Oghenekome Michael

Student Number 7750064

ECE 4240 LAB REPORT

Analog to digital Conversion of Internal Temperature and Position

**Table of Contents**

1 Table of contents

* 1. List of figures……………………………………………………………………………………………..3
  2. List of tables………………………………………………………………………………………………3
  3. Abbreviations……………………………………………………………………………………………..3

2 Abstract…………………………………………………………………………………………………………..4

1. Body of Report………………………………………………………………………………………………...4
   1. Problem and objectives……………………………………………………………………………....4
   2. Design Specification ………………………………………………………………………………...4
      1. General Specification …………………………………………………………………4.
      2. Experiment 1 Specification……………………………………………………………4
      3. Experiment 2 Specification……………………………………………………………4
   3. Hardware Description………………………………………………………………………………..5
      1. TM4C1249 Launch Pad. ……………………………………………………………...5
      2. USB cable……………………………………………………………………………..5
      3. Breadboard …………………………………………………………………………....5
      4. Internal Die Temperature Sensor……………………………………………………...5
      5. Linear Potentiometer…………………………………………………………………..5
      6. Physical Hardware Connection………………………………………………………..5
   4. Software Description………………………………………………………………………………...6
      1. TIVAWARE for C Series. ……………………………………………………………6
      2. Serial communication using Putty. …………………………………………………...6
      3. C program for implementing data acquisition and display……………………………6
   5. Experimental results and Observation……………………………………………………………….7
   6. Conclusions …………………………………………………………………………………………9
2. References……………………………………………………………………………………………………..9
3. Appendices………………………………………………………………………………………………........10
   1. Schematics and Charts………………………………………………………………………………10
   2. Questions to Section 3 of the report manual………………………………………………………...13
      1. Lab 4 Related Questions……………………………………………………………....13
      2. Design Related Questions……………………………………………………………..14
         1. Requirement………………………………………………………………....14
         2. Possible Solution…………………………………………………………….14
         3. Design Element……………………………………………………………...14
4. Specific Requirement…………………………………………………………………………………………15.
   1. TM4C1294 and TM4C1294 Launchpad…………………………………………………………….15
   2. Experiment 1…………………………………………………………………………………………23
   3. Experiment 2…………………………………………………………………………………………27

**1.1 List of Figures**

Figure 1.1 Output of Experiment 1………………………………………………………………………………. 7

Figure 1.2 Output of Experiment 2………………………………………………………………………………. 8

Figure 1.3 CPU block Diagram………………………………………………………………………………...…10

Figure 1.4 Schematic of the TM4C1249 Launchpad GPIO……………………………………………………....10

Figure 1.5 ADC UART Block Diagram…………………………………………………………………………..11

Figure 1.6 ADC Conversion result………………………………………………………………………………..12

Figure 1.7 ADC Signals…………………………………………………………………………………………...12

Figure 1.8: Block Diagram of a Heart Monitor………………………………………………..…………………..15

Figure 1.9: Relation ship between Voltage and Position………………………………..………………………....8

Figure 2.0: Tiva™ TM4C1294NCPDT Microcontroller High-Level Block Diagram……………………………16

Figure 2.1: ADC Module Block Diagram…………………………………………………………………………23

Figure 2.2: Internal Temperature Sensor Characteristic………………………..…………………………..……..24

Figure 2.3: Implementation of 2 ADC blocks……………………………………………………………………..24

Figure 2.4: ADC Voltage Reference………………………………………………………………………………25

Figure 2.5: Linear Potentiometer ………………………………………………………………………………….27

Figure 2.6: (a) Diagram of interfacing of the linear potentiometer (b) Demonstration of the midpoint …….……28

**1.1 List of Tables**

Table 1.1: Memory map of the TM4C1294 memory.………….………………………………………………... 22

Table 1.3: Linear relationship of position sensor………………………………………………………………....13

Table 2.1: Lab 4 manual table 1.1……………………………………………………………………….……..…25

Table 2.2: Booster Pack 1 for TM4C1294 Launchpad ………………………………………………………..…23

* 1. **Abbreviations**

ADC ………………………………………………………………………………….Analog to Digital Converter

PWM ………………………………………………………………………………………Pulse Width Modulator

GPIO ………………………………………………………………………………General Purpose Input Output

**2.0 Abstract**

This report aims to show the acquisition of analog signals and converting it to meaningful digital signal that we can process using digital logic. We use the ADC in the TM4C1249CPDT launchpad to convert signals, either temperature or resistance (position) and the internal temperature sensor on the board as well to get the analog temperature of the surrounding. This analog signal is converted to digital values that is displayed on a screen for readability with the use of internal UART using RS232 protocol.

Along with this report is some questions regarding critical concepts of Analog to Digital conversion. An experiment is also carried out to get the characteristics of a sliding potentiometer

**3.0 BODY OF THE REPORT**

**3.1 Problem and Objectives**

In the real world, signals from sensors like temperature, pressure and position are measured on a continuum of time with infinite resolution in magnitude value. However, we need to measure these signals and display them for readability and the result makes sense. The approach we take is that of converting analog sensor signals which is continuous on time to samples of the signal in discrete time using the specifications of the ADC on the Cortex M4 Micro-Controller. This conversion is possible because of the ability to represent these signals as voltages and currents. Another question arises as to how long or short should be the period of sampling such that the constraint for real time is still satisfied.

The objective of this report is to explain how we can maintain real time data acquisition of temperature and position values from its analog sensing input to digital representation of the quantities at any given time either when requested or on change, hence satisfying real time processing. This should also be achieved using the Tiva-Ware framework and give details as to how the sensor selection, estimation and sampling frequency is ideal for the experiment.

**3.2 Design Specification**

**3.2.1 General Specification**

* Use of the Universal Asynchronous Receiver and Transmitter (UART) for displaying the result.
* Use of the internal sensor and ADC of the Cortex M4 Microprocessor for temperature sensing.
* The code should be written in C programming language, Libraries are also allowed.
* Translate the digital code to readable form

**3.2.2 Experiment 1 Specification**

* Define a sampling rate for the temperature using triggered interrupt.
* Use hardware Averaging to minimize noise in the temperature signal.

**3.2.2 Experiment 2 Specification**

* Use of a linear position sensor
* Hardware and Software Interfacing
* ADC analog input configuration

**3.3 Hardware Description**

There are many hardware used in the Lab. Their function and what role they play in each of the experiments are explained below.

**3.3.1** **TM4C1249 Launch Pad.**

As will be explained in detail later in this report, the TM4C1249 is a powerful micro controller. This is the hardware where the Cortex M4 Micro-Processor is mounted on. On the output pins of the processor are busses and connections that enable interfacing with other peripherals. The launchpad contains the GPIO pin out lines of 3.3V or Ground 0V as well as the SPI and I2C bus connections. Note that some of these pins can be configured to be used as analog or digital input or output pins. The launchpad is powered by a USB connector that connects to a computer with 5 Volts power. The Launchpad houses many functional units and busses that will be explained in later section of the report.

Components of the Launchpad necessary for the experiments are:

* Internal ADC of the processor
* GPIO output pins

**3.3.2 USB Cable**

This is used to power the launchpad by providing 5 Volts from the Connecting Computer or device to the TM4C1249 Launch Pad. This also serves as an interfacing medium for programming the device. The hardware for the micro processor is configured for the application using the USB cable. Furthermore, the USB is used as an port COM communication for UART Serial communication with the host computer. Therefore, the USB cable serves multiple purpose and is crucial to the working of the project and satisfying part of the specification.

**3.3.3 Breadboard**

The bread board acts as a connector between the physical external non-microcontroller inter connections. The breadboard houses the launchpad GPIO pins as well as other external peripheral needed to be interfaces with the launchpad. It is an array of internally connected wires which keeps the external connection simple. For this lab, it is very helpful to have a breadboard to connect the sliding potentiometer to the board.

**3.3.4 Internal Die Temperature Sensor**

This is the microprocessors temperature sensor. It primarily serves two purposes. First, is to notify the system about the extreme internal temperatures and second is to provide temperature measurement for calibration. The internal temperature sensor converts the temperature measurement into a voltage value. This is always enabled in the system, therefore does not need to be enabled. However, we need to specify this as our analog input to use the values it outputs. [TI14 sec 15.3.6]

**3.3.5** **Linear Potentiometer**

This is a variable resistor which is used to vary the resistance between 0Ω and 10kΩ. The resistor has three pins used for interfacing. It has a ground pin and a Vdd pin. These two pins are used to power the device. There is a third pin used for varying the resistance, hence, varying the voltage between 0V ground and Vdd. [Kins2 Sec 2.3.4]

**3.3.6** **Physical Hardware Connection**

For experiment 1, the external (non launchpad) physical connections that must be made is the connection between the Host computer and the board for serial communications using the USB cable. Every other connection i.e. between the ADC, processor and GPIO pins are all on board connections.

