Signal chain

Original signal
(Weakish, noise- and interference-affected, carries additional higher frequency signals)

Amplified signal
(Stronger, noise- and interference-affected, carries additional higher frequency signals)

Signal after sample-and-hold circuit
(Samples are taken over short time-span and held until the next sample time)

Discrete values inside CPU memory (after A/D conversion)
(Discrete, quantized representation)

Filtered (“low passed”) signal
(Higher frequency signals have been eliminated – essential precondition for next stage)

Discrete signal at CPU output gate
(Discrete, quantized representation presented on digital output interface)
A/D, D/A & Interfaces

Sampling

- Sample data with frequency \( f_s \)
- Interpolation suggests a source signal
- The phenomenon of the wrongly observed signal at a lower frequency \( f_s \) is called aliasing.

Signal chain

- Analogue signal after D/A conversion (Limited bandwidth leads to glitches in the analogue signal)
- Smoothed ("deglitched") signal (Synchronized filter leads to predictable, analogue transition steps)
- Filtered ("low-passed") signal (Signals introduced by the conversion process are eliminated)

Nyquist's Criterion

- An analog signal with bandwidth \( f_a \) must be sampled at:
  \[ f_s = 2f_a \]
- Perfect measurements taken at \( f_s > 2f_a \) result in no information loss due to sampling.
- Due to actual (quantized) measurements: oversampling is required.
A resolution of $N$ bits provides $2^N$ possible discrete output levels:

- Smallest distinguishable value $q$ (Least Significant Bit or LSB):
  \[ q = \frac{1}{2^N} \]
- Ratio $\frac{q}{q}$ expressed in decibel (dB):
  \[ 10 \log 2^N = \frac{N \cdot 20 \log 2}{N} = 6.02 \text{dB} \]

Decibels (dB) are a ratio of powers defined as:

\[ 10 \log \left( \frac{P_1}{P_2} \right) \]

where $P_1$ and $P_2$ are the powers of two signals.

The mean square error over one step:

\[ E = \frac{1}{2} q^2 d^2 = \frac{q^2}{12} \]

Root mean square (rms) noise voltage:

\[ V_{\text{rms}} = \frac{q}{\sqrt{12}} \]

The signal $S$ with respect to the rms noise:

\[ \frac{S}{\sqrt{\text{noise}}} = 10 \log \left( \frac{S}{V_{\text{rms}}} \right) \]

or as the signal to noise ratio in decibel:

\[ \text{SNR} = 20 \log \left( \frac{S}{V_{\text{rms}}} \right) \]

Quantization

Assuming an ideal input signal:

\[ F(t) = A \sin t \quad \text{with} \quad q = \frac{2A}{2^N} \quad \text{and} \quad E = \frac{A^2}{2} \]

then the signal to noise ratio is:

\[ \text{SNR [dB]} = 10 \log \left( \frac{E}{P_{\text{noise}}} \right) = 10 \log \left( \frac{1}{2^{2N}} \right) \]

\[ \text{SNR [dB]} = 20 \log 2 + 10 \log \left( \frac{1}{2} \right) \]

\[ \text{SNR [dB]} = N \cdot 6.02 + 1.76 \]

Determining the effective number of bits (ENOB):

\[ \text{SNR}_{\text{ideal}} [\text{dB}] = 20 \log 2 + 10 \log \left( \frac{1}{2} \right) \]

\[ \text{SNR}_{\text{ideal}} [\text{dB}] = 20 \log 2 + 10 \log \left( \frac{1}{2} \right) \]

\[ \text{SNR}_{\text{ideal}} [\text{dB}] = 6.02 \]

\[ \text{SNR}_{\text{ideal}} [\text{dB}] = N \quad \text{for} \quad \text{SNR}_{\text{actual}} = \text{SNR}_{\text{ideal}} \]

A/D converters

Some criteria to select the fitting A/D converter for an application:

- Throughput (maximal sampling frequency).
- Accuracy (ENOB, SNR).
- Latency (time from sensing to delivery).
- Power consumption.
- Complexity (also affects: price).

Trade-offs are to be expected:

- Maximizing throughput will reduce accuracy and increase power consumption.
- Maximizing accuracy will reduce throughput and increase latency (… other trade-offs).

