CANTUS-CAN

- *TIMER* -

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 : #0003B TIMER

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 TIMER TIMER/COUNTER MODE 7

2.1 REGISTER SET 7
2.2 FUNCTION SET 1 13
2.3 FUNCTION SET 2 15

3 TIMER PWM MODE 17

3.1 REGISTER SET 17
3.2 FUNCTION SET 1 20
3.3 FUNCTION SET 2 23

4 TIMER CAPTURE MODE 24

4.1 REGISTER SET 24
4.2 FUNCTION SET 26

— List of Figures —

그림 2-1 TIMER Timer/Counter Mode Register Set *7*
그림 3-1 TIMER PWM Mode Register Set *17*
그림 3-1 TIMER Capture Mode Register Set *24*

— List of Tables —

표 2-1 Timer Pre-scale Control Register (TPxCTRL) 8
표 2-2 Timer Counter / PWM Period Register (TMxCNT) 8
표 2-3 Timer Control Register (TMxCTRL) 9
표 2-4 Interrupt Enable Register (INTEN) 10
표 2-5 Interrupt Pending Register (INTPEND) 11
표 2-6 Interrupt Pending Clear Register (PENDCLR) 12
표 4-1 Port Alternate Function Register 5 18
표 3-7 Capture Counter / PWM Duty (TMxDUT) 18
표 4-6 PWM Pulse Count (TMxPUL) 19

1 Summary

이 문서는 CANTUS의 TIMER[1]에 대한 Application Note이다.

CANTUS는 Timer/Counter, Capture, PWM 기능을 가진 32bit 8채널의 TIMER를 갖고 있다.
TIMER는 15bit Timer/Counter Pre-scaler에서 1/2, 1/8, 1/32, 1/128, 1/512, 1/2048, 1/8192, 1/32768로 분주된 Clock으로 동작한다. 15bit Timer/Counter Pre-scaler는 Source Clock으로 System Clock[2]과 External Clock 중 선택하여 사용하며, 이를 분주시켜 TIMER에 Clock을 공급한다.

Timer/Counter Mode는 0x0의 초기값에서 매 Clock 마다 내부 Counter 값을 1씩 증가하여 TMxCNT 값과 같을 때 0x0이 되면서 Core에 Interrupt Request를 발생한다.

PWM Mode는 Pin TMO#으로 설정된 Period와 Duty에 따라 PWM Pulse를 출력한다. 출력하는 PWM Pulse의 시작 Level과 Pulse Number를 설정할 수 있고, 1회 출력 후 정지할 수 있다.

Capture Mode는 Pin CAP#으로 입력된 신호의 Low/High Pulse, Low Pulse, High Pulse, Falling to Falling Edge, Rising to Rising Edge의 5가지 형태의 Period를 측정한다.

이 문서는 CANTUS의 Timer를 Timer/Counter Mode / Capture Mode / PWM Mode로 사용하기 위한 방법을 기술 한다.

CANTUS의 TIMER는 CANTUS Datasheet '10 TIMER'를 참조 하라.

