Timers II

ECE 3710

When I was a kid my parents moved a lot, but I always found them.

- Rodney Dangerfield

integrated
timer

SysTick:

1. set *CURR*$_{(ENT)}$ to some initial value, *RE* $_{(LOAD)}$

2. count down from *CURR* $_{(ENT)}$ to final value, *0*

3. reset to *CURR* $_{(ENT)}$ to *RE* $_{(LOAD)}$ and set bit, *COUNT*

```
while(1)
  {
    for(CURR=RE;CURR>0;CURR--);

    COUNT = 1;
  }
```

note: we can do something else while uC is counting

Using SysTick:

1. stop timer
   (ENABLE=0)
2. set initial value
   (RELOAD=X)
3. clear current value
   (CURRENT=0)
4. set clock src
   (core clock: CLK_SRC=1)
5. enable/disable interrupts
   (INTEN=0)
6. start counting
   (ENABLE=1)

use a single STR

Address	31-24	23-17	16	15-3	2	1	0	Name
$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
$E000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
$E000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

ex: 50% duty cycle

(SysTick as src, `PD.0` as output)

```
; assume timer and port are setup
  ldr R1,=ST_CTRL_R
  ldr R3,=PD0_DATA_B
  mov R2,#0x1
; set PD0 = ~PD0
SW
  ldr R4,[R3] ;R4=PD0
  mvn R4,R4   ;R4=~R4
  str R4,[R3] ;PD0=~PD0
; wait X amount of time then flip PD0
WAIT ldr R0,[R1]
  cmp R2,R0,LSR #16 ;R0[16]?=1
  bne WAIT
  b SW
```

polling

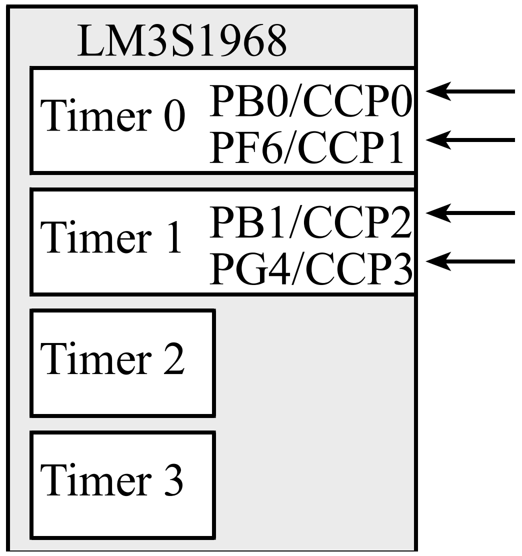
can we do anything here?

```
WAIT ldr R0,[R1]
    cmp R2,R0,LSR #16 ;R0[16]?=1
    bne WAIT
    b SW
```

polling

waste of uC CPU...

solution: interrupts (timer tells us when it rolls over)



General-Purpose Timers (GPTM):

1. four total (`TIMER0--3`)

2a. four 32-bit timers or

2b. eight 16-bit timers

3. run off system clock

also: integrated real-time clock, prescaler, pulse-width modulation, etc.

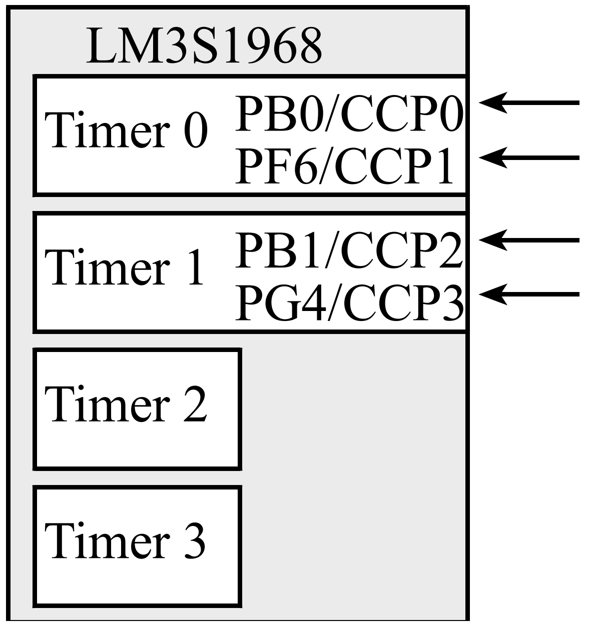
have to use
similar process for your board
(that's life...answer not always in book) $\longrightarrow$ from Stellaris LM3S1968 Microcontroller DATA SHEET

