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

Zonnekrachtmeter

Lekker simpel

Contents

[Analyse 2](#_Toc518303195)

[Ontwerp 3](#_Toc518303196)

[ADC 3](#_Toc518303197)

[Interrupt 4](#_Toc518303198)

[Communicatie 4](#_Toc518303199)

[Flowcharts 5](#_Toc518303200)

[Implementatie 6](#_Toc518303201)

[Evaluatie 8](#_Toc518303202)

[Aanbevelingen 8](#_Toc518303203)

# Analyse

Voor deze opdracht moeten twee waarden door de ADC worden uitgelezen. Eventueel nog worden omgerekend naar een andere eenheid en dan naar de pc worden gestuurd waar de signalen worden uitgelezen met behulp van een seriële monitor. Met behulp van een timer interrupt word er voor gezorgd dat dit niet constant gebeurt maar elke seconde een keer.

Het programma moet voldoen aan de volgende eisen:

* Twee analoge waarden uitlezen en deze omzetten in een digitale waarde
* Deze digitale waarde naar de computer sturen
* Zorgen dat het bovenstaande elke seconde gebeurt

Hoe de ADC moet worden gebruikt is behandeld voor opdracht 5 en hierin is ook het stukje communicatie met de pc herhaald. Opdracht 4 behandelde interrupts. Dit word gebruikt om te zorgen dat het programma elke seconde de waarde uitleest en verstuurd naar de computer.

# Ontwerp

## ADC

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

Voor deze opdracht heb ik de Vref ingesteld op de interne Vref van 4.1 V. hierdoor weet ik met grotere zekerheid wat de Vref is dan als het voltage van de

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.

Eerst word poort AN1 uitgelezen en daarna AN9

* ADCON1

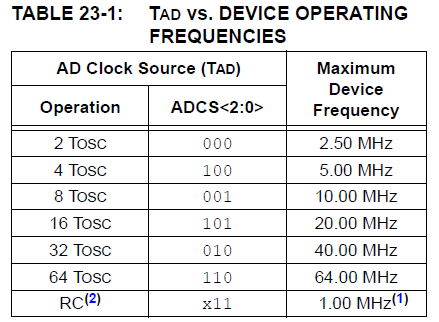
Welke Vref gebruikt word, eventueel negatief kanaal en andere triggers.

* ADCON2

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

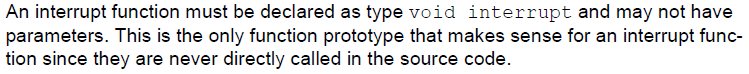
Onderdeel van dit register zijn het instellen van de Tad. In de documentatie van de ADC staat dat voor een 12 bits de tijd minimaal 14 Tad moet zijn, maar tevens niet te lang moet zijn. De instelling van 16 Tad is dus het meest passend.

Voor het instellen van de ADCS heb ik gekeken naar de volgende tabel.



## Interrupt

Voor deze opdracht gebruiken we een software interrupt op timer0. Interrupts moeten in een aparte functie staan. Hoe een interrupt functie genoemd moet worden staat vermeld in de xc8 documentatie. De compiler moet namelijk een functie als interrupt kunnen herkennen om deze op de juiste manier te compilen.

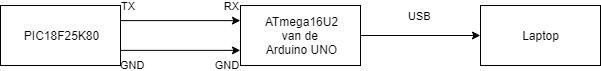


Figuur 1: MPLAB XC8 C Compiler User's Guide page 189

Bij een timer interrupt moet worden ingesteld hoe vaak deze afgaat. Dit gaat met het instellen van hoe lang het duurt voor het register waar de tijd in bijgehouden word vol loopt samen met een offset om in te stellen dat dit precies 1 seconde duurt.

## Communicatie

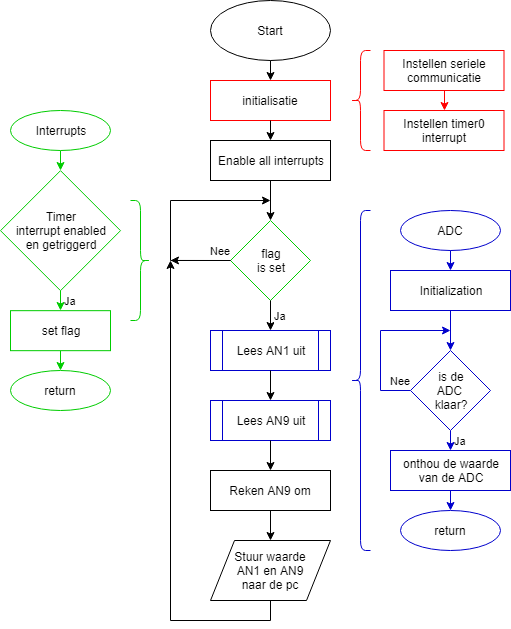
Om de communicatie tot stand te brengen word een Arduino tussen de microcontroller en de PIC geplaatst. Op het Arduino Uno bordje zit een microcontroller die de seriële communcatie om kan zetten naar een USB signaal.



Figuur : Blokschema seriële communicatie met de PIC18F25K80

Voor het instellen van de communicatie worden de volgende registers gebruikt: TXSTA, RCSTA, BAUDCON, SPBRG. Deze zijn voor het instellen van de transmit status, recieve status en welke baud rate word gebruikt.