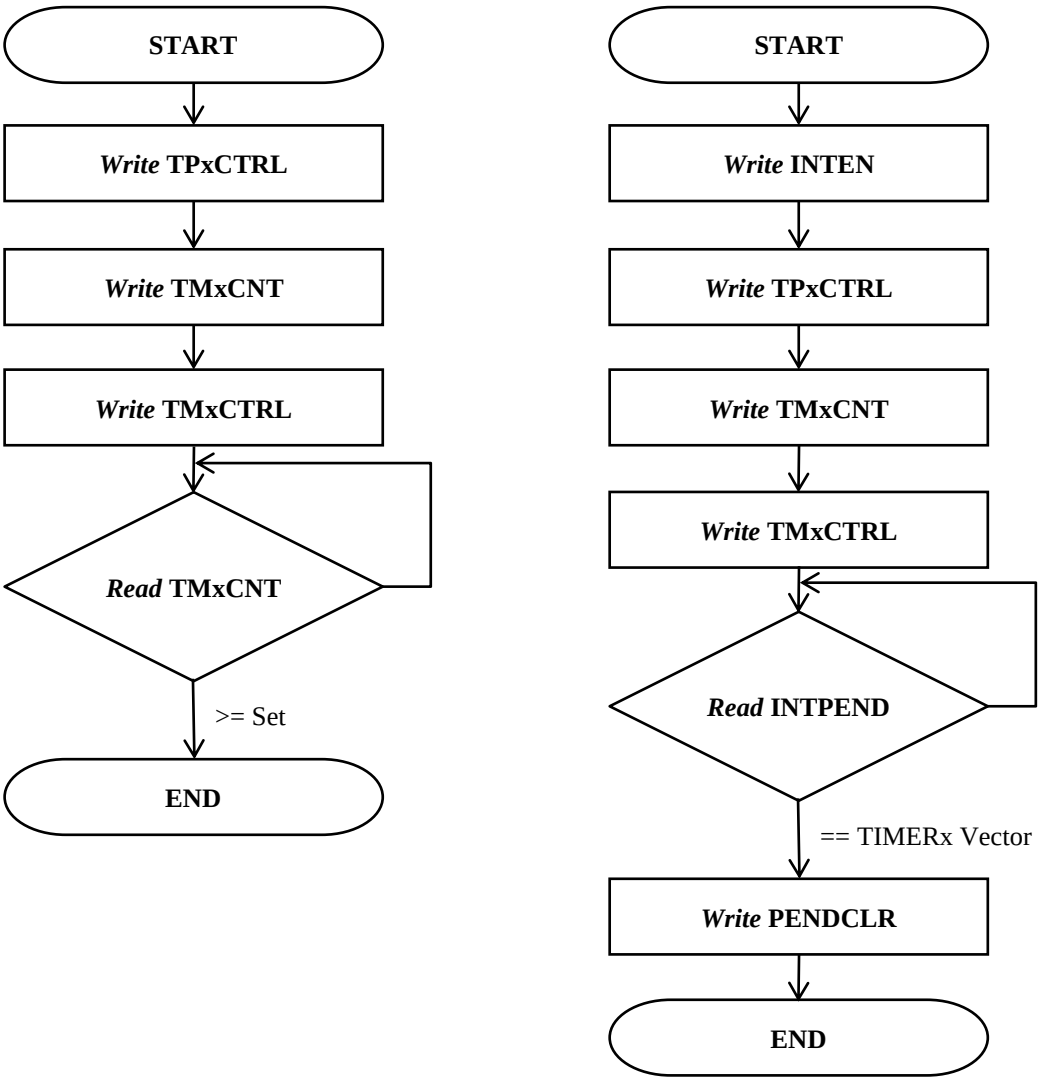
[1] TIMER 또는 Timer/Counter.
[2] APB Clock

2 TIMER Timer/Counter Mode

2.1 Register Set

CANTUS의 TIMER를 Timer/Counter Mode로 사용하기 위해 다음과 같은 순서로 Register를 설정한다. 설정한 TMxCNT값과 현재 TMxCNT값을 비교하는 방법과, Interrupt를 이용하는 방법이 있다.

그림 2-1 TIMER Timer/Counter Mode Register Set



TPxCTRL (Timer Pre-scale Control Register)

Timer/Counter Pre-scaler의 Source Clock으로 사용할 Clock을 선택하고, Timer/Counter Pre-Scaler를 Reset 한다.

표 2-1 Timer Pre-scale Control Register (TPxCTRL)

Bit	*R/W*	*Description*	*Default Value*
31 : 2	R	Reserved	-
1	R/W	CNTCLR : Pre-scale Counter and Timer Counter Reset When this bit is "1", the Timer Pre-scale and Counter will be reset.	0
0	R/W	CLKSEL : Pre-scale Clock Selection 0 : System clock 1 : CAPx	0

*** CAPx는 Timer 채널 별로 할당되어 있다.

TMxCNT (Timer Counter / PWM Period Register)

Timer Period를 설정하기 위해 TMxCNT에 값을 쓴다. Timer Period는 아래와 같은 식을 따른다.

표 2-2 Timer Counter / PWM Period Register (TMxCNT)

Bit	*R/W*	*Description*	*Default Value*
31 : 0	R/W	(Timer mode) - Write : Timer Counter Value - Read : Current Up-counter Value (PWM mode) - Read/Write : PWM Period Value	0xFFFF_FFFF

$$Timer\ \ Period = \frac{1}{Clock\ \ Source\ \ Frequency} \times \frac{1}{\Pr e - scaler\ \ Factor} \times (TMCNT)\ \ [\sec] \qquad \{\Pr e - scaler\ \ Factor \geq 3\}$$

$$Timer\ \ Period = \frac{1}{Clock\ \ Source\ \ Frequency} \times \frac{1}{\Pr e - scaler\ \ Factor} \times (TMCNT + 1)\ \ [\sec] \qquad \{\Pr e - scaler\ \ Factor < 3\}$$

TMxCTRL (Timer Control Register)

TMOD에 '00b'를 씀으로써 Timer/Counter Mode로 설정하고, Pre-scaler의 분주를 설정하기 위해 PFSEL에 값을 쓴다. TMEN에 '1b'을 Write하면 TIMER가 동작 한다.

