DESN2000: Engineering Design & Professional Practice (EE&T)

Week 8

Input and output interfaces

David Tsai

School of Electrical Engineering & Telecommunications

Graduate School of Biomedical Engineering

d.tsai@unsw.edu.au

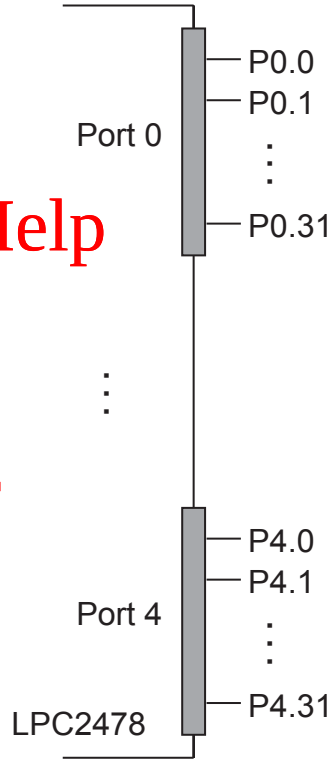
Biomedical Microsystems Lab

This week

- GPIO intricacies
- UART
- D/A (digital-to-analog) converters

Fast GPIO

- The LPC2478 has lots of I/Os (all mem mapped).
- The simplest type:
  5 Fast General Purpose I/O ports (GPIO), each with 32 pins:
  - Port 0
  - Port 1
  - Port 2
  - Port 3
  - Port 4



Fast GPIO

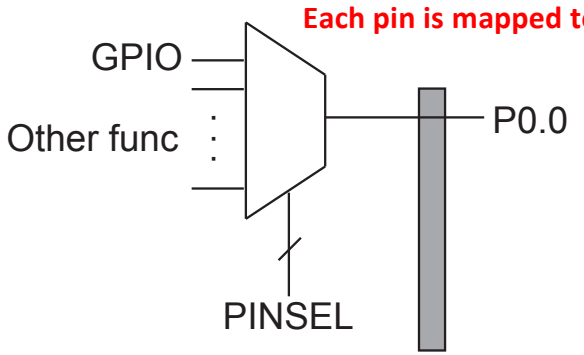
- **Five** types of **control registers** associated with each port.
- Each register has:
  - a particular name, specifying its purpose
  - an address in memory space for fast GPIO, that is 0x3FFFC000 – 0x3FFFFFFF.

Each port has:

DIR
PIN
SET
CLR
MASK

Memory mapped registers

Each pin is mapped to GPIO by default.



Fast GPIO

- Consider the following C code for blinking LEDs connected to pin0.0 – pin0.7 ON and OFF for ever.

```
#define FIO0DIR *(volatile unsigned int *) (0x3FFFC000)   // port-0 dir
#define FIO0PIN *(volatile unsigned int *) (0x3FFFC014).  // port-0 pin

int main(void) {
    FIO0DIR = 0xFF;
    while (1) {
        FIO0PIN = 0xFF;
        FIO0PIN = 0x00;
    }
    return 0;
}
```

Fast GPIO

- Compiling to assembly:

With “volatile”

```
FIO0DIR     EQU 0x3FFFC000
FIO0PIN     EQU 0x3FFFC014
            AREA, LEDONOFF, CODE
            ENTRY
            LDR  R0, =FIO0DIR
            LDR  R1, =FIO0PIN
            MOV  R2, #0xFF
            STR  R2, [R0] ; out
            MOV  R3, #0
loop        STR  R2, [R1] ;1. blink on
            STR  R3, [R1] ;2. blink off
            B    loop
            END
```

Without “volatile”

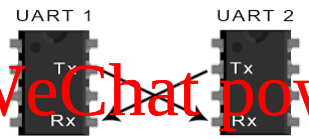
```
FIO0DIR     EQU 0x3FFFC000
FIO0PIN     EQU 0x3FFFC014
            AREA, LEDONOFF, CODE
            ENTRY
            LDR  R0, =FIO0DIR
            LDR  R1, =FIO0PIN
            MOV  R2, #0xFF
            STR  R2, [R0] ; out
            MOV  R3, #0
loop        STR  R3, [R1] ;1. blink off
            B    loop
            END
```

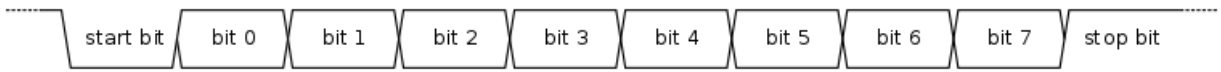

- Q: what happened when we omit “volatile” in the C code?

The compiler decided the first pin assignment is superfluous and optimized it out.

