advanced digital chips inc

CANTUS-CAN

- *INTERRUPT* -

32bits EISC Microprocessor *CANTUS*

Ver 1.1
April 24, 2013

Advanced Digital Chips Inc.

History

2013-02-19	Released	
2013-04-24	Modified	CANTUS-CAN

CANTUS-CAN Application Note : #0004B INTERRUPT

Office

(Gwanyang-dong, Keumkang Penterium IT Tower) 22F, A-Tower,
282, Hagui-ro, Dongan-gu, Anyang-si, Gyeonggi-do, SEOUL 431-810 Rep. of KOREA

Tel : +82-31-463-7500
Fax : +82-31-463-7588
URL : http://www.adc.co.kr

— Table of Contents —

1 SUMMARY 6

2 INTERNAL INTERRUPT 9

2.1 REGISTER SET 9
2.2 FUNCTION SET 15

3 EXTERNAL INTERRUPT 18

3.1 REGISTER SET 18
3.2 FUNCTION SET 1 20
3.3 FUNCTION SET 2 23
3.4 FUNCTION SET 3 25

— List of Figures —

그림 2-1 Internal Interrupt Register Set *9*
그림 3-1 External Interrupt Register Set *18*

— List of Tables —

표 1-1 Interrupt Vector and Priority 6
표 2-2 Internal Interrupt Mode Register (IINTMOD) 10
표 2-3 Interrupt Mask Set Register (MASKSET) 11
표 2-4 Interrupt Enable Register (INTEN) 12
표 2-5 Interrupt Pending Register (INTPEND) 13
표 2-6 Interrupt Pending Register (PENDCLR) 14
표 3-1 Port Alternate Function Register 3 19
표 3-2 External Interrupt Mode and External PIN Level Register (EINTMOD) 19

1 Summary

이 문서는 CANTUS의 INTERRUPT에 대한 Application Note이다.

CANTUS는 32개의 Interrupt Vector를 갖고 있다. 이중 30개는 TIMER, SPI, TWI, UART 등과 같은 내부 Peripheral에서 발생하는 Interrupt(Internal Interrupt)를 위함이며, 2개는 외부 Pin eirq 0, eirq 1에 의해 발생하는 Interrupt(External Interrupt)를 위함이다.

Interrupt Vector and Priority

Interrupt Priority는 EIRQ0가 가장 높다. Vector Table을 통해 해당 Vector Address에 Interrupt Service Routine을 설정한다.

표 1-1 Interrupt Vector and Priority

Vector No.	*Description*	*Vector Address*
0x3F	UART Ch7 | OCR 3B Interrupt	0x000000FC
0x3E	DMA Ch1 Interrupt	0x000000F8
0x3D	TIMER Ch7 | KEYSCAN Interrupt	0x000000F4
0x3C	GPIO5 Interrupt	0x000000F0
0x3B	UART Ch6 | OCR 3A Interrupt	0x000000EC
0x3A	NFCTRL | SDHC Interrupt	0x000000E8
0x39	TIMER Ch6 Interrupt	0x000000E4
0x38	GPIO4 Interrupt	0x000000E0
0x37	UART Ch5 / OCR 2B Interrupt	0x000000DC
0x36	VOICE Interrupt	0x000000D8
0x35	TIMER Ch5 Interrupt	0x000000D4
0x34	GPIO3 Interrupt	0x000000D0
0x33	UART Ch4 | OCR 2A Interrupt	0x000000CC
0x32	TWI / PWK and RTC Interrupt	0x000000C8
0x31	TIMER Ch4 Interrupt	0x000000C4
0x30	GPIO2 Interrupt	0x000000C0
0x2F	UART Ch3 Interrupt	0x000000BC
0x2E	SPI Interrupt	0x000000B8
0x2D	TIMER Ch3 Interrupt	0x000000B4
0x2C	GPIO1 Interrupt	0x000000B0
0x2B	UART Ch2 Interrupt	0x000000AC
0x2A	USB Device Interrupt	0x000000A8
0x29	TIMER Ch2 Interrupt	0x000000A4
0x28	GPIO0 Interrupt	0x000000A0
0x27	UART Ch1 Interrupt	0x0000009C
0x26	DMA Ch0 Interrupt	0x00000098
0x25	TIMER Ch1 Interrupt	0x00000094
0x24	EIRQ1 Interrupt	0x00000090
0x23	UART Ch0 Interrupt	0x0000008C
0x22	I2S Interrupt	0x00000088
0x21	TIMER Ch0 Interrupt	0x00000084
0x20	EIRQ0 Interrupt (Highest Priority)	0x00000080

Initialize Interrupt

CANTUS에서 Interrupt를 사용하기 위해서는 Vector Table을 정의하여 이를 0 번지에 위치(cantus.ld)하도록 하고, AE32000 Core를 Vectored Interrupt로 설정하고 Interrupt Enable을 설정 한다. 또한 Vector Base를 지정하여 동작 수행 시간을 단축하고, Interrupt의 hooking을 가능하게 한다.

Initialize Interrupt의 수행은 Interrupt.c에 구현되어 있으며, 사용자는 함수를 호출하여 Parameter만 전달한다. 또한 이 문서에서는 해당 내용을 다루지 않는다. 관심이 있는 사용자는 AE32000 Core에 관련된 Manual을 참조하라.