표 2-3 Timer Control Register (TMxCTRL)

Bit	*R/W*	*Description*	*Default Value*
31 : 16	R	Reserved	-
15 : 14	R/W	TMOD : Timer/Counter Mode 00 : Timer 01 : PWM 1x : Capture	00
13	R/W	OCEN : Output Compare Mode Enable bit for channel2 and channel 3	0
12	R	Reserved	0
12	R	OVST : Capture Overflow Status bit Read시 Overflow status bit가 clear된다.	0
10 : 8	R/W	CAPMOD : Capture Mode Selection 00x : Low/High Pulse Capture mode 010 : Low Pulse Capture mode 011 : High Pulse Capture mode 10x : Failing to Failing Period Capture mode 11x : Rising to Rising Period Capture mode	000
7	R	Reserved	-
6	R/W	PWMO : PWM Output One Period Generation 0 : Disable 1 : Enable	0
5	R/W	PWML : PWM Output Start Level 0 : Start Level is Low 1 : Start Level is High	0
4	R	Reserved	-
3 : 1	R/W	PFSEL : Pre-scale Factor Selection 000 : 1/2 001 : 1/8 010 : 1/32 011 : 1/128 100 : 1/512 101 : 1/2048 110 : 1/8192 111 : 1/32768	111
0	R/W	TMEN : Timer/Counter or PWM Enable 0 : Disable 1 : Enable	0

*** PWM Output One Period Generation : PWM 모드로 동작할 때, 한 주기만 발생시키는 bit이다.
한 주기가 발생한 이후에는 자동으로 PWM은 Disable된다.

INTEN (Interrupt Enable Register)

이 Register를 설정하면 Interrupt Event를 Interrupt Controller에서 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 Clear 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 1

아래는 CANTUS의 TIMER 2를 설정하여 대략 1초마다 i를 증가시키고, i 값을 UART로 출력하는 예이다. 여기서 APB Clock은 48MHz이다.

```
main()
{
  …

  U32 tmcnt;
  U32 temp;
  U32 i=0;

  tmcnt = 1464;

  *R_TP2CTRL = F_TPCTRL_CNTRST;
  *R_TP2CTRL &= ~( F_TPCTRL_CNTRST);

  *R_TM2CNT = 0xFFFF;
  *R_TM2CTRL = 0;

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

    *R_TM2CTRL = (F_TMCTRL_PFSEL_32768 | F_TMCTRL_TMEN);

    while(1)
    {
      temp = *R_TM2CNT;
      if(temp >= tmcnt)
      {
        *R_TM2CTRL = 0;

        *R_TP2CTRL = F_TPCTRL_CNTRST;
        *R_TP2CTRL &= ~( F_TPCTRL_CNTRST);

        i++;

        break;
      }
    }
  }
  …
}
```

***R_TP2CTRL**

Timer Pre-scale Control Register 는 SDK/Include/CANTUS/timer.h에

```
#define R_TP2CTRL ((volatile unsigned int*)0x80021040)
#define F_TPCTRL_CNTRST                (1<<1)
```

로 정의되어 있다.

▶ 여기서는 Timer Pre-scale Control Register 2에 F_TPCTRL_CNTRST을 대입하여 Pre-scale Counter와 Timer Counter를 초기화 한다.

*R_TM2CNT

Timer Counter / PWM Period Register 는 SDK/Include/CANTUS/timer.h에

#define R_TM2CNT ((volatile unsigned int*)0x80021048)

로 정의되어 있다.

▶ 여기서는 Timer Counter / PWM Period Register 2에 0xFFFFFFFF를 대입한다. 1초를 위한 값은 1464로 tmcnt에 쓰고, 이를 Timer Counter / PWM Period Register와 비교하여 1초가 지남을 판단한다.

*R_TM2CTRL

Timer Control Register는 SDK/Include/CANTUS/timer.h에

```
#define R_TM2CTRL ((volatile unsigned int*)0x80021044)
#define F_TMCTRL_PFSEL_32768        (7<<1)
#define F_TMCTRL_TMEN               (1<<0)
```

로 정의되어 있다.

▶ 여기서는 Timer Control Register 2에 (F_TMCTRL_PFSEL_32768 | F_TMCTRL_TMEN)을 대입하여 Pre-scale Factor를 1/32768로 설정하고, Timer/Counter를 동작 시킨다.

