|  |  |
| --- | --- |
| MECHATRONICS INTL-PROGRAM | ASSIGNMENT COVERSHEET |

|  |  |  |
| --- | --- | --- |
| STUDENT DETAILS | | |
| STUDENT NAME | 于航 | |
| STUDENT NAME (PINYIN) | YU HANG | |
| STUDENT ID | 17222054 | |
| CLASS NUMBER | 1702 | |
| ASSIGNMENT DETAILS | | |
| COURSE NAME |  | |
| COURSE NUMBER |  | |
| ASSIGNMENT NAME |  | |
| LECTURER/TUTOR |  | |
| SUBMISSION DETAILS | | |
| DATE DUE | 06/02/2020 | |
| SUBMITTION DATE | 06/01/2020 | |
| STAFF ONLY | | |
| RECEIVED BY |  | DATE/TIME |
| MARKER |  | GRADE |

Declaration and Acknowledgement

By submitting this, I declare that:

1. This assignment meets all the requirements of the subject as student in the relevant subject outline, which I have read.
2. (a) This assessment item is entirely my own work, except where I have included fully-documented references to the work of others.

(b) The material contained in this assessment item has not previously been submitted for assessment.

1. I acknowledge that:

(a) The marker of this assessment item may, for the purpose of assessing this assignment, reproduce this assignment and provide a copy to another member of academic staff.

(b) If required to do so, I will provide an electronic copy of this assessment item to the marker.

1. I am aware that late submission without an authorized extension from the subject coordinator may incur a penalty.

Please note: Assignments are not to be submitted by fax and must be submitted during Lectures/tutorials/laboratories or directly to the academic. Only under special circumstances will the Administrative Staff collect assignments

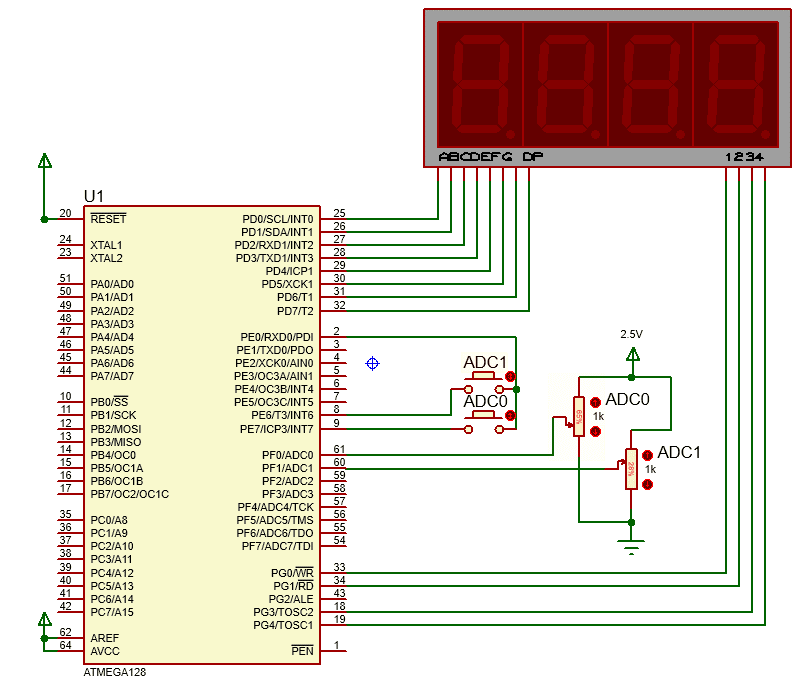
Experiment 3:ADC

1. Object of experiment
   1. Be able to recognize the ADC port of ATmega128, and explain the principle and structure of ADC
   2. Be able to design, write and debug the program of driving LED lamp with ATmega128 analog to digital conversion module.
2. Experiment content
   1. basic experiment

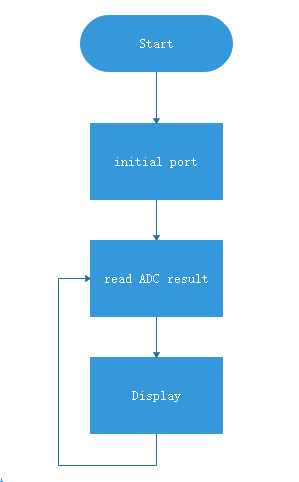
Adjust the input voltage through potentiometer. After AD0 input, carry out conversion, and display the result with digital tube·. ADC is initialized using the reference AVCC, channel 0, 8-prescale. The AD conversion result is read by query. After the ADC conversion, the ADIF and ADEN marks should be cleared.

