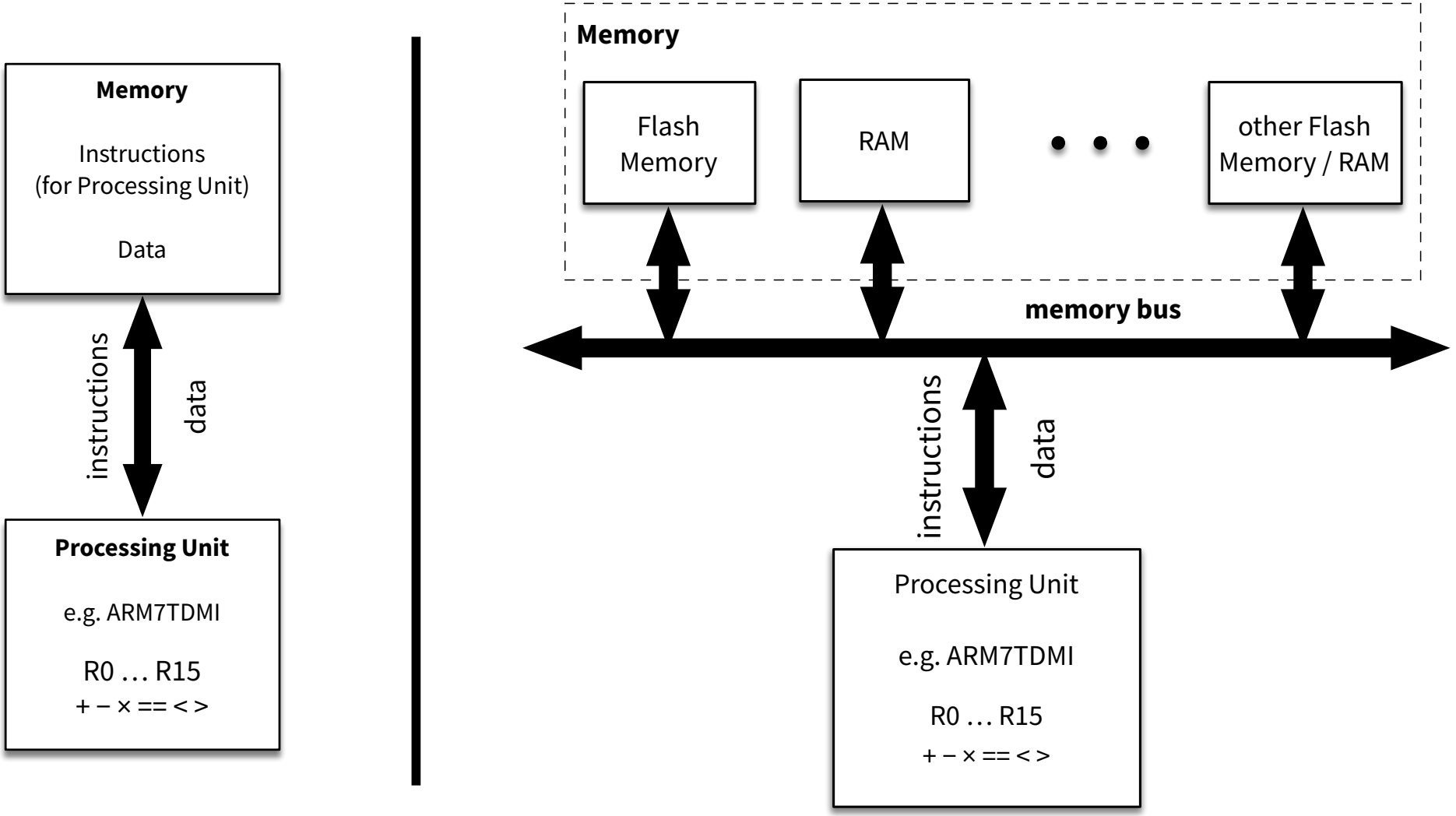
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

