an uncomfortable announcement...

(I assure you, for me, too)

exam one:

1. 20131009

2. everything up to and including timers

Timers IV

ECE 3710

It's a small world, but I wouldn't want to have to paint it.

- Steven Wright

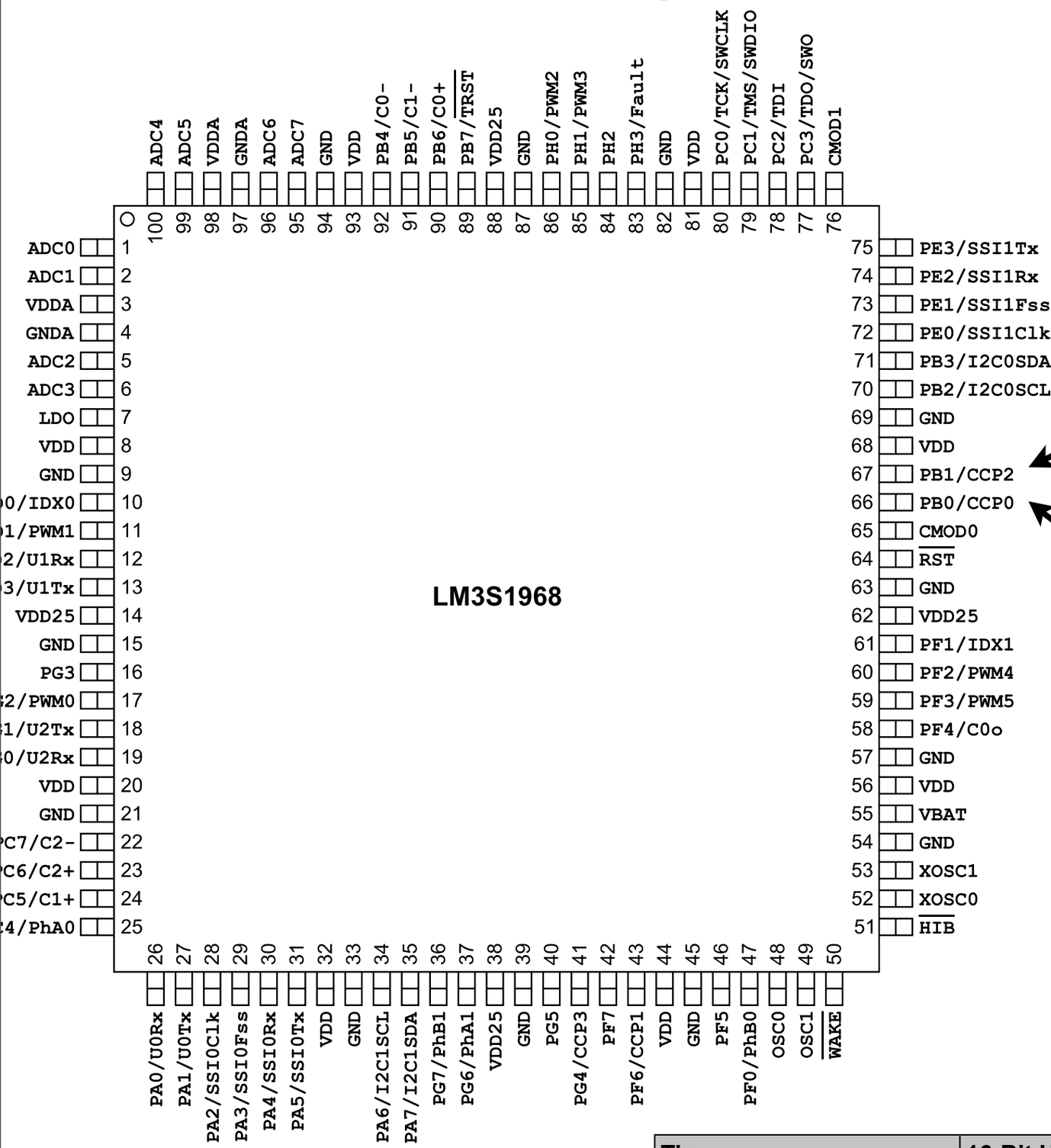
this is what
gets counted

timer (counter) programming:

1. internal (internal clock)
2. external (events)

will use these
interchangeably

Q: which ports to connect external to?



