Computer Engineering
WS 2012

Interruptverarbeitung im LPC 2468

HTM – SHF - SWR

Ausnahme- und Interrupt-verarbeitung im LPC 2468

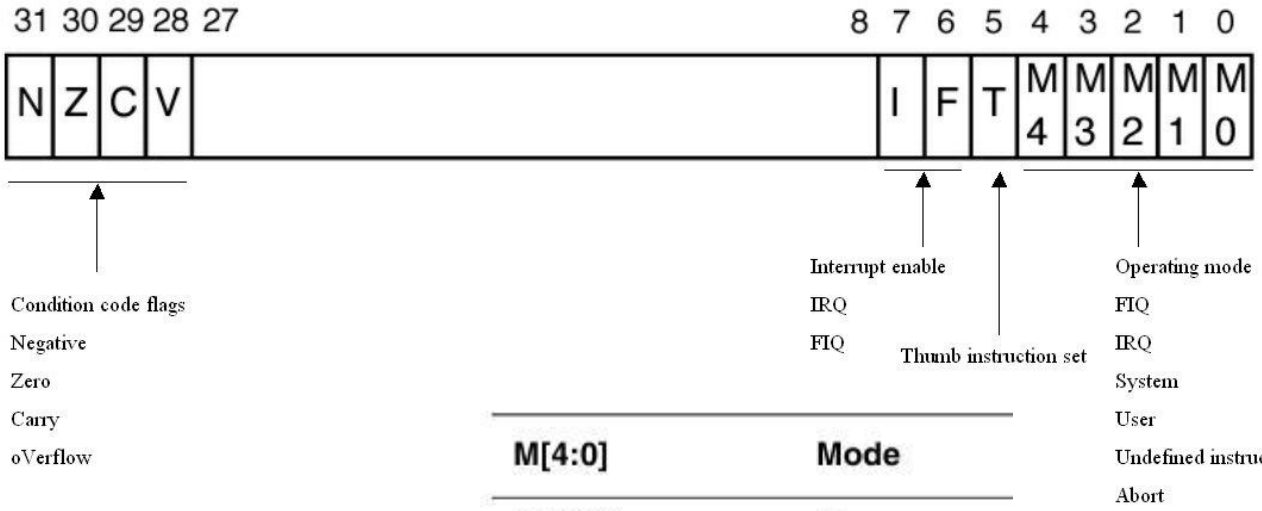
Betriebsarten, Statuswort

- ARM 7 TDMI unterscheidet 7 Betriebsarten.
- **User Mode** ist die normale Betriebsart.
- Wechsel in andere Betriebsarten durch **Exceptions** (**Ausnahmen**).
- Momentane Betriebsart steht im „Current Program Status Register“ (**CPSR**)
- Betriebsarten haben:
  - **eigene Stacks**
  - **eigene Register (z.B. LR und SP)** (siehe nächste Folie)
  - **„Saved Program Status Register“ (SPSR)**
    (beim Wechsel der Betriebsart wird hier das CPSR gespeichert)
  - **unterschiedliche Rechte bei Zugriff auf Systemregister**
    (z.B. im User Mode kann nicht der SP oder das SPSR verändert werden)

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Betriebsarten

Statuswort



M[4:0]	Mode
0b10000	User
0b10001	FIQ
0b10010	IRQ
0b10011	Supervisor
0b10111	Abort
0b11011	Undefined
0b11111	System

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Betriebsarten

Statuswort

System & User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

HITOP:

Register werden angezeigt mit

View→
SFR Window →
ARM Processor Register

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Betriebsarten, Statuswort

- **Wie kann man auf R8 der Betriebsart FIQ zugreifen?**

1. Momentane Betriebsart in R1 merken.
2. Auf Betriebsart FIQ umschalten
3. Registerinhalt von R8 nach R0 schreiben
4. Zurückschalten auf ursprüngliche Betriebsart

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Ausnahmen

- **Wenn eine Ausnahme auftritt:**
  - **CPU rettet CPSR nach SPSR und ändert Betriebsart**
  - **PC wird auf Eintrag in Vektortabelle gesetzt:**

Exception	Mode	Address
Reset	Supervisor	0x00000000
Undefined Instruction	Undefined	0x00000004
Software Interrupt (SWI)	Supervisor	0x00000008
Prefetch Abort (instruction fetch memory abort)	Abort	0x0000000C
Data Abort (data access memory abort)	Abort	0x00000010
IRQ (interrupt)	IRQ	0x00000018
FIQ (fast interrupt)	FIQ	0x0000001C