05 – Memory-Mapped I/O

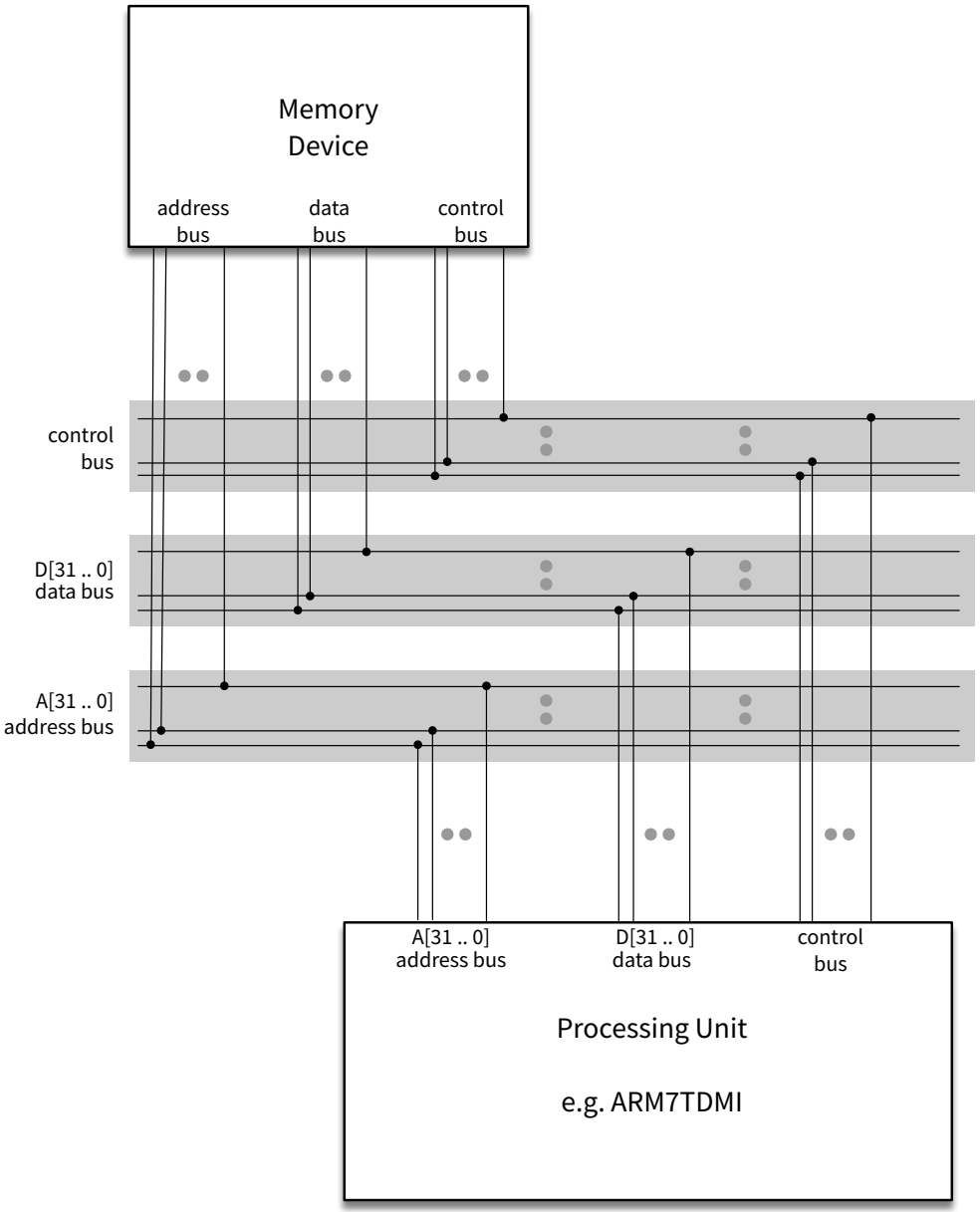
CS1022 – Introduction to Computing II

Dr Jonathan Dukes / jdukes@scss.tcd.ie
School of Computer Science and Statistics

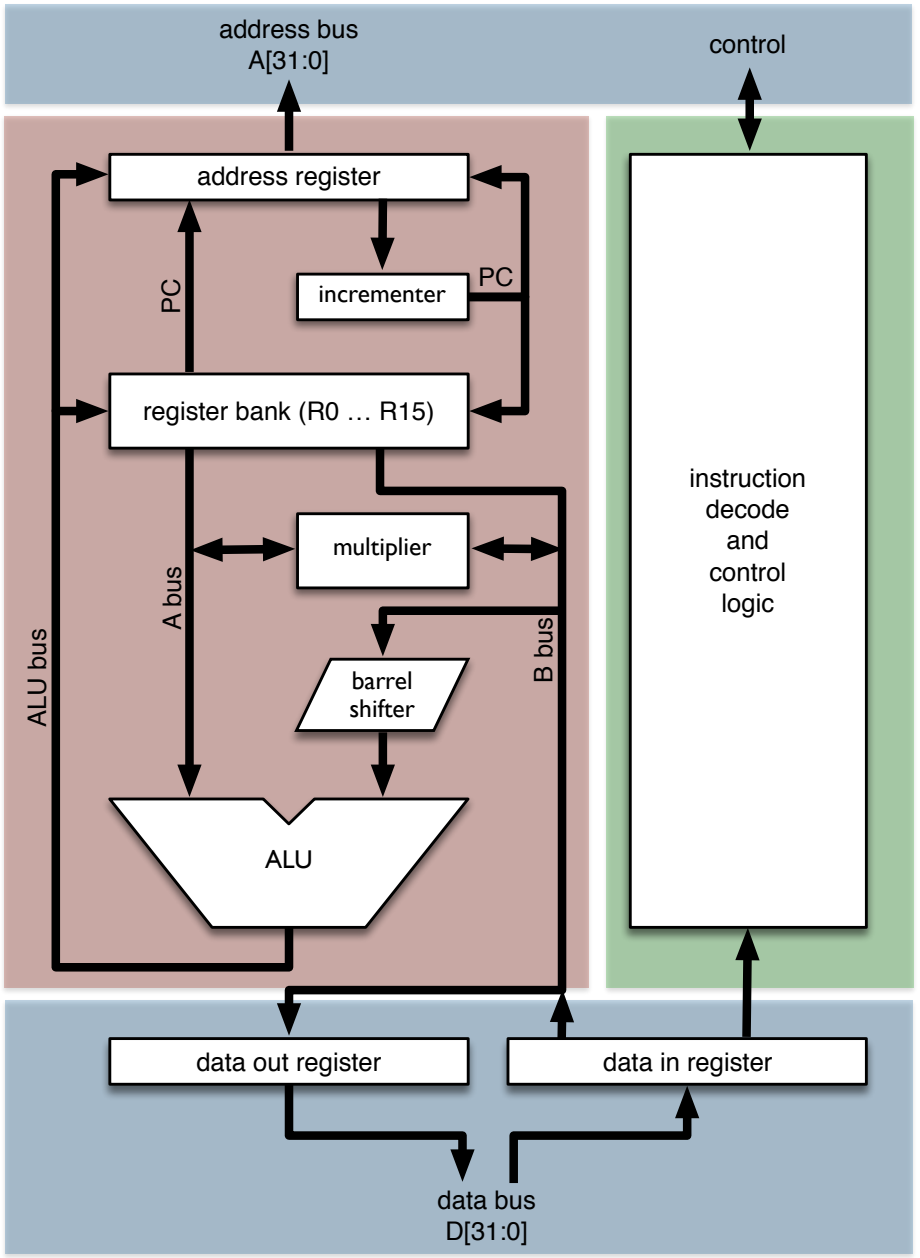
Simple Model of a Microprocessor



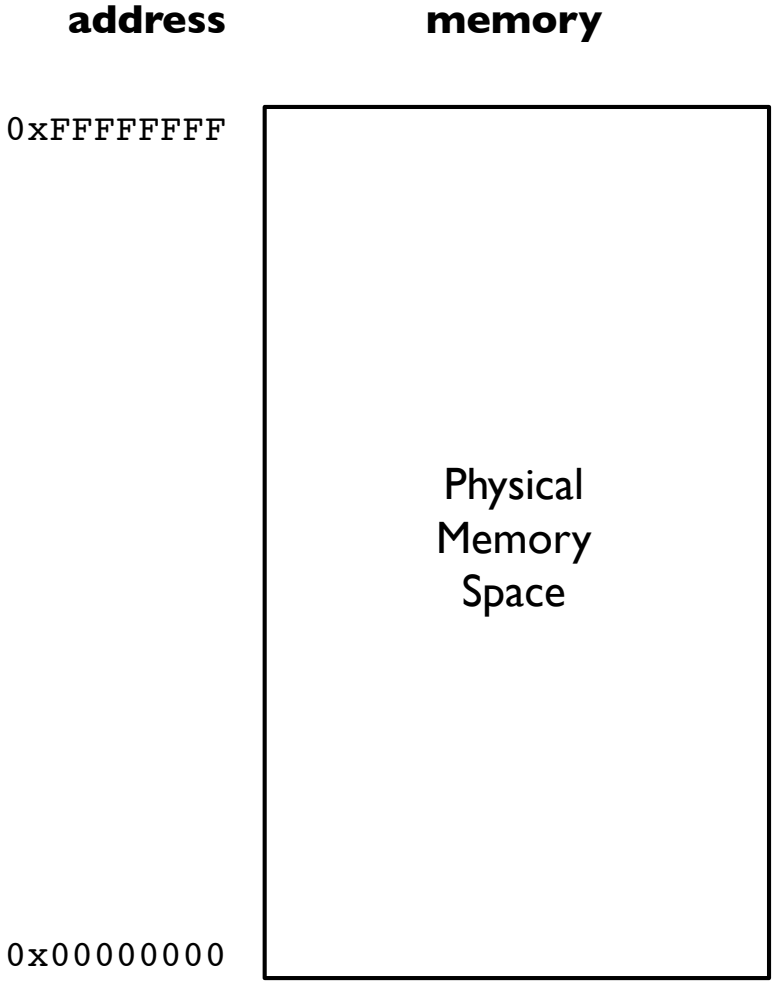
Memory Bus

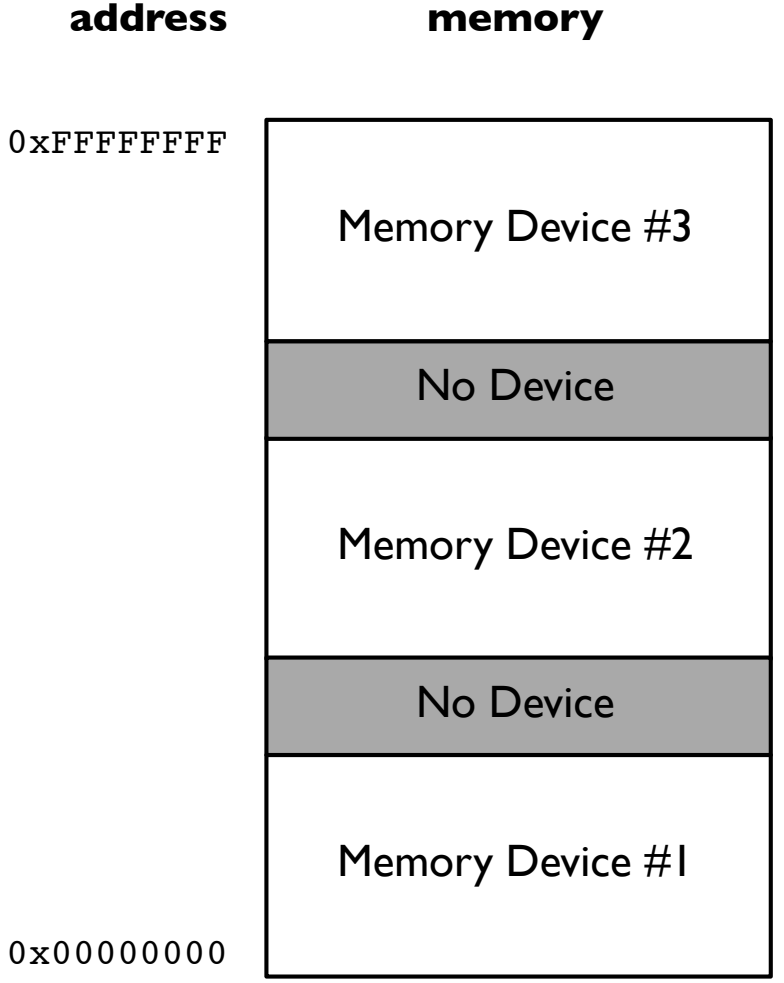

Memory
Device
address
bus
data
bus
control
bus
control
bus
D[31 .. 0]
data bus
A[31 .. 0]
address bus
A[31 .. 0]
address bus
D[31 .. 0]
data bus
control
bus
Processing Unit
e.g. ARM7TDMI

Steve Furber, ***"ARM System-on-Chip Architecture",*** 2nd edition, Addison Wesley Professional, 2000.



Address Space and Memory Map





NXP LPC2468

LPC2468 Development Board

On-Chip Flash (Read-Only) Memory (512KB)

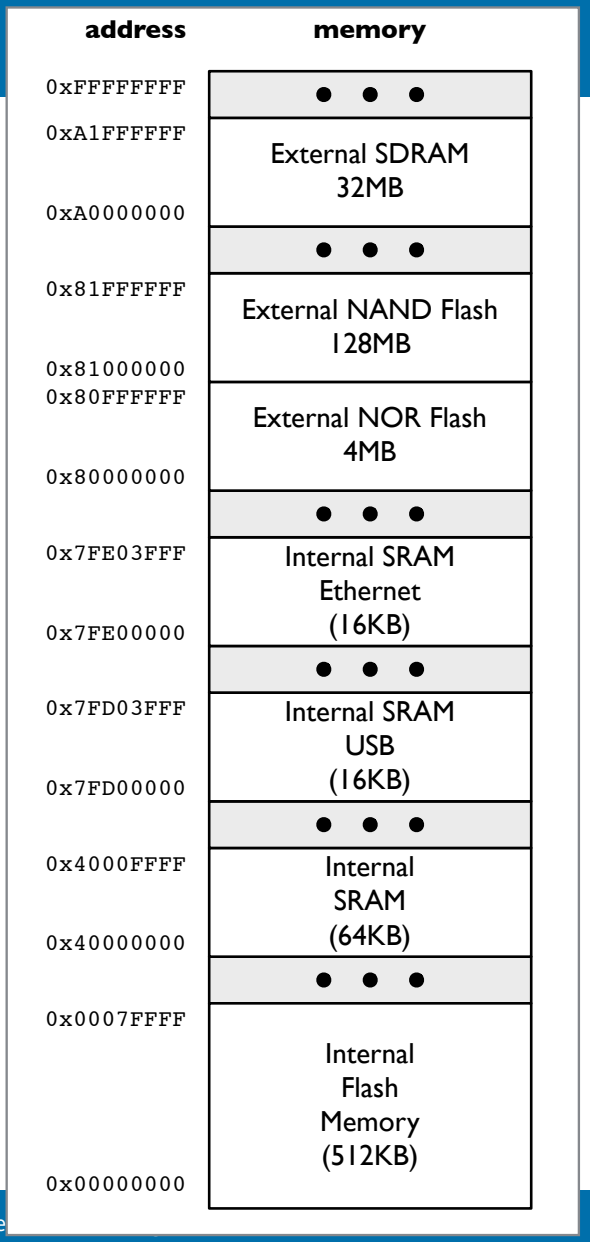
On-Chip RAM
(64KB + 16KB for Ethernet + 16KB = 96KB)

Off-Chip RAM (32MB)

4GB address space (32 bit addresses)

Each memory device is mapped into a region of the address space

Memory accesses (loads/stores) are directed to the device that is mapped into the LDR/STR effective address



Memory Mapping Example 1

512KB Internal Flash Memory

512KB = 524288 bytes = 2^{19} bytes

Address range: 0 … 524287_{10} = 0x00000 … 0x7FFFF

Device requires 19-bit addresses – A[18:0]

Choose address 0x00000000 as device base address

Mapped into processor address space 0x00000000 … 0x0007FFFF

All addresses with the binary prefix 0000000000000 map to this device

Similarly for Internal SRAM, External SDRAM, …

Memory Mapping Example 2

64KB Internal SRAM

64KB = 65536 bytes = 2^{16} bytes

Address range: 0 … 65535_{10} = 0x0000 … 0xFFFF

Device requires 16-bit addresses – A[15:0]

Choose address 0x40000000 as device base address

Mapped into processor address space 0x40000000 … 0x4000FFFF

All addresses with the binary prefix 0100000000000000 map to this device

Communicating with peripheral devices: TIMERx

Four TIMER peripherals: TIMER0/1/2/3

- Integrated on the LPC2468 micro controller
- But distinct from the ARM7TDMI microprocessor, Flash Memory and RAM

