Chapter 15: Hardware Insights: Clocks, Resets and Power Supply

tw rev. 17.11.16

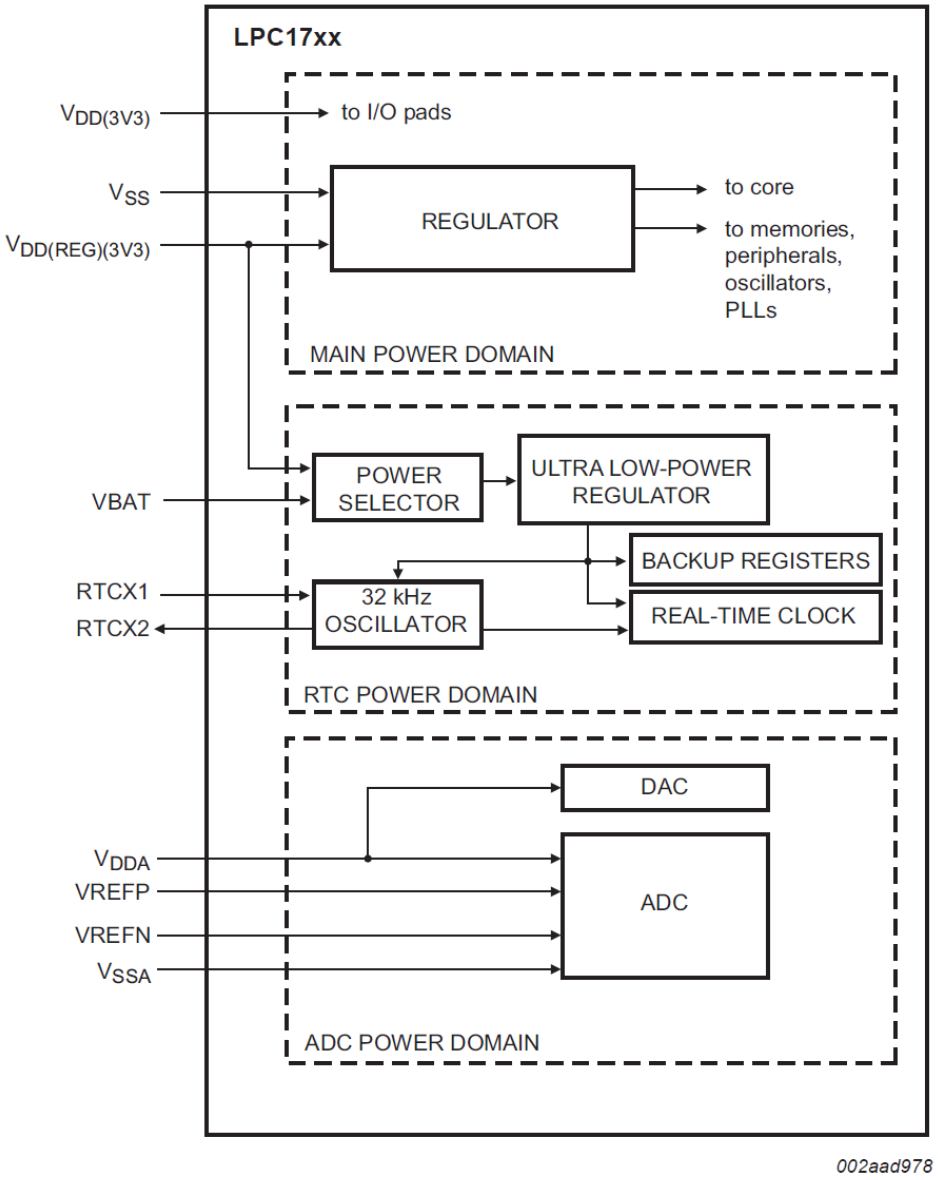
If you use or reference these slides or the associated textbook, please cite the original authors' work as follows:

Toulson, R. & Wilmshurst, T. (2016). Fast and Effective Embedded Systems Design - Applying the ARM mbed (2nd edition), Newnes, Oxford, ISBN: 978-0-08-100880-5.

Power Supply and the LPC1768

This figure shows the internal power distribution of the LPC1768. Locate these:

- two regular power supply inputs, labelled $V_{DD(3V3)}$ and $V_{DDREG(3V3)}$;
- V_{BAT}, which can be connected to an external battery to sustain only the Real Time Clock and backup registers;
- Supplies to ADC and DAC, V_{DDA} and V_{SSA} ;
- A voltage reference for ADC and DAC, V_{REFP} and V_{REFN}.



Power Supply and the LPC1768

The supply voltage requirements of the LPC1768 are shown. All are centred around 3.3 V, with V_{BAT} having the lowest permissible value at 2.1 V. All can go up to 3.6 V, except for the ADC positive reference voltage, which must not exceed the ADC supply voltage.

Table 8. Static characteristics

T_{amb} = −40 °C to +85 °C, unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ[1]	Max	Unit
Supply pins							
$V_{DD(3V3)}$	supply voltage (3.3 V)	external rail	[2]	2.4	3.3	3.6	V
$V_{DD(REG)(3V3)}$	regulator supply voltage (3.3 V)			2.4	3.3	3.6	V
V_{DDA}	analog 3.3 V pad supply voltage		[3][4]	2.5	3.3	3.6	V
$V_{i(VBAT)}$	input voltage on pin VBAT		[5]	2.1	3.3	3.6	V
$V_{i(VREFP)}$	input voltage on pin VREFP		[3]	2.5	3.3	V_{DDA}	V

Supply Voltage Requirements for the LPC1768

Power Supply and the mbed

The Figure shows the LPC1768 power and clock connections within the mbed. Each connection shows the microcontroller pin number, and the name of the signal.

