CHAPTER 2

Hardware

Microcontrollers power the modern world: from cars to microwaves to toys. These tiny microchips integrate every component of a modern computer—a central processing unit (CPU), random access memory (RAM), nonvolatile memory (flash), and peripherals—into a single chip. Although microcontrollers have significantly less processing power than their personal computer counterparts, they are also much smaller, cost less, and use less power. Additionally, their peripherals—devices that connect the CPU with the physical world—allow software to interact with circuits directly: flashing lights, moving motors, and reading sensors.

Companies including (but certainly not limited to) Microchip, Atmel, Freescale, Texas Instruments, and STMicroelectronics manufacture an overwhelming array of microcontrollers with vastly different specifications. Rather than attempt to discuss microcontrollers generally, we focus on the PIC32MX795F512H (which we usually abbreviate as PIC32). With a fast processor, ample memory, and numerous peripherals, the PIC32MX795F512H is excellent for learning about microcontrollers and completing embedded control projects. Much of what you learn about the PIC32MX795F512H also applies more generally to the PIC32MX family of microcontrollers, and the broader concepts translate to microcontrollers more generally. Appendix C describes the differences between the PIC32MX795F512H and other PIC32 models.

2.1 The PIC32

2.1.1 Pins, Peripherals, and Special Function Registers (SFRs)

The PIC32 requires a supply voltage between 2.3 and 3.6 V and features a maximum CPU clock frequency of 80 MHz, 512 KB of program memory (flash), and 128 KB of data memory (RAM). Its peripherals include a 10-bit analog-to-digital converter (ADC), many digital I/O pins, USB 2.0, Ethernet, two CAN modules, four I^2C and three SPI synchronous serial communication modules, six UARTs for asynchronous serial communication, five 16-bit counter/timers (configurable to give two 32-bit timers and one 16-bit timer), five pulse-width modulation outputs, and several pins that can generate interrupts based on external signals. Whew. Do not worry if you do not know what all of these peripherals do, much of this book is dedicated to explaining them.

Embedded Computing and Mechatronics with the PIC32 Microcontroller. http://dx.doi.org/10.1016/B978-0-12-420165-1.00002-0

Pins connect the peripherals to the outside world. To cram so much functionality into only 64 pins, many serve multiple functions. See the pinout diagram for the PIC32MX795F512H (Figure 2.1). For example, pin 12 can be an analog input, a comparator input, a change notification input (which can generate an interrupt when an input changes state), or a digital input or output.

Table 2.1 summarizes some of the major pin functions. Other pin functions can be found in the PIC32MX5xx/6xx/7xx Data Sheet.

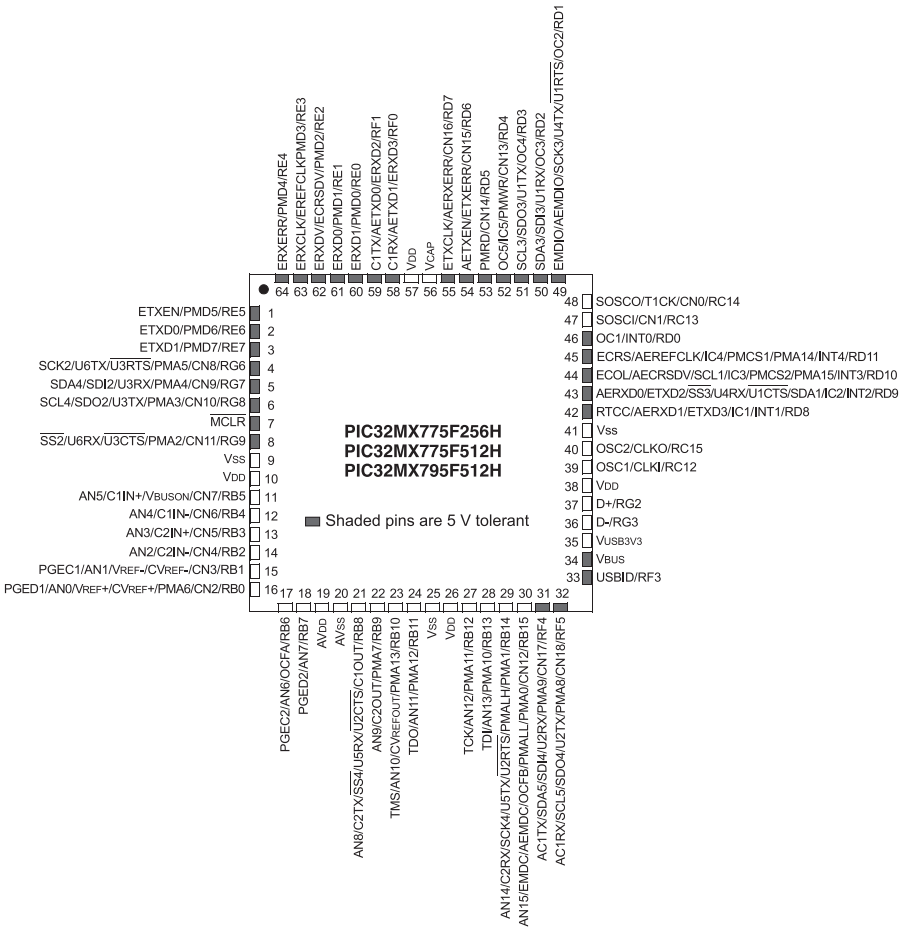

Figure 2.1
The pinout of the PIC32MX795F512H, the microcontroller used on the NU32 development board.