배포되는 SDK에서 Interrupt를 사용하는 방법은 3.3 Function Set 3를 참조하라.

```
cantus.ld 중

  SECTIONS
  {
    .text : {
    *(.vects)
    …

interrupt.c 중

  extern void Reset_Start();
  typedef void (*fp)(void);
  fp vector_table[] __attribute__((section (“.vects”)))=
  {
    Reset_Start,
    nmi_autovector,
    NOTUSERISRT,
    double_fault_exception,
    bus_error_exception,
    …
  }

  void init_interrupt()

startup.S 중

  .section .text
  .global _Reset_Start
  .type _Reset_Start, @function
  _Reset_Start:
```

Interrupt Source로부터 Interrupt가 발생 하면[A], INTEN에 의해 선별된 후 INTPEND에 기록되며[B], MASKSET에 의해 선별된 후 Core에 Interrupt Request를 전달한다[C]. Core는 복귀지점을 저장한 후 Interrupt Request에 따라 Interrupt Vector Table에 정의된 Vector Address로 분기하여 Interrupt Service Routine을 수행하고 복귀한다[D].

복귀 전에 INTPEND가 Clear되지 않으면 위 [C]에서 [D]를 수행하지 않으므로 반드시 복귀 전에 PENDCLR을 통해 해당 Vector의 Pending bit를 Clear하여야 한다.

Internal Interrupt는 IINTMOD에 따라 High Level 또는 Rising Edge에서 발생한다.
External Interrupt는 EINTMOD에 따라 Low Level, High Level, Falling Edge, Rising Edge, Any Edge에서 발생한다.

이 문서는 CANTUS의 Internal Interrupt와 External Interrupt를 사용하는 방법을 기술 한다.

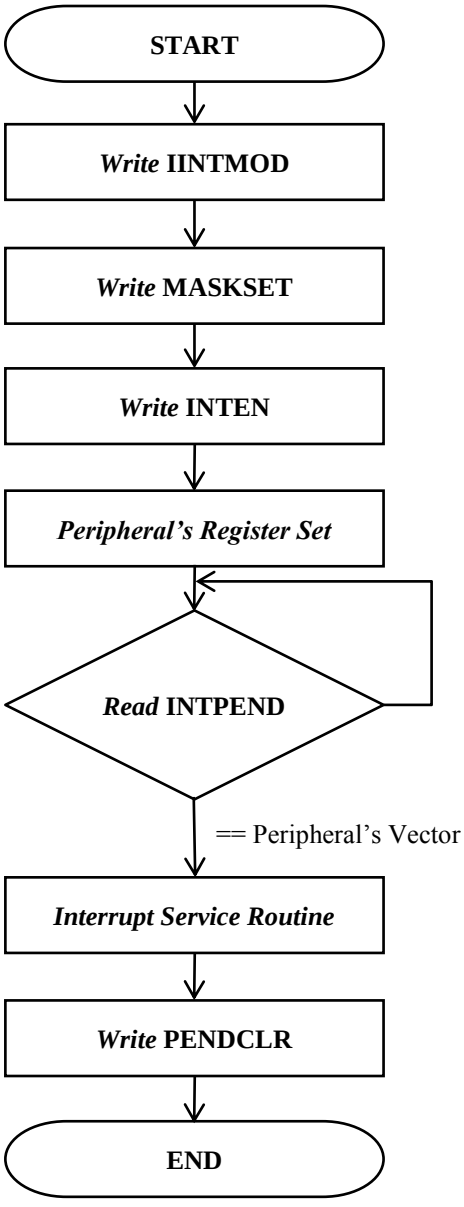
CANTUS의 Interrupt는 CANTUS Datasheet ‘9 INTERRUPT CONTROLLER’를 참조 하라.
AE32000 Core의 Exception은 EISC Software Developer Guide ‘5 Exception Handling’을 참조하라.

2 Internal Interrupt

2.1 Register Set

CANTUS의 Internal Interrupt를 사용하기 위해 다음과 같은 순서로 Register를 설정한다. 어떤 Peripheral은 특정 Register에 Interrupt Enable[1]을 갖고 있어 이를 설정하여야 한다.

그림 2-1 Internal Interrupt Register Set



[1] INTEN과는 별개이다.

IINTMOD (Internal Interrupt Mode Register)

Internal Interrupt Mode를 High Level 또는 Rising Edge로 설정한다.

표 2-2 Internal Interrupt Mode Register (IINTMOD)

Bit	*R/W*	*Description*	*Default Value*
31	R/W	Vector No. 0x3F Interrupt Mode bit	1
30	R/W	Vector No. 0x3E Interrupt Mode bit	1
29	R/W	Vector No. 0x3D Interrupt Mode bit	1
28	R/W	Vector No. 0x3C Interrupt Mode bit	1
27	R/W	Vector No. 0x3B Interrupt Mode bit	1
26	R/W	Vector No. 0x3A Interrupt Mode bit	1
25	R/W	Vector No. 0x39 Interrupt Mode bit	1
24	R/W	Vector No. 0x38 Interrupt Mode bit	1
23	R/W	Vector No. 0x37 Interrupt Mode bit	1
22	R/W	Vector No. 0x36 Interrupt Mode bit	1
21	R/W	Vector No. 0x35 Interrupt Mode bit	1
20	R/W	Vector No. 0x34 Interrupt Mode bit	1
19	R/W	Vector No. 0x33 Interrupt Mode bit	1
18	R/W	Vector No. 0x32 Interrupt Mode bit	1
17	R/W	Vector No. 0x31 Interrupt Mode bit	1
16	R/W	Vector No. 0x30 Interrupt Mode bit	1
15	R/W	Vector No. 0x2F Interrupt Mode bit	1
14	R/W	Vector No. 0x2E Interrupt Mode bit	1
13	R/W	Vector No. 0x2D Interrupt Mode bit	1
12	R/W	Vector No. 0x2C Interrupt Mode bit	1
11	R/W	Vector No. 0x2B Interrupt Mode bit	1
10	R/W	Vector No. 0x2A Interrupt Mode bit	1
9	R/W	Vector No. 0x29 Interrupt Mode bit	1
8	R/W	Vector No. 0x28 Interrupt Mode bit	1
7	R/W	Vector No. 0x27 Interrupt Mode bit	1
6	R/W	Vector No. 0x26 Interrupt Mode bit	1
5	R/W	Vector No. 0x25 Interrupt Mode bit	1
4	-	Reserved	-
3	R/W	Vector No. 0x23 Interrupt Mode bit	1
2	R/W	Vector No. 0x22 Interrupt Mode bit	1
1	R/W	Vector No. 0x21 Interrupt Mode bit	1
0	-	Reserved	-