focus

max init. value
= 0xFFFF

GPTM:

1. 16-bit
2. periodic vs. one-shot
3. input edge {count vs. timing}



timer connected
to ext. source
(dreaded alt. func.)

Timer Register Map

base address

Timer0: 0x4003.0000
Timer1: 0x4003.1000
Timer2: 0x4003.2000
Timer3: 0x4003.3000

TIMERn has:

1. TimerA (16-bits)
2. TimerB (16-bits)

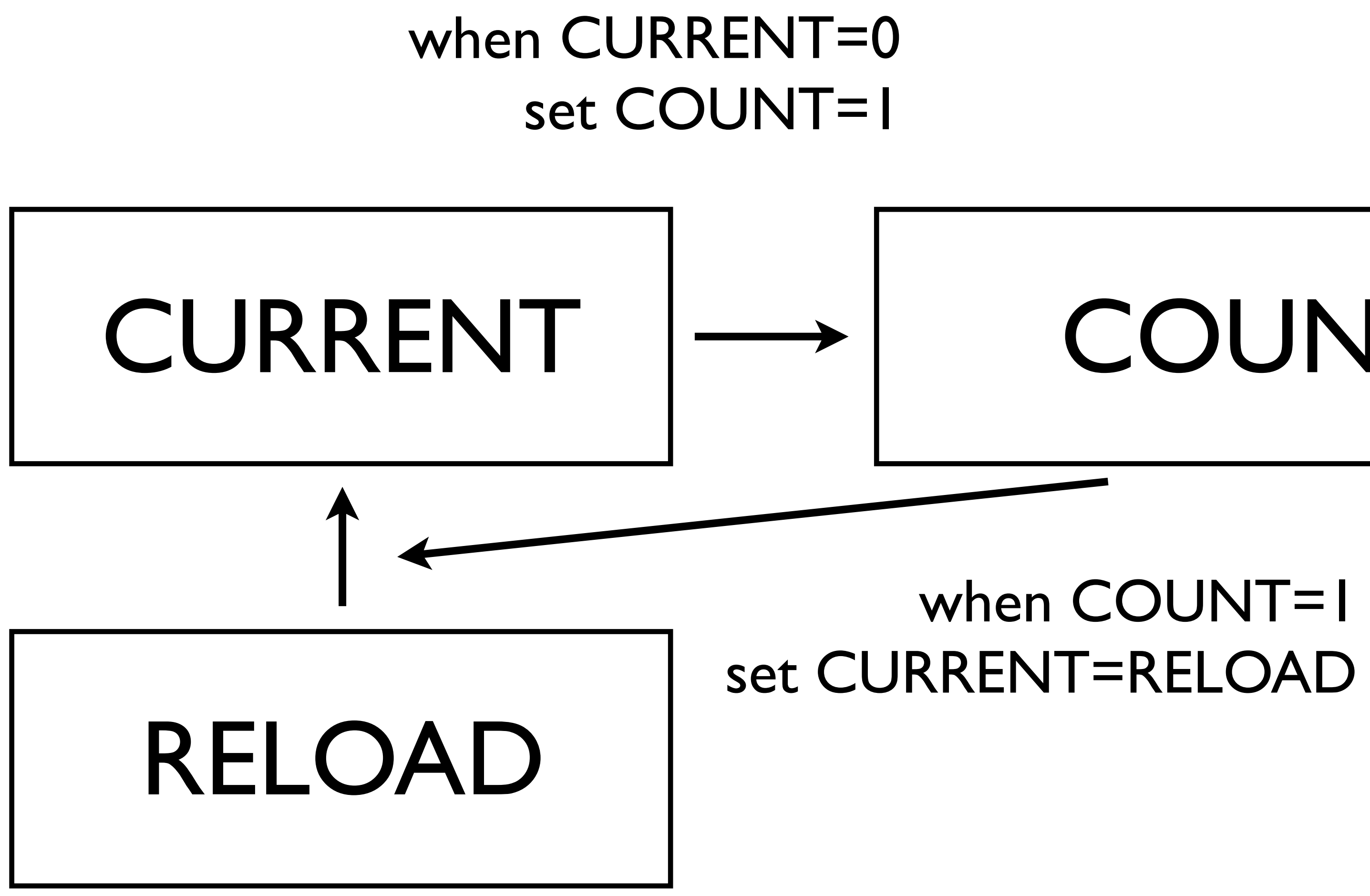
offset from base

