An analog signal with bandwidth $f_s$ must be sampled at $f_s > 2f_s$. Perfect measurements taken at $f_s > 2f_s$ result in no information loss due to sampling. Due to actual (quantized) measurements, oversampling is required.

The signal $f$ with respect to the thermal noise is $\frac{f}{\Delta f} = 2 \times 10^{-2}$ and $\frac{\Delta f}{\Delta f} = 2 \times 10^{-5}$.

The signal-to-noise ratio in decibels $\text{SNR}(\text{dB}) = 10 \log \left( \frac{f}{\Delta f} \right)$.

The sampling frequency $f_s$ must be sampled at $f_s > 2f_s$. Perfect measurements taken at $f_s > 2f_s$ result in no information loss due to sampling. Due to actual (quantized) measurements, oversampling is required.

Integrating A/D converters (Single Slope)

Integrate a reference voltage $U_{ref}$ until it matches the input voltage $A_{in}$.

- Sampling frequency depends on input signal.
- Accuracy depends on $U_{ref}$ integrator and clock.
- Simple components.
- Slow (typically $\approx 100\mu$s).

Integrating A/D converters (Dual Slope)

Input voltage $A_{in}$ is integrated for a constant time. The integrator is then discharged by a constant reference voltage $U_{ref}$. The discharge time is measured and is proportional to $A_{in}$.

- Can smooth the input signal, and suppress specific frequencies.

Flash A/D converters

$2^N - 1$ concurrent comparators identify the signal in one step.

- Fastest converter technology: Single-step conversion, minimal latency.
- Complex for higher resolutions; requires comparators with $2^{21} - 1$.
- Accuracy depends on the accuracy of the reference voltages.

Digital output

Perfect analog input no information loss due to sampling. Due to actual (quantized) measurements: oversampling is required.

Actual A/D converters are also characterized by:

- Integral Non-Linearity (INL): Maximal difference between the actual and ideal code centers.
- Differential Non-Linearity (DNL): Differences between transition code widths.
- Missing codes: reduce SNR by 20dB or 6dB/dec for each missing code.
- Response time: Latency: Throughput: Maximal sampling rate.

Accuracy (ENOB, SNR)

Latency (time from sensing to delivery).

Power consumption.

Complexity (also affects: price).

Analog input

Digital output

The signal to noise ratio in decibels

$\text{SNR}(\text{dB}) = 10 \log \left( \frac{f}{\Delta f} \right)$.

Actual A/D converters are also characterized by:

- Integral Non-Linearity (INL): Maximal difference between the actual and ideal code centers.
- Differential Non-Linearity (DNL): Differences between transition code widths.
- Missing codes: reduce SNR by 20dB or 6dB/dec for each missing code.
- Response time: Latency: Throughput: Maximal sampling rate.
Successive Approximation Register (SAR) converters
Single bit A/D converter converts one bit at a time, starting with the most significant bit, e.g. comparing to \(v_{LSB} = \frac{V_{ref}}{2^n} \) of full scale.
- **Minimal cycle time** - almost independent of resolution.
- **Typically slow**.
- **Accuracy** depends on the accuracy of the single bit ADC and the DAC.
- Typical applications: typically slow, budget applications - yet also used in high-accuracy applications.

Tracking Register (TR) A/D converters
Continuous single bit conversion compares the current digital output with the analog input and counts a register up/down accordingly.
- **Minimal cycle time** (not SAR) - almost independent of resolution.
- **Speed** depends on amplitude changes in the input signal or no constant sampling frequency.
- **Accuracy** depends on the accuracy of the single bit ADC and the DAC.
- **Typically slow**, e.g. tracking slow changing signals at high frequency (hundreds of MHz).

Σ-Δ A/D converters

The sampling frequency of the bitstream with respect to the digital output frequency (oversampling) determines accuracy.
- **Latency** depends on bitstream frequency.
- **Accuracy** depends on number of bits decimated.
- **Method** is inherently non-linear.
- **Typical use**: high accuracy, high-speed, high-resolution throughput (2.5 - 20 GSPS, 12-bit, 2.5 GSPS).

Σ-Δ D/A converters

Digital form converts to bitstream or to analog. Typical applications: audio bitstream.
- **Latency** depends on bitstream frequency.
- **Accuracy** depends on number of bits decimated.
- **Method** is inherently non-linear.

Higher order Σ-Δ A/D converters

Effective Number of Bits (Signal to Noise Ratio) can be improved by adding further integration stages.
- **Latency** depends on bitstream frequency.
- **Accuracy** depends on number of bits decimated.

ADC 08200 - Basic specifications
- **Resolution**: 24 bits
- **Sampling frequency**: 10 - 200 MHz
- **Differential Non-Linearity (DNL)**: \( \pm 0.4 \) LSB (typical), \( \pm 0.95 \) LSB (max.)
- **ENOB**: 11.5 - 15.5 (typical), 10.5 - 14.5 (max.)
- **SNR**: 57.5 - 59.5 (typical), 52.5 - 55.5 (max.)
- **No missing codes**
- **Power consumption**: 1.65W@5V, 1.6W (max.)
- **Latency**: 6 cycles (pipeline delay)
- **Aperture (sampling delay)**: 26 ns, with 2 ps rms jitter
Interfaces

LM12L458 – Register Bank

Instruction RAM entries consist of:

- 16-bit data: indicates the instruction or operation code.
- Type: specifies the type of operation.
- D15-D0: define the data values for the instruction.