## Flowcharts



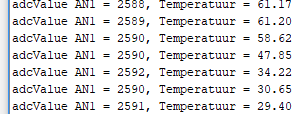
# Implementatie

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. // declare functions
19. **int** \_Analog\_Digital\_convertor\_AN1(**void**);
20. **int** \_Analog\_Digital\_convertor\_AN9(**void**);
22. // declare global variables
23. **char** sendData = 0;
25. // macros
27. **void** putch (**char** c)
28. {
29. **while**(TXSTA1bits.TRMT == 0);
30. TXREG1 = c;
31. }
33. **void** interrupt myIsr (**void**) {
35. // software interrupt
36. // timer0 interrupting every 1 sec
37. **if** (INTCONbits.TMR0IE && INTCONbits.TMR0IF) {       // enables TMR0 overflow interrupt - TMR0 reached overflow?
38. INTCONbits.TMR0IF = 0;                          // reset TMR0, clears the overflow
39. sendData = 1;
40. // offset: 3036
41. TMR0H = 0x5E;                           // 94
42. TMR0L = 0x1C;                           // 28
43. }
44. }
46. **void** main() {
48. unsigned **int** value1;
49. unsigned **int** value9;
50. **int** valueTemperatuur;
52. // Ini fase
54. // seriele communicatie
55. TRISCbits.TRISC6 = 0;   // Tx1 output
56. PIE3bits.RC2IE = 0;     // disable Rx interrupt USART2
57. PIE3bits.TX2IE = 0;     // disable Tx interrupt USART2
58. PIE1bits.RC1IE = 0;     // disable Rx interrupt USART1
59. PIE1bits.TX1IE = 0;     // disable Tx interrupt USART1
60. TXSTA1 = 0xA0;          //
61. RCSTA1 = 0x80;          //
62. BAUDCON1 = 0xC0;        //
63. SPBRG1 = 12;            //
65. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
66. // Setup Timer Interrupt ;
67. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
68. T0CON = 0b10000100;             // enables timer 1 as 16 bit with a /32 prescale
69. INTCON2bits.TMR0IP = 1;         // overflow interrupt priority
70. INTCONbits.TMR0IF = 0;          // clears the timer overflow
71. INTCONbits.TMR0IE = 1;          // enables the timer interrupt
73. ei();
75. //loop om het de ADC uit te lezen en deze waarden weer te geven
76. **while**(sendData == 1)
77. {
78. value1 = \_Analog\_Digital\_convertor\_AN1();
79. value9 = \_Analog\_Digital\_convertor\_AN9();
80. valueTemperatuur = ((value9 - 820)\*10) / 4;
81. printf("adcValue AN1 = %d, Temperatuur = %d.%02d\n\r", value1, (valueTemperatuur/100), (valueTemperatuur%100));
82. sendData = 0;
83. }
84. }
86. // functie om AN1 uit te lezen
87. **int** \_Analog\_Digital\_convertor\_AN1(**void**)
88. {
89. ADCON0 = 0b00000111;       //channel AN1[pin3](bit6-2), Start ADCconversion(bit1), ADC on(bit0)
90. ADCON1 = 0b00110000;       //trigger ECCP1(bit7-6), AVref 4.1V(bit5-4), AVss(bit3), Neg Channel00(AVss)(bit2-0)
91. ADCON2 = 0b10110001;       //right justified(bit7), Tad 16(bit5-3), conversion CLK Fosc/8(bit2-0)
92. **while** ( ADCON0bits.nDONE == 1);
93. **return** ADRESH << 8| ADRESL;
94. }
96. // functie om AN9 uit te lezen
97. **int** \_Analog\_Digital\_convertor\_AN9(**void**)
98. {
99. ADCON0 = 0b00100111;       //channel AN9[pin\_](bit6-2), Start ADCconversion(bit1), ADC on(bit0)
100. ADCON1 = 0b00110000;       //trigger ECCP1(bit7-6), AVref 4.1V(bit5-4), AVss(bit3), Neg Channel00(AVss)(bit2-0)
101. ADCON2 = 0b10110001;       //right justified(bit7), Tad 16(bit5-3), conversion CLK Fosc/8(bit2-0)
102. **while** ( ADCON0bits.nDONE == 1);
103. **return** ADRESH << 8| ADRESL;
104. }

# Evaluatie

Het programma moet aan de volgende eisen voldoen:

* Twee analoge waarden uitlezen en deze omzetten in een digitale waarde
* Deze digitale waarde naar de computer sturen
* Zorgen dat het bovenstaande elke seconde gebeurt



Bovenstaande is een screen capture van de seriële monitor. Zo als te zien is de waarde variabel en heeft het veranderen van de ene waarde geen invloed op de andere waarde. Het programma voldoet dus aan de bovenste twee eisen.

Er zit een tijdsinterval tussen de waarden, echter heb ik niet getimed hoe veel. Het is dus niet bewezen dat dit precies 1 seconde is.

# Aanbevelingen

Naast het weergeven van de waarden uit de ADC is het ook aan te raden om een timestamp mee te sturen zodat te zien is dat de waarde ook elke seconde word uitgelezen en verstuurd.