UART

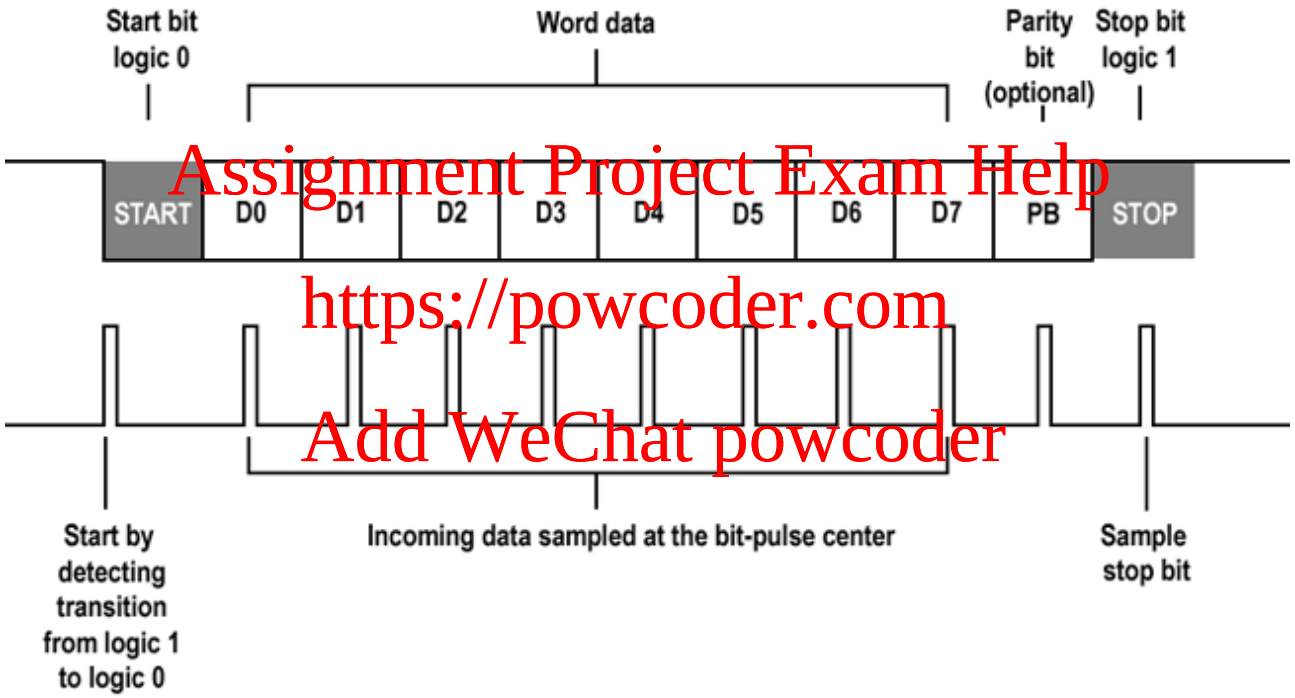
- The Universal Asynchronous Receiver/ Transmitter (UART) is a **protocol** for serial communication to asynchronously **send / receive sequence of bits**.
- A simple, inexpensive method for low-speed transmission (up to ~14 kB/s).
  E.g. CPU to peripheral device.
- Line held HIGH when idle. Start bit is LOW, followed by sequence of data bits, then a stop bit.





UART

- Example: UART transmission with 7-bit character length, (optional) parity and 1 stop bit.



UART register map

- All UART registers are memory mapped:
  - configuration registers
  - receive and transmit buffers
  - status registers
- UART0 has base address 0xE00C000.
  - Registers associated with it spans 0xE00C000 ~ 0xE00C030.
- Example: the Line Control Register for UART0 is at 0xE000C00C.

UART register map

LPC24XX User manual. Document No: UM10237

Table 377. UART Register Map

Generic Name	Description	Bit functions and addresses (MSB → LSB)								Access	Reset value[1]	UARTn Register Name & Address
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
X RBR (DLAB=0)	Receiver Buffer Register	8 bit Read Data								RO	NA	U0RBR - 0xE000 C000 U2RBR - 0xE007 8000 U3RBR - 0xE007 C000
X THR (DLAB=0)	Transmit Holding Register	8 bit Write Data								WO	NA	U0THR - 0xE000 C000 U2THR - 0xE007 8000 U3THR - 0xE007 C000
X DLL (DLAB=1)	Divisor Latch LSB	8 bit Data								R/W	0x01	U0DLL - 0xE000 C000 U2DLL - 0xE007 8000 U3DLL - 0xE007 C000
DLM (DLAB=1)	Divisor Latch MSB	8 bit Data								R/W	0x00	U0DLM - 0xE000 C004 U2DLM - 0xE007 8004 U3DLM - 0xE007 C004
X IER (DLAB=0)	Interrupt Enable Register	Reserved						Enable Auto-Baud Time-Out Interrupt	Enable End of Auto-Baud Interrupt	R/W	0x00	U0IER - 0xE000 C004 U2IER - 0xE007 8004 U3IER - 0xE007 C004
		0					Enable RX Line Status Interrupt	Enable THRE Interrupt	Enable RX Data Available Interrupt			
IIR	Interrupt ID Register	Reserved						ABTOInt	ABEOint	RO	0x01	U0IIR - 0xE000 C008 U2IIR - 0xE007 8008 U3IIR - 0xE007 C008
		FIFOs Enabled		0		IIR3	IIR2	IIR1	IIR0			
FCR	FIFO Control Register	RX Trigger		Reserved			TX FIFO Reset	RX FIFO Reset	FIFO Enable	WO	0x00	U0FCR - 0xE000 C008 U2FCR - 0xE007 8008 U3FCR - 0xE007 C008
X LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0x00	U0LCR - 0xE000 C00C U2LCR - 0xE007 800C U3LCR - 0xE007 C00C

