اینتراپت‌ها و منابع آن

در مبحث اینتراپت‌های میکروکنترلرهای Philips ARM دو نوع منبع اینتراپت خواهیم داشت: ۱. اینتراپتهای سخت افزاری ۲. اینتراپت‌های نرم افزاری یا swi که این‌گونه اینتراپت‌ها نظیر event در برنامه‌های کامپیوتری می‌باشند.

اینتراپت‌های سخت افزاری از نظر اولویت به دو دسته FIQ و RIQ تقسیم می‌شوند. وقفه FIQ دارای بالاترین اولویت رسیدگی می‌باشد و بهتر است که یکی از اینتراپت‌ها در حالت FIQ باشد و بقیه در حالت IRQ. وقفه‌های IRQ نیز دارای اولویت بندی خاصی می‌باشند.

نکته: وقفه با اولویت بالاتر ابتدا انجام می‌شود.

در این مبحث به اینتراپتهای سخت افزاری خواهیم پرداخت.

سری میکروکنترلرهای LPC23xx شامل ۳۲ منبع وقفه به ترتیب و شرح زیر می‌باشد:

31 I2S	30 I2C2	29 UART3	28 UART2	27 TIMER3	26 TIMER2	25 GPDMA	24 SD/MMC
23 CAN1/2	22 USB	21 ETHERNET	20 BOD	19 I2C1	18 ADC0	17 EINT3	16 EINT2
15 EINT1	14 EINT0	13 TRC	12 PLL	11 SSP1	10 SPI/SSP0	9 I2C0	8 PWM1
7 UART1	6 UART0	5 TIMER1	4 TIMER0	3 ARMCORE2	2 ARMCORE1	1 -	0 WDT

Table 1

نکته: در رجیسترهای کنترلی و وضعیت و اولویت بندی IRQ و ... ترتیب به صورت فوق می‌باشد.

رجیسترهای کنترلی و وضعیت اینتراپت‌ها:

رجیستر VICIntselect: این رجیستر در صورت یک شدن هر یک از بیت‌های آن اینتراپت مربوطه را در وضعیت FIQ قرار می‌دهد(صفر به منزله IRQ).

مثال: در زیر ETHERNET در حالت FIQ قرار می‌گیرد

```
VICIntselect=0X00200000;
```

رجیستر VICIntEnable: در این رجیستر با یک شدن هر بیت می‌توان اینتراپت مربوطه را فعال کرد. این نکته حائز اهمیت است که صفر بودن یک بیت تاثیری در فعالیت اینتراپت ندارد و این رجیستر در صورت یک شدن بیتی از آن صرفاً اینتراپت مربوطه را فعال می‌کند. مثال زیر این موضوع را مشخص می‌کند:

```
VICIntEnable=0x00000001; //enable WDT: watch dog timer

VICIntEnable=0x00000010; //enable timer0 without any effect on WDT
```

رجیسترهای VICVectPriority0-31: این ۳۲ رجیستر برای تغییر و جابجایی اولویت‌های مذکور می‌باشند، بطوریکه هر رجیستر در ابتدای راه‌اندازی مقدار ۱۵ را در خود دارا بوده و و قرار دادن عددی کمتر از ۱۵ در آن اولویت وقفه متناظر را جابجا می‌کند و در صورتی که دو وقفه در یک اولویت تنظیم شوند وقفه‌ای که در جدول یک نزدیکتر به صفر است سریعتر انجام می‌شود.

رجیستر VICFIQStatus: در صورت ایجاد وقفه از نوع FIQ این رجیستر منبع ایجاد کننده وقف را نشان می‌دهد. در مثال زیر دو وقفه timer0 و timer1 در حالت FIQ قرار دارند و با رجیستر مذکور منبع تولید کننده وقفه شناسایی می‌شود:

```
void FIQ_Handler( void ) __irq
{
  if ( VICFIQStatus & (0x1<<4) && VICIntEnable & (0x1<<4) )
  {
      Timer0FIQHandler();
  }
  if ( VICFIQStatus & (0x1<<5) && VICIntEnable & (0x1<<5) )
  {
      Timer1FIQHandler();
  }
}
```

نکته: همانطور که گفته شد وقفه از نوع FIQ بالاترین اولویت را داراست و این وقفه شامل یک زیر برنامه می‌باشد که در صورت تعیین بیش از یک وقفه در حالت FIQ باید به صورت فوق عمل کرد.

