USC Viterbi School of Engineering

Ming Hsieh Department of Electrical Engineering

EE 459Lx - Embedded Systems Design Laboratory

The Freescale MC908JL16 Microcontroller

by Allan G. Weber

1 Introduction

The Freescale MC908JL16 (also called the MC68HC908JL16) is one member of the HC08 microcontroller family. Each member of the family has different amounts of RAM, ROM, I/O ports, etc. Depending on the number of external pins required they may come in packages with more than a hundred pins, or with as few as eight. The MC908JL16 was selected for the EE 459 class for a variety of reasons:

- Availability of both the chips and a development system from a vendor.
- Available in 28-pin DIP (dual-inline package) that fits into available IC sockets.
- Enough TTL compatible I/O pins (22) to handle most EE 459L project tasks.
- FLASH memory for easy and fast reprogramming.

The following notes are meant as a short guide to getting started using the JL16. Additional documents on the EE 459 web site describe using the C and assembly language software development systems. For more complete information on any of the topics below, see the full Freescale datasheet or programming manual.

2 Hardware

The MC908JL16 contains the following components:

- 16kb of FLASH memory for program storage.
- 512 bytes of RAM memory.
- Two 16-bit timers. These can count internal clock cycles or external events and generate an interrupt when reaching a specified count value.
- 12 channels of 10-bit analog-to-digital converter (ADC).
- Serial communications port. This can be used to communicate to the COM port of a computer.
- I^2C interface port for communication with other I^2C compatible ICs
- 22 lines of general purpose I/O. Some of these have special capabilities for driving things like LEDs.

Not all of these functions are available at the same time. Most of the pins on the chip are connected to multiple functional units and it is up to the designer to decide what a particular pin does. For example, a pin might be use as a general purpose I/O line, or it might be ADC input, but it can't be both simultaneously.

2.1 Minimum Connections

In order to make the microcontroller operate the following connections must be made.

2.1.1 Power and Ground

The power supply voltage (5 volts) must be connected to the VDD input on pin 7. The ground connection (VSS) is on pin 3.

2.1.2 Clock

Some sort of clock signal must be provided in order for the microcontroller to operate. On the MC908JL16 the clock can come from one of three different sources.

A TTL-compatible clock signal can be generated externally by other logic and connected to the OSC1 input (pin 4.) This probably the easiest way to generate the clock for the EE 459 projects. The lab has a supply of DIP oscillators in some of the more common frequencies. These output a TTL level square wave that can be fed directly into the microcontroller and to other chips. If this method of providing a clock is used, the OSC2 output (pin 5) **must** be left open and functions that are available from pin 5 in other modes, such as Port A bit 6 (PTA6) are **not** available in this mode.

Alternatively, the processor can generate a clock if a crystal is connected to the OSC1 and OSC2 inputs. This method uses a plain crystal, not the DIP crystal oscillators as described above.

The third method uses an RC circuit connected to OSC1 input (pin 4.) The time constant of the RC circuit determines the frequency of the clock. This is the least accurate way to generate a clock and in most cases it would be difficult to have the clock frequency within 10% of the desired frequency. Do not use this method if your project requires a clock running close to a specified frequency. The advantage of the RC clock mode is that other functions are now available on pin 5. For example it can now be used as Port A bit 6 (PTA6) thus giving the microcontroller 23 I/O pins. In order to use the RC mode, the OSCSEL bit in the Mask Option Register (MOR) must be programmed to a zero when programming the chip. See the datasheet for more information on using the RC clocking mode.

However the clock signal is provided, it is divided by four internally before reaching the processor. This must be taken into account when calculating the execution time of instructions for delay loops. For example, a 12MHz clock signal into OSC1 will result in an internal clock rate of 3MHz, and a 333ns period for each instruction clock cycle. The internal clock is referred to as the "internal bus clock" in the documentation.

In applications where the SCI (Serial Communications Interface) capabilities of the MC908JL16 are being used, the choice of clock frequency determines the baud rates that can be used for transmitting and receiving serial data. In most applications, the requirement to support a particular baud rate will determine what clock rates can be used. See Sec. 2.4 for more information.

2.1.3 Reset and Interrupt

The reset input ($\overline{\text{RST}}$, pin 28) and the interrupt input ($\overline{\text{IRQ}}$, pin 1) must be in the high state for the processor to operate normally. Both of these pins have internal pull-ups and can be left unconnected if they are not being used. The pins do not have to be externally pulled-up to VDD in order for the processor to operate normally.