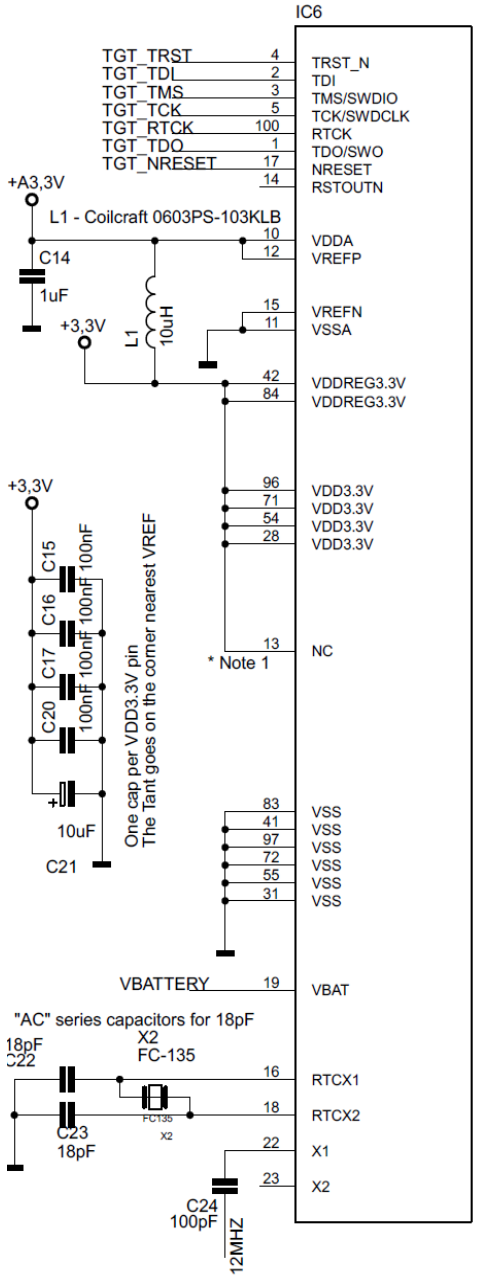
A complex integrated circuit tends to have more than one ground connection, and similar multiple power supply connections. This is because the interconnecting wires inside the IC are so very thin that they can have significant resistance.

There are six ground connections, labelled V_{SS}.

There are four $V_{DD(3V3)}$ connections and two for $V_{DDREG(3V3)}$. The mbed designers don't differentiate between these two, they just join them together.

The V_{BAT} connection is kept separate, and *can* be supplied via pin 3, V_B, of the mbed.

The V_{DD} supply is smoothed by the distributed capacitors C_{15} – C_{17} and C_{20}.



LPC1768 internal power/ clock connections on the mbed

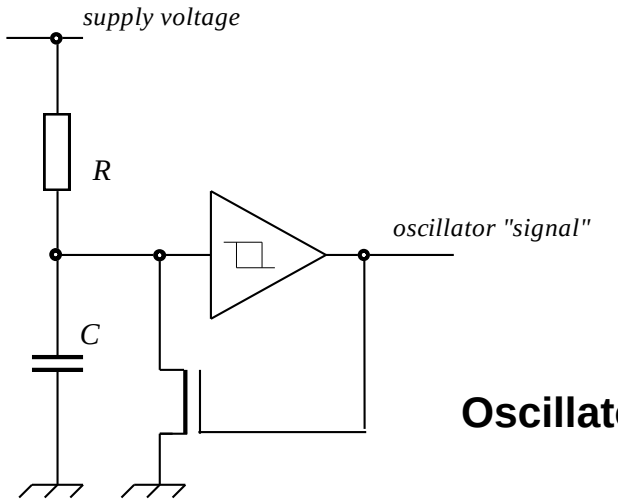
Clock Sources and their Selection

An essential part of the microcontroller system is the clock oscillator, a continuous square wave which drives forward most microcontroller action. It is also the basis of many time measurement or generation activities.

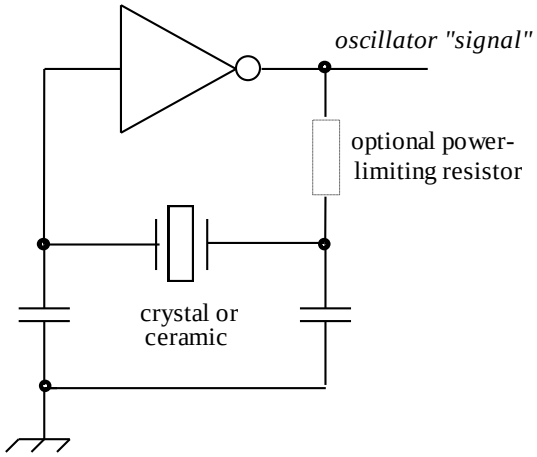
Clock oscillators can be based on resistor-capacitor (R-C) networks, or ceramic or crystal resonators; the designer should be aware of their relative advantages.

A popular R-C oscillator circuit is shown below left. The R-C circuit is the cheapest form of clock oscillator available and is widely used.

A quartz *crystal oscillator* is based on a very thin slice of crystal. This is piezoelectric. If electrical terminals are deposited on opposite surfaces of the crystal, then due to its piezoelectric property vibration can be induced and sustained electrically. A suitable circuit is shown below right.