2.3 Function Set 2

아래는 CANTUS의 TIMER 2를 설정하여 대략 1초마다 i를 증가시키고, i 값을 UART로 출력하는 예로 Interrupt를 사용한다. 여기서 APB Clock은 48MHz이고, INTNUM_TIMER2는 0x9이다. Interrupt에 관련된 내용은 AN_0004B_INTERRUPT를 참조하라.

```
main()
{
  …

  U32 tmcnt;
  U32 temp;
  U32 i=0;

  tmcnt = 1464;

  *R_INTEN |= (1<<INTNUM_TIMER2);

  *R_TP2CTRL = F_TPCTRL_CNTRST;
  *R_TP2CTRL &= ~(F_TPCTRL_CNTRST);

  *R_TM2CNT = tmcnt;
  *R_TM2CTRL = 0;

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

    *R_TM2CTRL = (F_TMCTRL_PFSEL_32768 | F_TMCTRL_TMEN);

    while(1)
    {
      temp = *R_INTPEND;

      if((temp &= (1<<INTNUM_TIMER2)) == (1<<INTNUM_TIMER2))
      {
        *R_TM2CTRL = 0;
        *R_PENDCLR = (INTNUM_TIMER2+0x20);

        i++;
        break;
      }
    }
  }
  …
}
```

***R_INTEN**

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

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

로 정의되어 있다.

▶ 여기서는 Interrupt Enable Register 0에 (1<<INTNUM_TIMER2)를 대입하여 TIMER 2 Interrupt Enable을 설정 한다.

*R_TM2CNT

Timer Counter / PWM Period Register 는 SDK/Include/CANTUS/timer.h에

#define R_TM2CNT ((volatile unsigned int*)0x80021048)

로 정의되어 있다.

▶ 여기서는 Timer Counter / PWM Period Register 2에 tmcnt를 대입한다. 1초를 위한 값은 1464로 tmcnt에 저장되어 있다.

*R_TM2CTRL

Timer Control Register는 SDK/Include/CANTUS/timer.h에

#define R_TM2CTRL ((volatile unsigned int*)0x80021044)
#define F_TMCTRL_PFSEL_32768 (7<<1)
#define F_TMCTRL_TMEN (1<<0)

로 정의되어 있다.

▶ 여기서는 Timer Control Register 2에 (F_TMCTRL_PFSEL_32768 | F_TMCTRL_TMEN)을 대입하여 Pre-scale Factor를 1/32768로 설정하고, Timer/Counter를 동작 시킨다.

*R_INTPEND

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

#define R_INTPEND ((volatile unsigned int*)(0x8002080c))

로 정의되어 있다.

▶ 여기서는 Interrupt Pending Register를 읽어 (1<<INTNUM_TIMER2)와 비교한다. R_INTEN로 Enable된 TIMER 2은 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.

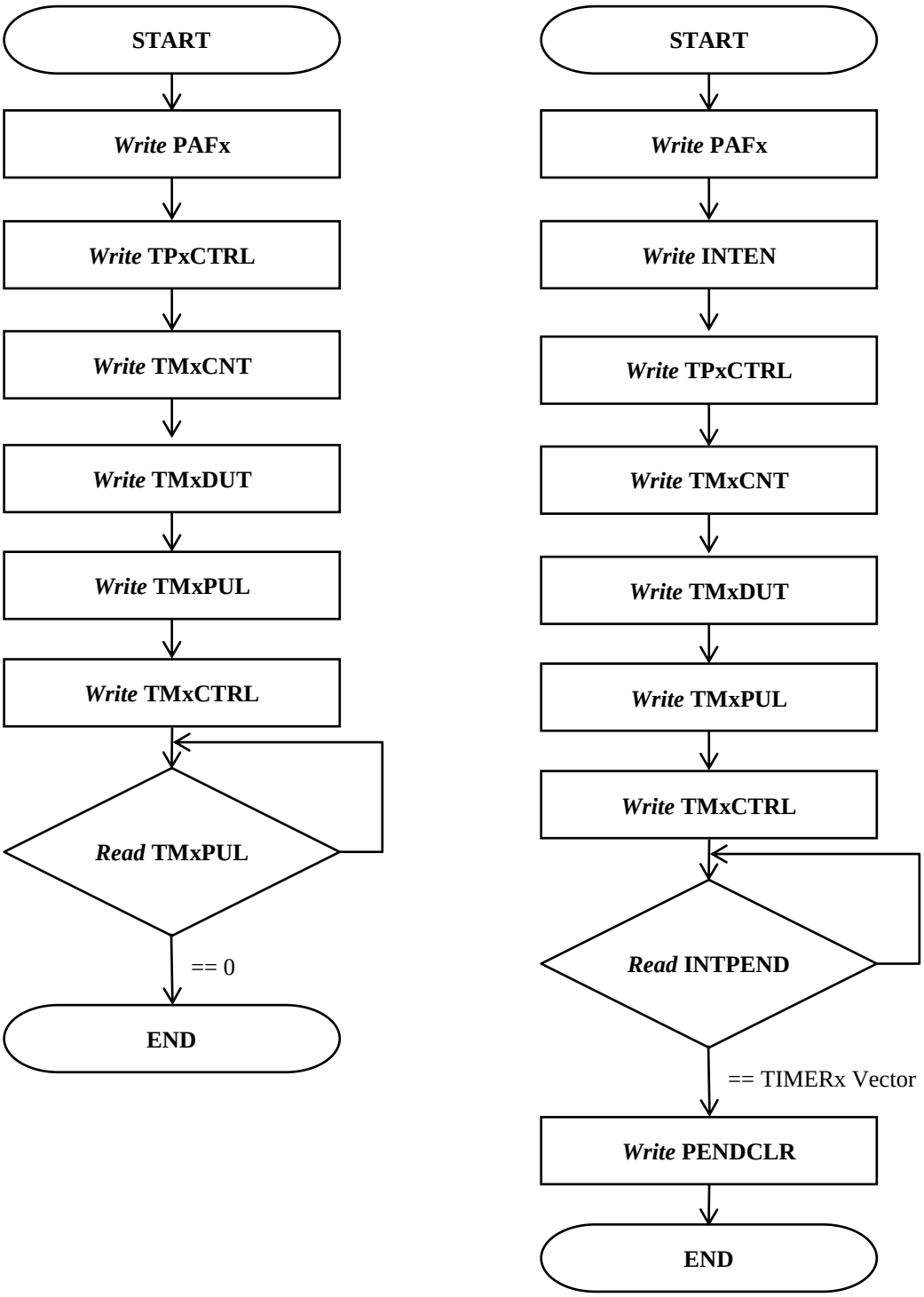
▶ 여기서는 Interrupt Pending Clear Register에 (INTNUM_TIMER2+0x20)을 대입하여 Interrupt Pending Register를 Clear한다. Interrupt Pending Register를 Clear하지 않으면 더 이상 Core에 Interrupt Request를 발생하지 않는다.