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Ausnahmen

- **Rückkehr von Ausnahme:**
  - **CPU hat dafür (im Gegensatz zu vielen anderen) keinen speziellen Befehl.**
  - **Daher müssen PC und CPSR mit regulären ARM-Assembler Befehlen geändert werden.**

Exception	Mode	Return Statement
Reset	Supervisor	-
Undefined Instruction	Undefined	SUBS PC,LR,#4
Software Interrupt (SWI)	Supervisor	MOVS PC,LR
Prefetch Abort (instruction fetch memory abort)	Abort	SUBS PC,LR,#4
Data Abort (data access memory abort)	Abort	SUBS PC,LR,#8
IRQ (interrupt)	IRQ	SUBS PC,LR,#4
FIQ (fast interrupt)	FIQ	SUBS PC,LR,#4

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Ausnahmeroutinen

- **HITOP Entwicklungsumgebung:**

  startup.s

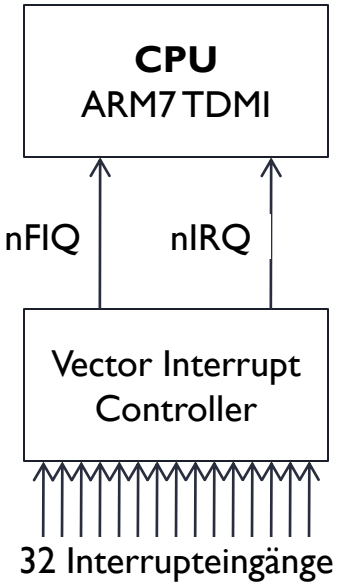
  - **definiert Standard-Ausnahmeroutinen, trägt die Aufrufe in die Vektortabelle ein und**
  - **richtet die Stacks für die Betriebsarten ein.**

Exception	Mode	Behandlung
Reset	Supervisor	Startup-Code, Aufruf von main
Undefined Instruction	Undefined	Endlosschleife
Software Interrupt (SWI)	Supervisor	Rahmen für eigene Routinen
Prefetch Abort (instruction fetch memory abort)	Abort	Endlosschleife
Data Abort (data access memory abort)	Abort	Endlosschleife
IRQ (interrupt)	IRQ	spezielle Behandlung
FIQ (fast interrupt)	FIQ	Endlosschleife

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Interruptverarbeitung

- **ARM-CPU:**
  - **2 Interruptrequest-Eingänge: Fast Interrupt (FIQ) und Interrupt (IRQ)**
- **LPC2468:**
  - **Zusätzlich „Vector Interrupt Controller“ (VIC)**
  - **Jede Interruptquelle des Chips ist fest mit einem bestimmten Eingang des VIC verbunden.**
  - **Per Software kann jede Interruptquelle auf FIQ oder vectored IRQ eingestellt werden.**
  - **Idealerweise sollte nur eine Interruptquelle auf FIQ gestellt werden.**
  - **Die restlichen Quellen benutzen vectored IRQ. Hierfür kann die Reihenfolge der Abarbeitung mittels 16 Prioritätsstufen festgelegt werden.**



Ausnahme- und Interrupt-verarbeitung im LPC 2468

Interruptquellen des LPC 2468

Bit	31	30	29	28	27	26	25	24
Symbol	I2S	I2C2	UART3	UART2	TIMER3	TIMER2	GPDMA	SD/MMC
Bit	23	22	21	20	19	18	17	16
Symbol	CAN1&2	USB	Ethernet	BOD	I2C1	AD0	EINT3	EINT2/ LCD(1)
Bit	15	14	13	12	11	10	9	8
Symbol	EINT1	EINT0	RTC	PLL	SSP1	SPI/SSP0	I2C0	PWM0&1
Bit	7	6	5	4	3	2	1	0
Symbol	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT

Ausnahme- und Interrupt-verarbeitung im LPC 2468

FIQ

- Aktionen zum Eintreten in die ISR
  1. Speichern der Adresse der nächsten Instruktion (PC) in LR
  2. Speichern von CPSR in SPSR
  3. Setzen der Betriebsart FIQ in CPSR, Registersatz wird umgeschaltet
  4. Setzen des F-Bits in CPSR (FIQ aus)
  5. Setzen des PC auf FIQ-Adresse der Vektortabelle

