**1200 Worlds 2017.c**

Cortex Pin Configuration

|  |  |  |  |
| --- | --- | --- | --- |
| # | Analog Pinout | Digital Pinout | Motor Pinout |
| 1 | Lift Potentiometer | Left Encoder | Back Left Drive |
| 2 | Gyroscope – Viewed As Analog | Left Encoder | Lift Motor 1 |
| 3 | Power Expander Power | Right Encoder | Front Right Drive |
| 4 | N/A | Right Encoder | Lift Motor 2 |
| 5 | N/A | Solenoid 1 | Lift Motor 3 |
| 6 | N/A | Solenoid 2 | Lift Motor 4 |
| 7 | N/A | N/a | Lift Motor 5 |
| 8 | N/A | N/A | Front Left Drive |
| 9 | N/A | N/A | Lift Motor 6 |
| 10 | N/A | N/A | Back Right Drive |
| 11 | N/A | LCD Encoder | N/A |
| 12 | N/A | LCD Encoder | N/A |

Joystick Controls

|  |  |
| --- | --- |
| Input | Output |
| Joystick Channel 1-2 | Drive System |
| Joystick Channel 3 | Lift System |
| Button 5U | Claw Toggle |

Function #1: **pre\_auton( )**

Purpose: To initialize the robot

* Steps

1. Start LCD Menu, if specified
2. Initialize Homebrew Integration

* Used

1. Run before every match
   1. As specified by the Vex Competition Template

Task #1: **autonomous( )**

Purpose: To effectively execute the autonomous period

Relevant Variables

|  |  |
| --- | --- |
| **Variable** | **Use** |
| bResetFromWatchdogTimer | True if the robot has reset |
| autoValue | Chooses which autonomous program to run |
| autonRunning | Tells the rest of the code the autonomous is running |

Relevant Functions

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| autoCode() | GeneralLib.c |

* Steps

1. Check if the robot has been reset
2. Run the autoCode Function

* Used
  + Run before every match
  1. As specified by the Vex Competition Template

Task #2: **usercontrol ( )**

Purpose: To effectively control the robot during the user control period

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| PID l | PID to run lift to desired spot |
| clawBool | Toggle to turn the claw solenoids on and off |
| leftSpeed | Speed of the left side of the drive |
| rightSpeed | Speed of the Right side of the drive |
| liftSpeed | Speed of the lift system |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| LCDMenu | LCDLib.c |
| pidInit | AutonLib.c |
| liftDrive | GeneralLib.c |
| chassisDrive | GeneralLib.c |
| simpleRecord | GeneralLib.c |
| clearSensors | GeneralLib.c |
| median | GeneralLib.c |

* Steps
  + Check to See if you want to run the lift Toggle
  + Calculate the PID for the Lift
  + See if we need to switch the claw value
  + Calculate speed values
  + Set median filter for Sonar
  + LCD Set
* Used
  + Run every match
  1. As specified by the Vex Competition Template

**GeneralLib.c**

Function #1: bool **shouldWeKeepGoing ( )**

Purpose: To create a simple and effective way to tell autonomous programs whether or not they “should keep going”

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| currentValue | Value the system is at |
| endValue | Value the system wants to get to |
| shouldGo | Returned Boolean to see if the system has reached its conclusion |

* Steps
  + Check to see if the current value is greater than the end value
  + Return true if it is, false otherwise
* Used
  + Autonomous Functions
  + Rerun Code

Function #2: **shouldWeKeepGoingSen ( )**

Purpose: To be able to choose any sensor and direction to determine when we should or should not keep going

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| shouldWeKeepGoing | Returned Value |
| SensroInfo | Variable for type and direction of Sensor Used |
| FinalSensorValue | Sensor Value wanted |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| shouldWeKeepGoingPE | GeneralLib.c |

* Steps
  + Pick a sensor and direction to use based on SensorInfo variable
  + Determine if the system “should keep going”
* Used
  + Rerun Code