Integrating A/D converters (Single Slope)

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

- Sampling frequency depends on input signal.
- Accuracy depends on $U_{\text{ref}}$ integrator and clock.
- Simple components.
- Slow (typically ~100Hz).
**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 resolution: Required circuitry scales with $2^N - 1$.
- Accuracy depends on the accuracy of the reference voltages.

**Pipelined A/D converters**

$p$ pipeline stages with $m$ bits each provide a $pm$ bits converter. Each stage subtracts the analog value which has been converted (provides the rest as a residue value $A_{RES}$) and accumulates the digital output.

- Keeps the throughput (almost): All pipeline stages operate concurrently.
- Traded resolution for latency.
- Accuracy depends on components.

**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 $\frac{V_{REF}}{2^1}$, $\frac{V_{REF}}{2^2}$, ..., $\frac{V_{REF}}{2^N}$ of full scale.

- Minimal circuitry – (almost) independent of resolution.
- Typically slow.
- Accuracy depends on the accuracy of the single bit ADC and the DAC.

**Tracking Register (TR) A/D converters**
**Interfaces**

**A/D, D/A & Interfaces**

**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 circuitry** (no S/H) – 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.

\[ U_{in} = U_{ref} \]

\[ \sum \]

\[ A/D \text{ converters} \]

\[ \Delta \]

\[ D/A \text{ converters} \]

\[ \Sigma \]

\[ A/D, D/A & Interfaces \]

\[ \Sigma - \Delta A/D converters \]

\[ + U_{ref} \text{ or } - U_{ref} \text{ is subtracted from the input signal and integrated.} \]

The high frequent comparison of the integrator against ground results in a bitstream signal (which is also fed back).

The density of '1's in the bitstream represents the input signal.

The bitstream can be deployed as such or be translated into digital words of varying lengths.

\[ \Sigma - \Delta D/D converters \]

**Digital form converts to bitstream or to analog.**

**Notes**

- Typical application: High accuracy typically 16-24 bits, moderate throughput (24 bit high quality audio converters are usually of this type).

**A/D converters**

**Higher order \( \Sigma - \Delta \) A/D converters**

Effective Number Of Bits (Signal to Noise Ratio) can be improved by adding further integrator stages.

\[ \Sigma - \Delta \]

\[ A/D, D/A & Interfaces \]

\[ \Sigma - \Delta \text{ converters are not subject to aliasing, as they implicitly implement a low pass filter via the integrators.} \]

\[ A/D converters matrix \]

**Integration**

<table>
<thead>
<tr>
<th>Throughput</th>
<th>( \Sigma - \Delta )</th>
<th>( \Sigma - \Delta \text{ latency vs. accuracy} )</th>
<th>Pipeline</th>
<th>Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 MHz</td>
<td>( O_1 )</td>
<td>( O_1 )</td>
<td>( O_1 )</td>
<td>( O_1 )</td>
</tr>
<tr>
<td>45 MHz</td>
<td>( O_2 )</td>
<td>( O_2 )</td>
<td>( O_2 )</td>
<td>( O_2 )</td>
</tr>
</tbody>
</table>

**Resolution**

<table>
<thead>
<tr>
<th>1-bit DAC</th>
<th>0.5 to 0.3</th>
<th>0.5 to 0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-bit DAC</td>
<td>0.3 to 0.2</td>
<td>0.3 to 0.2</td>
</tr>
</tbody>
</table>

**Notes**

- Can suppress some frequencies and can be very accurate.
- Flexible architecture, very accurate.
- Compromise between speed and cost.
- Fastest converter.
**ADC 08200 – Basic specifications**

- **Resolution:** 8 bit
- **Sampling frequency:** 10 - 200 (230) MHz
- **Differential Non-Linearity (DNL):** ±0.4 LSB (typical), ±0.95 LSB (max.)
- **Power consumption:** 15 mW; 6 W (power down, clocked stopped).
- **Programmable acquisition times, sequences and conversion rates.**
- **32 word conversion FIFO buffer.**
- **Self-calibration and diagnostic mode.**
- **8 or 16 bit wide data bus.**

Typical applications: Data logging, process control, low-power devices.

**Instruction RAM entries consist of:**
- **Sync (1 bit):** indicates the last instruction and branches to the first one.
- **Power (1 bit):** halts the sequencer before this instruction.
- **Yin, Yin→ (2 - 3 bits):** select input channels (‘000’ selects ground in Yin→).
- **S (1 bit):** wait for an external sync signal before this instruction.
- **Timer (1 bit):** wait for an external tick or counter delay before this instruction.