However, for experiment 2 there is a connection between the pin 2 of the potentiometer as in figure 2.6and an analog GPIO pin of the Launchpad. This connection represents the voltage corresponding to the resistance of the variable resistor.

**3.4 Software Description**

The software used in the Lab are described below as follows:

**3.4.1 TIVAWARE for C Series.**

This is a framework that includes libraries of peripherals, include statements, drivers and other libraries for accessing the hardware of the device i.e. the memory, registers and busses. This library is compiled using the C language. This encapsulates the API, devices and drivers. Detailed explanation on this Tiva-Ware will be explained later. However, it is crucial to note that in this report observations and findings, we will concretely implement functions and variables from this framework to initialize the ADC and the UART.

**3.4.2 Serial communication using Putty.**

This is an SSH and Telnet client used for serial control and data display using the terminal console of the computer.

**3.4.3 C program for implementing data acquisition and display.**

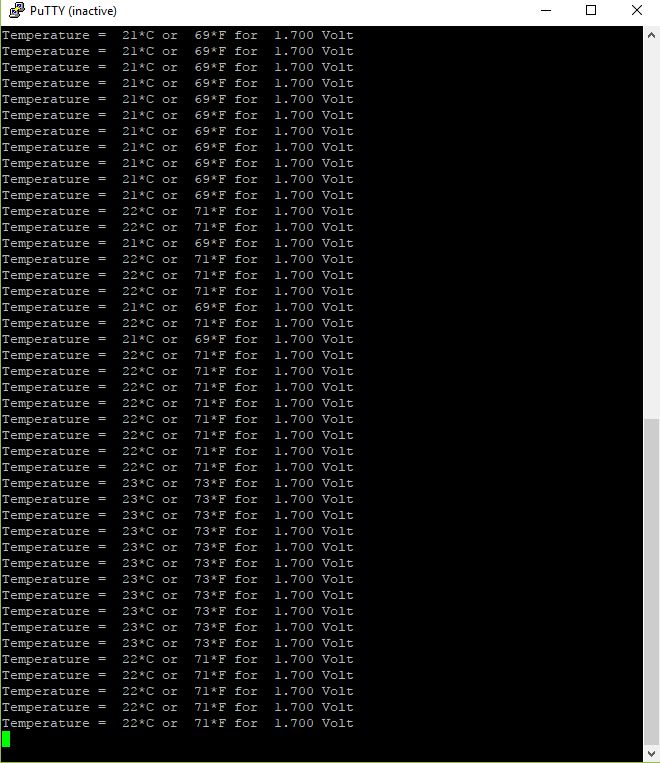
In addition to the libraries, we write C software code to implement:

* **Data acquisition:** This is the code necessary for setting up the ADC i.e. the initialization and the sampling and getting the discretized values into digital values we can then process later. This process involves configuring the ADC, initializing the value to get the data into and the ADC0 result buffer. Then the ADC Processor is triggered after which because of Successive approximations, we need 12 ADC clock cycle to wait for the data to be converted and an interrupt is raised. After which we clear the interrupt and get the data using one of the Tiva ware API Data get function. [Kins2]
* **Decoding**: After the digital data is acquired, a C code is necessary to convert the value of the temperature or position the sensor gives us to the correct physical value. The ADC converts the analog signal into digital values based on the value compared to the full scale of the analog signal. Hence, we need to know the value of the output at a certain temperature or position to measure the temperature for experiment 1 and position for experiment 2. Explanation for the decoding can be found in the datasheet [TI14 Sec 15.5.3]
* **Display**: After we get the data we are going to display through the serial interface, we must have previously initialised the UART and then send it to the output COM using the USB by implementing functions from the UART library of drivers and include from the Tiva Ware. (See section 6.2)

**3.5 Documented result and observations**

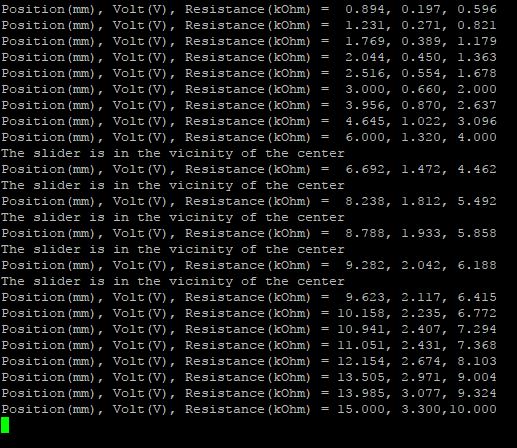
Below are the Serial outputs of both experiments from Putty:

* **Experiment 1:** Included in the digitised result is the output of the ADC, the conversion of this value to its corresponding temperature in Celsius and then Fahrenheit.



**Figure 1.1: Output of Experiment 1**

* **Experiment 2:** Included in the result is the output of the digitised value of the voltage, the Value of the voltage and the value of the resistance at the corresponding position from the 0mm position.

****

**Figure 1.2: Output of Experiment 2**

**Figure 1.9: Relation ship between Voltage and Position**

**3.6 Discussion of the result obtained.**

The results and observations regarding the results for the experiments are:

**Experiment 1 – Internal temperature sensor:** The ADC in the microprocessor produced digital values for the temperature. The values produced are in 12 bits representation of a Full-scale range of -40 to 85 Celsius which when converted to a voltage representation, gives VTSENS = 2.7 - ((TEMP + 55) / 75) as the voltage. When Temp is -40, the voltage is 2.5V and then the temperature is 85, the voltage is 0.83 Volts and then the temperature can be converted to the desired scale either in Celsius or Fahrenheit

**Experiment 2 –** **Position Measurement:** The result of the hardware sensor produced a voltage corresponding to the voltage of the position of the slider with respect to the ground voltage. As explained above, the resistance at that position is directly proportional to the voltage according to the full-scale voltage which is 3.3 volts.

The output of the ADC is the increment of the voltage in the corresponding steps of the ADC which is the resolution of the ADC. Because the ADC resolves the voltage into 12 bit, the ADC resolution is in increments of 2number of bits which means the ADC gives the digital equivalent of the voltage in full scale of 4096 corresponding to F.S voltage which is 3.3 volts.

From the graph in figure 1.9,we can see that the relationship between the voltages to the position is **Linear**  With the maximum distance of 15mm.

**3.7 Conclusions**

As seen from the result and observation above, Analog signal that are quantified in the real world as continuous signals. However, these analog signals can be represented in terms of voltages and currents. These voltages are converted into 2^N steps of values where the values correspond to the voltage level and N is the number of bits for sampling. This is repeated at the sampling frequency to get samples at different time.

1. **REFERENCES**

[Diff] Differencebetween.net. (2017). Difference Between - Difference Between AHB and APB. [online] Available at: http://www.differencebetween.net/technology/difference-between-ahb-and-apb/ [Accessed 4 Dec. 2017].

[Kins1] Witold Kinsner, *Microcontroller, Microprocessor, and Microcomputer Interfacing for Real Time Systems*. Lecture Notes. Winnipeg, MB. University of Manitoba, Dept. ECE, May 2017, 643 pp.

[Kins2] Witold Kinsner, *Laboratory 4,* *Analog to Digital Conversion of Internal Temperature and position*. Laboratory Manual Notes. Winnipeg, MB. University of Manitoba, Dept. ECE, Nov 2017, 36 pp.

[Omeg] Omega.ca. (2017). OMEGA ENGINEERING - Data Acquisition FAQ. [online] Available at: https://www.omega.ca/techref/das/se-differential.html [Accessed 4 Dec. 2017].