Function #3: **chassisDrive ( )**

Purpose: Drive the drive motors

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| leftSpeed | Set speed of left drive |
| rightSpeed | Set speed of right drive |
| reverseChassis | Boolean to reverse the two above variables |

* Steps
  + Set the Motors to the Inputted Variables
* Used
  + Drive Code
  + Autonomous Programs

Function #4: **liftDrive ( )**

Purpose: Drive the lift motors

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| speed | Set speed of lift drive |

* Steps
  + Set the Motors to the Inputted Variables
* Used
  + Drive Code
  + Autonomous Programs

Function #5: **clearSensors ( )**

Purpose: Clear all relevant sensors

* Steps
  + Clear encoder values, gyro integration values, and Timer values
* Used
  + Drive Code
  + Autonomous Programs

Function #6: **MaxMinClip ( )**

Purpose: To keep any value between a maximum and a minimum

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Max | Maximum Possible Value |
| Min | Minimum Possible Value |
| Value | Returned Value |

* Steps
  + Set the Motors to the Inputted Variables
* Used
  + Drive Code
  + Autonomous Programs

Function #9: **autoCode ( )**

Purpose: Choose and execute autonomous

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| auton | Value of auton selected |

* Steps
  + Based on the inputted value, run the specified autonomous
* Used
  + Autonomous Function

**LCDLib.c**

Function #1: **CaseValueFinder ( )**

Purpose: Find the case value for the menu; create infrastructure for the LCD Menu

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| autonRange | Range of the autonSelector for each case |
| numberOfCases | Number of cases for this set menu |

* Steps
  + Using a for loop, determine which case the autonSelector encoder is set to.
  + Return the set case value
* Used
  + Infrastructure of the LCD Menu

Function #2: **SensorsMenu ( )**

Purpose: Viewing sensors via the LCD Screen

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| TopString | Displayed LCD top string |
| BotString | Displayed LCD bottom string |
| CaseValue | Set case value |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| CaseValueFinder | LCDLib.c |

* Steps
  + Choose sensor to see via the CaseValueFinder function
  + Display the Sensor Value to the LCD screen
* Used
  + LCD Menu

Function #3: **MotorHealth ( )**

Purpose: Testing motors via LCD Screen

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| TopString | Displayed LCD top string |
| BotString | Displayed LCD bottom string, displays potential motor speeds |
| CaseValue | Set case value |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| CaseValueFinder | LCDLib.c |

* Steps
  + Choose motor to use via the CaseValueFinder function
  + Display the Sensor Value to the LCD screen
  + If the LCD Button is pressed, run the associated motor
* Used
  + LCD Menu

Function #4: **autonSelection ( )**

Purpose: Select an autonomous program using the LCD Screen

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| TopString | Displayed LCD top string; displays autonomous program’s name |
| BotString | Displayed LCD bottom string, displays Lock Now |
| autoValue | Set case value, for the autonomous selection |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| CaseValueFinder | LCDLib.c |

* Steps
  + Choose autonomous program to use via the CaseValueFinder function
  + Lock in potential autonomous program by clicking one of the LCD buttons
* Used
  + LCD Menu

Task #1: **LCDMenu ( )**

Purpose: To orchestrate the infrastructure of the LCD Screen

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| LCDMenuBool | Displayed LCD top string; displays autonomous program’s name |
| topMenuSwitch | Displayed LCD bottom string, displays Lock Now |
| topMenu | Set case value, for the autonomous selection |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| CaseValueFinder | LCDLib.c |

* Steps
  + Choose LCD program to use via the CaseValueFinder function
    - Set that value to the top menu switch
    - Display the possible function through the LCD Menu
    - Run possible function, within a while loop, if the user clicks a LCD button
  + Display TopString and BotString global variables to the LCD Menu
* Used
  + User Control

**RecorderLib.c**

Function #1: **SensorValueDecode ( )**

Purpose: To choose a sensor and direction for recording autonomous programs

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| PreviousJS | Joystick Previously recorded |
| DirSenVal | Direction and Sensor Indicating Variable |
| SensorInfoString | String to indicate above variable |

