PushButton Library - Documentation

Library Description:

The PushButton library can be used to detect a button press on a button connected to pin RP15 on the PIC24FJ64GA002. Pin RP15 should be connected to a push button switch which completes a circuit to ground when pressed. An internal pull up resistor is connected to pin RP15 when initialized. The button should be connected to ground and complete the circuit when pressed. A low pass filter must be connected to pin RP15 to filter out high frequency noise caused by a switch bounce. This library uses the change notification interrupt on the PIC24. Then, call the isButtonPressed() function when wanting to determine whether the button was pressed.

Hardware Description:

Microcontroller: PIC24FJ64GA002 | Microchip Technology
Any push button switch may be used as a switch, as long as it completes the circuit when pressed. Push button on Amazon

Full Documentation:

void initPushButton()

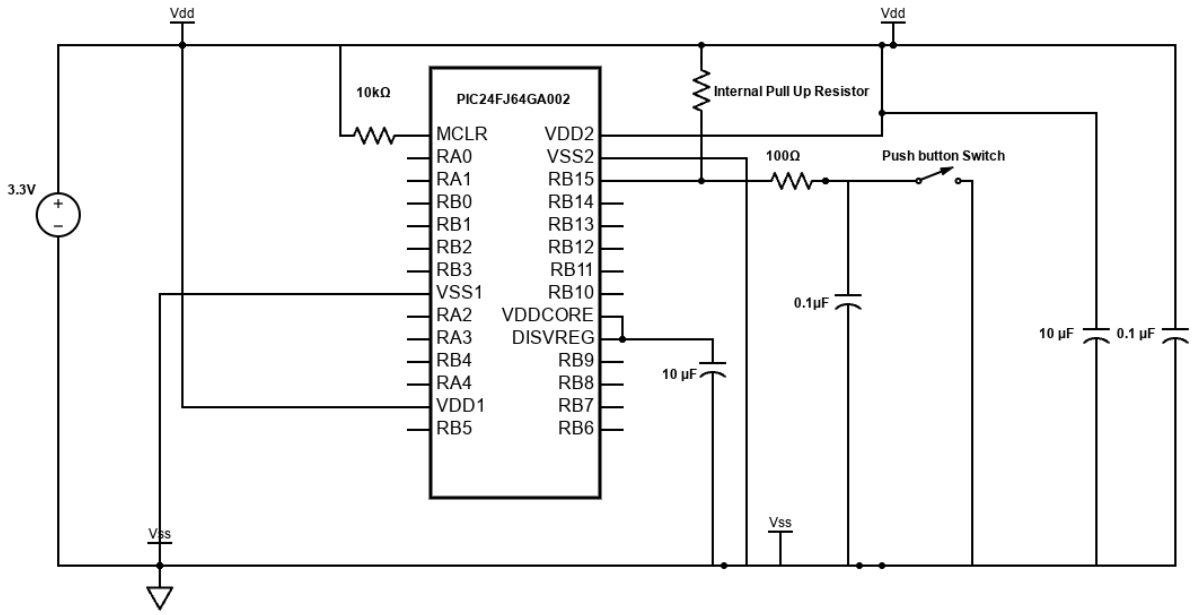
- No arguments, no output
- Description: Initialize pin RP15 in detecting button presses. The function will initialize the change notification interrupt for the pin and the internal pull-up resistor for it.

int isButtonPressed()

- No arguments
- Outputs an integer. The function will return 1 if a button was pressed, otherwise it will return 0. Note that after returning that a button was pressed, the function will discard that button press and return 0 until the next press

Basic Usage Example:

PIC24FJ64GA002 Circuit Schematic:



Ensure that a low pass filter is connected between the push button switch and pin RP15 (pin RP15 is also RB15), as the library contains no software debouncing. In the schematic, an RC circuit with a 100Ω and 0.1µF capacitor was used to construct a low pass filter, which corresponds to a corner frequency of 0.1MHz, filtering out changes in voltages that are <0.01ms long. This was sufficient during a test case to filter out bouncing, however bouncing may differ based on the technical specifications of the push button being used. It is recommended that the bouncing time is verified via oscilloscope when determining the adequate corner frequency for debouncing.