Timer1.A

Timer0.A

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	TimerA	CCP0	-
	TimerB	-	CCP1
Timer 1	TimerA	CCP2	-
	TimerB	-	CCP3

external timer modes

count mode

when high2low/low2high transitions on timer port:

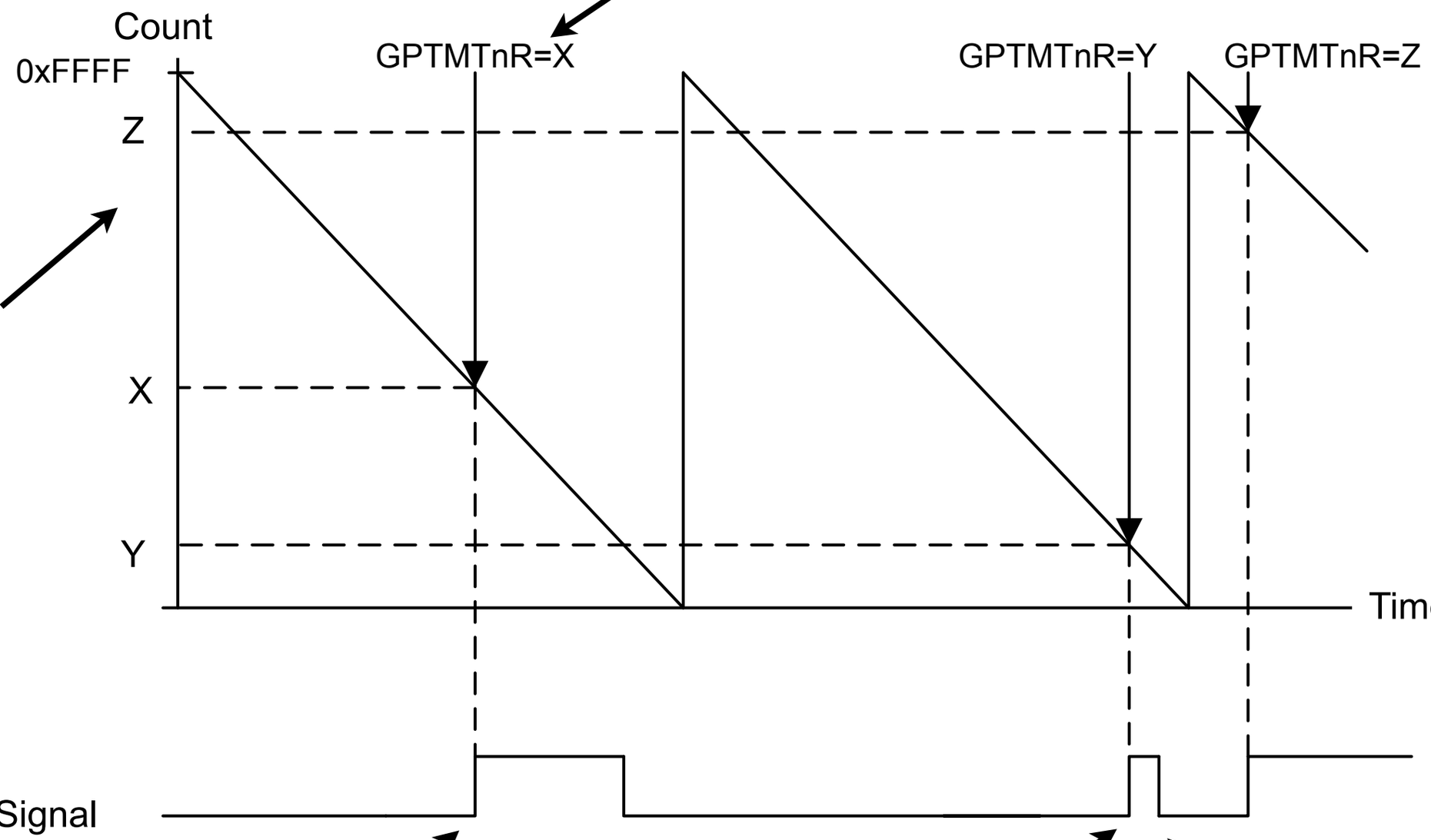
1. decrement
2. report timer value

timing mode

timing mode

1. set initial value

4. `CnERIS` asserted



2. timer starts counting down

3. transition puts counter value into TxR
(can be rising, falling, both)

positive transition

negative transition

GPTM setup (timing):

