```
#include "mbed.h"
#include "IOMacros.h"

Ticker readout_timer;  // recurring interrupt to readout sum to serial
Timer t_low;
float currLow;

//Function prototype/forward declaration for ISR.
void startL();
void stopL();
float readlow();
void readout();
void rohandle();

bool rotriggered = false;

//EINT3_IRQHandler
//extern "C" void EINT3_IRQHandler __irq (void) {
extern "C" void EINT3_IRQHandler() {
    // The "event" is connected to pin p10 which is LPC1768 P0_1
    // so lets trap that and ignore all other GPIO interrupts.
    // Test for IRQ on Port0.
    if (LPC_GPIOINT->IntStatus & 0x1) {
        // If P0_1/p10 falls, call startL()
        if (LPC_GPIOINT->IO0IntStatF & (1 << 1)) startL();
        if (LPC_GPIOINT->IO0IntStatR & (1 << 1)) stopL();
    }

    // Clear this and all other possible GPIO generated interrupts as they don't concern us.
    LPC_GPIOINT->IO2IntClr = (LPC_GPIOINT->IO2IntStatR | LPC_GPIOINT->IO2IntStatF);
    LPC_GPIOINT->IO0IntClr = (LPC_GPIOINT->IO0IntStatR | LPC_GPIOINT->IO0IntStatF);
}

void event_irq_init(void) {
    // Use macro to set p10 as an input.
    p10_AS_INPUT;
    // Enable P0_1/p10 for rising edge interrupt generation.
    LPC_GPIOINT->IO0IntEnR |= (1UL << 1);
    LPC_GPIOINT->IO0IntEnF |= (1UL << 1);
    // Enable the interrupt.
    NVIC_EnableIRQ(EINT3_IRQn);
}

void readout() {
    rotriggered = true;
}

void rohandle() {
    printf("%2.6f\n\r", readlow());
}

void startL() {
    t_low.start();
}

void stopL() {
    t_low.stop();
}

float readlow() {
    currLow = t_low.read();
    t_low.reset();
    return currLow;
}

int main() {
    event_irq_init();
```

```
    readout_timer.attach(&readout,5);
    while (1) {
        if (rotriggered) {
            rotriggered = false;
            rohandle();
        }
    }
}
```