Measure accurate real-time intervals

- Implement real-time delays by matching current *Timer Counter* against software controlled *Match Registers* (timer-like behaviour)
- Capture *Timer Counter* on hardware signal or software event (stopwatch-like behaviour)

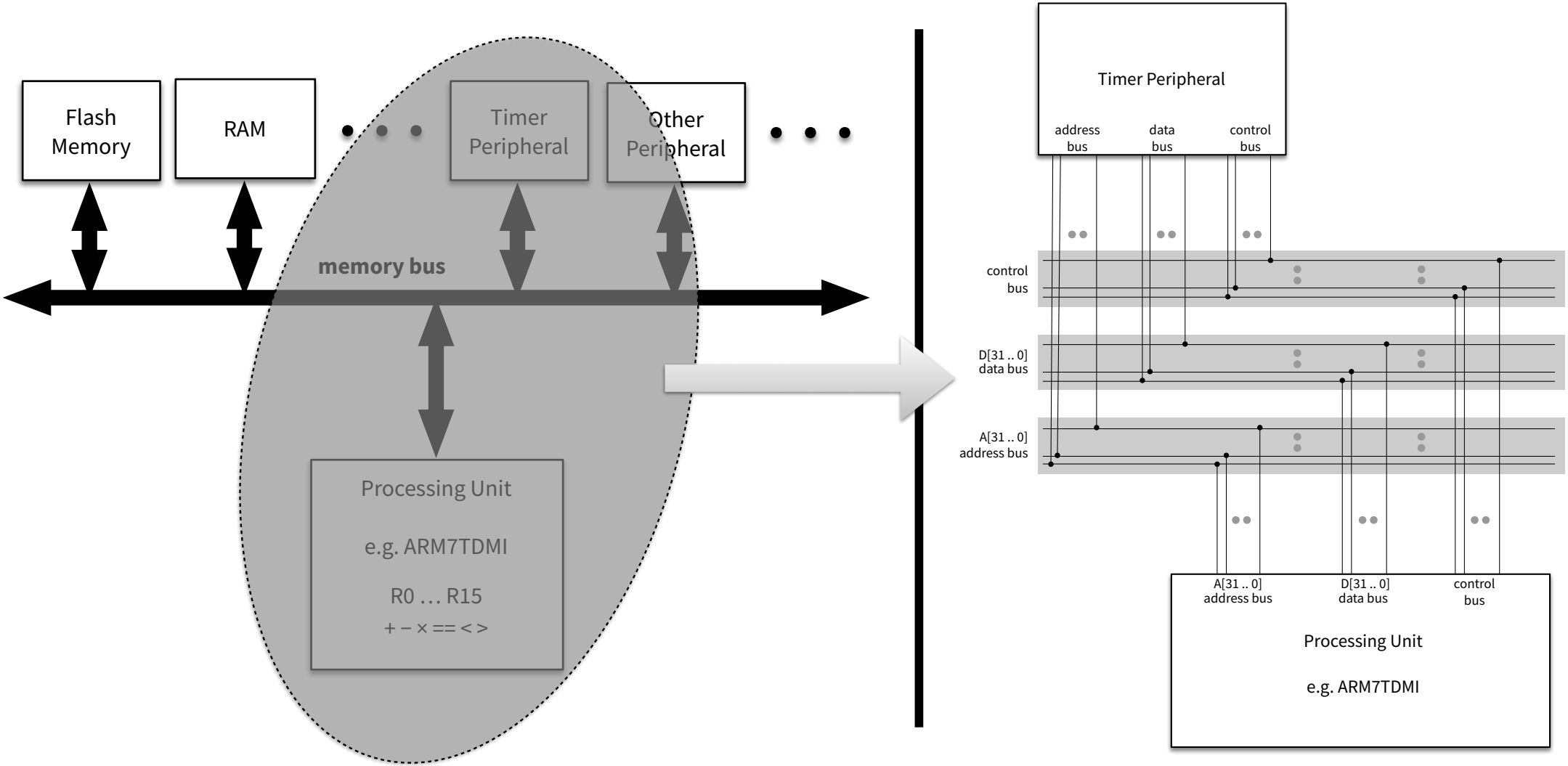
Control behaviour by storing values in TIMERx registers (STR)

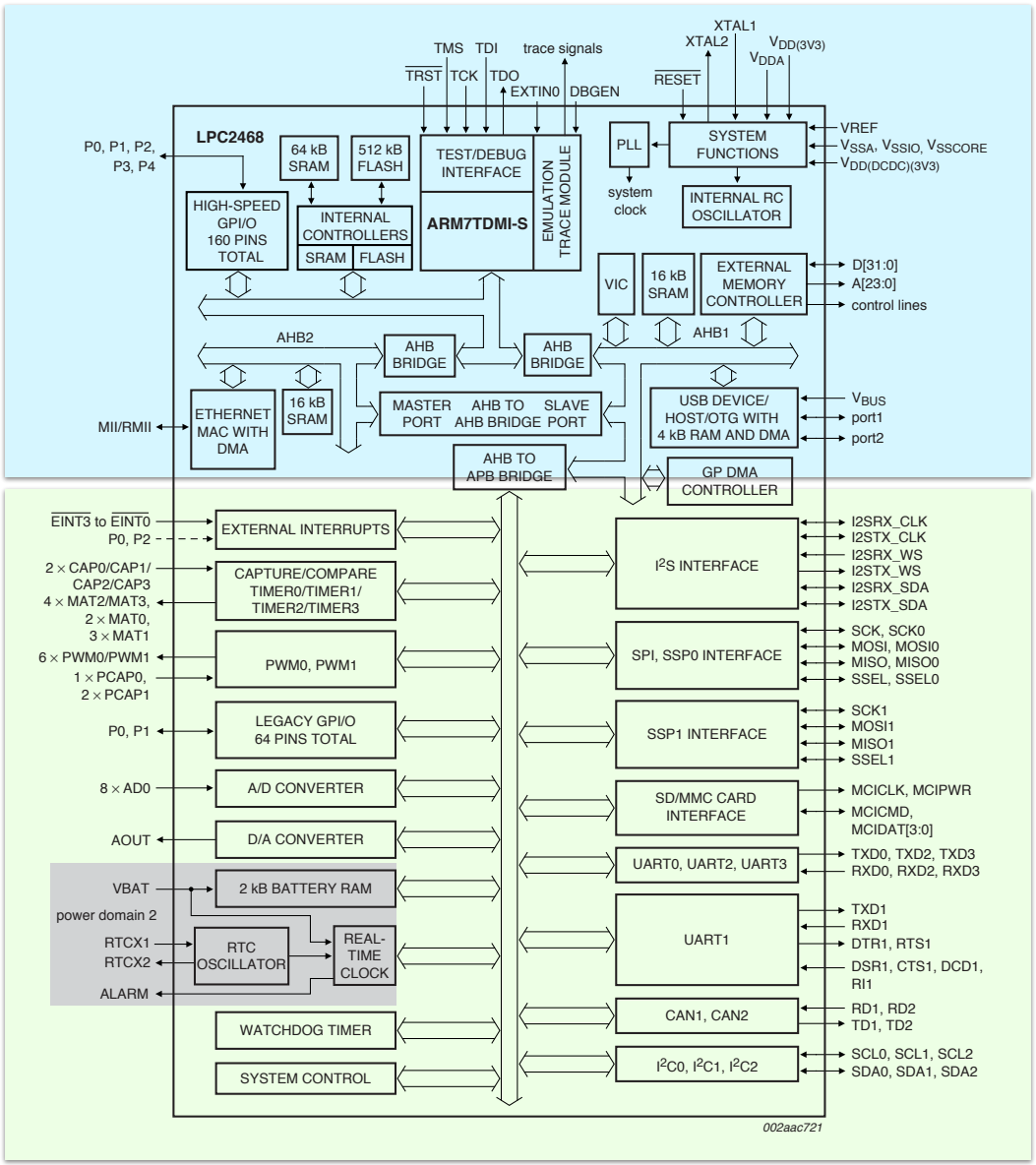
Read TIMER state or *Timer Counter* by loading from TIMERx registers (LDR)

