```c
/** @file API.h
 * @brief Provides the high-level user functionality intended for use by typical VEX Cortex
 * programmers.
 *
 * This file should be included for you in the predefined stubs in each new VEX Cortex PROS
 * project through the inclusion of "main.h". In any new C source file, it is advisable to
 * include main.h instead of referencing API.h by name, to better handle any nomenclature
 * changes to this file or its contents.
 *
 * 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 API_H_
#define API_H_


#include <stdlib.h>
#include <stdbool.h>
#include <stdarg.h>
#include <stdint.h>


#ifdef __cplusplus
extern "C" {
#endif

#define JOY_DOWN 1

#define JOY_LEFT 2

#define JOY_UP 4

#define JOY_RIGHT 8

#define ACCEL_X 5

#define ACCEL_Y 6

bool isAutonomous();

bool isEnabled();

bool isJoystickConnected(unsigned char joystick);

bool isOnline();

int joystickGetAnalog(unsigned char joystick, unsigned char axis);

bool joystickGetDigital(unsigned char joystick, unsigned char buttonGroup, unsigned char button);

unsigned int powerLevelBackup();

unsigned int powerLevelMain();

void setTeamName(const char *name);

#define BOARD_NR_ADC_PINS 8
```

```
#define BOARD_NR_GPIO_PINS 27

#define HIGH 1

#define LOW 0

#define INPUT 0x0A

#define INPUT_ANALOG 0x00

#define INPUT_FLOATING 0x04

#define OUTPUT 0x01

#define OUTPUT_OD 0x05

int analogCalibrate(unsigned char channel);

int analogRead(unsigned char channel);

int analogReadCalibrated(unsigned char channel);

int analogReadCalibratedHR(unsigned char channel);

bool digitalRead(unsigned char pin);

void digitalWrite(unsigned char pin, bool value);

void pinMode(unsigned char pin, unsigned char mode);

#define INTERRUPT_EDGE_RISING 1

#define INTERRUPT_EDGE_FALLING 2

#define INTERRUPT_EDGE_BOTH 3

typedef void (*InterruptHandler)(unsigned char pin);

void ioClearInterrupt(unsigned char pin);

void ioSetInterrupt(unsigned char pin, unsigned char edges, InterruptHandler handler);

int motorGet(unsigned char channel);

void motorSet(unsigned char channel, int speed);

void motorStop(unsigned char channel);

void motorStopAll();

void speakerInit();

void speakerPlayArray(const char * * songs);

void speakerPlayRtttl(const char *song);

void speakerShutdown();

#define IME_ADDR_MAX 0x1F

unsigned int imeInitializeAll();

bool imeGet(unsigned char address, int *value);

bool imeGetVelocity(unsigned char address, int *value);
bool imeReset(unsigned char address);

void imeShutdown();

typedef void * Gyro;

int gyroGet(Gyro gyro);

Gyro gyroInit(unsigned char port, unsigned short multiplier);

void gyroReset(Gyro gyro);

void gyroShutdown(Gyro gyro);

typedef void * Encoder;

int encoderGet(Encoder enc);

Encoder encoderInit(unsigned char portTop, unsigned char portBottom, bool reverse);

void encoderReset(Encoder enc);

void encoderShutdown(Encoder enc);

typedef void * Ultrasonic;

int ultrasonicGet(Ultrasonic ult);

Ultrasonic ultrasonicInit(unsigned char portEcho, unsigned char portPing);

void ultrasonicShutdown(Ultrasonic ult);

bool i2cRead(uint8_t addr, uint8_t *data, uint16_t count);

bool i2cReadRegister(uint8_t addr, uint8_t reg, uint8_t *value, uint16_t count);

bool i2cWrite(uint8_t addr, uint8_t *data, uint16_t count);

bool i2cWriteRegister(uint8_t addr, uint8_t reg, uint16_t value);

typedef int FILE;

#define SERIAL_DATABITS_8 0x0000

#define SERIAL_DATABITS_9 0x1000

#define SERIAL_STOPBITS_1 0x0000

#define SERIAL_STOPBITS_2 0x2000

#define SERIAL_PARITY_NONE 0x0000

#define SERIAL_PARITY_EVEN 0x0400

#define SERIAL_PARITY_ODD 0x0600

#define SERIAL_8N1 0x0000

void usartInit(FILE *usart, unsigned int baud, unsigned int flags);