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	345
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA Mode	346
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB Mode	348
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	350

periodic vs. one-shot

(nomenclature of SysTick)

periodic:



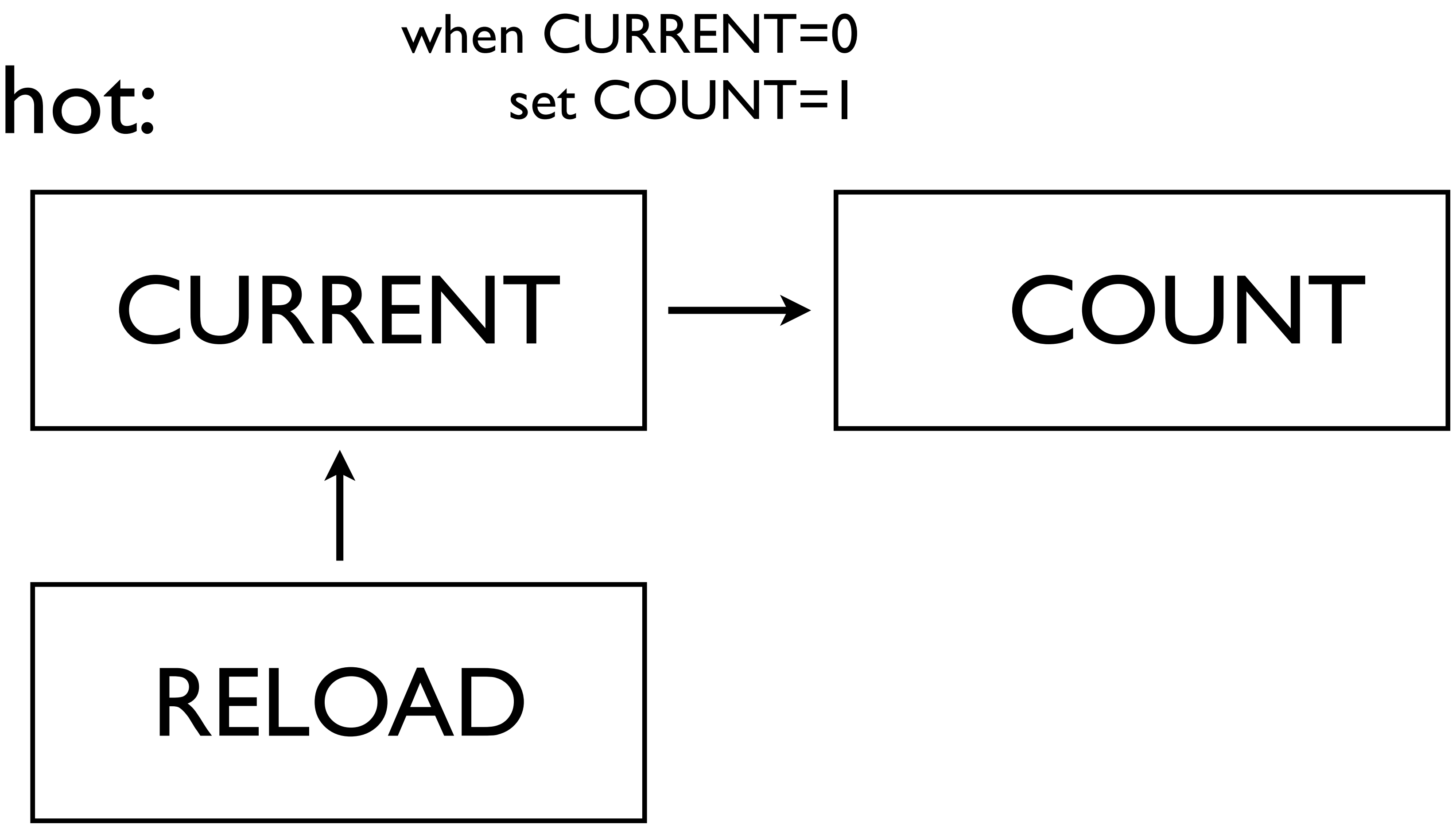
```
while(1)
  {
    for(CURR=RE;CURR>0;CURR--);

    COUNT = 1;
  }
```

