12.

- Chapter30 : General Purpose Timer (GPT)
- Chapter24 : Enhanced Periodic Interrupt Timer (EPIT)

12.1 GPT

GPT 32 GPT GPT

- 32
-
- “force compare”
- low power debug
- capture compare rollover
- restart free-run

12.1.1

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32 kHz)
ipg_clk_highfreq	perclk_clk_root	High-frequency reference clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

GPT ipg_clk Peripheral ipg_clk_perclk gpt ipg_clk_32k ipg_clk_hig

Image not found or type unknown

4 ipg_clk_highfreq ipg_clk_32k ipg_clk C
mode ipg_clk ipg_clk

Image not found or type unknown

PERCLK_CLK_ROOT IPG_CLK_ROOT OSC IPG_CLK_ROOT CSCMR1[PERCLK_PODF]
CLK_ROOT CBCDR[IPG_PODF] 1 2 AHB_CLK_ROOT SYS PLL PFD2 CBCDR[AHB_
PFD2 396M PERCLK_CLK_ROOT 66M

PERCLK_CLK_ROOT = 396M / 3 / 2 = 66M

12.1.2

GPT_CR CLKSRC CLKSRC GPT EN = 0

1. GPT_CR EN = 0 GPT
2. GPT GPT_IR
3. / — GPT_CR OM1 OM2 OM3 0
4. - GPT_CR IM1 IM2
5. GPT_CR CLKSRC
6. GPT_CR SWR 1
7. GPT GPT_SR 1 0
8. GPT_CR ENMOD = 1 GPT 0x00000000
9. GPT_CR GPT EN = 1
10. GPT GPT_IR

12.1.3 GPT

① Restart mode

GPT GPT_CR 0x00000000 1 1

② free-run mode

0xffffffff 0x00000000

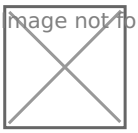
12.1.4 GPT

GPT GPT_CNT 32 EN = 1

- GPT EN = 0 EN 1 ENMOD GPT counter
- ENMOD 1 GPT EN = 1 0
- ENMOD 0 GPT EN = 1
- GPT STOP / WAIT GPT GPT ENMOD
- GPT OCR1 OCR2 OCR3 0x0 0xFFFF_FFFF
- GPT_CR SWR EN ENMOD STOPEN WAITEN DBGEN

12.1.5 GPT

/



12.1.6 GPT

GPT_CNT GPT_CNT 1

“ forced-compare ”

12.1.7 GPT

GPT 6

•

GPT 0xffffffff 0x00000000 GPT_IR ROVIE GPT_SR

• 1 2

“ ” IF2IE IF1IE GPT_IR IF2 IF1 GPT_SR

• 1 2 3

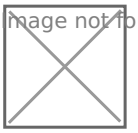
“ ” OF3IE OF2IE OF1IE GPT_IR OF3 OF2 OF1 GPT_S

cumulative cumulative

12.2 GPT

12.2.1 GPT Control Register (GPTx_CR)

GPT



bit31-29 FO3-1 0 1 OFn

bit28-26 bit25-23 bit22-20 OM3 OM2 OM1 000 001 010 0 011 1xx

bit19-18 bit17-16 IM2 IM1 00 01 10 11

bit15 SWR

bit10 EN_24M EN_24M EN_24M

bit9 FRR 0 restart 1 free-run

bit8-6 BLKSRC 000 001 Peripheral Clock (ipg_clk) 010 High Frequency Reference (ipg_clk_highfreq) 011 External Clock 100 Low Frequency Reference Clock (ipg_clk_32k) 101 C oscillator as Reference Clock (ipg_clk_24M)