*** Internal Interrupt Mode bit
0 : High Level Mode
1 : Rising Edge Mode

MASKSET (Interrupt Mask Set Register)

이 Register를 설정하면 INTPEND에 기록된 Interrupt Vector의 Interrupt Request를 Core에 전달하고 Interrupt Vector Address로 분기하여 정의한 Interrupt Service Routine을 수행 후 복귀 한다.

표 2-3 Interrupt Mask Set Register (MASKSET)

Bit	R/W	Description	Default Value
31	W	Vector No. 0x3F Interrupt Request Set bit	0
30	W	Vector No. 0x3E Interrupt Request Set bit	0
29	W	Vector No. 0x3D Interrupt Request Set bit	0
28	W	Vector No. 0x3C Interrupt Request Set bit	0
27	W	Vector No. 0x3B Interrupt Request Set bit	0
26	W	Vector No. 0x3A Interrupt Request Set bit	0
25	W	Vector No. 0x39 Interrupt Request Set bit	0
24	W	Vector No. 0x38 Interrupt Request Set bit	0
23	W	Vector No. 0x37 Interrupt Request Set bit	0
22	W	Vector No. 0x36 Interrupt Request Set bit	0
21	W	Vector No. 0x35 Interrupt Request Set bit	0
20	W	Vector No. 0x34 Interrupt Request Set bit	0
19	W	Vector No. 0x33 Interrupt Request Set bit	0
18	W	Vector No. 0x32 Interrupt Request Set bit	0
17	W	Vector No. 0x31 Interrupt Request Set bit	0
16	W	Vector No. 0x30 Interrupt Request Set bit	0
15	W	Vector No. 0x2F Interrupt Request Set bit	0
14	W	Vector No. 0x2E Interrupt Request Set bit	0
13	W	Vector No. 0x2D Interrupt Request Set bit	0
12	W	Vector No. 0x2C Interrupt Request Set bit	0
11	W	Vector No. 0x2B Interrupt Request Set bit	0
10	W	Vector No. 0x2A Interrupt Request Set bit	0
9	W	Vector No. 0x29 Interrupt Request Set bit	0
8	W	Vector No. 0x28 Interrupt Request Set bit	0
7	W	Vector No. 0x27 Interrupt Request Set bit	0
6	W	Vector No. 0x26 Interrupt Request Set bit	0
5	W	Vector No. 0x25 Interrupt Request Set bit	0
4	W	Vector No. 0x24 Interrupt Request Set bit	0
3	W	Vector No. 0x23 Interrupt Request Set bit	0
2	W	Vector No. 0x22 Interrupt Request Set bit	0
1	W	Vector No. 0x21 Interrupt Request Set bit	0
0	W	Vector No. 0x20 Interrupt Request Set bit	0

*** Interrupt Request Set bit
0 : No Effect interrupt Mask.
1 : Pending interrupt is allowed to become active (interrupts sent to CPU)

INTEN (Interrupt Enable Register)

이 Register를 설정하면 Interrupt Event를 INTPEND에 기록한다.

표 2-4 Interrupt Enable Register (INTEN)

Bit	*R/W*	*Description*	*Default Value*
31	R/W	Vector No. 0x3F Interrupt Enable bit	0
30	R/W	Vector No. 0x3E Interrupt Enable bit	0
29	R/W	Vector No. 0x3D Interrupt Enable bit	0
28	R/W	Vector No. 0x3C Interrupt Enable bit	0
27	R/W	Vector No. 0x3B Interrupt Enable bit	0
26	R/W	Vector No. 0x3A Interrupt Enable bit	0
25	R/W	Vector No. 0x39 Interrupt Enable bit	0
24	R/W	Vector No. 0x38 Interrupt Enable bit	0
23	R/W	Vector No. 0x37 Interrupt Enable bit	0
22	R/W	Vector No. 0x36 Interrupt Enable bit	0
21	R/W	Vector No. 0x35 Interrupt Enable bit	0
20	R/W	Vector No. 0x34 Interrupt Enable bit	0
19	R/W	Vector No. 0x33 Interrupt Enable bit	0
18	R/W	Vector No. 0x32 Interrupt Enable bit	0
17	R/W	Vector No. 0x31 Interrupt Enable bit	0
16	R/W	Vector No. 0x30 Interrupt Enable bit	0
15	R/W	Vector No. 0x2F Interrupt Enable bit	0
14	R/W	Vector No. 0x2E Interrupt Enable bit	0
13	R/W	Vector No. 0x2D Interrupt Enable bit	0
12	R/W	Vector No. 0x2C Interrupt Enable bit	0
11	R/W	Vector No. 0x2B Interrupt Enable bit	0
10	R/W	Vector No. 0x2A Interrupt Enable bit	0
9	R/W	Vector No. 0x29 Interrupt Enable bit	0
8	R/W	Vector No. 0x28 Interrupt Enable bit	0
7	R/W	Vector No. 0x27 Interrupt Enable bit	0
6	R/W	Vector No. 0x26 Interrupt Enable bit	0
5	R/W	Vector No. 0x25 Interrupt Enable bit	0
4	R/W	Vector No. 0x24 Interrupt Enable bit	0
3	R/W	Vector No. 0x23 Interrupt Enable bit	0
2	R/W	Vector No. 0x22 Interrupt Enable bit	0
1	R/W	Vector No. 0x21 Interrupt Enable bit	0
0	R/W	Vector No. 0x20 Interrupt Enable bit	0