Example:

<table>
<thead>
<tr>
<th>Address</th>
<th>Purpose</th>
<th>Type</th>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Instruction RAM R/W</td>
<td>Acquisition Watchdog</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Instruction RAM entries consist of (cont.):

- Address 81: records the actual number of instructions in the register bank.

Example:

<table>
<thead>
<tr>
<th>Address</th>
<th>Decoder Value</th>
<th>Instruction RAM R/W</th>
<th>Aquisition Time</th>
<th>Resolution</th>
<th>Sync</th>
<th>Pause</th>
<th>Vplus</th>
<th>Vminus</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>Instruction RAM R/W</td>
<td>Aquisition Time</td>
<td>Resolution</td>
<td>Sync</td>
<td>Pause</td>
<td>Vplus</td>
<td>Vminus</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>Instruction RAM R/W</td>
<td>Aquisition Time</td>
<td>Resolution</td>
<td>Sync</td>
<td>Pause</td>
<td>Vplus</td>
<td>Vminus</td>
</tr>
</tbody>
</table>

Data structures in 'C':

- Channels: (Start to end).
- Resolution: 32-word conversion.
- Acquisition Time: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.

Special applications data logging process control, power supply.

Data structures in 'C':

- Channels: (Start to end).
- Resolution: 32-word conversion.
- Acquisition Time: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.

Special applications data logging process control, power supply.

Data structures in 'C':

- Channels: (Start to end).
- Resolution: 32-word conversion.
- Acquisition Time: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.
- Vplus: Programmable acquisition times, sequences and conversion rates.

Special applications data logging process control, power supply.
**Interfaces**

### Control Register 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
<tr>
<td>1</td>
<td>ALIGN</td>
<td>Right/Center alignment of the data (read &amp; write)</td>
</tr>
<tr>
<td>2</td>
<td>EXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>3</td>
<td>JEXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>4</td>
<td>SWSTART</td>
<td>Start conversion of regular channels (read &amp; write)</td>
</tr>
<tr>
<td>5</td>
<td>JSTART</td>
<td>Start conversion of injected channels (read &amp; write)</td>
</tr>
<tr>
<td>6</td>
<td>ADON</td>
<td>A/D Converter ON / OFF (read &amp; write)</td>
</tr>
<tr>
<td>7</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
</tbody>
</table>

**Micro-controller definition**

- **Computer system on a chip**
  - CPU: instruction set, registers, memory, I/O ports, interrupts
  - Memory: main memory, cache, ROM, RAM
  - Peripheral interfaces: serial, parallel, I/O, DMA, etc.
  - Clock generation: internal and external clock sources
  - Power management: sleep mode, power-down, etc.
  - Security: encryption, authentication

### Interface Diagrams

- **A/D, D/A & Interfaces Examples: STM32F4**
  - Control Register 2
  - | Bit | Name        | Description                                      |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
<tr>
<td>1</td>
<td>ALIGN</td>
<td>Right/Center alignment of the data (read &amp; write)</td>
</tr>
<tr>
<td>2</td>
<td>EXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>3</td>
<td>JEXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>4</td>
<td>SWSTART</td>
<td>Start conversion of regular channels (read &amp; write)</td>
</tr>
<tr>
<td>5</td>
<td>JSTART</td>
<td>Start conversion of injected channels (read &amp; write)</td>
</tr>
<tr>
<td>6</td>
<td>ADON</td>
<td>A/D Converter ON / OFF (read &amp; write)</td>
</tr>
<tr>
<td>7</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
</tbody>
</table>

- **Micro-controller Examples**
  - MC68HC05
    - Clock: 3.36 MHz internal, 640 kHz external
    - RAM: 2 Kbytes
    - I/O Ports: 48
    - Port B
      - Polling:-ready status
      - Method: high, active
    - Port D
      - Select: tri-state, active
      - Method: low
    - Port E
      - Select: tri-state, active
      - Method: high, active
  - AVR32
    - CPU: 32-bit RISC architecture
    - registers: program counter, stack pointer, general registers
    - Memory: RAM, ROM, I/O ports
    - Interface: serial, parallel, UART, SPI
    - Interrupts: 24 interrupts
    - Power: 2.7-3.6 V
    - Package: 32-pin DIP, 32-pin SOIC