**ADC 08200 – Non-Linearities**

- **Integral Non-Linearity:**
  - Linear Code:
  - Output Code:
- **Differential Non-Linearity:**
  - Linear Code:
  - Output Code:

**LM12L458 – Basic specifications**

- **Channels:** 8 (multiplexed).
- **Resolution:** 8 bit + sign or 12 bit + sign (SAR converter).
- **Sampling frequency:** max. 106 kHz.
- **Power consumption:** 15 mW; 6 W (power down, clocked stopped).
- **Programmable acquisition times, sequences and conversion rates.**
- **32 word conversion FIFO buffer.**
- **Self-calibration and diagnostic mode.**
- **8 or 16 bit wide data bus.**

**LM12L458 – Register bank**

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>Initial State Register</td>
<td>16</td>
<td>Set initial state</td>
</tr>
<tr>
<td>000001</td>
<td>Internal Status Register</td>
<td>16</td>
<td>Set number of internal status registers</td>
</tr>
<tr>
<td>000010</td>
<td>Value Register</td>
<td>16</td>
<td>Set value</td>
</tr>
<tr>
<td>000011</td>
<td>Command Register</td>
<td>16</td>
<td>Set command</td>
</tr>
</tbody>
</table>

**LM12L458 – Instructions**

Instruction RAM entries consist of:
Instruction RAM entries consist of (cont.):

- A Timer (12 bit) selects the resolution (8 bit + sign or 12 bit + sign).
- A Watchdog (8 bit) activates comparisons with two programmed limits.

Acquisition time: (a) The converter takes 9 × 20 cycles (12 bit mode) or 2 × 20 cycles (12 bit mode) to sample the input.合理的 times depend on the input resistance and clock frequency. (b) 0 → 4 × 6 (4 bit) to 12 bit conversions.

Limits: (2) ‘R’s, including sign and comparator: trigger levels for watchdog operation.

0 0 0 Instruction RAM R/W

ADC_Instructions (0) := (EndOfLoop      => False,
Pause          => False,
Vplus          => Ch1,                         Vminus         => Ch2,                         Sync           => False,
Resolution     => EightBit;                     Watchdog       => False,
AquisitionTime => 0);


Data structures in ‘C’:

enum ChannelPlus (Ch0, Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Gnd);  
enum ChannelMinus (Ch0, Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Gnd);  
enum Resolutions (EightBit, TwelveBit);

struct {
    unsigned int EndOfLoop      : 1;
    unsigned int Pause          : 1;
    ChannelPlus Vplus          : 3;
    ChannelMinus Vminus         : 3;
    unsigned int Sync           : 1;
    unsigned int Resolution     : 1;
    unsigned int AquisitionTime : 4;
} Instruction;
### Interfaces

#### LM12L458 – Instructions

Configuration register entries consist of:
- **Start (1 bit):** starts the sequencer.
- **Reset (9 bits):** sets the instruction pointer to 0/0.
- **Auto-Zero (1 bit):** triggers a short calibration (9 cycles - 1 off example).
- **Cal (1 bit):** initiates a full calibration (64 samples) interrupts.
- **Standby (1 bit):** disconnects the external clock and preservers the registers.
- **Power-up (1 bit):** sets and holds power-up state.
- **Chan-Mask (1 bit):** format selection for the FIFO output registers.

#### LM12L458 – Configuration register

<table>
<thead>
<tr>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<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>0</td>
</tr>
</tbody>
</table>

© 2017 Uwe R. Zimmer, The Australian National University page 329 of 962 (chapter 3: “Interfaces” up to page 379)

---

#### A/D, D/A & Interfaces: Examples

**LM12L458 (National Semiconductor)**

In C, use macro-assembler style programming instead:

```c
unsigned int setbits(unsigned int n, unsigned int p);  // set n bits
unsigned int getbits(unsigned int n, unsigned int p);  // get position p
unsigned int bitstring(unsigned int n, unsigned int p);  // to bitstring
```

**Example:**

```c
unsigned int data = 0x1234;
unsigned int mask = 0x00FF;
unsigned int result = getbits(data, 8);
```

© 2017 Uwe R. Zimmer, The Australian National University page 332 of 962 (chapter 3: “Interfaces” up to page 379)

