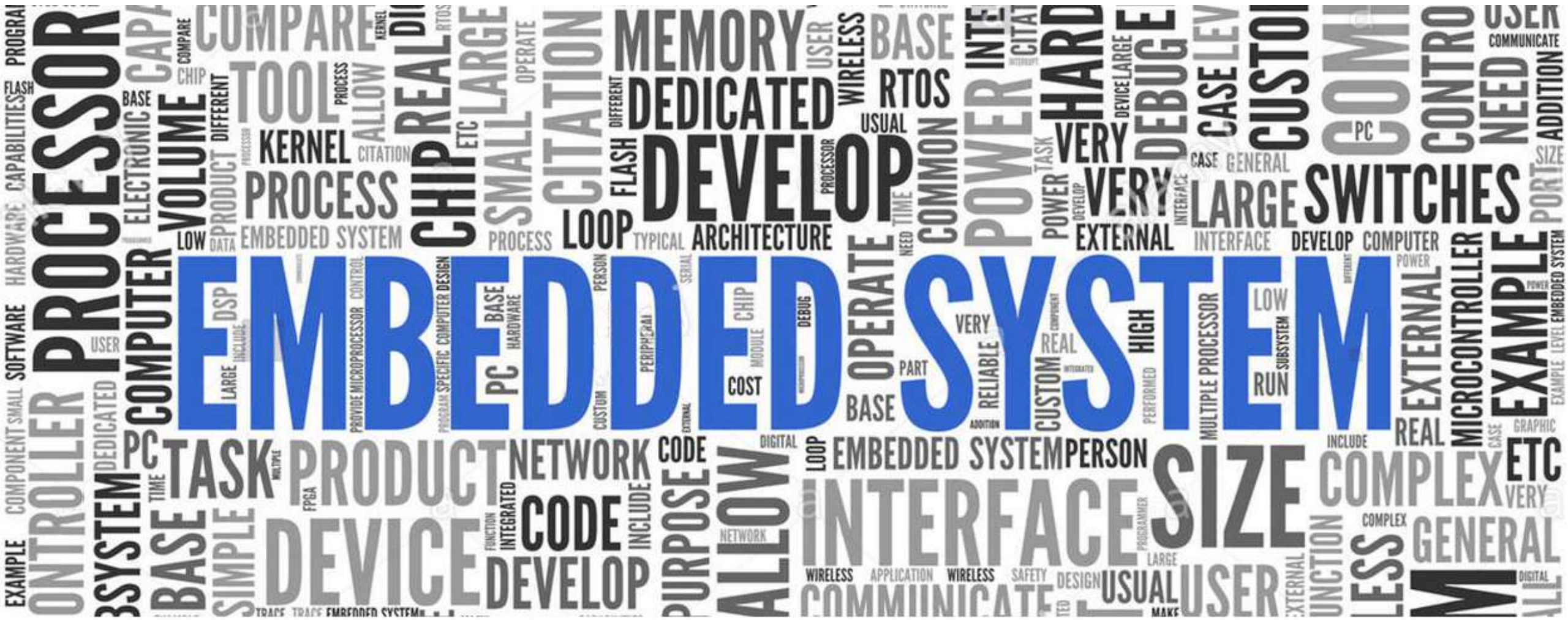


Analog Inputs of ATMEGA328P

Eng. R. Saahith Ahamed

Dept. of Electrical & Information Engineering

Faculty of Engineering, UoR

Learning Objectives

LO-3: Use microcontrollers to implement basic functions of typical embedded systems in Assembly and C languages.

Using analog inputs with ATMEGA 328P

At the end of this activity, you will be able to implement analog to digital converter on ATMEGA 328P microcontroller

You need to have **the data sheet for** ATMEGA 328P microcontroller in order to complete this exercise

Instructions to Complete the Workshop Activity

Workshops are group activities with three members for each group.

Every member of the group should have a copy of the activity and should be able to demonstrate the results to the instructor.

Evaluation deadlines will be given after each workshop session. Every student/group has to demonstrate their work and submit the required documents/simulation files to the LMS depending on the instructions give during the workshop.

It is your responsibility to get your work demonstrations evaluated by the instructor within one week from the date of workshop delivery.

A/D converter in Atmega 328P

- Features a 10-bit successive approximation ADC.
- Connected to an 8-channel analog multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port A.
- The minimum value represents GND.
- The maximum value can be customised to the voltage on the AREF/AVcc or internal 1.1V depending on the settings.
- 6 analog input pins in PORTA

