Timers and Analog to Digital Conversion: Tick tock tick tock...

Gabriel Hugh Elkaim
The world

- L2C
  - 10 ports
  - timing
  - comm

MC

- inputs
- 1/0 pins
- comm
- D/A

Actuators

Feedback

CMPE 118/218 - Intro. to Mechatronics
Timers

Counter

Stream of flip-flops

Timer

Counter which is driven by a fixed time base
Controller - medium complexity

Dedicated chips (SPI)

RTCC - 1 year/year

GPS - 1 Hz PPS 1 year/year
Carla size \[
\frac{B}{16}
\] - 0 - 25 C
32 - 0 - 65 H
0 - 4 Billion

clock pulse FCY
Announced elsewhere

\[
\text{PHY}
\]
\[
\text{TMRx}
\]
Figure 16-1: Output Compare Module Block Diagram

- OCxRS\(^{(1)}\)
- OCxR\(^{(1)}\)
- Comparator
- OCTSEL
- TMR register inputs from time bases\(^{(3)}\)
- Period match signals from time bases\(^{(3)}\)
- Set Flag bit OCxF\(^{(1)}\)
- Output Logic
- Output Enable
- OCxF\(^{(1)}\)
- OCM-2.0-Mode Select
- OCFA or OCFB\(^{(2)}\)
 gated timer indicator

GATED
Timer
\[ f = \frac{1}{\text{Period}} \]

\[ \text{Duty Cycle} = \frac{\text{High}}{\text{Period}} \times 100\% \]

0 - 99.9%
Period path

Interrupt
Figure 15-1: Input Capture Module Block Diagram

ICx Input

Prescaler 1, 4, 16

Edge Detect

ICTMR

C32

FIFO Control

ICxBUF<31:16> ICxBUF<15:0>

ICM<2:0> FEDGE

ICBNE ICOV

Interrupt Event Generation

Data Space Interface

Interrupt

Peripheral Data Bus

Timer3 Timer2

0 1
\[ \frac{1}{32} \text{ inch} = 800 \text{ mm} \]

FPB = \( \frac{\text{Fcm}}{2} = \frac{80 \times 10^6}{2} = 40 \times 10^6 \) mm

Tinch = 0.25 mm

\( \Delta T = \text{curr} - \text{prev} \)
PWM-Duty Modulation

\[ \frac{1}{D(x)C} \]

LEFT EDGE ALIGNED

CENTER ALIGNED (BLDC)
ADC conversion

Figure 17.1: 10-bit High-Speed ADC Block Diagram

Note 1: VREF+ and VREF- inputs can be multiplexed with other analog inputs.

Successive approximate registers
Figure 17-5: Simplified 10-bit High-Speed ADC Block Diagram for Alternate Sample Mode
**Conversion Code**

<table>
<thead>
<tr>
<th>Range of Analog Input Value (V)</th>
<th>Digital Output Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5 - 6.6</td>
<td>0_ 101</td>
</tr>
<tr>
<td>3.5 - 4.5</td>
<td>0_ 100</td>
</tr>
<tr>
<td>2.5 - 3.5</td>
<td>0_ 011</td>
</tr>
<tr>
<td>1.5 - 2.5</td>
<td>0_ 010</td>
</tr>
<tr>
<td>0.5 - 1.5</td>
<td>0_ 001</td>
</tr>
<tr>
<td>0 - 0.6</td>
<td>0_ 000</td>
</tr>
</tbody>
</table>

**Ideal Straight Line**

**Step Width (1 LSB)**

**Midstep Value of 0_ 011**

**Quantization Error**

**Inherent Quantization Error (±1/2 LSB)**

Elements of Transfer Diagram for an ideal Linear ADC
Single slope ADC
Double slope ADC
5 V R