* Steps
  + Run through an else if tree to determine which sensor to use
  + Set the DirSenVal and SensorInfoString, depending on the outcome of the else if tree
* Used
  + Rerun Code – Sensor Check

Function #2: **ShamalamE3 ( )**

Purpose: Record an autonomous program by driving the desired autonomous program

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| CurrentJS | Joystick Struct of values currently on the joystick |
| PreviousJS | Joystick Struct of values previously on the joystick |
| Buffer | Amount allowed for speeds to shift |
| N | Number of functions written per line |
| Driver | Variable to indicate who is driving |
| isCh1Move - isButtonMove | Variable to indicate whether certain joysticks or button values have changed |
| SCh1 - SBNs | Strings to hold the values of the current joystick struct |
| isMove | Indicate if we should write a function |
| finalE3Bool | Indicate if we should end the recording function |
| FirstTimeIsMoveLoop | Write whether or not this is the first time in this recording session |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| clearSensors | GeneralLib.c |
| SensorValueDecode | RecorderLib.c |

* Steps
  + Check to see if any of the buttons or joystick values have changed
    - If they have, prepare to write a function to the Debug Stream
  + Write down the joystick values to strings SCh1 to SBNs
  + When indicated by finalE3Bool, end the recording, and send relevant statistics to the Debug Stream
* Used
  + User Control

Function #3: **GetJoystick ( )**

Purpose: Set the Joystick values to the Current Joystick Struct

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| CurrentJS | Joystick Struct that joystick values will be set to |

* Steps
  + Set Joystick Struct CurrentJS to the joystick values
    - Set .Ch14 - .Ch4R to the joystick values
    - Set the .Buttons variable to the buttons on the controller via quasi-bitmap
* Used
  + Rerun Code

**AutonLib.c**

Function #1: **gyroStart ( )**

Purpose: To initialize a gyro struct, so we can integrate the gyro data ourselves

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Gyro | Gyro Struct |
| port | Port the gyro in question is plugged into |
| ave | Average zero rate of the gyro |
| stdDev | Std Dev of the gyro as an analog sensor at zero |
| collect | Variable to collect and hold 2000 points of gyro data |
| vMul | Voltage multiplier, from what it says on the datasheet, to the cortex |

* Steps
  + Take 2000 Data points of the gyro as an analog sensor while the gyro is not moving
    - Find the average and standard deviations of that data
  + Compare that data to what is on the data sheet to find the voltage multiplier
  + Set these variables to the Gyro struct
* Used
  + Any time we would use a gyro

Function #2: **gyroUpdate ( )**

Purpose: To integrate gyro data by hand, via trapezoid rule, giving us a more accurate angle reading

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Gyro | Gyro Struct |
| stdDev | Amount of standard deviations to omit in the data |
| val | Gyro value |
| gyroDif | Difference in current gyro and average zero rate |
| deltaTime | Change in time since last integration |

* Steps
  + Take the raw gyro value
    - Convert this gyro value to degrees per second via variables from the data sheet
  + Compare to the standard deviation of the zero rate of the gyro. If it is greater, integrate, otherwise, do not.
  + Set these variables to the Gyro struct
* Used
  + Any time we would use a gyro

Function #3: **pidInit ( )**

Purpose: Initialize a PID Struct, so we can effectively use PID in autonomous

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| pid | Pointer to the PID Struct |
| kP | P Constant for the PID |
| kI | I Constant for the PID |
| kD | D Constant for the PID |
| Max\_error | Maximum amount the error is allowed to have |
| Min\_error | Minimum amount the error is allowed to have |

* Steps
  + Set the PID struct variables to the inputs of the function
* Used
  + Autonomous Functionality

Function #4: **pidOut ( )**

**Purpose: Update the PID struct, and more importantly, update the speed of the drive in question**

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Error | The distance between the point the sensor is at, and the point we want it to be |
| deltaTime | Time between now and the last PID calculation |
| integral | Accumulation of the amount of error over time |
| Speed | The outputted motor power value from the PID |
| Derivitive | Rate of change of error |