*** Interrupt Enable bit
0 : Interrupt Disable
1 : Interrupt Enable

INTPEND (Interrupt Pending Register)

INTEN에 설정된 Interrupt Vector의 Interrupt Event를 기록한다.

표 2-5 Interrupt Pending Register (INTPEND)

Bit	*R/W*	*Description*	*Default Value*
31	R	Vector No. 0x3F Interrupt Pending bit	-
30	R	Vector No. 0x3E Interrupt Pending bit	-
29	R	Vector No. 0x3D Interrupt Pending bit	-
28	R	Vector No. 0x3C Interrupt Pending bit	-
27	R	Vector No. 0x3B Interrupt Pending bit	-
26	R	Vector No. 0x3A Interrupt Pending bit	-
25	R	Vector No. 0x39 Interrupt Pending bit	-
24	R	Vector No. 0x38 Interrupt Pending bit	-
23	R	Vector No. 0x37 Interrupt Pending bit	-
22	R	Vector No. 0x36 Interrupt Pending bit	-
21	R	Vector No. 0x35 Interrupt Pending bit	-
20	R	Vector No. 0x34 Interrupt Pending bit	-
19	R	Vector No. 0x33 Interrupt Pending bit	-
18	R	Vector No. 0x32 Interrupt Pending bit	-
17	R	Vector No. 0x31 Interrupt Pending bit	-
16	R	Vector No. 0x30 Interrupt Pending bit	-
15	R	Vector No. 0x2F Interrupt Pending bit	-
14	R	Vector No. 0x2E Interrupt Pending bit	-
13	R	Vector No. 0x2D Interrupt Pending bit	-
12	R	Vector No. 0x2C Interrupt Pending bit	-
11	R	Vector No. 0x2B Interrupt Pending bit	-
10	R	Vector No. 0x2A Interrupt Pending bit	-
9	R	Vector No. 0x29 Interrupt Pending bit	-
8	R	Vector No. 0x28 Interrupt Pending bit	-
7	R	Vector No. 0x27 Interrupt Pending bit	-
6	R	Vector No. 0x26 Interrupt Pending bit	-
5	R	Vector No. 0x25 Interrupt Pending bit	-
4	R	Vector No. 0x24 Interrupt Pending bit	-
3	R	Vector No. 0x23 Interrupt Pending bit	-
2	R	Vector No. 0x22 Interrupt Pending bit	-
1	R	Vector No. 0x21 Interrupt Pending bit	-
0	R	Vector No. 0x20 Interrupt Pending bit	-

*** Interrupt Pending Register의 각 비트의 값은 해당 인터럽트가 발생하였음을 나타낸다.

Interrupt Pending Register의 값은 Interrupt Pending Clear 레지스터에 의해 Clear된다.

일반적으로 해당 Interrupt가 끝날 때 Clear한다.

PENDCLR (Interrupt Pending Clear Register)

Interrupt Vector Number값으로 씀으로써 INTPEND의 Event를 Clear한다. INTPEND를 Clear하지 않으면 더 이상 Core에 Interrupt Request를 발생하지 않는다.

표 2-6 Interrupt Pending Register (PENDCLR)

Bit	*R/W*	*Description*	*Default Value*
31 : 8	R	Reserved	-
7 : 0	W	Interrupt Pending Register Clear Value (0x20 ~ 0x3F)	0xFF

*** Interrupt Pending Register를 Clear 하기 위해서는 Interrupt Vector No. 값으로 Write 한다.

2.2 Function Set

아래는 CANTUS의 Interrupt를 사용한 TIMER의 예이다. TIMER관련 설정은 AN_0003B_TIMER를 참조하라.

```
// #define TIMER2_MASKSET_ENABLE
#define F_TIMER_SET   1
#define F_TIMER_CLEAR 0
volatile unsigned int F_timer_set = F_TIMER_CLEAR;

#pragma interrupt
void TIMER2_ISR()
{
  #ifdef TIMER2_MASKSET_ENABLE
  F_timer_set = F_TIMER_SET;
  debugstring("TIMER2 ISR\r\n\r\n");

  *R_PENDCLR = (INTNUM_TIMER2+0x20);
  #endif
}
main()
{
  …

  unsigned int tmcnt=1647;
  unsigned int i = 0;

  *R_IINTMOD |= (1<<INTNUM_TIMER2);

  #ifdef TIMER2_MASKSET_ENABLE
  *R_MASKSET |= (1<<INTNUM_TIMER2);
  #endif

  *R_INTEN |= (1<<INTNUM_TIMER2);

  // Peripheral's Register Set
  *R_TP2CTRL = F_TPCTRL_CNTRST;
  *R_TP2CTRL &= ~(F_TPCTRL_CNTRST);
  *R_TM2CNT = tmcnt;

  while(1)
  {
    debugprintf("i = %d\r\n",i);

    F_timer_set = F_TIMER_CLEAR;
    *R_TM2CTRL = (F_TMCTRL_TMOD_TIMER
    | F_TMCTRL_PFSEL_32768 | F_TMCTRL_TMEN);

    #ifdef TIMER2_MASKSET_ENABLE
    while( !(F_timer_set == F_TIMER_SET));
    #else
    while( !(*R_INTPEND & (1<<INTNUM_TIMER2) ));
    *R_PENDCLR = (INTNUM_TIMER2+0x20);
    #endif

    *R_TM2CTRL &= ~(F_TMCTRL_TMEN);
    i++;
  }…
}
```

```
// interrupt.c

…

extern void TIMER2_ISR(void);
fp vector_table[] __attribute__((section (".vects")))=
{
  Reset_Start       , /* V00 : Reset Vector          */
  nmi_autovector    , /* V01 : NMI Vector            */

  …

  SDK_ExternISR9    , /* V29 : User Vector 9         */
  TIMER2_ISR        , /* V29 : User Vector 9         */
  SDK_ExternISR10   , /* V2A : User Vector 10        */

  …

};
```