void usartShutdown(FILE *usart);

#define stdout ((FILE *)3)

#define stdin ((FILE *)3)
```

```
#define uart1 ((FILE *)1)

#define uart2 ((FILE *)2)

#ifndef EOF

#define EOF ((int)-1)
#endif

#ifndef SEEK_SET

#define SEEK_SET 0
#endif
#ifndef SEEK_CUR

#define SEEK_CUR 1
#endif
#ifndef SEEK_END

#define SEEK_END 2
#endif

void fclose(FILE *stream);

int fcount(FILE *stream);

int fdelete(const char *file);

int feof(FILE *stream);

int fflush(FILE *stream);

int fgetc(FILE *stream);

char* fgets(char *str, int num, FILE *stream);

FILE * fopen(const char *file, const char *mode);

void fprint(const char *string, FILE *stream);

int fputc(int value, FILE *stream);

int fputs(const char *string, FILE *stream);

size_t fread(void *ptr, size_t size, size_t count, FILE *stream);

int fseek(FILE *stream, long int offset, int origin);

long int ftell(FILE *stream);

size_t fwrite(const void *ptr, size_t size, size_t count, FILE *stream);

int getchar();

void print(const char *string);

int putchar(int value);

int puts(const char *string);

int fprintf(FILE *stream, const char *formatString, ...);

int printf(const char *formatString, ...);

int snprintf(char *buffer, size_t limit, const char *formatString, ...);
```

```c
int sprintf(char *buffer, const char *formatString, ...);

#define LCD_BTN_LEFT 1

#define LCD_BTN_CENTER 2

#define LCD_BTN_RIGHT 4

void lcdClear(FILE *lcdPort);

void lcdInit(FILE *lcdPort);

#ifdef DOXYGEN
void lcdPrint(FILE *lcdPort, unsigned char line, const char *formatString, ...);
#else
void __attribute__ ((format (printf, 3, 4))) lcdPrint(FILE *lcdPort, unsigned char line,
        const char *formatString, ...);
#endif

unsigned int lcdReadButtons(FILE *lcdPort);

void lcdSetBacklight(FILE *lcdPort, bool backlight);

void lcdSetText(FILE *lcdPort, unsigned char line, const char *buffer);

void lcdShutdown(FILE *lcdPort);

#define TASK_MAX 16

#define TASK_MAX_PRIORITIES 6

#define TASK_PRIORITY_LOWEST 0

#define TASK_PRIORITY_DEFAULT 2

#define TASK_PRIORITY_HIGHEST (TASK_MAX_PRIORITIES - 1)

#define TASK_DEFAULT_STACK_SIZE 512

#define TASK_MINIMAL_STACK_SIZE 64

#define TASK_DEAD 0

#define TASK_RUNNING 1

#define TASK_RUNNABLE 2

#define TASK_SLEEPING 3

#define TASK_SUSPENDED 4

typedef void * TaskHandle;

typedef void * Mutex;

typedef void * Semaphore;

typedef void (*TaskCode)(void *);

TaskHandle taskCreate(TaskCode taskCode, const unsigned int stackDepth, void *parameters,
        const unsigned int priority);

void taskDelay(const unsigned long msToDelay);

void taskDelayUntil(unsigned long *previousWakeTime, const unsigned long cycleTime);
void taskDelete(TaskHandle taskToDelete);

unsigned int taskGetCount();

unsigned int taskGetState(TaskHandle task);

unsigned int taskPriorityGet(const TaskHandle task);

void taskPrioritySet(TaskHandle task, const unsigned int newPriority);

void taskResume(TaskHandle taskToResume);

TaskHandle taskRunLoop(void (*fn)(void), const unsigned long increment);

void taskSuspend(TaskHandle taskToSuspend);

Semaphore semaphoreCreate();

bool semaphoreGive(Semaphore semaphore);

bool semaphoreTake(Semaphore semaphore, const unsigned long blockTime);

void semaphoreDelete(Semaphore semaphore);

Mutex mutexCreate();

bool mutexGive(Mutex mutex);

bool mutexTake(Mutex mutex, const unsigned long blockTime);

void mutexDelete(Mutex mutex);

void delay(const unsigned long time);

void delayMicroseconds(const unsigned long us);

unsigned long micros();

unsigned long millis();

void wait(const unsigned long time);

void waitUntil(unsigned long *previousWakeTime, const unsigned long time);

#ifdef __cplusplus
}
#endif

#endif
```