*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate3_to_turn_dist;
00034             else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(137*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate2_to_turn_dist;
00035             else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(162*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_forward_crate1_to_turn_dist;
00036         }
00037         else if(g_end_point == 3)
00038         {
00039             if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(62*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_backwards_crate4_to_turn_dist;
00040             else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(100*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_backwards_crate3_to_turn_dist;
00041             else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(137*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
```

```
      g_backwards_crate2_to_turn_dist;
00042             else if(abs(HTANGreadAccumulatedAngle(angle_sensor))<(162*
      INT_ANGLE_SENSOR_CIRCUMFERENCE)) g_to_turn_dist =
      g_backwards_crate1_to_turn_dist;
00043         }
00044         wait1Msec(1000);
00045         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/6, 50, true);
00046         wait1Msec(500);
00047         servo[abdd] = g_abdd_up;
00048         wait1Msec(2000);
00049         servo[abdd] = g_abdd_down;
00050         break;
00051
00052     case 2:
00053         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate4_to_turn_dist;
00054         else g_to_turn_dist = g_forward_crate4_to_turn_dist;
00055         abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/40, 50, true);
00056         servo[abdd] = g_abdd_up;
00057         wait1Msec(2000);
00058         servo[abdd] = g_abdd_down;
00059         break;
00060
00061     case 3:
00062         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate3_to_turn_dist;
00063         else g_to_turn_dist = g_forward_crate3_to_turn_dist;
00064         abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/65, 50, true);
00065         servo[abdd] = g_abdd_up;
00066         wait1Msec(2000);
00067         servo[abdd] = g_abdd_down;
00068         break;
00069
00070     case 4:
00071         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate2_to_turn_dist;
00072         else g_to_turn_dist = g_forward_crate2_to_turn_dist;
00073         abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/115, 50, true);
00074         servo[abdd] = g_abdd_up;
00075         wait1Msec(2000);
00076         servo[abdd] = g_abdd_down;
00077         break;
00078
00079     case 5:
00080         if(g_end_point == 3)g_to_turn_dist = g_backwards_crate1_to_turn_dist;
00081         else g_to_turn_dist = g_forward_crate1_to_turn_dist;
00082         abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/140, 50, true);
00083         wait1Msec(2000);
00084         servo[abdd] = g_abdd_up;
00085         wait1Msec(2000);
00086         servo[abdd] = g_abdd_down;
00087         break;
00088
00089     case 6:
00090         abs_turn(CLOCKWISE, POINT, TURN, 75, 60);
00091         wait1Msec(200);
00092         abs_drive(FORWARD, E_ANGLE, 190, 50, true);
00093         abs_turn(COUNTERCLOCKWISE, POINT, TURN, 75, 60);
00094         abs_drive(FORWARD, E_ANGLE, 80, 50, true);
00095         break;
00096
00097     case 7:
00098         abs_turn(COUNTERCLOCKWISE, POINT, TURN, 98, 60);
00099         wait1Msec(200);
00100         abs_drive(FORWARD, E_ANGLE, 87, 50, true);
00101         motor[block_lift_motor] = 40;
00102         motor[block_lift_motor2] = 40;
00103         abs_turn(CLOCKWISE, POINT, TURN, 103, 60);
00104         motor[block_lift_motor] = 0;
00105         motor[block_lift_motor2] = 0;
00106         abs_drive(FORWARD, E_ANGLE, 80, 50, true);
00107         break;
00108
00109     case 140:
00110         int dist = 30;
00111         bool done = false;
00112         while(done == false)
00113         {
00114             int ac_start_time = nPgmTime;
00115             int i = 0;
00116             while((g_accelermoeter_sensor < dist+5) && (g_accelermoeter_sensor > dist-5) && ((ac_start_time
      - nPgmTime)<500))
00117             {
00118                 i++;
```

```
00119                 PlayTone(20,20);
00120                 wait1Msec(1);
00121             }
00122             if(i > 490) done = true;
00123             PlayTone(20,20);
00124         }
00125         break;
00126     }
00127     wait1Msec(g_end_delay*1000);
00128     switch(g_end_point)
00129     {
00130     case 1:
00131         wait1Msec(2000);
00132         servo[abdd] = g_abdd_down;
00133         abs_stop_robot();
00134         break;
00135     case 2:
00136         abs_end_r1(2000,40);
00137         break;
00138     case 3:
00139         abs_end_r2(2000,40);
00140         break;
00141     }
00142 }
00143
00144 #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

```
00001
00014 #ifndef ABS_S3_MISSION_EXECUTE_H
00015 #define ABS_S3_MISSION_EXECUTE_H
00016
```

```
00017 void abs_s3_mission_execute()
00018 {
00019     switch(g_mission_number)
00020     {
00021     case 1:
00022         break;
00023
00024     case 2:
00025         abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
00026         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
00027         abs_turn(CLOCKWISE, POINT, TURN_TO, 40, 60);
00028         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/100, 50, true);
00029         servo[abdd] = g_abdd_up;
00030         wait1Msec(2000);
00031         servo[abdd] = g_abdd_down;
00032         if(g_end_point == 3)g_to_turn_dist = 145;
00033         else g_to_turn_dist = g_forward_crate4_to_turn_dist;
00034         break;
00035
00036     case 3:
00037         abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
00038         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
00039         abs_turn(CLOCKWISE, POINT, TURN_TO, 40, 35);
00040         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/75, 50, true);
00041         servo[abdd] = g_abdd_up;
00042         wait1Msec(2000);
00043         servo[abdd] = g_abdd_down;
00044         if(g_end_point == 3)g_to_turn_dist = 120;
00045         else g_to_turn_dist = g_forward_crate3_to_turn_dist;
00046         break;
00047
00048     case 4:
00049         abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
00050         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/33, 50, true);
00051         abs_turn(CLOCKWISE, POINT, TURN_TO, 39, 50);
00052         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/25, 50, true);
00053         servo[abdd] = g_abdd_up;
00054         wait1Msec(2000);
00055         servo[abdd] = g_abdd_down;
00056         if(g_end_point == 3)g_to_turn_dist = 70;
00057         else g_to_turn_dist = g_forward_crate2_to_turn_dist;
00058         break;
00059
00060     case 5:
00061         abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
00062         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
00063         abs_turn(CLOCKWISE, POINT, TURN_TO, 35, 60);
00064         servo[abdd] = g_abdd_up;
00065         wait1Msec(2000);
00066         servo[abdd] = g_abdd_down;
00067         if(g_end_point == 3) g_to_turn_dist = 45;
00068         else if(g_end_point == 2) g_to_turn_dist = g_forward_crate1_to_turn_dist+5;
00069         break;
00070
00071     case 6:
00072         abs_turn(COUNTERCLOCKWISE, SWING, TURN_TO, 315, 60);
00073         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
00074         abs_turn(CLOCKWISE, POINT, TURN_TO, 35, 60);
00075         abs_drive(BACKWARD, E_ANGLE, g_to_turn_dist, 50, true);
00076         abs_turn(COUNTERCLOCKWISE, POINT, TURN, 90, 60);
00077         abs_drive(FORWARD, E_ANGLE, 180, 50, true);
00078         break;
00079
00080     case 7:
00081         break;
00082     }
00083     wait1Msec(g_end_delay*1000);
00084     switch(g_end_point)
00085     {
00086     case 1:
00087         wait1Msec(2000);
00088         servo[abdd] = g_abdd_down;
00089         abs_stop_robot();
00090         break;
00091     case 2:
00092         abs_end_r1(2000,40);
00093         break;
00094     case 3:
00095         abs_end_r2(2000,40);
00096         break;
00097     }
```

```
00098 }
00099
00100 #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

```
00001
00014 #ifndef ABS_S4_MISSION_EXECUTE_H
00015 #define ABS_S4_MISSION_EXECUTE_H
00016
00017 void abs_s4_mission_execute()
00018 {
00019     switch(g_mission_number)
00020     {
00021     case 1:
00022         break;
00023
00024     case 2:
00025         abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
00026         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
00027         abs_turn(CLOCKWISE, POINT, TURN_TO, 128, 60);
00028         servo[abdd] = g_abdd_up;
00029         wait1Msec(2000);
00030         servo[abdd] = g_abdd_down;
00031         if(g_end_point == 3) g_to_turn_dist = g_forward_crate1_to_turn_dist+5;
00032         else if(g_end_point == 2) g_to_turn_dist = 45;
00033         break;
00034
00035     case 3:
00036         abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
00037         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/29, 50, true);
00038         abs_turn(CLOCKWISE, POINT, TURN_TO, 135, 60);
00039         abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/23, 50, true);
```