TIMER2_MASKSET_ENABLE

▶ TIMER2_MASKSET_ENABLE이 define되면, R_MASKSET를 설정하여, Interrupt Request를 설정한다. define하지 않으면, Polling으로 R_INTPEND를 읽어 Interrupt 발생 유무를 확인한다.

TIMER2_ISR

▶ TIMER2_ISR함수는 TIMER2 Interrupt Service Routine이다. TIMER2 Interrupt가 발생하여 R_INTPEND에 기록되고, R_MASKSET에 TIMER2가 설정되어있으면, Core가 분기하여 실행할 함수가 된다. TIMER2_ISR은 위 interrupt.c와 같이 Interrupt Vector Table에 등록되어 있어야 한다.

*R_IINTMOD

Internal Interrupt Mode Register는 SDK/Include/CANTUS/interrupt.h에

#define R_IINTMOD ((volatile unsigned int*)(0x80020808))

로 정의되어 있다.

▶ 여기서는 R_IINTMOD에 `(1<<INTNUM_TIMER2)`를 씀으로써 TIMER2 Interrupt Mode를 Rising Edge로 설정한다.

*R_MASKSET

Interrupt Mask Set Register는 SDK/Include/CANTUS/interrupt.h에

#define R_MASKSET ((volatile unsigned int*)(0x80020814))

로 정의되어 있다.

▶ 여기서는 R_MASKSET에 `(1<<INTNUM_TIMER2)`를 씀으로써 TIMER2 Interrupt Request를 설정한다.

*R_INTEN

Interrupt Enable Register는 SDK/Include/CANTUS/interrupt.h에

#define R_INTEN ((volatile unsigned int*)(0x80020810))

로 정의되어 있다.

▶ 여기서는 R_INTEN에 `(1<<INTNUM_TIMER2)`를 씀으로써 TIMER2 Interrupt Enable을 설정한다.

*R_INTPEND

Interrupt Pending Register는 SDK/Include/CANTUS/interrupt.h에

#define R_INTPEND ((volatile unsigned int*)(0x8002080C))

로 정의되어 있다.

▶ 여기서는 R_INTPEND를 읽어 (1<<INTNUM_TIMER2)과 비교하여 TIMER2 Interrupt가 발생하였는지 판단 한다. R_INTEN으로 Enable된 TIMER2는 Interrupt Event를 R_INTPEND에 기록 한다.

*R_PENDCLR

Interrupt Pending Clear Register는 SDK/Include/CANTUS/interrupt.h에

#define R_PENDCLR ((volatile unsigned int*)0x80020800)

로 정의되어 있다.

▶ 여기서는 R_PENDCLR에 TIMER2 Interrupt Vector Number(0x29)를 씀으로써 R_INTPEND를 Clear한다.

TIMER2_MASKSET_ENABLE이 define된 상태에서 R_INTPEND를 Clear하지 않으면 더 이상 Core에 Interrupt Request를 하지 않는다. 따라서 `while( !(F_timer_set == F_TIMER_SET));`에 멈추게 된다.

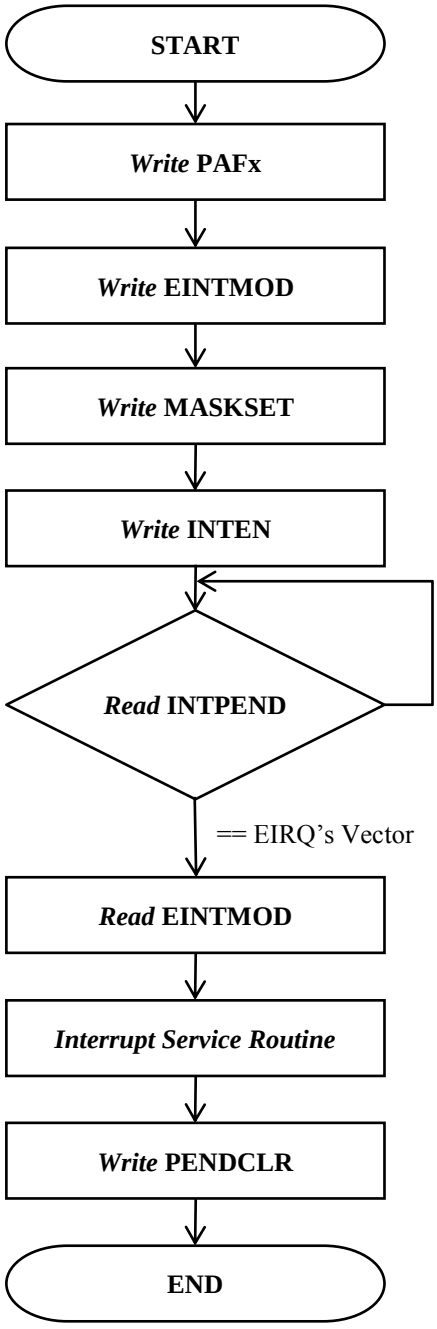
TIMER2_MASKSET_ENABLE이 define되지 않은 상태에서 R_INTPEND를 Clear하지 않으면 `while( !(*R_INTPEND & (1<<INTNUM_TIMER2) ));`가 항상 `while(0)`가 되어 `i`가 급격하게 증가한다.

