ACSYS –REAL TIME

BINARY MANIPULATION

CLEARING WITH AND:

And a register with 1's where you want it remain the same, 0's where it wants to be cleared.

WRITING IN WITH AN OR:

Or the register with 0 where it wants to be unaffected, and 1's where you want to write.

MULTIPLICATION/ DIVISION WITH SHIFTING:

Left shift to multiply by powers of 2, eg (R<<1) is R times 2, (R<<2) is R times 4,
Right shift to divide by powers of 2, eg (R>>1) is Divided by 2, (R>>2) is R Divided by 4,

READING WITH AND

and-ing a register with a bit set to one, will give a one if that bit on the register is set.e.g
R&(1<<5) would = 1 iff R.5 =1.

INCLUDES

- #include <stdio.h> - *Standard Input output*
- #include <lpc24xx.h> - *Register definitions, and function for LPC24xx board.*
- #include <textDisplay.h> - *LCD Functions*

LCD

Lcd uses Timer 1 during start up

Start up function : textInit() or LCDInit()
Needs External Memory

GENERAL I/O PINS

5 Ports P0 – P4 (32 bits)

P0 is the joy stick

P3 LED's

Register	Function
FIOxDIR	1 in a bit position sets that i/o bit to be an output, 0 makes it an input
FIOxPIN	The current state of digital port pins can be read from this register, regardless of pin direction.
FIOxSET	This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect.
FIOxCLR	This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect.

PINSEL appears to make certain pins input/ output to other chips. Assuming it stands for pin-select

PCONP (PERIPHERAL CONTROL REGISTER)

sends power different chips using bits:

- 1 = Timer 0
- 2 = Timer 1
- 12 = ADC
- 22 = Timer 2
- 23 = Timer 3

JOYSTICK

```
#define UP (1<<10)
#define DOWN (1<<11)
#define LEFT (1<<12)
#define RIGHT (1<<13)
#define CENTRE (1<<22)
```

ADC (ANALOG TO DITAL CONVERTER)

AD0CR = The Control Register

AD0CR:

0	0	0	0	0	START	0	0	P	0	0	0	0	0	CLKDIV	SEL

Bits	Function
START	001 = start a single conversion now
P	1 = enable ADC, 0 = power down ADC
CLKDIV	divider for main clock which sets ADC clock frequency (see below)
SEL	channel select (0 to 7) bits, use 00000010 for channel 1

Using it:

- Set PINSEL1 bits 16&17 to 01
- Turn on using PCONP
- AD0CR = 0 (clear ADC)
- AD0CR |= 2; (set channel to 1)
- AD0CR |=0x00000300; (Set the divider for the main clock, must >= 3)
- AD0CR |=0x1000000; (Start the ADC)
- AD0DR1.31 1
- Read AD0DR1.6-AD0DR1.16 for value

DAC (DIGITAL TO ANALOG CONVERTER)

DACR = DAC Control Register

Using it:

- Set PINSEL1 bits 21&20 to 10
- To output write to 10 bits to: DACR.6-DACR.16
- Voltage between 0-1023 (0 – 0x3FF)

TIMERS

4 Timers: T0/1/2/3/

Timer 1 is used by the LCD during start up

Timer 0 can be connected to the motor

Timer 0 & 1 are connected to the VIC, which means they can call Interrupt Service Routine

TxTCR is the Timer control register for Timer x

- TxTCR.0 = Enable
- TxTCR.1 Reset

TxMRy is Match Register y(this can be 0,1,2 or 3) for Timer x

- This sets values for the timer to count upto, what happens when these values are reached can be controlled by the TxMCR register

TxPR is the prescaler register for Timer x

- Setting PR to any number higher than 0 will cause it to send a pulse to the MCR each time it counts to that value. Eg
- PR =0 one pulse, per clock pulse
- PR = 2 one pulse per three clock pulses

TxMCR is the match control register it tells the timer what to do when a value in MR0, MR1, MR2 or MR3

TxMCR:

0	0	0	0	MR3S	MR3R	MR3I	MR2S	MR2R	MR2I	MR0S	MR1R	MR1I	MR0S	MR0R	MR0I

The “I” bits (MR0I, MR1I etc) allow an interrupt on match when set to 1.

The “R” bits (MR0R, MR1R etc) cause the count value to be reset on match when set to 1.

The “S” bits (MR0S, MR1S etc) cause the counter to stop on match when set to 1.

TxIR is the interrupt Register for Timer x

X	X	X	X	MR3	MR2	MR1	MR0

- These bits will become set to 1 if the value in MR0/1/2/3 becomes set, and TxMCR is set to cause an interrupt. They can be cleared by writing 1’s where they want to be reset.

Using it:

- Turn on using PCONP
- TxTCR =2;(stop and reset timer)
- Clear Interrupt Register (TxIR = 0xff)
- Set amount of pulses to count to TxMR0 =AMOUNT
- set interrupt /reset/stop flags using TxMCR
- start timer using TxTCR =1;
- Wait for an interrupt (catch this with an Interrupt service routine, or you can just use a while loop)
- Clear irrupt flags if you want to use again.

PWM (PULSE WIDTH MODULATOR) TO CONTROLA MOTOR.

Set PINSEL2.6 & 7 to 1 to give P1.3 the PWM function, this connects PWM0[2] to the motor.

Tell the PWM to output on PWM0[2] , by setting PWM peripheral control register , PWM0PCR.10 to 1

PWM0MR0 is the PWM0 Match register it will count to this value then output HIGH/ Binary 1 , and reset.

PWM0MR2 is the PWM0 Width register, this is how long after the PWM goes high, it will stay high

PWM0TCR is the timer control register for PWN0:

- bit 0 starts the timer
- bit 3 enables the PWM
- PWM0TCR = 9; // sets bits 0 and 3

To change the value in PWM0MR0 or PWM0MR2 set the register to the new values. Then in the PWM0LER register set bit

- 0 to update MR2 on the next cycle
- 2 to update MR0 on the next cycle

INTERRUPTS (VECTORED INTERRUPT CONTROLLER)

Runs ISR's when certain interrupt flags are set.

VIC addresses are connected to certain things.

- Timer0 – VIC input 4
- Timer1 – VIC input 5

Setting a ISR for VIC address x

- VICIntSelect &= ~(1<<x);
- VICVectAddrx = (unsigned int)nameofISR;
- VICIntEnable |= (1<<x);

ISR prototypes much have the following form:

```
void yourISR(void) __attribute__ ((interrupt));
```