* 1. Extended experiment
     1. Adjust the input voltage through potentiometer. After AD0 input, carry out conversion, and trans the result into the input voltage which is displayed the by LED.
     2. Two potentiometers are used to adjust the input voltage. After the input of AD0 and AD1, the input voltage of AD0 and AD1 is converted to AD, and the input voltage of AD0 and AD1 is displayed through the switch control digital tube.

1. Experimental schematic diagram



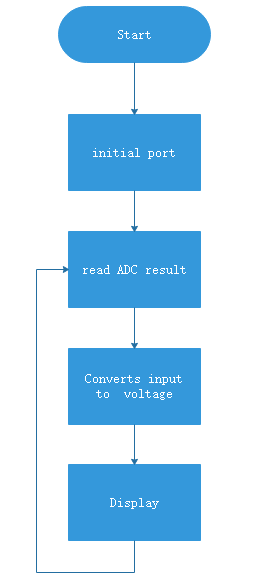
1. Code and flow diagram
   1. basic experiment I
      1. flow diagram



* + 1. code

|  |
| --- |
| 1. #include<iom128v.h> 2. #include<macros.h> 4. #define SEG1\_ON   PORTG |= BIT(4); 5. #define SEG1\_OFF  PORTG &= ~BIT(4); 7. #define SEG2\_ON   PORTG |= BIT(3); 8. #define SEG2\_OFF  PORTG &= ~BIT(3); 10. #define SEG3\_ON   PORTG |= BIT(1); 11. #define SEG3\_OFF  PORTG &= ~BIT(1); 13. #define SEG4\_ON   PORTG |= BIT(0); 14. #define SEG4\_OFF  PORTG &= ~BIT(0); 16. unsigned **char** tab[10]={0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,0x80, 0x90}; 17. unsigned **char** disp[4]; 18. unsigned **int** voltage; 20. //to get 1 us delay in 8MHz crystal 21. **void** delay\_us(unsigned **int** microsecond){ 22. **do**{ 23. microsecond--; 24. }**while** (microsecond>1); 25. } 27. // to get 1 ms delay 28. **void** delay\_ms(unsigned **int** millisecond){ 29. **while** (millisecond--){ 30. delay\_us(999); 31. } 32. } 34. **void** init\_port(**void**){ 35. DDRF  = 0X00; 36. PORTF = 0X00; 37. DDRD  = 0XFF; 38. PORTD = 0XFF; 39. PORTG = 0x1B; 40. DDRG  = 0x1B; 41. } 43. unsigned **int** Read\_ADC(unsigned **char** channel){ 44. unsigned **int** rhigh,rlow,result; 45. ADMUX=channel; 46. ADCSRA|=0XC3;//Enable ADC, start conversion,8-prescale 47. **while**(!(ADCSRA&(BIT(ADIF))));//Wait for the conversion to complete 48. ADCSRA&=~(BIT(ADIF)|BIT(ADEN));//cleat ADEN and ADIF 50. rlow =(unsigned **int**)ADCL;//read low resistor 51. rhigh=(unsigned **int**)ADCH;//read low resistor 52. result =(rhigh<<8)+rlow;//get final result 53. **return** result; 54. } 56. //Dynamic digital tube display 57. **void** display(**void**){ 58. PORTD=tab[voltage/1000]; 59. SEG4\_ON; 60. delay\_us(500); 61. SEG4\_OFF; 63. PORTD=tab[(voltage%1000)/100]; 64. SEG3\_ON; 65. delay\_us(500); 66. SEG3\_OFF; 68. PORTD=tab[(voltage%100)/10]; 69. SEG2\_ON; 70. delay\_us(500); 71. SEG2\_OFF; 73. PORTD=tab[(voltage%10)]; 74. SEG1\_ON; 75. delay\_us(500); 76. SEG1\_OFF; 77. } 79. **void** main(**void**){ 80. init\_port(); 81. **while**(1){ 82. voltage=Read\_ADC(0x00); 83. display(); 84. } 85. } |