```
00040         servo[abdd] = g_abdd_up;
00041         wait1Msec(2000);
00042         servo[abdd] = g_abdd_down;
00043         if(g_end_point == 3)g_to_turn_dist = g_forward_crate2_to_turn_dist;
00044         else g_to_turn_dist = 70;
00045         break;
00046
00047     case 4:
00048         abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
00049         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/29, 50, true);
00050         abs_turn(CLOCKWISE, POINT, TURN_TO, 135, 60);
00051         abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/75, 50, true);
00052         servo[abdd] = g_abdd_up;
00053         wait1Msec(2000);
00054         servo[abdd] = g_abdd_down;
00055         if(g_end_point == 3)g_to_turn_dist = g_forward_crate3_to_turn_dist;
00056         else g_to_turn_dist = 120;
00057         break;
00058
00059     case 5:
00060         abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
00061         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/29, 50, true);
00062         abs_turn(CLOCKWISE, POINT, TURN_TO, 135, 60);
00063         abs_drive(BACKWARD, E_ANGLE, /*distance in cm*/100, 50, true);
00064         servo[abdd] = g_abdd_up;
00065         wait1Msec(2000);
00066         servo[abdd] = g_abdd_down;
00067         if(g_end_point == 3)g_to_turn_dist = g_forward_crate4_to_turn_dist;
00068         else g_to_turn_dist = 145;
00069         break;
00070
00071     case 6:
00072         abs_turn(CLOCKWISE, SWING, TURN_TO, 60, 60);
00073         abs_drive(FORWARD, E_ANGLE, /*distance in cm*/30, 50, true);
00074         abs_turn(CLOCKWISE, POINT, TURN_TO, 120, 60);
00075         abs_drive(FORWARD, E_ANGLE, g_to_turn_dist, 50, true);
00076         abs_turn(CLOCKWISE, POINT, TURN, 90, 60);
00077         abs_drive(FORWARD, E_ANGLE, 180, 50, true);
00078         break;
00079
00080     case 7:
00081         break;
00082     }
00083     wait1Msec(g_end_delay*1000);
00084     switch(g_end_point)
00085     {
00086     case 1:
00087         wait1Msec(2000);
00088         servo[abdd] = g_abdd_up;
00089         abs_stop_robot();
00090         break;
00091     case 2:
00092         abs_end_r1(2000,40);
00093         break;
00094     case 3:
00095         abs_end_r2(2000,40);
00096         break;
00097     }
00098 }
00099
00100 #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

```
00001
00015 #ifndef ABS_SCREEN_H
00016 #define ABS_SCREEN_H
00017
00018 task abs_screen ()
00019 {
00020     while(true)
00021     {
00022         nxtDisplayBigTextLine(7, "              ");
00023         switch(g_screen_state)
00024         {
00025         case S_CLEAR:
00026             nxtDisplayBigTextLine(1, "              ");
00027             nxtDisplayBigTextLine(3, "              ");
00028             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00029             break;
00030         case S_MISSION:
00031             nxtDisplayBigTextLine(1, "Misson ","2%d", g_mission_number);
00032             //nxtDisplayBigTextLine(3, "%2d", g_mission_number);
00033             nxtDisplayBigTextLine(3, g_mission_names1[g_mission_number]);
00034             nxtDisplayBigTextLine(5, g_mission_names2[g_mission_number]);
00035             break;
00036         case S_DELAY:
00037             if(g_auto_selection_point == SELECTION_START_DELAY) nxtDisplayBigTextLine(
      1, "Start   ");
00038             else nxtDisplayBigTextLine(1, "Mission ");
00039             nxtDisplayBigTextLine(3, "Delay");
00040             nxtDisplayBigTextLine(5, "%2d", g_delay);
00041             break;
00042         case S_CAL_TIME:
00043             nxtDisplayBigTextLine(1, "CalTime");
00044             nxtDisplayBigTextLine(3, "%2d", g_gyro_cal_time);
00045             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00046             break;
00047         case S_GYRO_CAL:
00048             nxtDisplayTextLine(1, "Calibrating");
00049             nxtDisplayBigTextLine(3, "%2d", (g_gyro_cal_time*1000)-(nPgmTime-g_start_time));
00050             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00051             break;
00052         case S_READY:
00053             nxtDisplayBigTextLine(1, "Program");
00054             nxtDisplayBigTextLine(3, "Ready");
00055             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);
00056             else nxtDisplayBigTextLine(5, "%1d%1d%1d%1d%1d Y", g_start_point, g_start_delay,
      g_mission_number, g_end_delay, g_end_point);
00057             break;
00058         case S_DELAY_WAIT:
00059             nxtDisplayBigTextLine(1, "Delay");
00060             nxtDisplayBigTextLine(3, "%2d", (g_start_delay*1000)-(nPgmTime-g_start_time));
00061             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00062             break;
00063         case S_GYRO_SHOW:
00064             nxtDisplayBigTextLine(1, "GyroValue");
```

```
00065             nxtDisplayBigTextLine(3, "%2d", g_const_heading);
00066             nxtDisplayBigTextLine(5, "%2d", g_rel_heading);
00067             break;
00068         case S_IR_SHOW:
00069             nxtDisplayBigTextLine(1, "IR Value");
00070             nxtDisplayBigTextLine(3, "%2d  %2d", g_bearing_ac1,
    g_bearing_ac2);
00071             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00072             break;
00073         case S_AC_SHOW:
00074             nxtDisplayBigTextLine(1, "ac Value");
00075             nxtDisplayBigTextLine(3, "%2d  %2d", g_accelermoeter_sensor, g_misc);
00076             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00077             break;
00078         case S_ERROR:
00079             nxtDisplayBigTextLine(1, "ERROR");
00080             nxtDisplayBigTextLine(3, g_error_list1[g_error]);
00081             nxtDisplayBigTextLine(5, g_error_list2[g_error]);
00082             break;
00083         case S_SMOKE_TEST:
00084             nxtDisplayBigTextLine(1, "%2d", g_smoke_test_num);
00085             nxtDisplayBigTextLine(3, g_smoke_test1[g_smoke_test_num]);
00086             nxtDisplayBigTextLine(5, g_smoke_test2[g_smoke_test_num]);
00087             break;
00088         case S_SMOKE_RUN1:
00089             nxtDisplayBigTextLine(1, g_smoke_test1[g_smoke_test_num]);
00090             nxtDisplayBigTextLine(3, "%2d", g_test_value);
00091             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00092             break;
00093         case S_SMOKE_RUN2:
00094             nxtDisplayBigTextLine(1, g_smoke_test1[g_smoke_test_num]);
00095             nxtDisplayBigTextLine(3, "%2d  %2d", g_sensor_value, g_sensor_value2);
00096             nxtDisplayBigTextLine(5, g_sensor_list[g_sensor_num]);
00097             break;
00098         case S_SCREEN_CALL:
00099             nxtDisplayBigTextLine(1, g_smoke_test1[g_smoke_test_num]);
00100             nxtDisplayBigTextLine(3, "%2d", g_test_value);
00101             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00102             break;
00103         case S_MISC_SHOW:
00104             nxtDisplayBigTextLine(1, "misc Value");
00105             nxtDisplayBigTextLine(3, "%2d", g_misc);
00106             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00107         case S_ANGLE_SHOW:
00108             nxtDisplayBigTextLine(1, "angle Value");
00109             nxtDisplayBigTextLine(3, "%2d", HTANGreadAccumulatedAngle(angle_sensor));
00110             nxtDisplayBigTextLine(5, g_mission_names1[0]);
00111             break;
00112         case S_STARTING_POINT:
00113             nxtDisplayBigTextLine(1, "startPnt");
00114             nxtDisplayBigTextLine(3, g_starting_names1[g_start_point]);
00115             nxtDisplayBigTextLine(5, g_starting_names2[g_start_point]);
00116             break;
00117         case S_ENDING_POINT:
00118             nxtDisplayBigTextLine(1, "endPoint");
00119             nxtDisplayBigTextLine(3, g_ending_names1[g_end_point]);
00120             nxtDisplayBigTextLine(5, g_ending_names2[g_end_point]);
00121             break;
00122         case S_SELECTION_SUB_GRABBERS:
00123             nxtDisplayBigTextLine(1, "Grabbers");
00124             nxtDisplayBigTextLine(3, "inOrOut?");
00125             if(g_auto_grabber_selections == SUB_SELECTION_GRABBERS_IN)
    nxtDisplayBigTextLine(5, g_basic_word_list [1]);
00126             else if(g_auto_grabber_selections == SUB_SELECTION_GRABBERS_OUT)
    nxtDisplayBigTextLine(5, g_basic_word_list [2]);
00127             break;
00128         case S_SELECTION_SUB_RAMP:
00129             nxtDisplayBigTextLine(1, "Ramp    ");
00130             nxtDisplayBigTextLine(3, "continu?");
00131             if(g_auto_grabber_selection_ramp_options ==
    SUB_SELECTION_RAMP_CONTINUED) nxtDisplayBigTextLine(5, g_basic_word_list [3]);
00132             else if(g_auto_grabber_selection_ramp_options ==
    SUB_SELECTION_RAMP_STOP) nxtDisplayBigTextLine(5, g_basic_word_list [4]);
00133             break;
00134         case S_TIME_SHOW:
00135             nxtDisplayBigTextLine(1, "T1    T2");
00136             nxtDisplayBigTextLine(3, "%2d", g_debug_time_1);
00137             nxtDisplayBigTextLine(5, "%2d", g_debug_time_2);
00138             break;
00139         case S_MISSION_SHOW:
00140             nxtDisplayBigTextLine(1, "numbers");
```

```
00141             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]);
00142             nxtDisplayTextLine(5, "%1d-%1d-%1d-%1d-%1d", g_start_point, g_start_delay, g_mission_number,
      g_end_delay, g_end_point);
00143             break;
00144         case S_SELECTION_TYPE:
00145             nxtDisplayBigTextLine(1, "Selecton");
00146             nxtDisplayBigTextLine(3, "Type:   ");
00147             if(selection_type == SELECTION_TYPE_CUSTOM) nxtDisplayBigTextLine(5, "
      custom  ");
00148             else if(selection_type == SELECTION_TYPE_NUMBER) nxtDisplayBigTextLine(5,
      "number  ");
00149             else if(selection_type == SELECTION_TYPE_QUICK) nxtDisplayBigTextLine(5, "
      quick    ");
00150             break;
00151         case S_NUMBER_SELECTION:
00152             nxtDisplayBigTextLine(1, "Mission");
00153             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]);
00154             switch(g_graph_selection_tab)
00155             {
00156             case 1: nxtDisplayBigTextLine(5, " ^      "); break;
00157             case 2: nxtDisplayBigTextLine(5, "  ^     "); break;
00158             case 3: nxtDisplayBigTextLine(5, "   ^    "); break;
00159             case 4: nxtDisplayBigTextLine(5, "    ^   "); break;
00160             case 5: nxtDisplayBigTextLine(5, "     ^  "); break;
00161             }
00162             break;
00163         case S_QUICK_SELECTION:
00164             nxtDisplayBigTextLine(1, "Mission");
00165             nxtDisplayBigTextLine(3, g_quick_names1[g_quick_mission]);
00166             nxtDisplayBigTextLine(5, g_quick_names2[g_quick_mission]);
00167             break;
00168         }
00169     }
00170 }
00171
00172 #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