- Maximale Dauer bis Start ISR (Interruptlatenzzeit) setzt sich zusammen:
  1. Synchronisation des externen Signals — max 4 Takte
  2. Abarbeiten der aktuellen Instruktion:
     Längste Instruktion ist LDM mit allen Registern — max 20 Takte
  3. Möglicher Weise findet gerade eine Data Abort Exception statt (hat höhere Priorität): — 3 Takte
  4. FIQ-Aktivierung — 2 Takte

  Insgesamt: 29 Takte, bei 48 MHz also ca. 0.6 µsec

Ausnahme- und Interrupt-verarbeitung im LPC 2468

FIQ

- **Aktionen zum Verlassen der ISR**
  1. **Speichern des LR, verkleinert um 4, in den PC.**
  2. **Speichern von SPSR in das CPSR.**
     Damit automatisch:
     - **Umschalten der Betriebsart und**
     - **Reaktivierung des FIQs.**

- **Beide Aktionen lassen sich wie folgt durchführen:**

```
SUBS    PC,LR,#4        @ ISR verlassen
```

Basiert auf Sonderfunktion:

Wenn S bei Datenmanipulationsbefehlen gesetzt und das Zielregister PC ist,
dann wird SPSR nach CPSR kopiert!

Ausnahme- und Interrupt-verarbeitung im LPC 2468

FIQ-ISR: Eigenschaften

- **CPU:** schaltet Register R8 bis R14 um,
  können also von ISR beliebig verändert werden.

  rettet Statusregister und Rücksprungadresse.
- Änderungen an R0 bis R7 müssen am Ende der ISR durch die ISR rückgängig gemacht werden!
- Vor Aufruf eines Unterprogramms muss LR gerettet werden!
- Unterprogramme, die vom Hauptprogramm und von der ISR aufgerufen werden, müssen wiedereintrittsfest (**reentrant**) sein:

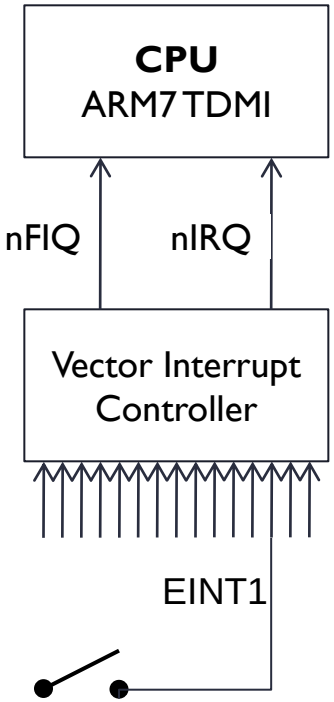
  Eine Funktion ist reentrant, wenn sie mehrmals gleichzeitig aktiv sein kann, ohne dass sich diese Aufrufe gegenseitig beeinflussen.

  - **Mehrfachaufrufe passieren häufig bei Bibliotheksfunktionen (strcpy).**
  - **printf und malloc sind meist nicht reentrant!**

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Beispiel: Auslösung eines FIQ durch eine Taste

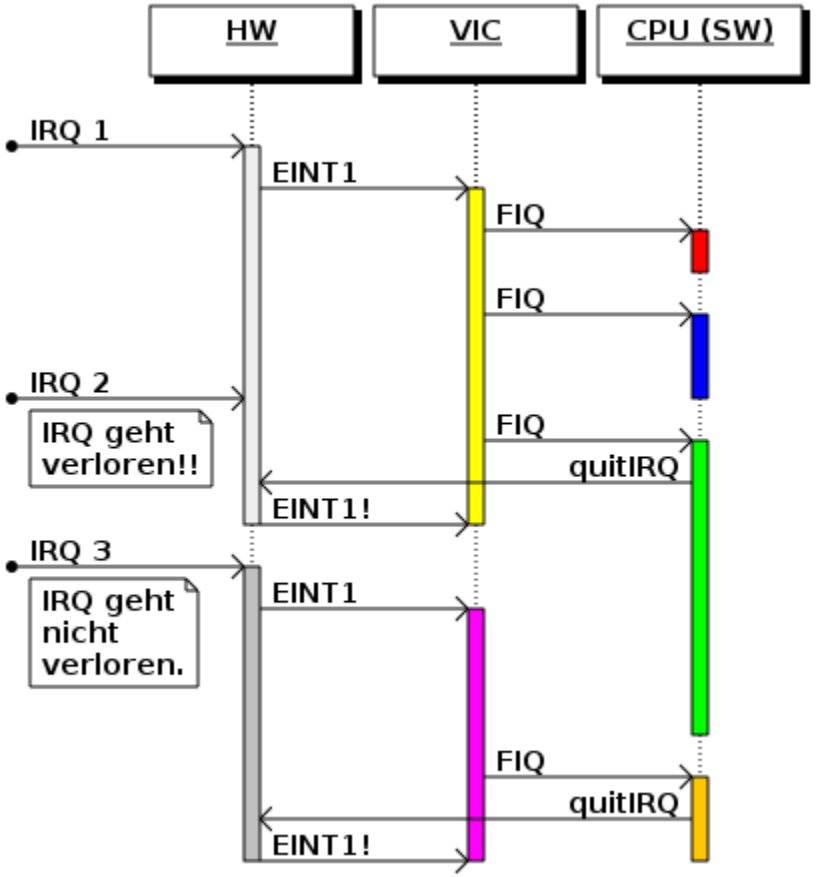
- FIQ-Serviceroutine : Zählen der Tastenbetätigungen.
- Verwendung des „Externen Interrupt 1“ (EINT1)
  - **Auslösung von EINT1 durch externes digitales Signal.**
  - **EINT1 kann eingestellt werden auf:**
    - **Pegelsensitiv, Auslösung bei High-Pegel**
    - **Pegelsensitiv, Auslösung bei Low-Pegel**
    - **Flankensensitiv, Auslösung bei steigender Flanke**
    - **Flankensensitiv, Auslösung bei fallender Flanke**