Oscillator circuits



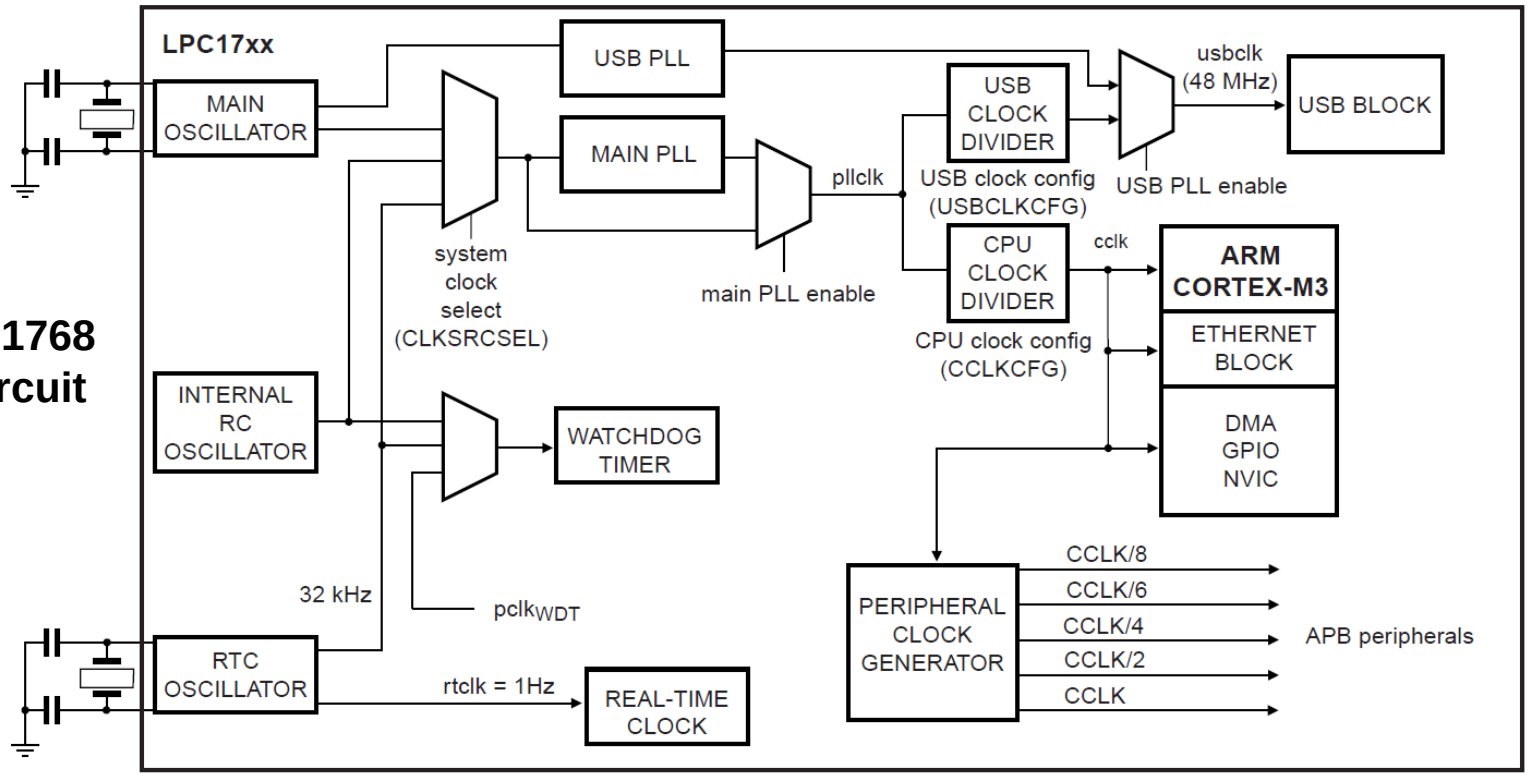
LPC1768 Clock Oscillators and the mbed Implementation

The Figure shows how clock sources within the LPC1768 are distributed.

There are three sources, seen to the left of the diagram:

- the main oscillator (an external crystal), which can operate between 1 and 25 MHz. It can also act in *slave mode* - an external clock signal can be connected via a capacitor to the XTAL1 pin;
- the internal R-C oscillator, running at a nominal frequency of 4 MHz;
- the RTC oscillator - another crystal, but of low frequency.

The LPC1768 clock circuit



Adjusting the Clock Configuration Register

The CPU Clock Divider block, seen in the previous Figure, is controlled by the Clock Configuration register **CCLKCFG,** seen below.

The frequency of **pllclk** is divided by the number held in the lower 7 bits of this register, plus one; this produces **cclk**.

Table 38. CPU Clock Configuration register (CCLKCFG - address 0x400F C104) bit description

Bit	Symbol	Value	Description	Reset value
7:0	CCLKSEL		Selects the divide value for creating the CPU clock (CCLK) from the PLL0 output.	0x00
		0 to 1	Not allowed, the CPU clock will always be greater than 100 MHz.	
		2	PLL0 output is divided by 3 to produce the CPU clock.	
		3	PLL0 output is divided by 4 to produce the CPU clock.	
		4	PLL0 output is divided by 5 to produce the CPU clock.	
		:	:	
		255	PLL0 output is divided by 256 to produce the CPU clock.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

LPC1768 Clock Configuration register CCLKCFG

Adjusting the Clock Configuration Register

```
/*Program Example 15.1 Adjusts clock divider through register CCLKCFG,
with trial blinky action                                    */

#include "mbed.h"     //keep this, as we are using DigitalOut
DigitalOut myled(LED1);
#define CCLKCFG (*(volatile unsigned char *)(0x400FC104))

// function prototypes
void delay(void);

int main() {
  CCLKCFG=0x00000005; // divider divides by this number plus 1
  while(1) {
    myled = 1;
    delay();
    myled = 0;
    delay();
  }
}
void delay(void){           //delay function.
  int j;                      //loop variable j
  for (j=0;j<5000000;j++) {
    j++;
    j--;                   //waste time
  }
}
```

This Program Example approximately replicates the original “blinky” program, using its own **delay( )** function. It starts by resetting the value of **CCLKCFG**, so the program then runs with a changed clock frequency.

Adjusting the Phase Locked Loop

The phase locked loop is a circuit which can multiply frequencies.

The main PLL of the LPC1768, PLL0, is actually made up of a divider followed by the PLL. Hence it can divide frequencies as well as multiply them.

Different combinations of multiply and divide give a huge range of possible output frequencies, which can be extremely useful in some situations.

A PLL needs to “lock” to an incoming frequency. However, it only locks if conditions are right, and it may take finite time to do this. These conditions include the requirement that the input to PLL0 must be in the range 32 kHz to 50 MHz. It’s interesting to see therefore that the PLL can be used to multiply up the RTC frequency if required.

Full use of the PLL0 sub-system is complex, and requires a very careful reading of relevant sections of the LPC1768 user manual.

Adjusting the Phase Locked Loop

The PLL is controlled by four registers, outlined in the Table. The PLL is enabled and connected through **PLL0CON**, with multiply and divide values set through **PLL0CFG**.

The Feed Register is a safety feature which blocks accidental changes to **PLL0CON** and **PLL0CFG**. A valid “feed sequence” is required before any update can be configured.