periodic vs. one-shot

(nomenclature of SysTick)

one-shot:



```
COUNT = 0;
while(COUNT != 1)
   {
     for(CURR=RE;CURR>0;CURR--);

     COUNT = 1;
   }
```

note: timer stops have to re-enable

GPTM setup:

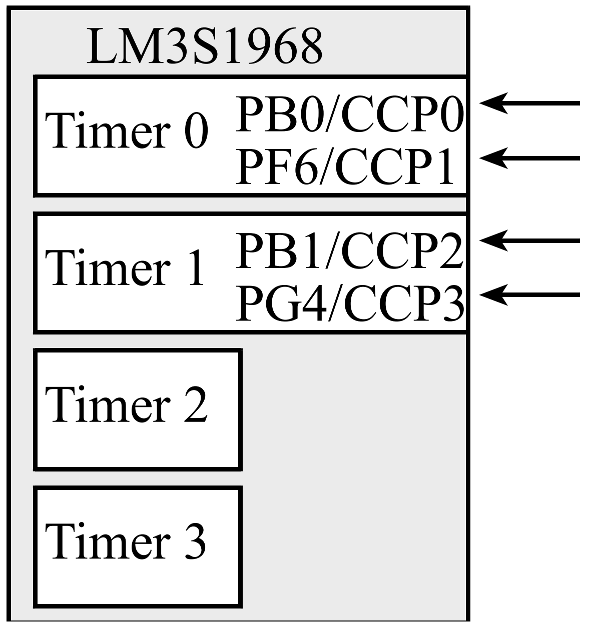
1. enable clock
(RCGC1) $\leftarrow$ affected register
2. stop timer
(GPTMCTL)
3. select 32-/16-bit timer
(GPTMCFG)
4. select periodic/one-shot
(GPTMTxMR) $\leftarrow$ x:=A or B
(TimerA or TimerB)
5. set initial value
(GPTMTxILR)
6. enable timer & start counting
(GPTMCTL)

note: have left out prescale and interrupts
(zero/off by default)

example: TIMER0, TimerA (TM0A), One-shot

0. constant initialisation:

```
; clock and port base address
RCGC1 EQU 0x400FE104
; Timer0.A is on pin PB0
PB EQU 0x40005000
; timer zero base address
TM0 EQU 0x40030000
```



GPIO Port A: 0x4000.4000
GPIO Port B: 0x4000.5000
GPIO Port C: 0x4000.6000
GPIO Port D: 0x4000.7000
GPIO Port E: 0x4002.4000
GPIO Port F: 0x4002.5000
GPIO Port G: 0x4002.6000
GPIO Port H: 0x4002.7000
p295