3.3

78 kHz

\[ V_{in} \]

\[ \text{AND} \]

\[ \text{DAC} \]
V_{in} 

\text{Dac} 

\text{Capacitor} \quad 1 \text{ mF} 

24 \text{ Ohms} 

\text{Inductor}
\[ T - \frac{1}{2} c_{50} \mathrm{Noise} \]

\[ \frac{1}{2} c_{50} \]

\[ \text{Delay } \frac{1}{2} T \]

\[ \frac{\text{Slew Rate}}{\Delta t} \text{Limited Slope} \]
Claude Shannon

Nyquist - \( \left( \frac{3\pi}{2} \right) \)

Anti-Aliasing Filter

\( \frac{f_s}{2} \)
Figure 19.1: Comparator Block Diagram

Comparator 1

Comparator 2

Note 1: On devices with a USB module, and when the module is enabled, this pin is controlled by the USB module, and therefore, is not available as a comparator input.

Note 2: Internally connected.
Figure 20.1: Comparator Voltage Reference Block Diagram

Note 1: These bits are not available on all devices. On such devices, CVREF is generated by the resistor network and IVREF is connected to 1.2V. Refer to the specific device data sheet for availability.
Elements of Transfer Diagram for an Ideal Linear DAC

<table>
<thead>
<tr>
<th>Digital Input Code</th>
<th>0...000</th>
<th>0...001</th>
<th>0...010</th>
<th>0...011</th>
<th>0...100</th>
<th>0...101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Output Value</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
Offset error of a Linear 3-Bit Natural Binary Code Converter
(Specified at Step 000)

(a) ADC
(b) DAC

Nominal Offset Point
Actual Offset Point
Offset Error (+1 1/4 LSB)

Digital Input Code
Analog Input Value
Offset Output Value (LSB)
Nominal Offset Point
Actual Offset Point
Offset Error (+1 1/4 LSB)
Gain Error of a Linear 3-Bit Natural Binary Code Converter (Specified at Step 111), After Correction of the Offset Error
End-Point Linearity Error of a Linear 3-Bit Natural Binary-Coded ADC or DAC (Offset Error and Gain Error are Adjusted to the Value Zero)
Differential Linearity Error of a Linear ADC or DAC

LINEARITY 
$\pm 1\%$ of Full Scale
Absolute Accuracy or Total Error of a Linear ADC or DAC

(a) ADC

- Total Error At Step 0...101 (±1 1/4 LSB)
- Total Error At Step 0...001 (1/2 LSB)

(b) DAC

- Total Error At Step 0...011 (1 1/4 LSB)
Questions?
Communication Links
What’s the frequency, Kenneth?

Gabriel Hugh Elkaim
Summarize

1. Last week Thursday @ 9:50 in Jacks
2. T-shirts
3. PDA cam
4. T8 evaluations / Instructor evals,
   @ Paolo Vlahos & Kyle Cortesi
Comm Links (1.3)

- Information passing between 2 (or more) devices
- Devices need to agree on “rules of the road”
- Lots of different ways to do this
  - Direct Wired: serial, RS-232, RS-485, Ethernet, SPI, I2C, USB, CAN
  - Light: IrDA, Fiber optic, RONJA
  - RF: WiFi (802.11a/b/g/n), Bluetooth, Zigbee
  - Fluid: Acoustic Modems
Comm Links (2.3)

- In order for communication to work:
  1. Both the source & the destination must have access to the media
  2. A priori - agree on symbols to be used
  3. A priori - agree on rules that govern who talks, who listens, how long, etc.
Comm Links (3.3)

- Digital communications:
- Parallel – One bit per wire, lots of wires
- Serial – One bit at a time, lots of time
- Synchronous – Has a common clock
- Asynchronous – No common clock, we agree at the risk of a bit error

Gabriel Hugh Elkaim
Common Standard Links

- UART/Serial – Asynchronous serial \( \frac{1}{2} \) duplex one-to-one

- SPI – Serial Peripheral Interface (Motorola)
  Synchronous, serial \( \frac{1}{2} \) duplex, master/slave

- I2C – Inter-Integrated Circuit Communication (Philips)
  Synchronous, multiplexed

- CAN – Controller Area Network (Bosch)
  Multiplexed/Shared Memory Space
  Synchronous/Semi-Synchronous
Basic **Synchronous** Serial Communication

- **3.3V**
- **0**

- **DATA**
- **Clock**

- Which edge is valid data?
- LSB first?
Basic Asynchronous Serial Communication

- **Sync**: Signal indicating the start of a frame.
- **Data**: The actual data being transmitted.
- **Done**: Signal indicating the end of the frame.
- **Start**: Initiates the transmission.