```
00001
00014 #ifndef ABS_SELECTION_CUSTOM_H
00015 #define ABS_SELECTION_CUSTOM_H
00016
00017 void abs_selection_custom()
00018 {
00019     //---------------------------------------
00020     // Start point selection
00021     //---------------------------------------
00022
00023     g_auto_selection_point = SELECTION_START_POINT;
00024     g_screen_state = S_STARTING_POINT;
00025
00026     while(nNxtButtonPressed != kEnterButton)
00027     {
00028         if(nNxtButtonPressed == kRightButton)
00029         {
00030             PlaySoundFile("! Click.rso");
00031             while(nNxtButtonPressed == kRightButton){}
00032             if(g_start_point < g_auto_starting_points) g_start_point++;
00033             else g_start_point = g_auto_starting_points;
00034         }
00035         if(nNxtButtonPressed == kLeftButton)
00036         {
00037             PlaySoundFile("! Click.rso");
00038             while(nNxtButtonPressed == kLeftButton){}
00039             if(g_start_point > 0) g_start_point--;
00040             else g_start_point = 0;
00041         }
00042     }
00043     PlaySoundFile("! Click.rso");
00044     while(nNxtButtonPressed == kEnterButton){}
00045     eraseDisplay();
00046
00047     //---------------------------------------
00048     // Start of start time selection
00049     //---------------------------------------
00050
00051     g_auto_selection_point = SELECTION_START_DELAY;
00052     g_screen_state = S_DELAY;
00053
00054     while(nNxtButtonPressed != kEnterButton)
00055     {
00056         g_delay = g_start_delay;
00057         if(nNxtButtonPressed == kRightButton)
00058         {
00059             PlaySoundFile("! Click.rso");
00060             while(nNxtButtonPressed == kRightButton){}
00061             if(g_start_delay < 30) g_start_delay++;
00062             else g_start_delay = 30;
00063         }
00064         if(nNxtButtonPressed == kLeftButton)
00065         {
00066             PlaySoundFile("! Click.rso");
00067             while(nNxtButtonPressed == kLeftButton){}
00068             if(g_start_delay > 0) g_start_delay--;
00069             else g_start_delay = 0;
00070         }
00071     }
00072
00073     PlaySoundFile("! Click.rso");
00074     while(nNxtButtonPressed == kEnterButton){}
00075
00076     //---------------------------------------
00077     // Start of mission selection
00078     //---------------------------------------
00079
00080     g_auto_selection_point = SELECTION_MISSION_POINT;
00081     g_screen_state = S_MISSION;
00082
00083     while(nNxtButtonPressed != kEnterButton)
00084     {
00085         if(nNxtButtonPressed == kRightButton)
00086         {
00087             PlaySoundFile("! Click.rso");
00088             while(nNxtButtonPressed == kRightButton){}
00089             if(g_mission_number < g_auto_missions) g_mission_number++;
00090             else g_mission_number = g_auto_missions;
```

```
00091         }
00092         if(nNxtButtonPressed == kLeftButton)
00093         {
00094             PlaySoundFile("! Click.rso");
00095             while(nNxtButtonPressed == kLeftButton){}
00096             if(g_mission_number > 0) g_mission_number--;
00097             else g_mission_number = 0;
00098         }
00099     }
00100     PlaySoundFile("! Click.rso");
00101     while(nNxtButtonPressed == kEnterButton){}
00102     eraseDisplay();
00103
00104     //---------------------------------------
00105     // Start of time selection
00106     //---------------------------------------
00107
00108     g_auto_selection_point = SELECTION_MISSION_DELAY;
00109     g_screen_state = S_DELAY;
00110
00111     while(nNxtButtonPressed != kEnterButton)
00112     {
00113         g_delay = g_end_delay;
00114         if(nNxtButtonPressed == kRightButton)
00115         {
00116             PlaySoundFile("! Click.rso");
00117             while(nNxtButtonPressed == kRightButton){}
00118             if(g_end_delay < 30) g_end_delay++;
00119             else g_end_delay = 30;
00120         }
00121         if(nNxtButtonPressed == kLeftButton)
00122         {
00123             PlaySoundFile("! Click.rso");
00124             while(nNxtButtonPressed == kLeftButton){}
00125             if(g_end_delay > 0) g_end_delay--;
00126             else g_end_delay = 0;
00127         }
00128     }
00129
00130     PlaySoundFile("! Click.rso");
00131     while(nNxtButtonPressed == kEnterButton){}
00132
00133     //---------------------------------------
00134     // Start of end point selection
00135     //---------------------------------------
00136
00137     g_screen_state = S_ENDING_POINT;
00138     g_auto_selection_point = SELECTION_END_POINT;
00139
00140     while(nNxtButtonPressed != kEnterButton)
00141     {
00142         if(nNxtButtonPressed == kRightButton)
00143         {
00144             PlaySoundFile("! Click.rso");
00145             while(nNxtButtonPressed == kRightButton){}
00146             if(g_end_point < g_auto_ending_points) g_end_point++;
00147             else g_end_point = g_auto_ending_points;
00148         }
00149         if(nNxtButtonPressed == kLeftButton)
00150         {
00151             PlaySoundFile("! Click.rso");
00152             while(nNxtButtonPressed == kLeftButton){}
00153             if(g_end_point > 0) g_end_point--;
00154             else g_end_point = 0;
00155         }
00156     }
00157     PlaySoundFile("! Click.rso");
00158     while(nNxtButtonPressed == kEnterButton){}
00159     eraseDisplay();
00160
00161     //---------------------------------------
00162     // Start of optional sub selection for grabbers on the ram
00163     //---------------------------------------
00164
00165     if(false)//g_end_point == 2 || g_end_point == 3)
00166     {
00167         g_auto_selection_point = SELECTION_SUB_GRABBERS;
00168         g_screen_state = S_SELECTION_SUB_GRABBERS;
00169
00170         int i = 1;
00171         while(nNxtButtonPressed != kEnterButton)
```

```
00172         {
00173             if(nNxtButtonPressed == kRightButton)
00174             {
00175                 PlaySoundFile("! Click.rso");
00176                 while(nNxtButtonPressed == kRightButton){}
00177                 if(i < 2)
00178                 {
00179                     i++;
00180                     g_auto_grabber_selections = SUB_SELECTION_GRABBERS_OUT;
00181                 }
00182                 else
00183                 {
00184                     g_end_delay = 2;
00185                     g_auto_grabber_selections = SUB_SELECTION_GRABBERS_OUT;
00186                 }
00187             }
00188             if(nNxtButtonPressed == kLeftButton)
00189             {
00190                 PlaySoundFile("! Click.rso");
00191                 while(nNxtButtonPressed == kLeftButton){}
00192                 if(i > 1)
00193                 {
00194                     i--;
00195                     g_auto_grabber_selections = SUB_SELECTION_GRABBERS_IN;
00196                 }
00197                 else
00198                 {
00199                     i = 1;
00200                     g_auto_grabber_selections = SUB_SELECTION_GRABBERS_IN;
00201                 }
00202             }
00203         }
00204         PlaySoundFile("! Click.rso");
00205         while(nNxtButtonPressed == kEnterButton){}
00206     }
00207 }
00208
00209 #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

```
00001
00014 #ifndef ABS_SELECTION_NUMBER_H
00015 #define ABS_SELECTION_NUMBER_H
00016
00017 void abs_selection_number()
00018 {
00019     //---------------------------------------
00020     // number selection
00021     //---------------------------------------
00022
00023     g_auto_selection_point = SELECTION_GRAPH_NUMBER_INPUT;
00024     g_screen_state = S_NUMBER_SELECTION;
00025
00026     while(g_graph_selection_tab<5)
00027     {
00028         g_graph_selection_tab++;
00029         while(nNxtButtonPressed != kEnterButton)
00030         {
00031             if(nNxtButtonPressed == kRightButton)
00032             {
00033                 PlaySoundFile("! Click.rso");
00034                 while(nNxtButtonPressed == kRightButton){}
00035                 g_intput_array[g_graph_selection_tab] ++;
00036             }
00037             if(nNxtButtonPressed == kLeftButton)
00038             {
00039                 PlaySoundFile("! Click.rso");
00040                 while(nNxtButtonPressed == kLeftButton){}
00041                 g_intput_array[g_graph_selection_tab] --;
00042             }
00043         }
00044         while(nNxtButtonPressed == kEnterButton){}
00045         PlaySoundFile("! Click.rso");
00046     }
00047     g_start_point = g_intput_array[1];
00048     g_start_delay = g_intput_array[2];
00049     g_mission_number = g_intput_array[3];
00050     g_end_delay = g_intput_array[4];
00051     g_end_point = g_intput_array[5];
00052
00053     g_screen_state = S_MISSION_SHOW;
00054 }
00055
00056 #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

Definition in file abs_selection_program.h.

2.36 abs_selection_program.h

```
00001
00014 #ifndef ABS_SELECTION_PROGRAM_H
00015 #define ABS_SELECTION_PROGRAM_H
00016
00017 void selection_program()
00018 {
00019     while(nNxtButtonPressed == kEnterButton){}
00020
00021     //---------------------------------------
00022     // number selection, quick selection, or custom selection
00023     //---------------------------------------
00024
00025     g_auto_selection_point = SELECTION_SELECTION_TYPE;
00026     g_screen_state = S_SELECTION_TYPE;
00027
00028     int j = 1;
00029     while(nNxtButtonPressed != kEnterButton)
00030     {
00031         if(nNxtButtonPressed == kRightButton)
00032         {
00033             PlaySoundFile("! Click.rso");
00034             while(nNxtButtonPressed == kRightButton){}
00035             if(j < 3) j++;
00036             else j = 3;
00037         }
00038         if(nNxtButtonPressed == kLeftButton)
00039         {
00040             PlaySoundFile("! Click.rso");
00041             while(nNxtButtonPressed == kLeftButton){}
00042             if(j > 1) j--;
00043             else j = 1;
00044         }
00045         switch(j)
00046         {
00047         case 1:
00048             selection_type = SELECTION_TYPE_CUSTOM;
00049             break;
00050         case 2:
00051             selection_type = SELECTION_TYPE_NUMBER;
00052             break;
00053             // in for future use
00054         case 3:
00055             selection_type = SELECTION_TYPE_QUICK;
00056             break;
00057         }
00058     }
00059     PlaySoundFile("! Click.rso");
00060     while(nNxtButtonPressed == kEnterButton){}
00061     eraseDisplay();
00062
00063     //---------------------------------------
00064     // selection executes
00065     //---------------------------------------
00066     if(selection_type == SELECTION_TYPE_CUSTOM) abs_selection_custom();
00067     else if(selection_type == SELECTION_TYPE_NUMBER) abs_selection_number();
00068     else if(selection_type == SELECTION_TYPE_QUICK) abs_selection_quick();
00069
00070     //---------------------------------------
00071     // Start of optional sub selection for ramp position
00072     //---------------------------------------
00073
```