[TI14] Texas Instruments (TI), *Tiva TM4C1294NCPDT Microcontroller, Data Sheet*. Plano, TX: Texas Instruments, 2014, 1890 pp, Available as of Sep 8, 2107 from TI at <http://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf>

[Valv] Jonathan W. Valvano, *Embeded System, Introduction to ARM Cortex Microcontrollers.* Volume 1. Middletown, DE: Create Space Independent Publishing Platform , 2016 (5th Ed.), 508 pp. {ISBN: 978-1-4775-0899-2, pbk; US$41.75; Kindle: $7.54} Available as of July 12 from Amazon at <http://www.amazon.com/Embedded-Systems-Introduction-Arm%AE-Cortex/dp/1477508996> eBook <http://www.amazon.com/Embedded-Introduction-Cortex-M-Microcontrollers-ebook/dp/B00DXVBRSC>

1. **APPENDICES**

**5.1** **Schematics and Charts**

**CPU block Diagram**

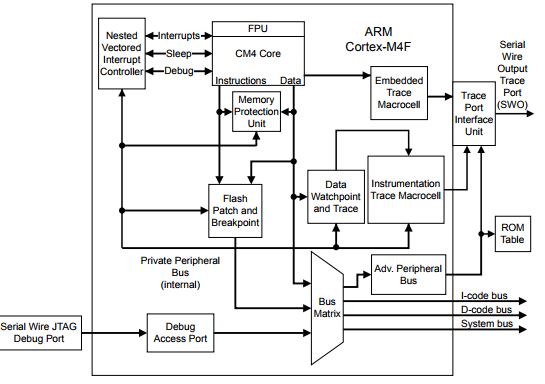
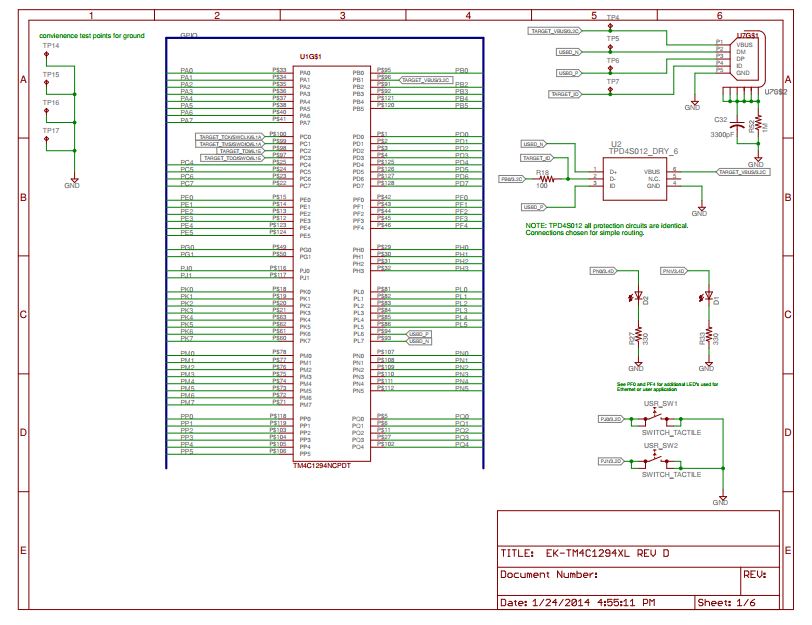


Fig 1.3 CPU block Diagram [TI14 Sec 2.1]

Figure 1.4 Schematic of the TM4C1249 Launchpad GPIO



UART block Diagram

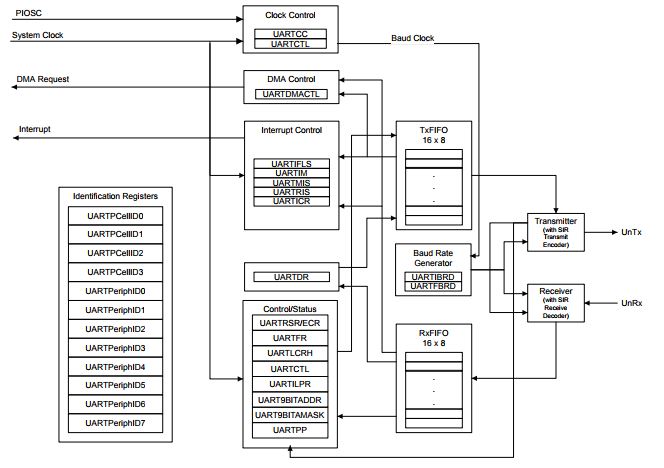


Figure 1.5 ADC UART Block Diagram [TI14 Sec 16.1]

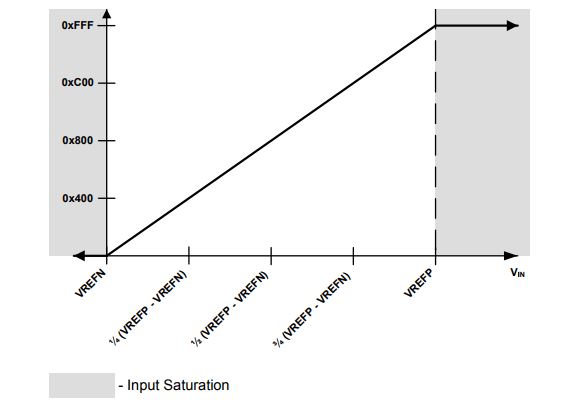
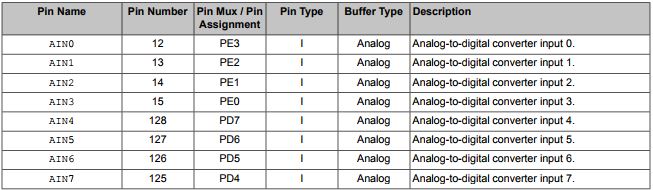


Figure 1.6 ADC Conversion result [TI14 Sec 15.3.4.1]



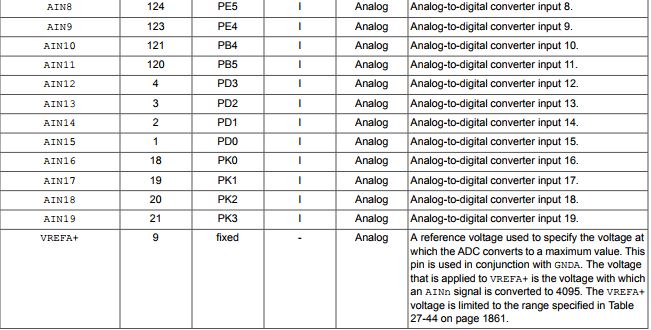


Figure 1.7 ADC Signals [TI14 Sec 15.2]

|  |  |  |
| --- | --- | --- |
| Position | Voltage | Resistance |
| 1.612 | 0.355 | 1.074 |
| 1.982 | 0.436 | 1.321 |
| 2.326 | 0.512 | 1.551 |
| 2.637 | 0.58 | 1.758 |
| 3.073 | 0.676 | 2.049 |
| 3.604 | 0.793 | 2.403 |
| 4.165 | 0.916 | 2.777 |
| 4.755 | 1.046 | 3.17 |
| 5.297 | 1.165 | 3.531 |
| 5.817 | 1.28 | 3.878 |
| 6.333 | 1.393 | 4.222 |
| 6.839 | 1.505 | 4.559 |
| 7.319 | 1.61 | 4.879 |
| 7.901 | 1.738 | 5.287 |
| 8.663 | 1.906 | 5.775 |
| 9.366 | 2.061 | 6.244 |
| 10.238 | 2.252 | 6.825 |
| 11.092 | 2.44 | 7.394 |
| 11.989 | 2.638 | 7.993 |
| 12.872 | 2.832 | 8.581 |
| 13.74 | 3.023 | 9.16 |
| 14.52 | 3.194 | 9.68 |
| 15 | 3.3 | 10 |

Table 1.3: Linear relationship of position sensor.

**5.2** **Questions to Section 3 of the report manual**

**5.2.1 Lab 4 Related Questions**

**Question 1**: **How many bits per sample does the ADC in the TM4C1294 microprocessor have?**

The TM4C1294 microprocessor has 12 bits for every sample for the ADC

**Question 2: How fast is the ADC?**

The ADC maximum sample rate to two million samples/second p76

**Question 3: What class is the ADC in the TM4C1294 microprocessor have?**

The class of ADC used in the TM4C1294 microprocessor is the serial converter Successive Approximation

**Question 4: This class of ADC has a very important feature for real time data acquisition. What is it?**

The feature of the Successive Approximation class is that it takes exactly the number of bits steps to produce the Digital representation of the binary code. And because we know the constant time of the A/D conversion, real time calculation are precise.

**Question 5: Is the ADC in the TM4C1294 Big Endian or Little Endian?**

The ADC is big endian because Successive approximation converts using big endian.

**Question 6: How many analog input channels can the ADC handle?**

20 analog input channels.

**Question 7: Does the ADC have a single ended or a differential input?**

The ADC has both a single ended input and a differential input configurations.

**Question 8: What is the advantage of single ended or a differential input?**

The single ended input is lower in cost and provides twice the number of input for the same size wiring connector Since it requires one analog high (+) and one LLGND (-).