رجیسترهای VICVectAddr0-31: همانطور که می‌دانیم هر وقفه باعث تغییر محارنده برنامه یا PC به عدد خاصی می‌شود یعنی با ایجاد وقفه PC محل خاصی از ROM را نشان می‌دهد که منحصراً برای اینتراپت خاصی تعیین شده است. در میکروکنترلرهای قدیمی این عدد خاص ثابت بود و غیر قابل تغییر و کاربر می‌بایست یا کد زیر برنامه اینتراپت را در محل مورد نظر و بشکل محدود می‌نوشت یا با دستورات پرش به محل دیگری کد را منتقل می‌کرد. این اعمال باعث کندی و پیچیدگی برنامه و اجرای زیر برنامه می‌شد اما در میکروکنترلرهای ARM این قابلیت وجود دارد که کاربر محل زیر برنامه وقفه را خود مشخص کند در اینصورت دیگر نه محدودیت ظرفیت وجود دارد نه جابجایی کد به محل دیگر بعد از ایجاد وقفه. رجیسترهای VICVectAddr برای آدرس دهی‌های مذکور می‌باشند و هر رجیستر اینتراپت متنظر با جدول یک را آدرس دهی می‌کند. مثال زیر نحوه آدرس دهی را نشان می‌دهد:

```
void int0_isr ( void ) __irq {

FIO0SET=0x200000;  //set led1

EXTINT=1;                //Acknowledge

VICVectAddr=0;           //Acknowledge

j=0x0;

}
```

```
int main (){

.

.

.

VICVectAddr14=(unsigned long)int0_isr;
```

همانطور که ملاحضه می‌کنید زیر برنامه وقفه خارجی صفر را int0_isr نام گذاری کرده و به VICVectAddr14 آدرس زیر برنامه را می‌دهیم.

نکته: برای تشکیل زیر برنامه وقفه باید کلمه __irq در ابتدا یا انتهای تعریف تابع نوشته شود تا کامپایلر آنرا بعنوان یک ISR شناسایی کند.

Acknowledge یا تائید اجرای وقفه: وقتی وقفه‌ای رخ می‌دهد پرچم مربوط به آن بالا می‌ماند تا زمانیکه رجیستر **VICVectAddr** با عددی دیگر مقدار دهی جدید شود این عمل باعث صفر شدن پرچم وقفه می‌شود. در صورتی که این عمل اجرا نشود ممکن است اعمالی مانند مدهای خواب و ذخیره سازی انرژی در میکروکنترلر انجام نشود(در واقع ممکن است میکروکنترلر وارد حالات idle یا موارد دیگر نشود).

نکته: وقفه‌ها برای تائیدیه رجیسترهای دیگری نیز دارند بطور مثال رجیستر EXTINT برای وقفه‌های خارجی می‌باشند که در ادامه به آن خواهیم پرداخت.

وقفه‌های خارجی

رجیستر EXTMODE: این رجیستر نوع حساسیت وقفه را مشخص می‌کند. حساسیت می‌تواند level **à** 0 یا حساسیت به سطح باشد یا edge **à** 1 یا حساسیت به لبه. جدول زیر ساختار این رجیستر را نشان می‌دهد:

-	EXTMODE3	EXTMODE2	EXTMODE1	EXTMODE0
Bit: 7:4	Bit: 0	Bit: 0	Bit: 0	Bit: 0

Table 2

رجیستر EXTPOLAR: این رجیستر برای تعیین حساسیت به نوع پلاریزاسیون سیگنال اعمالی می‌باشد، که 0 **à** LOW LEVEL OR FALLING EDGE و 1 **à** HIGH LEVEL OR RISING EDGE می‌باشد. ساختار این رجیستر را در جدول زیر ملاحضه می‌کنید:

-	EXTPOLAR3	EXTPOLAR2	EXTPOLAR1	EXTPOLAR0
Bit: 7:4	Bit: 0	Bit: 0	Bit: 0	Bit: 0

رجیستر EXTINT: پرچم وقفه‌های خارجی در این رجیستر قرار دارند و جدول زیر ساختار این رجیستر می‌باشد:

-	EXTINT3	EXTINT2	EXTINT1	EXTINT0
Bit: 7:4	Bit: 0	Bit: 0	Bit: 0	Bit: 0

همانطور که گفته شد باید بعد از اجرای زیر برنامه وقفه این رجیستر برای تائید یا Acknowledge مقدار دهی شود اما در اینجا باید بیت مورد نظر با یک مقدار دهی شود بصورت زیر:

```
EXTINT=1;
```

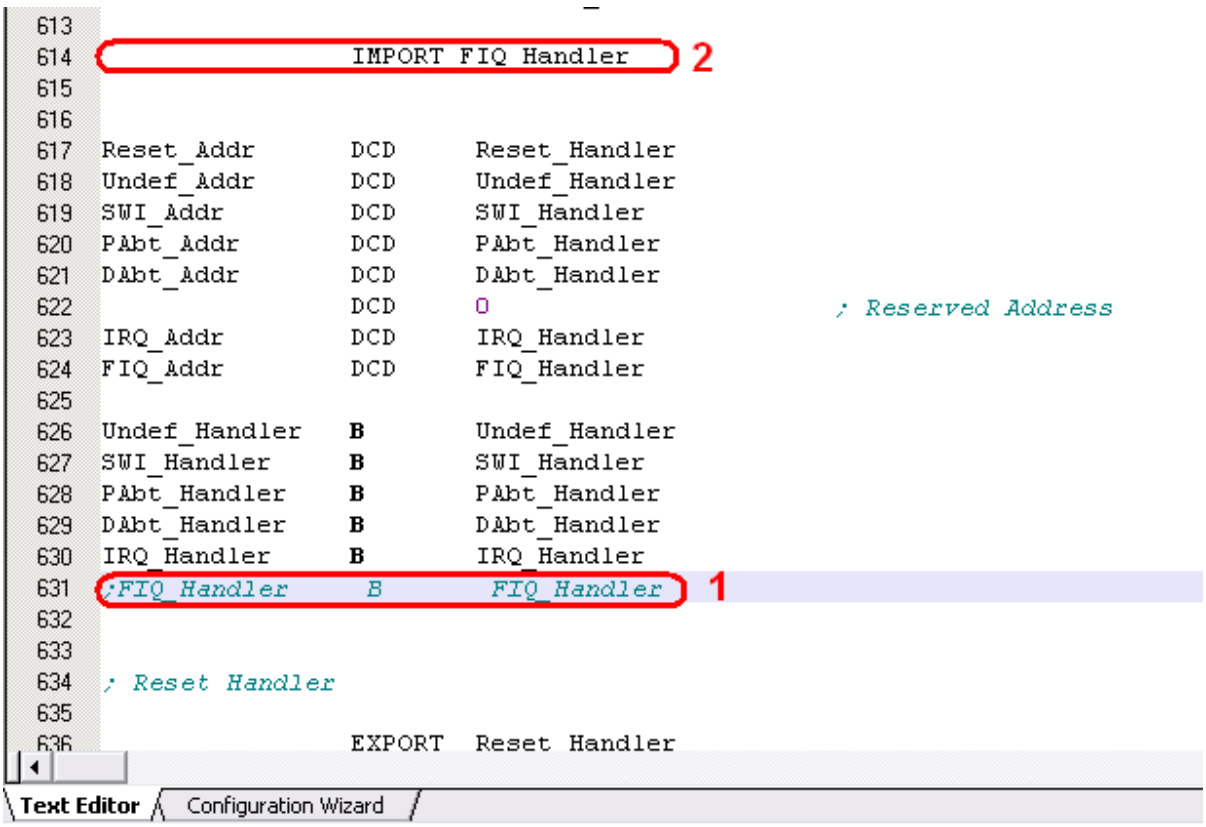
این خط برنامه Acknowledge برای وقفه خارجی صفر می‌باشد.

نکته: این عمل بعلاوه مقدار دهی VICVectAddr تکمیل کننده تائید وقفه می‌باشد که در مثال صفحه دو انجام شده است.