0 UART0
2 UART2
3 UART3

To set baud rate

Interrupt driven IO

Config UART protocol

UART register map

Table 377. UART Register Map

Generic Name	Description	Bit functions and addresses (MSB → LSB)	Access	Reset value[1]	UARTn Register Name & Address
LSR	Line Status Register	RX FIFO Error, TEMT, THRE, BI, FE, PE, OE, DR	RO	0x60	U0LSR - 0xE000 C014 U2LSR - 0xE007 8014 U3LSR - 0xE007 C014
SCR	Scratch Pad Register	8 bit Data	R/W	0x00	U0SCR - 0xE000 C01C U2SCR - 0xE007 801C U3SCR - 0xE007 C01C
ACR	Auto-baud Control Register	Reserved [31:10], ABEOIntClr, ABTOIntClr Reserved [7:3], Auto Reset, Mode, Start	R/W	0x00	U0ACR - 0xE000 C020 U2ACR - 0xE007 8020 U3ACR - 0xE007 C020
ICR	IrDA Control Register	Reserved, PulseDiv, FixPulseEn, IrDAInv, IrDAEn	R/W	0	U3ICR - 0xE000 C024 (UART3 only)
FDR	Fractional Divider Register	MulVal, DivAddVal	R/W	0x10	U0FDR - 0xE000 C028 U2FDR - 0xE007 8028 U3FDR - 0xE007 C028
TER	Transmit Enable Register	TXEN, Reserved	R/W	0x80	U0TER - 0xE000 C030 U2TER - 0xE007 8030 U3TER - 0xE007 C030

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

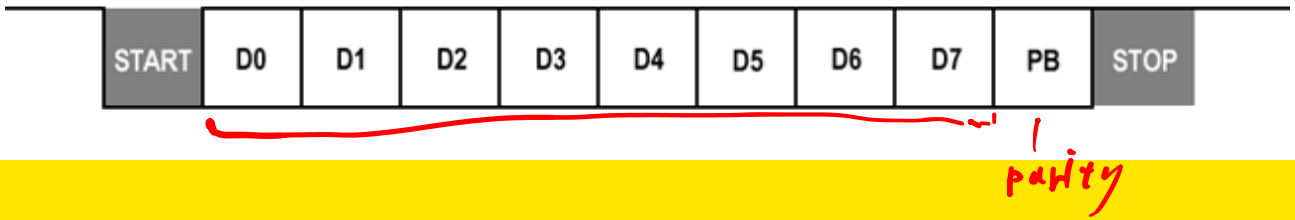
X

UART's register map

Table 386. UARTn Line Control Register (U0LCR - address 0xE000 C00C, U2LCR - 0xE007 800C, U3LCR - 0xE007 C00C) bit description

For UARTs 0, 2 and 3

Bit	Symbol	Value	Description	Reset Value
1:0	Word Length Select	00	5 bit character length	0
		01	6 bit character length	
		10	7 bit character length	
		11	8 bit character length	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if UnLCR[1:0]=00).	
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UART0 TXD is forced to logic 0 when UnLCR[6] is active high.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

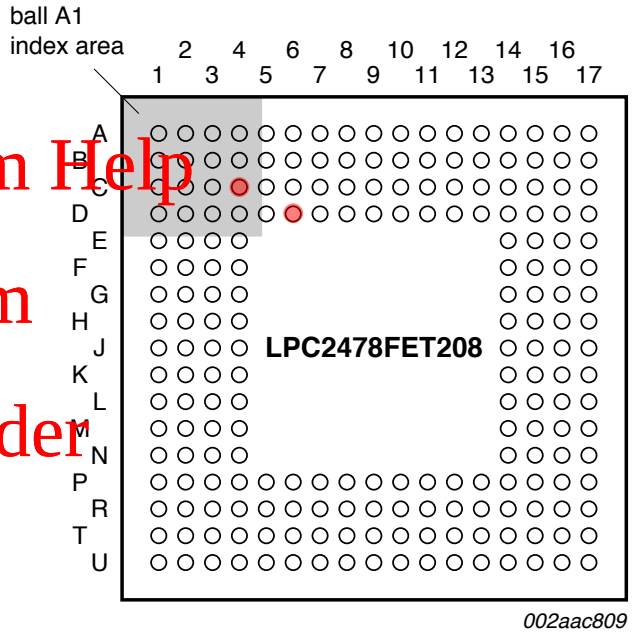


Using UART

- Two steps, configure the hardware pins and the software protocol:
  1. **Hardware**: selects the appropriate pins of LPC2478 as RXD0 (input) and TXD0 (output).
  2. **Software**: configuring UART's number of data bits, parity, number of stop bits, and baud rate.