Timer0: 0x4003.0000
Timer1: 0x4003.1000
Timer2: 0x4003.2000
Timer3: 0x4003.3000
p343

let's do this right and use offset in LDR

(tired of defining all of those variables)

example: TIMER0, TimerA (TM0A), One-shot

1. enable clock

```
ldr R1,=RCGC1
;timer0 enable: set bit 16
;0x10000=0b10000000000000000
mov R0,#0x10000
str R0,[R1]
```

Run Mode Clock Gating Control Register 1 (RCGC1)

Base 0x400F.E000
Offset 0x104
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			reserved			COMP2	COMP1	COMP0			reserved		TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

set to 1

p218

example: TIMER0, TimerA (TM0A), One-shot

2. stop timer

```
ldr R1,=TM0 ;timer0 base
mov R0,#0x0
str R0,[R1,#0xC]
```

GPTMCTL

Offset 0x00C
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT		TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

set to 0

p350

example: TIMER0, TimerA (TM0A), One-shot

3. select 16-bit mode

```
ldr R1,=TM0 ;timer0 base
mov R0,#0x4
str R0,[R1,#0x0]
```

GPTMCFG

Offset 0x000
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													GPTMCFG		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x0	32-bit timer configuration.
0x1	32-bit real-time clock (RTC) counter configuration.
0x2	Reserved
0x3	Reserved
0x4-0x7	16-bit timer configuration, function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**.

set to 0x4

p345

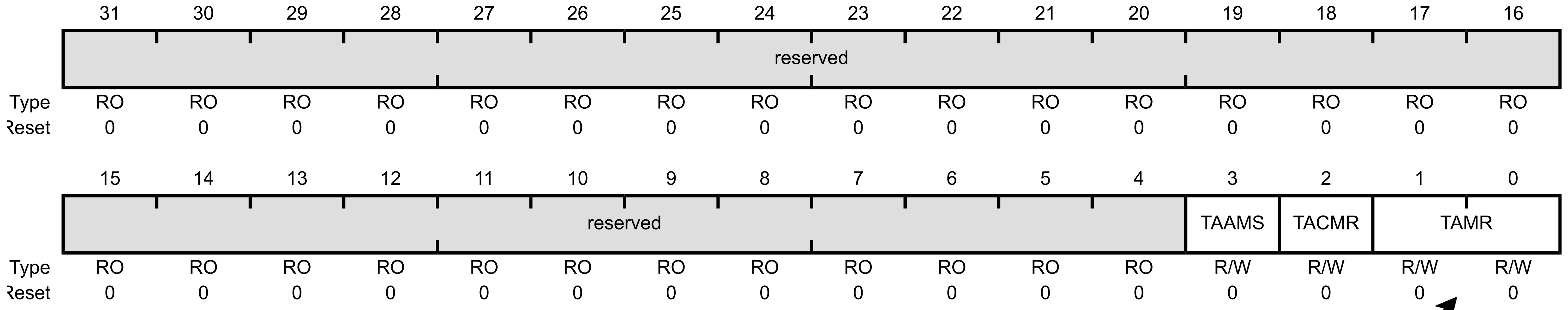
example: TIMER0, TimerA (TM0A), One-shot

4. select one-shot

```
ldr R1,=TM0 ;timer0 base
mov R0,#0x1
str R0,[R1,#0x4]
```