```
00074     if(g_end_point == 2 || g_end_point == 3)
00075     {
00076         g_auto_selection_point = SELECTION_SUB_RAMP;
00077         g_screen_state = S_SELECTION_SUB_RAMP;
00078 
00079         int i = 1;
00080         while(nNxtButtonPressed != kEnterButton)
00081         {
00082             if(nNxtButtonPressed == kRightButton)
00083             {
00084                 PlaySoundFile("! Click.rso");
00085                 while(nNxtButtonPressed == kRightButton){}
00086                 if(i < 2)
00087                 {
00088                     i++;
00089                     g_auto_grabber_selection_ramp_options =
      SUB_SELECTION_RAMP_CONTINUED;
00090                 }
00091             }
00092             if(nNxtButtonPressed == kLeftButton)
00093             {
00094                 PlaySoundFile("! Click.rso");
00095                 while(nNxtButtonPressed == kLeftButton){}
00096                 if(i > 1)
00097                 {
00098                     i--;
00099                     g_auto_grabber_selections = SUB_SELECTION_RAMP_STOP;
00100                 }
00101             }
00102         }
00103 
00104         PlaySoundFile("! Click.rso");
00105         while(nNxtButtonPressed == kEnterButton){}
00106     }
00107 
00108     //--------------------------------------
00109     // Start of gyro cal selection
00110     //--------------------------------------
00111 
00112     g_screen_state = S_CAL_TIME;
00113 
00114     while(nNxtButtonPressed != kEnterButton)
00115     {
00116         if(nNxtButtonPressed == kRightButton)
00117         {
00118             PlaySoundFile("! Click.rso");
00119             while(nNxtButtonPressed == kRightButton){}
00120             if(g_gyro_cal_time < 30) g_gyro_cal_time++;
00121             else g_gyro_cal_time = 30;
00122         }
00123         if(nNxtButtonPressed == kLeftButton)
00124         {
00125             PlaySoundFile("! Click.rso");
00126             while(nNxtButtonPressed == kLeftButton){}
00127             if(g_gyro_cal_time > 0) g_gyro_cal_time--;
00128             else g_gyro_cal_time = 0;
00129         }
00130     }
00131     PlaySoundFile("! Click.rso");
00132     while(nNxtButtonPressed == kEnterButton){}
00133     eraseDisplay();
00134     g_screen_state = S_GYRO_CAL;
00135 }
00136 
00137 #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

Definition in file abs_selection_quick.h.

2.38 abs_selection_quick.h

```
00001
00014 #ifndef ABS_SELECTION_QUICK_H
00015 #define ABS_SELECTION_QUICK_H
00016
00017 void abs_selection_quick()
00018 {
00019     //---------------------------------------
00020     // quick selection
00021     //---------------------------------------
00022
00023     g_auto_selection_point = SELECTION_QUICK_INPUT;
00024     g_screen_state = S_QUICK_SELECTION;
00025
00026     while(nNxtButtonPressed != kEnterButton)
00027     {
00028         if(nNxtButtonPressed == kRightButton)
00029         {
00030             PlaySoundFile("! Click.rso");
00031             while(nNxtButtonPressed == kRightButton){}
00032             if(g_quick_mission < g_max_quick_missions)g_quick_mission++;
00033         }
00034         if(nNxtButtonPressed == kLeftButton)
00035         {
00036             PlaySoundFile("! Click.rso");
00037             while(nNxtButtonPressed == kLeftButton){}
00038             if(g_quick_mission > 1)g_quick_mission--;
00039         }
00040     }
00041     while(nNxtButtonPressed == kEnterButton){}
00042     PlaySoundFile("! Click.rso");
00043
00044     switch(g_quick_mission)
00045     {
00046     case 1:
00047         g_start_point = 1;
00048         g_start_delay = 0;
00049         g_mission_number = 1;
00050         g_end_delay = 0;
00051         g_end_point = 2;
00052         break;
00053     case 2:
00054         g_start_point = 2;
00055         g_start_delay = 0;
00056         g_mission_number = 1;
00057         g_end_delay = 0;
00058         g_end_point = 3;
00059         break;
00060     }
00061 }
00062
00063 #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

```
00001
00015 #ifndef ABS_SENSOR_H
00016 #define ABS_SENSOR_H
00017
00018 task abs_sensors()
00019 {
00020     g_prev_time = nPgmTime;
00021
00022     while(true)
00023     {
00024         //-------------------------
00025         // Light Sensor
00026         //-------------------------
00027 //  g_light_sensor = SensorValue(lightSensor);
00028         //-------------------------
00029         // HiTechnic IR Sensor
00030         //-------------------------
00031         g_bearing_ac1 = HTIRS2readACDir(HTIRS2);               // Read the IR bearing from the
      sensor
00032         g_curr_dir1 = (float) g_bearing_ac1;
00033
00034         HTIRS2readAllACStrength(HTIRS2, g_acs1[0], g_acs1[1], g_acs1[2], g_acs1[3], g_acs1[4]);
00035         //-----------------------------------
00036         // code for the peaks of IR sensor
00037         //-----------------------------------
00038         if (g_bearing_ac1!=0)                                 // we have a valid IR signal
00039         {
00040             int maximum = -1;
00041             int peak = 0, offset=0;
00042             for (int i=0;i<5;i++)   // scan array to find the peak entry
00043             {   if (g_acs1[i]>maximum)
00044                 {
00045                     peak = i;
00046                     maximum = g_acs1[i];
```

```
00047               }
00048           }
00049           offset=0;
00050           if ((peak < 4) && (peak>0) && (g_acs1[peak] != 0))  // we are not working with extreme value
00051           {
00052               if (g_acs1[peak-1]!=g_acs1[peak+1]) // if the values either side of the peak are identical
    then peak is peak
00053               {
00054                   if (g_acs1[peak-1]>g_acs1[peak+1])  // otherwise decide which side has higher signal
00055                   {
00056                       offset = -25*(1-(float)(g_acs1[peak]-g_acs1[peak-1])/       // calculate the bias
    away from the peak
00057                       max(g_acs1[peak], g_acs1[peak-1]));
00058                   }
00059                   else
00060                   {
00061                       offset = 25*(1-(float)(g_acs1[peak]-g_acs1[peak+1])/
00062                       max(g_acs1[peak], g_acs1[peak+1]));
00063                   }
00064               }
00065           }
00066           g_ir_bearing1 = (float)((peak-2)*50) + offset;     // direction is the total of
    the peak bias plus the adjacent bias
00067           //nxtDisplayBigTextLine(3, "%2d", g_ir_bearing1);
00068       }
00069       //-------------------------
00070       // HiTechnic IR Sensor 2
00071       //-------------------------
00072       g_bearing_ac2 = HTIRS2readACDir(HTIRS2_2);             // Read the IR bearing from the
    sensor
00073       g_curr_dir2 = (float) g_bearing_ac2;
00074
00075       HTIRS2readAllACStrength(HTIRS2_2, g_acs2[0], g_acs2[1], g_acs2[2], g_acs2[3], g_acs2[4]);
00076       //---------------------------------
00077       // code for the peaks of IR sensor 2
00078       //---------------------------------
00079       if (g_bearing_ac2!=0)                               // we have a valid IR signal
00080       {
00081           int maximum = -1;
00082           int peak = 0, offset=0;
00083           for (int i=0;i<5;i++)   // scan array to find the peak entry
00084           {   if (g_acs2[i]>maximum)
00085               {
00086                   peak = i;
00087                   maximum = g_acs2[i];
00088               }
00089           }
00090           offset=0;
00091           if ((peak < 4) && (peak>0) && (g_acs2[peak] != 0))  // we are not working with extreme value
00092           {
00093               if (g_acs2[peak-1]!=g_acs2[peak+1]) // if the values either side of the peak are identical
    then peak is peak
00094               {
00095                   if (g_acs2[peak-1]>g_acs2[peak+1])  // otherwise decide which side has higher signal
00096                   {
00097                       offset = -25*(1-(float)(g_acs2[peak]-g_acs2[peak-1])/       // calculate the bias
    away from the peak
00098                       max(g_acs2[peak], g_acs2[peak-1]));
00099                   }
00100                   else
00101                   {
00102                       offset = 25*(1-(float)(g_acs2[peak]-g_acs2[peak+1])/
00103                       max(g_acs2[peak], g_acs2[peak+1]));
00104                   }
00105               }
00106           }
00107           g_ir_bearing2 = (float)((peak-2)*50) + offset;     // direction is the total of
    the peak bias plus the adjacent bias
00108           //nxtDisplayBigTextLine(3, "%2d", g_ir_bearing1);
00109       }
00110       //-------------------------
00111       // HiTechnic Gyro
00112       //-------------------------
00113
00114       g_curr_time=nPgmTime;
00115       g_raw_gyro = HTGYROreadRot(HTGYRO);
00116       g_const_heading += (g_raw_gyro - g_drift) * (float)(g_curr_time-g_prev_time)/1000;
00117       g_rel_heading += (g_raw_gyro - g_drift) * (float)(g_curr_time-g_prev_time)/1000;
00118       g_prev_time = g_curr_time;
00119
00120       g_recont_heading = g_const_heading % 360;
```

```
00121         if(g_recont_heading<0) g_recont_heading += 360;
00122
00123         //-------------------------
00124         // HiTechnic accelermoeter
00125         //-------------------------
00126
00127         HTACreadAllAxes(HTAC, g_x_axis, g_y_axis, g_z_axis);
00128         g_accelermoeter_sensor = g_x_axis;
00129
00130         if(g_sensor_reference_drive == true)
00131         {
00132             g_accelermoeter_reads++;
00133             g_accelermoeter_array[g_accelermoeter_reads%50]=g_accelermoeter_sensor;
00134             for(int i=0;i<30;i++)
00135             {
00136                 g_accelermoeter_total_value = g_accelermoeter_array[i];
00137             }
00138             g_accelermoeter_average = g_accelermoeter_total_value/50;
00139         }
00140         else
00141         {
00142             g_accelermoeter_reads = 0;
00143             g_accelermoeter_total_value = 0;
00144             g_accelermoeter_average = 0;
00145             memset(g_accelermoeter_array,0,30);
00146         }
00147         //-------------------------
00148         // HiTechnic angle sensor
00149         //-------------------------
00150         //if(g_reset_angle == true) HTANGresetAccumulatedAngle(HTANG);
00151         //else angle_sensor = HTANGreadAccumulatedAngle(HTANG);
00152     }
00153 }
00154 #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

Copyright 2013, Got Robot? FTC Team 5037

Definition in file abs_smoke_execute.h.