see LPC2468 User Manual, Chapter 24

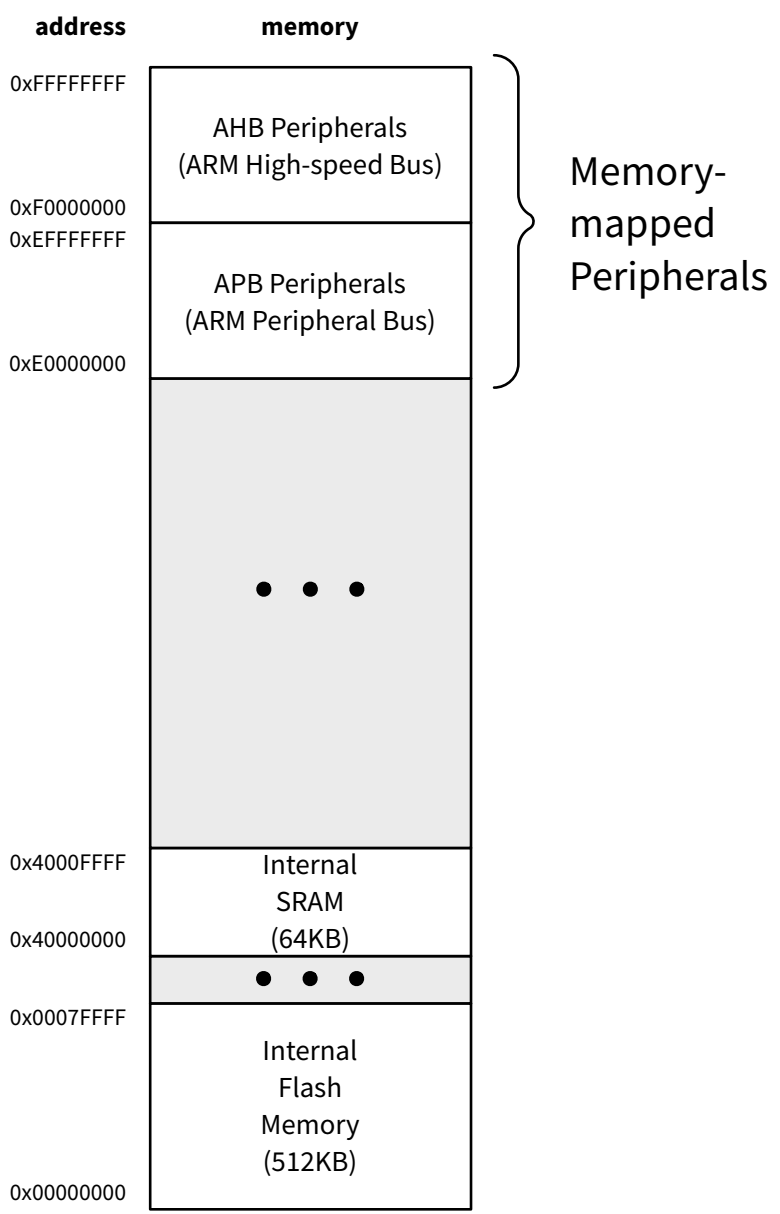
+0x70	Count Control Register
	● ● ●
+0x24	Match Register 3
+0x20	Match Register 2
+0x1C	Match Register 1
+0x18	Match Register 0
+0x14	Match Control Register
+0x10	Prescale Counter
+0x0C	Prescale Register
+0x08	Timer Counter
+0x04	Timer Control Register
+0x00	Interrupt Register

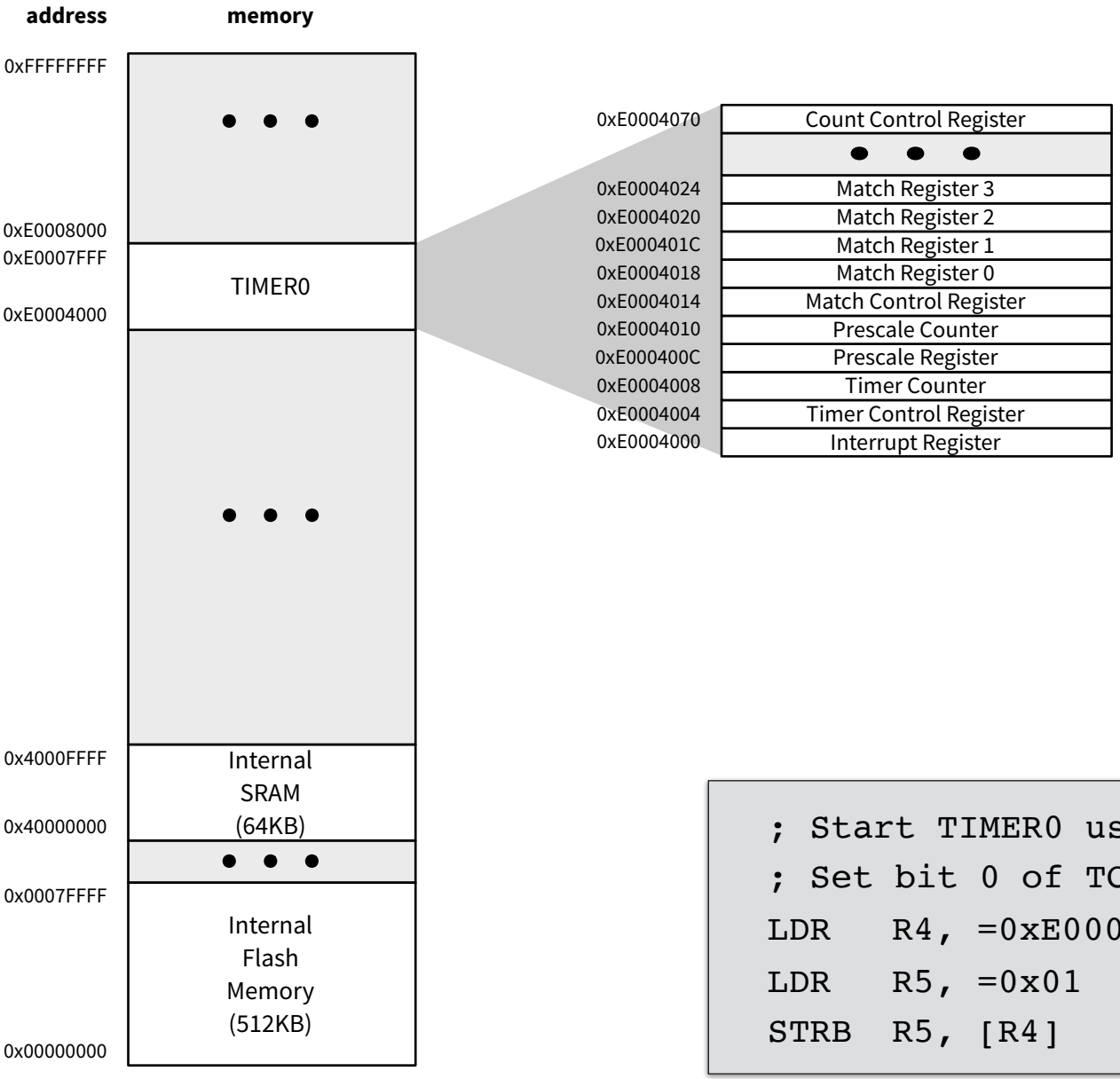
Communicating with peripheral devices: TIMERx