- **LSB**: Least Significant Bit, no parity.
- **8-N-1**: 8 data bits, 1 stop bit.

---

CMPE 118/218 – Intro. to Mechatronics

Gabriel Hugh Elkaim
Serial (UART) Communication (1.3)

Diagram showing the start bit, data bit, parity bit, and stop bit in UART communication.
Serial (UART) Communication (2.3)

8-bit

8 x example

# 3, 5, 7 - simple majority rule 1 bit

noise bit is 1, but remains
Serial (UART) Communication (3.3)
UART on PIC32

Figure 21-1: UART Simplified Block Diagram

Peripheral Bus Clock: 40 MHz

<table>
<thead>
<tr>
<th>Target Baud Rate</th>
<th>Actual Baud Rate</th>
<th>% Error</th>
<th>BRG Value (decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>110.0</td>
<td>0.00%</td>
<td>22726.0</td>
</tr>
<tr>
<td>300</td>
<td>300.0</td>
<td>0.00%</td>
<td>8332.0</td>
</tr>
<tr>
<td>1200</td>
<td>1200.2</td>
<td>0.02%</td>
<td>2082.0</td>
</tr>
<tr>
<td>2400</td>
<td>2399.2</td>
<td>-0.03%</td>
<td>1041.0</td>
</tr>
<tr>
<td><strong>9600</strong></td>
<td><strong>9615.4</strong></td>
<td><strong>0.16%</strong></td>
<td><strong>259.0</strong></td>
</tr>
<tr>
<td>19.2 K</td>
<td>19230.8</td>
<td>0.16%</td>
<td>129.0</td>
</tr>
<tr>
<td>38.4 K</td>
<td>38461.5</td>
<td>0.16%</td>
<td>64.0</td>
</tr>
<tr>
<td>56 K</td>
<td>55555.6</td>
<td>-0.79%</td>
<td>44.0</td>
</tr>
<tr>
<td><strong>115 K</strong></td>
<td><strong>113636.4</strong></td>
<td><strong>-1.19%</strong></td>
<td><strong>21.0</strong></td>
</tr>
<tr>
<td>250 K</td>
<td>250000.0</td>
<td>0.00%</td>
<td>9.0</td>
</tr>
<tr>
<td>300 K</td>
<td>300000.0</td>
<td>0.00%</td>
<td>4.0</td>
</tr>
<tr>
<td>500 K</td>
<td>500000.0</td>
<td>0.00%</td>
<td>0.0</td>
</tr>
<tr>
<td>Min. Rate</td>
<td>38.1</td>
<td>0.00%</td>
<td>65535</td>
</tr>
<tr>
<td>Max. Rate</td>
<td>2500000</td>
<td>0.00%</td>
<td>0.0</td>
</tr>
</tbody>
</table>
UART on PIC32

Diagram of UART operation on PIC32 microcontroller.
UART on PIC32

Figure 21-4: Transmission (8-Bit or 9-Bit Data)

- Write to UxTXREG
- BCLK/16 (Shift Clock)
- UxTX
- UxTXIF
- TRMT bit
- Character 1
- Start bit
- bit 0
- bit 1
- bit 7/8
- Stop bit
- Character 1 to Transmit Shift Register
- UxTXIF Cleared by User
UART on PIC32

![Diagram of UART on PIC32](image-url)
UART on PIC32

Figure 21-8: UART Reception

Note: This timing diagram shows 2 characters received on the UxRX input.
Software Serial Implementation