GPTMTAMR

Offset 0x004
Type R/W, reset 0x0000.0000



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TAAMS	TACMR	TAMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x0	Reserved
0x1	One-Shot Timer mode
0x2	Periodic Timer mode
0x3	Capture mode

set to 0x1

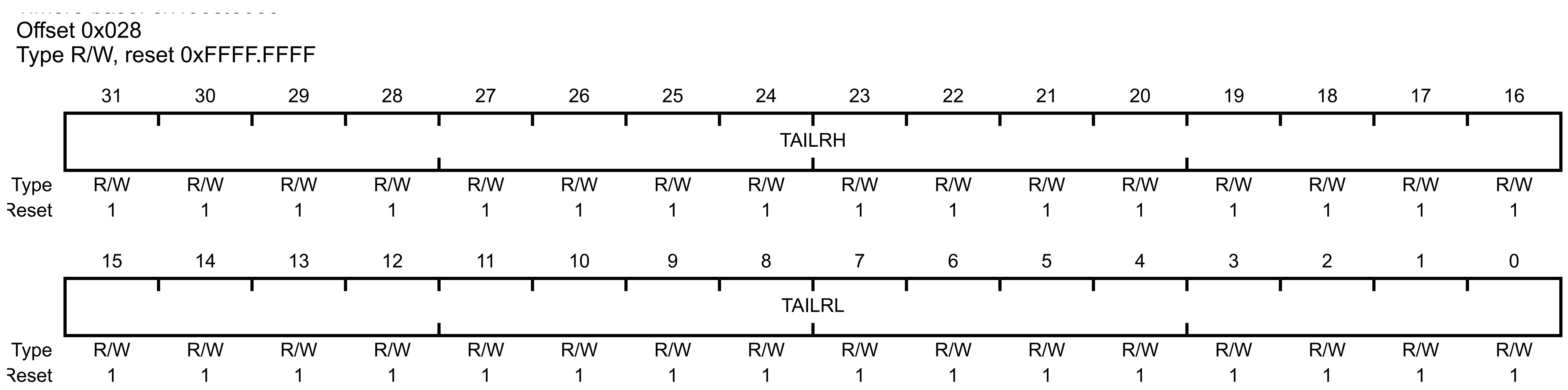
p346

example: TIMER0, TimerA (TM0A), One-shot

5. set initial value

```
ldr R1,=TM0 ;timer0 base
mov R0,#0xF
str R0,[R1,#0x28]
```

GPTMTAILR



Offset 0x028
Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAILRH															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAILRL															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

can only use lower 16 bits

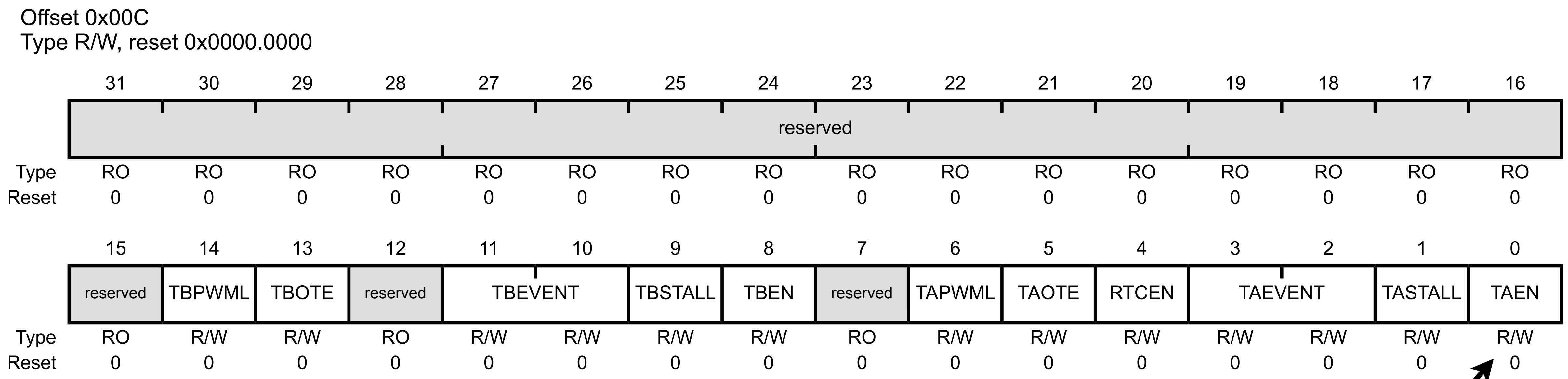
p359

example: TIMER0, TimerA (TM0A), One-shot

6. enable counter and go

```
ldr R1,=TM0 ;timer0 base
mov R0,#0x1
str R0,[R1,#0xC]
```