Ausnahme- und Interrupt-verarbeitung im LPC 2468

Beispiel „Zählen von externen Ereignissen“: Ablauf

- **Interrupts müssen in der Regel quittiert werden.**
- **Ohne Quittierung: Nach Beendigung der ISR wird sie sofort erneut aktiviert!**
  - **→ System ist dauerhaft blockiert**
- **Interrupts können verlorengehen**
- **daher: Quittierung des Interrupts möglichst früh durchführen!**



Ausnahme- und Interrupt-verarbeitung im LPC 2468

Beispiel „Zählen von externen Ereignissen“: ISR

1. **Interrupt muss quittiert werden**
   - **R10 enthält die Adresse des Ports EXTINT**
   - **R9 die notwendige Maske**

   Beides wurde während der Initialisierung in die FIQ-Register geladen
2. **Zähler erhöhen, R8 enthält aktuellen Zählerwert.**
3. **Interruptserviceroutine beenden.**

```
fiq:    STR     R10,[R9]        @ Interrupt quittieren
        ADD     R8,R8,#1        @ Zaehler erhoehen
        SUBS    PC,LR,#4        @ ISR verlassen
```

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Beispiel „Zählen von externen Ereignissen“: Initialisierung der ISR

```
        .section .text
        .global  fiq,fiqinitasm

        .equ EXTINT,0xE01FC140

fiqinitasm:                         @ Funktion kann nur in Betriebsart
                                    @ Supervisor aufgerufen werden!

        MRS     R0, CPSR            @ aktuelle Betriebsart retten
        MSR     CPSR_c,#0xD1        @ auf FIQ-Betriebsart umschalten
                                    @ FIQ und IRQ aus

        MOV     R8,#0               @ Zaehler loeschen
        LDR     R9,=EXTINT          @ Adr von EXTINT laden
        MOV     R10,#1<<1           @ Maske fuer EINT1 laden

        BIC     R0,#1<<6            @ FIQ aktivieren: FIQ-Bit im CPSR loeschen
        MSR     CPSR_c,R0           @ auf urspruengliche Betriebsart schalten

        MOV     PC,LR               @ Unterprogramm verlassen
```

Ausnahme- und Interrupt-verarbeitung im LPC 2468

C-Programm: Auslösung eines FIQ durch eine Taste

```
extern void fiqinitasm(void);

void fiqinit(void){

    PINSEL4       |= 1<<22;          //P2.11 is EINT1

    EXTMODE        = 1<<1;          //edge sensivity
    EXTPOLAR       = 0;             //Falling edge

    VICIntSelect |= 1<<15;          //EINT1 is FIQ
    VICIntEnable |= 1<<15;          //Enable EINT1

    fiqinitasm();                   //Restliche Initialisierung
                                    //mit Assemblerprogramm
}
```