2.42 abs_smoke_execute.h

```
00001
00015 #ifndef ABS_SMOKE_EXECUTE_H
00016 #define ABS_SMOKE_EXECUTE_H
00017
00018 void abs_smoke_execute ()
00019 {
00020     g_screen_state = S_SMOKE_RUN1;
00021     while(nNxtButtonPressed != kEnterButton)
00022     {
00023         switch(g_smoke_test_num)
00024         {
00025             //---------------------------------
00026             // Jolly Roger
00027             //---------------------------------
00028         case 1:
00029             if(nNxtButtonPressed == kLeftButton)
00030             {
00031                 motor[jolly_roger] = g_flag_speed_down;
00032                 g_test_value = g_flag_speed_down;
00033             }
00034             else if(nNxtButtonPressed == kRightButton)
00035             {
00036                 motor[jolly_roger] = g_flag_speed_up;
00037                 g_test_value = g_flag_speed_up;
00038             }
00039             else
00040             {
00041                 g_test_value = 0;
00042                 motor[jolly_roger] = 0;
00043             }
00044             break;
00045             //---------------------------------
00046             // Drive
00047             //---------------------------------
00048         case 2:
00049             if(nNxtButtonPressed == kLeftButton)
00050             {
00051                 motor[right_motor] = 60;
00052                 motor[left_motor] = 60;
00053                 g_test_value = 60;
00054             }
00055             else if(nNxtButtonPressed == kRightButton)
00056             {
00057                 motor[right_motor] = -60;
00058                 motor[left_motor] = -60;
00059                 g_test_value = -60;
00060             }
00061             else
00062             {
00063                 g_test_value = 0;
00064                 motor[right_motor] = 0;
00065                 motor[left_motor] = 0;
00066             }
00067             break;
00068             //---------------------------------
00069             // sensors
00070             //---------------------------------
00071         case 3:
00072             g_screen_state = S_SMOKE_RUN2;
00073             if(nNxtButtonPressed == kLeftButton)
00074             {
00075                 if(g_test_value > 1) g_test_value--;
00076                 while(nNxtButtonPressed == kLeftButton) {}
00077             }
00078             if(nNxtButtonPressed == kRightButton)
00079             {
00080                 if(g_test_value < g_sensor_max) g_test_value++;
00081                 while(nNxtButtonPressed == kRightButton) {}
00082             }
00083             g_sensor_num = g_test_value;
00084             switch(g_sensor_num)
00085             {
00086             case ST_GYRO:
00087                 g_sensor_value = g_rel_heading;
00088                 break;
00089             case ST_IR:
00090                 g_sensor_value = g_bearing_ac1;
00091                 g_sensor_value2 = g_bearing_ac2;
```

```
00092               break;
00093           case ST_TILT:
00094               g_sensor_value = HTANGreadAccumulatedAngle(angle_sensor);
00095               break;
00096           case ST_ACCELEROMETER:
00097               g_sensor_value = g_accelermoeter_sensor;
00098               break;
00099           }
00100           break;
00101           //---------------------------------
00102           // Block lift
00103           //---------------------------------
00104       case 4:
00105           if(nNxtButtonPressed == kLeftButton)
00106           {
00107               motor[block_lift_motor] = g_robot_lift_down;
00108               motor[block_lift_motor2] = g_robot_lift_down;
00109               g_test_value = g_robot_lift_down;
00110           }
00111           else if(nNxtButtonPressed == kRightButton)
00112           {
00113               motor[block_lift_motor] = g_robot_lift_up;
00114               motor[block_lift_motor2] = g_robot_lift_up;
00115               g_test_value = g_robot_lift_up;
00116           }
00117           else
00118           {
00119               motor[block_lift_motor] = 0;
00120               motor[block_lift_motor2] = 0;
00121               g_test_value = 0;
00122           }
00123           break;
00124           //---------------------------------
00125           // grabbers
00126           //---------------------------------
00127       case 5:
00128           if(nNxtButtonPressed == kLeftButton)
00129           {
00130               if(g_test_value>1) g_test_value--;
00131               while(nNxtButtonPressed == kLeftButton){}
00132           }
00133           if(nNxtButtonPressed == kRightButton)
00134           {
00135               if(g_test_value<3) g_test_value++;
00136               while(nNxtButtonPressed == kRightButton){}
00137           }
00138           switch(g_test_value)
00139           {
00140           case 1:
00141           servo[grabber_left] = GRABBER_LEFT_OPEN;
00142           servo[grabber_right] = GRABBER_RIGHT_OPEN;
00143               break;
00144           case 2:
00145           servo[grabber_left] = GRABBER_LEFT_MID;
00146           servo[grabber_right] = GRABBER_RIGHT_MID;
00147               break;
00148           case 3:
00149           servo[grabber_left] = GRABBER_LEFT_CLOSE;
00150           servo[grabber_right] = GRABBER_RIGHT_CLOSE;
00151               break;
00152           }
00153       break;
00154       //---------------------------------
00155       // sky hook
00156       //---------------------------------
00157   case 6:
00158       if(nNxtButtonPressed == kLeftButton)
00159       {
00160           motor[sky_hook] = g_robot_lift_up;
00161           g_test_value = g_robot_lift_up;
00162       }
00163       else if(nNxtButtonPressed == kRightButton)
00164       {
00165           motor[sky_hook] = g_robot_lift_down;
00166           g_test_value = g_robot_lift_down;
00167       }
00168       else
00169       {
00170           motor[sky_hook] = 0;
00171           g_test_value = 0;
00172       }
```

```
00173         break;
00174         //---------------------------------
00175         // roger slide
00176         //---------------------------------
00177     case 7:
00178         if(nNxtButtonPressed == kLeftButton)
00179         {
00180             servo[roger_slide] = 255;
00181             g_test_value = 255;
00182         }
00183         else if(nNxtButtonPressed == kRightButton)
00184         {
00185             servo[roger_slide] = 0;
00186             g_test_value = 0;
00187         }
00188         else
00189         {
00190             servo[roger_slide] = 127;
00191             g_test_value = 127;
00192         }
00193         break;
00194     }
00195 }
00196 PlaySoundFile("! Click.rso");
00197 }
00198
00199 #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

```
00001
00013 #ifndef ABS_STOP_ROBOT_H
00014 #define ABS_STOP_ROBOT_H
```

```
00015
00016 void abs_stop_robot()
00017 {
00018     motor[left_motor] = 0;
00019     motor[right_motor] = 0;
00020     motor[block_lift_motor] = 0;
00021     motor[block_lift_motor2] = 0;
00022     motor[sky_hook] = 0;
00023     servo[roger_slide] = 127;
00024 }
00025
00026 #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

```
00001
00013 #ifndef ABS_TELE_OP_INITIALIZE_H
00014 #define ABS_TELE_OP_INITIALIZE_H
00015
00016 void abs_tele_op_initialize()
00017 {
00018     if(joystick.joy1_TopHat == -1) g_joy1_enabled = true;
00019     if(joystick.joy2_TopHat == -1) g_joy2_enabled = true;
00020
00021     servo[abdd] = g_abdd_down;
00022
00023     StartTask(screen);
00024     g_screen_state = S_MISC_SHOW;
00025     getJoystickSettings(joystick);
00026 }
00027 #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;
```

```
00031
00032     if(e_stop == TURN_TO)
00033     {
00034         if(dir == COUNTERCLOCKWISE)
00035         {
00036             if(degree<g_recont_heading) target = -(g_recont_heading-degree);
00037             else target = -(360-(degree-g_recont_heading));
00038         }
00039         else
00040         {
00041             if(degree<g_recont_heading) target = 360-(g_recont_heading-degree);
00042             else target = degree-g_recont_heading;
00043         }
00044         abs_turn(dir, turn_method, TURN, target, speed);
00045         PlaySoundFile("! Click.rso");
00046     }
00047     else
00048     {
00049         //-------------------------
00050         // swing turn
00051         //-------------------------
00052         if(turn_method == SWING)
00053         {
00054             if(dir == COUNTERCLOCKWISE)
00055             {
00056                 motor[right_motor] = speed;
00057                 motor[left_motor] = 0;
00058             }
00059             else
00060             {
00061                 motor[right_motor] = 0;
00062                 motor[left_motor] = speed;
00063             }
00064         }
00065
00066         //-------------------------
00067         // point turn
00068         //-------------------------
00069         else
00070         {
00071             if(dir == COUNTERCLOCKWISE)
00072             {
00073                 motor[right_motor] = speed;
00074                 motor[left_motor] = -speed;
00075             }
00076             else
00077             {
00078                 motor[right_motor] = -speed;
00079                 motor[left_motor] = speed;
00080             }
00081         }
00082     }
00083     //-------------------------
00084     // turn condition
00085     //-------------------------
00086
00087     if(e_stop == TURN)
00088     {
00089         while(i < 5)
00090         {
00091             if (abs(g_rel_heading) > abs(degree)) i++;
00092             nxtDisplayCenteredBigTextLine(5, "%d", g_recont_heading);
00093         }
00094         motor[right_motor] = 0;
00095         motor[left_motor] = 0;
00096     }
00097 }
00098
00099 #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