1. enable clock: GPTM and GPIO (RCGC1)
2. enable alt. functionality and port
3. stop timer (GPTMCTL)
4. select 32-/16-bit timer (GPTMCFG)
5. select capture & timer mode (GPTMTxMR)
6. selection edge type (GPTMCTL)
7. set initial value (GPTMTxILR)
8. enable timer & start counting (GPTMCTL)

affected register

x:=A or B (TimerA or TimerB)

or just use single instr.

restarts when zero reached

note: have left out interrupts (off by default)

TIMER0, TimerA (TM0A)

2. enable alt func. and pin

```
ldr R1,=PB
;Timer0.A is on pin PB0
mov R0,#0x1
str R0,[R1,#0x420] ;alt. func.
str R0,[R1,#0x51C] ;pin
```

GPIOAFSEL

Offset 0x420
Type R/W, reset -

	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								AFSEL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

0 TimerA is disabled.

1 TimerA is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.

set to 1

p307

TIMER0, TimerA (TM0A)

5. select capture & timer mode

```
ldr R1,=TM0
mov R0,#0x7 ;0b111
str R0,[R1,#0x4]
```

GPTMTxMR

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x004
Type R/W, reset 0x0000.0000

Value	Description
0	Edge-Count mode
1	Edge-Time mode

set to 1

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

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

set to 3

p346

TIMER0, TimerA (TM0A)

6. selection edge type

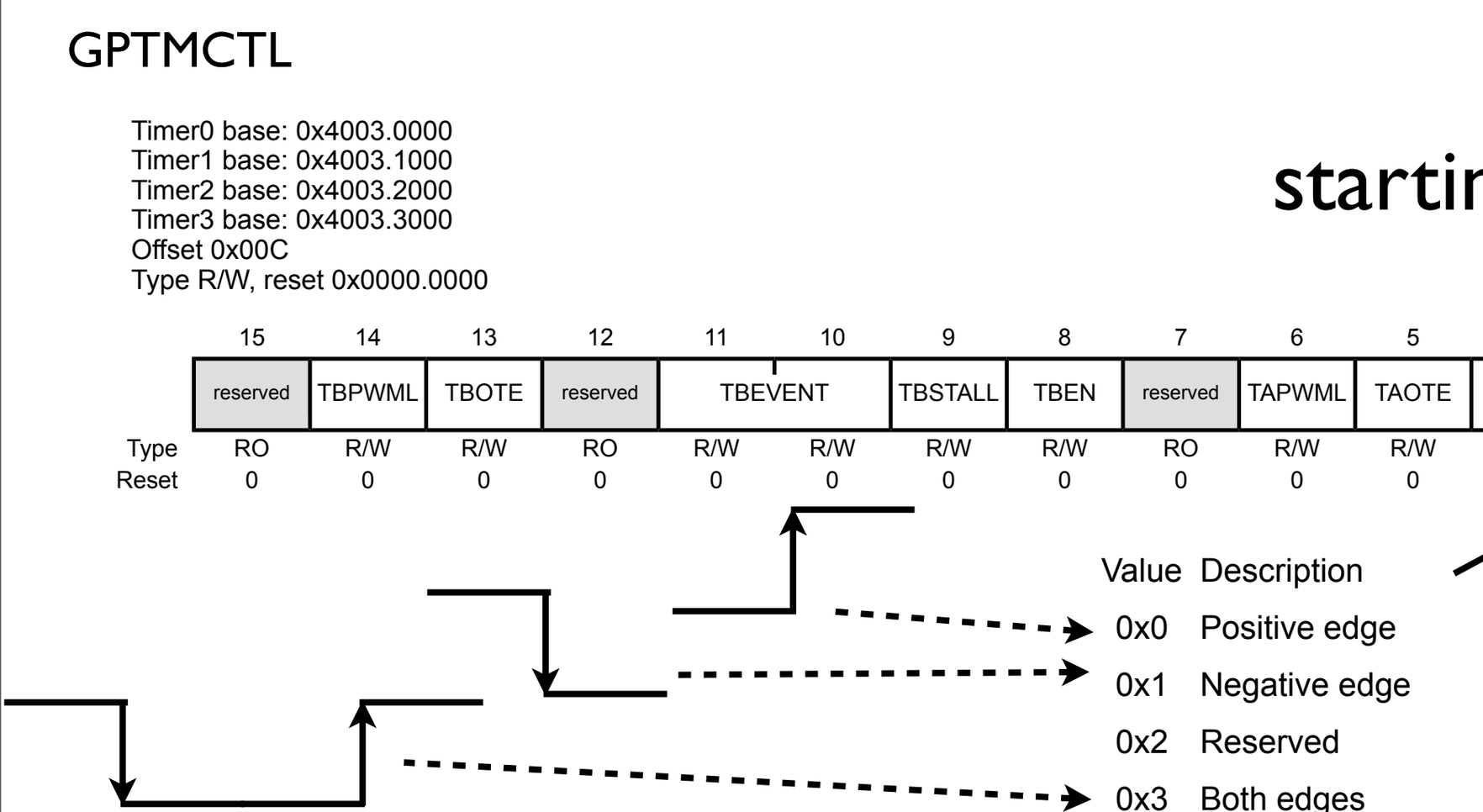
```
ldr R1,=TM0
mov R0,#0x4 ;0b0100
str R0,[R1,#0xC]
```

