Sensors, Actuators and Analogue Signals

1 Sensors and Actuators

Many embedded systems feature a mixture of sensors and actuators. Examples include:

- Actuators :
  - Aircraft control surface movement (fly-by-wire)
- Analogue temperature sensors
- Accelerometers
  - Computing => hard-drive and handheld protection
  - Automotive => stability control, airbag deployment
  - Entertainment => Nintendo Wii controller (http://www.analog.com/en/prod/0,2877,ADXL330,00.html)
- Motors, Servos
- Flow meters
- Odometers
- Microphones and speakers

It is through these devices that an embedded system interacts with the "real" world. The sensors/actuators used will depend on the application of the embedded system. The details of the operational mechanisms of these devices is beyond the scope of this module. However, there are several important concepts which need consideration when an embedded system is designed and built. Importantly both the external devices and microprocessor must be chosen so that they are compatible. Information on each component can be found in its data sheet (usually available from a suppliers or manufacturers website). These important considerations include:

- **The external devices data type**, i.e. Analogue or Digital. If the input/output is analogue, then a process and/or device needs to be considered so that an analogue input can be converted to a digital input and a digital output can be converted to a analogue output. This is covered briefly in the next section. If the input/output is digital the external device and microprocessor need to be matched so that they are capable of reading the same format of digital data.
- **How the data is transmitted**, i.e. whether the data transmission is in serial or parallel format, or whether a particular transmission format is required (e.g. a standard protocol).
- **What data transmission initialisation routine does the device support**, i.e. does it send data periodically, or is it more sporadic and the device supports mechanisms such as interrupts.
- **The voltage range of the external device**, i.e. does an analogue output from a microprocessor need to be amplified to drive an external device, or does an analogue input from an external device need to be amplified to an acceptable range to be read by a microprocessor? Similarly, is the voltage/current output from an external device too large to be safely input directly into a microprocessor?

2 Analogue ↔ Digital Conversion

This section briefly covers why A/D and D/A are important, analogue signal to digital conversion techniques (ADC) and digital signal to analogue conversion techniques (DAC and PWM). It does not cover the physical implementation of these converters. However, this topic is covered by numerous references in the library if you are interested.

Many embedded systems feature analogue devices. Analogue to Digital Converters (A/D, ADC, A2D), Digital to Analogue Converters (DAC, D/A, D2A) and Pulse Width Modulation (PWM) provide the link between an embedded system and non-digital devices.

A/D and D/A is important because it links the virtual world of computer software to the physical world. Electronic control and monitoring of devices is often better or more convenient than manual operation, for example reading sensors and data acquisition systems. Therefore electronic (digital) embedded control systems are often implemented. Note that all embedded systems are likely to be digital systems today, but peripheral devices such as sensors or actuators can be digital or analogue. The input from the physical world is mostly in a continuous form (analogue), e.g. temperature. These inputs need sampling and converting to a digital form before they can be correctly interpreted by the electronic system. The output from the electronic system is naturally in digital form and thus needs converting to the analogue domain if the embedded system controls an analogue device. However we can only approximate the analogue signal.

A familiar example is recording digital music:

1. The microphone picks up analogue sound waves and creates an electrical signal with changing current or voltage.

2. The A/D samples the analogue electrical signal and outputs a stream of digital numbers. This is the digital representation of the sound waves.

3. The stream of numbers is recorded onto the media.

CD quality sound uses 16 bits to represent each analogue sample. Therefore it has 2^{16} = 65536 possible different levels to capture an instantaneous signal. For CD quality sound, the sampling rate is 44100 Hz, thus:

- Every second 88,200 bytes are recorded
- Every minute > 5 Mbytes are recorded
- Every hour > 300 Mbytes are recorded (×2 for stereo)

2.1 A/D and D/A Conversion process outline

A/D Conversion Basics:

- Sample an analogue signal at a fixed or non-fixed sampling interval, or rate.

- Collect the sequence of digital values that represent a continuous analogue signal.

D/A Conversion Basics:

- Convert digital numbers as defined by a software process into an analogue signal to drive a circuit/device.

- The digital numbers are not analogue and do not provide a continuous stream of data.

Note: A/D and D/A conversion take a finite amount of time, leading to some time lag in the system – this can be potentially important for control systems, for example the controlled systems stability.

