SJSU EE138
Introduction to Embedded Control System Design
Lab 1: General Purpose Input / Output

Readings:

- Atmel SAMD20 Data Sheet: Section 21: PORT

Lab Description:

Students will develop a calculator using the keypad and 7-segment display as the input/output devices. As a minimum, the calculator should be able to **add** and **subtract** two integer numbers (**with sign**). The digit key "#" will represent the 'add' (+), "*" will represent subtraction (-), "A" as the equal (=) and, and "B" as backspace. Do not worry about consecutive add and subtract operation as well as over/underflow (more than 9999 or less than -9999) condition. Extra credit will be given to project with additional functions or features such as implementing leading zero blanking, multiple, divide, decimal point or floating point. Figure 1.1 shows the schematic of the keypad and 7-segment display circuit and their connections to the SAMD20 board.

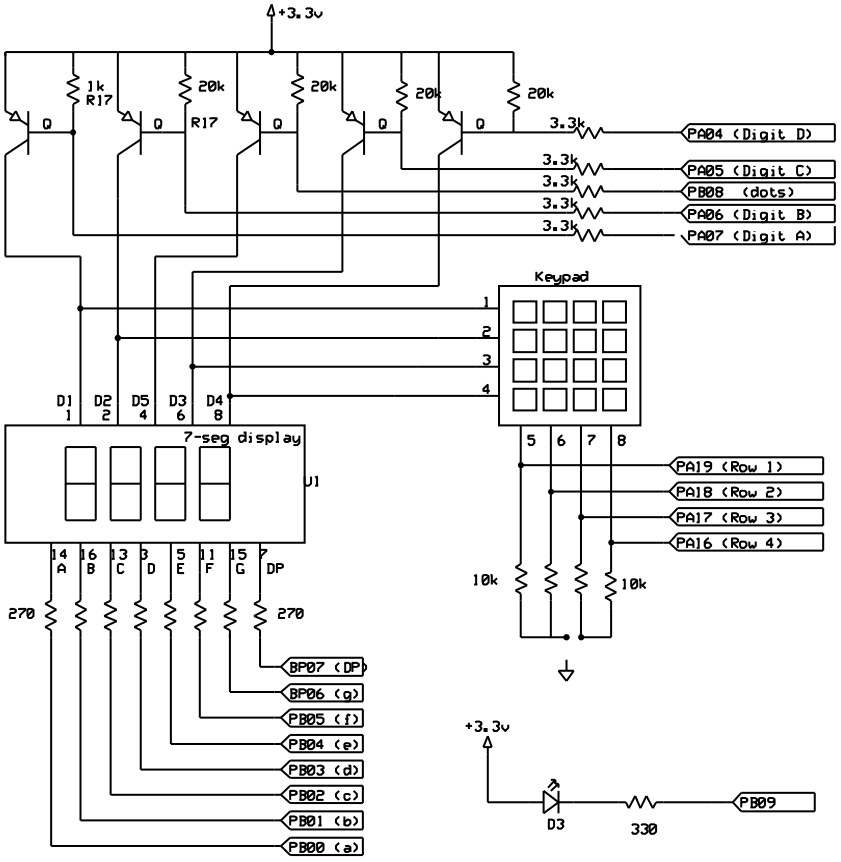

Figure 1.1

Theory of Operation and interfacing circuit:

1. Circuit (PNP transistors, LED current consideration, keypad matrix, pull up resistors)

- There are 8 LEDs in a single 7-segments display (7 LEDs for the digit and one for the decimal point). All 8 LEDs' anodes are connected together. This common anode is connected to the positive supply voltage +3.3v through a PNP transistor. The PNP transistor acts as a switch (see Figure 1.2). To turn on any or all of the LEDs, the PNP transistor must be 'turned on' by applying a base current by pulling the base voltage down by an output pin (e.g., PA07) through a current limiting resistor (R2). R2 should be selected so that the base current is high enough to drive the transistor to saturation in order to 'pass' the +3.3v (minus V_{CE_sat}) to the common anode pin. Now, to turn on a LED segment, the corresponding cathode pin should be pulled down by another output pin (e.g., PB00) through a current limiting resistor (R3) to complete the current path through the LED to ground. This output pin should be configured to high drive strength (i.e., DRVSTR=1). R3 should be designed so the current is high enough to power the LED to its full brightness but not exceeds the maximum current the output pin can handle.