3 TIMER PWM Mode

3.1 Register Set

CANTUS의 TIMER를 PWM Mode로 사용하기 위해 다음과 같은 순서로 Register를 설정한다. PWM Mode에서는 설정한 PWM Pulse Number에서 Interrupt Request를 발생한다.

그림 3-1 TIMER PWM Mode Register Set



PAFx

해당 Port의 Pin을 TMO#으로 사용하기 위해 '00b'를 쓴다. Port의 Pin의 Alternation은 해당 bit(2)에 쓰는 값으로 설정된다.

표 3-1 Port Alternate Function Register 5

Group	Index	Pin	1st	2nd	3rd	4th(default)
			00	01	10	11
PAF5 0x80020034	0	34	TMO[0]	CAP[0]	SDCD_DATA[1]	P5.0
	1	35	TMO[1]	CAP[1]	SDCD_DATA[0]	P5.1
	2	36	TMO[2]	CAP[2]	I2S_LRCK	P5.2
	3	38	TMO[3]	CAP[3]	I2S_SCLK	P5.3
	4	39	TMO[4]	CAP[4]	I2S_MCLK	P5.4
	5	40	TMO[5]	CAP[5]	A[16]	P5.5
	6	41	TMO[6]	CAP[6]	A[17]	P5.6
	7	42	TMO[7]	CAP[7]	A[18]	P5.7

TMxCNT (Timer Counter / PWM Period Register)

PWM Mode에서는 Pin TMO#으로 출력할 신호의 Period를 설정한다. Period는 아래와 같은 식을 따른다. Read할 경우 Current Up-counter Value이다.

$$PWM\ \ Pulse\ \ Period = \frac{1}{Clock\ \ Source\ \ Frequency} \times \frac{1}{Pre-scaler\ \ Factor} \times (TMCNT)\ [\sec] \qquad \{Pre-scaler\ \ Factor \geq 3\}$$

$$PWM\ \ Pulse\ \ Period = \frac{1}{Clock\ \ Source\ \ Frequency} \times \frac{1}{Pre-scaler\ \ Factor} \times (TMCNT+1)\ [\sec] \qquad \{Pre-scaler\ \ Factor < 3\}$$

TMxDUT (Capture Counter / PWM Duty Register)

PWM Mode에서는 Pin TMO#으로 출력할 신호의 Duty를 설정한다. Duty는 TMxCTRL.PWML의 길이 이다.

표 3-2 Capture Counter / PWM Duty (TMxDUT)

Bit	R/W	Description	Default Value
31 : 0	R/W	(Capture mode) - Read : Result value of counting at the sampling period (PWM mode) - Read/Write : PWM Duty Value	0xFFFF_FFFF

*** PWM Duty : First Halt Duty of PWM Pulse

TMxPUL (PWM Pulse Count Register)

PWM Mode에서는 Pin TMO#으로 출력할 신호의 개수를 설정한다. PWM Mode가 동작하면 신호가 출력될 때 마다 1씩 감소하여 0에 다다르고, Interrupt Request를 발생한다. 0가 된 후 PWM 신호의 출력을 멈추기 위해서는 TMxCTRL.PWMO를 설정하여야 한다.