```
00001 #pragma config(Hubs,  S1, HTMotor,  HTMotor,  HTMotor,  HTServo)
00002 #pragma config(Sensor, S1,     ,               sensorI2CMuxController)
00003 #pragma config(Sensor, S2,     GYRO_MUX,       sensorI2CCustom)
00004 #pragma config(Sensor, S3,     SENSOR_MUX,     sensorI2CCustom)
00005 #pragma config(Sensor, S4,     angle_sensor,   sensorI2CCustom)
00006 #pragma config(Motor,  mtr_S1_C1_1,     block_lift_motor, tmotorTetrix, openLoop, encoder)
00007 #pragma config(Motor,  mtr_S1_C1_2,     sky_hook,      tmotorTetrix, openLoop, reversed, encoder)
```

```
00008 #pragma config(Motor,  mtr_S1_C2_1,     jolly_roger,   tmotorTetrix, openLoop)
00009 #pragma config(Motor,  mtr_S1_C2_2,     block_lift_motor2, tmotorTetrix, openLoop)
00010 #pragma config(Motor,  mtr_S1_C3_1,     right_motor,   tmotorTetrix, openLoop)
00011 #pragma config(Motor,  mtr_S1_C3_2,     left_motor,    tmotorTetrix, openLoop)
00012 #pragma config(Servo,  srvo_S1_C4_1,    grabber_right,        tServoStandard)
00013 #pragma config(Servo,  srvo_S1_C4_2,    grabber_left,         tServoStandard)
00014 #pragma config(Servo,  srvo_S1_C4_3,    roger_slide,          tServoContinuousRotation)
00015 #pragma config(Servo,  srvo_S1_C4_4,    light_sensor,         tServoStandard)
00016 #pragma config(Servo,  srvo_S1_C4_5,    servo5,               tServoNone)
00017 #pragma config(Servo,  srvo_S1_C4_6,    abdd,                 tServoStandard)
00018 //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
00019
00030 /*Includes*/
00031
00032 //-----------------------
00033 // sensor/mux/joystick includes
00034 //-----------------------
00035
00036 #include "joystickdriver.c"
00037 #include "lib/xander/hitechnic-sensormux.h"
00038 #include "lib/xander/hitechnic-irseeker-v2.h"
00039 #include "lib/xander/hitechnic-gyro.h"
00040 #include "lib/xander/hitechnic-angle.h"
00041 #include "lib/xander/hitechnic-accelerometer.h"
00042
00043 //-----------------------
00044 // custom functions includes
00045 //-----------------------
00046
00047 #include "lib/global_varaibles.h"
00048 #include "lib/abs_selection_number.h"
00049 #include "lib/abs_selection_custom.h"
00050 #include "lib/abs_selection_quick.h"
00051 #include "lib/abs_selection_program.h"
00052 #include "lib/abs_screen.h"
00053 #include "lib/abs_gyro_cal.h"
00054 #include "lib/math_utils.h"
00055 #include "lib/abs_sensors.h"
00056 #include "abs_move_utils.h"
00057 #include "lib/abs_turn.h"
00058 #include "abs_gyro_drive.h"
00059 #include "lib/abs_drive.h"
00060 #include "lib/abs_initialize.h"
00061 #include "lib/abs_motor.h"
00062 #include "lib/abs_stop_robot.h"
00063
00064 //-----------------------
00065 // auto mission includes
00066 //-----------------------
00067
00068 #include "lib/abs_end_r1.h"
00069 #include "lib/abs_end_r2.h"
00070
00071 #include "lib/abs_s1_mission_execute.h"
00072 #include "lib/abs_s2_mission_execute.h"
00073 #include "lib/abs_s3_mission_execute.h"
00074 #include "lib/abs_s4_mission_execute.h"
00075
00076 //=======================================
00077 // Main program
00078 //=======================================
00079
00080 task main()
00081 {
00082     initialize();
00083     while(true)
00084     {
00085         g_rel_heading = 0;
00086         switch(g_start_point)
00087         {
00088         case 1:
00089             abs_s1_mission_execute();
00090             break;
00091         case 2:
00092             abs_s2_mission_execute();
00093             break;
00094         case 3:
00095             abs_s3_mission_execute();
00096             break;
00097         case 4:
00098             abs_s4_mission_execute();
```

```
00099             break;
00100         }
00101     }
00102 }
```

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

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

```
00001 #pragma systemFile // treat as system file to eliminate warnings for unused variables
00002
00015 //
00016 //*********************************************************
00017 // Define sensor multiplexor connectivity and port allocations
00018 //*********************************************************
00019
00020 const tMUXSensor HTIRS2 = msensor_S3_1;     // HiTechnic Infrared sensor
00021 const tMUXSensor HTAC = msensor_S3_2;
00022 const tMUXSensor HTGYRO = msensor_S2_1;    // HiTechnic GYRO sensor
00023 const tMUXSensor HTIRS2_2 = msensor_S3_3;     // HiTechnic Infrared sensor 2
00024 //const tMUXSensor HTANG = msensor_S3_3;
00025 const tMUXSensor LEGOLS = msensor_S3_4;
00026
00027 bool g_gyro_true = false;
00028
00029 //*********************************************************
00030 // Robot constants
00031 //*********************************************************
00032
00055 #define INT_ANGLE_SENSOR_CIRCUMFERENCE 18
00056 #define FLOAT_ANGLE_SENSOR_CIRCUMFERENCE 17.6
00057 #define DRIVE_WHEELS_CIRCUMFERENCE 26
00058
00059 #define GRABBER_LEFT_OPEN 3
00060 #define GRABBER_RIGHT_OPEN 245
00061 #define GRABBER_LEFT_MID 60
00062 #define GRABBER_RIGHT_MID 180
00063 #define GRABBER_LEFT_CLOSE 120
00064 #define GRABBER_RIGHT_CLOSE 131
00065
00092 const int g_block_speed_down = -60;
00093 const int g_block_speed_up = 100;
00094
00095 const int g_robot_lift_down = -40;
00096 const int g_robot_lift_up = 100;
00097
00098 const int g_flag_speed_down = 90;
00099 const int g_flag_speed_right = 20;
00100 const int g_flag_speed_up = -90;
00101 const int g_flag_speed_left = -20;
00102
00103 const int g_abdd_up = 10;
00104 const int g_abdd_down = 235;
00105
00106 const int g_gyro_adjust = 10;
00107
00108 //*********************************************************
00109 // auto selection points
00110 //*********************************************************
00134 typedef enum
00135 {
00136     SELECTION_START_POINT,
00137     SELECTION_START_DELAY,
00138     SELECTION_MISSION_POINT,
00139     SELECTION_MISSION_DELAY,
00140     SELECTION_END_POINT,
00141     SELECTION_SUB_GRABBERS,
00142     SELECTION_GYRO_CAL,
00143     SELECTION_SELECTION_TYPE,
00144     SELECTION_GRAPH_NUMBER_INPUT,
00145     SELECTION_QUICK_INPUT,
00146     SELECTION_SUB_RAMP
00147 } e_auto_selection_points;
00148
00149 e_auto_selection_points g_auto_selection_point =
      SELECTION_START_POINT;
00150
00151 //*********************************************************
00152 // auto selection type options
00153 //*********************************************************
00163 typedef enum
00164 {
00165     SELECTION_TYPE_NUMBER,
00166     SELECTION_TYPE_CUSTOM,
00167     SELECTION_TYPE_QUICK
00168 } e_selection_types;
00169
```

```
00170 e_selection_types selection_type = SELECTION_TYPE_CUSTOM;
00171
00172 //=========================================================
00173 // auto sub selections
00174 //=========================================================
00183 typedef enum
00184 {
00185     SUB_SELECTION_GRABBERS_OUT,
00186     SUB_SELECTION_GRABBERS_IN
00187 } e_auto_sub_selection;
00188
00189 e_auto_sub_selection g_auto_grabber_selections =
      SUB_SELECTION_GRABBERS_IN;
00190
00199 typedef enum
00200 {
00201     SUB_SELECTION_RAMP_STOP,
00202     SUB_SELECTION_RAMP_CONTINUED
00203 } e_auto_sub_selection_ramp;
00204
00205 e_auto_sub_selection_ramp g_auto_grabber_selection_ramp_options =
      SUB_SELECTION_RAMP_STOP;
00206
00207 //=========================================================
00208 // auto movements
00209 //=========================================================
00210 int g_to_turn_dist = 0;
00211
00212 bool g_IR_angle_dist_complete = false;
00213
00214 const int g_forward_crate1_to_turn_dist = 135;
00215 const int g_forward_crate2_to_turn_dist = 110;
00216 const int g_forward_crate3_to_turn_dist = 60;
00217 const int g_forward_crate4_to_turn_dist = 35;
00218
00219 const int g_backwards_crate1_to_turn_dist = 45;
00220 const int g_backwards_crate2_to_turn_dist = 70;
00221 const int g_backwards_crate3_to_turn_dist = 120;
00222 const int g_backwards_crate4_to_turn_dist = 145;
00223
00224 //=========================================================
00225 // Smoke test varaibles
00226 //=========================================================
00227
00228 int g_smoke_test_num = 1;
00229 int g_smoke_test_total = 12;
00230 int g_smoke_run = false;
00231 int g_test_value = 0;
00232
00233 //=========================================================
00234 // auto number input variable
00235 //=========================================================
00236
00237 int g_intput_array[6];
00238
00239 //=========================================================
00240 // Misc
00241 //=========================================================
00242
00243 int g_debug_time_1 = 0;
00244 int g_debug_time_2 = 0;
00245
00246 int g_auto_ending_points = 4;
00247 int g_travel_dist = 0;
00248 int g_auto_starting_points = 4;
00249 int g_auto_missions = 10;
00250 int g_drive_heading = 0;
00251 int g_ir_heading = 5;
00252 bool g_program_done = false;
00253
00254 bool g_joy1_enabled = false;
00255 bool g_joy2_enabled = false;
00256
00257 int g_selection_value = 0;
00258
00259 //=========================================================
00260 // Define user configurable parameters
00261 //=========================================================
00262 int g_end_point = 1;
00263 int g_start_point = 1;
00264 int g_mission_number = 1;
```

