

Microcontroller

Freescal HCS08 Sample Code

Folgende Code-Fragmente sind im Zusammenhang mit dem Modul "Microcontroller" der Hochschule Luzern entstanden. Der Code wurde in der Entwicklungsumgebung Freescal CodeWarrior erstellt und kompiliert. Das Demo-Board M68DEMO908GB60 mit dem HCS08 Microcontroller diene als Testplattform.

1. Flashing LED using a Polling Approach

```
//main.c
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */

void initClock(void)
{
    SOPT = 0x73;          // Watchdog=disabled, Stop-Mode=enabled

    ICGC1 = 0x28;        // ICG-Mode: FEI
    ICGC2 = 0x72;        // N=10, R04
    while(!ICGS1_LOCK);  // Warten bis Clock stabil (locked) ist.
}

void wait(long cycles){
    long i=0;
    for(i=0; i<cycles; i++){
        //busy wait
    }
}

void main(void) {
    long i = 0; // datatype long takes much more busy than integer
               // time in the for loops below

    initClock();
    EnableInterrupts; /* enable interrupts */

    PTADD = 0;          //initialize as input (Data Direction Register)
    PTAPE = 0xf0; //Pullups on upper 4 bits

    /*initialize bits 0-3 of Port F as outputs (connected to led's)*/
    PTFDD = 0x0f;

    LED1 = OFF;
    LED2 = OFF;
    LED3 = OFF;
    LED4 = OFF;
    LED5 = OFF;

    for(;;) {
        LED1 = ON;
        wait(2000);
        LED1 = OFF;
        LED2 = ON;
        wait(2000);
        LED2 = OFF;
        LED3 = ON;
        wait(2000);
        LED3 = OFF;
        LED4 = ON;
        wait(2000);
        LED4 = OFF;
        LED5 = ON;
        wait(2000);
    }
}
```

```

        LED5 = OFF;

        __RESET_WATCHDOG(); /* feeds the dog */
    }
    /* loop forever */
    /* make sure that you never leave main */
}

```

mc9s08gb60a.h

```

/*define toggle values for LED's*/
#define ON 0
#define OFF 1

/*define LED's*/
#define LED1 PTFD_PTFD0
#define LED2 PTFD_PTFD1
#define LED3 PTFD_PTFD2
#define LED4 PTFD_PTFD3
#define LED5 PTFD_PTFD4

```

2. Flashing LED using Interrupts

//main.c

```

#include <hidef.h>          // for EnableInterrupts macro
#include "derivative.h"    // include peripheral declarations

void initClock(void)
{
    SOPT = 0x73;           // Watchdog=disabled, Stop-Mode=enabled

    ICGC1 = 0x78;         // ICG-Mode: FEI
    ICGC2 = 0x02;         // N=10, R04
    while(!ICGS1_LOCK);   // Warten bis Clock stabil (locked) ist.
}

void initPorts(void)
{
    PTFDD = 0xff;         // Port F = Output
    PTADD = 0x00;         // Port A = Input
    PTAPE = 0xff;         // Port A = Pull-Up enable
}

static unsigned int uiCounter = 0;

// key word interrupt is used to declare, that we want to jump back to main
// program after interrupt code execution
interrupt void ISR_RTI(void)
{
    int i = 0;
    // call the Interrupt Service Routine
    // for 300 times before switching the LEDs
    uiCounter++;
    if(uiCounter>=300) {
        uiCounter = 0;
        PTFD = ~PTFD;
    }

    SRTISC_RTIACK = 1; // acknowledge interrupt
                       // if you do not acknowledge an interrupt,
                       // your application turns in an endless loop
                       // and never gets back to the main program
}

void initRTI(void) {
    SRTISC_RTIS = 7; // Sets the counter // RTIS2=1, RTIS1=1, RTIS0=1 (dezimal=7)
    SRTISC_RTICLKS = 1; // External Clock
    SRTISC_RTIE = 1; // RTI Interrupt Enable
}

```

```
/**
 * main program
 */
void main(void)
{
    initClock();           // Clock und Watchdog konfigurieren

    initPorts();          // Ports konfigurieren

    EnableInterrupts;     // Interrupts aktivieren

    initRTI();

    for(;;)
    {
        // nothing to do here
        // we're just waiting for interrupts
    }
}
```