GPTMCTL

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x00C
Type R/W, reset 0x0000.0000

starting here

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

TIMER0, TimerA (TM0A)

8. enable timer & start counting

```
ldr R1,=TM0 ;rmw cycle
ldr R0,[R1,#0xC] ;read
orr R0,#0x1 ;modify
str R0,[R1,#0xC] ;write
```

GPTMCTL

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x00C
Type R/W, reset 0x0000.0000

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

need read-modify-write cycle to preserve

set to 1

p350

note:

`CnERIS` of `GPTMRIS` asserted after transition

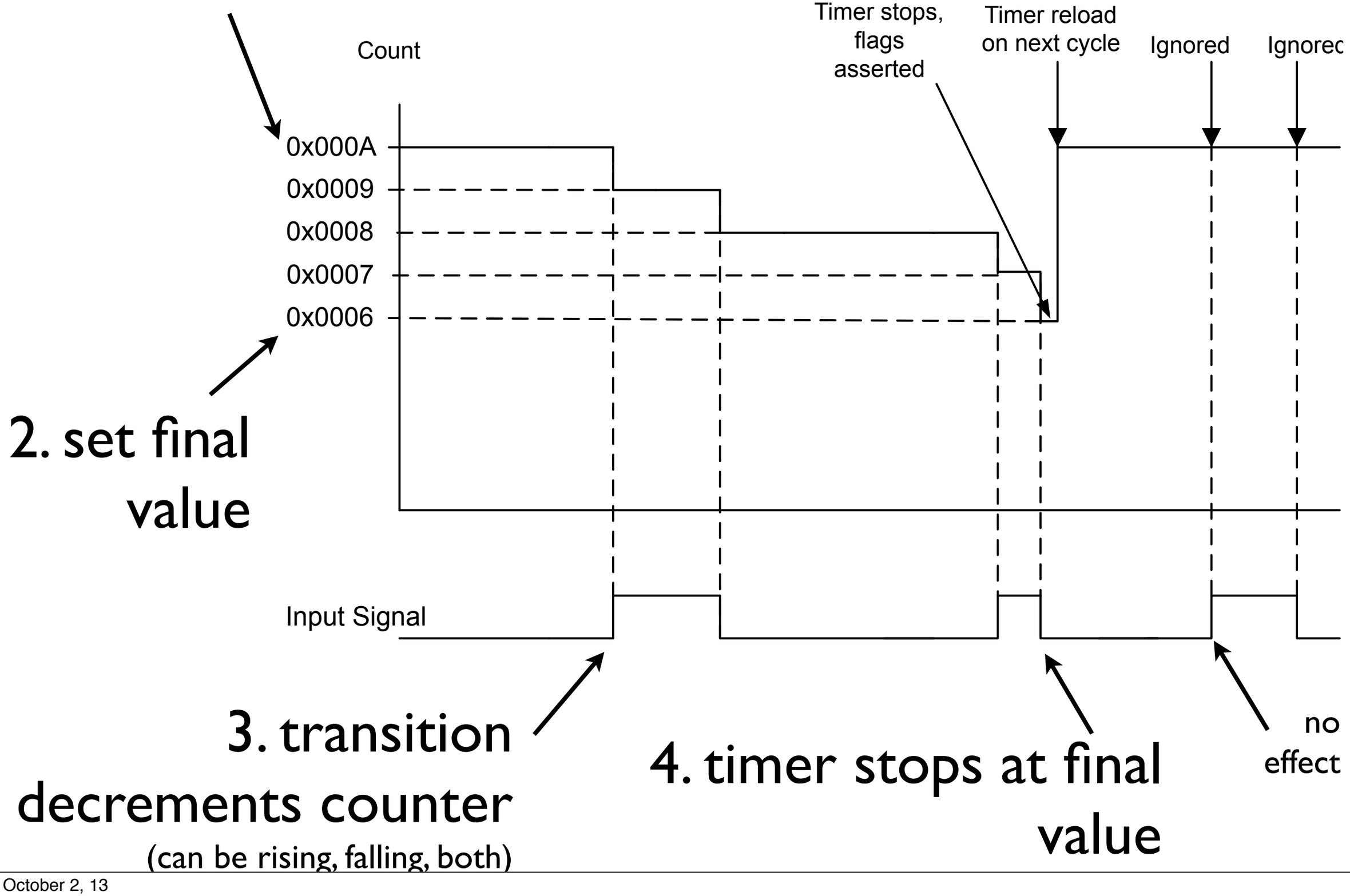
to clear: write one to `CnECINT` of `GPTMICR`

