Microcontrollers en Elektronische Basisschakelingen Les 3

Analoog-Digitaal Conversie in XC8 met PIC Microcontrollers

A.M. Gieling, M. Oldenburg

Domein Techniek, Ontwerpen en Informatica
Cluster ICT
Opleiding Technische Informatica

19 september 2023

Inhoudsopgave

ADC?

- AD-Conversie is het samplen van een **analoge spanning** en deze representeren als een **digitale meetwaarde**.
- De 16F1829 heeft 11 analoge input-kanalen en kan samplen met een resolutie van 10 bits (waarden tussen 0 en 1023).
- De relatie tussen digitale meetwaarde ('ADC') en analoge spanning ('V') is als volgt (datasheet):

EQUATION 3-2:

$$ADC = (V/V_{REF}) * 1023$$

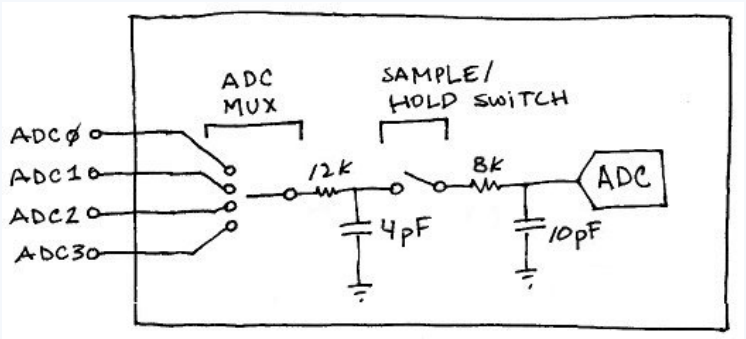
Converting the answer from the ADC back to voltage requires solving for V.

$$V = (ADC/1023) * V_{REF}$$

Sample & Hold

- Acquisitie van V_{IN} middels een 'sample & hold' circuit. Hoe kun je een spanning 'even vasthouden' en isoleren van de rest van het circuit op een moment dat het jou (de developer) uitkomt?

Sample & Hold

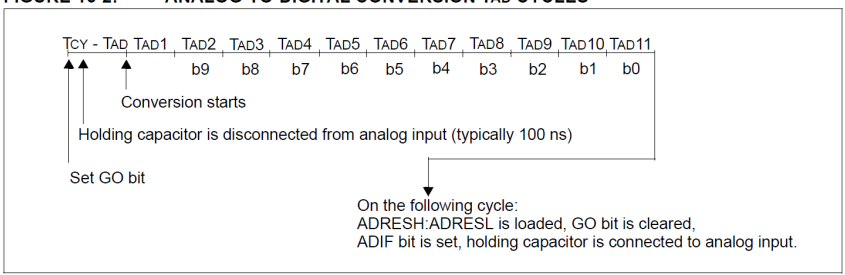


- Een RC-circuit achter een schakelaar, bedienbaar vanuit code (*ADON* en *GO*/$\overline{DONE}$ bits)!
- Meten kost tijd, want het opladen van C_{HOLD} kost tijd.
- Selecteer altijd eerst een kanaal in de ADC multiplexer.