bit5 STOPEN stop mode GPT

bit4 DOZEEN doze mode GPT

bit3 WAITEN wait mode GPT

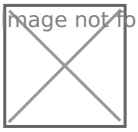
bit2 DBGEN debug mode GPT

bit1 ENMOD 0 GPT 1 GPT 0

bit0 EN GPT

12.2.2 GPT Prescaler Register (GPTx_PR)

GPT

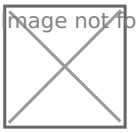


bit15-12 PRESCALER24M 24M crystal

bit12-0

12.2.3 GPT Status Register (GPTx_SR)

GPT



bit5 ROV

bit4-3 IF2 IF1

bit2-0 OF3-1

12.2.4 GPT Interrupt Register (GPTx_IR)

GPT

Image not found or type unknown

12.2.5 GPT Output Compare Register 1~3 (GPTx_OCR1~3)

GPT

GPTx_OCR1-3 3 restart mode channael1 GPT

Image not found or type unknown

12.2.6 GPT Input Capture Register 1~2 (GPTx_ICR1~2)

GPT

GPTx_ICR1-2

Image not found or type unknown

12.2.7 GPT Counter Register (GPTx_CNT)

GPT

GPT

Image not found or type unknown

12.3 GPT

12.3.1

gpt_poll_init gpt_poll_restart gpt_poll_init GPT gpt restart Peripheral C

gpt_poll_init **Git NoosProgramProject/(12_ /009_timer_gpt_poll/gpt.c**

```
void gpt_poll_init(GPT_Type *base)
{
    /* bit15 SWR, Software reset*/
    base->CR |= (1 << 15);
    /* Wait reset finished. */
    while((base->CR >> 15) & 0x1) {
    }

    /*
     *bit10: Enable 24 MHz clock input from crystal
     *bit9: 0 restart mode, 1 free-run mode:set 0
     *bit8-6: Clock Source select :001 Peripheral Clock (ipg_clk)
     *bit5: GPT Stop Mode enable
     *bit3: GPT Wait Mode enable.
     *bit1: GPT Enable Mode
     */
    base->CR = (1 << 6) | (1 << 5) | (1 << 3) | (1 << 1);

    /*
     *bit15-bit12:PRESCALER24M
     *bit11-0:PRESCALER
     */
    base->PR = 0;
}
```

gpt_poll_restart GPT comp

Git NoosProgramProject/(12_ /009_timer_gpt_poll/gpt.c

```
void gpt_poll_restart(GPT_Type *base, enum gpt_comp_channel chan, unsigned int us)
{
```

```
    base->OCR[chan] =  USEC_TO_COUNT(us);
    /* write 1 to clear int status to avoid unexpected compare event*/
    base->SR |= (1 << chan);
    /* enable interrupt*/
    base->IR |= (1 << chan);
    /* gpt enable*/
    base->CR |= (1 << 0);
    /*wait for compare flag set*/
    while(!((base->SR >> chan) & 0x1))
    ;
    /* gpt disable*/
    base->CR &= ~(1 << 0);
    /* disable interrupt*/
    base->IR &= ~(1 << chan);
    /* write 1 to clear int status*/
    base->SR |= (1 << chan);
}
```

gpt_poll_init GPT1 gpt_poll_restart(GPT1, OUT_COMP1, 1000000); 1s

Git NoosProgramProject/(12_ /009_timer_gpt_poll/main.c

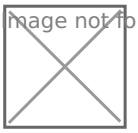
```
    gpt_poll_init(GPT1);
    while(1) {
        gpt_poll_restart(GPT1, OUT_COMP1, 1000000);
        GPIO5->DR &= ~(1<<3);//led on
        printf("led is on\r\n");
        gpt_poll_restart(GPT1, OUT_COMP1, 1000000);
        GPIO5->DR |= (1<<3);//led off
        printf("led is off\r\n");
    }
```

12.3.2 4-1.4

Git NoosProgramProject/(12_ /009_timer_gpt_poll)