Externer Interrupt

Vector Interrupt Controller

Ausnahme- und Interrupt-verarbeitung im LPC 2468

ISR in C programmiert

- Nicht im C-Standard vorgesehen!
- Implementierung abhängig von
  - **CPU-Hersteller**
  - **Compiler-Hersteller**
- Nachfolgend: Implementierung von ISR mittels GNU C-Compiler (gcc) für ARM-CPUs.
- Im gcc können Funktionen zusätzliche Attribute erhalten:

```
void f () __attribute__ ((interrupt ("FIQ")));
void f () __attribute__ ((interrupt ("IRQ")));
```

- Compiler rettet alle notwendigen Register und restauriert sie am Ende der ISR.
- **Return** am Ende der Funktion wird ersetzt durch „**Return from Interrupt**“

Link auf gcc-Dokumentation:
http://gcc.gnu.org/onlinedocs/gcc/Function-Attributes.html

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Beispiel „Zählen von externen Ereignissen“: C-Version der ISR

- Übersetzt mit –O2: Hohe Optimierung. Im Praktikum: -O0

C-Code

```
void __attribute__ ((interrupt("FIQ"))) cfiq(void) {

        counter++;




        EXTINT = 1<<1;




}
```

Erzeugter Assemblercode

```
stmfd   sp!, {r1, r2, r3}

ldr r2, .L3
ldr r3, [r2, #0]
add r3, r3, #1
str r3, [r2, #0]

mov r3, #-536870912
add r3, r3, #2080768
mov r1, #2
add r3, r3, #256
strb    r1, [r3, #64]

ldmfd   sp!, {r1, r2, r3}
subs    pc, lr, #4
```

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Beispiel „Zählen von externen Ereignissen“: Eintrag in Vektortabelle

- Hitop: Vektortabelle ist in start.s definiert

```
_startup:
# -------------------------------------------
# Interrupt vector table at address 0
# -------------------------------------------
Reset_Vec:      LDR     PC, _ResetEntry
                LDR     PC, _Undef_Addr
                LDR     PC, _SWI_Addr
                LDR     PC, _PAbt_Addr
                LDR     PC, _DAbt_Addr
                .word   CheckSum            /* Reserved Vector */
                LDR     PC, [PC,#-0x120]
                LDR     PC, _FIQ_Addr       /* Calling the FIQ handler */
# -------------------------------------------
_ResetEntry:     .word   ResetEntry
_Undef_Addr:     .word   Undef_Handler
_SWI_Addr:       .word   SWI_Handler
_PAbt_Addr:      .word   PAbt_Handler
_DAbt_Addr:      .word   DAbt_Handler
_FIQ_Addr:       .word   fiq
```

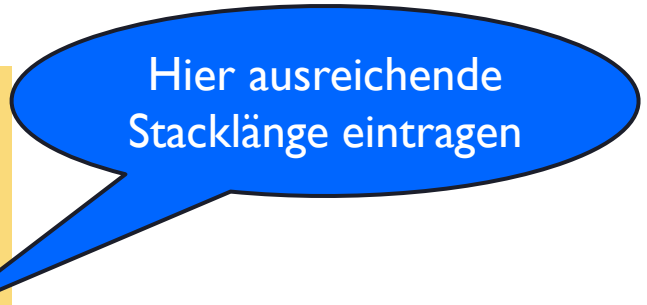
Hier Startadresse der ISR eintragen

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Beispiel „Zählen von externen Ereignissen“: Stack definieren

- **Hitop: Stack ist in start.s definiert**
- **Standardeinstellung für FIQ-Stacklänge: nur 8 Worte!**

