

```
/* Simple Interrupt program for AT91RM9200 ARM9
```

This program is meant to show you how to use an interrupt on a n ATMEL AT91RM9200 ARM9 with ICE.

The program prints a variable (mytestvalue) to the debug channel every second, then every fifth second it will increment the variable (mytestvalue).

The interrupt used is the external interrupt PA26 IRQ1. The AT91RM9200 has a feature to set/clear interrupts in software. (So we can set interrup IRQ1 wihout the use of external signals). See AIC\_ISCR in the datasheet.

Thanks to Patrick Filippi AT91 Support Group and Dimitri Princen

Written by Barry de Graaff (www.barrydegraaff.tk)

```
*/
```

```
#include "AT91RM9200.h"
```

```
// Declare subs and vars
```

```
// Print dimi converts a variable to ASCII
```

```
unsigned char * Print_dimi(AT91_REG *arg);
```

```
unsigned int mytestvalue;
```

```
// AT91F_DBGU_Printk is defined in init.c
```

```
extern void AT91F_DBGU_Printk(char *);
```

```
//This defines the interrupt
```

```
void ext_IRQ0_handler(void);
```

```
// AT91F_InitFlash
```

```
// Flash init, this mat differ for you!!
```

```
void AT91F_InitFlash()
```

```
{  
    /* Setup MEMC to support CS0=Flash
```

```
    AT91C_BASE_EBI->EBI_CSA |= AT91C_EBI_CS0A_SMC;
```

```
    AT91C_BASE_EBI->EBI_CFGR = (AT91C_EBI_DBPUC & 0x00) | (AT91C_EBI_EBSEN & 0x00);
```

```
    /* Setup Flash
```

```
    AT91C_BASE_SMC2->SMC2_CSR[0] = (AT91C_SMC2_NWS & 0x4) | AT91C_SMC2_WSEN |
```

```
    (AT91C_SMC2_TDF & 0x200) | AT91C_SMC2_BAT |
```

```
    AT91C_SMC2_DBW_16;
```

```
}
```

```
// Interrupt handler
void ext_IRQ0_handler(void)
{
    AT91C_BASE_AIC->AIC_IDCR = 0xFFFFFFFF; // Interrupt Disable command register
    AT91C_BASE_AIC->AIC_ICCR = 0x04000000; // This should clear PID26=IRQ1??

    mytestvalue++; // Increment the variable

    // This clears the I bit in the CPSR re-enabling the Interrupts
    asm("mrs r7,CPSR"); // MRS is status reg to reg
    asm("bic r7,r7,#0x80"); // BIC instruction = bit clear
    asm("msr CPSR,r7"); // MSR is reg to status reg

    AT91C_BASE_AIC->AIC_IECR = 0x04000000; // Interrupt enable command register
    // Writing anything to IVR will unstack and clear in debug mode
    AT91C_BASE_AIC->AIC_IVR = 0x0;
    // Here acknowledge the end of interrupt
    AT91C_BASE_AIC->AIC_EOICR = 0x0;
}

int Main()
{
    // Test interrupts can only be triggered in edge triggered mode
    //AT91C_BASE_AIC->AIC_SMR[0x1A]= 0x00000007; // Assign highest priority to IRQ1 and Low Level Sensitive

    // Assign highest priority to IRQ1 and Edge Triggered (Edge trigger is needed for test with AIC_ISCR)
    AT91C_BASE_AIC->AIC_SMR[0x1A]= 0x00000067;
    AT91C_BASE_AIC->AIC_SVR[0x1A]= (unsigned int)&ext_IRQ0_handler; // Assign the location of the int handler to the vector reg
    AT91C_BASE_AIC->AIC_IMR = 0x04000000; // Set mask for IRQ1 PA26
    AT91C_BASE_AIC->AIC_IECR = 0x04000000; // Enable IRQ1 PA26

    //AT91C_BASE_AIC->AIC_DCR = 0x00000001; // Debug control reg

    mytestvalue = 0x19;

    unsigned int tick; // We use tick as a timer
    tick = *(AT91C_ST_CRTR);
    unsigned int timeout;
```

```
while (1) {
    // Print the testvalue every second
    AT91F_DBGU_Printk(Print_dimi(&mytestvalue));

    timeout++;
    if(timeout>4){           //Generate a test-interrupt every 5 seconds
        AT91F_DBGU_Printk("\r\n");

        // Test interrupt can only be used if int is EDGE TRIGGERED
        // Every 5th second
        AT91C_BASE_AIC->AIC_ISCR = 0x04000000; // This should set PA26=IRQ1
        timeout=0;
    }

    while (tick == *(AT91C_ST_CRTR));
    tick = *(AT91C_ST_CRTR);
}

// Unsigned INT Variable to ASCII (expects pointer)
unsigned char * Print_dimi(AT91_REG *arg)
{
    signed long i;
    unsigned long temp=*arg;
    unsigned char buff;
    unsigned static char tekst[9];

    for(i=7;i>=0;i--){
        tekst[i]=(temp&0x0f);
        if(tekst[i]>9)tekst[i]+=55;
        else tekst[i]+=48;
        temp>>=4;
    }
    tekst[8]=0;
    return tekst;
}
```