12.3.3 4-1.4

1s



12.4 GPT

12.4.1 GPT1

chapter 3 Table3-1 gpt1

gic 32 gpt1 gic 55+32=87

12.4.2

12.4.2.1 gpt_init gpt

GPT gpt restart Peripheral Clock (ipg_clk) 66M 0 1

gpt_init **Git NoosProgramProject/(12_ /009_timer_gpt_int/gpt.c**

```
/* assume use ipc clk which is 66MHz, 1us against to 66 count */
#define USEC_TO_COUNT(us) (us * 66 - 1)

void gpt_init(GPT_Type *base, enum gpt_comp_channel chan, int us)
{
    /* bit15 SWR, Software reset*/
    base->CR |= (1 << 15);
    /* Wait reset finished. */
    while((base->CR >> 15) & 0x1) {
    }
```

```
    /*
     *bit10: Enable 24 MHz clock input from crystal
     *bit9: 0 restart mode, 1 free-run mode:set 0
     *bit8-6: Clock Source select :001 Peripheral Clock (ipg_clk)
     *bit5: GPT Stop Mode enable
     *bit3: GPT Wait Mode enable.
     *bit1: GPT Enable Mode
     */
    base->CR = (1 << 6) | (1 << 5) | (1 << 3) | (1 << 1);

    /*
     *bit15-bit12:PRESCALER24M
     *bit11-0:PRESCALER
     */
    base->PR = 0;

    /* GPTx_OCR1  bit31-0: Compare Value
     * When the counter value equals the COMP bit field value,
     * a compare event is generated on Output Compare Channel 1.
     */
    base->OCR[chan] = USEC_TO_COUNT(us);
}
```

12.4.2.2 gpt

.

Git NoosProgramProject/(12_ /09_timer_gpt_int/gpt.c

```
void gpt_enable_interrupt(GPT_Type *base, enum gpt_interrupt_bit bit, int on)
{
    if (on)
        base->IR |= (1 << bit);
    else
        base->IR &= ~(1 << bit);
}
```

12.4.2.3 gpt

Git NoosProgramProject/(12_ /009_timer_gpt_int/gpt.c

```
void gpt_run(GPT_Type *base, int on)
{
    /* bit0: GPT Enable */
    if (on)
        base->CR |= (1 << 0);
    else
        base->CR &= ~(1 << 0);
}
```

12.4.2.4

GPT1

Git NoosProgramProject/(12_ /009_timer_gpt_int/main.c

```
void GPT1_COMP1_handle_irq(void)
{
    static int on = 1;

    printf("GPT1 comp0 interrupt happened\r\n");
    /*
     * bit0: OF1 Output Compare 1 Flag
     * write 1 clear it */
    GPT1->SR |= 1;

    /* read GPIO5_DR to get GPIO5_IO01 status*/
    if(on) {
        /* led off, set GPIO5_DR to configure GPIO5_IO03 output 1 */
        GPIO5->DR |= (1<<3); //led off
        on = 0;
    } else {
        /* led on, set GPIO5_DR to configure GPIO5_IO03 output 0 */
        GPIO5->DR &= ~(1<<3); //led on
        on = 1;
    }
```

12.4.2.5

gpt_init GPT1 1s gic GPT1_IRQn GPT1 1 GPT1

Git NoosProgramProject/(12_ /009_timer_gpt_int/main.c

```
gpt_init(GPT1, OUT_COMP1, 1000000);// set 1s
request_irq(GPT1_IRQn, (irq_handler_t)GPT1_COMP1_handle_irq, NULL);
gic_enable_irq(GPT1_IRQn);
gpt_enable_interrupt(GPT1, IR_OF1IE, 1);
gpt_run(GPT1, 1);
```

12.4.3 4-1.4

Git NoosProgramProject/(12_ /009_timer_gpt_int)

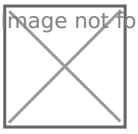
12.4.4 4-1.4

led

1s

12.5 EPIT

EPIT 32 EPIT



12.5.1 EPIT

EPIT

- 32
- 12

-
-
-

GPT ipg_clk ipg_clk 66M

12.5.2

EPIT set-and-forget free-running EPIT_CR [RLD]

