Alarm Library - Documentation

Library Description:

The Alarm library is used to control a piezoelectric buzzer which generates a continuous sound when connected to 3.3V on pin RP14 on the PIC24FJ64GA002. The library uses Timer2 and the output compare 1 register on the microcontroller. Ensure these modules are not being used elsewhere. When the alarm is turned on, the library will send pulses to the buzzer with an input frequency greater than 0.477 Hz.

Hardware Description:

Microcontroller: PIC24FJ64GA002 | Microchip Technology
Piezoelectric buzzer: Alarm

Full Documentation:

void initAlarm()

- Argument is frequency at which to buzz alarm (must be greater than 0.477), no output
- Description: Initializes pin RP14 as output for the piezoelectric buzzer, as well as the output compare register and Timer2 used to send pulses to the buzzer.

void turnOnAlarm()

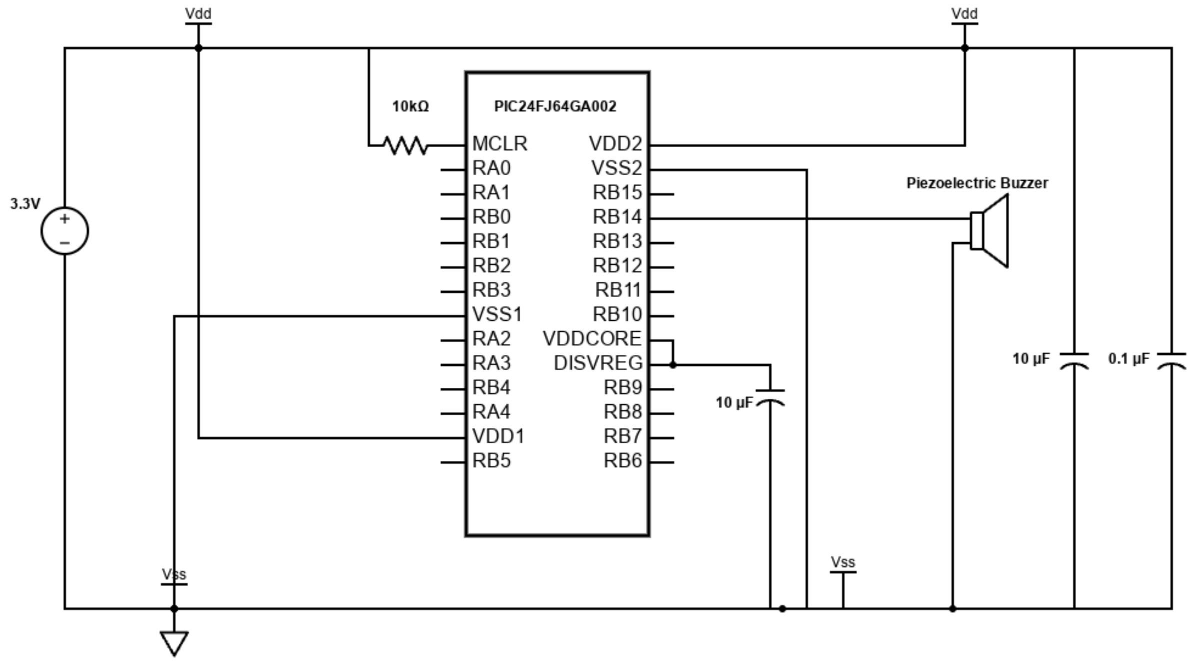
- No arguments, no output
- Description: Toggles alarm on with desired frequency.

void turnOffAlarm()

- No arguments, no output
- Description: Toggles alarm off.

Basic Usage Example:

PIC24FJ64GA002 Circuit Schematic:



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 "Alarm.h"

int main(void) {
        initAlarm();
        turnOnAlarm(); // turns alarm on
while(1)
{
}
        return 0;
}
```

Advanced Usage Example:

The alarm is useful as an indicator of another sensor hitting a desired threshold. In this example, if a button is pressed, the alarm will turn on. It will continue to ring until the button is pressed again disabling the alarm.

```
// 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 "Alarm.h"

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

void setup() {
   initAlarm(10); // 10 Hz frequency
   initPushButton();

}

void loop() {
   while(1) {
      if(isButtonPressed()) {
         turnOnAlarm();
         while(!isButtonPressed()); //waits until button is pressed again
        turnOffAlarm(); //button was pressed breaking loop
```

```
		}
}
return 0;
}
```