```
00265 int g_delay = 0;
00266 int g_end_delay = 0;
00267 int g_start_delay = 0;
00268 int g_gyro_cal_time = 5;
00269
00270 //===============================================================
00271 // Gyro variables
00272 //===============================================================
00273 int g_gyro_noise = 0;
00274 long g_start_time = 0;
00275 int g_drift = 0;
00276 float g_const_heading = 0;
00277 float g_rel_heading = 0;
00278 long g_curr_time = 0;
00279 long g_prev_time = 0;
00280 int g_raw_gyro = 0;
00281 int g_recont_heading = 0; //this is the recalculated const gyro heading
00282
00301 int g_light_sensor;
00302 int g_bearing_ac1 = 0;
00303 int g_bearing_ac2 = 0;
00304 float g_ir_bearing1 = 0.0;
00305 float g_ir_bearing2 = 0.0;
00306 int g_acs1[5];
00307 int g_acs2[5];
00308 float g_curr_dir1 = 0.0;
00309 float g_curr_dir2 = 0.0;
00310 int g_misc = 0;
00311 bool g_reset_angle = false;
00312
00313 //-----------------------------
00314 // accelermoeter variables
00315 //-----------------------------
00316 int g_accelermoeter_sensor = 0;
00317 int g_x_axis = 0;
00318 int g_y_axis = 0;
00319 int g_z_axis = 0;
00320 const int g_target_angle = 110;
00321 ubyte g_accelermoeter_reads = 0;
00322 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};
00323 ubyte g_accelermoeter_total_value = 0;
00324 int g_accelermoeter_average = 0;
00325
00326 int g_sensor_num = 1;
00327 int g_sensor_max = 4;
00328 int g_sensor_value = 0;
00329 int g_sensor_value2 = 0;
00342 #define ST_GYRO 1
00343 #define ST_IR 2
00344 #define ST_ACCELEROMETER 3
00345 #define ST_TILT 4
00346
00347 bool g_sensor_reference_drive = false;
00348
00349 string g_sensor_list [] = {
00350     "unknown ",
00351     "gyro    ",
00352     "IR   IR2",
00353     "accel   ",
00354     "tilt    "};
00355
00356 string g_basic_word_list [] = {
00357     "unknown ",
00358     "in      ",
00359     "out     ",
00360     "yes     ",
00361     "no      "};
00362
00363 //===============================================================
00364 // Define screen related variables
00365 //===============================================================
00419 #define S_CLEAR 0
00420 #define S_MISSION 1
00421 #define S_DELAY 2
00422 #define S_CAL_TIME 3
00423 #define S_GYRO_CAL 4
00424 #define S_READY 5
00425 #define S_DELAY_WAIT 6
00426 #define S_GYRO_SHOW 7
00427 #define S_ERROR 8
```

```
00428 #define S_SMOKE_TEST 9
00429 #define S_SMOKE_RUN1 10
00430 #define S_SMOKE_RUN2 11
00431 #define S_SMOKE_RUN3 12
00432 #define S_SCREEN_CALL 13
00433 #define S_IR_SHOW 14
00434 #define S_AC_SHOW 15
00435 #define S_MISC_SHOW 16
00436 #define S_STARTING_POINT 17
00437 #define S_ENDING_POINT 18
00438 #define S_SELECTION_SUB_GRABBERS 19
00439 #define S_ANGLE_SHOW 20
00440 #define S_TIME_SHOW 21
00441 #define S_SELECTION_TYPE 22
00442 #define S_NUMBER_SELECTION 23
00443 #define S_SELECTION_SUB_RAMP 24
00444 #define S_MISSION_SHOW 25
00445 #define S_QUICK_SELECTION 26
00446
00447 int g_screen_state = 1;
00448
00449 //===============================================================
00450 // Define graph selection variables
00451 //===============================================================
00452
00453 int g_graph_selection_tab = 0;
00454 long g_graph_selection_number = 10000;
00455
00456 //===============================================================
00457 // Define error numbers
00458 //===============================================================
00475 #define ERR_NONE 0
00476 #define ERR_GYRO_CAL 1
00477 #define ERR_GYRO_MUX 2
00478 #define ERR_SENSOR_MUX 3
00479 #define ERR_JOYSTICKS 4
00480 #define ERR_ACCELERMOETER 5
00481
00482 int g_error = 0;
00483
00484 //=============================================================================
00485 // Define the text to be displayed for each starting point line 1
00486 //=============================================================================
00487 string g_starting_names1 [] = {
00488     "       ",
00489     "S1     ",
00490     "S2     ",
00491     "S3     ",
00492     "S4   ",
00493     "Test 5  ",
00494     "Test 6  ",
00495     "Test 7  ",
00496     "Test 8  ",
00497     "Test 9  ",
00498     "Test 10 ",
00499     "Test 11 ",
00500     "Test 12 ",
00501     "Test 13 ",
00502     "Test 14 ",
00503     "Test 15 ",
00504     "Test 16 ",
00505     "Test 17 ",
00506     "Test 18 ",
00507     "Test 19 ",
00508     "Test 20 ",
00509     "Test 21 ",
00510     "Test 22 "};
00511
00512 //=============================================================================
00513 // Define the text to be displayed for each starting point line 2
00514 //=============================================================================
00515 string g_starting_names2 [] = {
00516     "       ",
00517     "       ",
00518     "Test 2  ",
00519     "Test 3  ",
00520     "Test 4  ",
00521     "Test 5  ",
00522     "Test 6  ",
00523     "Test 7  ",
00524     "Test 8  ",
```

```
00525     "Test 9  ",
00526     "Test 10 ",
00527     "Test 11 ",
00528     "Test 12 ",
00529     "Test 13 ",
00530     "Test 14 ",
00531     "Test 15 ",
00532     "Test 16 ",
00533     "Test 17 ",
00534     "Test 18 ",
00535     "Test 19 ",
00536     "Test 20 ",
00537     "Test 21 ",
00538     "Test 22 "};
00539
00540 //============================================================================
00541 // Define the text to be displayed for each ending point line 1
00542 //============================================================================
00543 string g_ending_names1 [] = {
00544     "        ",
00545     "Stop    ",
00546     "Ramp 1  ",
00547     "Ramp 2  ",
00548     "Test 4  ",
00549     "Test 5  ",
00550     "Test 6  ",
00551     "Test 7  ",
00552     "Test 8  ",
00553     "Test 9  ",
00554     "Test 10 ",
00555     "Test 11 ",
00556     "Test 12 ",
00557     "Test 13 ",
00558     "Test 14 ",
00559     "Test 15 ",
00560     "Test 16 ",
00561     "Test 17 ",
00562     "Test 18 ",
00563     "Test 19 ",
00564     "Test 20 ",
00565     "Test 21 ",
00566     "Test 22 "};
00567
00568 //============================================================================
00569 // Define the text to be displayed for each ending point line 2
00570 //============================================================================
00571 string g_ending_names2 [] = {
00572     "        ",
00573     "        ",
00574     "        ",
00575     "Test 3  ",
00576     "Test 4  ",
00577     "Test 5  ",
00578     "Test 6  ",
00579     "Test 7  ",
00580     "Test 8  ",
00581     "Test 9  ",
00582     "Test 10 ",
00583     "Test 11 ",
00584     "Test 12 ",
00585     "Test 13 ",
00586     "Test 14 ",
00587     "Test 15 ",
00588     "Test 16 ",
00589     "Test 17 ",
00590     "Test 18 ",
00591     "Test 19 ",
00592     "Test 20 ",
00593     "Test 21 ",
00594     "Test 22 "};
00595
00596 //============================================================================
00597 // Define the text to be displayed for each mission
00598 //============================================================================
00599 string g_mission_names1 [] = {
00600     "        ",
00601     "IR crate",
00602     "crate 4 ",
00603     "crate 3 ",
00604     "crate 2 ",
00605     "crate 1 ",
```

```
00606     "defence ",
00607     "Test 7  ",
00608     "Test 8  ",
00609     "Test 9  ",
00610     "Test 10 ",
00611     "Test 11 ",
00612     "Test 12 ",
00613     "Test 13 ",
00614     "Test 14 ",
00615     "Test 15 ",
00616     "Test 16 ",
00617     "Test 17 ",
00618     "Test 18 ",
00619     "Test 19 ",
00620     "Test 20 ",
00621     "Test 21 ",
00622     "Test 22 "};
00623
00624 //================================================================================
00625 // Define the text to be displayed on the second line for each mission
00626 //================================================================================
00627 string g_mission_names2 [] = {
00628     "        ",
00629     "Test 1  ",
00630     "Test 2  ",
00631     "Test 3  ",
00632     "Test 4  ",
00633     "Test 5  ",
00634     "score 4 ",
00635     "score 3 ",
00636     "Test 8  ",
00637     "Test 9  ",
00638     "Test 10 ",
00639     "Test 11 ",
00640     "Test 12 ",
00641     "Test 13 ",
00642     "Test 14 ",
00643     "Test 15 ",
00644     "Test 16 ",
00645     "Test 17 ",
00646     "Test 18 ",
00647     "Test 19 ",
00648     "Test 20 ",
00649     "Test 21 ",
00650     "Test 22 "};
00651
00652 //================================================================================
00653 // Define the text to be displayed for quick selection
00654 //================================================================================
00655 string g_quick_names1 [] = {
00656     "Unknown ",
00657     "S1 IR E1",
00658     "S1 IR E2",
00659     "Test 3  ",
00660     "Test 4  ",
00661     "Test 5  ",
00662     "Test 6  "};
00663
00664 string g_quick_names2 [] = {
00665     "Unknown ",
00666     "Test 1  ",
00667     "Test 2  ",
00668     "Test 3  ",
00669     "Test 4  ",
00670     "Test 5  ",
00671     "Test 6  "};
00672
00673 int g_quick_mission = 1;
00674 int g_max_quick_missions = 6;
00675
00676 //================================================================================
00677 // Define the text to be displayed for the errors
00678 //================================================================================
00679 string g_error_list1 [] = {
00680     "Unknown ",
00681     "GyroCal ",
00682     "Gyro    ",
00683     "Sensor  ",
00684     "joystick",
00685     "Test 5  ",
00686     "Test 6  ",
```