LPC2468 Block Diagram






```
; Start TIMER0 using the Timer Control Register
; Set bit 0 of TCR (0xE0004004) to enable the timer
LDR   R4, =0xE0004004
LDR   R5, =0x01
STRB  R5, [R4]
```

Example: Use TIMER0 to wait 5 seconds


```
T0TCR   EQU     0xE0004004
T0MR0   EQU     0xE0004018
T0MCR   EQU     0xE0004014

        LDR     R4, =T0MR0
        LDR     R5, =5000000
        STR     R5, [R4]

        LDR     R4, =T0MCR
        LDR     R5, =0x04
        STRH    R5, [R4]

        LDR     R4, =T0TCR
        LDR     R5, =0x01
        STRB    R5, [R4]

whWait
        LDRB    R5, [R4]
        TST     R5, #1
        BNE     whWait

        ; timer finished!

STOP
        B       STOP
```

Define labels for TIMER0 memory-mapped register addresses

Configure Match Register to match after 5 seconds
Assuming a 1Mhz clock input to TIMER0, set match register MR0 (0xE0004018) to 5,000,000

Stop on match using Match Control Register
Set bit 2 of MCR (0xE0004014) to 1

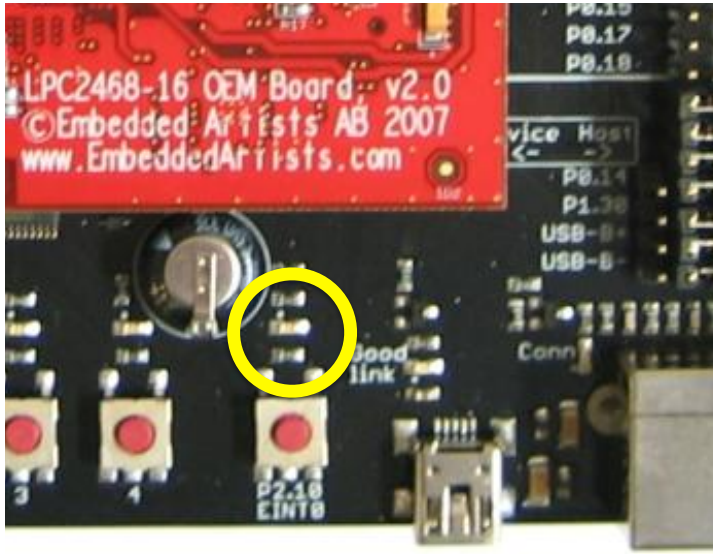
Start TIMER0 using the Timer Control Register
Set bit 0 of TCR (0xE0004004) to enable the timer

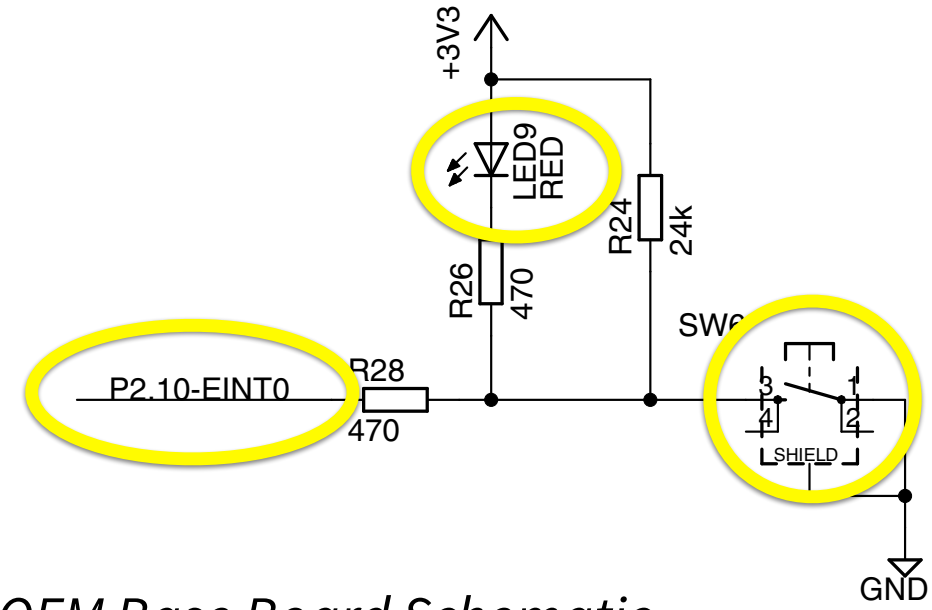
We configured TIMER0 to stop itself when the Timer Counter matches the value we placed in MR0
We can repeatedly test the least significant bit of the Timer Control Register (0xE0004004) waiting for the bit to become 0, indicating that TIMER0 has stopped.