---

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

**Control Register 2**

<table>
<thead>
<tr>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

© 2017 Uwe R. Zimmer, The Australian National University page 335 of 962 (chapter 3: “Interfaces” up to page 379)
Interfaces

Micro-controllers

Typical elements found in a micro-controller include:

- CPU (typ. 44-bit word size — also as microcores)
- Memory (typ. a few hundred bytes to many MBs) as ROM, RAM, EPROM, and/or Flash
- Clock generator
- Timers and general interrupts/ADC
- Basic I/O (often as multiple, partly autonomous I/O units)
  - General-purpose digital I/O lines — often combined with PWM generation, signal width detectors, counters, watchdogs, or time-stamps
  - AD and DA converters (typ. 8-12 bit)
- Higher level I/O sharable: Ethernet, UART, PTT, Retinex, Can-In...

Micro-controller examples

**MC68HC05**

- Clock: max. 2.1 MHz internal
- RAM: 176 bytes
- ROM: 3504 bytes
- EPROM: 256 bytes
- Power saving modes (stop, wait, slow)
- Parallel I/O: 3-bit, Parallel in: 1-bit
- Timer: 16-bit
- A/D 8 channels, 8-bit
- PWM 2 generators

**AVR32**

- CPU: 32-bit RISC with DSP extensions
- Clock: 66 MHz
- Memory: up to 32 MB RAM, up to 512 K Flash
- Separate DMA bus for peripherals
- Power up to 32 MHZ (125 µW in deep
- New debug port
- up to 115 GPIO with up to 7 UART channels.
- 1: 32-bit real-time counter.
- 6 UART lines
- 1: Ethernet, 1: SCI, 4: UART (incl. SPI)
- 8: 16-bit ADC (800ks)
- 2: IIC fow attack output (23-l)

Alternative Processor Architectures: Parallax Propeller

- Low cost 32-bit processor ($16)
- Small shared memory
- No interrupts!
Important note: The document contains several images and diagrams that are not transcribed here as they are not directly relevant to the text content. The text content is focused on the sections of the document that do not rely on the images.

### Micro-controller examples

**STM32F40xxx**

- **Power:** power dissipation: 0.8 - 1.2W, 40°C to 125°C
- **CPU:** PowerPC equipped with DDR & B.I.C., 36MHz
- **RAM:** flash 1MB, static 36kB
- **Time processing units:** 3 (two dual-ported RAM)
- **Timers:** 20 channels (PWM & RTC supported)
- **A/D converters:** 40 channels, 10bit, 250kHz
- **Can bus:** 3 TOUCAM modules
- **Serial:** 2 interfaces
- **Data link controller:** SAE J1939 class II communications module
- **Real-time embedded application development time face:** NEUSIS debug port (IEEE-1149.1-1999)
- **Packing:** 50I/SWIFN PBCA

**MPC565**

- **Power:** power dissipation: 7 - 12W, 40°C to 125°C
- **CPU:** PowerPC equipped with DDR & B.I.C., 36MHz
- **RAM:** flash 1MB, static 36kB
- **Time processing units:** 3 (two dual-ported RAM)
- **Timers:** 20 channels (PWM & RTC supported)
- **A/D converters:** 40 channels, 10bit, 250kHz
- **Can bus:** 3 TOUCAM modules
- **Serial:** 2 interfaces
- **Data link controller:** SAE J1939 class II communications module
- **Real-time embedded application development time face:** NEUSIS debug port (IEEE-1149.1-1999)
- **Packing:** 50I/SWIFN PBCA

### Micro-controller examples

**Time Processing Unit – Emulation Mode**

Create your own µ-engine

Refer the control store of the µ-engine to the dual-ported RAM instead of the integrated ROM area and supply:

- **Up to 16 µ-engine commands (functions).**
- **In 24kB of long word (32 bit) organized memory.**
- **Programmed in a 32bit µ-instruction format (explained next).**

The dual-ported RAM is then cut off from the CPU (the TPU parameter RAM is not affected)

### Micro-controller examples

**Time Processing Unit – µ-instructions formats**

1. Execution unit and ROM:
   - Instruction unit, flag, and channel control
   - Instruction unit, flag, and channel control
   - Conditional branch, flag, and channel control
   - Jump, flag, and RAM
   - Execution unit, immediate, and flag

