Interrupt Based Blinkies

Signal Generator

3.5V

\( f = 2 \text{Hz} \)

Configure P1.4

input, no resistor, interrupt enabled, rising edge interrupt

Configure P1.0 \( \rightarrow \) output pin

In the ISR toggle P1.0 output

LED will blink at the rate of 1 blink/second

<table>
<thead>
<tr>
<th>0.5 sec</th>
<th>1 sec</th>
<th>Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>toggle LED ON</td>
<td>toggle LED OFF</td>
<td>toggle LED ON</td>
</tr>
</tbody>
</table>
Configuring P1.4 for interrupts

x: Port Number
1, 2

y: Bit position in Register
0, 1, 2, 3, 4, 5, 6, 7

Example:

PxSEL.y means
yth bit of register PxSEL

Default values are in red

PxSEL.y

0 Digital Input/Output

1 Alternative Function

0 Input

Read PxIN.y For input

1 Input

Interrupt Enabled

0 Interrupt Disabled

0 Pull Down

1 Pull Up

PxDIR.y

0 Output

Write to PxOUT.y for output

1 Output

PxOUT.y

PxREN.y

0 Resistor Disabled

1 Resistor Enabled

PxIE.y

0 Interrupt on Rising Edge

1 Interrupt on Lowering Edge

PxIES.y

bic.b $\times$ BIT4, & P1SEL
bic.b $\times$ BIT4, & P1REN
bic.b $\times$ BIT4, & P1IE

bis.b $\times$ BIT4, & PDIR

* (default)
Configuring P1.0 (RED LED)

x: Port Number
1, 2

y: Bit position in Register
0, 1, 2, 3, 4, 5, 6, 7

Example:

PxSEL.y means
yth bit of register PxSEL

Default values are in red

PxSEL.y

0 Digital Input/Output

1 Alternative Function

PxDIR.y

0 Input

Read PxIN.y For input

PxIE.y

1 Output

Write to PxOUT.y for output

PxOUT.y

0 Interrupt Disabled

1 Interrupt Enabled

PxIES.y

0 Interrupt on Rising Edge

1 Interrupt on Lowering Edge

PxREN.y

0 Resistor Disabled

1 Resistor Enabled

PxlE.y

0 Pull Down

1 Pull Up

bic b & BITO, & PISEL

bis b & BITO, & PIDIR

* default
Interrupt Vectors

int. vector table

int. type address

Int. Flag

int0 → FFFE0

int1 → FFFE2

int2 → FFFE4

int3 → FFFE6

int4 → FFFE8

int5 → FFEA

更高的优先级

中断类型

非可屏蔽

Bottom of memory addressable space

NMI

Osc. Fault

Flash access violation

RESET

single sourced interrupt

multisourced interrupt

Note:

For single sourced interrupts the interrupt flag is cleared automatically after the interrupt is serviced by the ISR.

For multiple sourced interrupts your ISR must clear the interrupt flag.
Program: InterruptBadBlinky

bis.b  #BIT0,  &P1OUT ; Set P1.0 as output pin
    &P1DIR ; P1.0 connected to the red LED
bis.b  #BIT4,  &P1IE ; Enable P1.4 interrupts
bis.w  #GIE,  SR ; Enable General Interrupts

loop:   jmp    loop ; loop and wait for interrupts
        ; bad Blinky bad!
        ; CPU should be sleeping

;------------------------------------  ISR  ------------------------------------
PORT1_ISR:
xor.b  #BIT0,  &P1OUT ; Toggle P1.0
bic.b  #BIT4,  &P1IFG ; multi-sourced, therefore clear IF
reti

;------------------------------------  Interrupt Vectors  ------------------------------------
;------------------------------------  Short  ------------------------------------
.sect  ".int02"
.short  PORT1_ISR
Low Power Modes

CPU can go to sleep (in a low power mode) and can be woken up by an interrupt.