Table 2.1: Some of the pin functions on the PIC32

Pin Label	Function
ANx (x = 0 to 15)	Analog-to-digital (ADC) inputs
AVDD, AVSS	Positive supply and ground reference for ADC
CxIN-, CxIN+, CxOUT (x = 1, 2)	Comparator negative and positive input and output
CxRX, CxTx (x = 1, 2)	CAN receive and transmit pins
CLKI, CLKO	Clock input and output (for particular clock modes)
CNx (x = 0 to 18)	Change notification; voltage changes on these pins can generate interrupts
CVREF-, CVREF+, CVREFOUT	Comparator reference voltage low and high inputs, output
D+, D-	USB communication lines
ENVREG	Enable for on-chip voltage regulator that provides 1.8 V to internal core (on the NU32 board it is set to VDD to enable the regulator)
ICx (x = 1 to 5)	Input capture pins for measuring frequencies and pulse widths
INTx (x = 0 to 4)	Voltage changes on these pins can generate interrupts
$\overline{\text{MCLR}}$	Master clear reset pin, resets PIC when low
OCx (x = 1 to 5)	Output compare pins, usually used to generate pulse trains (pulse-width modulation) or individual pulses
OCFA, OCFB	Fault protection for output compare pins; if a fault occurs, they can be used to make OC outputs be high impedance (neither high nor low)
OSC1, OSC2	Crystal or resonator connections for different clock modes
PMAx (x = 0 to 15)	Parallel master port address
PMDx (x = 0 to 7)	Parallel master port data
PMENB, PMRD, PMWR	Enable and read/write strobes for parallel master port
Rxy (x = B to G, y = 0 to 15)	Digital I/O pins
RTCC	Real-time clock alarm output
SCLx, SDAx (x = 1, 3, 4, 5)	I^2C serial clock and data input/output for I^2C synchronous serial communication modules
SCKx, SDIx, SDOx (x = 2 to 4)	Serial clock, serial data in, out for SPI synchronous serial communication modules
$\overline{\text{SSx}}$ (x = 2 to 4)	Slave select (active low) for SPI communication
T1CK	Input pin for counter/timer 1 when counting external pulses
$\overline{\text{UxCTS}}$, $\overline{\text{UxRTS}}$, UxRX, UxTX (x = 1 to 6)	UART clear to send, request to send, receive input, and transmit output for UART modules
VDD	Positive voltage supply for peripheral digital logic and I/O pins (3.3 V on NU32)
VDDCAP	Capacitor filter for internal 1.8 V regulator when ENVREG enabled
VDDCORE	External 1.8 V supply when ENVREG disabled
VREF-, VREF+	Can be used as negative and positive limit for ADC
VSS	Ground for logic and I/O
VBUS	Monitors USB bus power
VUSB	Power for USB transceiver
USBID	USB on-the-go (OTG) detect

See Section 1 of the Data Sheet for more information.

Which function a particular pin actually serves is determined by *Special Function Registers* (SFRs). Each SFR is a 32-bit word that sits at a memory address. The values of the SFR bits, 0 (cleared) or 1 (set), control the functions of the pins as well as other PIC32 behavior.

For example, pin 51 in Figure 2.1 can be OC4 (output compare 4) or RD3 (digital I/O number 3 on port D). If we wanted to use pin 51 as a digital output we would set the SFRs that control this pin to disable the OC4 functionality and enable RD3 as a digital output. The Data Sheet explains the memory addresses and meanings of the SFRs. Be careful, because it includes information for many different PIC32 models. Looking at the Data Sheet section on Output Compare reveals that the 32-bit SFR named "OC4CON" determines whether OC4 is enabled. Specifically, for bits numbered 0-31, we see that bit 15 is responsible for enabling or disabling OC4. We refer to this bit as OC4CON⟨15⟩. If it is cleared (0), OC4 is disabled, and if it is set (1), OC4 is enabled. So we clear this bit to 0. (Bits can be "cleared to 0" or simply "cleared," or "set to 1" or simply "set.") Now, referring to the I/O Ports section of the Data Sheet, we see that the input/output direction of Port D is controlled by the SFR TRISD, and bits 0-11 correspond to RD0-RD11. Bit 3 of the SFR TRISD, i.e., TRISD⟨3⟩, should be cleared to 0 to make RD3 (pin 51) a digital output.

According to the Memory Organization section of the Data Sheet, OC4CON⟨15⟩ is cleared by default on reset, so it is not necessary for our program to clear OC4CON⟨15⟩. On the other hand, TRISD⟨3⟩ is set to 1 on reset, making pin 51 a digital input by default, so the program must clear TRISD⟨3⟩. For safety, all pins are inputs on reset to prevent the PIC32 from imposing an unwanted voltage on external circuitry.

In addition to setting the behavior of the pins, SFRs are the primary means of communication between the PIC32's CPU and its peripherals. You can think of a peripheral, such as a UART communication peripheral, as an independent circuit on the same chip as the CPU. Your program, running on the CPU, configures behavior of this circuit (such as the speed of UART communication) by writing bits to one or more SFRs which are read by the peripheral circuit. The CPU sends data to the peripheral (e.g., data to be sent by the UART) by writing to SFRs, and the CPU receives data from the peripheral (e.g., data received by the UART) by reading SFRs controlled by the peripheral.

We will see and use SFRs repeatedly as we learn about the PIC32.

2.1.2 PIC32 Architecture

Peripherals

Figure 2.2 depicts the PIC32's architecture. Of course there is a CPU, program memory (flash), and data memory (RAM). Perhaps most interesting to us, though, are the *peripherals*, which make microcontrollers useful for embedded control. We briefly discuss the available peripherals here; subsequent chapters cover them in detail. The peripherals are listed roughly in top to bottom, left to right order, as they appear in Figure 2.2.