* Steps
  + Calculate necessary variables for PID calculation
    - Find Error
    - Find the accumulation of error (integral)
    - Set integral to zero if we are within a tolerable range of error
    - Find the rate of change (derivative)
  + Calculate speed using those values
  + Set relevant data points of PID struct
* Used
  + Autonomous Functionality

Function #5: **pidOutNC ( )**

Purpose: The same as pidOut, however without any integral limitations, as there are some instances where that only hinders the drive system

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Error | The distance between the point the sensor is at, and the point we want it to be |
| deltaTime | Time between now and the last PID calculation |
| integral | Accumulation of the amount of error over time |
| Speed | The outputted motor power value from the PID |
| Derivitive | Rate of change of error |

* Steps
  + Calculate necessary variables for PID calculation
    - Find Error
    - Find the accumulation of error (integral)
    - Set integral to zero if we are within a tolerable range of error
    - Find the rate of change (derivative)
  + Calculate speed using those values
  + Set relevant data points of PID struct
* Used
  + Autonomous Functionality

Function #6: Drive **( )**

Purpose: To provide an effective autonomous framework

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Tiles | Number of tiles you want to drive forward |
| Lift | Lift Position you want to go to |
| clawState | Do you want to open or close the claw? |
| maxTime | Maximum amount of time PID can run |
| clawStart | When do you want the claw action to take place |
| clawRelease | If you are dumping, what height do you want to open the claw at? |
| liftStart | When do you want the lift function to start> |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| pidInit | AutonLib.c |
| pidOut | AutonLib.c |
| clearSensors | GeneralLib.c |
| liftDrive | GeneralLib.c |
| chassisDrive | GeneralLib.c |

* Steps
  + Initialize the PID systems
  + Set the claw systems
  + Calculate the speeds of the drive, lift, and claw based on the PID
  + Set Motor Speeds
  + Run until the time allotted runs out
* Used
  + Autonomous Functionality

Function #7: Turn **( )**

Purpose: To provide an effective autonomous framework

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Angle | Degree you want to turn to |
| Lift | Lift Position you want to go to |
| clawState | Do you want to open or close the claw? |
| maxTime | Maximum amount of time PID can run |
| clawStart | When do you want the claw action to take place |
| clawRelease | If you are dumping, what height do you want to open the claw at? |
| liftStart | When do you want the lift function to start> |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| pidInit | AutonLib.c |
| pidOut | AutonLib.c |
| clearSensors | GeneralLib.c |
| liftDrive | GeneralLib.c |
| chassisDrive | GeneralLib.c |

* Steps
  + Initialize the PID systems
  + Set the claw systems
  + Calculate the speeds of the drive, lift, and claw based on the PID
  + Set Motor Speeds
  + Run until the time allotted runs out
* Used
  + Autonomous Functionality

Function #8: sTurn **( )**

Purpose: To provide an effective autonomous framework

**Relevant Variables**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| left | Number of Degrees to go forward on the left side |
| Lift | Number of Degrees to go forward on the right side |
| clawState | Do you want to open or close the claw? |
| maxTime | Maximum amount of time PID can run |
| clawStart | When do you want the claw action to take place |
| clawRelease | If you are dumping, what height do you want to open the claw at? |
| liftStart | When do you want the lift function to start> |

**Relevant Functions**

|  |  |
| --- | --- |
| **Function** | **Referenced From** |
| pidInit | AutonLib.c |
| pidOut | AutonLib.c |
| clearSensors | GeneralLib.c |
| liftDrive | GeneralLib.c |
| chassisDrive | GeneralLib.c |

* Steps
  + Initialize the PID systems
  + Set the claw systems
  + Calculate the speeds of the drive, lift, and claw based on the PID
  + Set Motor Speeds
  + Run until the time allotted runs out
* Used
  + Autonomous Functionality