Patrick Austin  
CPE 301, Section 1104, 9/7/2016  
Homework 1

Part I

1.5 a. {0111 0110 1101 0001 1001 1011 1111 0000}2

= 24+25+26+27+28+29+211+212+215+216+220+222+223+225+226+228+229+230

= {1,993,448,432}10

b. {0111 0110 1101 0001 1001 1011 1111 0000}2

= {76D19BF0}16

1.6 a. {24687531}16

= {0010 0100 0110 1000 0111 0101 0011 0001}2

= 20+24+25+28+210+212+213+214+219+221+222+226+229

= {610,825,521}10

b. {24687531}16

= {0010 0100 0110 1000 0111 0101 0011 0001}2

1.7 a. {24,687,531}10

= {0001 0111 1000 1011 0011 1010 1011}2

= {178B3AB}16

b. {24,687,531}10

= {0001 0111 1000 1011 0011 1010 1011}2

1.8 a. {6F} 16 – one byte is required.

b. {6F}16 – one byte is required.

c. {0123}16 – two bytes are required.

d. {313233}16 – three bytes are required.

1.9 a. Cannot be represented.

b. {85}16 – one byte is required.

c. Cannot be represented.

d. {2D313233}16 – four bytes are required.

1.10 a. {DDD5}16 – two bytes are required.

b. {00DDD5}16 – three bytes are required.

c. {056789}16 – three bytes are required.

d. {3536373839}16 – five bytes are required.

e. {475DD500}16 – four bytes are required.

Part II

Investigating Input/Output Devices for Single Board Computers

In this essay I will discuss what I learned about two different types of input/output devices featured on single board computers, with a focus on those offered by Arduino. I will discuss the Arduino board offerings for UARTs and timers, including a discussion of what these features are, how they work, and some suggestions about what can be accomplished by using them.

I began my investigation on Arduino’s website, which has a useful page comparing the specifications of the range of Single Board Computers (hereafter SBCs) on offer, located here: <https://www.arduino.cc/en/Products/Compare>, ARD1. The only issue with this comparison is that it doesn’t show board prices. Here is a third party site that lists fewer specs but includes the MSRP of the boards: <https://learn.sparkfun.com/tutorials/arduino-comparison-guide>, SPAR1. (That said, Arduino board specs are open source, and a wide variety of third party manufacturers sell “knock-off” boards at lower cost.) Specs that will be relevant to the discussion of UARTs and timers are the number of UARTs on the board, and the speed of the board’s CPU respectively. Of Arduino’s non-retired boards, some offer no UARTs on accessible pins, some offer one or two, and the Mega 2560 offers four of them. Likewise, clock speed varies depending on the board between 8 to 48 MHz.

UART stands for universal asynchronous receiver/transmitter, and a UART is a unit of circuitry designed to send and receive packets of serial information.

First it is necessary to talk about what serial transmission means (as opposed to parallel transmission) and why it is used. Serial information is sent along a single line in sequence according to some communication protocol, unlike parallel information where each bit of data is sent along its own separate line. Serial information can be sent according to a clock, which is called synchronous transmission, or sent without a clock according to some protocol of transmission both devices have been designed to communicate using, which is called asynchronous transmission. Parallel transmission requires many more lines to transmit information than serial. Synchronous serial transmission is fast, but requires an additional line to transmit the clock value. An asynchronous serial transmission has the advantage of simplicity and cost of using a single line without a clock, which is important on SBCs which have a limited number of pins available to the user. (This information was adapted from this tutorial: <https://learn.sparkfun.com/tutorials/serial-communication/wiring-and-hardware>, SPAR2.)