In the main code, include the “xc.h” library. Additionally, ensure to include the necessary flash configuration words into the main file (see Section 24.1 of the PIC24 Family Reference Manual for details). #include “PushButton.h” to use the PushButton functions in the main code.

Main Code Example:

```
// CW1: FLASH CONFIGURATION WORD 1 (see PIC24 Family Reference Manual 24.1)
#pragma config ICS = PGx1          // Comm Channel Select (Emulator EMUC1/EMUD1 pins are shared with PGC1/PGD1)
#pragma config FWDTEN = OFF        // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config GWRP = OFF          // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF           // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF        // JTAG Port Enable (JTAG port is disabled)


// CW2: FLASH CONFIGURATION WORD 2 (see PIC24 Family Reference Manual 24.1)
#pragma config I2C1SEL = PRI       // I2C1 Pin Location Select (Use default SCL1/SDA1 pins)
#pragma config IOL1WAY = OFF       // IOLOCK Protection (IOLOCK may be changed via unlocking seq)
#pragma config OSCIOFNC = ON       // Primary Oscillator I/O Function (CLKO/RC15 functions as I/O pin)
#pragma config FCKSM = CSECME      // Clock Switching and Monitor (Clock switching is enabled,
                                   // Fail-Safe Clock Monitor is enabled)
#pragma config FNOSC = FRCPLL      // Oscillator Select (Fast RC Oscillator with PLL module (FRCPLL))
```

```
#include "xc.h"
#include "PushButton.h"

int main(void) {
        initPushButton();
        Int wasButtonPressed = isButtonPressed(); // Determine if a button was pressed.
// Variable wasButtonPressed is 1 if a button was pressed, otherwise the variable is
zero.
        return 0;
}
```

Advanced Usage Example:

The PushButton library is useful at enabling button presses to activate other actuators on the PIC24FJ64GA002 to operate. For example, the below code shows how a push button can be used to enable or disable a device. The example code uses a separate Neopixel library to blink a Neopixel LED red or green to indicate whether the device was turned on or off.

```
// CW1: FLASH CONFIGURATION WORD 1 (see PIC24 Family Reference Manual 24.1)
#pragma config ICS = PGx1          // Comm Channel Select (Emulator EMUC1/EMUD1 pins are shared with PGC1/PGD1)
#pragma config FWDTEN = OFF        // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config GWRP = OFF          // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF           // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF        // JTAG Port Enable (JTAG port is disabled)


// CW2: FLASH CONFIGURATION WORD 2 (see PIC24 Family Reference Manual 24.1)
#pragma config I2C1SEL = PRI       // I2C1 Pin Location Select (Use default SCL1/SDA1 pins)
#pragma config IOL1WAY = OFF       // IOLOCK Protection (IOLOCK may be changed via unlocking seq)
#pragma config OSCIOFNC = ON       // Primary Oscillator I/O Function (CLKO/RC15 functions as I/O pin)
#pragma config FCKSM = CSECME      // Clock Switching and Monitor (Clock switching is enabled,
                                   // Fail-Safe Clock Monitor is enabled)
#pragma config FNOSC = FRCPLL      // Oscillator Select (Fast RC Oscillator with PLL module (FRCPLL))
```

```
#include "xc.h"
#include "PushButton.h"
#include "Neopixel.h"

void setup();
void loop();

int main(void) {
    setup();
    loop();
}

void setup() {
```

```
    initPushButton();
    initNeopixel();
}

void loop() {
    while(1) {
        if(isButtonPressed()) {
            blinkGreen(); // Indicate device is turned on
            while(!isButtonPressed()) {
                // Steps for determining if backpack is stolen...
            }
            blinkRed(); // Indicate device is turned off
        }
    }
}
```