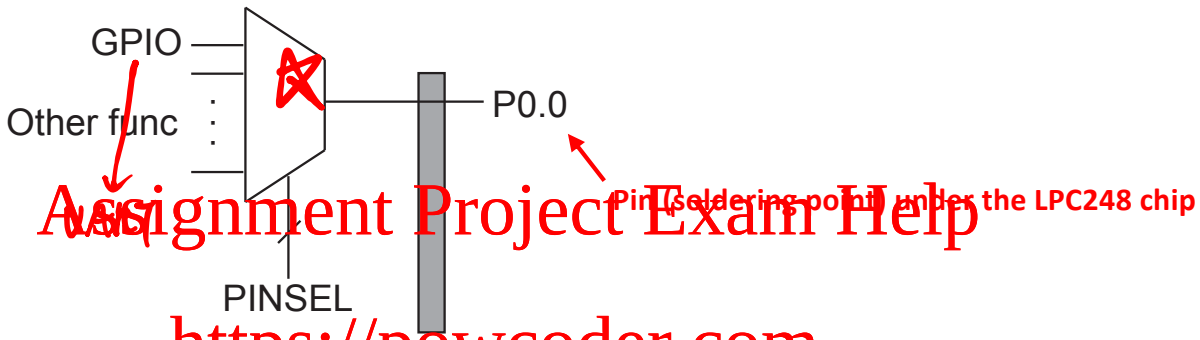
Using UART: configuring pins (HW)

- LPC2478 has 208 pins.
- Pins are multifunctional:
  - Row-C column-4 pin can be either GPIO0[2] or **UART0 TXD0**.
  - Row-D column-6 pin can be either GPIO0[3] or **UART0 RXD0**.
- For UART: row-C column-4 pin should be configured as **TXD0** and Row-D column-6 pin should be configured as **RXD0**.



Using UART: configuring pins (HW)

- Each I/O pin (under the chip package) is connected to a multiplexer, which determines the internal circuits routed to the pin.



- There are 12 PINSEL registers (PINSEL0 – PINSEL11) for LPC2478, all memory-mapped.

- The address of PINSEL0 register is **0xE002C000**.
- Set PINSEL0[5:4] to 0x01 for P0.2 to be used as TXD0.
  (the pins would be 0x00 if P0.2 is used as GPIO)
- Set PINSEL0[7:6] to 0x01 for P0.3 to be used as RXD0.
  (the pins would be 0x00 if P0.3 is used as GPIO)

Using UART: configuring pins (HW)

- Configuring the pins as RXD0 and TXD0 using read-modify-write:

```
PINSEL0     EQU  0XE002C000
            LDR  R5, =PINSEL0
            LDR  R6, [R5]        ; read
            BIC  R6, R6, 0xF0    ; clearing bit 7~4
            ORR  R6, R6, #0x50   ; P0.2 to TXD0, P0.3 to RXD0
            STR  R6, [R5]        ; write
```

bits 7 6 5 4

0 1 0 1

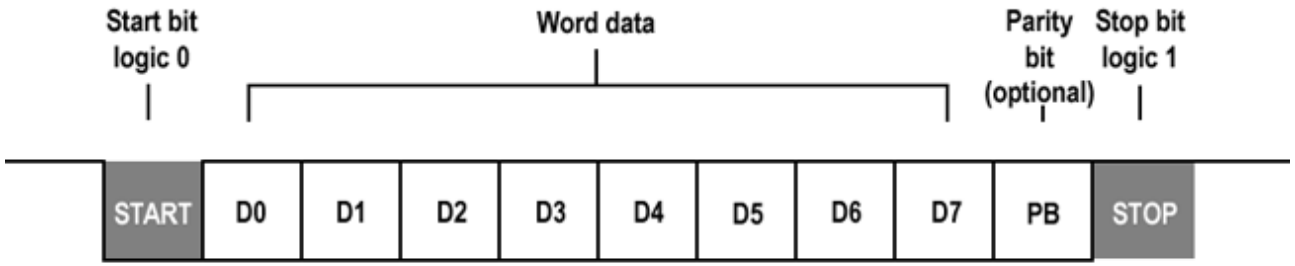
8 4 2 1

Using UART: configuring protocol (SW)

- Base address of UART0 configuration register is **0xE000C000**.
- Address offset for registers:
  - **LCR0 = 0xC** addressing the Line Control Register.
  - **LSR0 = 0x14** addressing the Status Register.

- Setting UART0 for **8 bits** data, **no parity**, **1 stop bit**, **9600 baud** @ 17.75 MHz APB clock (peripheral clock, reset value 71 MHz ÷ 4).

  - Baud rate: transmission speed in bits-per-second (bps).
  - APB clock: clock for IO devices.