The Differential input produces stable reading with the presence of Electromagnetic radiation or Radio Frequency Interference. [Omeg]

**Question 9: Is the clock of the ADC the same as the System Clock?**

The clock the ADC uses is the Global alternate clock (ALTCLK) or the System Clock (SYSCLK). Either one is used to generate the ADC clock

**Question 10: The ADC has Sample and Hold (S&H) device. Why is this important?**

The sample and hold device is required to freeze the continuous sample for a period of time so that sampling and conversion can take place for the current analog sample and when the next sample comes after a specific time, greater than the conversion time, it releases the value and freezes the next voltage. It is important because if the conversion time is too long, there is enough time for sampling the next signal before after conversion.

**Question 11: What is a sampling sequencer on the TM4C1294 micro controller?**

The sampling sequencer on the controller allow rapid sampling for multiple analog input sources without the help of the micro controller.

**Question 12: Is oversampling allowed on the ADC, what is the highest oversampling on the ADC?**

Yes, oversampling is allowed on the ADC. The highest over sampling on the ADC is 64x hardware oversampling.

**Question 13: What is the role of VREFA+ on the ADC?**

The voltage that is applied to VFERA+ is the voltage with which an AINn signal is converter to 4095. It is limited to the range. It is a reference voltage used to specify the voltage at which the ADC converts to a maximum value.

**Question 13:** Is the power and ground on the analog module the same as the power and ground on the digital module?

No, they are different. They should be separate because the power on the analog module is varying from 0 to Full scale while the digital module had fixed values and options.

**Question 13: How do we trigger sampling on the ADC?**

Sampling of the ADC is triggered using interrupts as trigger signal from the software on the GPIO ADC control register(GPIOADCCTL). As seen from the code, we wait for the interrupt flag to be set before starting conversion. The interrupt is generated on the rising edge and the ADC is then triggered on the falling edge. [TI14 Sec 15]

Although the ADC on the TM4C1294 has multiple triggering sources. I.e. Internal software based triggering, Timer based triggering, analog comparator triggering and PWM and as mentioned above, GPIO based triggering.

**5.2.2 Design Related Questions**

**5.2.2.1 Requirements**

* The Amplifier/ signal conversion circuit must be powered by a low voltage, current restricted battery.
* The output of this module must be safely interfaced to a remote computer for digital processing.

**5.2.2.2 Possible Solution**

We isolate the data acquisition and conversion input and the output which is connected to a computer for processing with optical transducers. However, these are non-linear in transferring signal and a negative feedback could violate the isolation and the effect could be dangerous.

**5.2.2.3 Design Element**

**Select an alternative optimal ADC architecture that gets around this linearity problem (As discussed in class)?**

We use the integrating A/D converter [Kins1 chap 4 p3-46]

**Explain the options considered and their potential for the solution**

Other options considered is the Successive Approximation A/D converter. The reason for this being a candidate is that it is very precise to calculate how long it takes to make the conversion because it only converts in the number of bits step.

**Explain the fundamental reason for your final decision**

The reason for the decision is because the integrator A/D converter is monotonic i.e. the first derivative never changes its sign and monotonicity is critical in applications that such as feedback loop. If a negative feedback changes to positive, there is catastrophic failure. [Kins1 chap 4 p3-46]

**Draw a block diagram for your heart monitor**

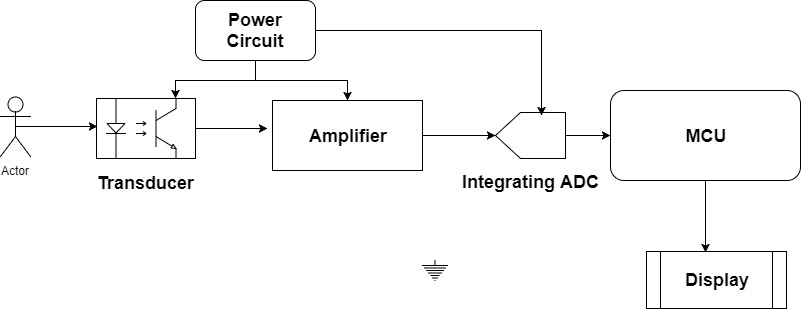


Figure 1.8: Block Diagram of a Heart Monitor

**6.0 SPECIFIC REQUIREMENTS**

**6.1 The TM4C1294 and TM4C1294 Launchpad**

1. **Draw a block diagram of the TM4C1294 micro controller showing its units and busses:**

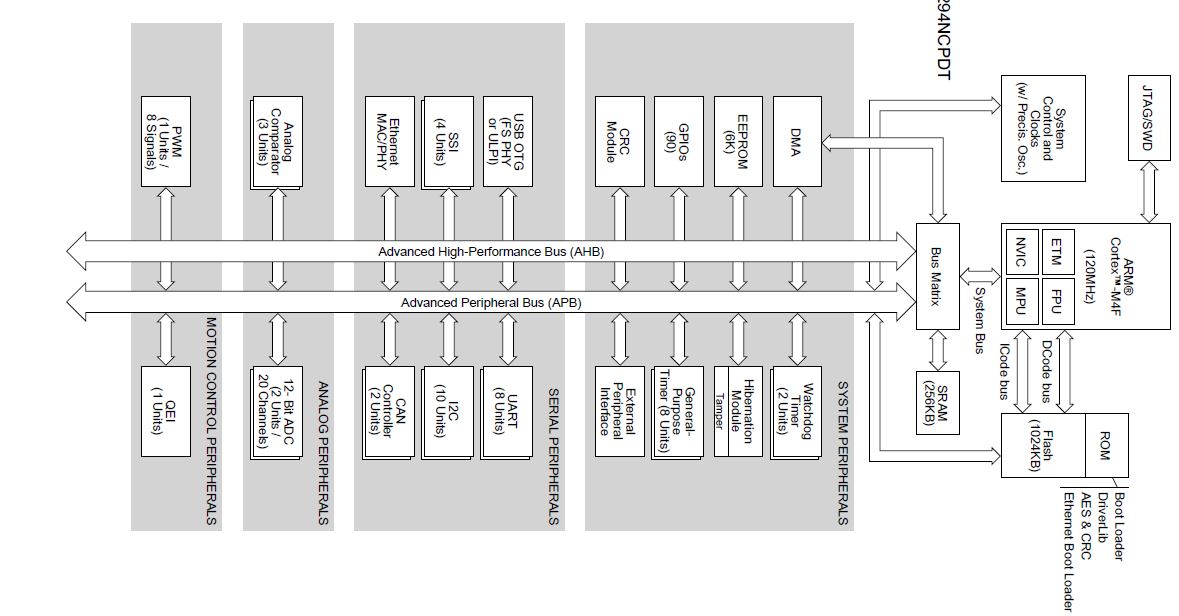
****

FIGURE 2.0: Tiva™ TM4C1294NCPDT Microcontroller High-Level Block Diagram [TI14 Sec 1.2]

* **Show and describe all the units**

**ARM Cortex-M4F Processor Core**: This provides a minimal optimal efficient core structure for the high performance, low cost platform that meets the needs of minimal memory implementation. This includes;

* + **Processor core:** A 32bit architecture. With thumb 2 mixed 16/32bit instruction to deliver high performance.
  + **System Timer (Systick):** This is a Cortex M4 integrated system timer that which provides a clear on write decrementing, wrap on zero counter with a flexible control mechanism. It can be used as a RTOS tick timer, clock source, a counter and as explained above used to model other timers like the ADC timer amongst others.
  + **Nested Vector Interrupt Controller (NVIC):** This is used to prioritize and handle exception in handler mode.
  + **System Control Block (SCB):** This is used to provide system implementation of information and system control including configuration control and reporting of System exceptions
  + **Memory Protection Unit (MPU):** IT provides full support for protection regions, overlapping protection regions, access permissions and sending the memory access permissions.
  + **Floating point unit (FPU):** This offers a single precision arithmetic like add, subtract, divide and multiply as well as accumulate and square root operations. This units also makes the processor convert between fixed and floating-point data format and floating-point constant instructions.