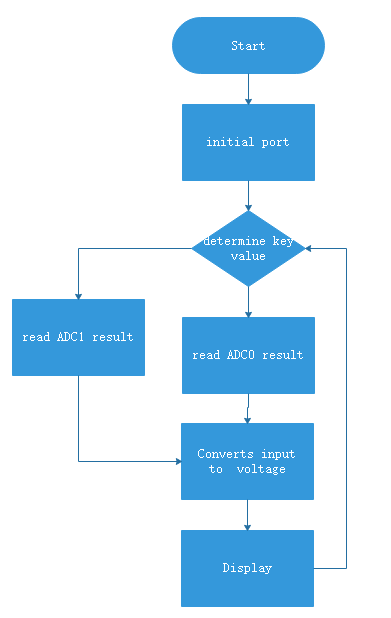
* 1. Extended experiment I
     1. flow diagram



* + 1. code

|  |
| --- |
| 1. #include<iom128v.h> 2. #include<macros.h> 4. #define SEG1\_ON   PORTG |= BIT(4); 5. #define SEG1\_OFF  PORTG &= ~BIT(4); 7. #define SEG2\_ON   PORTG |= BIT(3); 8. #define SEG2\_OFF  PORTG &= ~BIT(3); 10. #define SEG3\_ON   PORTG |= BIT(1); 11. #define SEG3\_OFF  PORTG &= ~BIT(1); 13. #define SEG4\_ON   PORTG |= BIT(0); 14. #define SEG4\_OFF  PORTG &= ~BIT(0); 16. unsigned **char** tab[10]={0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,0x80, 0x90}; 17. unsigned **char** key\_value=0; 18. unsigned **int** voltage; 20. //to get 1 us delay in 8MHz crystal 21. **void** delay\_us(unsigned **int** microsecond){ 22. **do**{ 23. microsecond--; 24. }**while** (microsecond>1); 25. } 27. // to get 1 ms delay 28. **void** delay\_ms(unsigned **int** millisecond){ 29. **while** (millisecond--){ 30. delay\_us(999); 31. } 32. } 34. **void** init\_port(**void**){ 35. DDRF  = 0X00; 36. PORTF = 0X00; 37. DDRD  = 0XFF; 38. PORTD = 0XFF; 39. PORTG = 0x1B; 40. DDRG  = 0x1B; 41. DDRE  = 0x01; 42. PORTE = 0xc0; 43. } 44. unsigned **int** Read\_ADC(unsigned **char** channel){ 45. unsigned **int** rhigh,rlow,result; 46. ADMUX=channel; 47. ADCSRA|=0XC3;//Enable ADC, start conversion,8-prescale 48. **while**(!(ADCSRA&(BIT(ADIF))));//Wait for the conversion to complete 49. ADCSRA&=~(BIT(ADIF)|BIT(ADEN));//cleat ADEN and ADIF 51. rlow =(unsigned **int**)ADCL;//read low resistor 52. rhigh=(unsigned **int**)ADCH;//read low resistor 53. result =(rhigh<<8)+rlow;//get final result 54. **return** result; 55. } 57. //Dynamic digital tube display 58. **void** display(**void**){ 59. PORTD=tab[voltage/1000]; 60. SEG4\_ON; 61. delay\_us(500); 62. SEG4\_OFF; 64. PORTD=tab[(voltage%1000)/100]; 65. SEG3\_ON; 66. delay\_us(500); 67. SEG3\_OFF; 69. PORTD=tab[(voltage%100)/10]; 70. SEG2\_ON; 71. delay\_us(500); 72. SEG2\_OFF; 74. PORTD=tab[(voltage%10)]; 75. SEG1\_ON; 76. delay\_us(500); 77. SEG1\_OFF; 78. } 80. **void** main(**void**){ 81. init\_port(); 82. **while**(1){ 83. voltage=(unsigned **int**)(Read\_ADC(0x00)\*2.44);**break**; 84. display(); 85. } 86. } |