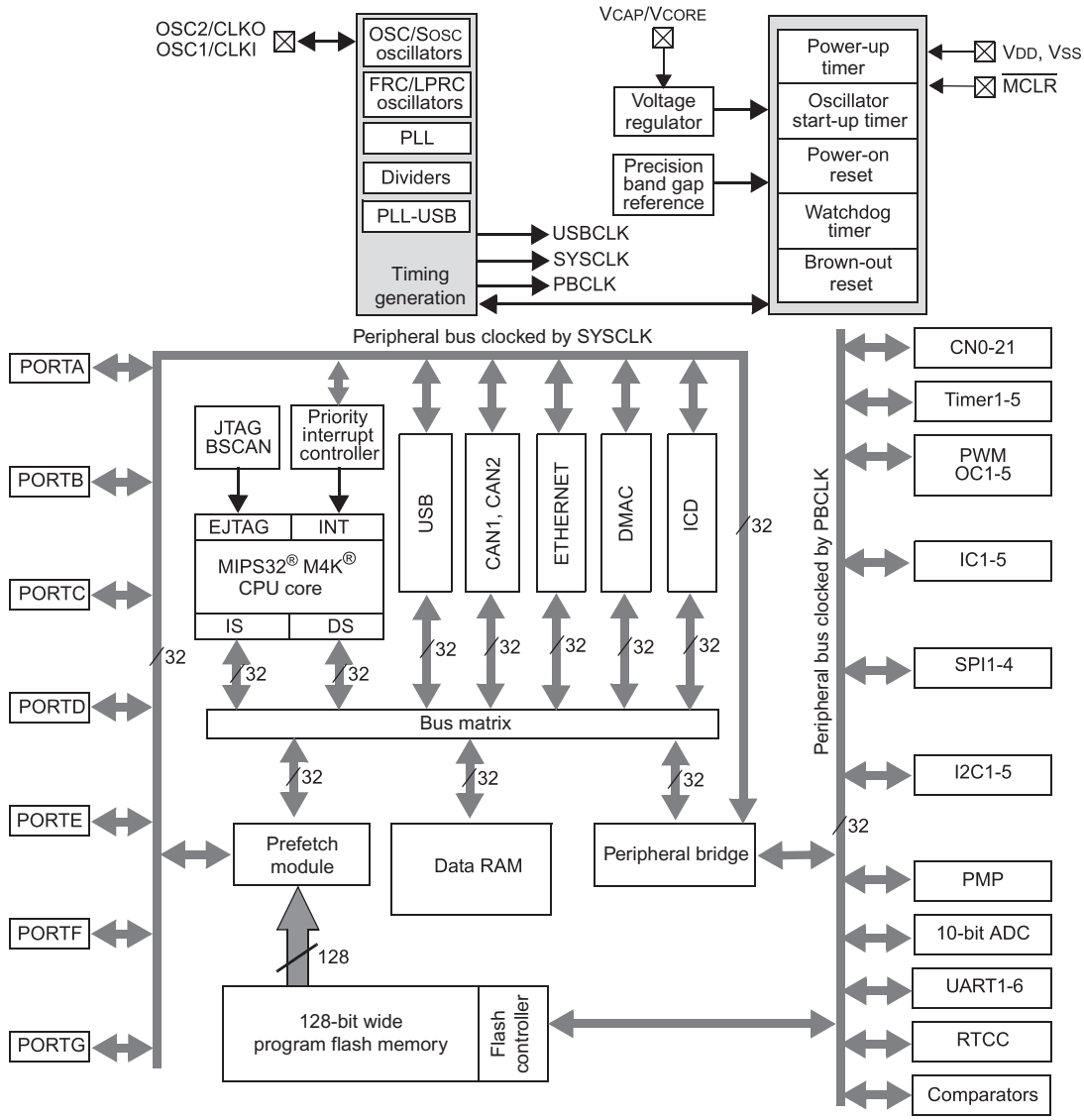

Figure 2.2
The PIC32MX5XX/6XX/7XX architecture. The PIC32MX795F512H is missing the digital I/O PORTA and has only 19 change notification inputs, 3 SPI modules, and 4 I^2C modules.

Digital input and output

Digital I/O ports (PORTB to PORTG on the PIC32MX795F512H) allow you to read or output a digital voltage. A digital I/O pin configured as an input can detect whether the input voltage is low or high. On the NU32, the PIC32 is powered by 3.3 V, so voltages close to 0 V are considered low and those close to 3.3 V are considered high. Some input pins can tolerate up

to 5.5 V, while voltages over 3.3 V on other pins could damage the PIC32 (see Figure 2.1 for the pins that can tolerate 5.5 V).

A digital I/O pin configured as an output can produce a voltage of 0 or 3.3 V. An output pin can also be configured as *open drain*. In this configuration, the pin is connected by an external pull-up resistor to a voltage of up to 5.5 V. This allows the pin's output transistor to either sink current (to pull the voltage down to 0 V) or turn off (allowing the voltage to be pulled up as high as 5.5 V), increasing the range of output voltages the pin can produce.

Universal Serial Bus

The Universal Serial Bus (USB) is an asynchronous communication protocol heavily used by computers and other devices. One master communicates with one or more slaves over a four-line bus: +5 V, ground, D+, and D− (differential data signals). The PIC32 has a single USB peripheral implementing USB 2.0 full-speed and low-speed options, and can communicate at theoretical data rates of up to several megabits per second.

Controller area network

Controller area network (CAN) is pervasive in industrial and automotive applications, where electrical noise can be problematic. CAN allows many devices to communicate over a single two-wire bus. Data rates of up to 1 megabit per second are possible. The CAN peripheral uses an external transceiver chip to convert between signals on the bus and signals that the PIC32 can process. The PIC32 contains two CAN modules.

