CLOCK

Primary (crystal) oscillator

The clock module in the PIC24FJ64GA002 chip has the versatility of providing any of six clock sources to clock the CPU. For example, an 8 MHz crystal can be connected to the OSCI and OSCO pins to provide the 100 parts-per-million frequency accuracy of a 50¢ crystal. In addition, a 27 pF capacitor must be connected between OSCI and ground and another one between OSCO and ground. Finally, an XT (3-10 MHz) or HS 10-32 MHz) configuration selection must be made at the top of the user program for a crystal frequency between 3 MHz and 32 MHz. For the 8 MHz crystal, the format is:

```
_CONFIG2(POSCMOD_XT);
```

Figure CLOCK-1 PIC24F clock module

CPU clock

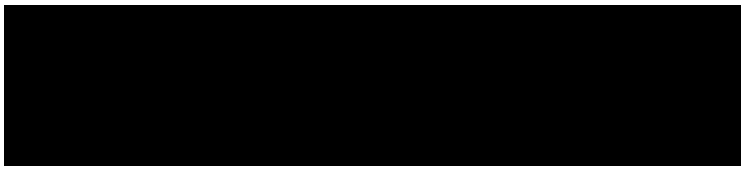
The CPU makes use of four quarter cycles as it executes most instructions in just one CPU cycle. These quarter cycles break execution into four parts:

- First quarter cycle - decode instruction to figure out how to use remaining three quarter cycles
- Second quarter cycle - if a read from RAM or from a register is required, do it now
- Third quarter cycle - if what was read requires manipulation, do it now
- Fourth quarter cycle - if a write to RAM or to a register is required, do it now

For example, the read-modify-write instruction to set bit 15 of PORTB will read the entire PORTB into a hidden CPU register during the second quarter cycle. It will set bit 15 of the hidden register during the third quarter cycle. It will copy the content of the hidden register back to PORTB during the fourth quarter cycle.

Given this need for quarter-cycle timing signals, the frequency obtained from one of the oscillators of Figure CLOCK-1 must be divided into four parts.
To do this, the oscillator frequency is divided by two. Then the original oscillator square wave, Fosc, and the divided-by-two square wave, Fcpu, can be gated together to obtain the four quarter cycles, as shown in Figure CLOCK-2.

Figure CLOCK-2 Fcpu = Fosc/2



Thus, for example, if a 10 MHz crystal is connected to the primary oscillator, Fcpu will equal 5 MHz.

Phase locked loop use

The next alternative provided for in Figure CLOCK-1 is to use the primary oscillator together with a 4x phase locked loop. This results in a high system clock frequency while using a crystal in the 4 to 8 MHz range (an optimal range for a crystal). With its maximum crystal frequency of 8 MHz, the phase locked loop will produce

Fosc = 32 MHz and Fcpu = 16 MHz, the maximum CPU clock frequency for this PIC24F family part. The configuration lines for this choice are:

```
_CONFIG2(POSCMOD_HS & FNOSC_PRIPLL);
```

Fast RC oscillator (FRC)

An alternative way to obtain a maximum CPU clock frequency uses the phase locked loop together with the internal 8 MHz fast RC oscillator FRC) shown in Figure CLOCK-1. Specify this configuration choice with:

```
_CONFIG2(FNOSC_FRCPLL);             // 8 MHz FRC oscillator with PLL module
```

Fcpu = 16 MHz is obtained via the Initial function instruction of

```
CLKDIV = 0x0000;                    // Divide by 1; Fosc = 32 MHz; Fcpu = 16 MHz
```

Alternatively, Fcpu = 8 MHz results via

```
CLKDIV = 0x0100;                    // Divide by 2; Fosc = 16 MHz; Fcpu = 8 MHz
```

These are the only CLKDIV choices that can be used with the phase locked loop and still meet its input frequency requirement.

The ±1% accuracy of the internal 8 MHz fast RC oscillator (FRC) shown in Figure CLOCK-1 is sufficient for many applications and is the choice employed on the PIC24 Development Board. This choice frees up the two pins that would otherwise be needed for the primary oscillator crystal. The alternative functions of these two pins become

OSCO → CLKO (used to make Fcpu available for scope probing on the H2 header)
OSCI → RA2 (used to power the potentiometer and RPG pullup resistors)

The configuration line for this choice is:

```
_CONFIG2(FNOSC_FRCDIV);             // 8 MHz FRC oscillator plus divide-by-N via CLKDIV setting
```

The first instruction of the Initial function

```
CLKDIV = 0x0200;                    // Divide by 4; Fosc = 2 MHz; Fcpu = 1 MHz
```

can then be used to provide a CPU clock frequency of Fcpu = 1 MHz.

Secondary (crystal) oscillator

The chip includes the provision for a low-power 32,768 Hz watch crystal secondary oscillator, with the crystal connected to two pins, SOSCI and SOSCO, together with two 27 pF capacitors to ground. While we will not derive the CPU clock from this slow clock source, we will use it as the clock source for Timer1. The benefit derived thereby is that the CPU clock can be periodically put to sleep to save power. At the same time Timer1 can continue to be clocked by the secondary oscillator and used to awaken the CPU periodically. This secondary oscillator circuit in combination with Timer1 is specified to draw less than 1.0 μA of current, much less than the 160 μA typically drawn by the 1 MHz FRC oscillator/CLKDIV combination when it is running but not clocking the CPU (i.e., in “idle” mode) or internal peripheral modules.

Watchdog timer

If we use the watchdog timer to control the loop time of our user code, a 31 kHz internal low-power oscillator (LPRC) is automatically invoked. The accuracy of this oscillator is specified as ±15%. While it can be used as the system clock, it is so slow and inaccurate that we will avoid doing so. On the other hand, its current draw is specified to be less than 1.0 μA, so it is a fine choice for the role of continuing to run while the rest of the chip is asleep and drawing negligible current.

CPU vs peripheral clock

A final feature of the PIC24F clock module can be invoked by the divider shown at the right side of Figure CLOCK-1. This divider permits the on-board peripherals (e.g., timers, UART) to be clocked at a fixed rate while the CPU clock is periodically decreased in frequency in order to save power. An example of this use would be to maintain the clock to the UART at its specified baud rate so that it can continue to receive serial data even when the CPU's operation has been slowed down.