**A/D, D/A & Interfaces Examples: STM32F4**

- **Control Register 2**
  - | Bit | Name        | Description                                      |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
<tr>
<td>1</td>
<td>ALIGN</td>
<td>Right/Center alignment of the data (read &amp; write)</td>
</tr>
<tr>
<td>2</td>
<td>EXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>3</td>
<td>JEXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>4</td>
<td>SWSTART</td>
<td>Start conversion of regular channels (read &amp; write)</td>
</tr>
<tr>
<td>5</td>
<td>JSTART</td>
<td>Start conversion of injected channels (read &amp; write)</td>
</tr>
<tr>
<td>6</td>
<td>ADON</td>
<td>A/D Converter ON / OFF (read &amp; write)</td>
</tr>
<tr>
<td>7</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
</tbody>
</table>

**Micro-controller Examples**

- **MC68HC05**
  - Clock: 3.36 MHz internal, 640 kHz external
  - RAM: 2 Kbytes
  - I/O Ports: 48
  - Port B
    - Polling:-ready status
    - Method: high, active
  - Port D
    - Select: tri-state, active
    - Method: low
  - Port E
    - Select: tri-state, active
    - Method: high, active
  - **AVR32**
    - CPU: 32-bit RISC architecture
    - registers: program counter, stack pointer, general registers
    - Memory: RAM, ROM, I/O ports
    - Interface: serial, parallel, UART, SPI
    - Interrupts: 24 interrupts
    - Power: 2.7-3.6 V
    - Package: 32-pin DIP, 32-pin SOIC

**A/D, D/A & Interfaces Examples: STM32F4**

- **Control Register 2**
  - | Bit | Name        | Description                                      |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
<tr>
<td>1</td>
<td>ALIGN</td>
<td>Right/Center alignment of the data (read &amp; write)</td>
</tr>
<tr>
<td>2</td>
<td>EXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>3</td>
<td>JEXTEN</td>
<td>External event select for regular group (read &amp; write)</td>
</tr>
<tr>
<td>4</td>
<td>SWSTART</td>
<td>Start conversion of regular channels (read &amp; write)</td>
</tr>
<tr>
<td>5</td>
<td>JSTART</td>
<td>Start conversion of injected channels (read &amp; write)</td>
</tr>
<tr>
<td>6</td>
<td>ADON</td>
<td>A/D Converter ON / OFF (read &amp; write)</td>
</tr>
<tr>
<td>7</td>
<td>CONT</td>
<td>Single/Dual conversion selection (read &amp; write)</td>
</tr>
</tbody>
</table>
**Interfaces**

**Micro-controller examples**

**Composing the µengine:**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Determining actual TPU latencies**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Determining maximal TPU latencies**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**MPC565: Time-base & Real-time clock**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Interface architectures**

**Basic sampling control mechanisms**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

---

**Composing the µengine:**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Determining actual TPU latencies**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Determining maximal TPU latencies**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**MPC565: Time-base & Real-time clock**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Interface architectures**

**Basic sampling control mechanisms**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

---

**Composing the µengine:**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Determining actual TPU latencies**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Determining maximal TPU latencies**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**MPC565: Time-base & Real-time clock**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**

**Interface architectures**

**Basic sampling control mechanisms**

- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
- **Non-interruptible µ-code-blocks.**
Handling device responses

Responses from devices can be:

- Immediate.
- With a constant delay or within a defined time-frame.
- Unpredictable / sporadic.

Device handlers may thus:

- Perform a 'busy-wait' for the response.
- Reschedule the device-process by a constant delay.
- Schedule the device-process periodically and employ different time-slots for sending control / data and receiving status / data.
- React to triggers / calls / interrupts from the device.

The device handler can be implemented as a process / interrupt routine / dedicated µ-controller / DMA-controller or a mixture of those.

How to embed the unpredictable in predictable systems?

By providing the resources to cope with the assumed worst case and fall back to a lower, yet safe functionality beyond that.

Concrete:

- Either the unpredictable events need to be synchronized with the remaining real-time tasks without violating real-time constraints.
- Or exclusive processing resources (e.g., a dedicated microcontroller) for a specific device need to be provided.

Language requirements for interfaces

- Specify the device interface (protocol and formats) in all detail (candidates: Ada, CHILL, ERLANG, Modula-2, C, …).
- Handling asynchronous hardware messages (devices, timers, …).
- Many different methods to implement a context-switch (candidates: all languages with some real-time orientation: PEARL, CHILL, ERLANG, Ada, RT-Java, POSIX, …).

The term “high-level languages” in the real-time interface context allow for strong abstractions while being time and physics specific down to the actual level of interface realities.

Summary

- Analogue signal chain in a digital system
- Sampling rate, aliasing, Nyquist's criterion, oversampling
- Quantization (LSB, error voltage, THD, ENOB, SNR, DNL, INL)
- A/D converters
- Integrating, Switched-Source: A/D, Flash, Pipeline, SAR, Tracking, 1,2,3,4-bit, D/A, with or without A/D.
- Examples:
  - Fast and simple A/D example: National Semiconductor ADC08200
  - Multi-channel A/D data logging interface example: National Semiconductor LM12L458
  - Simple 8-bit µ-controller example: Motorola MC68HC05, Propeller.
  - Complex 32-bit µ-controller examples: AVR32 and Motorola MPC565 (including TPUs).
- General device handling / sampling control / language requirements