```
00687     "Test 7  ",
00688     "Test 8  ",
00689     "Test 9  ",
00690     "Test 10 ",
00691     "Test 11 ",
00692     "Test 12 ",
00693     "Test 13 ",
00694     "Test 14 ",
00695     "Test 15 ",
00696     "Test 16 ",
00697     "Test 17 ",
00698     "Test 18 ",
00699     "Test 19 ",
00700     "Test 20 ",
00701     "Test 21 ",
00702     "Test 22 "};
00703
00704 string g_error_list2 [] = {
00705     "error   ",
00706     "Failure ",
00707     "Mux     ",
00708     "Mux     ",
00709     "fail    ",
00710     "Test 5  ",
00711     "Test 6  ",
00712     "Test 7  ",
00713     "Test 8  ",
00714     "Test 9  ",
00715     "Test 10 ",
00716     "Test 11 ",
00717     "Test 12 ",
00718     "Test 13 ",
00719     "Test 14 ",
00720     "Test 15 ",
00721     "Test 16 ",
00722     "Test 17 ",
00723     "Test 18 ",
00724     "Test 19 ",
00725     "Test 20 ",
00726     "Test 21 ",
00727     "Test 22 "};
00728
00729 //============================================================================
00730 // Define the text to be displayed for smoke test line 1
00731 //============================================================================
00732 string g_smoke_test1 [] = {
00733     "Unknown ",
00734     "Jolly   ",
00735     "Drive   ",
00736     "Sensor  ",
00737     "Block   ",
00738     "Grabbers",
00739     "sky hook",
00740     "roger   ",
00741     "Test 8  ",
00742     "Test 9  ",
00743     "Test 10 ",
00744     "Test 11 ",
00745     "Test 12 ",
00746     "Test 13 ",
00747     "Test 14 ",
00748     "Test 15 ",
00749     "Test 16 ",
00750     "Test 17 ",
00751     "Test 18 ",
00752     "Test 19 ",
00753     "Test 20 ",
00754     "Test 21 ",
00755     "Test 22 "};
00756
00757 //============================================================================
00758 // Define the text to be displayed for smoke test line 2
00759 //============================================================================
00760 string g_smoke_test2 [] = {
00761     "Unknown ",
00762     "Roger   ",
00763     "Train   ",
00764     "Sensor  ",
00765     "Lift    ",
00766     "        ",
00767     "        ",
```

```
00768     "slide   ",
00769     "Test 8  ",
00770     "Test 9  ",
00771     "Test 10 ",
00772     "Test 11 ",
00773     "Test 12 ",
00774     "Test 13 ",
00775     "Test 14 ",
00776     "Test 15 ",
00777     "Test 16 ",
00778     "Test 17 ",
00779     "Test 18 ",
00780     "Test 19 ",
00781     "Test 20 ",
00782     "Test 21 ",
00783     "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

```
00001 #pragma config(Hubs,  S1, HTMotor,  HTMotor,  HTMotor,  HTServo)
00002 #pragma config(Sensor, S1,     ,             sensorI2CMuxController)
00003 #pragma config(Sensor, S2,     GYRO_MUX,     sensorI2CCustom)
00004 #pragma config(Sensor, S3,     SENSOR_MUX,   sensorI2CCustom)
00005 #pragma config(Sensor, S4,     angle_sensor, sensorI2CCustom)
00006 #pragma config(Motor,  mtr_S1_C1_1,     block_lift_motor, tmotorTetrix, openLoop, encoder)
00007 #pragma config(Motor,  mtr_S1_C1_2,     sky_hook,      tmotorTetrix, openLoop, reversed, encoder)
00008 #pragma config(Motor,  mtr_S1_C2_1,     jolly_roger,   tmotorTetrix, openLoop)
00009 #pragma config(Motor,  mtr_S1_C2_2,     block_lift_motor2, tmotorTetrix, openLoop)
00010 #pragma config(Motor,  mtr_S1_C3_1,     right_motor,   tmotorTetrix, openLoop, reversed)
00011 #pragma config(Motor,  mtr_S1_C3_2,     left_motor,    tmotorTetrix, openLoop)
00012 #pragma config(Servo,  srvo_S1_C4_1,    grabber_right,        tServoStandard)
00013 #pragma config(Servo,  srvo_S1_C4_2,    grabber_left,         tServoStandard)
00014 #pragma config(Servo,  srvo_S1_C4_3,    roger_slide,          tServoContinuousRotation)
00015 #pragma config(Servo,  srvo_S1_C4_4,    light_sensor,         tServoStandard)
00016 #pragma config(Servo,  srvo_S1_C4_5,    servo5,               tServoNone)
00017 #pragma config(Servo,  srvo_S1_C4_6,    abdd,                 tServoStandard)
00018 //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
00019
00030 //-----------------------
00031 // sensor/mux/joystick includes
00032 //-----------------------
00033
00034 #include "JoystickDriver.c"
00035 #include "lib/xander/hitechnic-sensormux.h"
00036 #include "lib/xander/hitechnic-gyro.h"
00037 #include "lib/xander/hitechnic-angle.h"
00038 #include "lib/xander/hitechnic-irseeker-v2.h"
00039 #include "drivers/hitechnic-accelerometer.h"
00040
00041 //-----------------------
00042 // custom functions includes
00043 //-----------------------
```

```
00044 #include "lib/global_varaibles.h"
00045 #include "lib/abs_screen.h"
00046 #include "lib/math_utils.h"
00047 #include "lib/abs_sensors.h"
00048 #include "lib/abs_smoke_execute.h"
00049
00050 //=================================
00051 // main program
00052 //=================================
00053 task main()
00054 {
00055     StartTask(screen);
00056     StartTask(abs_sensors);
00057     g_test_value = 1;
00058     while(true)
00059     {
00060         while(nNxtButtonPressed == kEnterButton){}
00061         g_screen_state = S_SMOKE_TEST;
00062         //--------------------------------------
00063         // Start of mission selection
00064         //--------------------------------------
00065
00066         while(nNxtButtonPressed != kEnterButton)
00067         {
00068             if(nNxtButtonPressed == kRightButton)
00069             {
00070                 PlaySoundFile("! Click.rso");
00071                 while(nNxtButtonPressed == kRightButton){}
00072                 if(g_smoke_test_num < g_smoke_test_total) g_smoke_test_num++;
00073                 }
00074             if(nNxtButtonPressed == kLeftButton)
00075             {
00076                 PlaySoundFile("! Click.rso");
00077                 while(nNxtButtonPressed == kLeftButton){}
00078                 if(g_smoke_test_num > 1) g_smoke_test_num--;
00079             }
00080         }
00081         PlaySoundFile("! Click.rso");
00082         while(nNxtButtonPressed == kEnterButton){}
00083         eraseDisplay();
00084
00085         abs_smoke_execute();
00086     }
00087 }
```

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

```
00001 #pragma config(Hubs,  S1, HTMotor,  HTMotor,  HTMotor,  HTServo)
00002 #pragma config(Sensor, S2,     GYRO_MUX,       sensorI2CCustom)
00003 #pragma config(Sensor, S3,     SENSOR_MUX,     sensorI2CCustom)
00004 #pragma config(Sensor, S4,     angle_sensor,   sensorI2CCustom)
00005 #pragma config(Motor,  mtr_S1_C1_1,     block_lift_motor, tmotorTetrix, openLoop, encoder)
00006 #pragma config(Motor,  mtr_S1_C1_2,     sky_hook,      tmotorTetrix, openLoop, reversed, encoder)
00007 #pragma config(Motor,  mtr_S1_C2_1,     jolly_roger,   tmotorTetrix, openLoop, reversed)
00008 #pragma config(Motor,  mtr_S1_C2_2,     block_lift_motor2, tmotorTetrix, openLoop)
00009 #pragma config(Motor,  mtr_S1_C3_1,     right_motor,   tmotorTetrix, openLoop, reversed)
00010 #pragma config(Motor,  mtr_S1_C3_2,     left_motor,    tmotorTetrix, openLoop)
00011 #pragma config(Servo,  srvo_S1_C4_1,    grabber_right,        tServoStandard)
00012 #pragma config(Servo,  srvo_S1_C4_2,    grabber_left,         tServoStandard)
00013 #pragma config(Servo,  srvo_S1_C4_3,    roger_slide,          tServoContinuousRotation)
00014 #pragma config(Servo,  srvo_S1_C4_4,    light_sensor,         tServoStandard)
00015 #pragma config(Servo,  srvo_S1_C4_6,    abdd,                 tServoStandard)
00016 //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
00017
00032 //-----------------------
00033 // sensor/mux/joystick includes
00034 //-----------------------
00035
00036 #include "JoystickDriver.c"
00037 #include "lib/xander/hitechnic-sensormux.h"
00038 #include "drivers/hitechnic-accelerometer.h"
00039 #include "lib/xander/hitechnic-irseeker-v2.h"
00040 #include "lib/xander/hitechnic-gyro.h"
00041 #include "lib/xander/hitechnic-angle.h"
00042
00043 //-----------------------
00044 // custom functions includes
00045 //-----------------------
00046
00047 #include "lib/global_varaibles.h"
00048 #include "lib/abs_screen.h"
00049 #include "lib/abs_teleop_utils.h"
00050 #include "lib/abs_joystick_drive.h"
00051 #include "lib/abs_joystick_gunner.h"
00052 #include "lib/abs_tele_op_initialize.h"
00053 //=======================================
00054 // Main program
00055 //=======================================
00056
00057 task main ()
00058 {
00059     abs_tele_op_initialize();
00060     StartTask(abs_joystick_gunner);
00061     while(g_program_done==false)
00062     {
00063         abs_joystick_drive(LINEAR);
00064     }
00065 }
```