Once implemented, changes can be tested in **PLL0STAT**. This further carries the important **PLOCK0** bit, which tests whether successful lock has been achieved.

Name	Description	Access	Address
PLL0CON	Control Register. Holding register for updating PLL0 control bits. Values written to this register do not take effect until a valid PLL0 feed sequence has taken place. There are only 2 useful bits: bit 0 to enable; PLL0, bit 1 to connect. Connection must only take place after the PLL is enabled, configured, and locked.	Read/Write	0x400F C080
PLL0CFG	Configuration Register. Holding register for updating PLL0 configuration values. Bits 14:0 hold the value for the frequency multiplication, less one; Bits 23:16 hold the value for the pre-divider, less one. Values written to this register do not take effect until a valid PLL0 feed sequence has taken place.	Read/Write	0x400F C084
PLL0STAT	Status Register. Read-back register for PLL0 control and configuration information. If PLL0CON or PLL0CFG have been written to, but a PLL0 feed sequence has not yet occurred, they will not reflect the current PLL0 state. Reading this register provides the actual values controlling PLL0, as well as the PLL0 status. Bits 14:0 and bits 23:16 reflect the same multiply and divide bits as in PLL0CFG. Bits 24 and 25 reflect the two useful bits of PLL0CON. When either is zero, PLL0 is bypassed. When both are 1, PLL0 is selected. Bit 26, PLOCK0, gives the lock status of the PLL.	Read Only	0x400F C088
PLL0FEED	Feed Register. Correct use of this register enables loading of the PLL0 control and configuration information from the PLL0CON and PLL0CFG registers into the shadow registers that actually affect PLL0 operation. The required feed sequence is 0xAA followed by 0x55.	Write Only	0x400F C08C

PLL0 Control Registers

Adjusting the Phase Locked Loop

This program illustrates PLL0 control. The **main( )** function immediately disconnects and turns off the PLL, “feeding” the PLL control as required. The mbed then runs with the PLL disabled and bypassed.

```
/*Program Example 15.2 Switches off PLL0, with blinky action
                                                    */
#include "mbed.h"
DigitalOut myled(LED1);
#define CCLKCFG (*(volatile unsigned char *)(0x400FC104))
#define PLL0CON (*(volatile unsigned char *)(0x400FC080))
#define PLL0FEED (*(volatile unsigned char *)(0x400FC08C))
#define PLL0STAT (*(volatile unsigned int *)(0x400FC088))

// function prototypes
void delay(void);

int main() {
  // Disconnect PLL0
  PLL0CON &= ~(1<<1); // Clears bit 1 of PLL0CON, the Connect bit
  PLL0FEED = 0xAA;    // Feed the PLL. Enables action of above line
  PLL0FEED = 0x55;    //
  // Wait for PLL0 to disconnect. Wait for bit 25 to become 0.
  while ((PLL0STAT & (1<<25)) != 0x00);//Bit 25 shows connection status
  // Turn off PLL0; on completion, PLL0 is bypassed.

  //continued over…
```

Adjusting the Phase Locked Loop

```
  PLL0CON &= ~(1<<0); //Bit 0 of PLL0CON disables PLL
  PLL0FEED = 0xAA;    // Feed the PLL. Enables action of above line
  PLL0FEED = 0x55;
  // Wait for PLL0 to shut down
  while ((PLL0STAT & (1<<24)) != 0x00);//Bit 24 shows enable status

  /****Insert Optional Extra Code Here****
        to change PLL0 settings or clock source.
  **OR** just continue with PLL0 disabled and bypassed*/

  //blink at the new clock frequency
  while(1) {
    myled = 1;
    delay();
    myled = 0;
    delay();
  }
}

void delay(void){          //delay function.
  int j;                     //loop variable j
  for (j=0;j<5000000;j++) {
    j++;
    j--;                  //waste time
  }
}
```

Adjusting the Phase Locked Loop

This code fragment can be inserted in the previous program to introduce clock frequency changes with PLL0.

```
// Set PLL0 multiplier
   PLL0CFG = 07;  //arbitrary multiply value, divide value left at 1
   PLL0FEED = 0xAA; // Feed the PLL
   PLL0FEED = 0x55;
   // Turn on PLL0
   PLL0CON |= 1<<0;
   PLL0FEED = 0xAA; // Feed the PLL
   PLL0FEED = 0x55;
   // Wait for main PLL (PLL0) to come up
   while ((PLL0STAT & (1<<24)) == 0x00);
   // Wait for PLOCK0 to become 1
   while ((PLL0STAT & (1<<26)) == 0x00);
   // Connect to the PLL0
   PLL0CON |= 1<<1;
   PLL0FEED = 0xAA; // Feed the PLL
   PLL0FEED = 0x55;
   while ((PLL0STAT & (1<<25)) == 0x00); //Wait for PLL0 to connect
```

Selecting the Clock Source

If the clock source is to be changed, through the input multiplexer, it must be done with PLL0 shut down. This change is controlled by the Clock Source Select Register, **CLKSRCSEL,** with details shown.

Table 17. Clock Source Select register (CLKSRCSEL - address 0x400F C10C) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CLKSRC		Selects the clock source for PLL0 as follows:	0
		00	Selects the Internal RC oscillator as the PLL0 clock source (default).	
		01	Selects the main oscillator as the PLL0 clock source. **Remark:** Select the main oscillator as PLL0 clock source if the PLL0 clock output is used for USB or for CAN with baudrates > 100 kBit/s.	
		10	Selects the RTC oscillator as the PLL0 clock source.	
		11	Reserved, do not use this setting.	
		Warning: Improper setting of this value, or an incorrect sequence of changing this value may result in incorrect operation of the device.		
31:2	-	0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Clock Source Select Register, CLKSRCSEL

Questions from the Quiz

1. Name two advantages and two disadvantages of R-C and quartz oscillators.
2. Following reset, the LPC1768 always starts running from the internal R-C oscillator. Why is this?
3. In a certain application, the main oscillator in an LPC1768 application is running at 18.000 MHz, the PLL multiplies by 8, and the lower 7 bits of register **CCLKCFG** are set to 5. What is the frequency of **cclk**?