표 3-3 PWM Pulse Count (TMxPUL)

Bit	*R/W*	*Description*	*Default Value*
31 : 0	R/W	(PWM mode) - Read/Write : PWM Pulse Number Value (Output Compare Mode) - Read/Write : Output Comparer B Value - Supported in channel2 and channel3	0xFFFF_FFFF

TMxCTRL (Timer Control Register)

TMOD에 '01b'를 씀으로써 PWM Mode로 설정하고, PWMO와 PWML을 설정한다. 그리고 Pre-scaler의 분주를 설정하기 위해 PFSEL에 값을 쓴다. TMEN에 '1b'을 쓰면 TIMER가 동작 한다. PWMO를 설정하면 PWM의 출력이 TMxPUL에서 0까지 1회 출력된다.

3.2 Function Set 1

아래는 CANTUS의 TIMER6을 PWM Mode로 설정하여 Duty Rate 50%의 Pulse 3개를 출력하는 예이다. 먼저 PAF5를 TMO6으로 설정한다. 그리고 TPxCTRL을 통해 Pre-scaler를 Reset하고, TMxCNT를 통해 출력할 신호의 Period를 설정, TMxDUT를 통해 출력할 신호의 Duty를 설정, TMxPUL을 통해 출력할 신호의 개수를 설정, TMxCTRL을 통해 TIMER를 PWM Mode로, PWMO와 PWML을 설정, Pre-scaler를 1/32768로 설정한다. 설정이 끝나면 TMxCTRL의 TMEN을 통해 TIMER를 동작시킨다. 여기서 APB Clock은 48MHz이다.

```
main()
{
  …
  U32 pulse_period, pulse_cnt;
  U32 temp;

  *R_PAF5 = F_PAF5_6_TMO6;

  *R_TP6CTRL = F_TPCTRL_CNTRST;
  *R_TP6CTRL &= ~(F_TPCTRL_CNTRST);

  pulse_period = 732;
  *R_TM6CNT = pulse_period;
  *R_TM6DUT = pulse_period/2;

  pulse_cnt = 3;
  *R_TM6PUL = pulse_cnt;

  *R_TM6CTRL = F_TMCTRL_TMOD_PWM | F_TMCTRL_PWMO | F_TMCTRL_PWML_HIGH
  | F_TMCTRL_PFSEL_32768;

  *R_TM6CTRL |= F_TMCTRL_TMEN;

  while(*R_TM6PUL == 0);

  while(1)
  {
    temp = *R_TM6PUL;

    if(temp == 0)
    {
      debugprintf("PWM Pulse Generated = 0x%x \r\n",pulse_cnt);
      while(1);
    }
  }
  …
}
```

*R_PAF5

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

```
#define R_PAF5 ((volatile unsigned int*) 0x80020034)
#define F_PAF5_6_TMO6                    (0<<12)
```

로 정의되어 있다.

▶ 여기서는 Port Alternate Function Register 5에 F_PAF5_6_TMO6를 대입하여 Port 5의 6번 Pin을 TMO6로 설정 한다.

*R_TM6CNT

Timer Counter / PWM Period Register 는 SDK/Include/CANTUS/timer.h에

```
#define R_TM6CNT ((volatile unsigned int*)0x800210C8)
```

로 정의되어 있다.

▶ 여기서는 PWM Period Register에 pulse_period의 값을 쓴다. PWM Period Register에 쓴 값을 읽을 수 없으므로, pulse_period라는 변수를 사용한다. pulse_period의 값 732는 약 500ms를 뜻한다.

*R_TM6DUT

Capture Counter / PWM Duty Register는 SDK/Include/CANTUS/timer.h에

```
#define R_TM6DUT ((volatile unsigned int*)0x800210CC)
```

로 정의되어 있다.

▶ 여기서는 PWM Duty Register에 pulse_period의 1/2을 씀으로써 Duty Rate를 50%로 설정한다.

*R_TM6PUL

PWM Pulse Count Register는 SDK/Include/CANTUS/timer.h에

```
#define R_TM6PUL ((volatile unsigned int*)0x800210D0)
```

로 정의되어 있다.

▶ 여기서는 PWM Pulse Count Register에 pulse_cnt를 쓴다. PWM Pulse Count Register에 쓴 값을 읽을 수 없으므로, pulse_cnt라는 변수를 사용한다. TIMER가 동작하면 Pulse를 출력할 때마다 TM6PUL은 1씩 감소하여 0가 된다. TMxCTRL.PWMO Enable이면 0가 되었을 때 Pulse의 출력을 멈춘다.

***R_TM6CTRL**

Timer Control Register는 SDK/Include/CANTUS/timer.h에

#define R_TM6CTRL ((volatile unsigned int*)0x800210C4)

로 정의되어 있다.