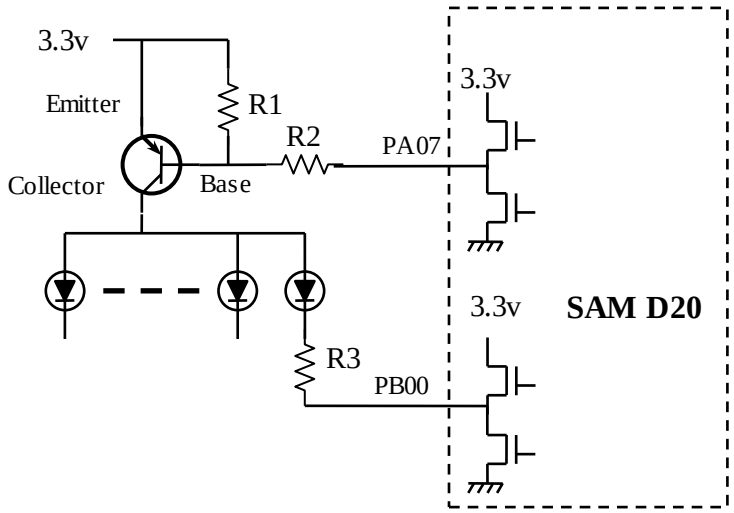

Figure 1.2 Seven-segment display control circuit (only one digit is shown).

- The keypad has 4 row-pins and 4 column-pins (see Figure 1.3). When a button is pressed, a specific row and column are connected together. Think about a button as a light switch. When you turn a switch on, two wires are connected. The row-pins are connected to the same PNP transistors for the 7-segment display. When one of the PNP transistors is turned on, the corresponding row of the keypad is energized (and also the corresponding display digit). At this point, if one of the 4 keys on the energized row is pressed, a high is output to the corresponding column-pin. The resistor R4 in Figure 1.3 is a pull-down resistor and can be replaced by the 'pull-down' option (PULLEN) of the input pin. R4 should be selected so that it does not draw significant amount of current when it is energized.

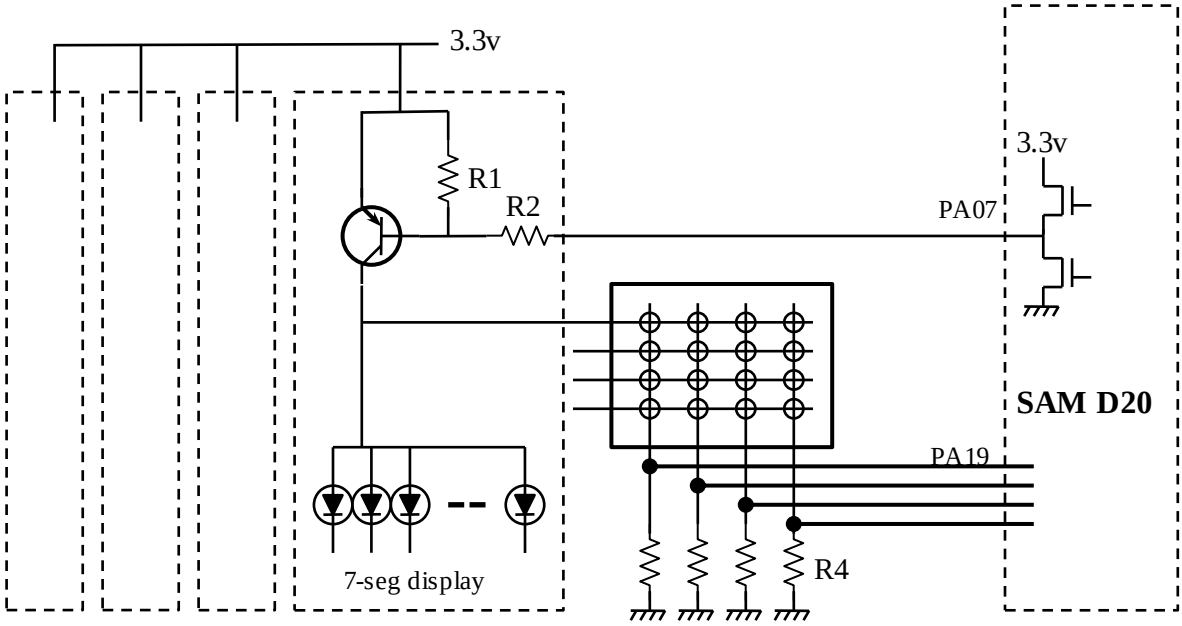

Figure 1.3 Keypad circuit (only one row is shown).

