Zeno Scheltens

[z.f.scheltens@st.hanze.nl](mailto:z.f.scheltens@st.hanze.nl)

ELVE1 groep 3

Kit Yi Feng

[k.y.feng@st.hanze.nl](mailto:k.y.feng@st.hanze.nl)

ELVM1 groep 1

Ontwerp en evaluatie document programmeren

Analog to digital converter

ADC, is het niet ABC? Ook wel: van een golf naar een ladder.

Inhoud

[Analyse 3](#_Toc498286840)

[Ontwerp 4](#_Toc498286841)

[Flowcharts 4](#_Toc498286842)

[Main() 4](#_Toc498286843)

[Functies 4](#_Toc498286844)

[Implementatie 5](#_Toc498286845)

[Evaluatie 6](#_Toc498286846)

# Analyse

Een microcontroller kan eigenlijk alleen maar goed rekenen met digitale signalen. Iets moet een 1 of een 0 zijn. Analoge waarden kunnen echter elke spanning tussen 0V en Vref aannemen. Om toch met een analoog signaal te kunnen werken moet deze worden omgezet in een digitaal signaal. Dit gebeurt met een Analog to digital converter, afgekort ADC.

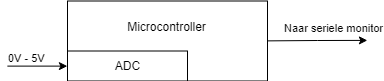
Er zijn verschillende soorten ADC’s. Deze onderscheiden zich door in hoeveel bits het signaal kan worden omgezet en de frequentie waarbij dit gebeurt. Doorgaans zijn technieken die een kleinere resolutie hebben trager.

Het doel van deze opdracht is om uit te zoeken of de ADC in de microcontroller lineair werkt. Om dit te testen moet de microcontroller worden geprogrammeerd zodat deze de output van de ADC via een seriële verbinding naar de pc stuurt. Voor het instellen van de ADC word uitgezocht welke registers hier bij horen en welke instellingen deze heeft.

Om de lineariteit te testen worden er verschillende spanningen op de ADC gezet en in een grafiek geplot met de waarden die de ADC bij deze spanningen geeft.

# Ontwerp

De ADC krijgt als input een variabele spanning tussen de 0 en 5 volt. De output gaat via ttl-serial omzettingen naar de seriële monitor op de laptop.



Er zijn meerdere pinnen op de microcontroller die gebruikt kunnen worden voor een signaal dat via de ADC moet worden omgezet. Dit zijn AN0 t/m AN4 en AN8 t/m 10. De bijbehorende pinnen zijn 2, 3, 4, 5, 7, 21, 22, 25. Op het PCB zijn AN1 en AN9 doorgezet om gebruikt te worden.

Met een Vref stel je de resolutie van de ADC in, hiervoor zijn een aantal opties:

* Internal Vref 4.1V
* Internal Vref 2.0V
* External Vref max 5V

De volgende registers moeten worden ingesteld voor deze opdracht om de ADC werkend te krijgen. Daarnaast moeten ook de registers voor seriële communicatie worden ingesteld.

* ADCON0

Selecteert welke poort gebruikt word en de status van de ADC.

* ADCON1

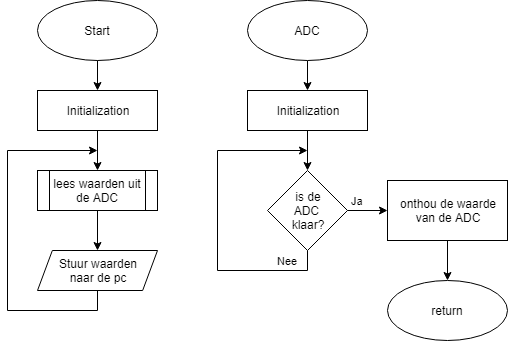
Welke Vref gebruikt word, eventueel negatief kanaal en andere triggers.

* ADCON2

Formaat van het resultaat en met welke snelheid de ADC werkt.

* ANCON0
* ANCON1

## Flowcharts



# Implementatie

1. /\*
2. \*  Embedded Programming 2018
3. \*
4. \*  Opdracht: ADC
5. \*
6. \*  Code by
7. \*      - Zeno Scheltens
8. \*      - Kit Yi Feng
9. \*
10. \*/
12. #include <stdio.h>
13. #include <xc.h>
14. #include "fuses.h"
16. #define \_XTAL\_FREQ 8000000  // X-tal = 8 MHz
18. **int** \_Analog\_Digital\_convertor\_AN1(**void**);
19. **int** \_Analog\_Digital\_convertor\_AN9(**void**);
21. **void** putch (**char** c)
22. {
23. **while**(TXSTA1bits.TRMT == 0);
24. TXREG1 = c;
25. }
27. **void** main() {
29. unsigned **int** value1;
30. unsigned **int** value9;
32. // Ini fase
33. TRISCbits.TRISC6 = 0;   // Tx1 output
34. PIE3bits.RC2IE = 0;     // disable Rx interrupt USART2
35. PIE3bits.TX2IE = 0;     // disable Tx interrupt USART2
36. PIE1bits.RC1IE = 0;     // disable Rx interrupt USART1
37. PIE1bits.TX1IE = 0;     // disable Tx interrupt USART1
38. TXSTA1 = 0xA0;          //
39. RCSTA1 = 0x80;          //
40. BAUDCON1 = 0xC0;        //
41. SPBRG1 = 12;            //
43. //loop om het de ADC uit te lezen en deze waarden weer te geven
44. **while**(1)
45. {
46. value1 = \_Analog\_Digital\_convertor\_AN1();
47. value9 = \_Analog\_Digital\_convertor\_AN9();
48. printf("adcValue AN1 = %d, adcValue AN9 = %d\n\r", value1, value9);
49. }
50. }
52. // functie om AN1 uit te lezen
53. **int** \_Analog\_Digital\_convertor\_AN1(**void**)
54. {
55. ADCON0 = 0b00000111;       //channel AN1[pin3](bit6-2), Start ADCconversion(bit1), ADC on(bit0)
56. ADCON1 = 0b00110000;       //trigger ECCP1(bit7-6), AVref 4.1V(bit5-4), AVss(bit3), Neg Channel00(AVss)(bit2-0)
57. ADCON2 = 0b10110001;       //right justified(bit7), Tad 16(bit5-3), conversion CLK Fosc/8(bit2-0)
58. **while** ( ADCON0bits.nDONE == 1);
59. **return** ADRESH << 8| ADRESL;
60. }
62. // functie om AN9 uit te lezen
63. **int** \_Analog\_Digital\_convertor\_AN9(**void**)
64. {
65. ADCON0 = 0b00100111;       //channel AN1[pin3](bit6-2), Start ADCconversion(bit1), ADC on(bit0)
66. ADCON1 = 0b00110000;       //trigger ECCP1(bit7-6), AVref 4.1V(bit5-4), AVss(bit3), Neg Channel00(AVss)(bit2-0)
67. ADCON2 = 0b10110001;       //right justified(bit7), Tad 16(bit5-3), conversion CLK Fosc/8(bit2-0)
68. **while** ( ADCON0bits.nDONE == 1);
69. **return** ADRESH << 8| ADRESL;
70. }

# Evaluatie