3 External Interrupt

3.1 Register Set

CANTUS의 External Interrupt를 사용하기 위해 다음과 같은 순서로 Register를 설정한다.

그림 3-1 External Interrupt Register Set



PAFx

해당 Port의 Pin을 eirq로 사용하기 위해 '00b'를 쓴다. Port의 Pin은 bit(2)에 쓴 값으로 설정된다.

표 3-1 Port Alternate Function Register 3

Group	*Index*	*Pin*	*1st*	*2nd*	*3rd*	*4th(default)*
			00	*01*	*10*	*11*
PAF3 0x8002002C	**0**	66	EIRQ[0]		nWAIT	P3.0
	1	67	EIRQ[1]		nBE[1]	P3.1
	2	68			NDFL_nWE	P3.2
	3	69			NDFL_ALE	P3.3
	4	70			NDFL_CLE	P3.4
	5	71			NDFL_nCS	P3.5
	6	72			NDFL_nRE	P3.6
	7	73	nNMI		NDFL_nBUSY	P3.7

EINTMOD (External Interrupt Mode and External PIN Level Register)

External Interrupt Mode를 설정한다. 또한 현재 EIRQ PIN의 Level을 읽을 수 있다.

표 3-2 External Interrupt Mode and External PIN Level Register (EINTMOD)

Bit	*R/W*	*Description*	*Default Value*
31:8	R	Reserved	-
7	R	EIRQ1ST : EIRQ1 PIN Level	-
6 : 4	R/W	EIRQ1MOD : EIRQ1 Active State 000 : Low Level 001 : High Level 010 : Falling Edge 011 : Rising Edge 1xx : Any Edge	010
3	R	EIRQ0ST : EIRQ0 PIN Level	-
2 : 0	R/W	EIRQ0MOD : EIRQ0 Active State 000 : Low Level 001 : High Level 010 : Falling Edge 011 : Rising Edge 1xx : Any Edge	010

3.2 Function Set 1

아래는 CANTUS의 External Interrupt 0를 Any Edge Mode로 사용한 예이다.

```
#define EIRQ0_MASKSET_ENABLE

#pragma interrupt
void EIRQ0_ISR()
{
  if(*R_EINTMOD & (F_EINTMOD_EIRQ0ST) )
  {
    debugstring("EIRQ0 Rising Edge In ISR\r\n\r\n");
  }
  else
  {
    debugstring("EIRQ0 Falling Edge In ISR\r\n");
  }

  *R_PENDCLR = (INTNUM_EIRQ0+0x20);
}
int main()
{
  …
  *R_PAF3 = F_PAF3_0_EIRQ0;

  *R_EINTMOD = F_EINTMOD_EIRQ0MOD_AE;
  *R_IINTMOD |= (1<<INTNUM_EIRQ0);

  #ifdef EIRQ0_MASKSET_ENABLE
  *R_MASKSET |= (1<<INTNUM_EIRQ0);
  #endif

  *R_INTEN |= (1<<INTNUM_EIRQ0);

  while(1)
  {
    #ifndef EIRQ0_MASKSET_ENABLE
    if(*R_INTPEND & (1<<INTNUM_EIRQ0) )
    {
      if(*R_EINTMOD & (F_EINTMOD_EIRQ0ST) )
      {
        debugstring("EIRQ0 Rising Edge\r\n\r\n");
      }
      else
      {
        debugstring("EIRQ0 Falling Edge\r\n");
      }

      *R_PENDCLR = (INTNUM_EIRQ0+0x20);
    }
    #endif
  }
  …
}
```

```
// interrupt.c

…

extern void EIRQ0_ISR(void);
fp vector_table[] __attribute__((section (".vects")))=
{
  Reset_Start     , /* V00 : Reset Vector          */
  nmi_autovector  , /* V01 : NMI Vector            */

  …

  // SDK_ExternISR0  , /* V20 : User Vector 20        */
  EIRQ0_ISR       , /* V20 : User Vector 20        */
  SDK_ExternISR1  , /* V21 : User Vector 21        */

  …

};
```

EIRQ0_MASKSET_ENABLE

▶ EIRQ0_MASKSET_ENABLE이 define되면, R_MASKSET를 설정하여, Interrupt Request를 설정한다. define하지 않으면, Polling으로 R_INTPEND를 읽어 Interrupt 발생 유무를 확인한다.

EIRQ0_ISR

▶ EIRQ0_ISR함수는 EIRQ0 Interrupt Service Routine이다. EIRQ0 Interrupt가 발생하여 R_INTPEND에 기록되고, R_MASKSET에 EIRQ0가 설정되어있으면, Core가 분기하여 실행할 함수가 된다. EIRQ0_ISR은 위 interrupt.c와 같이 Interrupt Vector Table에 등록되어 있어야 한다.

