8. ADC Assembly Code Example

8.1 Full Code

Here is an example of AVR Assembly code to read the ADC from the ADC0 pin using the internal 2.56V reference and a CLK/128 prescaler:

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;------------------------
.global main

main:
    LDI R20, 0xFF
    OUT DDRD, R20       ; Set Port D as output (ADC result low byte)
    OUT DDRB, R20       ; Set Port B as output (ADC result high byte)
    SBI DDRC, 0         ; Set pin PC0 as input for ADC0

    ;-- ADC Initialization --
    LDI R20, 0xC0       ; REFS1:REFS0 = 11 → Internal 2.56V
                        ; ADLAR = 0 → Right-justified
                        ; MUX4:MUX0 = 00000 → ADC0
    STS ADMUX, R20

    LDI R20, 0x87       ; ADEN = 1 → Enable ADC
                        ; ADPS2:ADPS0 = 111 → Prescaler CLK/128
    STS ADCSRA, R20

;-- ADC Reading Loop --
read_ADC:
    LDI R20, 0xC7       ; Set ADSC = 1 to start conversion
    STS ADCSRA, R20

wait_ADC:
```

```
    LDS R21, ADCSRA     ; Read ADCSRA status register
    SBRS R21, 4         ; Skip jump if ADIF (bit 4) = 1 (conversion complete)
    RJMP wait_ADC       ; Wait loop until ADIF is set

    ;-- Reset ADIF flag --
    LDI R17, 0xD7       ; Set ADIF = 1 so the controller can reset the flag
    STS ADCSRA, R17

    ;-- Read conversion result --
    LDS R18, ADCL       ; Read low-byte from ADCL (MUST read first)
    LDS R19, ADCH       ; Read high-byte from ADCH

    ;-- Output result --
    OUT PORTD, R18      ; Send low-byte to Port D
    OUT PORTB, R19      ; Send high-byte to Port B

    RJMP read_ADC       ; Repeat reading
```

8.2 Code Explanation

Initialization Section:

```
LDI R20, 0xC0
STS ADMUX, R20
```

- `0xC0` in binary = `1100 0000`
- REFS1=1, REFS0=1 → Internal 2.56V reference voltage
- ADLAR=0 → Right-justified output
- MUX4:MUX0 = 00000 → Reading pin ADC0 (A0)
- This part runs only **once** during initialization.

```
LDI R20, 0x87
STS ADCSRA, R20
```

- `0x87` in binary = `1000 0111`
- ADEN=1 → ADC is enabled
- ADPS2:ADPS0 = 111 → Prescaler CLK/128 (125 kHz at 16 MHz)

Reading Section (Loop):

```
LDI R20, 0xC7
STS ADCSRA, R20
```

- `0xC7` in binary = `1100 0111`
- ADEN=1, ADSC=1 → Start one conversion cycle
- ADPS remains the same (CLK/128)

```
wait_ADC:
    LDS R21, ADCSRA
    SBRS R21, 4
    RJMP wait_ADC
```

- Polling loop: continuously reads ADCSRA and checks bit 4 (ADIF)
- `SBRS` = Skip if Bit in Register Set → if ADIF=1 (conversion complete), skip RJMP
- As long as ADIF=0 (conversion not complete), continue looping

```
LDI R17, 0xD7
STS ADCSRA, R17
```

- Resets the ADIF flag by writing a 1 to the ADIF bit
- This is necessary so the next conversion can be detected

```
LDS R18, ADCL
LDS R19, ADCH
OUT PORTD, R18
OUT PORTB, R19
```

- Read ADCL first (mandatory), then ADCH
- Send the results to Port D (low-byte) and Port B (high-byte)
- Since it is right-justified: ADCH only contains 2 bits (bits 9 and 8)

Revision #2
Created 2026-04-12 06:55:55 UTC by DS
Updated 2026-04-14 00:44:42 UTC by DS