AVR Freaks

DESIGN NOTE #033

AUTHOR: VASILIOS THEODOROU

KEYWORDS: ADC, DAC, PULL-UP

DAC Utilizing ADC

Introduction

This design note describes a way to produce an analog voltage which is proportional to an 10-bit digital word. In other words, the design note describes how we can create a Digital-to-Analog Converter by utilizing the Analog-to-Digital Converter as well as some other hardware peripherals which are already present in several members of the AVR® family.

Overview

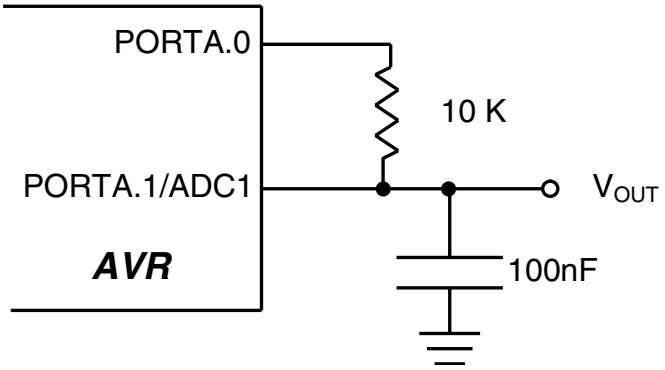
As it is commonly known the AVR microcontrollers does not include any kind of integrated DAC. However there are many ways to build DACs using vital resources of the micro, i.e., a free output port or a PWM channel on the accompanying timer. Obviously the DAC should be attended of a kind of software which is absolutely necessary for the proper operation.

Clearly DAC can be implemented only in those devices which have an internal ADC (ATtiny15, AT90S4433, AT90S8535…). For the operation of the DAC only an external RC network is required. Typical values for the components of the net are 10K and 100 nF for the resistor and the capacitor respectively. The unconnected resistor lead must be connected to an unused I/O line (in our code PORTA.0). The common lead of the resistor and the capacitor which is the VOUT of the DAC must be connected simultaneously to an ADC input (in our code PORTA.1/ADC1). The remaining capacitor lead goes to ground.

The software operates the ADC in Free Running mode. Thus the valuable hardware resources such as Timer, PWM and the comparator are not involved in the DAC operation. Only the ADC interrupt complete vector is needed which in many cases is not a huge trade-off. While the Analog-to-Digital conversion is complete the corresponding interrupt is invoked and the interrupt service routine is executed. In that routine the 10-bit result is compared to the desired output voltage. In case that the output voltage is smaller than the desired the internal pull-up of PORTA.0 pin is activated and the capacitor is charged via the resistor. The pull-up is deactivated in case the desired voltage is achieved or the output voltage is bigger than those expected. In the second case the capacitor is discharged gradually via the resistor.

The output ripple can be trimmed to acceptable levels for a specific application. However note that the above described DAC is not ideal. Moreover the ripple varies in relation to the desired output voltage due to the non linear nature of the capacitor charge. At any case the ripple can be eliminated by setting the ADC prescaler to an appropriate value.

Figure 1. RC Network



Code

```
;---------------------------------------------------------------
; Author:Robotechnics Laboratories/Vasilios Theodorou
; Project name:DAC utilizing ADC
; Last modification date: 14/5/2002
;---------------------------------------------------------------
.include "c:\microcon\avrtools\appnotes\8535def.inc"
;---------------------------------------------------------------
.def    temp      =r16
.def    temp1     =r17
.def    temp2     =r18
.equ    DAC_val   =750
;---------------------------------------------------------------
.cseg
.org    0x00
        rjmp      start
.org    ADCCaddr
        rjmp      ADC_INT
;---------------------------------------------------------------
start:                              ;Initialize Stack Pointer
        ldi       temp,low(RAMEND)
        out       SPL,temp
        ldi       temp,high(RAMEND)
        out       SPH,temp

        ldi       temp,0
        out       porta,temp
        ldi       temp,0b00000000   ;Porta as input
        out       ddra,temp

                                    ;ADC initialization
        ldi       temp,0b00000001
        out       ADMUX,temp
        ldi       temp,0b10101000
        out       ADCSR,temp
```

```
        sei                         ; Global interrupt enable
        sbi       ADCSR,ADSC

main:
        rjmp      main
;-----------------------------------------------------------------
                                    ;ADC complete interrupt service routine
ADC_INT:
        in        temp1,ADCL
        in        temp2,ADCH

        subi      temp1,low(DAC_val)
        sbci      temp2,high(DAC_val)
        brpl      ad_lo

        sbi       PORTA,0
        rjmp      ad_end
ad_lo:
        cbi       PORTA,0
ad_end:
        reti
;-------------------------------
```