Ethernet

The Ethernet module allows the PIC32 to connect to the Internet. It uses an external physical layer protocol transceiver (PHY) chip and direct memory access (DMA) to offload the heavy processing requirements of Ethernet communication from the CPU. The NU32 board does not include a PHY chip.

DMA controller

The direct memory access (DMA) controller (DMAC) transfers data without involving the CPU. For example, DMA can allow an external device to dump data through a UART directly into PIC32 RAM.

In-Circuit Debugger

The In-Circuit Debugger (ICD) is used by Microchip debugging tools to control the PIC32's operation during debugging.

Watchdog timer

If the watchdog timer (WDT) is used by your program, your program must periodically reset a counter. Otherwise, when the counter reaches a specified value, the PIC32 will reset. The WDT allows the PIC32 to recover from an unexpected state or infinite loop caused by software errors.

Change notification

A change notification (CN) pin can be used to generate an interrupt when the input voltage changes from low to high or vice-versa. The PIC32 has 19 change notification pins (CN0 to CN18).

Counter/timers

The PIC32 has five 16-bit counters/timers (Timer1 to Timer5). A counter counts the number of pulses of a signal. If the pulses occur at a regular frequency, the count can be used as a time; hence timers are just counters with inputs at a fixed frequency. Microchip uniformly refers to these devices as "timers", so we adopt that terminology from now on. Each timer can count from 0 up to $2^{16} - 1$, or any preset value less than $2^{16} - 1$ that we choose, before rolling over. Timers can count external events, such as pulses on the T1CK pin, or internal pulses on the peripheral bus clock. Two 16-bit timers can be configured to make a single 32-bit timer. Two different pairs of timers can be combined, yielding one 16-bit and two 32-bit timers.

Output compare

The five output compare (OC) pins (OC1 to OC5) are used to generate a single pulse of specified duration, or a continuous pulse train of specified duty cycle and frequency. They work with timers to generate pulses with precise timing. Output compare is commonly used to generate PWM (pulse-width modulated) signals that can control motors or be low-pass filtered to create a specified analog voltage output. (You cannot output an arbitrary analog voltage from the PIC32.)

Input capture

The five input capture (IC) pins (IC1 to IC5) store the current time, as measured by a timer, when an input changes. Thus, this peripheral allows precise measurements of input pulse widths and signal frequencies.

Serial Peripheral Interface

The PIC32 has three Serial Peripheral Interface (SPI) peripherals (SPI2 to SPI4). The SPI bus provides a method for synchronous serial communication between a master device (typically a

microcontroller) and one or more slave devices. The interface typically requires four communication pins: a clock (SCK), data in (SDI), data out (SDO), and slave select (SS). Communication can occur at up to tens of megabits per second.

Inter-integrated circuit

The PIC32 has four inter-integrated circuit (I^2C) modules (I2C1, I2C3, I2C4, I2C5). I^2C (pronounced "I squared C") is a synchronous serial communication standard (like SPI) that allows several devices to communicate over only two wires. Any device can be the master and control communication at any given time. The maximum data rate is less than for SPI, usually 100 or 400 kilobits per second.

Parallel master port

The parallel master port (PMP) module is used to read data from and write data to external parallel devices. Parallel communication allows multiple data bits to be transferred simultaneously, but each bit requires its own wire.

Analog input

The PIC32 has one analog-to-digital converter (ADC), but 16 different pins can be connected to it, allowing up to 16 analog voltage values (typically sensor inputs) to be monitored. The ADC can be programmed to continuously read data from a sequence of input pins, or to read a single value. Input voltages must be between 0 and 3.3 V. The ADC has 10 bits of resolution, allowing it to distinguish $2^{10} = 1024$ different voltage levels. Conversions are theoretically possible at a maximum rate of 1 million samples per second.

Universal asynchronous receiver/transmitter

The PIC32 has six universal asynchronous receiver transmitter (UART) modules (UART1 to UART6). These peripherals provide another method for serial communication between two devices. Unlike synchronous serial protocols such as SPI, the UART has no clock line; rather the devices communicating each have their own clocks that must operate at the same frequency. Each of the two devices participating in UART communication has, at minimum, a receive (RX) and transmit (TX) line. Often request to send (RTS) and clear to send (CTS) lines are used as well, allowing the devices to coordinate when to send data. Typical data rates are 9600 bits per second (9600 baud) up to hundreds of thousands of bits per second. The `talkingPIC.c` program uses a UART configured to operate at 230,400 baud to communicate with your computer.

Real-time clock and calendar

The real-time clock and calendar (RTCC) module maintains accurate time, in seconds, minutes, days, months, and years, over extended periods of time.

Comparators

The PIC32 has two comparators, each of which compares two analog input voltages and determines which is larger. A configurable internal voltage reference may be used in the comparison, or even output to a pin, resulting in a limited-resolution digital-to-analog converter.

Other components

Note that the peripherals are on two different buses: one is clocked by the system clock SYSCLK, and the other is clocked by the peripheral bus clock PBCLK. A third clock, USBCLK, is used for USB communication. The timing generation block that creates these clock signals and other elements of the architecture in Figure 2.2 are briefly described below.

CPU

The central processing unit runs everything. It fetches program instructions over its "instruction side" (IS) bus, reads data over its "data side" (DS) bus, executes the instructions, and writes the results over the DS bus. The CPU can be clocked by SYSCLK at up to 80 MHz, meaning it can execute one instruction every 12.5 ns. The CPU is capable of multiplying a 32-bit integer by a 16-bit integer in one cycle, or a 32-bit integer by a 32-bit integer in two cycles. There is no floating point unit (FPU), so floating point math is carried out by software algorithms, making floating point operations much slower than integer math.