Using UART: configuring protocol (SW)

- Configuring the UART protocol:

line ctrl reg

```
*UOSTART    EQU  0XE000C000
 LCR0       EQU  0XC
            LDR  R5, =UOSTART
            MOV  R6, #0x83          ; DLAB=1,no parity, 1 stop bit, 8 bits
            STRB R6, [R5, #LCR0]    ; write control byte to LCR
            MOV  R6, #0x73          ; 9600baud @ 17.75MHz APB clk
            STRB R6, [R5]           ; store control byte (DLAB=1)
            MOV  R6, #3             ; DLAB=0, turn off Divisor Latch Access mode
            STRB R6, [R5, #LCR0]
```

Using UART: writing data

- Now ready to send and receive UART data.
- Putting a character in the transmit buffer, but only after the previous character has been transmitted (check the *Transmitter Holding Register Empty* bit):

```
UOSTART     EQU  0XE000C000
LSR0        EQU  0X14
            LDR  R5, =UOSTART
wait        LDRB R6, [R5, #LSR0]  ; get buffer status
            CMP  R6, #0x20        ; THRE (TX buffer empty)?
            BNE  wait             ; spin till buffer empty
            STRB R0, [R5]         ; R0 contains the character
```


- For general purpose TX: write a subroutine that sends one character at a time, then call it as many times as needed to send a message out of UART0.

Using UART: complete transmitter

```
                AREA UARTDEMO, CODE, READONLY
PINSEL0         EQU  0xE002C000
U0START         EQU  0xE000C000
LCR0            EQU  0xC
LSR0            EQU  0x14
RAMSTART        EQU  0x40800200   ; start of LPC2478 on-chip RAM

                ENTRY

start           LDR  SP, =RAMSTART  ; set up stack pointer
                BL   uartConfig
                LDR  R1, =CharData  ; string addr
loop            LDRB A1, [R1], #1
                CMP  A1, #0
                BLNE transmit
                BNE  loop
done            B    done
```

Transmit msg till '/0'

Mem-mapped I/O definitions

Main function

Using UART: complete transmitter

```
uartConfig  STMIA SP!, {R5, R6, LR}
            LDR  R5, =PINSEL0
            LDR  R6, [R5]
            BIC  R6, R6, 0xF0
            ORR  R6, R6, #0x50
            STR  R6, [R5]
            LDR  R5, =U0START
            MOV  R6, #0x83
            STRB R6, [R5, #LCR0]
            MOV  R6, #0x73
            STRB R6, [R5]
            MOV  R6, #3
            STRB R6, [R5, #LCR0]
            LDMDB SP!, {R5, R6, PC}
```

UART pin configuration (HW)

UART protocol configuration (SW)

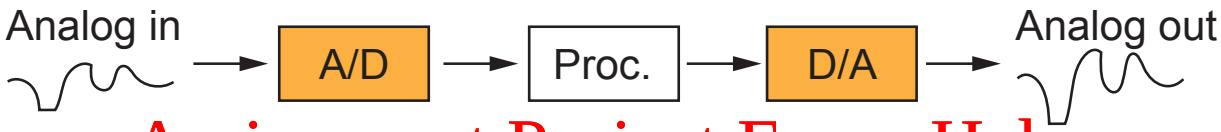
Using UART: complete transmitter

```
transmit    STMIA SP!, {R5, R6, LR}
            LDR   R5, =UOSTART
wait        LDRB  R6, [R5, #LSR0]
            CMP   R6, #0x20
            BNE   wait
            STRB  A1, [R5]
            LDMDB SP!, {R5, R6, PC}

charData    DCB "Watson, come quickly!", 0

            END
```

TX a byte at a time

D/A converter

- In many signal processing and control applications, an analog input is sampled, processed then converted back into an analog output.

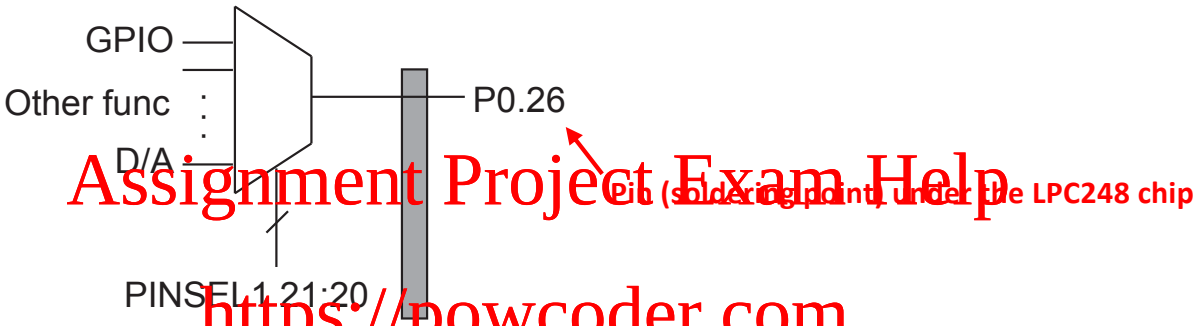


- The LPC2478's D/A converter takes a 10-bit binary value and generates a voltage on A_{OUT} which is proportional to a reference voltage V_{REF}:

$$V_{out} = \frac{V}{1024} V_{ref}$$

D/A converter: configuring pins

- To use the D/A converter, pin P0.26 (row E, column 1) must be configured for A_{OUT}, achieved by setting PINSEL1[21:20] (at 0xE002C004) to 0x10:




```
PINSEL1     EQU  0xE002C004
            LDR  r6, =PINSEL1
read →      LDR  r7, [r6]          ; read pin
modify      ORR  r7,r7, #1:SHL:21  ; set bit-21 HI
            BIC  r7,r7, #1:SHL:20  ; set bit-20 LO
write →     STR  r7, [r6]          ; write (P0.26 now Aout)
```

D/A converter: configuring DACR

- DACR register (0xE006C000) is used to
  - configure the converter
  - provide a value for conversion
- DACR[15:6]: 10-bit value for setting A_{OUT}.
- DACR[16]: Sets the DAC's settling speed.
  Fast settling → quicker output change → use more power.

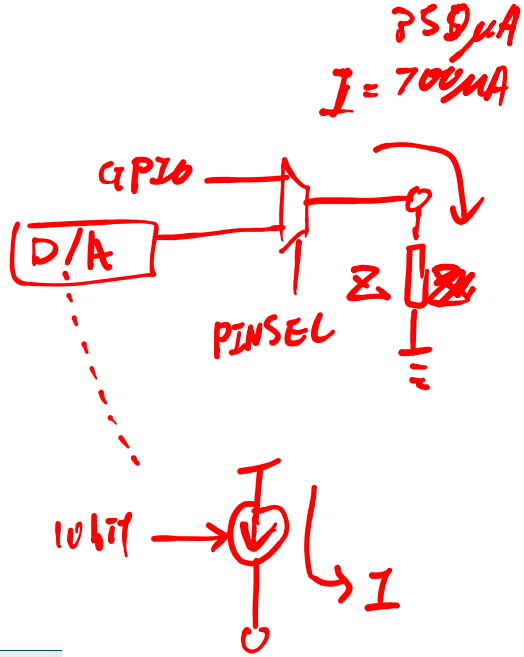
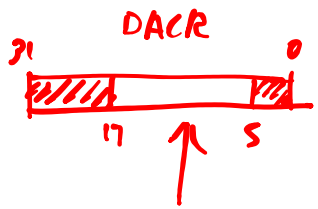


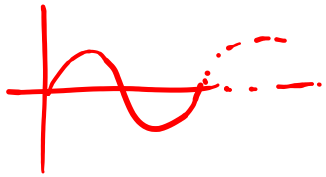
Table 599: D/A Converter Register (DACR - address 0xE006 C000) bit description

Bit	Symbol	Value	Description	Reset Value
5:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:6	VALUE		After the selected settling time after this field is written with a new VALUE, the voltage on the A_{OUT} pin (with respect to V_{SSA}) is VALUE/1024 × VREF.	0
16	BIAS	0	The settling time of the DAC is 1 μs max, and the maximum current is 700 μA.	0
		1	The settling time of the DAC is 2.5 μs and the maximum current is 350 μA.	
31:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA



Sine wave generator

- Can generate arbitrary waveforms (e.g. sine waves) using the D/A converter.
- General strategy:
  1. Create a table of sine wave values *f: time → value*.
  2. Build a loop counting from 0º to 359º to generate a complete sine wave.



Representing sine values

- Sine waves have fractional values between 0 ~ 1:
  sin (30) = 0.500000000000
  sin (1) = 0.017452406437
  sin (70) = 0.9396926207859

- **Q-notation** is a simple technique to represent fractions.

- Q-16 would use bits 15 – 0 to represent the fractional part and bits 31 – 16 for the integer part.

- Representing a number *x* using Q-notation:
  1. Multiply *x* by 2^m, where *m* is the Q-number.
  2. Convert the resulting integer part into 32-bit binary.

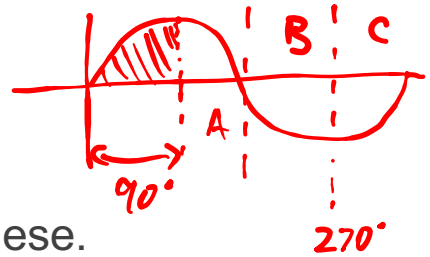
Representing sine values

- The integer part is never greater than 1 for our sine wave. So use Q-31, giving 31 bits to the fraction.
- sin(30) in Q-31:
  = 0.500000000000 * 2^{31}
  = 1,073,741,824
  = 0x40000000

- sin(1) in Q-31:
  = 0.017452406437 * 2^{31}

  = 37,478,757 (ignoring the decimals)
  = 0x023BE164

Sine wave generator



- Only store values for 0 ~ 90 º, other angles are symmetries of these.
- If register R1 has the angle, then:
  - 90 < R1 ≤ 180
    sin(R1) = sin(180 – R1) A
  - 180 < R1 ≤ 270
    sin(R1) = -sin(R1 – 180) B
  - 270 < R1 ≤ 360
    sin(R1) = -sin(360 – R1) C

- The sine routine:
  1. Uses a table of Q-31 values for angles ≤ 90º
  2. Compute all other values using the above relationship.
  3. Returns the sine value in register R0 for an angle given in R1.