**On-Chip Memory:** This is the memory the micro controller has direct access to use for on chip processing. This includes;

* + **Static Random-Access Memory (SRAM):** This gives allows the microcontroller access to a 256KB of single cycle-on-chip SRAM. It is located at offset 0x2000.0000 of the device memory map. Data can be transferred to the RAM through the micro-DMA, USB and an Ethernet Controller. To reduce the number of time consuming read-modify-write (RMW) operations, ARM provides bit-banding technology in the processor.
  + **Flash Memory:** The TM4C1294NCPDT microcontroller provides 1024 KB of on-chip Flash memory. The Flash memory is configured as four banks of 16K x 128 bits (4 \* 256 KB total) which are two-way interleaved. Memory blocks can be marked as read-only or execute-only, providing different levels of code protection [TI14 Sec 1.3.2.2]
  + **Read Only Memory (ROM):** The ROM is program with Tivaware peripheral driver library, Tivaware Boot loader, Advanced Encryption Standard (AES) cryptography tables and Cyclic Redundancy Check (CRC) error-detection functionality.
  + **EEPROM:** This is features a 6KB of memory accessible as 1536 32-bit words with 96 blocks of 16 words each, Built-in wear leveling, Access protection per block with Lock protection option for the whole peripheral as well as per block using 32-bit to 96-bit unlock codes (application selectable). It also features Interrupt support for write completion to avoid polling and Endurance of 500K writes to 15M operations per each 2-page block. [Datasheet pp59]
* **Show and describe all the busses**

**Advanced High-Performance bus:** This connects all the interfaces and peripherals in the microcontroller. Both the system peripheral, the Serial peripheral, the analog peripherals and the Motion control peripherals. Both the AHB and the APB are on chip Bus standards. The Advanced High-performance Bus is capable of waits, errors and bursts. The ADH, which is pipelined, mainly connects to memories. [Diff]

**Advanced High-Performance bus:** This connects all the interfaces and peripherals in the microcontroller. Both the system peripheral, the Serial peripheral, the analog peripherals and the Motion control peripherals. When comparing the usage, the APB is simpler than the AHB. Unlike the AHB, there is no pipelining in APB. The APB is mainly proposed for connecting to simple peripherals. Looking at the AHB and the APB, the APB comes with a low power peripheral.

It can also be seen that Advanced Peripheral Bus is sometimes optimized for reduced interface complexity and minimal power consumption for supporting peripheral functions. This Bus can also be used in union with either

version of the system bus. [Diff]

**System Bus:** This includes micro DMA and USB and Ethernet. This bus is used when data needs to be exchanged between a device and the processor without the processor having to fetch the data by itself. This bus helps increase speed and overall efficiency of memory retrieval and input.

**CPU instruction bus (I code Bus) :** This bus runs through the microprocessor and the memory. It is mainly used to fetch instructions used by the processor. This is usually accessed before any program is run in any given instruction clock cycle.

**CPU Data bus (D code Bus):** This bus is critical in transferring Data to and from the processor. Computations and processing is done in the Arithmetic and Logic unit of the processor but Data is stored in memory hence a need for a bus to both fetch data for processing and store data after the data is being processed

* **Show and describe all the interfaces**

**Cyclical Redundancy Check (CRC)** The TM4C1294NCPDT microcontroller includes a CRC computation module for uses such as message transfer and safety system checks. [TI14]

**Serial Communication**: This can further be subdivided into dofferent aspects:

* **Ethernet MAC and PHY:** This is mainly used for wireless communication
* **CAN: Controller Area Network.** It is used to connect electronic control units. It is really efficient in electromagnetic noise filled environment, it is now used for Embeded application with a bit rate of up to 1Mbps.
* **Universal Serial Bus:**  This is a standardised interface for connection between peripherals and devices. It is a serial bus that allows connection without having to reboot the system after connection.
* **UART:** This is a universal Asynchronous Receiver and Transmitter. It used serial communications between devices that have either receiver, transmitters or a combination of the two. It has a programmable baud rate allowing speeds up to 7.5Mbps
* **I2C:** This is used to provide a bi-directional data transfer between two devices through a 2-wire design. This uses the Master slave protocol to transfer data where one master can service different slaves and the slaves can communicate with the masters. Communication is both ways with this protocol
* **QSSI:** This is a Quad Synchronous Serial Interface which is capable of converting data between parallel and serial form. It converts serial-to-parallel conversion on data received from a peripheral device, and vice versa on data transmitted to a peripheral device.

**System Integration**: A few of some of the functions integrated into the device which also interfaces are:

* **Direct Memory Access**: This is used to ease the processor having to transfer data to and from memory so it can do other computations.
* **System Control and Checks**: This This determines the overall operation of the device. [TI14 Sec 1.3.6.2]
* **Programmable timers**: This counts external events that drive the timer input pins [TI14 Sec 1.3.6.3]
* **CC Pins:** Capture compare PWM pins Is used to count external events just line the programmable timers but with the CCP pins as input. The General-Purpose Timer can also generate PWM on this pin.
* **Hibernation module**: The QSSI module performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data transmitted to a peripheral device [TI14 Sec 1.3.6.5]
* **Watchdog Timers**
* **Programmable GPIO**
* **Advanced Motion Control**
* **PWM**
* **QEI**

**Analog:** There are two analog functions integrated into the device. They are

* **ADC:** This is used for converting Analog Data into Digital data for digital processing. The ADC converts voltages or current into discrete digital voltages.
* **Analog Comparator:** This is used to compare two voltages and gives an output depending on the condition on the comparison result. The output is also an analog signal as well.

**JTAG and ARM Serial Wire Debug**: The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic [TI14 Section 1.3.9]

1. **Draw a Memory Map of the TM4C1294 micro controller and comment on the logical partitioning on the memory:**