TX

Circular Buffer

Is Transmit Empty ()

RX
Software Serial Implementation

```c
void _ISR(_UART1_VECTOR, ipl4) IntUart1Handler(void)
{
    if (INTGetFlag(INT_U1RX)) {
        INTClearFlag(INT_U1RX);
        writeBack(receiveBuffer, (unsigned char) U1RXREG);
    }
    if (INTGetFlag(INT_U1TX)) {
        INTClearFlag(INT_U1TX);
        if (!getLength(transmitBuffer) == 0)) {
            U1TXREG = readFront(transmitBuffer);
        }
    }
}

void PutChar(char ch)
{
    if (getLength(transmitBuffer) != QUEUESIZE) {
        writeBack(transmitBuffer, ch);
        if (U1STAbits.TRMT) {
            INTSetFlag(INT_U1TX);
        }
    }
}

char GetChar(void)
{
    char ch;
    if (getLength(receiveBuffer) == 0) {
        ch = 0;
    } else {
        ch = readFront(receiveBuffer);
    }
    return ch;
}
```
SPI – Serial Peripheral Interface

Synchronous – carrier clock – 100 MHz
920 MHz

Sync IN SCL

RX

TX

Bit TX
Clock Phase (CPHA) vs. Clock Polarity (CPOL)

Clock Phase 1

Sample

IDLE

"mode 2"

Sample

"mode 1"

Sample

"mode 3"
**SPI -- Implementation**

Figure 23-1: Typical SPI Master-to-Slave Device Connection Diagram

---

**Note 1:** In Normal mode, the usage of the Slave Select pin (SSx) is optional.
**Note 2:** Control of the SDO pin can be disabled for Receive-Only modes.
SPI – Serial Peripheral Interface
SPI on PIC32

Figure 23-5: SPI Module Block Diagram
SPI vs. I2C

SPI mode

Processor 1
Master

Processor 2
Slave

I²C mode

Processor 1
Master

Processor 2
Slave
I2C – Inter-Integrated Circuit

- Start
- Address
- 7 Bits
- Data/Command
- Ack

- SDA
- SCL

- 3.3V
- 5V
I2C – Inter-Integrated Circuit
I2C – Inter-Integrated Circuit

Master (MCU)

START

Address 1
Address 2
Address 3

Slave
Slave
Slave

EEPROM

DATA TRANSFER START

ADDRESS
BYTE 1
BYTE 2
BYTE 3
BYTE n

DATA TRANSFER END

Acknowledgment

START bit (0)
R/W
0 = WRITE
1 = READ

Address

BYTE n-1
7 6 5 4 3 2 1 0

Data

STOP bit (1)
I2C – Inter-Integrated Circuit

Start sequence

Stop sequence

SDA
SCL

A6 A5 A4 A3 A2 A1 A0 R/W ACK

SDA
SCL

1 2 3 4 5 6 7 8 9

Compass uses address 0xC0

Write The register number that you want to read from

Repeated Start bit

Write address with b8=0 set - 0xC1

Read one or more registers

Stop bit

ACK

CMPE 118/218 – Intro. to Mechatronics
I2C on PIC32

Diagram of the I2C interface on PIC32 microcontroller, showing various components such as SCK, SDA, I2CxRCV, I2CRSR, Match detect, Addr_Match, I2CADRMASK, I2CxADD, Start and Stop bit detect, Start, Restart, Stop bit generate, Collision Detect, Acknowledge Generation, Clock Stretching, I2CxTRN, Reload Control, BRG Down Counter, and PBCLK.
I2C on PIC32

Figure 24-7: Typical I²C Message: Read of Serial EEPROM (Random Address Mode)
CAN – Controller Area Network

Designed for CANs

Robust

Differential Signal

Common Mode Noise

CSMA/NDT
CAN – Controller Area Network

![Diagram of CAN network with nodes and connections]