Reset

At certain times, any microcontroller needs to start its program from the beginning. At this moment it also needs to put all of its control registers into a known state, so that peripherals are safe and initially disabled. This “ready-to-start” condition is called *reset*.

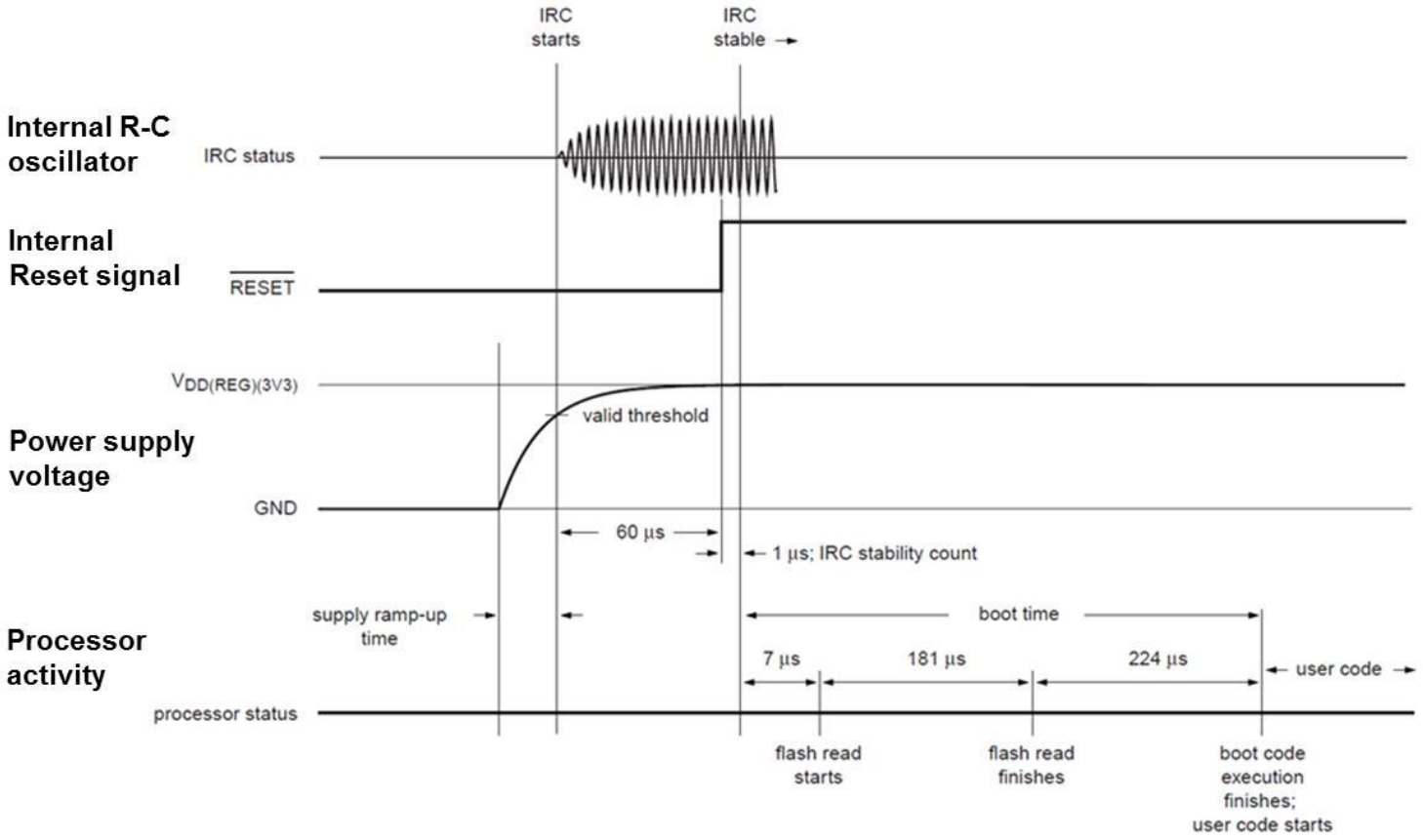
The CPU starts running its program when it leaves the reset condition.

In an advanced processor like the LPC1768, the user code is preceded by some “boot code”, hard-wired into the processor, which undertakes preliminary configuration.

Apart from power-up, there are other times when reset is needed, including the possibility that the user may want to force a reset if a system locks or crashes.

Power-on Reset

The moment that power is applied is a critical one for any embedded system. Both the power supply and the clock oscillator take finite time to stabilise, and in a complex system power to different parts of the circuit may stabilise at different times.



LPC1768 internal signals during start-up

Other Sources of Reset

External reset

The LPC1768 has an external reset input. As long as this is held low, the microcontroller is held in reset. When it is taken high, a sequence is followed very similar to the power-on reset described above.

The mbed has a reset button. This is however connected to the interface microcontroller. This can force a reset to the LPC1768 itself.

Watchdog timer

A common failure mode of any computer-based system is for the computer to lock up. For most embedded systems this is unacceptable.

An uncompromising solution is the Watchdog Timer (WDT), which *resets the processor if the WDT is ever allowed to overflow*. The WDT runs continuously, counting towards its overflow value.

The programmer must ensure that this overflow doesn't happen. This is done by including periodic WDT resets throughout the program.

If the program crashes, then the WDT overflows, the controller resets, and the program starts from its very beginning.

A WDT overflow causes a reset very similar to the power-on reset described above.