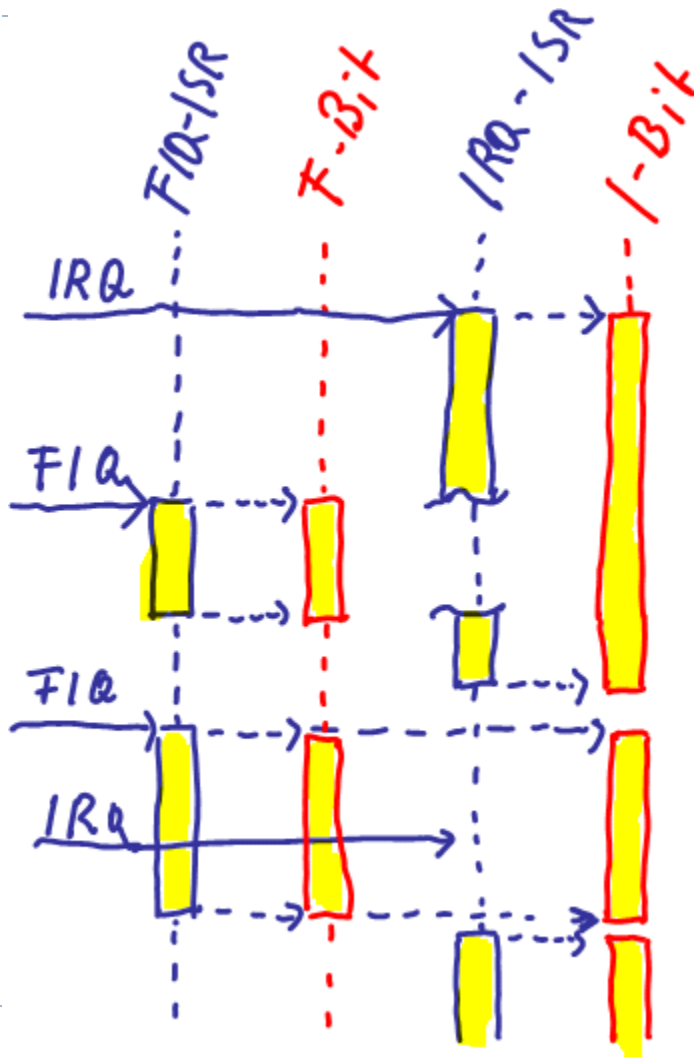
```
# Stack definitions
# size in words!
                .equ    UND_Stack_Size , 8
                .equ    SVC_Stack_Size , 1024
                .equ    ABT_Stack_Size , 8
                .equ    FIQ_Stack_Size , 8
                .equ    IRQ_Stack_Size , 512
```



Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ

- IRQ verhält sich innerhalb der CPU wie FIQ, hat aber eine geringere Priorität.
- Priorität entscheidet, welcher Interrupt als nächstes bearbeitet wird,
  wenn beide Interrupts gleichzeitig auftreten.
- Unterbrechbarkeit wird gesteuert durch Setzen und Löschen des I- und des F-Bits im Statusregister (siehe Ablaufdiagramm).
  - **Aktivieren von IRQ setzt nur I-Bit.**
  - **Aktivieren von FIQ setzt I- und F-Bit.**
- Damit kann FIQ eine laufende IRQ-ISR unterbrechen.
- Aber IRQ kann eine FIQ-ISR nicht unterbrechen!



Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Zuordnung der ISR

- VIC verwaltet für jeden Interrupteingang:
  - **Startadresse der ISR**
  - **Priorität des Interrupts**
- Aktivierung eines Interrupteingangs:
  - **VIC kopiert zugeordnete Startadresse in das Register VICVectAddr.**
  - **VIC setzt internes Prioritätsregister auf Priorität des Interrupts.**
  - **VIC aktiviert IRQ EIngang der CPU.**
- Ausnahmeverarbeitung der CPU liest **VICVectAddr** (Adr 0xFFFF FF00) und verzweigt direkt in die ISR.

Vektortabelle

```
0000        LDR     PC, _ResetEntry
0004        LDR     PC, _Undef_Addr
0008        LDR     PC, _SWI_Addr
000C        LDR     PC, _PAbt_Addr
0010        LDR     PC, _DAbt_Addr
0014        .word   CheckSum            /* Reserved Vector */
0018        LDR     PC, [PC,#-0x120]    /* Calling the IRQ handler */
001C        LDR     PC, _FIQ_Addr
```

Lesen von VicVectAddr und Sprung in die ISR

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Prioritäten

- Nach Auslösen des IRQs:
  - **Internes Register des VIC enthält Priorität des aktuellen Interrupts.**
- Nachfolgende Interrupts mit **kleinerer** oder **gleicher** Priorität:
  - **Werden von VIC nicht bearbeitet.**
- Nachfolgende Interrupts mit **höherer** Priorität:
  - **VIC aktualisiert sofort VICVectAddr und**
  - **aktiviert IRQ-Eingang der CPU**
- Ermöglicht die Implementierung von **unterbrechbaren (preemptiven)** ISRs!
- Erfordert allerdings zusätzlichen Programmieraufwand:
  - **Frühzeitige Freigabe des IRQs.**
  - **Zusätzliche Verwaltungsmaßnahmen.**

