Analog to Digital Converter/Digital to Analog Converter

Lab 2

Readings:

Atmel SAMD20 DataSheet

- Section 14: GCLK - Generic Clock Controller
- Section 28: ADC - Analog-to-Digital Converter
- Section 30: DAC - Digital-to-Analog Converter

Lab Description:

Students will create two things

1) A voltage meter with the use of the SAMD20's ADC, the external circuit board's 7-segments display and potentiometer (POT).

2) A 1KHz tone that can be heard on the speaker. This will require the use of the SAMD20's DAC and the external circuit board's filters and amplifiers.

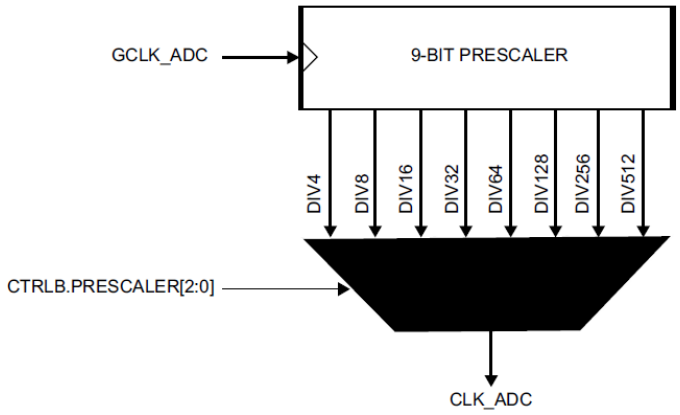
Theory of Operation:

Analog to Digital Converter

1. Generic Clock

   a. Generic Clock setup starts up by turning on a specified module through the power manager. The power manager will power specified modules and allow the user to communicate to the module. In the ADC and DAC setup, the modules are already powered for the user, hence it does not need to be configured. After setting up the power management, the module needs to set up a generic clock source in order to collect or send data at a fixed rate and to enable the clock at the end.

   b. The ADC utilized a generic clock hence it must be configured and enabled to read a voltage. The generic clk is asynchronous to the CLK_ADC, therefore certain registers will need to be configured. The rate of conversion is dependant on the combination of the ADC gclk frequency and prescaler.



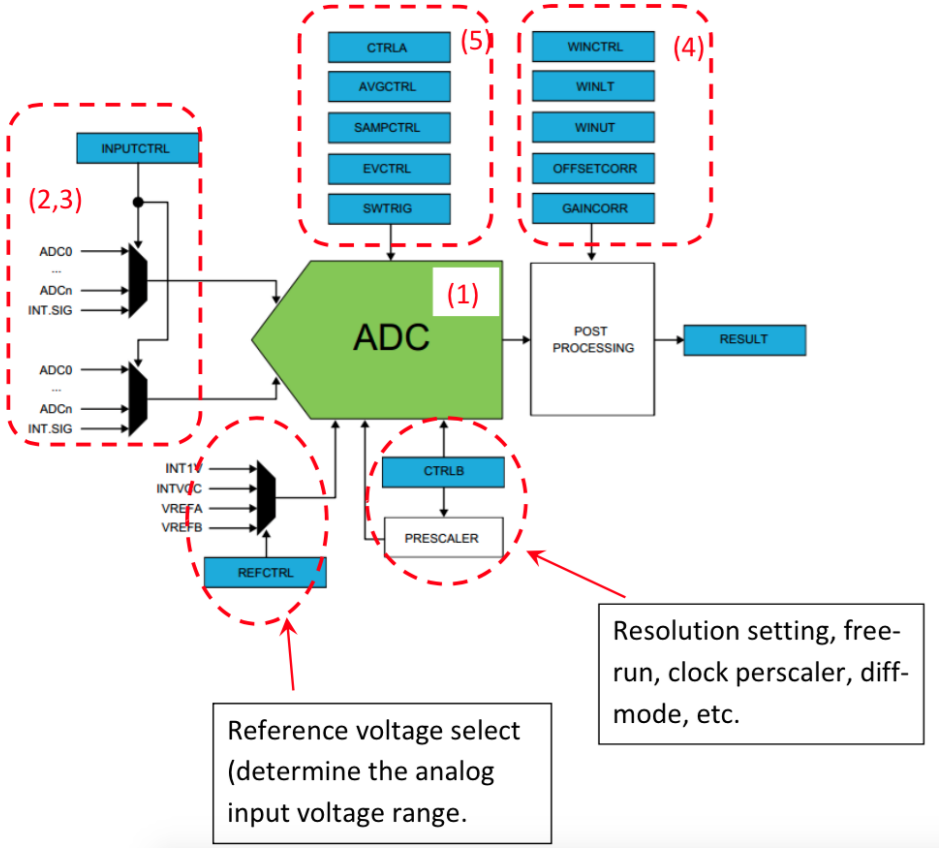
b. The prescaler will set the CLK_ADC at a lower clk rate, scaled from the GCLK_ADC. In order to prescale the GCLK_ADC, one must set the prescaler bit in CTRLB.reg.