The CPU is the MIPS32® M4K® microprocessor core, licensed from Imagination Technologies. The CPU operates at 1.8 V (provided by a voltage regulator internal to the PIC32, as it's used on the NU32 board). The interrupt controller, discussed below, can notify the CPU about external events.

Bus matrix

The CPU communicates with other units through the 32-bit bus matrix. Depending on the memory address specified by the CPU, the CPU can read data from, or write data to, program memory (flash), data memory (RAM), or SFRs. The memory map is discussed in Section 2.1.3.

Interrupt controller

The interrupt controller presents "interrupt requests" to the CPU. An interrupt request (IRQ) may be generated by a variety of sources, such as a changing input on a change notification pin or by the elapsing of a specified time on one of the timers. If the CPU accepts the request, it will suspend whatever it is doing and jump to an *interrupt service routine* (ISR), a function defined in the program. After completing the ISR, program control returns to where it was suspended. Interrupts are an extremely important concept in embedded control and are discussed thoroughly in Chapter 6.

Memory: Program flash and data RAM

The PIC32 has two types of memory: flash and RAM. Flash is generally more plentiful on PIC32's (e.g., 512 KB flash vs. 128 KB RAM on the PIC32MX795F512H), nonvolatile (meaning that its contents are preserved when powered off, unlike RAM), but slower to read and write than RAM. Your program is stored in flash memory and your temporary data is stored in RAM. When you power cycle the PIC32, your program is still there but your data in RAM is lost.[1]

Because flash is slow, with a max access speed of 30 MHz for the PIC32MX795F512H, reading a program instruction from flash may take three CPU cycles when operating at 80 MHz (see Electrical Characteristics in the Data Sheet). The prefetch cache module (described below) can minimize or eliminate the need for the CPU to wait for program instructions to load from flash.

Prefetch cache module

You might be familiar with the term *cache* from your web browser. Your browser's cache stores recent documents or pages you have accessed, so the next time you request them, your browser can provide a local copy immediately, instead of waiting for the download.

The *prefetch cache module* operates similarly—it stores recently executed program instructions, which are likely to be executed again soon (as in a program loop), and, in linear code with no branches, it can even run ahead of the current instruction and predictively *prefetch* future instructions into the cache. In both cases, the goal is to have the next instruction requested by the CPU already in the cache. When the CPU requests an instruction, the cache is first checked. If the instruction at that memory address is in the cache (a cache *hit*), the prefetch module provides the instruction to the CPU immediately. If there is a *miss*, the slower load from flash memory begins.

In some cases, the prefetch module can provide the CPU with one instruction per cycle, hiding the delays due to slow flash access. The module can cache all instructions in small program loops, so that flash memory does not have to be accessed while executing the loop. For linear code, the 128-bit wide data path between the prefetch module and flash memory allows the prefetch module to run ahead of execution despite the slow flash load times.

The prefetch cache module can also store constant data.

Clocks and timing generation

There are three clocks on the PIC32: SYSCLK, PBCLK, and USBCLK. USBCLK is a 48 MHz clock used for USB communication. SYSCLK clocks the CPU at a maximum

[1] It is also possible to store program instructions in RAM, and to store data in flash, but we ignore that for now.

frequency of 80 MHz, adjustable down to 0 Hz. Higher frequency means more calculations per second but higher power usage (approximately proportional to frequency). PBCLK is used by many peripherals, and its frequency is set to SYSCLK's frequency divided by 1, 2, 4, or 8. You might want to set PBCLK's frequency lower than SYSCLK's if you want to save power. If PBCLK's frequency is less than SYSCLK's, then programs with back-to-back peripheral operations will cause the CPU to wait a few cycles before issuing the second peripheral command to ensure that the first one has completed.

All clocks are derived either from an oscillator internal to the PIC32 or an external resonator or oscillator provided by the user. High-speed operation requires an external circuit, so the NU32 provides an external 8 MHz resonator as a clock source. The NU32 software sets the PIC32's configuration bits (see Section 2.1.4) to use a phase-locked loop (PLL) on the PIC32 to multiply this frequency by a factor of 10, generating a SYSCLK of 80 MHz. The PBCLK is set to the same frequency. The USBCLK is also derived from the 8 MHz resonator by multiplying the frequency by 6.

2.1.3 The Physical Memory Map

The CPU accesses peripherals, data, and program instructions in the same way: by writing a memory address to the bus. The PIC32's memory addresses are 32-bits long, and each address refers to a byte in the *memory map*. Thus, the PIC32's memory map consists of 4 GB (four gigabytes, or 2^{32} bytes). Of course most of these addresses are meaningless; there are far more addresses than needed.

The PIC32's memory map consists of four main components: RAM, flash, peripheral SFRs that we write to (to control the peripherals or send outputs) or read from (to get sensor input, for example), and *boot flash*. Of these, we have not yet seen "boot flash." This extra flash memory, 12 KB on the PIC32MX795F512H, contains program instructions that are executed immediately upon reset.[2] The boot flash instructions typically perform PIC32 initialization and then call the program installed in program flash. For the PIC32 on the NU32 board, the boot flash contains a "bootloader" program that communicates with your computer when you load a new program on the PIC32 (see Chapter 3).

The following table illustrates the PIC32's *physical* memory map. It consists of a block of "RAMsize" bytes of RAM (128 KB for the PIC32MX795F512H), "flashsize" bytes of flash (512 KB for the PIC32MX795F512H), 1 MB for the peripheral SFRs, and "bootsize" for the boot flash (12 KB for the PIC32MX795F512H):

[2] The last four 32-bit words of the boot flash memory region are Device Configuration Registers (see Section 2.1.4).