2. Execution unit and RAM:
   - Instruction unit, flag, and channel control
   - Instruction unit, flag, and channel control
   - Conditional branch, flag, and channel control
   - Jump, flag, and RAM

3. Execution unit, flag, and channel control

4. Jump, flag, and RAM

5. Execution unit, immediate, and flag

**Operation groups:**

- µ-instruction unit
- µ-instruction unit
- µ-instruction unit
- µ-instruction unit
- µ-instruction unit

**Operation groups:**

- Instruction unit
- Instruction unit
- Instruction unit
- Instruction unit
- Instruction unit

**Execution of µ-instructions:**

- Conditional branch
- Jump
- Execution

### Micro-controller examples

**Micro-controller examples**

**STM32F40xxx**

**Discovery board**

- **Power dissipation:** 7 - 12W, 40°C to 125°C
- **CPU:** PowerPC equipped with DDR & B.I.C., 36MHz
- **RAM:** flash 1MB, static 36kB
- **Time processing units:** 3 (two dual-ported RAM)
- **Timers:** 20 channels (PWM & RTC supported)
- **A/D converters:** 40 channels, 10bit, 250kHz
- **Can bus:** 3 TOUCAM modules
- **Serial:** 2 interfaces
- **Data link controller:** SAE J1939 class II communications module
- **Real-time embedded application development time face:** NEUSIS debug port (IEEE-1149.1-1999)
- **Packing:** 50I/SWIFN PBCA

**Micro-controller examples**

**MPC565**

- **Power dissipation:** 7 - 12W, 40°C to 125°C
- **CPU:** PowerPC equipped with DDR & B.I.C., 36MHz
- **RAM:** flash 1MB, static 36kB
- **Time processing units:** 3 (two dual-ported RAM)
- **Timers:** 20 channels (PWM & RTC supported)
- **A/D converters:** 40 channels, 10bit, 250kHz
- **Can bus:** 3 TOUCAM modules
- **Serial:** 2 interfaces
- **Data link controller:** SAE J1939 class II communications module
- **Real-time embedded application development time face:** NEUSIS debug port (IEEE-1149.1-1999)
- **Packing:** 50I/SWIFN PBCA

### Micro-controller examples

**Time Processing Unit – Some predefined µ-engine functions**

- **Period / Pulse-width accumulator**
- **Position-synchronized pulse generator**
- **Packing**
- **Real-time embedded application development time face:** NEUSIS debug port (IEEE-1149.1-1999)
- **Packing:** 50I/SWIFN PBCA
**Micro-controller examples**

### Time Processing Unit – µinstructions formats:

*Micro-controller examples*  
**Time Processing Unit – µinstructions formats:**

**2**: Execution unit, flag, and channel control:
- TFRS: T1 A-Bus Source Control
- TMRD: T3 A-Bus Destination Control
- SHF: AU Shift Control
- TMR: CC Transition Detect Latch Control
- MRL: CC Match Recognition Latch Control
- TIBS: T1 B-Bus Source Control
- ON: AU B-Bus Source Control
- BNV: AU B-Bus Invert Control
- PGA: CC Pin Action Control
- LES: CC Link Service latch Control
- PSC: CC Pin State Control
- FLC: CC Flag Control
- CIR: CC Channel Interrupt Request
- **LEGEND**: SEQ = Decrementer/End Control

**Operation groups**:

- Execution control
- Channel control
- Instruction control

**Micro-controller examples**

### Time Processing Unit – µinstructions formats:

*Micro-controller examples*  
**Time Processing Unit – µinstructions formats:**

**3**: Conditional branch, flag, and channel control:
- BCC: SEQ Branch Condition Control
- TBS: CC TimeBase Select Control
- PAC: CC Pin Action Control
- BCF: SEQ Branch Condition Control
- PAC: CC Pin Action Control
- MFR: CC Match/Transition Detect Request
- **LEGEND**: SEQ = Decrementer/End Control