c. When the conversion is taking place, the result of the conversion is never instant. There will always be a delay in order to read a voltage level. This is called “propagation delay” (equation in the figure below).

$$PropagationDelay = \frac{1 + \frac{Resolution}{2} + DelayGain}{f_{CLK-ADC}}$$

2. Polling/Interrupt
   a. The ADC utilize interrupts in order to collect data and output it to the user. In this case, we use interrupts (INTFLAG.reg), in order to check when the conversion is result is ready to output or read.

3. Registers
   a. As displayed in the figure below, we can see all the registered revolving around the SAMD20’s on-board ADC.



ADC block diagram (1) One 12-bit, 350ksps ADC with up to 20 channels (2) Differential and single-ended channels (3) 1/2x to 16x gain stage (4) Automatic offset and gain error compensation (5) Oversampling and decimation in hardware to support 13-, 14-, 15- or 16-bit resolution

DAC:

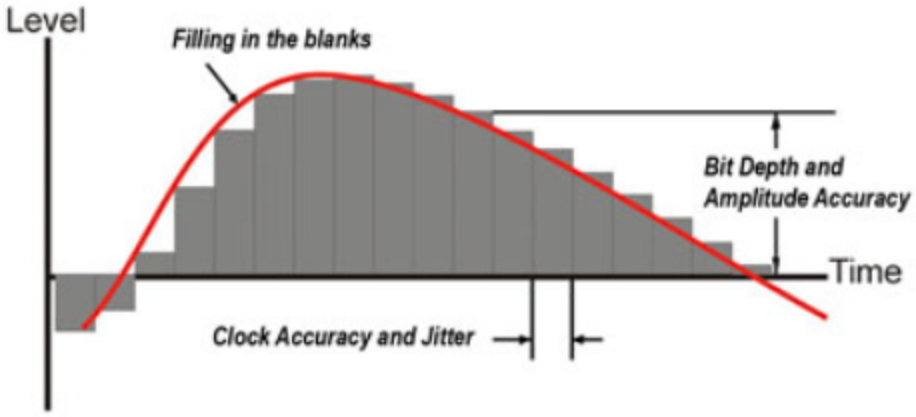
1. Register Synchronization
    a. In order to write to some registers for the DAC the CLK_DAC_APB and GCLK_DAC need to be synchronized.
        i. The following registers need synchronization when written
            1. CTRLA.ENABLE
            2. DATA
        ii. The following registers need synchronization when read
            1. DATA
    b. The following line of code synchronizes by looking at the SYNCBUSY bit in the STATUS register
        i. *while(DacPointer->STATUS.reg & DAC_STATUS_SYNCBUSY){};*

2. Voltage Output
    a. The DAC on the SAMD20 takes in DATA to output the corresponding analog output the following equaiton show the voltage that the DAC outputs.