GPTMCTL



Offset 0x00C
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT		TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

set to 1

p350

offset = 0x48

using GPTM:

1. current value of timer in `GPTMTxR`
2. upon 1->0 transition, `TxTORIS` of `GPTMRIS` (offset = 0x1C) is set
3. to clear `TxTORIS`, set `TxTOCINT` of `GPTMICR` (offset = 0x24)

we poll this

after we notice timer expired

GPTMRIS
offset 0x1C
p355

note: offset 0x1D

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					CBERIS	CBMRIS	TBTORIS	reserved				RTCRIS	CAERIS	CAMRIS	TATORIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

using GPTM: polling

```
;assume timer is configured as above
  ldr R1,=TM0
wait
  ldr R0,[R1,#0x1C] ;timer status
                    ;(TxTORIS of GPTMRIS)
                    ;if timer expired
                    ;R0[0]=1
  ands R0,#0x1      ;sets Z=1 if R0==0
  beq wait          ;Z=1 => then equal
```

saves us from using AND+CMP

(have to mask GPTMRIS)

Q: setup+polling routine for Timer1B

ex: 500ms delay with SysClk = 8 MHz

re: the homework: resist the urge...

max delay w/16-bit counter
so we'll need to loop

strategy:

0. 1/8e6*2^16 ~ 8 ms << 500 ms
1. create 5ms delay
2. loop 100 times

use periodic timer
and counter overflows

ex: 500ms delay with SysClk = 8 MHz

strategy:

1. get 5 ms
2. run 100 times

$$(INITIAL + 1) \times \frac{1}{8e6}$$

$$n \times 0.125\mu\text{s} = 0.005 \rightarrow n = 40,000$$

$$INITIAL + 1 = 40,000 \rightarrow INITIAL = 39,999$$

$\swarrow$

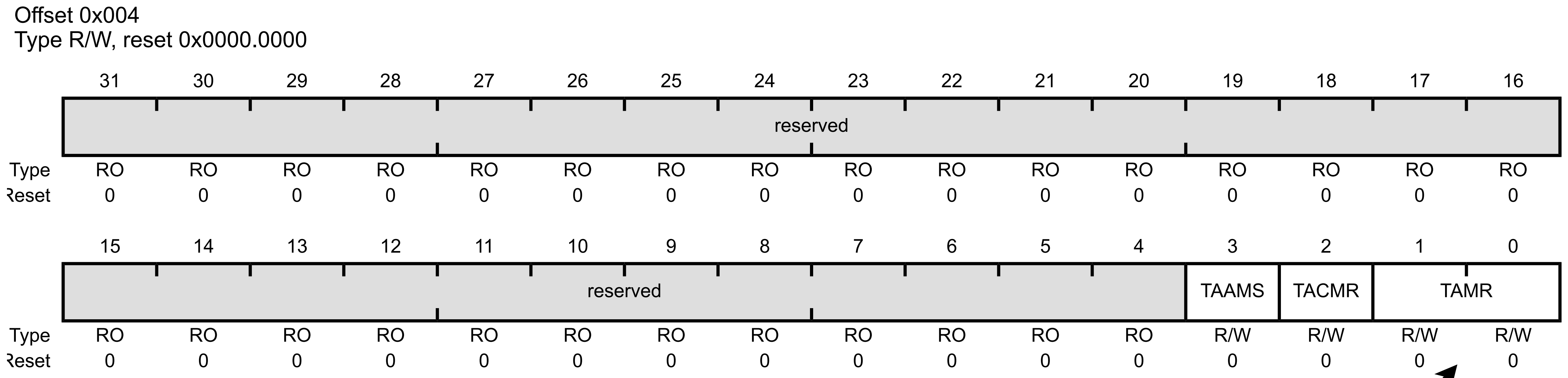
0x9C3F

ex: 500ms delay with SysClk = 8 MHz

4. select periodic

```
ldr R1,=TM0 ;timer0 base
mov R0,#0x2
str R0,[R1,#0x4]
```