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Nicht unterbrechbare ISR

- Erfordert zwei wesentliche Aktivitäten:
  1. **Quittierung des Interrupts beim Interrupt auslösenden Gerät:**
     - **Gerät deaktiviert IRQ.**
  2. **Quittierung des Interrupts beim VIC:**
     - **VIC setzt internes Prioritätenregister zurück.**
       Sollte erst am Ende der ISR erfolgen.
- **IRQ ist während der gesamten Abarbeitung gesperrt (I-Bit gesetzt):**
  - **ISR kann nicht unterbrochen werden (bzw. nur durch FIQ).**

```
void __attribute__ ((interrupt("IRQ"))) little_isr(void) {

    EXTINT = 1<<1;                  /* Interrupt bestaetigen:          */
                                    /* Device deaktiviert IRQ-Leitung  */
    counter++;

    VICVectAddr = 0;                /* VIC mitteilen: ISR ist fertig   */
                                    /* VIC setzt internes Prio Register */
                                    /* zurueck                         */
}
```

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Initialisierung von nicht unterbrechbaren ISRs

```
void irqinit(void){
    PINSEL4            |= 1<<24;            // P2.12 is EINT2
    EXTMODE             = 1<<2;             // edge sensivity
    EXTPOLAR            = 0;                // Falling edge

    VICIntSelect       &= ~(1<<16);         // EINT2 is IRQ
    VICVectAddr16       = (int)little_isr;  // EINT2 fires little_isr
    VICVectPriority16   = 5;                // EINT2 gets Prio 5
    VICIntEnable       |= 1<<16;            // Enable EINT2
}
```

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Nichtunterbrechbare ISR, Latenzzeiten

- Es wurden 3 ISR implementiert:
  - **ISR1 hat die höchste Priorität und dauert 200 µs,**
  - **ISR2 hat eine mittlere Priorität und dauert 400 µs und**
  - **ISR3 hat die niedrigste Priorität und dauert 300 µs.**

  Jede Interruptquelle feuert maximal einmal pro 1 msec.
- Welche Interrupt-Latenzzeiten sind im schlimmsten Fall auf Grund der obigen Angaben für die ISRs mindestens zu erwarten?

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Preemptive ISR

- Erfordert zwei zusätzliche Aktivitäten:
  1. **Nach Quittierung des Interrupts:**
     - Löschen des I-Bits (Freigabe des IRQs).
     - Retten von LR und SPSR (nicht trivial)
  2. **Vor Quittierung des Interrupts beim VIC:**
     - Sperren des IRQs
     - Zurückspeichern von LR und SPSR

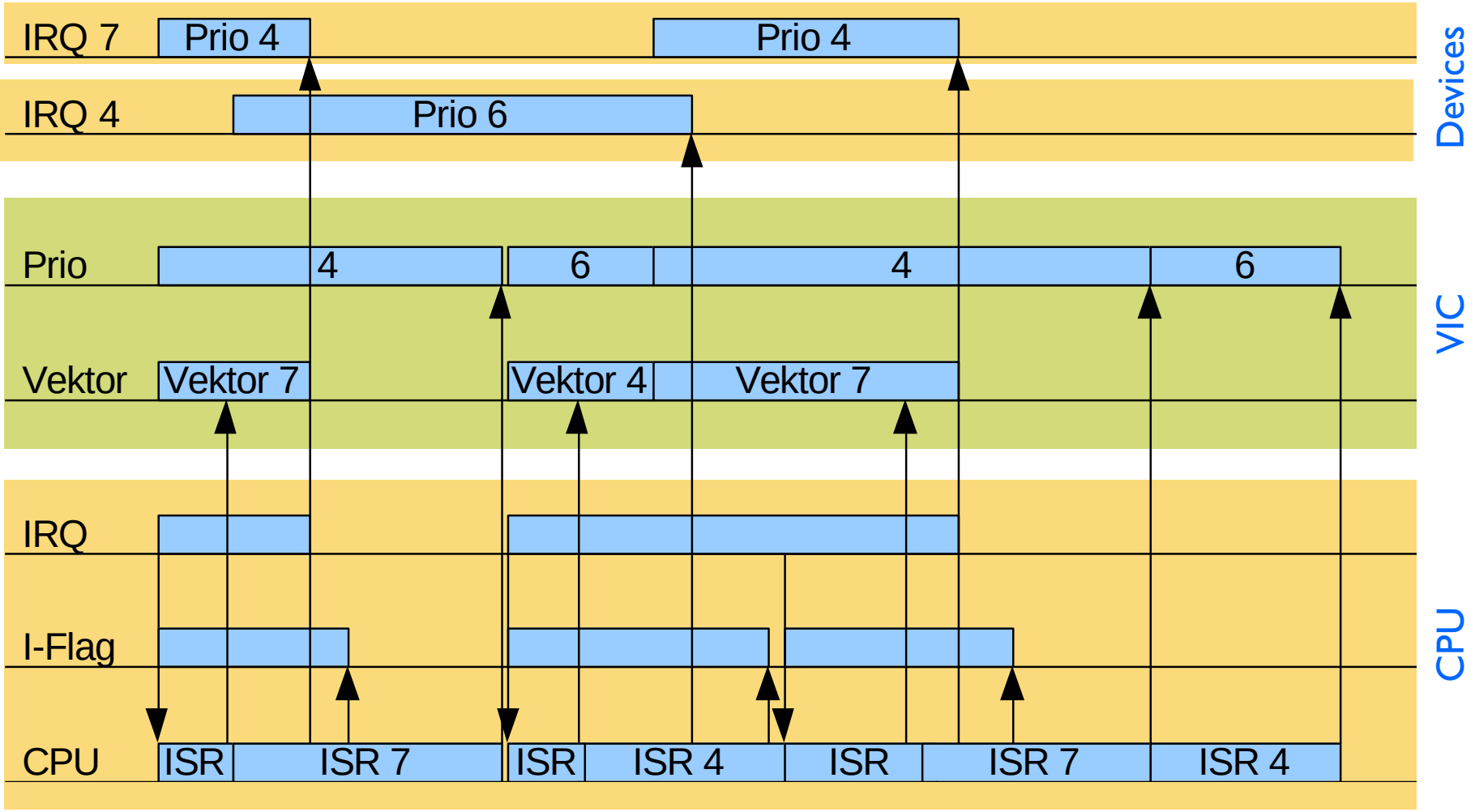
```
void __attribute__ ((interrupt("IRQ"))) little_isr(void) {

    EXTINT = 1<<1;                  /* Interrupt bestaetigen:          */
                                    /* Device deaktiviert IRQ-Leitung  */
    ENABLE_NESTED_ISR;
    .....
    .....
    .....
    DISABLE_NESTED_ISR;

    VICVectAddr = 0;                /* VIC mitteilen: ISR ist fertig   */
}
```