A/D converter in Atmega 328P

A typical A to D converter is illustrated in Figure 1

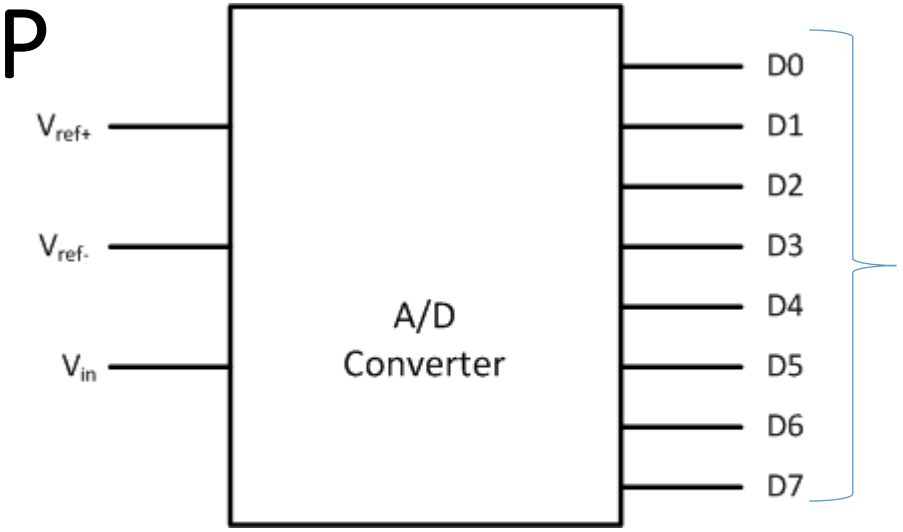


Figure 1

Number of output bits, D, corresponds to the number of quantization levels, N.

$N = 2^D$ and the number of quantization intervals is $Q = N - 1$.

$ADC\ output = Q * (V_{in} - V_{ref-})/(V_{ref+} - V_{ref-})$

In some cases Vref- may not be present. They consider the Vref- to be the ground voltage.

A/D converter in Atmega 328P

ADC output = Q * Vin / Vref+

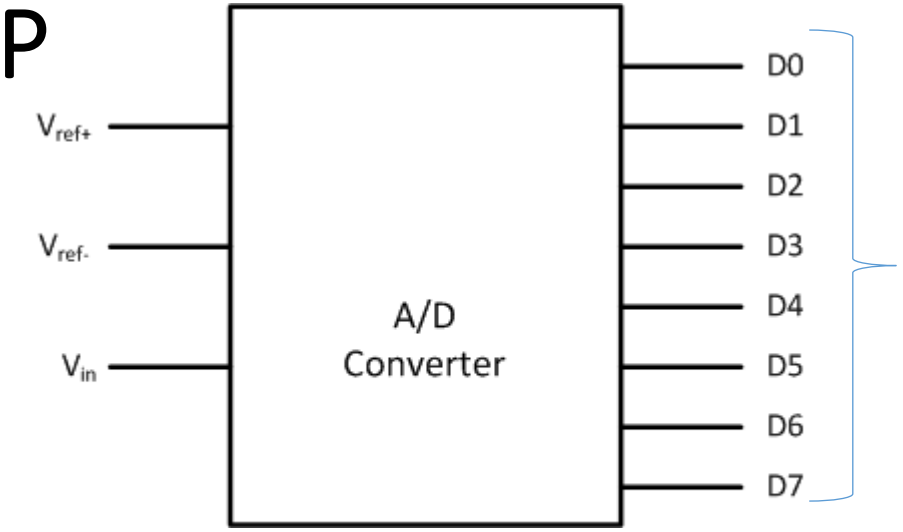


Figure 1

In some cases Vref- may not be present. They consider the Vref- to be the ground voltage.

3. Settings Required for A/D Converter

Name: ADMUX
Offset: 0x7C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

Figure 3.1

REFS[1:0]	Voltage Reference Selection
00	AREF, Internal V_{ref} turned off
01	AV_{CC} with external capacitor at AREF pin
10	Reserved
11	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Set 00 at REFS1 : REFS0 in order to enable AREF external reference voltage

ADLAR bit is to be set if you are to left align the result

MUX[3:0]	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5

Select channel 0 by dressing MUX3:MUX0 to 0000

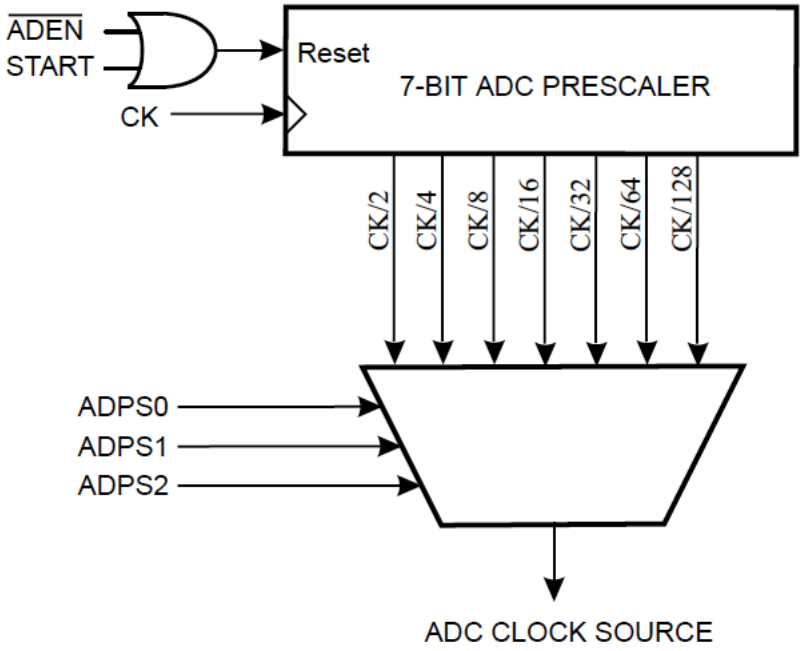
Settings Required for A/D Converter

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
ccess	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Prescaling and Conversion Timing

Figure 28-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

Prescalar	Frequency
2	8 Mhz
4	4 Mhz
8	2 Mhz
16	1 Mhz
32	500 khz
64	250 khz
128	125 khz

ADPS[2:0]	Division Factor
000	2
001	2
010	4
011	8
100	16
101	32
110	64
111	128

125 khz is between 50 and 200 khz.