① set-and-forget

EPIT_CR RLD 1 EPIT_LR EPIT_LR

② free-runnning

RLD 0000 0000h FFFF FFFFh E

12.5.3

EPIT 32 EPIT EPIT EN

- ENMOD RLD = 1 FFFF FFFFh RLD = 0 000h
- ENMOD

EPIT STOP / WAIT EPIT EPIT ENMOD

12.5.4

EPIT_EPITCMPR EPIT_EPITCNR OCIEN 1

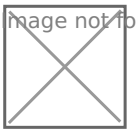
OM 0x0000_0000

EPIT IOVW EPIT

12.6 EPIT

12.6.1 Control register (EPITx_CR)

EPIT



bit25-24 CLKSRC 00 01 Peripheral clk 10 Hign-frequency 11 low-frequency

bit23-22 OM 00 01 10 0 11

bit21 STOPEN stop mode EPIT

bit19 WAITEN wait mode EPIT

bit18 DBGEN debug mode EPIT

bit17 IOVW

bit16 SWR

bit15-4 PRESCALER

bit3 RLD 0 0xFFFF_FFFF

bit2 OCIEN

bit1 ENMOD 0 1 RLD 1 0xFFFF_FFFF

bit0 EN EPIT

12.6.2 Status register (EPITx_SR)

EPIT

1

bit0 OCIF

12.6.3 Load register (EPITx_LR)

EPIT

EPIT_CR RLD EPIT 0 EPIT_LR IOVW RLD

12.6.3 Compare register (EPITx_CMPR)

EPIT

12.6.4 Counter register (EPITx_CNR)

EPIT

EPIT IOVW EPIT_IR

12.7 EPIT

12.7.1

12.7.1.1 epit_poll_init

epit_poll_init epit_poll_restart epit_poll_init EPIT EPIT set-and-forget enab
(ipg_clk) 66M 0 1 overwrite CMPR 0 OCIEN

epit_poll_init **Git NoosProgramProject/(12_ /009_timer_epit_poll/epit.c**

```
void epit_poll_init(EPIT_Type *base)
{
    base->CR = 0;

    /* software reset
     * bit16
     */
    base->CR |= (1 << 16);
    /* wait for software reset self clear*/
    while((base->CR) & (1 << 16))
        ;

    /*
     * EPIT_CR
     * bit21 stopen; bit19 waiten; bit18 debugen
     * bit17 overwrite enable; bit3 reload
     * bit2 compare interrupt enable; bit1 enable mode
     */
    base->CR |= (1 << 21) | (1 << 19) | (1 << 17) | (1 << 3) | (1 << 1);

    /*
     * EPIT_CR
     * bit25-24: 00 off, 01 peripheral clock(ipg clk), 10 high, 11 low
     * bit15-4: prescaler value, divide by n+1
     */
    base->CR &= ~((0x3 << 24) | (0xFFF << 4));
    base->CR |= (1 << 24);

    /* EPIT_CMPR: compare register */
    base->CMPR = 0;
    /* EPIT_LR: load register , assue use ipc clk 66MHz*/
    //base->LR = USEC_TO_COUNT(us);
```

```
    /* EPIT_CR bit2 OCIEN compare interrupt enable */
    base->CR |= (1 << 2);
}
```

12.7.1.2 epit_poll_restart

epit_poll_restart EPIT LR OCIF EPIT C

Git NoosProgramProject/(12_ /009_timer_epit_poll/epit.c

```
void epit_poll_restart(EPIT_Type *base, unsigned int us)
{
    epit_run(base, 0);
    /* EPIT_LR: load register , assue use ipc clk 66MHz*/
    base->LR = USEC_TO_COUNT(us);
    /* write 1 clear it, avoid it happened before */
    EPIT1->SR |= (1 << 0);
    epit_run(base, 1);
    /* wait compare event happened*/
    while(!(EPIT1->SR & 0x1))
    /* write 1 clear it */
    EPIT1->SR |= (1 << 0);
}
```

12.7.1.3 epit_run

epit_run EPIT

Git NoosProgramProject/(12_ /009_timer_epit_poll/epit.c

```
void epit_run(EPIT_Type *base, int on)
{
    /*  EPIT_CR bit0 EN */
    if (on)
        base->CR |= (1 << 0);
    else
        base->CR &= ~(1 << 0);
}
```

12.7.1.4

epit_poll_init EPIT1 epit_poll_restart(EPIT1, 1000000) 1s

Git NoosProgramProject/(12_ /009_timer_epit_poll/main.c

```
epit_poll_init(EPIT1);
while(1) {
    epit_poll_restart(EPIT1, 1000000);
    GPIO5->DR &= ~(1<<3); //led on
    printf("led is on\r\n");
    epit_poll_restart(EPIT1, 1000000);
    GPIO5->DR |= (1<<3); //led off
    printf("led is off\r\n");
}
```

12.7.2 4-1.4

Git NoosProgramProject/(12_ /009_timer_epit_poll)