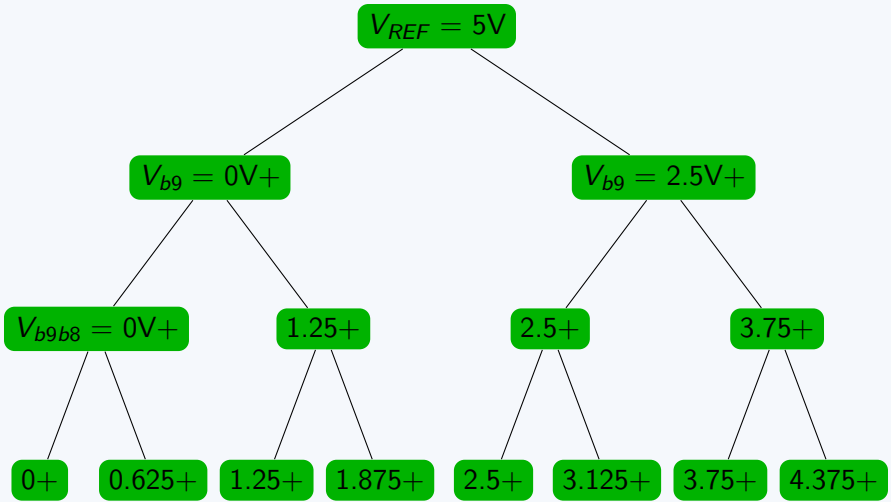
Successive Approximation

- Binary Search toepassen om van MSb naar LSb een vergelijkingswaarde te vinden:
- T_{AD2}: zet $b9$ *tentatief* op 1 en maak via spanningsdeler (DAC) $V_{b9} = \frac{V_{REF}}{2}$. Als $Meting_{ADC} < V_{b9} (= 2.5V)$, clear $b9$ (maakt $V_{b9} = 0V$).
- T_{AD3}: zet $b8$ op 1 en maak via spanningsdeler (DAC) $V_{b9b8} = V_{b9} + \frac{(\frac{V_{REF}}{2})}{2}$. Als $Meting_{ADC} < [0V \oplus 2.5V] + 1.25V$, clear $b8$. Etc., etc., zie ook de datasheet:

FIGURE 16-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES



Successive Approximation, 5V, b9b8b7



T_{AD} vs. F_{OSC}

- Kies een economische en levensvatbare klokdeler (F_{OSC} / X) voor je applicatie, zie de datasheet:

TABLE 16-1: ADC CLOCK PERIOD (TAD) VS. DEVICE OPERATING FREQUENCIES

ADC Clock Period (TAD)		Device Frequency (FOSC)					
ADC Clock Source	ADCS<2:0>	32 MHz	20 MHz	16 MHz	8 MHz	4 MHz	1 MHz
Fosc/2	000	62.5ns(2)	100 ns(2)	125 ns(2)	250 ns(2)	500 ns(2)	2.0 µs
Fosc/4	100	125 ns(2)	200 ns(2)	250 ns(2)	500 ns(2)	1.0 µs	4.0 µs
Fosc/8	001	0.5 µs(2)	400 ns(2)	0.5 µs(2)	1.0 µs	2.0 µs	8.0 µs(3)
Fosc/16	101	800 ns	800 ns	1.0 µs	2.0 µs	4.0 µs	16.0 µs(3)
Fosc/32	010	1.0 µs	1.6 µs	2.0 µs	4.0 µs	8.0 µs(3)	32.0 µs(3)
Fosc/64	110	2.0 µs	3.2 µs	4.0 µs	8.0 µs(3)	16.0 µs(3)	64.0 µs(3)
FRC	x11	1.0-6.0 µs(1,4)	1.0-6.0 µs(1,4)	1.0-6.0 µs(1,4)	1.0-6.0 µs(1,4)	1.0-6.0 µs(1,4)	1.0-6.0 µs(1,4)

Legend: Shaded cells are outside of recommended range.

Note 1: The FRC source has a typical TAD time of 1.6 µs for VDD.

2: These values violate the minimum required TAD time.

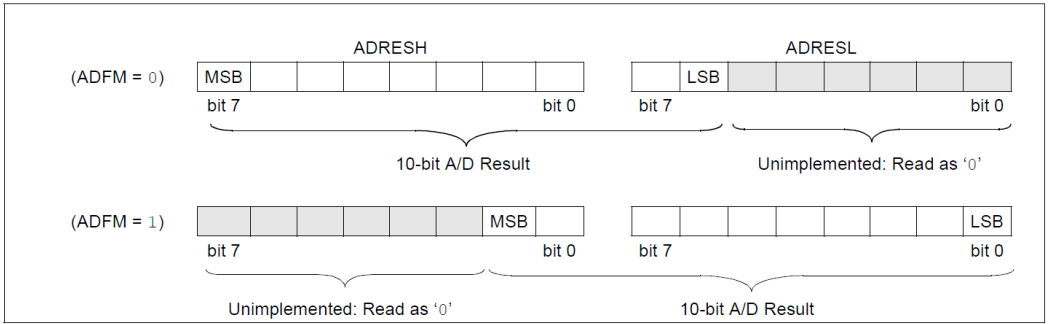
3: For faster conversion times, the selection of another clock source is recommended.

4: The ADC clock period (TAD) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the FRC clock source must be used when conversions are to be performed with the device in Sleep mode.