**Operation groups**:

- Execution control
- Channel control
- Instruction control

**Micro-controller examples**

### Time Processing Unit – µinstructions formats:

*Micro-controller examples*  
**Time Processing Unit – µinstructions formats:**

**4**: Jump, flag, and RAM:
- BCE: SEQ Branch Condition Code Field
- FLS: SEQ µPC Flash Control
- BDF: SEQ Branch Address Field
- PAC: CC Pin Action Control
- BCE: SEQ Branch Condition Code Field
- PAC: CC Pin Action Control
- MFR: CC Match/Transition Detect Request
- **LEGEND**: SEQ = Decrementer/End Control

**Operation groups**:

- Execution control
- Channel control
- Instruction control

**Micro-controller examples**

### Time Processing Unit – µinstructions formats:

*Micro-controller examples*  
**Time Processing Unit – µinstructions formats:**

**5**: Execution unit, immediate, and flag:
- TBRS: T1 A-Bus Source Control
- TMRD: T3 A-Bus Destination Control
- SHF: AU Shift Control
- TMR: CC Transition Detect Latch Control
- MRL: CC Match Recognition Latch Control
- TIBS: T1 B-Bus Source Control
- ON: AU B-Bus Source Control
- BNV: AU B-Bus Invert Control
- PGA: CC Pin Action Control
- LES: CC Link Service latch Control
- PSC: CC Pin State Control
- FLC: CC Flag Control
- CIR: CC Channel Interrupt Request
- **LEGEND**: SEQ = Decrementer/End Control

**Operation groups**:

- Execution control
- Channel control
- Instruction control

**Micro-controller examples**

### Composing the engine:

**Entities to consider:**

- States: non-interceptive code-blocks.
- Functions: constructed of one or multiple states.
- Channels: 16 digital I/O lines with match and capture.
- Priorities of channels.
- Timers: 2 16-bit time-bases.
- Timer interrupts.

**TPU Fixed scheduled, prioritized time slots**

Round Robin schedule for all runnable states inside each priority.
No state will be starved.

**TPU Fixed scheduled, prioritized time slots**

Unused slots will be re-assigned according to priorities and channel numbers.
Interfaces

Micro-controller examples

TPU Fixed scheduled, prioritized time slots

<table>
<thead>
<tr>
<th>Time Slot</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20-25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40-45</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

States have different and variable lengths.

Calculating actual and maximal latencies requires full understanding of all states.

Interfaces

Micro-controller examples

Latencies on TPU

... for latencies of capture and match at each channel mind:
- only the time-base resolution (all channels are evaluated independently and in parallel).
- for the functions associated with individual channels mind the:
  - number of active channels (max. 16).
  - number of channels on each priority level (add max. 2 µ-cycles each state-switch).
  - number of µ-cycles to execute individual states (2-µ-cycles per µ-instruction).
  - number of RAM accesses during the execution of a state (each access may still for 2CPU cycles).
  - TPU clock cycle frequency.

Interfaces

Micro-controller examples

Determining actual TPU latencies

Emulate the known executing ready times for all states.
- Add two 2 µ-cycles for each state switch.

Interfaces

Micro-controller examples

Determining maximal TPU latencies

Assume all states on the same priority runnable at all times and run their maximal lengths.
- Assume the longest state out of all higher priorities runnable at all times.
- Assume the longest state out of all lower priority runnable at all times.
- Deploy a set of those states which will cause the longest latency.
- Determine the longest latency inside a full hyper-cycle.

Interfaces

Micro-controller examples

Determining maximal TPU latencies

A special-purpose micro-controller:
- 64 bit time base (driven by an external clock e.g. 200MHz: resolution: 50ns, range ~30,000 years).
- Free running (not influenced by any CPU action or results).
- 2 reference registers (used for compare and interrupt generation).
- Real-Time clock supplies full seconds (32bit with range ~136 years) (not affected by CPU, resets, and operates in all low-power modes).
Micro-controller examples

**MPC565**

- **Interrupt controller**
  - Handles up to 48 different sources (32 from internal modules, 8 from timers, and clock, and 8 external vectorized sources) and support each of them with a unique interrupt vector.
  - 8 interrupt levels are distinguished by the interrupt controller (8 interrupt levels are supplied by the internal modules, prioritized and vectorized interrupts are supplied by external sources).
  - Latency: 20 clock cycles
    - bus collisions + CPU state saving + tasking system overhead

- **Power**
  - Power dissipation: 0.8 - 1.12 W, -40º - +125ºC

- **CPU**
  - PowerPC core (incl. FPU & IMC), 56 MHz

- **On-Line mode**
  - Handles up to 48 different sources
  - Data trace: via data write messaging and data read messaging (can be reduced to selected areas).
  - Owner trace: via ownership trace messaging (also indicates task creation and activation)
  - Run-time access to memory map and special CPU registers.
  - Watchpoints: CPU watchpoint status signals are snooped and transferred with high priority

- **Off-line mode**
  - Read / Write access the READY module can take over the L-bus to manipulate data.
  - Access to all CPU registers during build.