نکته: در معماری میکروکنترلر ARM وقفه عمومی تعبیه نشده است یعنی نمی‌توان با یک دستور یا تغییر در بیت خاصی کلیه اینتراپتها را فعال یا غیر فعال کرد. اما در کتابخانه irq.h دو تابع تعبیه شده است که می‌توان در هنگام اجرای زیر برنامه وقفه جلوی کلیه وقفه‌ها را گرفت و تا زمانیکه زیر برنامه تمام شود وقفه‌ای ایجاد نشود(اصطلاحا جلوی وقفه‌های تو در تو را گرفت). این توابع به شرح زیر می‌باشند(بخشی از کد موجود در کتابخانه مذکور):

```
static DWORD sysreg;          /* used as LR register */
#define IENABLE __asm {MRS sysreg, SPSR; MSR CPSR_c, #SYS32Mode}
#define IDISABLE __asm {MSR CPSR_c,#(IRQ32Mode|I_Bit);MSR SPSR_cxsf, sysreg}
```

نکته: در زمینه وقفه نوع FIQ، فایل startup.s مخصوص برنامه keil نیاز به تنظیمات خاصی می‌باشد تا زیر برنامه FIQ در صورت بروز وقفه اجرا شود در غیر اینصورت میکروکنترلر بعد از ایجاد وقفه وارد یک حلقه بی‌نهایت قرار می‌گیرد. تصویر زیر این فایل را در برنامه نشان می‌دهد:



فایل را در حالت Text Editor قرار دهید تا برنامه بشکل فوق نمایش یابد. قسمت یک مشخص شده تصویر را پیدا و حذف کنید و قسمت دو را به آن اضافه کنید. در اینصورت زیر برنامه با نام FIQ_Handler که در برنامه اصلی خود قرار داده اید اجرا خواهد شد.

مثال: در این برنامه با توجه به نکات فوق وقفه خارجی صفر در حالت FIQ پیکره بندی شده است

```
#include "LPC23xx.H"
void delay(unsigned long i);
unsigned long j;
void FIQ_Handler( void ) __irq {
FIO0SET= 0x200000; //set led1
EXTINT=1; //Acknowledge
VICVectAddr=0; //Acknowledge
j= 0x0;
}
int main (){
SCS |= 0x00000001;
FIO0MASK= 0x00000000;
FIO0DIR= 0xFFFFFFFF;
FIO0CLR= 0x00000000;
PINSEL4 = 0x00100000; //set PORT2.10 to EINT0
EXTMODE=1; //set EINT0 to edge sensitive
EXTPOLAR=0; //set EINT0 to falling edge sensitive
VICIntSelect= 0x00004000; //set eint0 to FIQ
VICIntEnable= 0x00004000; //enable Eint0
```

```
while(1){
delay(3000000);
FIO0CLR= 0x200000; //reset led1
}
}
void delay(unsigned long i){
for (j=0;j<i;j++);
}
```

مثال: در مثال زیر همان برنامه بالا در حالت IRQ پیکره بندی شده است

```
#include "LPC23xx.H"
void delay(unsigned long i);
unsigned long j;
void int0_isr( void ) __irq {
FIO0SET= 0x200000; //set led1
EXTINT=1; //Acknowledge
VICVectAddr=0; //Acknowledge
j= 0x0;
}
int main (){
SCS |= 0x00000001;
FIO0MASK= 0x00000000;
FIO0DIR= 0xFFFFFFFF;
FIO0CLR= 0x00000000;
PINSEL4 = 0x00100000; //set PORT2.10 to EINT0
EXTMODE=1; //set EINT0 to edge sensitive
EXTPOLAR=0; //set EINT0 to falling edge sensitive
VICIntSelect= 0x00004000; //set eint0 to IRQ
VICIntEnable= 0x00004000; //enable Eint0
VICVectAddr14=(unsigned long) int0_isr; //set int0_isr for interrupt service
rout
ine
while(1){
delay(3000000);
FIO0CLR= 0x200000; //reset led1
}
}
void delay(unsigned long i){
for (j=0;j<i;j++);
}
```