Physical Memory Start Address	Size (bytes)	Region
0x00000000	RAMsize (128 KB)	Data RAM
0x1D000000	flashsize (512 KB)	Program Flash
0x1F800000	1 MB	Peripheral SFRs
0x1FC00000	bootsize (12 KB)	Boot Flash

The memory regions are not contiguous. For example, the first address of program flash is 480 MB after the first address of data RAM. An attempt to access an address between the data RAM segment and the program flash segment would generate an error.

It is also possible to allocate a portion of RAM to hold program instructions.

In Chapter 3, when we discuss programming the PIC32, we will introduce the *virtual* memory map and its relationship to the physical memory map.

2.1.4 Configuration Bits

The last four 32-bit words of the boot flash are the Device Configuration Registers, DEVCFG0 to DEVCFG3, containing the *configuration bits*. The values in these configuration bits determine important properties of how the PIC32 will function. You can learn more about configuration bits in the Special Features section of the Data Sheet. For example, DEVCFG1 and DEVCFG2 contain configuration bits that determine the frequency multiplier converting the external resonator frequency to the SYSCLK frequency, as well as bits that determine the ratio between the SYSCLK and PBCLK frequencies. On the NU32 board (below), the PIC32's configuration bits were programmed along with the bootloader.

2.2 The NU32 Development Board

The NU32 development board is shown in Figure 1.1, and the pinout is given in Table 2.2. The NU32 board provides easy breadboard access to most of the PIC32MX795F512H's 64 pins. The NU32 acts like a big 60-pin DIP (dual in-line package) chip and plugs into a standard prototyping breadboard as shown in Figure 1.1. More details and the latest information on the NU32 can be found on the book's website.

Beyond simply breaking out the pins, the NU32 provides many features that make it easy to get started with the PIC32. For example, to power the PIC32, the NU32 provides a barrel jack that accepts a 1.35 mm inner diameter, 3.5 mm outer diameter center-positive power plug. The plug should provide 1 A at DC 6 V or more. The PIC32 requires a supply voltage VDD between 2.3 and 3.6 V, and the NU32 provides a 3.3 V voltage regulator providing a stable voltage source for the PIC32 and other electronics on board. Since it is often convenient to have a 5 V supply available, the NU32 also has a 5 V regulator. The power plug's raw input

Table 2.2: The NU32 pinout (in gray, with power jack at top) with PIC32MX795F512H pin numbers

Function	PIC32			PIC32	Function
GND		GND	GND		GND
3.3 V		3.3 V	3.3 V		3.3 V
5 V		5 V	5 V		5 V
VIN		VIN	VIN		VIN
C1RX/RF0	√ 58	**F0**	GND		GND
C1TX/RF1	√ 59	**F1**	**G9**	8 √	U6RX/$\overline{\text{U3CTS}}$/PMA2/CN11/RG9
PMD0/RE0	√ 60	E0	**G8**	6 √	SCL4/SDO2/U3TX/PMA3/CN10/RG8
PMD1/RE1	√ 61	E1	**G7**	5 √	SDA4/SDI2/U3RX/PMA4/CN9/RG7
PMD2/RE2	√ 62	E2	**G6**	4 √	SCK2/U6TX/$\overline{\text{U3RTS}}$/PMA5/CN8/RG6
PMD3/RE3	√ 63	E3	**$\overline{\text{MCLR}}$**	7 √	$\overline{\text{MCLR}}$
PMD4/RE4	√ 64	E4	**D7**	55 √	CN16/RD7
PMD5/RE5	√ 1	E5	D6	54 √	CN15/RD6
PMD6/RE6	√ 2	E6	D5	53 √	PMRD/CN14/RD5
PMD7/RE7	√ 3	E7	D4	52 √	OC5/IC5/PMWR/CN13/RD4
AN0/PMA6/CN2/RB0	16	B0	D3	51 √	SCL3/SDO3/U1TX/OC4/RD3
AN1/CN3/RB1	15	B1	D2	50 √	SDA3/SDI3/U1RX/OC3/RD2
AN2/C2IN-/CN4/RB2	14	B2	D1	49 √	SCK3/U4TX/$\overline{\text{U1RTS}}$/OC2/RD1
AN3/C2IN+/CN5/RB3	13	B3	D0	46 √	OC1/INT0/RD0
AN4/C1IN-/CN6/RB4	12	B4	C14	48	T1CK/CN0/RC14
AN5/C1IN+/CN7/RB5	11	B5	C13	47	CN1/RC13
AN6/OCFA/RB6	17	B6	D11	45 √	IC4/PMA14/INT4/RD11
AN7/RB7	18	B7	D10	44 √	SCL1/IC3/PMA15/INT3/RD10
AN8/C2TX/U5RX/$\overline{\text{U2CTS}}$/RB8	21	B8	D9	43 √	U4RX/$\overline{\text{U1CTS}}$/SDA1/IC2/INT2/RD9
AN9/PMA7/RB9	22	B9	D8	42 √	IC1/INT1/RD8
AN10/PMA13/RB10	23	B10	G2	37	D+/RG2
AN11/PMA12/RB11	24	B11	G3	36	D−/RG3
AN12/PMA11/RB12	27	B12	VBUS	34 √	VBUS
AN13/PMA10/RB13	28	B13	F3	33 √	USBID/RF3
AN14/C2RX/SCK4/U5TX/$\overline{\text{U2RTS}}$/PMA1/RB14	29	B14	F4	31 √	SDA5/SDI4/U2RX/PMA9/CN17/RF4
AN15/OCFB/PMA0/CN12/RB15	30	B15	F5	32 √	SCL5/SDO4/U2TX/PMA8/CN18/RF5