*R_PAF3

Port Alternate Function Register는 SDK/Include/CANTUS/paf.h에

#define R_PAF3 ((volatile U32*)0x8002002C)
#define F_PAF3_0_EIRQ0 (0 << 0)

로 정의되어 있다.

▶ 여기서는 R_PAF3에 F_PAF3_0_EIRQ0를 씀으로써 Pin 66를 EIRQ0로 설정한다.

*R_EINTMOD

External Interrupt Mode Register는 SDK/Include/CANTUS/interrupt.h에

#define R_EINTMOD ((volatile unsigned int*)0x80020804)

로 정의되어 있다.

▶ 여기서는 R_EINTMOD에 F_EINTMOD_EIRQ0MOD_AE를 Write하여 EIRQ0MOD를 Any Edge Mode로 설정한다. Interrupt가 발생하면, R_EINTMOD를 읽어 EIRQ0 PIN의 Level을 알 수 있다. Any Edge Mode에서 Interrupt가 발생하고 EIRQ0 PIN이 Low Level이면 Falling Edge Event, High Level이면 Rising Edge Event이다.

***R_MASKSET**

▶ 여기서는 R_MASKSET에 (1<<INTNUM_EIRQ0)를 씀으로써 External 0 Interrupt Request를 설정한다.

***R_INTEN**

▶ 여기서는 R_INTEN에 (1<<INTNUM_EIRQ0)를 씀으로써 External 0 Interrupt Enable을 설정한다.

***R_INTPEND**

▶ 여기서는 R_INTPEND를 읽어 (1<<INTNUM_EIRQ0)와 비교하여 EIRQ0 Interrupt가 발생하였는지 판단한다. R_INTEN으로 Enable된 EIRQ0는 Interrupt Event를 R_INTPEND에 기록 한다.

***R_PENDCLR**

▶ 여기서는 R_PENDCLR에 EIRQ0 Interrupt Vector Number(0x20)를 씀으로써 R_INTPEND를 Clear한다. R_INTPEND를 Clear하지 않으면 더 이상 Core에 Interrupt Request를 하지 않는다.

3.3 Function Set 2

위와 같이 사용하면 Interrupt Service Routine을 interrupt.c에서 정의 하여야 하지만, CANTUS SDK는 임의로 Vector Table이 미리 정의되어 있고, 다음과 같이 미리 정의된 이름과 같은 함수를 외부에 정의하면, 미리 정의된 함수를 대체 할 수 있다.

```
#define EIRQ0_MASKSET_ENABLE

#pragma interrupt
void SDK_ExternISR0()
{
  if(*R_EINTMOD & (F_EINTMOD_EIRQ0ST) )
  {
    debugstring("EIRQ0 Rising Edge In ISR\r\n\r\n");
  }
  else
  {
    debugstring("EIRQ0 Falling Edge In ISR\r\n");
  }

  *R_PENDCLR = (INTNUM_EIRQ0+0x20);
}
int main()
{
  …
  *R_PAF3 = F_PAF3_0_EIRQ0;

  *R_EINTMOD = F_EINTMOD_EIRQ0MOD_AE;
  *R_IINTMOD |= (1<<INTNUM_EIRQ0);

  #ifdef EIRQ0_MASKSET_ENABLE
  *R_MASKSET |= (1<<INTNUM_EIRQ0);
  #endif

  *R_INTEN |= (1<<INTNUM_EIRQ0);

  while(1)
  {
    #ifndef EIRQ0_MASKSET_ENABLE
    if(*R_INTPEND & (1<<INTNUM_EIRQ0) )
    {
      if(*R_EINTMOD & (F_EINTMOD_EIRQ0ST) )
      {
        debugstring("EIRQ0 Rising Edge\r\n\r\n");
      }
      else
      {
        debugstring("EIRQ0 Falling Edge\r\n");
      }

      *R_PENDCLR = (INTNUM_EIRQ0+0x20);
    }
    #endif
  }
  …
}
```

```
// interrupt.c

…

extern void EIRQ0_ISR(void);
fp vector_table[] __attribute__((section (".vects")))=
{
  Reset_Start     , /* V00 : Reset Vector          */
  nmi_autovector  , /* V01 : NMI Vector            */

  …

  SDK_ExternISR0  , /* V20 : User Vector 20        */
  SDK_ExternISR1  , /* V21 : User Vector 21        */

  …

};
```

SDK_ExternISR0

▶ `SDK_ExternISR0()`함수는 EIRQ0 Interrupt Service Routine이다. EIRQ0 Interrupt가 발생하여 R_INTPEND에 기록되고, R_MASKSET에 EIRQ0가 설정되어있으면, Core가 분기하여 실행할 함수가 된다. 또한 이 함수는 interrupt.c에서 미리 정의가 되어 있는 함수 이다. 하지만 `__attritube__((weak))`에 의해 외부에서 다시 정의된 함수로 대체 되게 된다. 이렇게 외부에서 정의된 Interrupt Service Routine은 반드시 `#pragma interrupt`로 정의 되어야 하며, 내부에서 Pending Clear를 해주어야 한다.