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Preemptive ISR



Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Unterbrechbare ISR

- Retten von LR funktioniert nur mit Umschaltung in eine andere Betriebsart!
- Ohne Umschaltung geht aktuelles LR beim erneuten Interrupt verloren.
- Im Beispiel: Rumpf der ISR wird in Supervisor-Mode abgearbeitet.
- Achtung: Verwendung des Supervisorstacks!

```
#define ENABLE_NESTED_ISR                                                    \
  asm volatile(" mrs   r1, spsr\n"                                           \
               " stmfd sp!,{r1,lr}\n"           /*SPSR_irq und LR_irq retten*/ \
               " msr  cpsr_c, #0x53\n"          /*Umschalten nach SVC-Mode */  \
               " stmfd sp!,{lr}\n" : : : "r1" ); /*LR_svc retten*/
```

```
#define DISABLE_NESTED_ISR                                                   \
  asm volatile(" ldmfd sp!,{lr}\n"           /*LR_svc restaurieren*/         \
               " msr  cpsr_c, #0xd2\n"       /*Umschalten auf IRQ-Mode*/     \
               " ldmfd sp!,{r1,lr}\n"        /*SPSR und LR wieder herstellen*/ \
               " msr   spsr_fc,r1\n" : : : "r1" );
```

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Vectored IRQ: Unterbrechbare ISR, Latenzzeiten

- Es wurden 3 ISR implementiert:
  - **ISR1 hat die höchste Priorität und dauert 200 µs,**
  - **ISR2 hat eine mittlere Priorität und dauert 400 µs und**
  - **ISR3 hat die niedrigste Priorität und dauert 300 µs.**

  Jede Interruptquelle feuert maximal einmal pro 1 msec.
- Welche Interrupt-Latenzzeiten sind im schlimmsten Fall auf Grund der obigen Angaben für die ISRs mindestens zu erwarten?

Ausnahme- und Interrupt-verarbeitung im LPC 2468

Ein- und Ausschalten von Interrupts in der Hitex Laufzeitumgebung

- **Funktionen in armVIC.h:**
  - **unsigned disableIRQ(void);**
  - **unsigned enableIRQ(void);**
  - **unsigned restoreIRQ(unsigned oldCPSR);**
  - **unsigned disableIFIQ(void);**
  - **unsigned enableFIQ(void);**
  - **unsigned restoreFIQ(unsigned oldCPSR);**