There are five Low Power Modes (LPMs). We will look at only two.

Active Mode: Fully Awake (CPU, all clocks and enable modes active)
Current consumed ≈ 300 μA

LPM3: CPU, MCLK, SMCLK and DCO are disabled, only ACLK remains active.
Current consumed ≈ 0.5 μA

Status Register (SR)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>V</td>
<td>SCG1</td>
<td>SCG0</td>
<td>OSC OFF</td>
<td>CPU OFF</td>
<td>GIE</td>
<td>N</td>
<td>Z</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These 4 bits control the power modes
Status Register (SR)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>V</td>
<td>SCG1</td>
<td>SCG0</td>
<td>OSC OFF</td>
<td>CPU OFF</td>
<td>GIE</td>
<td>N</td>
<td>Z</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 0 0 0 .
Active Mode

Header file defines:

```c
#define LPM0 (CPUOFF)
#define LPM1 (SCG0+CPUOFF)
#define LPM2 (SCG1+CPUOFF)
#define LPM3 (SCG1+SCG0+CPUOFF)
#define LPM4 (SCG1+SCG0+OSCOFF+CPUOFF)
```

Status Register (SR)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>V</td>
<td>SCG1</td>
<td>SCG0</td>
<td>OSC OFF</td>
<td>CPU OFF</td>
<td>GIE</td>
<td>N</td>
<td>Z</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 1 0 1 .
LPM3 Mode

Set MCU in Low Power Mode 3:

`BIS.W @LPM3, SR`

Enable general interrupts & LPM3

`BIS.W @GIE|LPM3, SR`
Program: InterruptGoodBlinky

; Set P1.0 as output pin
; P1.0 connected to the red LED

; Enable P1.4 interrupts

; sleep and wait for interrupts
; Good Blinky!

; not required. Good for debugging

ISR

PORT1_ISR:

; Toggle P1.0
; multi-sourced, therefore clear IF

reti

;

Interrupt Vectors

sect ".int02"
.short PORT1_ISR
Stack picture

mov.w $GIE\|LPM3$, SR

not required. Good for debugging (will never be executed)

SP 0x03FC

SR 0x0000

cleaned CPU Active

myISR:

mov.w R11, R10
reti

PC

myISR

RAM

0x0200 0x0202
0x03F8 0x03FA
0x03FC 0x03FE
0x0400

Address of SR(...1101...)

Stack contains two words

picture of the stack just before the ISR starts execution

GIE 0xDDDD

0xCCCC
When reti is executed

* The SP pops from the stack. All previous settings of GIE and Low power mode control bits are in effect again, i.e., maskable interrupts are enabled again, and the MCU goes back to sleep again.
How do we accomplish the following?

```
Sleep:  bis.W × GIE | LPM3, SR
```

These never get executed

```
myISR:
```

```
LPM3
```

```
SR[...1101...]
```

```
SR[...1101...]
```

```
GIE
```

```
SR[...0000...]
```

```
GIE
```

```
do other stuff before going to sleep again
```

```
jmp sleep
```

```
reti
```

```
RAM
```

```
0xDDDD
```

```
0xCCCC
```

```
0(SR) / SR[...1101...]
```

```
Address of
```

```
Stack contains two words
```

```
mcu awake
```
Here is how!

\[ \text{Sleep: } \text{bis.W} \times \text{GIE|LPM3, SR} \]

\begin{align*}
\text{GIE} \quad \text{SR}[\ldots1101\ldots] \\
\text{mCU awake} \\
\end{align*}

\[ \text{do other stuff before going to sleep again} \]

\[ \text{myISR: } \text{bic.W} \times \text{LPM3, 0(SR)} \]

\[ \text{reti} \]

Clear the power mode bits on the SR saved on the stack

\[ \text{Stack contains two words} \]

\[ \text{Address of} \]

\[ O(SR) \]

\[ O \times \text{DDDD} \]

\[ O \times \text{CCCCC} \]