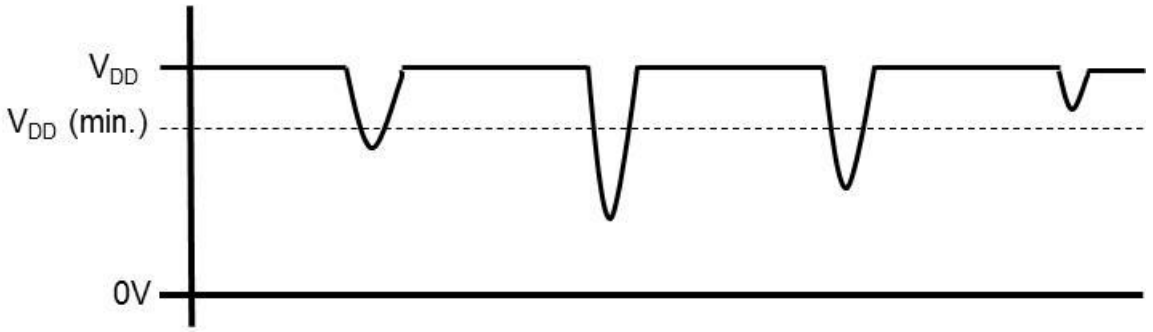
Other Sources of Reset

Brownout detect

An awkward failure condition for an embedded system is when the power dips, and then returns to normal. This is called a *brownout*, and is illustrated below. A brownout won't be detected as a full loss of power, and may not be noticed at all. However that momentary loss of full power could cause partial system failure.

Like many microcontrollers, the LPC1768 has a brownout detect capability, which must be enabled by the user. It is not normally enabled in the mbed.

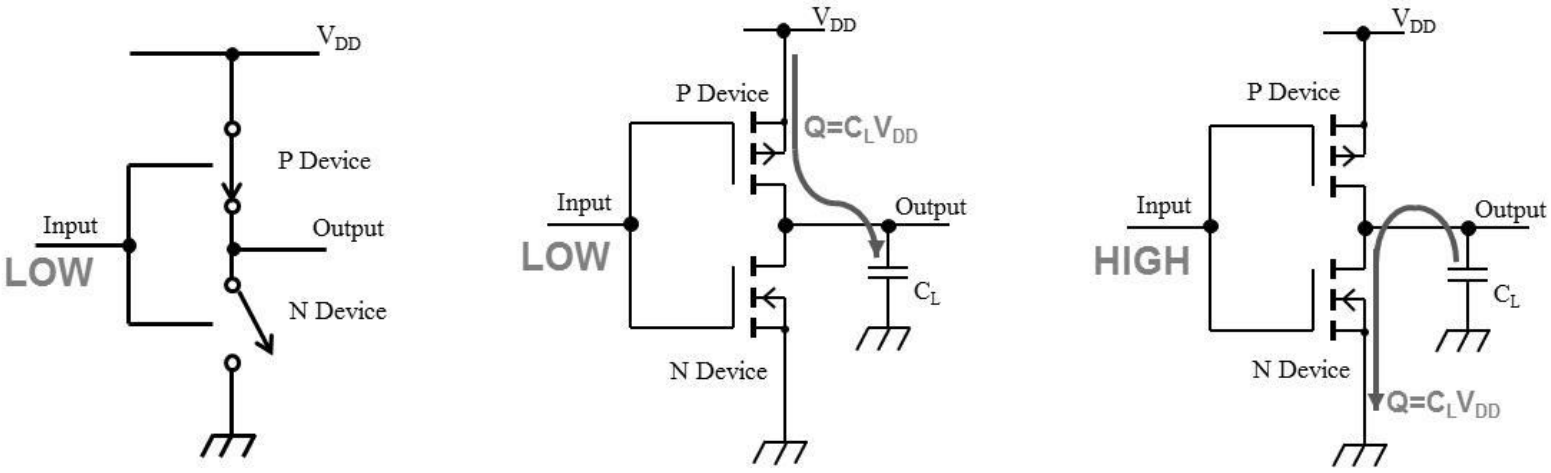
Detection of brownout, when enabled, causes a reset very similar to the power-on reset described above.



Voltage "brownouts"

How Power is Consumed in a Digital Circuit

Complementary Metal Oxide Semiconductor (CMOS) digital technology is the basis for the mobile phone, laptop, and any other portable electronic device. To understand how to minimise the power consumption these devices, it is useful to have some understanding as to how CMOS consumes power.



(a) Idealised circuit **(b) actual circuit, input going low** **(c) actual circuit, input going high**

Power consumption in a CMOS inverter (Eq. 15.1)

$$I_T = I_Q + \{ V_{DD} \times f\, C_{eq} \}$$

I_T : total current taken
f : switching frequency
V_{DD} : supply voltage

I_Q : quiescent current (due to leakage in the device)
C_{eq} equivalent capacitance of load and “shoot through” effect

Cells and Batteries

Cells are classified either as primary (non-rechargeable), or as secondary (rechargeable).

They are based on a variety of metal/chemical combinations, each of which has special characteristics in terms of energy density, whether rechargeable or not, and other electrical characteristics.

The two most important electrical characteristics of a cell are terminal voltage and capacity; the latter generally measured in Amp-hours (Ah), or milliamp hours (mAh). The Table gives some examples.

Manufacturer	Technology	Shape/ Package	Nominal Terminal Voltage (V)	Capacity (mAh)
Varta	Silver Oxide	V301, Button	1.55	96
Varta	Silver Oxide	V303, Button	1.55	160
Procell	Alkaline	AAA cylinder	1.5	1175
Procell	Alkaline	AA cylinder	1.5	2700
Procell	Alkaline	PP3	9.0	550

Questions from the Quiz

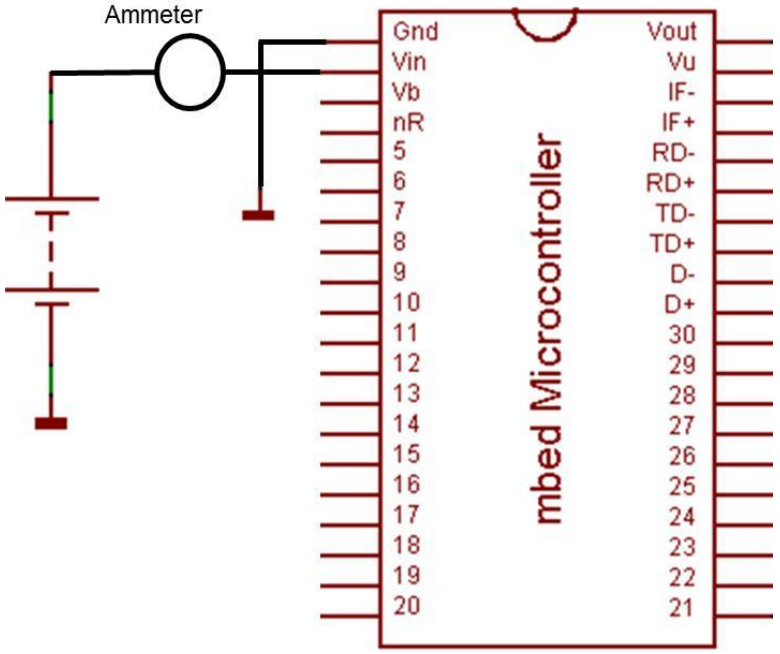
4. A certain logic circuit is powered from 3.0 V. It has a quiescent current of 120 nA, and an “equivalent capacitance” in the circuit of 56 pF. Applying Equation 15.1, what is its current consumption when the clock frequency is 1 kHz, 1 MHz, and when it is not clocked at all?