- **Latency**
  - 20 clock cycles
  - 20 clock cycles

- **Timers**
  - A/D convertors: 40 channels, 10 bit, 250 kHz

- **Serial**
  - 32 interrupt levels are supplied by the internal modules, prioritized

- **Can-bus**
  - 32 interrupt levels are supplied by the internal modules, prioritized
  - 3 interrupt levels are supplied by the internal modules, prioritized

- **Data trace**
  - Data trace: via data write messaging and data read messaging (can be reduced to selected areas).

- **Owner trace**
  - Owner trace: via ownership trace messaging (also indicates task creation and activation)

- **Run-time access**
  - Run-time access to memory map and special CPU registers.

- **Watchpoints**
  - Watchpoints: CPU watchpoint status signals are snooped and transferred with high priority

**Micro-controller examples**

**MPC565**

- Power: power dissipation 0.8 - 1.12 W, -40º - +125ºC
- CPU: PowerPC core (incl. FPU & IMC), 56 MHz
- RAM: flash: 1 MB, static: 36 kB, 25 MHz
- Time processing units: 3 via dual-port RAM
- Timers: 22 channels (PWM & ITC supported)
- A/D convertors: 40 channels, 10 bit, 250 kHz
- Can-bus: 3 TOUCAN modules
- Serial: UARTs
- Data link controller: SAE J1850 class B communication module
- Real-time enabled application development interface: NEXUS debug port (IEEE-ISTO 5001-1999)
- Packing: 352/388 ball PBGA

**Interface architectures**

**Basic sampling control mechanisms**

- **Status driven**
  - The computer polls for information (used in dedicated micro-controllers and prescheduled hard real-time environments).

- **Interrupt driven**
  - The data generating device issues an interrupt when new data has been detected / converted or when internal buffers are full.
  - The program is blocked until the interrupt is handled by the CPU.
  - Handling using special interrupt handling routines, e.g., changing status information, execution of a procedure, raising an exception, etc.

- **Program controlled**
  - The interrupts are handled by the CPU.
  - Handling using special interrupt handling routines, e.g., changing status information, execution of a procedure, raising an exception, etc.

- **Program initiated**
  - The interrupts are handled by a DMA controller. No processing is performed. Depending on the DMA setup, cycle stealing can occur and needs to be considered for worst case computing times.

- **Channel program controlled**
  - The interrupts are handled by a dedicated channel device. The data is transferred and processed. Optional memory-based communication with the CPU.
  - The channel controller is usually itself a dedicated processor / controller.

**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 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.**

**Language requirements for interfaces**

- Specify the device interface (protocol and formats) in all detail (candidates: Ada, C++, ERLANG, Modula-2, C, ... or Macro-Assemblers level) (platform independence or abstraction is not required!).
- Handling asynchronous hardware messages (devices, timers, ...) Many different methods to implement a context-switch (candidates: all languages with some real-time orientation: PEARL, C++, ERLANG, Ada, C++/POSIX, ...)
- The term “high-level languages” in the real-time interface context: Allow for strong abstraction while being time and physics specific down to the actual level of interface realities

**Converters & Interfaces**

- **Analogue signal chain in a digital system**
  - Sampling data, aliasing, Nyquist’s criterion, oversampling
  - Quantization (LSB, rms noise voltage, SNR, ENOB), Missing codes, DNL, INL
  - Many different methods to implement a context-switch (candidates: all languages with some real-time orientation: PEARL, C++, ERLANG, Ada, C++/POSIX, ...)
- **Examples**
  - Fast and simple A/D converter example: National Semiconductor ADC08200
  - Multi-channel A/D: data logging interface example: National Semiconductor UM1203
  - Simple interface example: Micro-µC (µC8051C, Freescale)
  - Complex A/D interface example: AD922 and Mentor A/D converters (MPC565 including TPs)