**Below is a table of the memory map of the microcontroller.**

**Start End Description**

**Memory**

**0x0000.0000 0x000F.FFFF On-chip Flash**

**0x0010.0000 0x01FF.FFFF Reserved -**

**0x0200.0000 0x02FF.FFFF On-chip ROM (16 MB)**

**0x0300.0000 0x1FFF.FFFF Reserved -**

**0x2000.0000 0x2006.FFFF Bit-banded on-chip SRAM**

**0x2007.0000 0x21FF.FFFF Reserved - Bit-band alias of bit-banded on-chip SRAM starting at0x2000.0000**

**0x2200.0000 0x2234.FFFF**

**0x2235.0000 0x3FFF.FFFF Reserved -**

**Peripherals**

**0x4000.0000 0x4000.0FFF Watchdog timer 0**

**0x4000.1000 0x4000.1FFF Watchdog timer 1**

**0x4000.2000 0x4000.3FFF Reserved**

**0x4000.4000 0x4000.4FFF GPIO Port A**

**0x4000.5000 0x4000.5FFF GPIO Port B**

**0x4000.6000 0x4000.6FFF GPIO Port C**

**0x4000.7000 0x4000.7FFF GPIO Port D**

**0x4000.8000 0x4000.8FFF SSI0**

**0x4000.9000 0x4000.9FFF SSI1**

**0x4000.A000 0x4000.AFFF SSI2**

**0x4000.B000 0x4000.BFFF SSI3**

**0x4000.C000 0x4000.CFFF UART0**

**0x4000.D000 0x4000.DFFF UART1**

**0x4000.E000 0x4000.EFFF UART2**

**0x4000.F000 0x4000.FFFF UART3**

**0x4001.0000 0x4001.0FFF UART4**

**0x4001.1000 0x4001.1FFF UART5**

**0x4001.2000 0x4001.2FFF UART6**

**0x4001.3000 0x4001.3FFF UART7**

**0x4001.4000 0x4001.FFFF Reserved -**

**0x4002.0000 0x4002.0FFF i2C 0**

**0x4002.1000 0x4002.1FFF i2C 1**

**0x4002.2000 0x4002.2FFF i2C 2**

**0x4002.3000 0x4002.3FFF i2C 3**

**0x4002.4000 0x4002.4FFF GPIO Port E**

**0x4002.5000 0x4002.5FFF GPIO Port F**

**0x4002.6000 0x4002.6FFF GPIO Port G**

**0x4002.7000 0x4002.7FFF GPIO Port H**

**0x4002.8000 0x4002.8FFF PWM 0**

**0x4002.9000 0x4002.BFFF Reserved -**

**0x4002.C000 0x4002.CFFF QEI0**

**0x4002.D000 0x4002.FFFF Reserved -**

**0x4003.0000 0x4003.0FFF 16/32-bit Timer 0**

**0x4003.1000 0x4003.1FFF 16/32-bit Timer 1**

**0x4003.2000 0x4003.2FFF 16/32-bit Timer 2**

**0x4003.3000 0x4003.3FFF 16/32-bit Timer 3**

**0x4003.4000 0x4003.4FFF 16/32-bit Timer 4**

**0x4003.5000 0x4003.5FFF 16/32-bit Timer 5**

**0x4003.6000 0x4003.7FFF Reserved -**

**0x4003.8000 0x4003.8FFF ADC0**

**0x4003.9000 0x4003.9FFF ADC1**

**0x4003.A000 0x4003.BFFF Reserved -**

**0x4003.C000 0x4003.CFFF Analog Comparators**

**0x4003.D000 0x4003.DFFF GPIO Port J**

**0x4003.E000 0x4003.FFFF Reserved -**

**0x4004.0000 0x4004.0FFF CAN0 Controller**

**0x4004.1000 0x4004.1FFF CAN1 Controller**

**0x4004.2000 0x4004.FFFF Reserved -**

**0x4005.0000 0x4005.0FFF USB**

**0x4005.1000 0x4005.7FFF Reserved -**

**0x4005.8000 0x4005.8FFF GPIO Port A (AHB aperture)**

**0x4005.9000 0x4005.9FFF GPIO Port B (AHB aperture)**

**0x4005.A000 0x4005.AFFF GPIO Port C (AHB aperture)**

**0x4005.B000 0x4005.BFFF GPIO Port D (AHB aperture)**

**0x4005.C000 0x4005.CFFF GPIO Port E (AHB aperture)**

**0x4005.D000 0x4005.DFFF GPIO Port F (AHB aperture)**

**0x4005.E000 0x4005.EFFF GPIO Port G (AHB aperture)**

**0x4005.F000 0x4005.FFFF GPIO Port H (AHB aperture)**

**0x4006.0000 0x4006.0FFF GPIO Port J (AHB aperture)**

**0x4006.1000 0x4006.1FFF GPIO Port K (AHB aperture)**

**0x4006.2000 0x4006.2FFF GPIO Port L (AHB aperture)**

**0x4006.3000 0x4006.3FFF GPIO Port M (AHB aperture)**

**0x4006.4000 0x4006.4FFF GPIO Port N (AHB aperture)**

**0x4006.5000 0x4006.5FFF GPIO Port P (AHB aperture)**

**0x4006.6000 0x4006.6FFF GPIO Port Q (AHB aperture)**

**0x4006.7000 0x400A.EFFF Reserved -**

**0x400A.F000 0x400A.FFFF EEPROM/ Key Locker**

**0x400B.0000 0x400B.7FFF Reserved -**

**0x400B.8000 0x400B.8FFF C 8**

**0x400B.9000 0x400B.9FFF C 9**

**0x400B.A000 0x400B.FFFF Reserved -**

**0x400C.0000 0x400C.0FFF C 4**

**0x400C.1000 0x400C.1FFF C 5**

**0x400C.2000 0x400C.2FFF C 6**

**0x400C.3000 0x400C.3FFF C 7**

**0x400C.4000 0x400C.FFFF Reserved -**

**0x400D.0000 0x400D.0FFF EPI 0**

**0x400D.1000 0x400D.FFFF Reserved -**

**0x400E.0000 0x400E.0FFF 16/32-bit Timer 6**

**0x400E.1000 0x400E.1FFF 16/32-bit Timer 7**

**0x400E.2000 0x400E.BFFF Reserved -**

**0x400E.C000 0x400E.CFFF Ethernet Controller**

**0x400E.D000 0x400F.8FFF Reserved -**

**0x400F.9000 0x400F.9FFF System Exception**

**Module**

**0x400F.A000 0x400F.BFFF Reserved -**

**0x400F.C000 0x400F.CFFF Hibernation Module**

**0x400F.D000 0x400F.DFFF Flash memory control**

**0x400F.E000 0x400F.EFFF System control**

**0x400F.F000 0x400F.FFFF µDMA**

**0x4010.0000 0x41FF.FFFF Reserved -**

**0x4200.0000 0x43FF.FFFF Bit-banded alias of 0x4000.0000 through 0x400F.FFFF -**

**0x4400.0000 0x4402.FFFF Reserved -**

**0x4403.0000 0x4403.0FFF CRC Module -**

**0x4403.1000 0x4403.1FFF Reserved [4 kB] -**

**0x4403.2000 0x4403.3FFF Reserved [8 kB] -**

**0x4403.4000 0x4403.EFFF Reserved -**

**0x4403.F000 0x4403.FFFF Reserved [4 kB] -**

**0x4404.0000 0x4404.FFFF Reserved [64 kB] -**

**0x4405.0000 0x4405.3FFF Reserved -**

**0x4405.4000 0x4405.4FFF EPHY 0 1467**

**0x4405.5000 0x5FFF.FFFF Reserved -**

**0x6000.0000 0xDFFF.FFFF EPI0 mapped peripheral and RAM -**

**Private Peripheral Bus**

**0xE000.0000 0xE000.0FFF Instrumentation Trace Macrocell (ITM)**

**0xE000.1000 0xE000.1FFF Data Watchpoint & Trace**

**0xE000.2000 0xE000.2FFF Flash Patch &Breakpoint**

**0xE000.3000 0xE000.DFFF Reserved -**

**0xE000.E000 0xE000.EFFF Cortex-M4F Peripherals**

**0xE000.F000 0xE003.FFFF Reserved -**

**0xE004.0000 0xE004.0FFF Trace Port Interface Unit**

**0xE004.1000 0xE004.1FFF Embedded Trace**

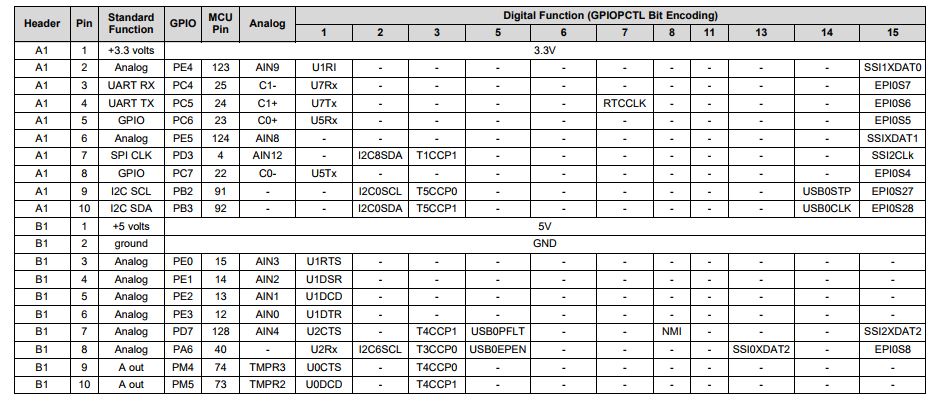
**0xE004.2000 0xFFFF.FFFF Reserved -**

Table 1.1: Memory map of the TM4C1294 memory.

The advantage of the logical partitioning is to define memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

1. **Describe the IO pins TM4C1294 Launchpad:**

**Below is the description of the IO pins**

The IO pins are divided into different Ports. These ports are then subdivided into different pins. The IO pins are physically separated on the board according to the booster pack configurations. There are 2 main booster packs on the board with different functionality. The table below shows booster pack 1. As we can see from the table some of the Pins can be also used as analog pins, power pins, interface pins and GPIO pins. There are 40 Booster pack pins, And 100 breadboard connection headers. In addition to the Analog function, there are also different configurable digital functions that can be programmed by using the GPIOCTL bit encoding. The remaining pins in the board are jumpers which are used for hardware configurations for the device.****