A1 θ

Sine wave generator

```
sine_data   DCD 0X00000000, 0X023BE164, 0X04779630,0X06B2F1D8
            DCD 0X08EDC7B0, 0X0B27EB50, 0X0D613050, 0X0F996A30
            DCD 0X11D06CA0, 0X14060B80,0X163A1A80, 0X186C6DE0
            DCD 0X1A9CD9C0, 0X1CCB3220, 0X1EF74C00, 0X2120FB80
            DCD 0X234815C0, 0X256C6F80, 0X278DDE80, 0X29AC3780
            DCD 0X2BC750C0, 0X2DDF0040,0X2FF31BC0,0X32037A40
            DCD 0X340FF240, 0X36185B00,0X381C8BC0, 0X3A1C5C80
            DCD 0X3E17D980, 0X3E13D000, 0X40000000, 0X41EC1480
            DCD 0X43D46500, 0X45B6BB80, 0X4793A200, 0X496AF400
            DCD 0X4B3C8C00, 0X4D084600, 0X4EC0EF00, 0X508D9200
            DCD 0X5246DD00,0X53F9BE00,0X55A61280,0X574BB900
            DCD 0X58EA9100,0X5A827980,0X5C135380,0X5D9CFF80
            DCD 0X5F1F5F00, 0X60BA5E00, 0X6220B100, 0X63798500
            DCD 0X64DD8900,0X6639B080,0X678DDE80,0X68D9F980
            DCD 0X6A1DE700,0X6B598F00,0X6C8CD700,0X6DB7A880
            DCD 0X6ED9EC00, 0X6FF38A00, 0X71046D00, 0X720C8080
            DCD 0X730BAF00, 0X7401E500, 0X74EF0F00, 0X75D31A80
            DCD 0X76ADF600, 0X777F9000,0X7847D900,0X7906C080
            DCD 0X79BC3880, 0X7A683200, 0X7B0A9F80, 0X7BA5F580
            DCD 0X7C32A680, 0X7CB82880, 0X7D33F100, 0X7DA5F580
            DCD 0X7E0E2E00, 0X7E6C9280, 0X7EC11A80, 0X7F0BC080
            DCD 0X7F4C7E80, 0X7F834F00, 0X7FB02E00, 0X7FD31780
            DCD 0X7FEC0A00, 0X7FFB0280, 0X7FFFFFFF
```

32 bits

Sine wave generator

```
sine        MOV   r7, r1          ; copy for use later
            LDR   r5, =270        ; won't fit into rotation scheme (see next wk)
            ADR   r4, sine_data   ; addr of Q-31 table

            CMP   r1, #90         ; r1 <= 90 ?
            BLE   sine_ret
            CMP   r1, #180        ; 90 < r1 <= 180 ?
            RSBLE r1, r1, #180
            BLE   sine_ret
            CMP   r1, r5          ; 180 < r1 <= 270 ?
            SUBLE r1, r1, #180
            BLE   sine_ret
            RSB   r1, r1, #360    ; r1 > 270
sine_ret    LDR   r0, [r4, r1, LSL #2]  ; look up table
            CMP   r7, #180
            RSBGT r0, r0, #0      ; negation for > 180
```

Case 1

Case 2

Case 3

Case 4

Let R1 hold the angle, then:

R1 ≤ 90:
sin(R1)

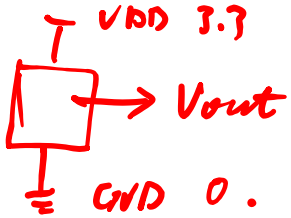
90 < R1 ≤ 180:
sin(180 – R1)

180 < R1 ≤ 270:
-sin(R1 – 180)

270 < R1 ≤ 360:
-sin(360 – R1)

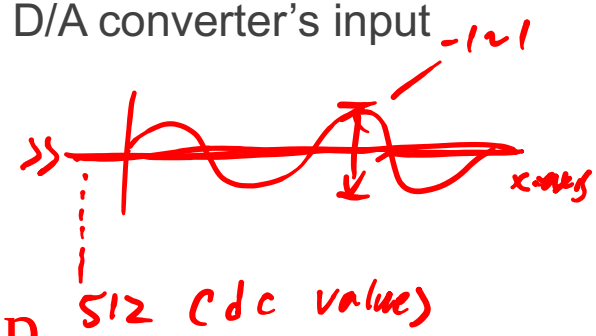
$0 - \theta = -\theta$

Sine wave generator



- The sine routine outputs have to be scaled and shifted to fit the D/A converter's input range (0 ~ 1023):

$$\begin{aligned}\text{DACR} &= 512 \times \sin(\text{R1}) + 512 \\ &= 512 \times \text{R0} + 512\end{aligned}$$