3.4 Function Set 3

Function Set 1과 같이 사용하면 Interrupt Service Routine을 interrupt.c에 정의 하여야 하고, Function Set 2와 같이 사용하면 내부에서 미리 정의한 이름을 알아야 한다. 다행스럽게도 CANTUS SDK는 임의로 미리 정의된 Vector Table을 다음과 같이 `setinterrupt()`를 통해 외부에서 정의한 함수로 대체하여 Interrupt Service Routine으로 사용할 수 있도록 한다.

```
#define EIRQ0_MASKSET_ENABLE

void EIRQ0_ISR()
{
  if(*R_EINTMOD & (F_EINTMOD_EIRQ0ST) )
  {
    debugstring("EIRQ0 Rising Edge In ISR\r\n\r\n");
  }
  else
  {
    debugstring("EIRQ0 Falling Edge In ISR\r\n");
  }
}
int main()
{
  …
  *R_PAF3 = F_PAF3_0_EIRQ0;

  *R_EINTMOD = F_EINTMOD_EIRQ0MOD_AE;
  setinterrupt(INTNUM_EIRQ0, EIRQ0_ISR);

  #ifdef EIRQ0_MASKSET_ENABLE
  EnableInterrupt(INTNUM_EIRQ0, TRUE);
  #else
  *R_INTEN |= (1<<INTNUM_EIRQ0);
  #endif

  while(1)
  {
    #ifndef EIRQ0_MASKSET_ENABLE
    if(*R_INTPEND & (1<<INTNUM_EIRQ0) )
    {
      if(*R_EINTMOD & (F_EINTMOD_EIRQ0ST) )
      {
        debugstring("EIRQ0 Rising Edge\r\n\r\n");
      }
      else
      {
        debugstring("EIRQ0 Falling Edge\r\n");
      }

      *R_PENDCLR = (INTNUM_EIRQ0+0x20);
    }
    #endif
  }
  …
}
```

```
// interrupt.c

…

#pragma interrupt_handler
static void NOTUSEDISR()
{
  U32 INTST = *R_INTPEND;
  debugprintf(" NOT USED ISR : regst:0x%x\r\n",INTST);
  …

}

static void (*UserVector_table[32])(void) =
{
  NOTUSEDISR,
  NOTUSEDISR,
  …

};

#define ISRHOOK(A)  void SDK_ExternISR##A(void) __attribute__((weak)) ;\
…
asm("ldi     0x20 + "#A",%r8");\
asm("ldi     0x80020800,%r9");\
asm("st      %r8,(%r9,0)");\
asm("ldi     _UserVector_table +(4 * "#A") ,%r8");\
asm("ld      (%r8,0),%r9");\
asm("jalr    %r9");\
…

ISRHOOK(0)
ISRHOOK(1)
…

fp vector_table[] __attribute__((section (".vects")))=
{
  Reset_Start     , /* V00 : Reset Vector          */
  nmi_autovector  , /* V01 : NMI Vector            */

  …

  SDK_ExternISR0  , /* V20 : User Vector 20        */
  SDK_ExternISR1  , /* V21 : User Vector 21        */

  …

};
```

EIRQ0_MASKSET_ENABLE

▶ EIRQ0_MASKSET_ENABLE이 define되면, `EnableInterrupt()`함수를 사용하여, R_MASKSET, R_INTEN을 설정한다. define하지 않으면, Polling으로 R_INTPEND를 읽어 Interrupt 발생 유무를 확인한다.

EIRQ0_ISR

▶ EIRQ0_ISR함수는 EIRQ0 Interrupt Service Routine이다. EIRQ0 Interrupt가 발생하여 R_INTPEND에 기록되고, R_MASKSET에 EIRQ0가 설정되어있으면, Core가 분기하여 실행할 함수가 된다.

SDK의 Interrupt는 위와 같이 정의 되어 있고, 아래와 같이 동작 한다.
vector_table[#+0x20]은 IRQ에 의해 Core가 분기할 주소(함수의 주소)를 정의하고 있다. 이 주소 중 SDK_ExternISR#은 ISRHOOK(#)에서 UserVector_table[#]으로 분기할 수 있도록 Inline Assembly로 정의되어 있다. UserVector_table[#]은 모두 NOTUSEDISR()의 주소로 등록된다.
따라서 모든(User) Vector에 대해 IRQ가 발생하면 NOTUSEDISR()함수를 실행하게 된다.

여기서 setinterrupt()함수를 통해 특정 Vector(#)에 대해 UserVector_table[#]에 정의된 NOTUSEDISR()의 주소를 일반함수 EIRQ0_ISR()의 주소로 변경한다. 이렇게 변경된 Vector(#+0x20)는 IRQ가 발생하면, 다음과 같은 순서로 참조되어 Interrupt Service Routine을 실행하게 된다.

vector_table[#+0x20] → SDK_ExternISR# → UserVector_table[#] → EIRQ0_ISR()

등록된 일반함수는 내부에서 Pending Clear를 하지 않는다.

setinterrupt()

setinterrupt()는 SDK/include/ CANTUS/interrupt.h에

```
BOOL setinterrupt(INTERRUPT_TYPE intnum, void (*fp)());
```

로 선언되어 있고,

SDK/Library/interrupt.c에

```
BOOL setinterrupt(INTERRUPT_TYPE intnum, void (*fp)())
{
        if (intnum >= INTNUM_MAX)
                return FALSE;

        UserVector_table[intnum]=fp;
        return TRUE;

}
```

로 정의되어 있다.

EnableInterrupt()

EnableInterrupt()는 SDK/include/ CANTUS/interrupt.h에

void EnableInterrupt(INTERRUPT_TYPE num, BOOL b);

로 선언되어 있고,

SDK/Library/interrupt.c에

```
void EnableInterrupt(INTERRUPT_TYPE num, BOOL b)
{
	CRITICAL_ENTER();
	if (!b) //disable
	{
		*R_INTEN &= (~(1 << num));
		*R_INTMASKCLR |= ( 1 << num);
	}
	else
	{
		*R_MASKSET = (1 << num);
		*R_INTEN |= ( 1 << num);
	}
	CRITICAL_EXIT();
}
```

로 정의되어 있다.