2.2 Analogue to Digital Conversion

An analogue input signal is usually a voltage V_{in} with range V_{fs} (fs = full scale). The result of ADC is a digital value – represented as a binary number

Several parameters determine the performance and output signal quality of ADC:

1. ADC Precision (Quantisation Levels)
    - The number of distinct inputs, e.g. 8-bit precision has 256 distinct levels.
2. ADC Range (V_{fs})
    - The minimum and maximum value of the ADC input, usually in volts, defines the range, i.e. $V_{fs} = V_{max} - V_{min}$
3. ADC Resolution
    - The change in input which causes the digital output binary value to change by 1.
    - ADC Resolution (volts) = ADC Input Range (V_{fs}) / ADC Precision
4. ADC Linearity
    - ADC's are termed linear if the resolution is constant throughout the whole input range and non-linear otherwise.
5. ADC Bandwidth
    - The frequency range that the analogue input can be received with minimal amplitude attenuation.
6. ADC Conversion time
    - The time taken to calculate the digital output for a particular analogue input.

Analogue to Digital conversion is not infinitely quick, for example, it takes a minimum of 0.5 microseconds for A/D onboard the STM32F407 in 12-bit standard mode. **ADC Speed** is the time taken to convert the signal into a binary number. The speed depends on the precision and the electronic implementation. For example, for 6-bit precision the minimum conversion time falls to 0.3 microseconds for the STM32F407.

The digital output from ADC's is usually encoded in one of four schemes:

1. Straight binary
2. Complementary binary (complement of each bit : "~" operator)
3. 2's Complement binary
4. Offset binary (complement most significant bit of 2's complement)

There are many methods that A/Ds can be implemented electronically, but this is beyond the scope of this course. However, it is useful to understand some ideas common to A/D conversion:

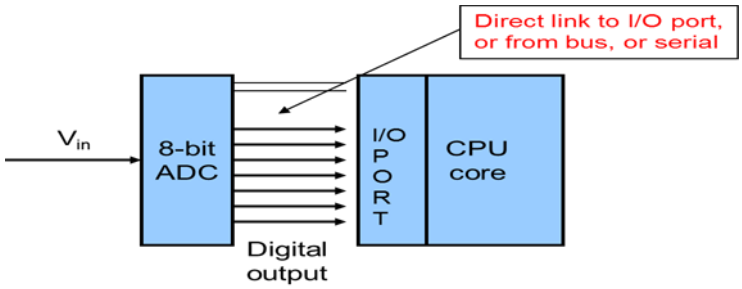
- **Ramp ADCs**: Using a DAC, this ADC builds up (ramps) an output voltage from the DAC until DAC output is just > the ADC's V_{in}. Thus, we now have a digital value from the DAC input for the ADC output.

- **Successive Approximation ADCs**: If V_{out} is the output from a DAC, a successive approximation determines the digital value of V_{in} by successively testing the bits input into the DAC (stored in a register, SAR).

  - If setting a bit makes $V_{out} > V_{in}$, the bit is cleared
  - If setting a bit makes $V_{out} < V_{in}$, the bit remains set

  Therefore, once all of the DAC input bits have been tested, we have a digital value of V_{in}.

The ADC usually interfaces with a microprocessor through a direct link with an I/O port, which often uses parallel data lines, the data bus or a serial I/O port:



The process of reading an ADC:

- The software sets the ADC START register (or bit within a control register).
- The software waits (polling/interrupt driven) until DONE is signalled from the ADC.
- The software reads the result from the ADC data register.

2.2.1 Digital to Analogue Conversion

The operation of outputting analogue signals from an embedded system is executed by a Digital to Analogue Converter (DAC), or alternatively by PWM (see section 2.3). DACs are ubiquitous in audio and video equipment, e.g. Digital music files are heard through analogue speakers, via a DAC. The DAC input is a stream of digital values, usually a binary representation of a variable in software. The DAC output can be voltage or current and is piecewise linear, i.e. the output voltages are held until a new input is processed. For simplicity, we will only consider linear & monotonic DACs.

The parameters which determine the DAC performance and output signal quality are:

1. DAC Precision
   - The number of distinct output values. This usually depends on the range of values available from the digital input, e.g. 8-bit, 10-bit, 12-bit.
2. DAC Range
   - The minimum and maximum value of the DAC output, volts or amps, defines the range.
3. DAC Resolution
   - The smallest distinct change in output level from the DAC (volts/amps).