As the name would imply, the UART uses asynchronous transmission. When receiving information, the UART transforms a packet of serial information sent to the device to a parallel bus of data lines along a “receive” line. When sending information, the UART converts the parallel data into a packet and sends it serially along the “transmit” data line, one bit at a time (plus some extra start/stop/verification data required by the protocol in question). In this sense the UART is built around the shift registers we learned about in CPE 201, sending or taking in one bit of data at a time in sequence from a word composed of multiple bits. In this fashion the UART can communicate back and forth with another device on just two lines, using just two of the SBC’s pins. With its four UARTs, the Mega we will use in this course can communicate with multiple such devices using only a few pins. The boards with no accessible UARTs still contain a UART internally for some manner of communication of programming information, but no such pins are directly available on the board for the user. (Information adapted from this tutorial: <https://learn.sparkfun.com/tutorials/serial-communication/uarts>, SPAR3.)

A timer is another familiar piece of hardware we studied in CPE 201 that will be relevant to using an SBC such as an Arduino to solve problems. A timer uses the cycling of an internal clock to increment a value in memory in a fashion that corresponds to the passage of time. This timer can be read and configured by the one programming on the board to create some desired outcome.

It should be noted that these timers will use the clock of the board as a baseline, which varies in speed from model to model. So programming a timer involves an assumption about the clock speed of the board and care may need to be taken when programming different models of board. The Mega has a 16 MHz clock, but some Arduino offerings have 8, 32, or 48 MHz clocks, so programmer beware.

Some Arduino boards have only three timers, though the Mega has six. These timers differ in terms of the amount of memory available for incrementing as time passes (8-bit versus 16-bit) and in terms of which Arduino library functions utilize which timer’s values. Some of the pins also have pre-mapped relationships to certain timers. (More detail about this is available at Arduino’s tutorial on pulse-width modulation here: <https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM>, ARD2.)

The timers on the Arduino can be used to generate interrupts, which halt the currently running code, run some special routine, and then resume where the program left off. For example, it is possible to generate an interrupt whenever the memory specified for one of the timers overflows. One could cause an instruction to happen at each timer overflow, ie at some regular interval, manipulating the rate at which the timer is incremented to reach the desired timeframe.

(The foregoing discussion on timers was distilled from useful resources on Arduino timers here: <https://oscarliang.com/arduino-timer-and-interrupt-tutorial/>, OSC1, and here: <https://arduino-info.wikispaces.com/Timers-Arduino>, ARDW1, despite unfortunate Comic Sans font.)

All told I think this investigation was a useful introduction to SBCs. I knew little about Arduinos going in, and feel I walked away more familiar with the differences between Arduino models as well as the properties of UARTs and timers, which are vital tools for programming an SBC and transferring data to and from other devices. As the prompt notes, these are hardware components present in almost any modern SBC, so this knowledge goes beyond merely Arduino products and clones thereof as well. Web search quickly reveals a much larger world of SBC manufacturers and a wide variety of products catering to industrial applications, among others. (See here for just one such example of a more industrially oriented SBC vendor: <http://www.futureelectronics.com/en/display-solutions/single-board-computer.aspx>, FE1.) This investigation was an interesting first step into a wide world of SBCs.

List of References

ARD1. Arduino board spec comparison, <https://www.arduino.cc/en/Products/Compare>, (copied

9/7/2016)

ARD2. Secrets of Arduino PWM, <https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM> ,

(copied 9/7/2016)

ARDW1. Arduino timers and interrupts, <https://arduino-info.wikispaces.com/Timers-Arduino> ,

(copied 9/7/2016)

SPAR1. Sparkfun Arduino board comparison,

<https://learn.sparkfun.com/tutorials/arduino-comparison-guide> , (copied 9/7/2016)

SPAR2. Sparkfun serial communication wiring tutorial,

<https://learn.sparkfun.com/tutorials/serial-communication/wiring-and-hardware> , (copied

9/7/2016)

SPAR3. Sparkfun UART tutorial, <https://learn.sparkfun.com/tutorials/serial-communication/uarts>

, (copied 9/7/2016)

OSC1. Oscarliang.com Arduino timer and interrupt tutorial,

<https://oscarliang.com/arduino-timer-and-interrupt-tutorial/> , (copied 9/7/2016)

FE1. Future Electronics: What is a single board computer?,

<http://www.futureelectronics.com/en/display-solutions/single-board-computer.aspx> ,

(copied 9/7/2016)