Pins marked with a √ are 5.5 V tolerant. Not all pin functions are listed; see Figure 2.1 or the PIC32 Data Sheet. Board pins in **bold** should only be used with care, as they are shared with other functions on the NU32. In particular, the NU32 pins G6, G7, G8, G9, F0, F1, D7, and MCLR should be considered outputs during normal usage. The value of $\overline{\text{MCLR}}$ is determined by the $\overline{\text{MCLR}}$ button on the NU32; the value of D7 is determined by the USER button; F0 and F1 are used by the PIC32 as digital outputs to control LED1 and LED2 on the NU32, respectively; and G6 through G9 are used by the PIC32's UART3 for communication with the host computer through the mini-B USB jack.

voltage Vin and ground, as well as the regulated 3.3 V and 5 V supplies, are made available to the user as illustrated in Figure 1.1. The power jack is directly connected to the Vin and GND pins so you could power the NU32 by putting Vin and GND on these pins directly and not connecting the power jack.

The 3.3 V regulator provides up to 800 mA and the 5 V regulator provides up to 1 A of current, provided the power supply can source that much current. In practice you should stay well under each of these limits. For example, you should not plan to draw more than 200-300 mA or so from the NU32. Even if you use a higher-current power supply, such as a battery, you should respect these limits, as the current has to flow through the relatively thin traces of the PCB. It is also not recommended to use high voltage supplies greater than 9 V or so, as the regulators will heat up.

Since motors tend to draw lots of current (even small motors may draw hundreds of milliamps up to several amps), do not try to power them from the NU32. Use a separate battery or power supply instead.

In addition to the voltage regulators, the NU32 provides an 8 MHz resonator as the source of the PIC32's 80 MHz clock signal. It also has a mini-B USB jack to connect your computer's USB port to a USB-to-UART FTDI chip that allows your PIC32 to use its UART to communicate with your computer.

A USB micro-B jack is provided to allow the PIC32 to speak USB to another external device, like a smartphone.

The NU32 board also has a power switch which connects or disconnect the input power supply to the voltage regulators, and two LEDs and two buttons (labeled USER and RESET) allowing very simple input and output. The two LEDs, LED1 and LED2, are connected at one end by a resistor to 3.3 V and the other end to digital outputs RF0 and RF1, respectively, so that they are off when those outputs are high and on when they are low. The USER and RESET buttons are attached to the digital input RD7 and $\overline{\text{MCLR}}$ pins, respectively, and both buttons are configured to give 0 V to these inputs when pressed and 3.3 V otherwise. See Figure 2.3.

Because pins RG6 through RG9, RF0, RF1, and RD7 on the PIC32 are used for UART communication with the host computer, LEDs, and the USER button, other potential functions of these pins are not available if you would like to use the communication, LEDs, and USER button. In particular:

- UART6 is unavailable (conflicts with pins RG6 and RG9). Since UART3 is used for communication with the host computer, this leaves UART1, UART2, UART4, and UART5 for your programs.
- SPI2 is unavailable (conflicts with pins RG6 and RG7). This leaves SPI3 and SPI4.
- I2C4 is unavailable (conflicts with pins RG7 and RG8). This leaves I2C1, I2C3, and I2C5.

- The default CAN1 pins C1RX and C1TX cannot be used (they conflict with pins RF0 and RF1), but the configuration bit FCANIO in DEVCFG3 has been cleared to zero on the NU32, thereby setting CAN1 to use the alternate pins AC1RX (RF4) and AC1TX (RF5). Therefore no CAN module is lost.
- Media-independent interface (MII) Ethernet is unavailable (conflicts with pins RD7, RF0, and RF1). The PIC32 can implement Ethernet communication using either the MII or the reduced media-independent interface (RMII), and RMII Ethernet communication, which uses many fewer pins than MII, is still available on the NU32.
- Several change notification and digital I/O pins are unavailable, but many more remain.

In all, very little functionality is unavailable due to connections on the NU32, and advanced users can find ways to bypass even these limitations.

Although the NU32 comes with a bootloader installed in its flash memory, you have the option to use a programmer to install a standalone program. The five plated through-holes on the USB board align with the pins of devices such as the PICkit 3 programmer (Figure 2.4).

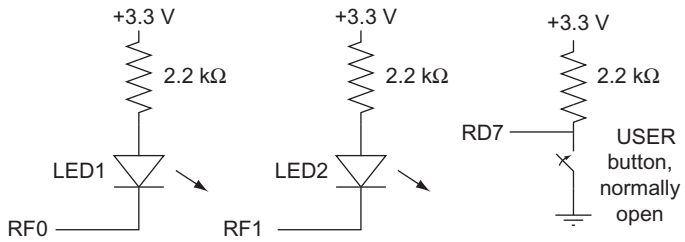

Figure 2.3
The NU32 connection of the PIC32 pins RF0, RF1, and RD7 to LED1, LED2, and the USER button, respectively.

Figure 2.4
Attaching the PICkit 3 programmer to the NU32 board.