**CAN Node**
- Microcontroller
- CAN Controller
- CAN Transceiver

**Data Link Layer**
- ISO 11898-1
- Medium Access Unit (Electrical Levels) ISO 11898-2.3

**Nominal Bit Time Diagram**
- Previous bit: Sync, Prop, Phase 1, Phase 2
- Next bit: Sample Point, Time Quanta

**Network Connections**
- SG 1, SG 2, SG n
- CAN-Hi
- CAN-Low
- 120 Ω terminations
CAN – Controller Area Network

Two logic states:
"1" = recessive
"0" = dominant

Bus State: dominant

CAN-Bus
(single logical bus line)

Node A
dominant

Node B
recessive

Node C
recessive

5V
CAN – Controller Area Network

Standard Frame Format
- SOF
- 11-Bit Arbitration ID
- RTR
- IDE
- DLC
- 0–8 Data Bytes
- 16-Bit CRC
- ACK
- End of Frame

Extended Frame Format
- SOF
- High 11-Bits of Arbitration ID
- IDE
- Low 10-Bits of Arbitration ID
- RTR
- DLC
- 0–8 Data Bytes
- 16-Bit CRC
- ACK
- End of Frame
CAN – Bus Arbitration

Device A transmits
ID = 110 0100 0111 (647 hex)

Device B transmits
ID = 110 1100 0111 (6C7 hex)

Device B loses and goes idle until end of frame
Device A wins, and proceeds
CAN – Bus Arbitration

- Node X
- Node A
- Node B
- Node C

Arbitration phase:
- Start
- Identifier Field
- Bit

Remaining:
- Node A
- Node B
- Node C
- CAN Bus

Transmit Request:
- Node B loses Arbitration
- Node C loses Arbitration
CAN on PIC32

Figure 34-1: Typical CAN Bus Network

- PIC32
  - CAN1
  - CAN2
  - CAN Transceiver
  - CAN Transceiver

- CAN bus

- CAN Transceiver
- dsPIC33F with Integrated ECAN™
- dsPIC30F with Integrated CAN
- PIC® with Integrated ECAN
CAN on PIC32

Interframe Space

Arbitration Field

Control Field

Data Field

CRC Field

ACK Field

End-of-Frame

Interframe Space

SOF

11-bit Identifier

SIDE 10  ---  SID1  ---  SID0

11-bit Identifier

Arbitration Field

Control Field

Data Field

CRC Field

ACK Field

End of Frame

SOF

11-bit Identifier

SIDE 10  ---  SID1  ---  SID0

29-bit Identifier

SIDE 10  ---  SID1  ---  SID0  ---  EID17  ---  EID1  ---  EID0

IDE is Dominant (Logical '0')

RTR is Dominant (Logical '0')

RB0 is Dominant (Logical '0')

IDE is Recessive (Logical '1')

SRR is Recessive (Logical '1')

RTR is Dominant (Logical '0')

RB0 is Recessive (Logical '0')

RB1 is Dominant (Logical '0')
CAN Message Filtering

<table>
<thead>
<tr>
<th>MASK3</th>
<th>MASK2</th>
<th>MASK1</th>
<th>MASK0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>FFFFFFFE</td>
<td>FFF00FF</td>
<td>FFFFFF</td>
</tr>
</tbody>
</table>

- **ID Match = 1**
- **Message Assembly Buffer (MAB) = 01234567**
- **CAN FSM** Filter Select = 1
- **MSEL**
  - MSEL=1 RXF31 Filter = 2A2B3FD FSEL31 = 9
  - MSEL=2 RXF1 Filter = 01234566 FSEL1 = 2
  - MSEL=0 RXF0 Filter = 01234565 FSEL0 = 10

- **Current Mask = FFFFFFFE**
- **Current Filter = 01234566**
CAN Fault Isolation

- **Error Active**
  - RERRCNT > 127 or TERRCNT > 127
  - RERRCNT < 127 or RERRCNT < 127

- **Error Passive**

- **Bus Off**
  - TERRCNT > 255

- **Reset**
  - 128 occurrences of 11 consecutive "recessive" bits