12.7.3 4-1.4

led

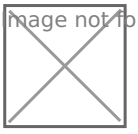
1s

”led is on” “led is off”

12.8 EPIT

12.8.1 EPIT1

chapter 3 Table3-1 EPIT1



gic 32 EPIT1 gic 56+32=88

12.8.1

12.8.1.1 EPIT1

epit_init EPIT1 EPIT EPIT set-and-forget enable mode Peripheral Clock (i

epit_init **Git NoosProgramProject/(12_ /009_timer_epit_int/epit.c**

```
/* assume use ipc clk which is 66MHz, 1us against to 66 count */
#define USEC_TO_COUNT(us) (us * 66 - 1)

void epit_init(EPIT_Type *base, unsigned int us)
{
    base->CR = 0;

    /* software reset
     * bit16
     */
    base->CR |= (1 << 16);
    /* wait for software reset self clear*/
    while((base->CR) & (1 << 16))
        ;

    /*
     * EPIT_CR
     * bit21 stopen; bit19 waiten; bit18 debugen
     * bit17 overwrite enable; bit3 reload
     * bit2 compare interrupt enable; bit1 enable mode
     */
    base->CR |= (1 << 21) | (1 << 19) | (1 << 3) | (1 << 1);

    /*
     * EPIT_CR
     * bit25-24: 00 off, 01 peripheral clock(ipg clk), 10 high, 11 low
     * bit15-4: prescaler value, divide by n+1
     */
    base->CR &= ~((0x3 << 24) | (0xFFF << 4));
    base->CR |= (1 << 24);
```

```
    /* EPIT_CMPR: compare register */
    base->CMPR = 0;
    /* EPIT_LR: load register , assue use ipc clk 66MHz*/
    base->LR = USEC_TO_COUNT(us);
}
```

12.8.1.2

epit_enable_interrupt

** Git NoosProgramProject/(12_ /009_timer_epit_int/epit.c **

```
void epit_enable_interrupt(EPIT_Type *base, int on)
{
    /*  EPIT_CR bit2 OCIEN compare interrupt enable */
    if (on)
        base->CR |= (1 << 2);
    else
        base->CR &= ~(1 << 2);
}
```

12.8.1.3 EPIT

Git NoosProgramProject/(12_ /009_timer_epit_int/epit.c

```
void epit_run(EPIT_Type *base, int on)
{
    /*  EPIT_CR bit0 EN */
    if (on)
        base->CR |= (1 << 0);
    else
        base->CR &= ~(1 << 0);
}
```

12.8.1.4

EPIT1_handle_irq 1

Git NoosProgramProject/(12_ /009_timer_epit_int/main.c

```
void EPIT1_handle_irq(void)
{
    static int on = 1;

    printf("EPIT1 interrupt happened\r\n");
    /* write 1 clear it */
    EPIT1->SR |= (1 << 0);

    /* read GPIO5_DR to get GPIO5_IO01 status*/
    if(on) {
        /* led off, set GPIO5_DR to configure GPIO5_IO03 output 1 */
        GPIO5->DR |= (1<<3); //led on
        on = 0;
    } else {
        /* led on, set GPIO5_DR to configure GPIO5_IO03 output 0 */
        GPIO5->DR &= ~(1<<3); //led off
        on = 1;
    }
}
```

12.8.1.5

epit_init(EPIT1, 1000000) 1s EPIT1_handle_irq gic_enable_irq(EPIT1_IRQn) EPIT1 gic 1) EPIT epit_run(EPIT1, 1) EPIT

Git NoosProgramProject/(12_ /009_timer_epit_int/main.c

```
epit_init(EPIT1, 1000000);// set 1s
request_irq(EPIT1_IRQn, (irq_handler_t)EPIT1_handle_irq, NULL);
gic_enable_irq(EPIT1_IRQn);
epit_enable_interrupt(EPIT1, 1);
epit_run(EPIT1, 1);
```

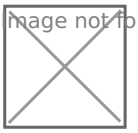
12.8.1.6 4-1.4

Git NoosProgramProject/(12_ /009_timer_epit_int)

12.8.1.7 3-1.4

led

1s



Revision #1
Created 3 March 2022 02:44:50 by
Updated 3 March 2022 02:45:09 by