* 1. extended experiment II
     1. flow diagram



* + 1. code

|  |
| --- |
| 1. #include<iom128v.h> 2. #include<macros.h> 4. #define SEG1\_ON   PORTG |= BIT(4); 5. #define SEG1\_OFF  PORTG &= ~BIT(4); 7. #define SEG2\_ON   PORTG |= BIT(3); 8. #define SEG2\_OFF  PORTG &= ~BIT(3); 10. #define SEG3\_ON   PORTG |= BIT(1); 11. #define SEG3\_OFF  PORTG &= ~BIT(1); 13. #define SEG4\_ON   PORTG |= BIT(0); 14. #define SEG4\_OFF  PORTG &= ~BIT(0); 16. unsigned **char** tab[10]={0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,0x80, 0x90}; 17. unsigned **char** key\_value=0; 18. unsigned **int** voltage; 20. //to get 1 us delay in 8MHz crystal 21. **void** delay\_us(unsigned **int** microsecond){ 22. **do**{ 23. microsecond--; 24. }**while** (microsecond>1); 25. } 27. // to get 1 ms delay 28. **void** delay\_ms(unsigned **int** millisecond){ 29. **while** (millisecond--){ 30. delay\_us(999); 31. } 32. } 34. **void** init\_port(**void**){ 35. DDRF  = 0X00; 36. PORTF = 0X00; 37. DDRD  = 0XFF; 38. PORTD = 0XFF; 39. PORTG = 0x1B; 40. DDRG  = 0x1B; 41. DDRE  = 0x01; 42. PORTE = 0xc0; 43. } 45. **void** key(**void**){ 46. **if**(PINE!=0xC0){ 47. **if** (PINE==0X80) key\_value = 1; 48. **else** **if** (PINE==0X40)  key\_value = 0; 49. } 50. } 52. unsigned **int** Read\_ADC(unsigned **char** channel){ 53. unsigned **int** rhigh,rlow,result; 54. ADMUX=channel; 55. ADCSRA|=0XC3;//Enable ADC, start conversion,8-prescale 56. **while**(!(ADCSRA&(BIT(ADIF))));//Wait for the conversion to complete 57. ADCSRA&=~(BIT(ADIF)|BIT(ADEN));//cleat ADEN and ADIF 59. rlow =(unsigned **int**)ADCL;//read low resistor 60. rhigh=(unsigned **int**)ADCH;//read low resistor 61. result =(rhigh<<8)+rlow;//get final result 62. **return** result; 63. } 65. //Dynamic digital tube display 66. **void** display(**void**){ 67. PORTD=tab[voltage/1000]; 68. SEG4\_ON; 69. delay\_us(500); 70. SEG4\_OFF; 72. PORTD=tab[(voltage%1000)/100]; 73. SEG3\_ON; 74. delay\_us(500); 75. SEG3\_OFF; 77. PORTD=tab[(voltage%100)/10]; 78. SEG2\_ON; 79. delay\_us(500); 80. SEG2\_OFF; 82. PORTD=tab[(voltage%10)]; 83. SEG1\_ON; 84. delay\_us(500); 85. SEG1\_OFF; 86. } 88. **void** main(**void**){ 89. init\_port(); 90. **while**(1){ 91. key(); 92. **switch**(key\_value){ 93. **case** 0:voltage=(unsigned **int**)(Read\_ADC(0x00)\*2.44);**break**; 94. **case** 1:voltage=(unsigned **int**)(Read\_ADC(0x01)\*2.44);**break**; 95. **default**:**break**; 96. } 97. display(); 98. } 99. } |

1. Result
   1. base experiment

The voltage collected by ADC0 is displayed by the digital tube, and the indicator number of the rotating potentiometer will change. The display range is 0-1023

* 1. Extended experiment I

The voltage collected by ADC0 is displayed by the digital tube, and the indicator number of the rotating potentiometer will change. The display range is 0-2496mV

* 1. Extended experiment II

The voltage collected by ADC0 is displayed by the digital tube, and the indicator number of the rotating potentiometer will change. The display range is 0-2496mV

1. Reflection
   1. the procession to initial ADC
      1. Setting ADMUX the reference voltage and input channel and gain.
      2. Setting ADCSR to control the state of ADC which include enable and start the ADC, prescale and ADC interrupt etc.
   2. Read ADC result
      1. Read ADCH and ADCL respectively, then the result of ADC is ADCH\*256+ADCL which is stored in binary.
2. Experience

In this experience, I learned how to set ADC register, and in ATmega128 The ADC converts the input analog voltage into a 10-bit digital quantity by successive approximations. Also when we initial ADC, the corresponding pin should be set as input, otherwise an error will occur.