

LabRat Utilities API (TH v2.1)

Revision 2

For examples on how to use this functionality, see the example and test programs included with the LabRat Library.

void init()

Initialize General Robot Functionality

Basic I/O initialization function. This should always be called once at the beginning of your program.

void initADC()

Initialize Analog to Digital Conversion Hardware

This only enables ADC in general. When getADC is called THEN the IO in changes functions.

uint16_t getADC(int ch)

Get Analog to Digital from Channel

Retrieves a 10-bit integer from an ADC pin. Pass a number 0 - 7 to select which ADC Input to read from.

void initOptSensor()

Initialize Optical Sensor

Initializes the background process for the optical sensor. Sets up the interrupt to take measurements. The interrupt updates the total_angle, total_x and total_y variables, which are the accumulated deltas from the optical sensor, and raw_x and raw_y, which are the most recent deltas. This is not the robot pose, but pose can be calculated from it.

void OptSensorWrite(uint8_t addr, uint8_t data)

Write to Optical Sensor

Write to Optical Sensor Register. Do not use this unless you disable measurement interrupts.

uint8_t OptSensorRead(uint8_t addr)

Read From Optical Sensor

Read from Optical Sensor Register. Do not use this unless you disable measurement interrupts.

void optSensorWriteByte(uint8_t data)

Send a byte to the optical sensor.

Do not use this unless measurement interrupts are disabled.

motionData12b get12bXYMotion(void)

Get 12-bit XY motion data from optical sensor.

Currently under development, use get8bXYMotion instead.

motionData8b get8bXYMotion(void)

Get 8-bit motion data from optical sensor.

Returns structure containing delta X and delta Y from the optical sensor.

void optSensorToggleClk(void)

Toggle Optical Sensor Clock for data transactions.

Used by OptSensorRead/Write functions. Do not use unless measurement interrupts are disabled.

void initClock()

Initialize Clock

Deprecated, Do not use.

int getTimeElapsed()

Get Time Elapsed

Deprecated, do not use.

Get the number of ticks elapsed since last call of initClock() or getTimeElapsed(). Each tick of the clock is 0.000124 seconds. getTimeElapsed() must be called at most every 67 seconds, or the clock will overflow and the returned value will be invalid.

void initPWM(int val, ...)

Initialize PWM Channels

Sets which PWM outputs will be active (overrides other pin functions). Use constants A0 - B2 to refer to pins. To initialize OC0A pin, call: initPWM(A0);

Multiple arguments acceptable such as: initPWM(A0,B0,B2); Be careful using channels A0, B0, A2, B2, as these are connected to the on-board motor control H-bridges. Initiating PWM on channel pairs A0,B0 and A2, B2 simultaneously can damage the H-bridge. Use initMotorPwm() and setMotorSpeed(...) to safely control these channels.

void setLevel(int num, int level)

Set PWM Level

Sets a PWM level 0-255 representing 0-100% duty cycle. PWM must be initialized first using initPWM. To set OC0A (PB3) to 50% call: setLevel(A0, 127);

void initUART1(int UART1Speed, int baudrate)

Initialize UART 1 (SV1)

Initialize UART1 with 8bit frame - 1 stop bit, 9600 BAUD, no-parity

unsigned char getChar1()

Get Character from UART1

Gets a single character from UART1.

*void printAll1(const unsigned char *c, double var, ...)*

Print String to UART1

This is a custom printf-like function

- %d = +/- int to 65536
- %c = character code
- %f = +/- float to 9999.9999
- If you are only printing a string, put '0' in for second argument.
- ADD '\0' TO END OF STRING TO PREVENT ISSUES

void putChar1(char c)

Print Character to UART1

Write a character to UART1. You can use a while loop and an array of chars to write strings.

void putInt1(int num)

Print Integer (in ASCII) to UART1

Write an integer (0-65536) to UART1 (converts to ASCII equivalent).

void putFloat1(double dbl)

Print Floats (in ASCII) to UART1

Write float to UART1.

void initXBee(int XBeeSpeed, int baudrate)

Initialize XBee UART (UART0)

Initialize XBee (UART0) with 8bit frame - 1 stop bit, 9600 BAUD, no-parity.

unsigned char getCharXBee()

Get Character from XBee

Gets a single character from XBee (UART0)

*void printAllXBee(const unsigned char *c, double var, ...)*

Print String to XBee

Custom printf-like function

****XBee Version**** (UART0)

- %d = +/- int to 65536
- %c = character code
- %f = +/- float to 9999.9999
- If only printing a string, put '0' in for second argument.
- ADD '\0' TO END OF STRING TO PREVENT ISSUES

void putCharXBee(char c)

Print Character to XBee

Write a character to XBee (UART0). You can use a while loop and an array of chars to write strings.

void putIntXBee(int num)

Print Integer (in ASCII) to XBee

Write an integer (0-65536) to XBee (UART0) (converts to ASCII equivalent).

void putFloatXBee(double dbl)

Print Float (in ASCII) to XBee

Write float to XBee (UART0)

void SetLED(uint8_t state)

Set User LED State

Turn the user LED on and off.

void SetMotor(uint8_t motnum, uint8_t dir)

Set Motor(s) Direction - Full on/off

Set motor(s) binary value direction (full on/off).

void InitMotorPWM()

Initialize Motor PWM Channels

Initialize this before using setMotorSpeed.

void SetMotorSpeed(uint8_t motnum, uint8_t dir, uint8_t level)

Set Motor Speed Using PWM

Set the duty cycle and direction of a motor.

uint8_t getBump(uint8_t bumper)

Get status of left or right bumper

Returns 1 if passed bumper makes contact, 0 otherwise.

uint8_t getUserButton(void)

Get Status of User Button

Returns 1 if user button is pressed.

void initIRComm(void)

Initialize Infrared Communications Channel

Sets up a recurring interrupt to drive the center infrared LED as a serial communications channel.

void setIR(unsigned int IR_out, int level)

Set IR LED on or off.

void txData(void)

Transmit IR Serial Data worker function.

Function called by interrupt, do not use. When transmit in progress, global variable sendBusy is set to 1. A new message should not be initiated until sendBusy is clear.

void rcvData(void)

Receive IR Serial Data worker function.

Function called by interrupt, do not use. When a new message is received completely, the global variable newMsg is set to 1.

unsigned char irReadByte(void)

Read Byte from IR Communication Channel.

If a message is ready, function returns a byte of message data. Clears message buffer on call.

void irWriteByte(unsigned char data)

Write Byte to IR Communication Channel.

Initiates a new 1-byte IR serial transmission.

void initIRBumpers(void)

Initializes Left and Right Infrared bumpers

unsigned int obstacleDetect(unsigned int detector)

Gets state of Infrared Bumpers.

Returns 1 if obstacle is detected by infrared bumpers, 0 otherwise.