5. An mbed is found to draw 140 mA, when powered from 4 AAA cells in series, each of capacity 1175 mAh. Approximately how long will the cells last if they run continuously?

8. Power consumption to a digital circuit is being carefully monitored. It is found that connecting a long cable to a digital interface marginally increases the power consumption, even though nothing is connected at the far end of the cable. Why is this?

Exploring mbed Power Consumption

The mbed was not designed for low power, yet it provides a good opportunity to study power supply in an embedded system, and to explore ways of reducing that power. It's interesting to set up a simple current measurement circuit as shown, and monitor current in the different examples and exercises.



(Fig. 3.13)

LPC1768 Current Consumption

The current consumption characteristics of the LPC1768 are shown, along with the LPC1769, which can run at a slightly higher clock frequency. The Table also refers to Sleep and Power-down modes that we shall meet soon.

Table entries reinforce the message that clock frequency dominates current consumption.

The mbed runs with a clock frequency of 96 MHz, with PLL enabled. Hence we estimate that the LPC1768 on the mbed takes a supply current just under 42 mA, say 40 mA. But the values shown are with all peripherals disabled, so in practice the consumption will be higher.

$I_{DD(REG)(3V3)}$	regulator supply current (3.3 V)	active mode; code `while(1){}` executed from flash; all peripherals disabled; PCLK = $^{CCLK}/_{8}$				
		CCLK = 12 MHz; PLL disabled	[6][7]	-	7	- mA
		CCLK = 100 MHz; PLL enabled	[6][7]	-	42	- mA
		CCLK = 100 MHz; PLL enabled (LPC1769)	[6][8]	-	50	- mA
		CCLK = 120 MHz; PLL enabled (LPC1769)	[6][8]	-	67	- mA
		sleep mode	[6][9]	-	2	- mA
		deep sleep mode	[6][10]	-	240	- µA
		power-down mode	[6][10]	-	31	- µA
		deep power-down mode; RTC running	[11]	-	630	- nA
I_{BAT}	battery supply current	deep power-down mode; RTC running				
		$V_{DD(REG)(3V3)}$ present	[12]	-	530	- nA
		$V_{DD(REG)(3V3)}$ not present	[13]	-	1.1	- µA

LPC1768/9 current consumption characteristics (Table 15.5)

Reducing mbed Power Consumption: Switching Unwanted Things Off

A brief glance at the mbed block diagram reminds you how much there is in the overall mbed circuit. All of this is continuously powered, whether you use it or not.

This program explores aspects of managing mbed power consumption. It uses the **EthernetPowerControl** library file, imported from the mbed Cookbook site. The **PCONP** register is used to switch on or off individual peripherals in the LPC1768 itself.

```
/*Program Example 15.4
Powers down certain elements of the mbed, when not in use.
 */

#include "mbed.h"
 //import next from mbed site
#include "PowerControl/EthernetPowerControl.h"

DigitalOut myled1(LED1);
DigitalOut myled4(LED4);
#define PCONP           (*(volatile unsigned long *)(0x400FC0C4))

Ticker blinker;
void blink() {
  myled1=!myled1;
  myled4=!myled4;
}
 //continued over…
```

Switching Unwanted (mbed) Things Off

```
int main() {
  myled1=!myled4;
  PHY_PowerDown();  //**comment this in and out

  //Turn all peripherals OFF, except repetitive interrupt timer,
  PCONP = 0x00008000;         //which is needed for Ticker

  blinker.attach(&blink, 0.0625);
  while (1) {
    wait(1);
  }
}
```

Reducing mbed Power Consumption: Manipulating the clock frequency

We saw earlier that the LPC1768 has extensive capabilities to vary the clock frequency, e.g. through the **CCLKCFG** and **PLL0CFG** registers already shown.

This means we can trade off speed of execution with power consumption. A program with significant computational demands will need to run fast, and will consume more power; one with low computational demands can run slowly, and consume less power.

It is attractive to imagine that clock speeds can be reduced at will to reduce current consumption. While this is true in principal, take care! Many of the peripherals depend on that clock frequency as well, along with their mbed API libraries, for example for the setting of a serial port bit rate.

LPC1768 Low Power Modes

Aside from varying clock frequency, there are other, even more effective techniques. These involve either switching off the clock altogether at times when nothing needs doing, or switching it off to certain parts of a microcontroller.

The LPC1768 uses the modes summarised below, with power consumptions given in the earlier Table.

Sleep mode: The clock to the core, including the CPU, is stopped, so program execution stops. Peripherals can continue to function. Exit from this mode can be achieved by an enabled Interrupt, or a Reset.

Deep Sleep mode: Here the main oscillator and PLL0 are powered down, and the output of the internal R-C oscillator disabled, so no internal clocks are available. The internal R-C oscillator can continue running, and can run the WDT, which can cause a wakeup. The 32 kHz RTC continues, and can generate an interrupt. Wake-up is by Reset, by the RTC, or by any other interrupt which can function without a clock.

Power Down mode: This is similar to Deep Sleep, but the internal R-C oscillator and flash memory are turned off. Wake-up time is thus a little longer.

Deep Power Down mode: In this mode all power is switched off, except to the RTC. Wake-up is only through external Reset, or from the RTC.