- With the sine value in R0, update D/A converter output:


```
DACR        EQU  0xE006C000
            LDR  r8, =DACR
            MOV  r0, r0, ASR #16    ; Q-31 → Q-15
            MOV  r0, r0, LSL #9     ; multiply by 512
            MOV  r0, r0, ASR #15    ; remove fraction (Q-15 → int)
            ADD  r0, r0, #512       ; 512*r0 + 512
            MOV  r0, r0, LSL #6     ; DACR[5:0] are reserved bits
            STRH r0, [r8]           ; write to DACR
```


half word.

Sine wave generator

```
PINSEL1     EQU  0xE002C004
DACR        EQU  0xE006C000
STACKST     EQU  0x40000200
            AREA SINEWAVE, CODE
            ENTRY
main        LDR sp, =STACKST
            LDR r6, =PINSEL1
            LDR r8, =DACR
            LDR r7, [r6]              ; config D/A
            ORR r7, r7, #1:SHL:21
            BIC r7, R7, #1:SHL:20
            STR r7, [r6]

outloop     MOV  r6, #360
inloop      RSB  r1, r6, #360
            BL   sine                 ; sine()
            MOV  r0, r0, ASR #16      ; to D/A
            MOV  r0, r0, LSL #9
            MOV  r0, r0, ASR #15
            ADD  r0, r0, #512
            MOV  r0, r0, LSL #6
            STRH r0, [r8]

            SUBS r6, r6, #1
            BNE  inloop
            B    outloop
```

```
sine        STMFD sp!, {r4, r5, r7, lr}
            MOV   r7, r1
            LDR   r5, =270
            ADR   r4, sine_data

            CMP   r1, #90
            BLE   sine_ret
            CMP   r1, #180
            RSBLE r1, r1,#180
            BLE   sine_ret
            CMP   r1, r5
            SUBLE r1, r1, #180
            BLE   sine_ret
            RSB   r1, r1, #360
sine_ret    LDR   r0, [r4, r1, LSL #2]
            CMP   r7, #180
            RSBGT r0, r0, #0
            LDMFD sp!, {r4, r5, r6, r7, pc}
done        B     done
```

Sine wave generator

```
sine_data   DCD 0X00000000, 0X023BE164, 0X04779630,0X06B2F1D8
            DCD 0X08EDC7B0, 0X0B27EB50, 0X0D613050, 0X0F996A30
            DCD 0X11D06CA0, 0X14060B80,0X163A1A80, 0X186C6DE0
            DCD 0X1A9CD9C0, 0X1CCB3220, 0X1EF74C00, 0X2120FB80
            DCD 0X234815C0, 0X256C6F80, 0X278DDE80, 0X29AC3780
            DCD 0X2BC750C0, 0X2DDF0040,0X2FF31BC0,0X32037A40
            DCD 0X340FF240, 0X36185B00,0X381C8BC0, 0X3A1C5C80
            DCD 0X3C17A500, 0X3E0E3DC0,0X40000000,0X41ECC480
            DCD 0X43D46500, 0X45B6BB80, 0X4793A200, 0X496AF400
            DCD 0X4B3C8C00, 0X4D084600, 0X4ECDFF00, 0X508D9200
            DCD 0X5246DD00,0X53F9BE00,0X55A61280,0X574BB900
            DCD 0X58EA9100,0X5A827980,0X5C135380,0X5D9CFF80
            DCD 0X5F1F5F00, 0X609A5280, 0X620DBE80, 0X63798500
            DCD 0X64DD8900,0X6639B080,0X678DDE80,0X68D9F980
            DCD 0X6A1DE700,0X6B598F00,0X6C8CD700,0X6DB7A880
            DCD 0X6ED9EC00, 0X6FF38A00, 0X71046D00, 0X720C8080
            DCD 0X730BAF00, 0X7401E500, 0X74EF0F00, 0X75D31A80
            DCD 0X76ADF600, 0X777F9000,0X7847D900,0X7906C080
            DCD 0X79BC3880, 0X7A683200, 0X7B0A9F80, 0X7BA5F580
            DCD 0X7C32A680, 0X7CB82880, 0X7D33F100, 0X7DA5F580
            DCD 0X7E0E2E00, 0X7E6C9280, 0X7EC11A80, 0X7F0BC080
            DCD 0X7F4C7E80, 0X7F834F00, 0X7FB02E00, 0X7FD31780
            DCD 0X7FEC0A00, 0X7FFB0280, 0X7FFFFFFF
```

This week

- GPIO intricacies
- UART
- D/A (digital-to-analog) converters

In Moodle:

- Start working on Lab
  (Due: end of your 3-hr lab)
- Start doing Week 8 exercise