Example: GPIO (General Purpose Input Output)

Design and write an ARM Assembly Language program that will cause an LED to blink on and off repeatedly

"Blinky"





see *LPC2468 OEM Base Board Schematic*

Pin Connect Block

Many external LPC2468 pins have multiple uses

Functionality of a pin is configured …

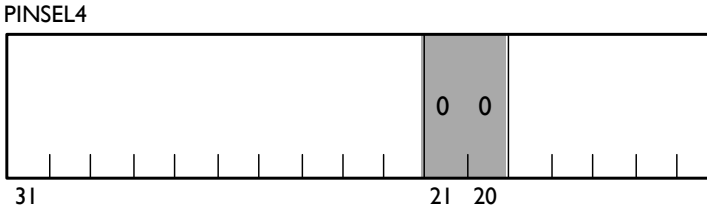
- by software, at runtime
- using the *Pin Connect Block* peripheral
- specifically, the **PINSELx** memory mapped register

PINSELx defines pin function

- Each 32-bit register controls 16 pins
  (2 bits to select one of $2^2 = 4$ possible functions for each physical pin)

Table 135. LPC2420/60/68/70/78 pin function select register 4 (PINSEL4 - address 0xE002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
21:20	P2[10]	GPIO Port 2.10	$\overline{\text{EINT0}}$	Reserved	Reserved	00



LPC2468 User Manual
Chapter 9: LPC24xx Pin Connect

LPC2468 User Manual
Chapter 9: LPC24xx Pin Connect

Table 135. LPC2420/60/68/70/78 pin function select register 4 (PINSEL4 - address 0xE002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2[0]	GPIO Port 2.0	PWM1[1]	TXD1	TRACECLK[1]/ LCDPWR	00
3:2	P2[1]	GPIO Port 2.1	PWM1[2]	RXD1	PIPESTAT0[1]/ LCDLE	00
5:4	P2[2]	GPIO Port 2.2	PWM1[3]	CTS1	PIPESTAT1[1]/ LCDDCLK	00
7:6	P2[3]	GPIO Port 2.3	PWM1[4]	DCD1	PIPESTAT2[1]/ LCDFP	00
9:8	P2[4]	GPIO Port 2.4	PWM1[5]	DSR1	TRACESYNC[1]/ LCDENAB/ LCDM	00
11:10	P2[5]	GPIO Port 2.5	PWM1[6]	DTR1	TRACEPKT0[1]/ LCDLP	00
13:12	P2[6]	GPIO Port 2.6	PCAP1[0]	RI1	TRACEPKT1[1]/ LCDVD[0]/ LCDVD[4]	00
15:14	P2[7]	GPIO Port 2.7	RD2	RTS1	TRACEPKT2[1]/ LCDVD[1]/ LCDVD[5]	00
17:16	P2[8]	GPIO Port 2.8	TD2	TXD2	TRACEPKT3[1]/ LCDVD[2]/ LCDVD[6]	00
19:18	P2[9]	GPIO Port 2.9	$\overline{\text{USB\_CONNECT1}}$	RXD2	EXTIN0[1]/ LCDVD[3]/ LCDVD[7]	00
21:20	P2[10]	GPIO Port 2.10	$\overline{\text{EINT0}}$	[illegible]	[illegible]	[illegible]
23:22	P2[11]	GPIO Port 2.11	$\overline{\text{EINT1}}$/ LCDCLKIN	MCIDAT1	I2STX_CLK	00
25:24	P2[12]	GPIO Port 2.12	$\overline{\text{EINT2}}$/ LCDVD[4]/ LCDVD[3]/ LCDVD[8]/ LCDVD[18]	MCIDAT2	I2STX_WS	00
27:26	P2[13]	GPIO Port 2.13	$\overline{\text{EINT3}}$/ LCDVD[5]/ LCDVD[9]/ LCDVD[19]	MCIDAT3	I2STX_SDA	00
29:28	P2[14]	GPIO Port 2.14	$\overline{\text{CS2}}$	CAP2[0]	SDA1	00
31:30	P2[15]	GPIO Port 2.15	$\overline{\text{CS3}}$	CAP2[1]	SCL1	00

From the schematic, we know we are looking for pin P2[10]. This pin has two functions: GPIO or EINT (external interrupt). We want to use the pin for GPIO so we need to set bits 21:20 to 00.

Notation: bits 21:20 means bits 20 and 21. Similarly, 19:0 means bits 0 ... 19.

Enable GPIO mode for Pin P2.10

Need to set bits 21:20 to 00_2 to enable GPIO functionality for pin P2[10]

IMPORTANT! Need to leave the other 30 bits (31:22 and 19:0) of PINSEL4 unmodified

Use a ***Read-Modify-Write*** operation to set register value

RMW is a common operation when interfacing with peripherals

(RMW is also an expensive operation as it requires two memory operations!)

PINSEL4 address – 0xE002C010 (from LPC2468 User Manual)

```
; Enable P2.10 for GPIO
LDR   R5, =0xE002C010   ; Address of PINSEL4
LDR   R6, [R5]          ; Read current PINSEL4 value
BIC   R6, #(0x3 << 20)  ; Modify to clear bits 21:20
STR   R6, [R5]          ; Write new PINSEL4 value
```

Set Pin P2.10 for output

GPIO pins can be either inputs or outputs

Controlling LED requires configuration of P2.10 as output

GPIO configured and controlled by GPIO peripheral

- Direction (I/O) set using **FIOxDIRy** register
- User Manual tells us we need to use FIO2DIR1 to configure P2.10
- Set bit 2 of FIO2DIR1 to 1 to configure P2.10 for output

```
; Set P2.10 for output
LDR   R5, =0x3FFFC041   ; Address of FIO2DIR1
LDRB  R6, [R5]          ; Read FIO2DIR1
ORR   R6, #(0x1 << 2)   ; Modify bit 2 to value 1 to configure output
STRB  R6, [R5]          ; Write FIO2DIR1
```

Repeatedly invert state of LED

Set output value (0/1) using bit 2 of FIO2PIN1 register

- Must not change other bits of FIO2PIN1

Read, Modify, Write again

- test if LED is on or off [READ]
- if it is off then turn it on, if it is on then turn it off [MODIFY]
- output new value [WRITE]

Delay implemented using TIMER0

Blinky


```
; TIMER0 registers
T0TCR    EQU     0xE0004004
T0MR0    EQU     0xE0004018
T0MCR    EQU     0xE0004014

; Pin Control Block registers
PINSEL4  EQU     0xE002C010

; GPIO registers
FIO2DIR1 EQU     0x3FFFC041
FIO2PIN1 EQU     0x3FFFC055

         ; Enable P2.10 for GPIO
         LDR     R5, =PINSEL4     ; load address of PINSEL4
         LDR     R6, [R5]         ; read current PINSEL4 value
         BIC     R6, #(0x3 << 20) ; modify bits 20 and 21 to 00
         STR     R6, [R5]         ; write new PINSEL4 value

         ; Set P2.10 for output
         LDR     R5, =FIO2DIR1    ; load address of FIO2DIR1
         LDRB    R6, [R5]         ; read current FIO2DIR1 value
         ORR     R6, #(0x1 << 2)  ; modify bit 2 to 1 for output, leaving other bits unmodified
         STRB    R6, [R5]         ; write new FIO2DIR1
```

Blinky (continued)


```
        ; Set match register for 5 secs using Match Register
        ;   Assuming a 1Mhz clock input to TIMER0, set MR
        ;   MR0 (0xE0004018) to 5,000,000
        LDR     R4, =T0MR0
        LDR     R5, =5000000
        STR     R5, [R4]

        ; Stop on match using Match Control Register
        ;   Set bit 2 of MCR (0xE0004014) to 1 to stop the counter after
        ;   match (5 secs)
        LDR     R4, =T0MCR
        LDR     R5, =0x04
        STRH    R5, [R4]
```