Questions from the Quiz

7. Propose situations where each of the LPC1768 low power modes (Sleep, Deep Sleep and so on) can be used effectively.

10. A designer wishes to estimate power consumption characteristics of a new product based on the LPC1769, and applies the current consumption data of Table 15.5 (see earlier slide). A 3.3 V battery is available, with capacity 1200 mAh. She anticipates a behaviour whereby the processor will need to wake once per minute to perform a task, made up of two parts. In the first part, the processor must run with a clock speed of 120 MHz, PLL enabled for 200 ms. It then runs at 12 MHz, PLL disabled, for 400 ms. For the rest of the time it is in power-down mode. For the purposes of this question, the current taken by the rest of the circuit can be assumed to be negligible.

    a. Estimate the average current drawn from the battery.

    b. Estimate the battery life.

the EFM32 Zero Gecko Starter Kit

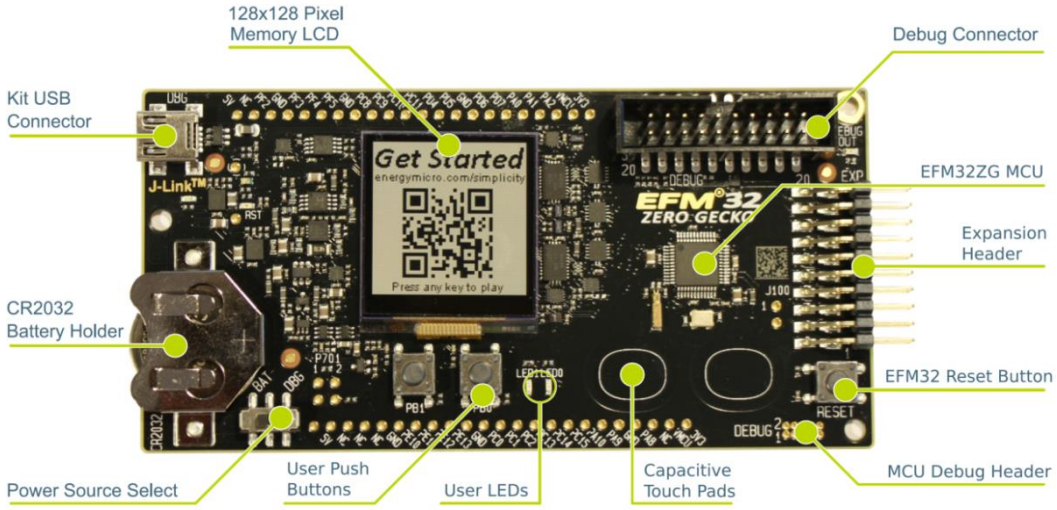
There are a number of versions of the ARM Cortex core, the LPC1768 mbed uses the M3.

The simplest are the M0 or M0+ cores. These are specifically designed for small scale, low power and low cost applications.

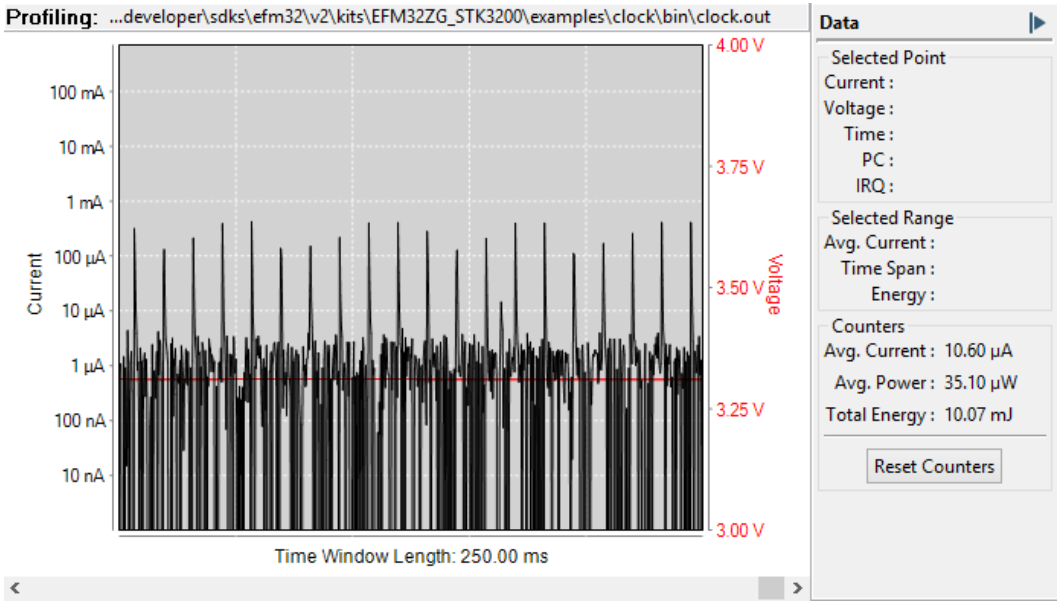
The EFM Zero Gecko Starter Kit is a development board based around the Silicon Labs Zero Gecko microcontroller.

It forms a useful example of a truly low-power mbed-enabled design. It is based on an M0+ core, and is designed for extreme low power.

Of great interest to us, it has an on-board *Energy Profiler*, which can measure current consumption from 0.1 uA to 50 mA.



The EFM32 Zero Gecko Starter Kit



The EFM32 Zero Gecko Energy Profiler

Chapter Review

- The LPC1768 has clearly defined power supply demands, in terms of supply voltages and currents, with the potential for power-conscious applications.
- The mbed designers have satisfied these power supply requirements, but without the intention of minimising power consumption.
- A range of clock sources are available to the LPC1768 and other microcontrollers; they can be applied in different ways and for different purposes, and their frequency multiplied or divided.
- The mbed power consumption can be improved, by switching off unnecessary circuit elements, adjusting clock frequency, and applying the special low power modes, e.g. *Sleep* and *Deep Sleep.* Even with this optimisation, current consumption remains too high for effective battery-powered applications.
- For competitive low power design, circuits must be designed specifically and rigorously for that purpose. The EFM32 Zero Gecko is one example where this has been achieved, to good effect.