▶ 여기서는 Timer Control Register 6에 (F_TMCTRL_TMOD_PWM | F_TMCTRL_PWMO | F_TMCTRL_PWML_HIGH | F_TMCTRL_PFSEL_32768)을 대입하여 Timer/Counter를 PWM Mode, PWM Output One Period Generation, PWM Output Start Level High, Pre-scale Factor 1/32768로 설정하고, F_TMCTRL_TMEN을 대입하여 Timer/Counter를 동작 시킨다.

3.3 Function Set 2

아래는 CANTUS의 TIMER6를 PWM Mode로 설정하여 Duty Rate 50%의 Pulse 3개를 출력하는 예이다. 먼저 PAF5를 TMO6으로 설정하고, TIMER Interrupt Enable을 설정 한다. 그리고 TPxCTRL을 통해 Pre-scaler를 Reset하고, TMxCNT를 통해 출력할 신호의 Period를 설정, TMxDUT를 통해 출력할 신호의 Duty를 설정, TMxPUL을 통해 출력할 신호의 개수를 설정, TMxCTRL을 통해 TIMER를 PWM Mode로, PWMO와 PWML을 설정, Pre-scaler를 1/32768로 설정한다. 설정이 끝나면 TMxCTRL의 TMEN을 통해 TIMER를 동작시킨다. 여기서 APB Clock은 48MHz이다.

```
main()
{
  …
  U32 pulse_period, pulse_cnt;
  U32 temp;

  *R_PAF5 = F_PAF5_6_TMO6;

  *R_INTEN |= (1<<INTNUM_TIMER6);

  *R_TP6CTRL = F_TPCTRL_CNTRST;
  *R_TP6CTRL &= ~(F_TPCTRL_CNTRST);

  pulse_period = 732;
  *R_TM6CNT = pulse_period;
  *R_TM6DUT = pulse_period/2;

  pulse_cnt = 3;
  *R_TM6PUL = pulse_cnt;

  *R_TM6CTRL = F_TMCTRL_TMOD_PWM | F_TMCTRL_PWMO | F_TMCTRL_PWML_HIGH
  | F_TMCTRL_PFSEL_32768;

  *R_TM6CTRL |= F_TMCTRL_TMEN;

  while(1)
  {
    temp = *R_INTPEND;

    if((temp & (1<<INTNUM_TIMER6)) == (1<<INTNUM_TIMER6))
    {
      *R_PENDCLR = (INTNUM_TIMER6+0x20);
      debugprintf("PWM Pulse Generated = 0x%x \r\n",pulse_cnt);
    }
  }
  …
}
```

***R_TM6PUL**

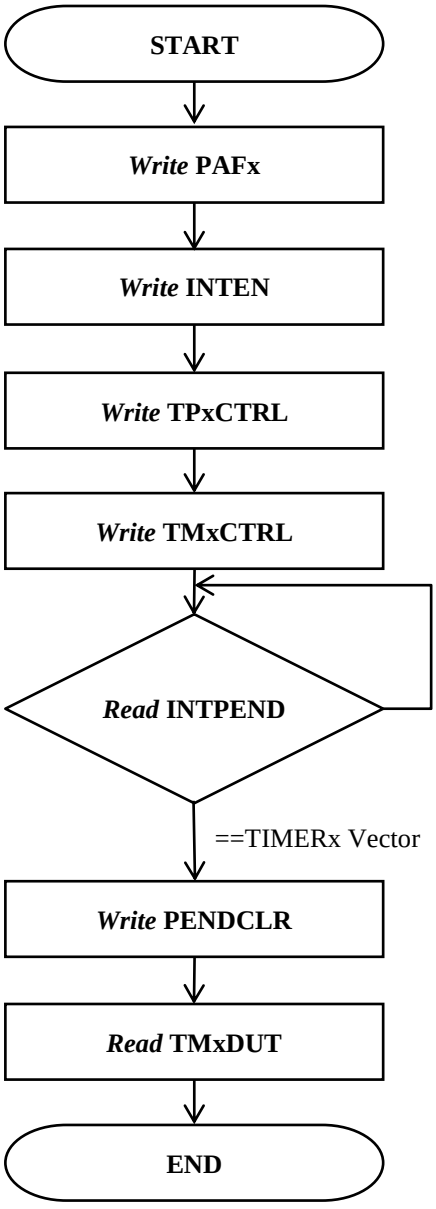
▶ 여기서는 PWM Pulse Count Register에 pulse_cnt를 쓴다. PWM Pulse Count Register에 쓴 값을 읽을 수 없으므로, pulse_cnt라는 변수를 사용한다. TIMER가 동작하면 Pulse를 출력할 때마다 TM3PUL은 1씩 감소하여 0가 되면 Interrupt Request를 발생한다. TMxCTRL.PWMO Enable이면 0가 되었을 때 Pulse의 출력을 멈춘다.