and now for *count* mode:

still with me?

count mode

1. set initial value



GPTM setup (count):

1. enable clock: GPTM and GPIO (RCGC1)
2. enable alt. functionality and port
3. stop timer (GPTMCTL)
4. select 32-/16-bit timer (GPTMCFG)
5. select capture & count mode (GPTMTxMR)
6. selection edge type (GPTMCTL)
7. set initial value (GPTMTxILR)
8. set final value (GPTMTnMATCHR)
9. enable timer & start counting (GPTMCTL)

affected register

x:=A or B (TimerA or TimerB)

stops when final value reached (must be restarted)

`CnMRIS` of `GPTMRIS` asserted (must be reset)

note: have left out interrupts (off by default)

upon expiration

bits asserted for...

GPTM Raw Interrupt Status (GPTMRIS)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x01C
Type RO, 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					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

timing mode

count mode

timer mode

upon expiration

assert to clear...

GPTM Interrupt Clear (GPTMICR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x024
Type W1C, 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					CBECINT	CBMCINT	TBTOCINT	reserved				RTCCINT	CAECINT	CAMCINT	TATOCINT
Type	RO	RO	RO	RO	RO	W1C	W1C	W1C	RO	RO	RO	RO	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

timing mode

count mode

timer mode

ex: read sensor

scenario: over the course of 10ms, a thermostat outputs temp by number of high-to-low transitions

(e.g. 10 transitions = 10 degrees F)

to save power: the thermostat begins outputting temp when it receives low-to-high transition

(connected to uC on PD.0)

ex: read sensor

we need:

1. display temp on LEDs connected to PA (binary temp display)
2. tell thermostat to output temp (PD.0)
3. capture transitions from thermostat (PB.0)

ex: read sensor

```
; assume PA/PD.0 as output
; PB.0 as timer in count mode
ldr R1,=TM0
ldr R3,=PA_DATA_R
ldr R5,=PD0_DATA_B
mov R2,#0x64 ;range of sensor is 0--100 C
mov R4,#0x1  ;for writing one
mov R6,#0x0  ;for writing zero

GETTEMP
  ldr R0,[R1,#0xC]  ;1. reload timer by stopping it
  and R0,#0xFE      ;   just turn off, preserve all else
  strb R0,[R1,#0xC] ;   modify first byte, only
  orr R0,#0x1       ;2. restart timer
  str R0,[R1,#0xC]
  str R4,[R5]       ;3. tell thermostat to send temp (l2h)
  bl DELAY10ms      ;   wait 10ms
  ldr R0,[R1,#0x48] ;4. current value of counter (100-temp)
  str R6,[R5]       ;5. go low so have low2high for sensor
```