2.3 Chapter Summary

- The PIC32 features a 32-bit data bus and a CPU capable of performing some 32-bit operations in a single clock cycle.
- In addition to nonvolatile flash program memory and RAM data memory, the PIC32 provides peripherals particularly useful for embedded control, including analog inputs, digital I/O, PWM outputs, counter/timers, inputs that generate interrupts or measure pulse widths or frequencies, and pins for a variety of communication protocols, including USB, Ethernet, CAN, I^2C, and SPI.
- The functions performed by the pins and peripherals are determined by Special Function Registers. SFRs are also used for communication back and forth between the CPU and peripherals.
- The PIC32 has three main clocks: the SYSCLK that clocks the CPU, the PBCLK that clocks peripherals, and the USBCLK that clocks USB communication.
- Physical memory addresses are specified by 32 bits. The physical memory map contains four regions: data RAM, program flash, SFRs, and boot flash. RAM can be accessed in one clock cycle, while flash access may be slower. The prefetch cache module can be used to minimize delays in accessing program instructions.
- Four 32-bit configuration words, DEVCFG0 to DEVCFG3, set important behavior of the PIC32. For example, these configuration bits determine how an external clock frequency is multiplied or divided to create the PIC32 clocks.
- The NU32 development board provides voltage regulators for power, includes a resonator for clocking, breaks out the PIC32 pins to a solderless breadboard, provides a couple of LEDs and buttons for simple input and output, and simplifies communication with the PIC32 via your computer's USB port.

2.4 Exercises

You will need to refer to the PIC32MX5XX/6XX/7XX Data Sheet and PIC32 Reference Manual to answer some questions.

1. Search for a listing of PIC32 products on Microchip's webpage, showing the specifications of all the PIC32 models.
   a. Find PIC32s that meet the following specs: at least 128 KB of flash, at least 32 KB of RAM, and at least 80 MHz max CPU speed. What is the cheapest PIC32 that meets these specs, and what is its volume price? How many ADC, UART, SPI, and I^2C channels does it have? How many timers?
   b. What is the cheapest PIC32 overall? How much flash and RAM does it have, and what is its maximum clock speed?
   c. Among all PIC32s with 512 KB flash and 128 KB RAM, which is the cheapest? How does it differ from the PIC32MX795F512H?

2. Based on C syntax for bitwise operators and bit-shifting, calculate the following and give your results in hexadecimal.
   a. `0x37 | 0xA8`
   b. `0x37 & 0xA8`
   c. `~0x37`
   d. `0x37>>3`
3. Describe the four functions that pin 12 of the PIC32MX795F512H can have. Is it 5 V tolerant?
4. Referring to the Data Sheet section on I/O Ports, what is the name of the SFR you have to modify if you want to change pins on PORTC from output to input?
5. The SFR CM1CON controls comparator behavior. Referring to the Memory Organization section of the Data Sheet, what is the reset value of CM1CON in hexadecimal?
6. In one sentence each, without going into detail, explain the basic function of the following items shown in the PIC32 architecture block diagram Figure 2.2: SYSCLK, PBCLK, PORTA to PORTG (and indicate which of these can be used for analog input on the NU32's PIC32), Timer1 to Timer5, 10-bit ADC, PWM OC1-5, Data RAM, Program Flash Memory, and Prefetch Cache Module.
7. List the peripherals that are *not* clocked by PBCLK.
8. If the ADC is measuring values between 0 and 3.3 V, what is the largest voltage difference that it may not be able to detect? (It's a 10-bit ADC.)
9. Refer to the Reference Manual chapter on the Prefetch Cache. What is the maximum size of a program loop, in bytes, that can be completely stored in the cache?
10. Explain why the path between flash memory and the prefetch cache module is 128 bits wide instead of 32, 64, or 256 bits.
11. Explain how a digital output could be configured to swing between 0 and 4 V, even though the PIC32 is powered by 3.3 V.
12. PIC32's have increased their flash and RAM over the years. What is the maximum amount of flash memory a PIC32 can have before the current choice of base addresses in the physical memory map (for RAM, flash, peripherals, and boot flash) would have to be changed? What is the maximum amount of RAM? Give your answers in bytes in hexadecimal.
13. Examine the Special Features section of the Data Sheet.
    a. If you want your PBCLK frequency to be half the frequency of SYSCLK, which bits of which Device Configuration Register do you have to modify? What values do you give those bits?
    b. Which bit(s) of which SFR set the watchdog timer to be enabled? Which bit(s) set the postscale that determines the time interval during which the watchdog must be reset to prevent it from restarting the PIC32? What values would you give these bits to enable the watchdog and to set the time interval to be the maximum?

c. The SYSCLK for a PIC32 can be generated several ways, as discussed in the Oscillator chapter in the Reference Manual and the Oscillator Configuration section in the Data Sheet. The PIC32 on the NU32 uses the (external) primary oscillator in HS mode with the phase-locked loop (PLL) module. Which bits of which device configuration register enable the primary oscillator and turn on the PLL module?

14. Your NU32 board provides four power rails: GND, regulated 3.3 V, regulated 5 V, and the unregulated input voltage (e.g., 6 V). You plan to put a load from the 5 V output to ground. If the load is modeled as a resistor, what is the smallest resistance that would be safe? An approximate answer is fine. In a sentence, explain how you arrived at the answer.
15. The NU32 could be powered by different voltages. Give a reasonable range of voltages that could be used, minimum to maximum, and explain the reason for the limits.
16. Two buttons and two LEDs are interfaced to the PIC32 on the NU32. Which pins are they connected to? Give the actual pin numbers, 1-64, as well as the name of the pin function as it is used on the NU32. For example, pin 37 on the PIC32MX795F512H could have the function D+ (USB data line) or RG2 (Port G digital input/output), but only one of these functions could be active at a given time.

Further Reading

PIC32 family reference manual. Section 03: Memory organization. (2010). Microchip Technology Inc.
PIC32 family reference manual. Section 02: CPU for devices with the M4K core. (2012). Microchip Technology Inc.
PIC32 family reference manual. Section 32: Configuration. (2013). Microchip Technology Inc.
PIC32MX5XX/6XX/7XX family data sheet. (2013). Microchip Technology Inc.