4 TIMER Capture Mode

4.1 Register Set

CANTUS의 TIMER를 Capture Mode로 사용하기 위해 다음과 같은 순서로 Register를 설정한다.

그림 4-1 TIMER Capture Mode Register Set



PAFx

해당 Port의 Pin을 CAP#으로 사용하기 위해 '01b'를 쓴다. Port의 Pin의 Alternation은 해당 bit(2)에 쓰는 값으로 설정된다.

TMxCTRL (Timer Control Register)

TMOD에 '1xb'를 씀으로써 Capture Mode로 설정하고, CAPMOD를 설정한다. Pre-scaler의 분주를 설정하기 위해 PFSEL에 값을 쓴다. TMEN에 '1b'을 쓰면 TIMER가 동작 한다.

TMxDUT (Capture Counter / PWM Duty Register)

Capture Mode에서는 Pin CAP#으로 입력된 신호를 TIMER Clock[3]으로 계수한 값을 나타낸다.

[3] 15bit Timer/Counter Pre-scaler가 TIMER로 공급하는 Clock

4.2 Function Set

아래는 CANTUS의 TIMER7를 Capture Mode로 설정하여 Rising to Rising Edge로 1Hz의 Period를 측정하는 예이다. 먼저 TIMER Interrupt Enable로 설정 한다. 그리고 TPxCTRL을 통해 Pre-scaler를 Reset하고, TMxCTRL을 통해 TIMER를 Capture Mode로, Capture Type을 Rising to Rising으로, Pre-scaler를 1/512로 설정한다. 설정이 끝나면 TMxCTRL의 TMEN을 통해 TIMER를 동작시킨다. Capture Mode에서 첫 번째 Capture 값은 유효하지 않다. 따라서 두 번째 Capture 되는 값을 사용하여야 한다. 여기서 APB Clock은 48MHz이고, INTNUM_TIMER7는 0x3D이다. Interrupt에 관련된 내용은 AN_0004B_INTERRUPT를 참조하라.

```
main()
{
  …

  *R_PAF5 = F_PAF5_7_CAP7;

  *R_INTEN = (1<<INTNUM_TIMER7_KEYSCAN);

  *R_TP7CTRL = F_TPCTRL_CNTRST;
  *R_TP7CTRL &= ~(F_TPCTRL_CNTRST);

  *R_TM7CTRL = F_TMCTRL_TMOD_CAP | F_TMCTRL_CAPMOD_RR | F_TMCTRL_PFSEL_512;
  *R_TM7CTRL |= F_TMCTRL_TMEN;

  U32 i,temp,period;
  i=0;

  while(1)
  {
    temp = *R_INTPEND;

    if(temp & (1<<INTNUM_TIMER7_KEYSCAN))
    {
      if(i!=0)
      {
        *R_TM7CTRL &= ~(F_TMCTRL_TMEN);

        i=0;
        period = *R_TM7DUT;
        debugprintf("Period = %d 0x%x \r\n", period, period);

        *R_TM7CTRL |= F_TMCTRL_TMEN;
        *R_PENDCLR = (INTNUM_TIMER7_KEYSCAN+0x20);
      }
      else
      {
        i++;
        *R_PENDCLR = (INTNUM_TIMER7_KEYSCAN+0x20);
      }
    }
  }
  …
}
```

*R_PAF5

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

```
#define R_PAF5 ((volatile unsigned int*) 0x80020034)
#define F_PAF5_7_CAP7                    (1<<14)
```

로 정의되어 있다.

▶ 여기서는 Port Alternate Function Register 5에 F_PAF5_7_CAP7를 대입하여 Port 5의 7번 Pin을 CAP7로 설정 한다.

*R_TM7CTRL

Timer Control Register는 SDK/Include/CANTUS/timer.h에

```
#define R_TM7CTRL ((volatile unsigned int*)0x800210E4)
```

로 정의되어 있다.

▶ 여기서는 Timer Control Register 2에 (`F_TMCTRL_TMOD_CAP | F_TMCTRL_CAPMOD_RR | F_TMCTRL_PFSEL_2`)을 대입하여 Timer/Counter를 Capture Mode, Rising to Rising Edge, Pre-scale Factor 1/2로 설정하고, F_TMCTRL_TMEN을 대입하여 Timer/Counter를 동작 시킨다.

*R_TM7DUT

Capture Counter / PWM Duty Register는 SDK/Include/CANTUS/timer.h에

```
#define R_TM7DUT ((volatile unsigned int*)0x800210EC)
```

로 정의되어 있다.

▶ 여기서는 Capture를 시작하고 두 번째 Capture 완료에서 Capture Counter Register를 읽어 Capture된 결과를 취한다. 이 값은 TIMER Clock으로 계수한 값이다.