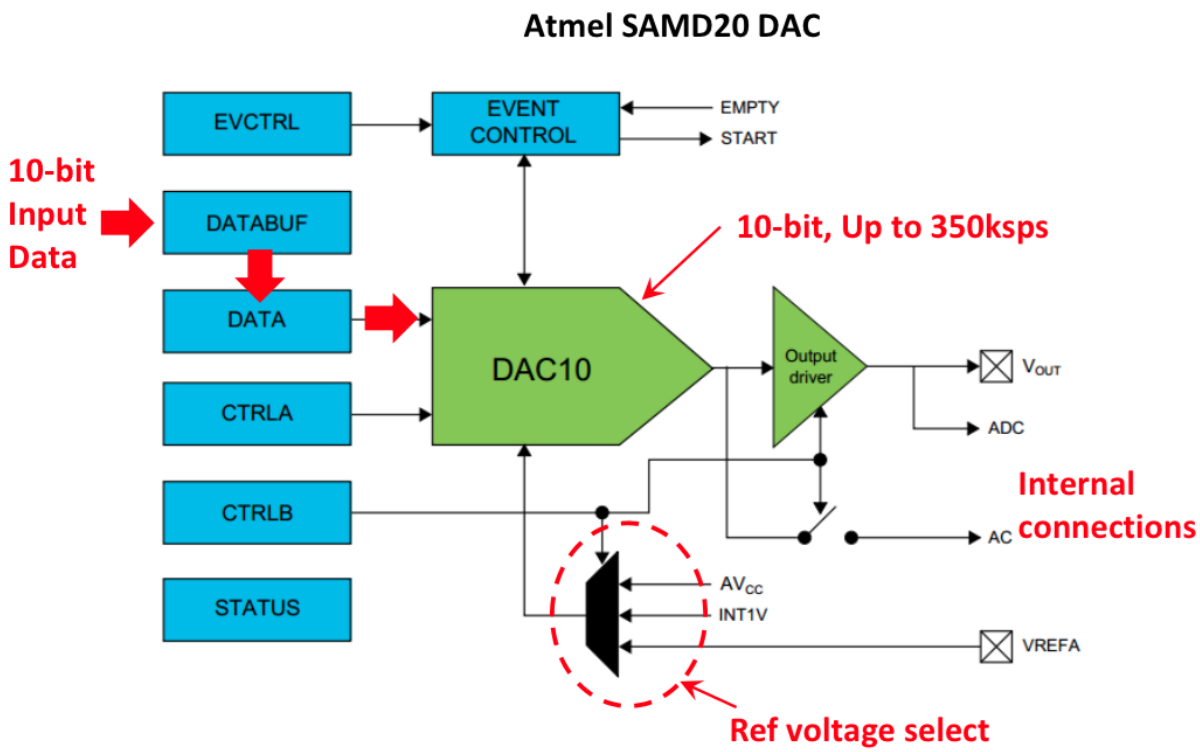
$$V_{DAC} = \frac{DATA}{0x3FF} \cdot VREF$$

The reason that the input DATA is divided by the value 0x3FF is because the DAC is a 10bit dac and the top most value of those 10 bits would be 0x3FF or 0b001111111111. Vref is a reference point that can be altered depending on the chosen input in the CTRLB register.

The DAC outputs DATA as rectangular pulses similar the graph below. These rectangular pulses are then filtered to give a smooth looking waveform.



Sample Output of a DAC



DAC Block Diagram

Peripheral and Coding:

```
Address-        SAMD20 Syntax Code                  "*por" is pointer variable name
------------------------------------------------------------------------------
0x41004400   -     Port *por = PORT_INSTS;              // set up ports instance
offset 0x00  -     PortGroup *porA = &(por->Group[0]);  // set up group A ports
offset 0x80  -     PortGroup *porB = &(por->Group[1]);  // set up group B ports
offset 0x20  -     IN.reg                  // register used to detect a high/low logic
offset 0x40  -     PINCFG[x]               // enables a pin peripheral
1u << xx     -     PORT_P(A/B)xx// port location

0x42004000   -     ADC                     // definition address for ADC functionality
offset 0x00  -     CTRLA.reg               // enable & disable adc
offset 0x01  -     REFCTRL.reg             // reference voltage
offset 0x02  -     AVGCTRL.reg             // average # of samples to be collected
offset 0x03  -     SAMPCTRL.reg            // sampling time control of the ADC clock cycle
offset 0x04  -     CTRLB.reg               // resolution/prescaler
offset 0x0C  -     SWTRIG.reg              // software trigger
offset 0x10  -     INPUTCTRL.reg           // gain/mux input
offset 0x1A  -     RESULT.reg              // result value
offset 0x18  -     INTFLAG.reg             // flag to set interrupt

0x42004800   -     DAC                     // definition address for DAC functionality
```