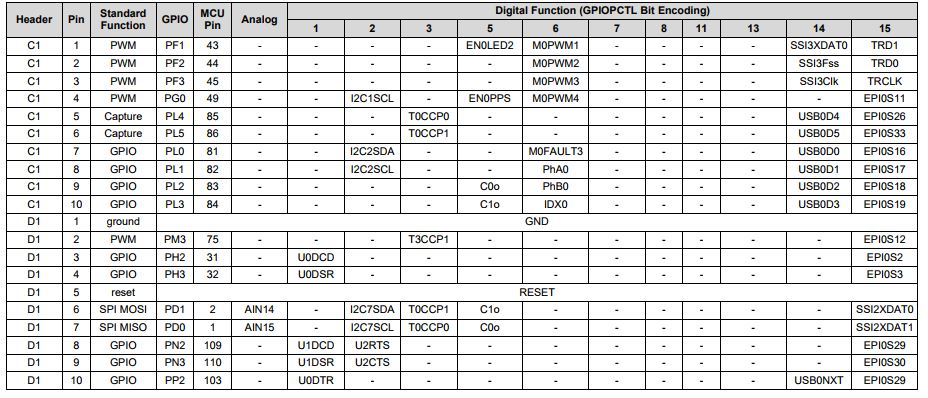
****

Table 2.2: Booster Pack 1 for TM4C1294 Launchpad

**6.2 Experiment 1**

**Describe the internal ADC in the TM4C1294 microcontroller.**

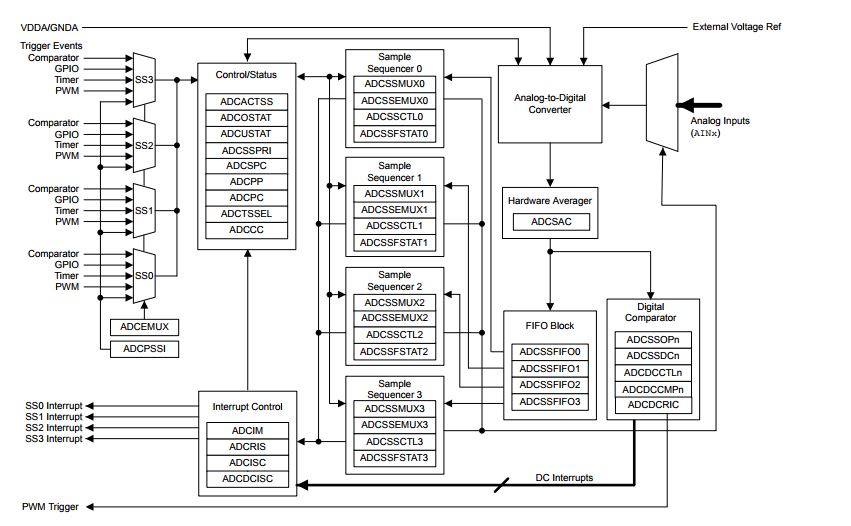


Figure 2.1: ADC Module Block Diagram

An ADC is a device that converts analog input from a channel to discrete digital signals. With the micro controller, there are two modules of ADC which share 20 input channels. It is also a 12-bit resolution which uses the Successive approximation conversion scheme.

Each module on the TM4C1294NCPDT microcontroller provides the following features:

As mentioned before, a 20-shared analog input channel with 12-bit precision. These 12 bits enables us to have 2^12 steps of the full value of the voltage. A single ended and differential input. This is configurable according to the use. Either for simplicity or ability to use floating inputs. It also has on chip internal sensors which can measure ambient temperatures from -40 to 85 degrees Celsius with

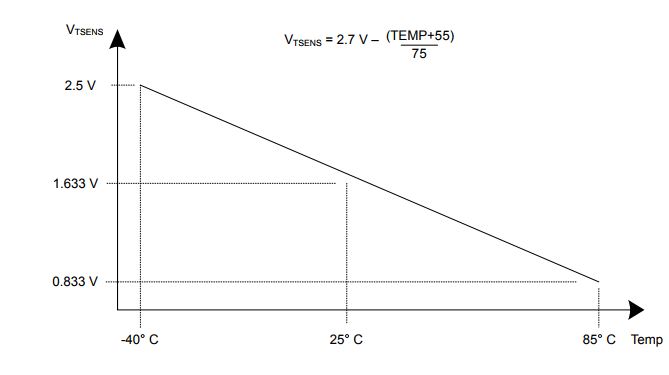


Figure 2.2: Internal Temperature Sensor Characteristic

Maximum sampling rate of 2 million samples per seconds. This is a fast conversion speed because the rate at which we get analog data can be converted to digital data is in the micro seconds scale which is really food for constantly changing signals.

Optional programmable phase delay for the 2 ADC.

Sample and hold window programmability depending on how fast the input signal is and holds the discretized value coming in for a specified amount of time before releasing it to get another sample and repeats it and then we can decide to use the sample and hold ability.

Each ADC’s has four programmable sequencers which allows multiple analog input sources. Each sequencer offers flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. [data sheet pp1053]

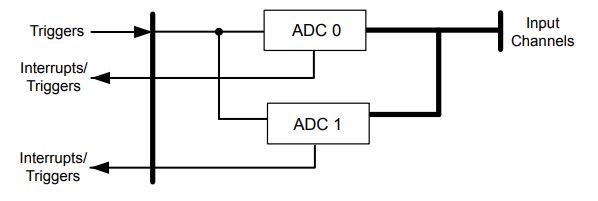


Figure 2.3: Implementation of 2 ADC blocks

The ADC also has flexible trigger control Like software control triggering which is allowing the program to begin conversion as required by the software, Timer trigger which is getting samples after specific time as programmed, Analog comparators which allow the ADC to be triggered if there is an event that occurs and there is need for the ADC to be triggered, PWM and GPIO which allows the ADC to be triggered by external peripherals and change in the pulse width through the GPIO pins and PWM signals.

The ADC also has internal Hardware Averaging for up to 64 Samples with 8 Digital comparators

The ADC module is clocked by the system clock but the conversion is run by the ADC clock ie 16MHz for 1Mega Samples per second.

The ADC uses two voltage signals, the Analog reference voltage VREFA+ and the analog ground voltage GNDA to specify the full scale of the ADC0020 [TI; 15.3.4.1].

The analog power supply and ground are analog ground are required by the ADC are separate from the power and ground of the digital circuitry. The ADC uses internal signals VREFP and VREFN as references to produce a conversion value from the selected analog input.

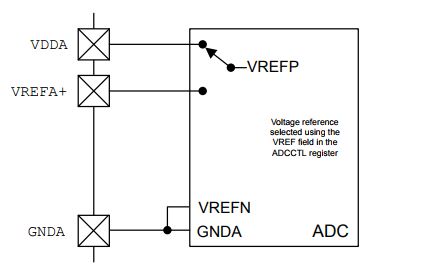


Figure 2.4: ADC Voltage Reference

**Reproduce Table 1.1 from lab 4 Manual. Comment on the last 2 Entries in the table.**

|  |  |
| --- | --- |
| **ITEM** | **Value /Name** |
| ADC VREFA+ Voltage | 3.3 Volts |
| GNDA Voltage | 0 Volts |
| What bit is modified in the GPIO Enable Register to Enable ADC Signal Routing. | These signals are configured by clearing the corresponding DEN bit in the GPIO Digital Enable (GPIODEN) register for the pin used |
| What bit is modified in the Analog Mode Select Register to Enable ADC Signal Routing. | These signals are configured by setting the corresponding AMSEL bit in the GPIO Analog Mode Select (GPIOAMSEL) register for the pin used as .analog input |

Table 2.1: Lab 4 manual table 1.1

**Summary of the APIs and the Tivaware Library**

The definition of am API is a set of well defined methods of communication between software components. These components include protocols, device drivers, and tools for building application software. The peripheral driver libraries for the TI Tivaware can access the devices that are on the TM4C1294 microcontroller, Int his lab we will use the ADC and the UART drivers for accessing the ADC and UART devices. Among other drivers, we will be accessing the GPIO ports and pins. The language the drivers are written in is C, which is also the language used in the experiments. This allows for computations at compile time rather than at run time. This can be built with a variety of tool chains. The choice for the Microcontroller Development Kit(MDK) in this experiment is Keil Real view MDK.

However, since we are using the Tivaware libraries to access the device drivers, we cannot use any other software frameworks at the same times as the configurations might be different and having multiple frameworks set different configurations might lead to errors.

The API for the ADC in the cortex gives us 40 functions for accessing and programming the ADC in the TM4C1294 launchpad. Some of the functions is setting the sampling rates, configuring the sequencers and handling interrupts. The sampling sequencers can be configured on which cannel to use and the hardware oversampling rate. All this can be done with the list of functions accessible to the user through the API.