2. Scanning (both display and keyboard)

- In order to read the keypad and to display 4 different numbers on the 4-digit display, they must be time-multiplexed. This means that the SAMD20 must constantly scan the digits and the keypad rows at a fast rate. This scanning action can be done simultaneously for the display and the keypad but it does not have to be. Scanning of the keypad can be done in a short burst but doing so for the display will result in dim or uneven display intensity. The scanning the keypad can be done in a short burst while all display segments are turned off. This momentary blanking of the display is not noticeable. For display, the program must be structured in such a way that each of the 4 digits is turned on about ¼ of the time. This time multiplexing is not necessary if each LED segment and each key has its own dedicated IO pin but doing so will take 7*4+16=44 IO pins of the processor. Using the multiplexing circuit configuration, it only takes 15 pins.

3. BCD to 7-segment decoding

- Once an input is read from the keypad, the input needs to be displayed onto the 7-segment display. Pins PB00 to PB06 are connected to A to G respectively with the decimal point (DP) connected to pin PB07. Digit 1 to Digit 4 are connected to pins PA04 to PA07 respectively. The three extra dots in the display are connected to a common anode. This common anode is connected to the 5th transistor.
- As shown in Figure 1-1, there is a discrete LED that is positioned to the left of the 7-segment display. This discrete LED can be used as the sign of the displayed number.

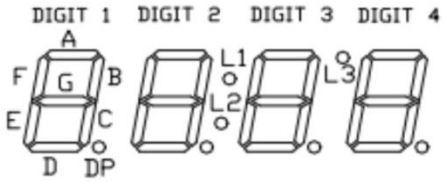

Figure 1-3 Four digits 7-segment display module

4. Software de-bouncing (simple state machine)

   - When a key is pressed, the contact between the row and the column can 'bounce' several times before a stable contact is made. Software de-bouncing is a program that 'filters' out this undesirable bouncing behavior and accepts only one input per key-press. See Lecture note Chapter 2 for a state machine based de-bouncing program.

Input/output Instruction Summary:

The following instructions (under 'SAMD20 Syntax Code) can be used as-is. All symbolic definitions and structure member names (for example, PORT_INSTS and DIRSET.reg) are defined in a header file already.

Address	-	SAMD20 Syntax Code	
0x41004400	-	Port *por = PORT_INSTS;	//set up port struct address
offset 0x00	-	PortGroup *porA = &(por->Group[0]);	//assign pointer to group A ports
offset 0x80	-	PortGroup *porB = &(por->Group[1]);	//assign pointer to group B ports
offset 0x00	-	DIR.reg	//set input/output ports
offset 0x04	-	DIRCLR.reg	//set input port(s)
offset 0x08	-	DIRSET.reg	//set output port(s)
offset 0x10	-	OUT.reg	//set low/high logic to ports
offset 0x14	-	OUTCLR.reg	//set low logic to port(s)
offset 0x18	-	OUTSET.reg	//set high logic to port(s)
offset 0x20	-	IN.reg	//register checking input
offset 0x40	-	PINCFG[x]	//pin configuration
1u << xx	-	PORT_P(A/B)xx	//port syntax

C:\...\("project_name")\src\ASF\sam0\utils\cmsis\samd20\include\component\component_port.h

Required Task:

Task 1: (10% of Lab grade)

- Turn on-board SAMD20 LED on and off (blinking) using only addresses and pointers. The on-board LED is on PORT_PA14.

Task 2: (40% of Lab grade)

- Display 4 unique symbols onto the 7-segment display. *(ex: display "1 2 3 4" or "A, C, 5, F")*
  - Use the appropriate ports corresponding to the external board.
- Use the state machine in lecture note Chapter 2 to read keypad and display the digits on the 7-segment.
  - **Debouncing** - If a button is tapped very quickly, it is not registered as a valid key-press. Also if a button is held down, make sure that your program does not repeatedly accept this key-press multiple times.
  - The display intensity of all 4 digits should be even and they should not be turned off when any of the key is held down.

Task 3: (50% of Lab grade including 25% on report which is due at end of week 3)

- Create a program that can imitate the basic functions of a calculator. It should do the basic addition, subtraction, backspace, and negative numbers. Do not worry about numbers larger than the 4-digit display can hold.
- Input, operation, input, equals
- **Ex:** 321 – (-22) = result

Extra References:

Atmel SAMD20 Datasheet
- Section 5: I/O Multiplexing and Considerations
- Section 8: Product Mapping (address locations)
- Section 13: Clock System

7-segment Display Data Sheet
http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTC-4627JR.pdf
Keypad Data Sheet
http://www.grayhill.com/assets/1/7/Keypads_96.pdf