GPTMTAMR



Offset 0x004
Type R/W, reset 0x0000.0000
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
reserved
Type RO RO RO RO RO RO RO RO RO RO RO RO RO RO RO RO
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
reserved
TAAMS
TACMR
TAMR
Type RO RO RO RO RO RO RO RO RO RO RO RO R/W R/W R/W R/W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0x0	Reserved
0x1	One-Shot Timer mode
0x2	Periodic Timer mode
0x3	Capture mode

set to 0x2

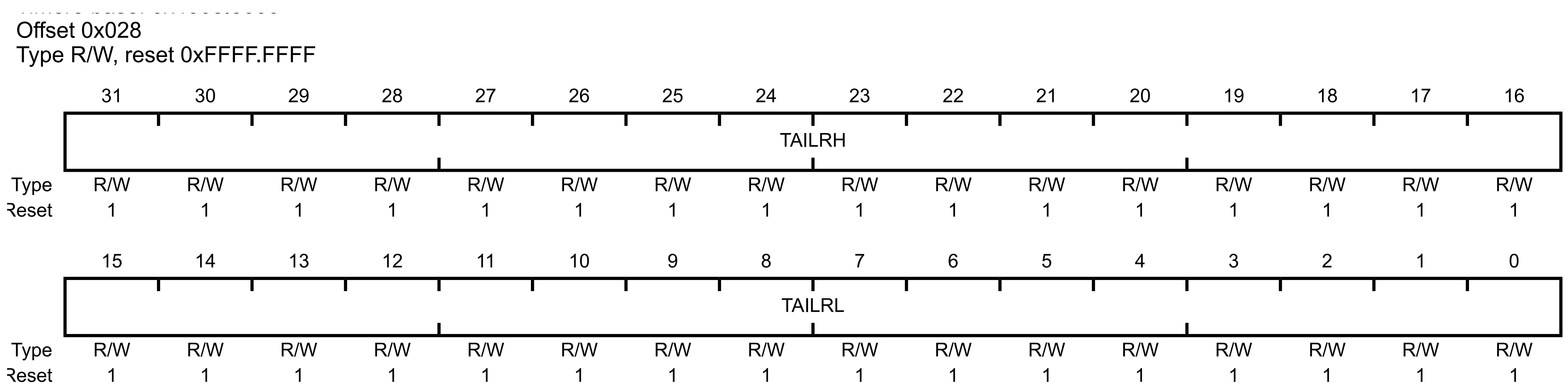
p346

ex: 500ms delay with SysClk = 8 MHz

5. set initial value

```
ldr R1,=TM0 ;timer0 base
ldr R0,=0x9C3F
str R0,[R1,#0x28]
```

GPTMTAILR



Offset 0x028
Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAILRH															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAILRL															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

can only use lower 16 bits

p359

ex: 500ms delay with SysClk = 8 MHz

```
;assume timer is configured as above
  ldr R1,=TM0
  mov R2,#0 ;use R2 as counter
  mov R3,#1 ;for reseting expiry flag
  mov R4,#0 ;for stopping counter
wait
  <as above>
  beq wait
  ;clear timer expiry flag
  str R3,[R1,#0x24]
  add R2,#1 ;increment counter
  cmp R2,#100
  bne wait
  ;stop timer
  str R4,[R1,#0xC]
  ;clear timer expiry flag
  str R3,[R1,#0x24]
```

note: with such large delay, can ignore the odd machine cycle