ADRESH/L SFRs en Result Justification

- Selecteer wat voor jouw applicatie het handigst is, zie datasheet:

FIGURE 16-3: 10-BIT A/D CONVERSION RESULT FORMAT



ADC Procedure

Zie de datasheet, §16.2.6, pagina 154:

1. Configureer poort voor analoge input (TRIS/ANSEL).
2. Configureer ADC module: conversion clock, V_{REF}, kanaal, zet module AAN.
3. Wacht tot de aqcuisitie-condensator (C_{HOLD}) vol is ($> 4\mu$s, max. 6μs).
4. Start conversie door ADCON0.GO/$\overline{DONE}$ hoog te maken.
5. Wacht tot ADCON0.GO/$\overline{DONE}$ laag wordt (conversie klaar).
6. Lees het ADC-resultaat uit SFRs ADRESH en ADRESL.
7. Herhaal bij meerdere metingen vanaf stap 3!

Lesopdracht (30 min)

- Maak een leeg nieuw XC8-project aan in MPLABX.
- Schrijf ‘Midpoint Check’ met LED-indicatie voor de potmeter.
- D.w.z. bedenk een programma dat LED1 aan- of uitzet afhankelijk van de stand van de potmeter: onder het midden van de pot is de LED ‘uit’ en boven het midden is de LED ‘aan’! Hoe doe je dat?

Voorbeelduitwerking

Hieronder is alleen de inhoud van de main program loop te zien. De rest van de code is uiteraard vormgegeven volgens instructies rondom **modulariteit** (zie de startercode op M00dle als voorbeeld):

```
1   while(1)
2   {
3       __delay_us(5);      //wait for ADC charging cap to settle
4       GO = 1;             //ADCON0.GO
5       while (GO) continue; //wait for conversion to be finished
6       // Can you optimize this, e.g. by 'hacking' successive approximation?
7       if (ADRES >= 512)
8           LATCbits.LATC0 = 1;
9       else
10          LATCbits.LATC0 = 0;
11  }
```

Deze code is alleen een kort voorbeeld voor de aanpak (nadruk = begrijpelijkheid). Leesbaarheid heeft hier echter geen aandacht gehad (zelf doen!). '`LATCbits.LATC0 = 1`' schrijf je natuurlijk leesbaar op, bijvoorbeeld zoiets als '`LED = AAN`' ...

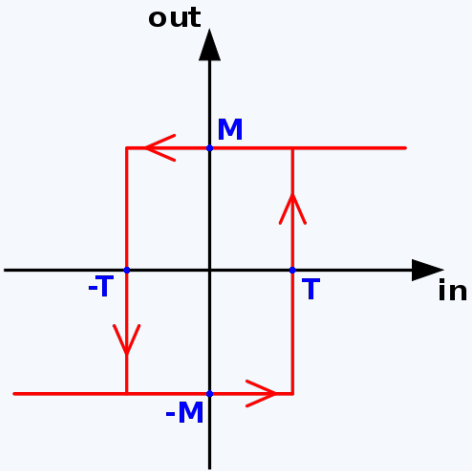
Jitter

- Nabij de overgangswaarde van 512 ($\sim 2.5V$) wil de LED wel eens 'zomaar' aan en uit springen (jitter). Hoe komt dit?

Jitter

- Nabij de overgangswaarde van 512 ($\sim 2.5V$) wil de LED wel eens 'zomaar' aan en uit springen (jitter). Hoe komt dit?
- Noise (ruis). Dit is in de praktijk niet te voorkomen!
- Hoe zorgen we voor een nette discretisatie?

Jitter: hysterese (dead zone) inbouwen



Lesopdracht (15 min)

- Experimenteer met de code van Lesson 4.
- Verbouw de XC8 code van Lesson 4 en de eerste lesopdracht zodanig, dat je een Midpoint Check maakt met ingebouwde hysterese. De LED mag rondom de overgang geen ongedetermineerd(e) geknipper (jitter) meer vertonen!

Voorbeelduitwerking

```
enum bistates
{
    LOW,
    HIGH
} bistate;
```

```
    while(1)
    {
        __delay_us(5);      //wait for ADC charging cap to settle
        GO = 1;             //ADCON0.GO
        while (GO) continue; //wait for conversion to be finished
        // Total dead area of about 20 'units' around the exact middle
        // I.e. hysteresis has a magnitude of 10 units (to each side).
        if (ADRES > 522)
            bistate = HIGH;
        else if (ADRES < 502)
            bistate = LOW;
        LATCbits.LATC0 = bistate;
    }
}
```

Huiswerk

Verdeel je zelfstudie-uren over de volgende zaken:

- Bouw de schakeling die hoort bij week 3 en voer de bijbehorende programmeeropdracht uit. Laat je resultaat tijdens een van de komende lessen aan 1 van de docenten zien.