     DAC Resolution = DAC Range / DAC Precision

     e.g. If a DAC has an output range of 0-5 V and precision 8-bit

     Resolution = 5 / 256 = 0.0195 V
4. DAC Accuracy
   - The difference between the desired and actual output

     Accuracy (%) = 100 x (actual output – ideal output) / ideal output
5. DAC Settling time
   - The time taken for the output to reach its desired value following a change in output.
6. DAC Speed
   - The time taken for a complete conversion
   - Determines the maximum output rate.
7. DAC Linearity
   - A DAC is linear if the output changes in equal steps across the full range of input values.

The encoding scheme determines how the DAC uses a binary number to output a voltage. Two common 8-bit encoding schemes, one for an unsigned DAC output, another for a signed output are shown below:

V_{out} (unsigned) = V_{fs} (b7/2 + b6/4 + b5/8 … b0/256) + V_{offset}

V_{out} (signed) = V_{fs} (**-b7/2** + b6/4 + b5/8 … b0/256) + V_{offset}

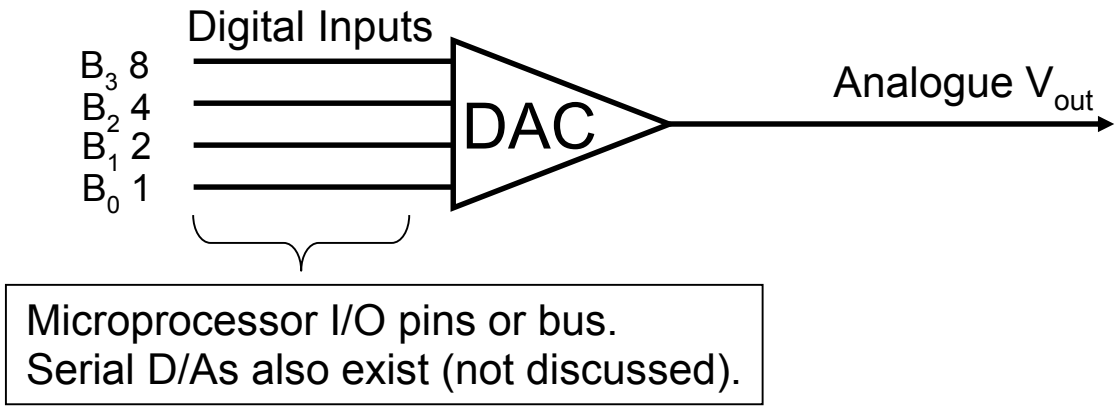
e.g. If a binary input to an 8-bit DAC (0...3V) = 01101101

V_{out} (unsigned) = 3(0/2 + 1/4 +1/8 + 0/16 + 1/32 + 1/64 + 0/128 + 1/256) + V_{offset}

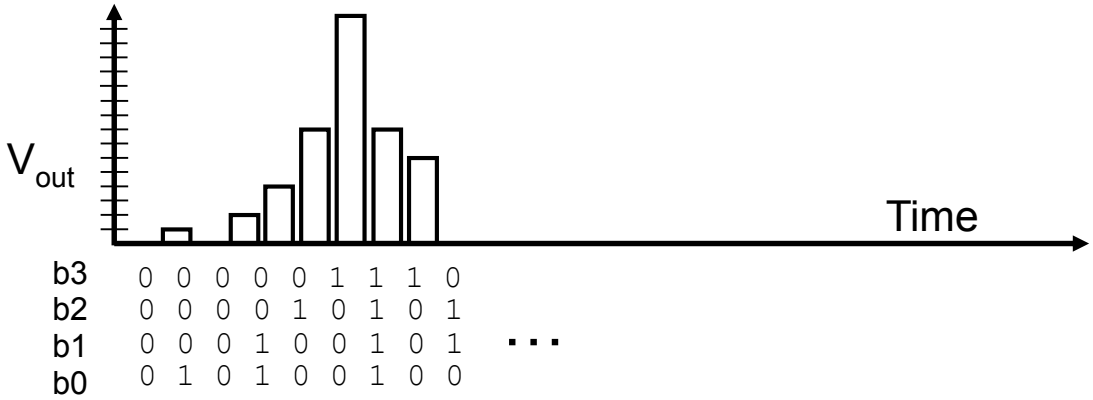
V_{out} (unsigned) = 3(0.426) + 0 = 1.277 volts

The input to D/As are digital numbers and we see D/As in diagrams as follows:

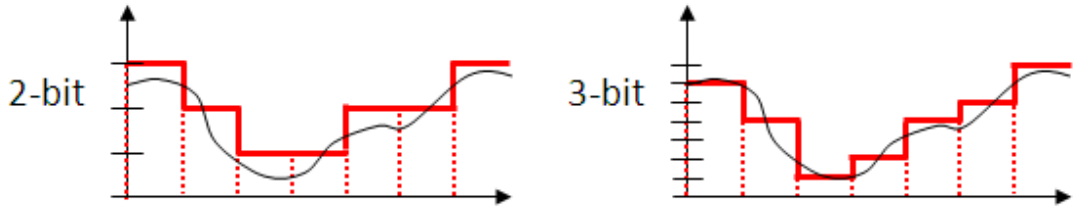
4-bit DAC:



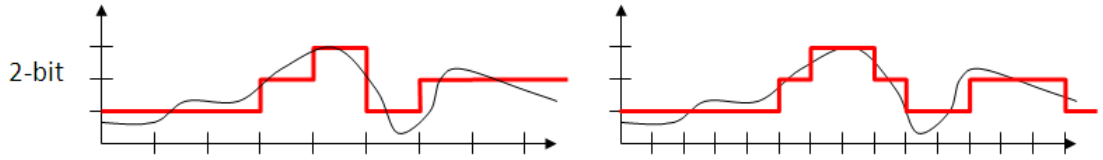
The DAC I/O variation could be:



On the subject of DAC quality, the precision, range and resolution have already been introduced. Generally, more bits provide a greater precision and thus a finer approximation of a desired analogue output. This is illustrated below:



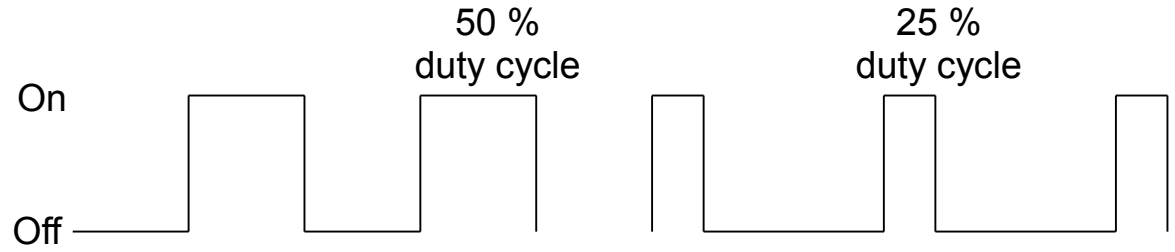
Other factors include DAC speed (output settling time and output rate). If a DAC can't process input values at the rate they arrive, then errors in the system are introduced. In the simple example below, the calculation of the output (red) in the figure on the right is faster.



2.3 D/A – Pulse Width Modulation

Many microcontrollers have a PWM facility, which provides a way of using a microcontroller's digital output to control analogue circuits. Applications of PWM include communications and power control for devices (motors, light sources, servos).

PWM controllers output a square wave with an associated **duty cycle** and **modulation frequency**:



Duty cycle: The proportion of the output signal that is high (asserted).

Modulation frequency: The period of a cycle. Modulation frequency depends on the response time of a device. The cycle period must be short with respect to the response time to avoid twitchy behaviour of a controlled device.

By varying the duty cycle, an analogue output signal can be encoded, for example:

Small servos use PWM with a modulation frequency of about 50Hz (20ms). A 1.5 ms pulse centres a servo (duty cycle ~ 7.5%). 1ms and 2ms pulses are the usual limits for full angular control of the servo.

2.3.1 Practical issues with A/D and D/A conversion

Many microcontrollers have ADC, DAC and PWM built-in, such as the STM32F407. However it is common to see microcontrollers with ADC, but no DAC facilities. To create an analogue signal with no on-chip DAC a free general purpose I/O port can be used for the digital output. This I/O port is directly connected to a separate DAC (on-board), or via a bus.

Many on-chip ADCs provide special registers to control the ADC and store results, for example:

- ADC_CR1 and ADC_CR2 registers configure ADC operation on the STM32F407
- ADC_DR register stores the result of the ADC conversion on the STM32F407

ADC can be configured so that completion of a conversion can raise an interrupt rather than polling a status register.

You can find out more information about the STM32F407, including the ADC, DAC and PWM functions be consulting the relevant sections in the following two documents.

STM32F - Processor Reference manual

STM32F - Datasheet