2.2 I/O Ports

The MC908JL16 has 22 pins that can be configured for general purpose I/O. Many of these can also be used for other purposes such as analog-to-digital conversion, timers, etc. Each port has a corresponding Data Direction Register that is used to determine whether the pins for that port are serving as inputs or outputs. Initially, or upon a reset signal, the bits in the DDRs are all zero which makes the corresponding I/O port bits inputs. To use a I/O port bit as an output, the corresponding bit in the DDR must be set to a one. Each of the three ports are slightly different and it is important to carefully read the datasheet to determine how to use each port.

2.2.1 Port A (PTA)

Port A on the MC908JL16 has six pins (PTA0 through PTA5). A seventh bit, PTA6, is described in the manual but is only available when the RC clock mode is used. It is **NOT** available when using a DIP

oscillator or TTL signal to provide the clock. Each bit in PTA has a software configurable pull-up resistor that can be used if the bit is an input. These are controlled by bits in the PTAPUE register. Each of the bits PTA0-PTA5 are capable of sinking sufficient current (max. 25mA) to directly drive LEDs.

2.2.2 Port B (PTB)

Port B on the JL16 has eight pins (PTB0 through PTB7). These pins are shared with the analog-to-digital converter so if the ADC function is used one or more pins will not be available for general purpose I/O.

2.2.3 Port D (PTD)

Port D on the JL16 has eight pins (PTD0 through PTD7). These pins are shared with the serial communications interface, the ADC module and the timer module and some pins can not be used as I/O if one or more of these functions are used. PTD6 and PTD7 have programmable pull-ups similar to those for PTA. PTD2, PTD3, PTD6 and PTD7 are capable of driving LEDs.

2.3 Timers

The two internal timers (TIM1 and TIM2) can be used to count events and generate an interrupt when a specified number of events has occurred. A common use of a timer is to implement a delay function by counting the number of internal clock cycles that occur. See the sample programs for examples of using the timers for this function.

To implements a delay first determine the rate of the internal clock which is the external clock frequency divided by four. From this calculate what count value the counter will need to count to. This is referred to as the counter modulo value in the manual. If the modulo value exceeds the range of the timer's 16-bit register (greater then 65,535), use the prescaler to divide the internal clock by 2, 4, 8, etc. before it reaches the timer. The modulo value is loaded into the two 8-bit modulo registers, the interrupt is enabled in the status and control register , and then the counter is started by clearing the TSTOP bit in the control register.

When the counter reaches the modulo value, it generates an interrupt, resets the count value to zero and continues to count. The interrupt uses the vectors "TIM1 Overflow" ($FFF2:FFF3) or "TIM2 Overflow" ($FFEC:FFED) for storing the address of the interrupt service routine. The user program should service the interrupt and take whatever action is necessary. The interrupt service routine must read the counter status register and clear the TOF (Timer Overflow) bit in order to receive the next interrupt. Keep in mind that the counter does not stop and wait for the interrupt to be serviced. It continues to count regardless of when or if the user program services the interrupt.

The timers must be in the stop state before any of the timer registers other than the status and control register can be changed. For example if you wish to change the counter's modulo value, the counter must first be stopped before altering the contents of the modulo registers.

2.4 Serial Communications Interface

The MC908JL16 contains a Serial Communications Interface (SCI) that can be used to interface to something like a COM port on a PC or to another device. The data is transmitted from the MC908JL16 on the TxD pin, and data is received on the RxD pin. If the external device is using RS-232 voltage levels for signaling, an external RS-232 transceiver chip like a Maxim MAX232 is needed to translate between the voltage levels the MC908JL16 uses and RS-232 positive and negative voltages.

The SCI transmitter (TxD) and receiver (RxD) use the same pins on the chip as I/O ports PTD6 and PTD7. This means that applications that use the SCI can not also use these two bits in Port D. If the ENSCI (Enable SCI) bit is set in SCC1 (SCI Control Register 1), this enables both PTD6/TxD and PTD7/RxD for SCI use. It is not necessary to set any bits in the DDRD register in order to use the SCI.

The baud rates for sending and receiving serial data are derived from the main clock that drives the processor and for any clock rate only a limited number of baud rates (26) are available. When using the SCI, it is necessary to select an oscillator with a frequency that can provide the desired baud rate. The baud rate is determine by three things: the oscillator clock rate, the baud rate prescaler value, and the baud rate divisor value. The two bits, SCP1 and SCP0, allow a selection of one of four prescaler values to divide the

clock rate by: 1, 3, 4 and 13. The three bits, SCR2, SCR1 and SCR0, allow a selection of one of eight divisor values to further divide the clock: 1, 2, 4, 8, 16, 32, 64 and 128. The baud rate is determined by the formula

$$\text{baud rate} = \frac{\text{oscillator frequency}}{256 \times PD \times BD}$$

where PD is the prescaler value and BD is the baud rate divisor value.