Blinky (continued)

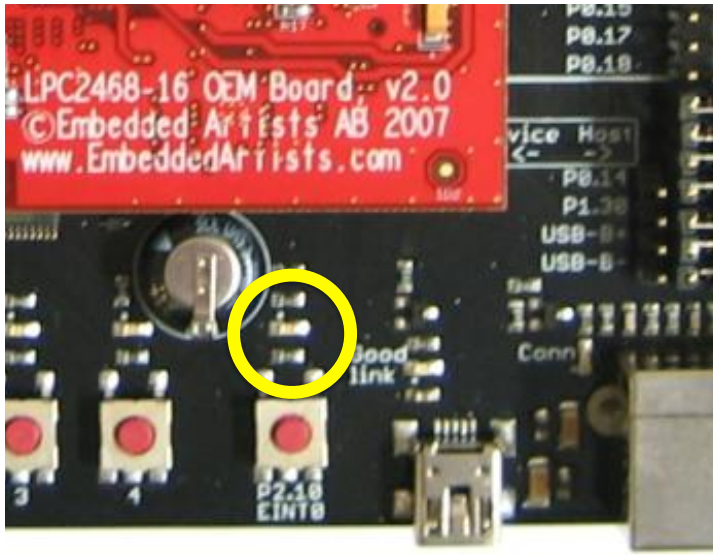

```
whBlink
        ; Reset TIMER0 using Timer Control Register
        ;   Set bit 0 of TCR to 0 to stop TIMER
        ;   Set bit 1 of TCR to 1 to reset TIMER
        LDR     R5, =T0TCR
        LDR     R6, =0x2
        STRB    R6, [R5]

        ; Start TIMER0 using the Timer Control Register
        ;   Set bit 0 of TCR to enable the timer
        LDR     R4, =T0TCR
        LDR     R5, =0x01
        STRB    R5, [R4]

        ; Keep testing TCR until the timer has stopped
whWait
        LDRB    R5, [R4]
        TST     R5, #1
        BNE     whWait

        ;
        ; Timer finished ... invert the LED
        ;   Another Read-Modify-Write operation!!
        ;
```

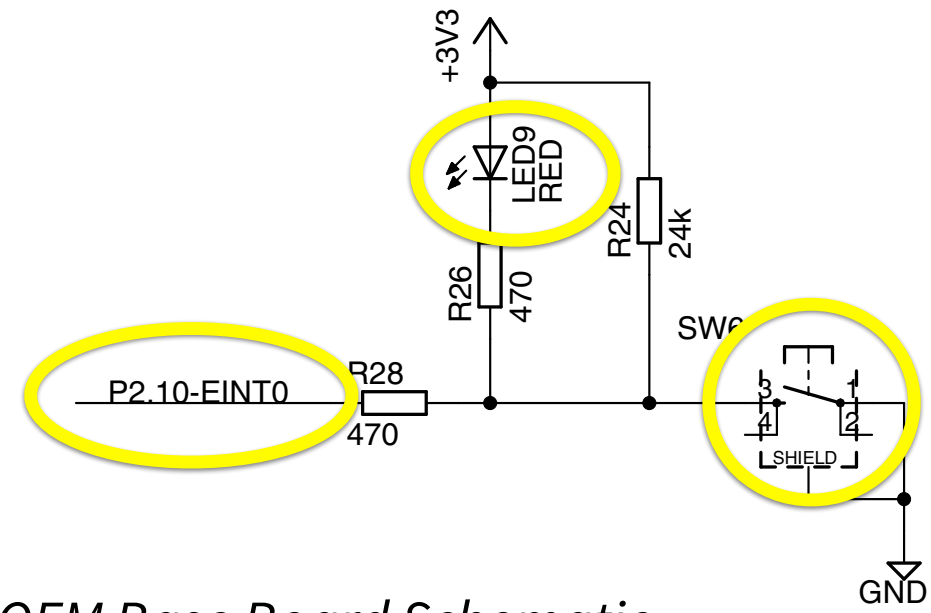
Blinky (continued)


```
        ; read current P2.10 output value
        ;   0 or 1 in bit 2 of FIO2PIN1
        LDR    R4, =0x04        ; setup bit mask for P2.10 bit in FIO2PIN1
        LDR    R5, =FIO2PIN1    ; load address of FIO2PIN1
        LDRB   R6, [R5]         ; read FIO2PIN1

        ; modify P2.10 output (leaving other pin outputs controlled by
        ;   FIO2PIN1 with their original value)
        TST    R6, R4           ;   if (bit 2 is zero)
        BNE    elsOff           ;   {
        ORR    R6, R6, R4       ;     set bit 2 (turn LED off)
        B      endIf            ;   }
elsOff                          ;   else {
        BIC    R6, R6, R4       ;     clear bit 2 (turn LED on)
endIf                           ;   }

        ; write new FIO2PIN1 value
        STRB   R6, [R5]

        ; repeat forever
        B      whBlink
```

GPIO Input

Design and write an ARM Assembly Language program that will count the number of times the pushbutton connected to pin P2.10 is pressed





see *LPC2468 OEM Base Board Schematic*

Polling


```
; Pin Control Block registers
PINSEL4           EQU               0xE002C010

; GPIO registers
FIO2DIR1  EQU     0x3FFFC041
FIO2PIN1  EQU     0x3FFFC055

          AREA    RESET, CODE, READONLY
          ENTRY

          ; Enable P2.10 for GPIO
          LDR     R4, =PINSEL4      ; load address of PINSEL4
          LDR     R5, [R4]          ; read current PINSEL4 value
          BIC     R5, #(0x3 << 20)  ; modify bits 20 and 21 to 00
          STR     R5, [R4]          ; write new PINSEL4 value

          ; Set P2.10 for input
          LDR     R4, =FIO2DIR1     ; load address of FIO2DIR1
          LDRB    R5, [R4]          ; read current FIO2DIR1 value
          BIC     R5, #(0x1 << 2)   ; modify bit 2 to 0 for input, leaving other bits unmodified
          STRB    R5, [R4]          ; write new FIO2DIR1

          LDR     R4, =FIO2PIN1     ; load address of FIO2PIN1

          MOV     R7, #0            ; count = 0
```

Polling (continued)


```
whRepeat                        ; while (forever) {
        LDRB   R6, [R4]         ;   lastState = FIO2DIR1 & 0x4
        AND    R6, R6, #0x4     ;

        ; keep testing pin state until it changes

whPoll                          ;   do {
        LDRB   R5, [R4]         ;     currentState = FIO2DIR1 & 0x4
        AND    R5, R5, #0x4     ;
        CMP    R5, R6           ;
        BEQ    whPoll           ;   } while (currentState == lastState)

        ; pin state has changed ... but has it changed to 0?

        CMP    R5, #0           ;   if (currentState == 0) {
        BNE    eIf              ;

        ADD    R7, R7, #1       ;     tmp++
eIf
        B      whRepeat       ; }
```