```c
/** @file main.h
 * @brief Header file for global functions
 *
 * Any experienced C or C++ programmer knows the importance of header files. For those who
 * do not, a header file allows multiple files to reference functions in other files without
 * necessarily having to see the code (and therefore causing a multiple definition). To make
 * a function in "opcontrol.c", "auto.c", "main.c", or any other C file visible to the core
 * implementation files, prototype it here.
 *
 * This file is included by default in the predefined stubs in each VEX Cortex PROS Project.
 *
 * Copyright (c) 2011-2016, Purdue University ACM SIGBots.
 * All rights reserved.
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, You can obtain one at http://mozilla.org/MPL/2.0/.
 *
 * PROS contains FreeRTOS (http://www.freertos.org) whose source code may be
 * obtained from http://sourceforge.net/projects/freertos/files/ or on request.
 */

#ifndef MAIN_H_

// This prevents multiple inclusion, which isn't bad for this file but is good practice
#define MAIN_H_

#include <API.h>

// Allow usage of this file in C++ programs
#ifdef __cplusplus
extern "C" {
#endif

// A function prototype looks exactly like its declaration, but with a semicolon instead of
// actual code. If a function does not match a prototype, compile errors will occur.

// Prototypes for initialization, operator control and autonomous

/**
 * Runs the user autonomous code. This function will be started in its own task with the def
 * priority and stack size whenever the robot is enabled via the Field Management System or
 * VEX Competition Switch in the autonomous mode. If the robot is disabled or communications
 * lost, the autonomous task will be stopped by the kernel. Re-enabling the robot will resta
 * the task, not re-start it from where it left off.
 *
```

```
 * Code running in the autonomous task cannot access information from the VEX Joystick. Howe
 * the autonomous function can be invoked from another task if a VEX Competition Switch is n
 * available, and it can access joystick information if called in this way.
 *
 * The autonomous task may exit, unlike operatorControl() which should never exit. If it doe
 * so, the robot will await a switch to another mode or disable/enable cycle.
 */
void autonomous();
/**
 * Runs pre-initialization code. This function will be started in kernel mode one time while
 * VEX Cortex is starting up. As the scheduler is still paused, most API functions will fail
 *
 * The purpose of this function is solely to set the default pin modes (pinMode()) and port
 * states (digitalWrite()) of limit switches, push buttons, and solenoids. It can also safel
 * configure a UART port (usartOpen()) but cannot set up an LCD (lcdInit()).
 */
void initializeIO();
/**
 * Runs user initialization code. This function will be started in its own task with the def
 * priority and stack size once when the robot is starting up. It is possible that the VEXne
 * communication link may not be fully established at this time, so reading from the VEX
 * Joystick may fail.
 *
 * This function should initialize most sensors (gyro, encoders, ultrasonics), LCDs, global
 * variables, and IMEs.
 *
 * This function must exit relatively promptly, or the operatorControl() and autonomous() ta
 * will not start. An autonomous mode selection menu like the pre_auton() in other environme
 * can be implemented in this task if desired.
 */
void initialize();
/**
 * Runs the user operator control code. This function will be started in its own task with t
 * default priority and stack size whenever the robot is enabled via the Field Management Sy
 * or the VEX Competition Switch in the operator control mode. If the robot is disabled or
 * communications is lost, the operator control task will be stopped by the kernel. Re-enabl
 * the robot will restart the task, not resume it from where it left off.
 *
 * If no VEX Competition Switch or Field Management system is plugged in, the VEX Cortex wil
 * run the operator control task. Be warned that this will also occur if the VEX Cortex is
 * tethered directly to a computer via the USB A to A cable without any VEX Joystick attache
 *
 * Code running in this task can take almost any action, as the VEX Joystick is available an
 * the scheduler is operational. However, proper use of delay() or taskDelayUntil() is highl
 * recommended to give other tasks (including system tasks such as updating LCDs) time to ru
 *
```

```
 * This task should never exit; it should end with some kind of infinite loop, even if empty
 */
void operatorControl();

// End C++ export structure
#ifdef __cplusplus
}
#endif

#endif
```