offset 0x00	-	CTRLA.reg	// enable disable DAC
offset 0x01	-	CTRLB.reg	//Reference selection internal/external output enables
offset 0x07	-	STATUS.reg	// Synchronization Busy Status
offset 0x08	-	DATA.reg	// Data to write

Required Tasks:

***please comment your code**

Task 1:

- Read and display a voltage reading from PORT_PA02 onto the 7-segment Display.
    - 12-bit Resolution
    - Seven Segment display in Volts (display up to at least two decimal place)
    - Reference Voltage of 3.3V
    - **Extra Credit:** display up to 3 decimal places

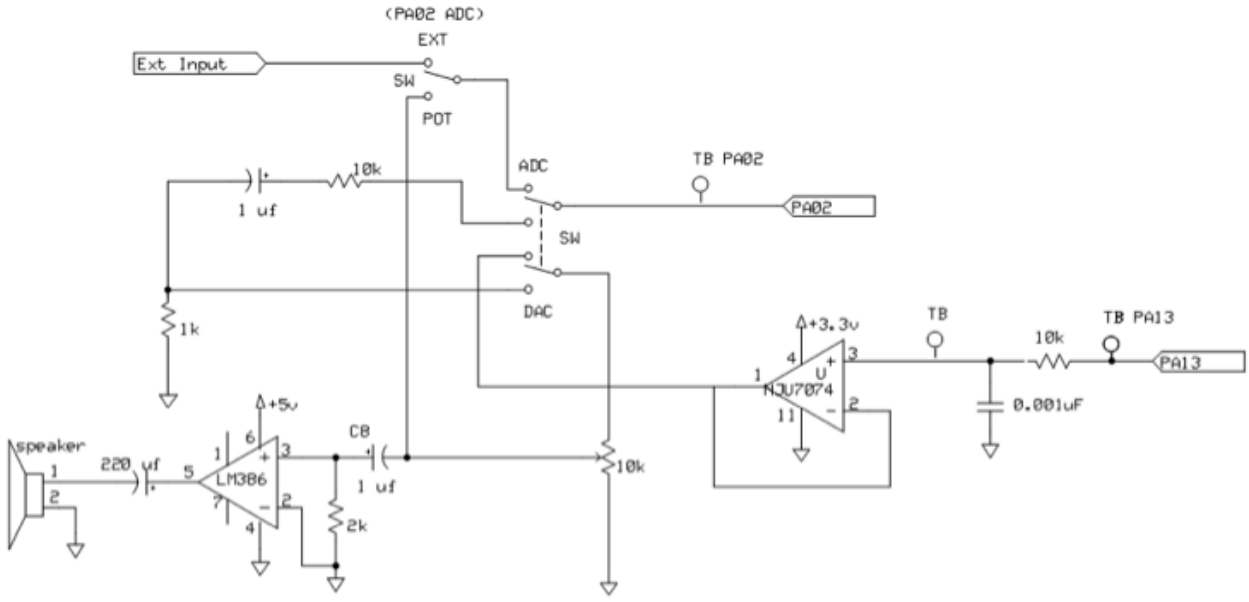
hint: PA13 is the location of your reference voltage for your ADC circuit.

Task 2:

- Output a 1Khz sine wave using the DAC through PA02.
    - Display the waveform on an Oscilloscope
    - Hear the sine wave through the speaker
    - **Extra Credit:**
        - create a musical song
        - use keypad like a piano or soundboard

hint: search for methods of creating a sine wave in hex

ADC and DAC Schematic:



ADC and DAC schematic for lab 2.