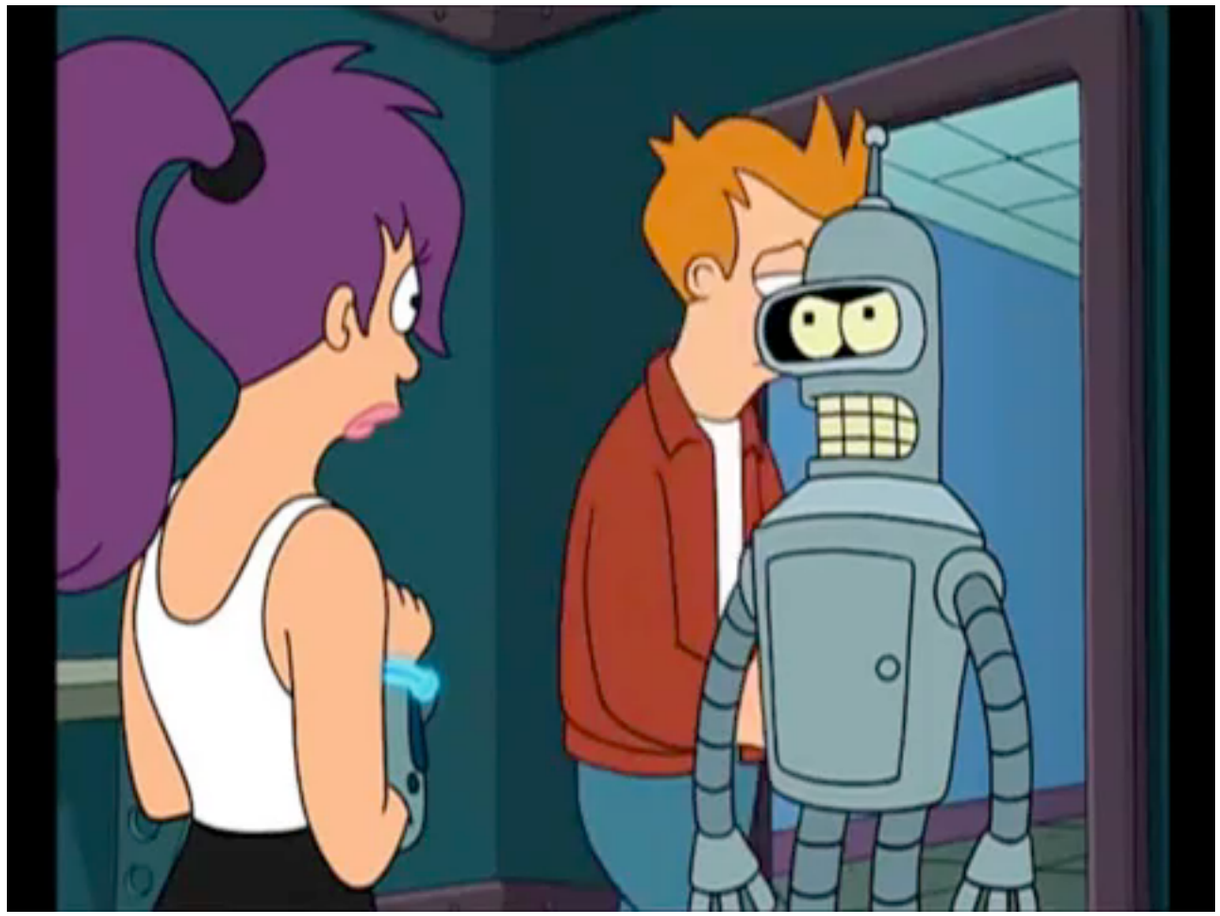
ex: read sensor

```
; assume PA/PD.0 as output
; PB.0 as timer in count mode
ldr R1,=TM0
ldr R3,=PA_DATA_R
ldr R5,=PD0_DATA_B
mov R2,#0x64 ;range of sensor is 0--100 C
mov R4,#0x1  ;for writing one
mov R6,#0x0  ;for writing zero

GETTEMP
  ...
  ldr R0,[R1,#0x48] ;4. current value of counter (100-temp)
  str R6,[R5]       ;5. go low so have low2high for sensor
  sub R0,R2,R0      ;6. output temp
                    ;   e.g. temp = 10: count=100-10=90
                    ;   => temp = 100-count
  str R0,[R3]       ;   send temp to LEDs
  b GETTEMP         ;7. do it again
```

timers:

remember, it won't always be so bad...



Bender = professor (writing exam)?