**Describe the internal temperature sensor interfacing:**

* **Configuring the ADC**: The type of trigger we will use to trigger the ADC is software based trigger. To configure the ADC, we first enable the ADC0 peripheral. We then configure the ADC sequencer for sample sequence 0 with a processor (software) based trigger. We then configure the sequence to sample temperature sensor on step 0 of sequence 0 for the temperature sensor and we also set the hardware oversampling and averaging depending on your analog input source. This raises an interrupt when there is a sample. And then specifying the last conversion on sequence 0. The sequence is then enabled for ADC 0 and then we clear the interrupt to ensure there is no existing interrupt to interrupt the sampling before it begins. Below is the code;

void configureADC(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ADCSequenceDisable (ADC0\_BASE, 0);

ADCSequenceConfigure (ADC0\_BASE, 0, ADC\_TRIGGER\_PROCESSOR, 0);

ADCSequenceStepConfigure (ADC0\_BASE, 0, 0, ADC\_CTL\_TS | ADC\_CTL\_IE | ADC\_CTL\_END);

ADCSequenceEnable (ADC0\_BASE, 0);

ADCHardwareOversampleConfigure(ADC0\_BASE,0x4); //Hardware Oversample

ADCIntClear(ADC0\_BASE,0);

}

* **Data acquisition**: After configuring the ADC, the ADC0 is triggered to start sampling conversion. We then wait until there is a sample; The reason for this wait is because we are using a successive approximation A/D conversion the amount of time for conversion is known which is the number of bits, in our case 12-bits and outputs a sample. This takes 12 clock cycles; However, this clock is the ADC clock not the system clock. If there is a sample, we clear interrupt flag and get the sequence data and then we are done until the next trigger. Below is the code for the data acquisition.

configureADC();

uint32\_t pui32adc0val[1];

uint32\_t U32tempValC;

uint32\_t U32tempValF;

ADCProcessorTrigger(ADC0\_BASE,0);

unsigned long waitTime1 = 400;

while (!ADCIntStatus(ADC0\_BASE, 0 , false))

{

}

ADCIntClear(ADC0\_BASE, 0);

ADCSequenceDataGet(ADC0\_BASE, 0 , pui32adc0val);

* **Preparing the Data for Display**: For the data given by the ADC, we need to make sense of this result. We know the ADC measures in steps on 2number of bits and has a Full-scale range. For the temperature sensor, we prepare the data to be displayed by the formula below Where VREF N and P is the Full scale We get the Celsius value by using the formula above and the Fahrenheit we get from the formula . We get the Digital code and then get the ratio compared to the full scale and then multiply it by the Full scale of the temperature range. Below is the code for the temperature sensor.

U32tempValC = 147.5 - ((75 \* 3.3 \*(float)pui32adc0val[0])/4096);

volt = 2.7 - ((U32tempValC + 55) /75);

U32tempValF = (((U32tempValC \*9) +160 ))/5;

* **Displaying the Data**: The data is displayed to the screen is displayed using the printf function. For the temperature, we print the ambient Celsius and the Fahrenheit temperature values.

Printf ("Temperature = %3d\*C or %3d\*F \r\n", U32tempValC, U32tempValF);

* **High level Description of the Source Code**:

When we enter the main method, we initialize the serial, the ADC and the GPIO, after initialization we initialize a clock variable we use as a delay to get data from the sensor. We loop continuously while we take the value of the temperature and position. In this loop we start sampling, we get the data and then we prepare the data for display and then we display the data. This process continues forever until the reset button is pressed on the Launchpad. [See Appendix code Temp]

**6.3 Experiment 2**

**Describe the position sensor used in this Lab 4.**

The position sensor uses a linear displacement potentiometer. This is a three-terminal device which the state of the resistance depends on the position of the slider relative to one of the ends of the Potentiometer. When current is made to pass through the ends of the resistor, thee is a potential difference between the ends of the resistor. Because the resistance is changing and the current is constant the voltage across the ends of the resistor changes and is proportional to the change in the resistance.

The name of the potentiometer used for experiment 2 is PTA1543-2010CIB103 Sliding potentiometer and slides in a 15mm displacement with a carbon resistance of 10kΩ and a tolerance of ±20%. The device has three pins. Two on one of the ends and one at the other end. Pin 1 and pin 3 are both used to power the potentiometer and pin 2 is used to vary the resistance hence, vary the voltage.

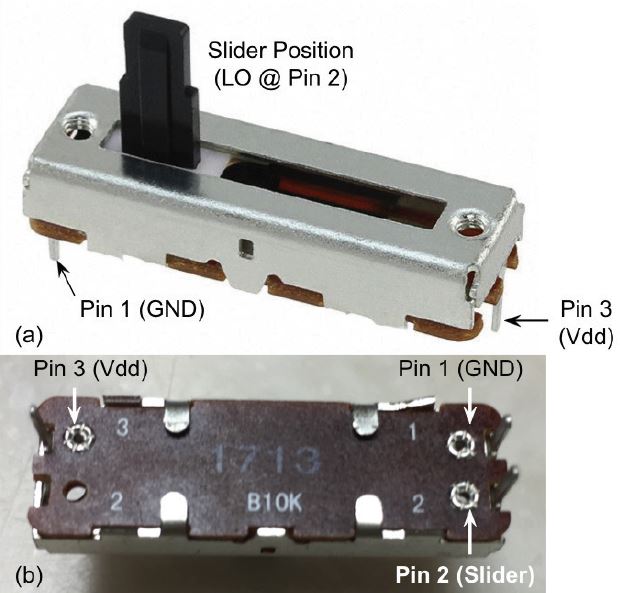


Fig 2.5: Linear Potentiometer.

**Describe the interfacing of the sensor in hardware and software.**

The interfacing of the sensor is done in the hardware and in the software.

* **Hardware:** The potentiometer is put on a bread board such that its pins are not internally connected on the board. Since the potentiometer has three pins, we define the pins to be the values on the figure above with pin 1 to ground, pin 3 to Vdd i.e. Full-scale voltage and Pin 2 from the slider to an analog voltage input pin on the microcontroller i.e. one of the 20 ADC channel pins on the launchpad GPIO.

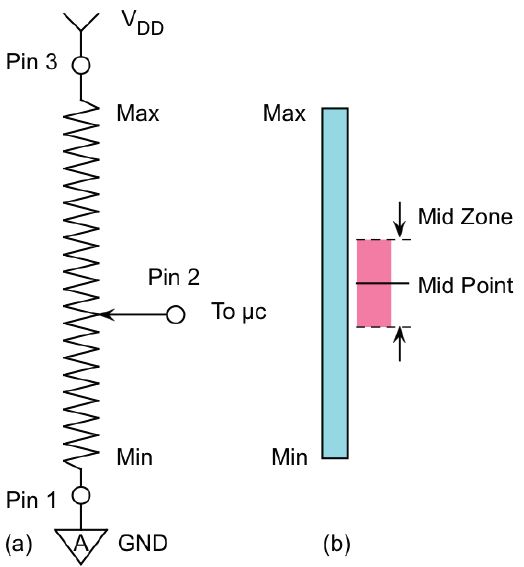


Figure 2.6: (a) Diagram of interfacing of the linear potentiometer (b) Demonstration of the midpoint. [Kins2 Sec 2.4.1]

**Software:** For the software, we enable the GPIO peripheral port to configure it. And then we set the pin type as an ADC GPIO with the specified port and the desired pin for the corresponding to the port. We then acquire the data just as we did in ADC for the temperature sensor, we only should replace ADC0\_BASE with the corresponding GPIO input. And then we prepare the value gotten to the ratio of the full scale and multiply this ratio by 15 because of the linear relationship between the voltage and the position. Then we print the position to the screen and notify when the slider is ±1/8 on the middle and exactly at the middle through the printf function. This prints to the serial output and can be seen trough an SSH client like putty.

ADCIntClear(ADC0\_BASE, 0);

startSampling();

uint32\_t val;

ADCSequenceDataGet(ADC0\_BASE, 0 , &val);

float volt;

volt = (float)val \* 15.0/4095.0;

sprintf(string ,"Position(mm) = %6.3f \r\n", volt);

printf(string);

**Answer to the question in section 2.4.3 of the lab manual**

***How fast can you move the slider?*** This is achieved by calculating how fast a user moves the slider from one end to the other. The slider from one end to the other is 15mm long **[kins1]**. Using the system clock of the micro-processor, the user can calculate the time it takes to move the slider from position 0 to position 15mm. We found after three tests that the time it takes to move the slider is **1/2seconds.** Using the distance, time, speed relationship we measured the time it takes to move the slider on average to be

***Are there any conditions the microprocessor is too slow to measure the position? Estimate the conditions***

The conditions the microprocessor is too slow to measure the position is if the changing speed i.e. the aperture time is less than the processing or conversion time. But from our code **[code appendix]**, the speed at which the slider changes its position is way less than the conversion time hence the microprocessor has enough time to measure the position.

***From the Observations of the result, is the transfer characteristics linear?***

From the graph in figure 1.9.The transfer characteristics is **linear**.

***Estimate the current flowing through the linear pot.***

***I.e. IR = \_\_ mA***

***Is the above current in the limit of the microcontroller?***

Yes, the above current is well in the limit of the microcontroller. The limit of the microcontroller is 18mA.

***What should be the voltage at the midpoint?***

The voltage at the midpoint i.e. the voltage when the slider is at 7.5mm way from the end of the potentiometer is

**Answer to the question in section 2.5.1 of the lab manual**

**Measure the actual resistance of the potentiometer. Ractual** = 9.75kΩ

**Ractual expressed as a percentage of the full scale** = 97.5%

**How accurate is it**: The resistor is really accurate because, it is within the ±20% tolerance.