The web page "http://ee-classes.usc.edu/ee459/library/jl16baud" can be used to calculate all of the possible baud rates that can be obtained for a given oscillator frequency, and corresponding settings for the bits SCP1, SCP0, SCR2, SCR1 and SCR0.

For more detailed information on using the serial interface, see the document "Using the Serial Communications Interface" on the class web site.

2.5 I^2C Interface

The MC908JL16 has the ability to communicate to other IC's using the IIC or I^2C serial interface. This is a two line bi-directional interface designed for medium speed communications between ICs on a board. One line is the clock, the other is for data. Numerous ICs are available on the market that have I^2C interface such as EEPROMs, real-time-clocks, RAM, etc.

On the JL16, the I^2C interface is available on either pins 13 and 14 or on pins 8 and 9. The default configuration is for it to be on pins 13 and 14. Setting the IICSEL bit in the CONFIG2 register to a one moves the I^2C interface to pins 8 and 9. These pins are all shared by multiple functions on the JL16. Pins 13 and 14 are part of I/O port D and are also used for serial communications. Pin 8 and 9 are part of port A. If the I^2C interface is enabled then any other function on those pins is not available.

If none of the two pairs of pins mentioned above are available for use, it is still possible to implement an I^2C interface by using software to perform all the I^2C transactions. Several I^2C libraries are available from Internet sites, all of which will have to be modified to work with the JL16.

For more information on using I^2C, see the document "Using the IIC Interface" on the class web site. This document also contains information on using the oscilloscopes in the lab to debug I^2C transactions.

2.6 A/D Conversion

The JL16 has an internal 10-bit analog-to-digital converter for converting analog voltages to binary value. The ADC inputs can be used for a variety of tasks such as sampling a voltage to determine the position of an input control. The ADC generates a value from 0 to 1023 in 10-bit mode, or 0 to 255 in 8-bit mode, for input levels between ground (VSS, pin 3) and the supply voltage (VDD, pin 7).

The ADC can accept input from any one of 12 pins depending on the bits in control register. It can only do a conversion of one input at a time. The following example shows how to make the ADC sample a voltage between ground and VDD on input ADC11 (pin 16). The conversion process requires an clock that can be generated from either an internal asynchronous clock (not synchronous with the microcontrollers clock), or by using the microcontrollers clock. If the asynchronous clock is used it will be in the correct frequency range for conversions. If the microcontrollers clock is used it must be divided down to a frequency in the correct range (1 - 2 MHz for high speed mode, 0.5 - 1 MHz for low power mode).

The following lines set the ADC for low power mode, set the clock divisor to one, set the conversion for longer samples times, set the clock source to be the internal asynchronous clock, and set the ADC to produce 8-bit results.

```
ADCLK_ADLPC = 1;            // Low power mode
ADCLK_ADIV = 0;             // Clock divisor ratio = 1
ADCLK_ADLSMP = 1;           // Long sample time
ADCLK_ACLKEN = 1;           // Asynchronous clock
ADCLK_MODE = 0;             // 8-bit conversions
```

Once the ADC is configured, the following loop starts a conversion, waits for it to finish and then loads the eight-bit value into the variable `x`.

```
while (1) {
    ADCSC_ADCH = 11;          // Select channel 11 (pin 16)
                              // also starts conversion
    while (ADCSC_COCO == 0); // wait for conversion complete

    x = ADRL;                 // Get converted value
}
```

See the full JL16 manual for more information on using the ADC.

2.7 FLASH Memory

The 16kb of FLASH memory is normally used for storing the program and data needed to run the processor. However it can also be used as non-volatile read-write memory by the program. For more information on reading and writing to FLASH memory, see Freescale Application Note AN2874 on the class web site.

3 Software Development

There are several ways to create the code that gets programmed into the FLASH memory of the processor. The MacPro systems in the EE 459 lab have Freescale's Codewarrior development software for programming in both C and assembler, and hardware for downloading the executable code to the microcontroller. The Codewarrior development software runs in the Windows virtual environment on these systems. It includes the compiler, assembler and linker, and software for downloading the program to the MC908JL16. The software is free and can be installed on any Windows system. Software for C and assembly language programming for the JL16 is discussed in separate documents available on the EE459 web site.