Pre-scalar should be 128

Settings Required for A/D Converter

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
ccess	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Enable A/D converter when Set

In Single Conversion mode, write this bit to one to start each conversion

A/D converter interrupt enable

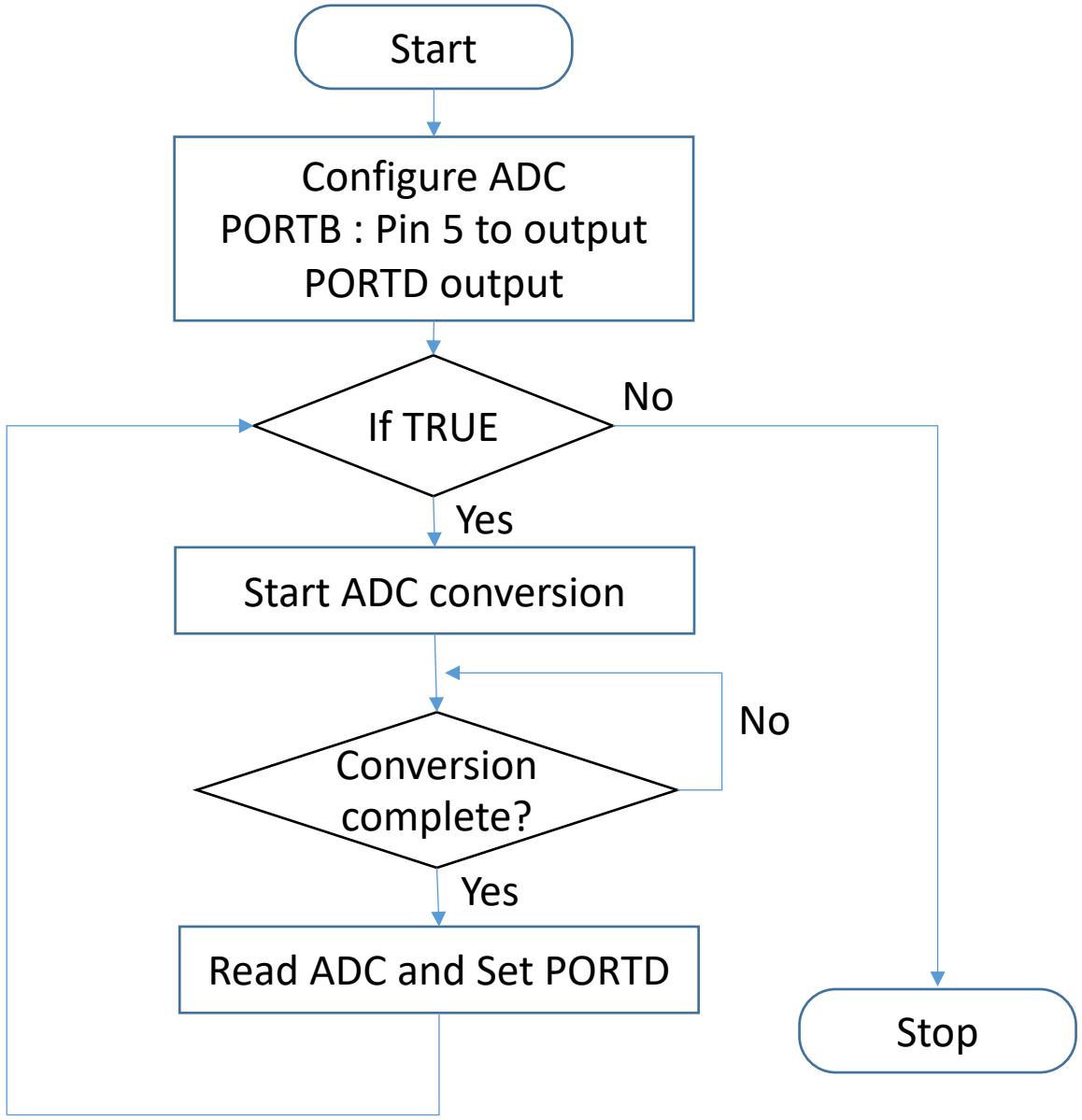
Auto Triggering of ADC is enabled when set to 1.
ADC start a conversion on a positive edge of the selected trigger signal.
The trigger source is selected by ADTS in ADCSRB.

A/D converter interrupt flag

Look at the data sheet for ADCSRB

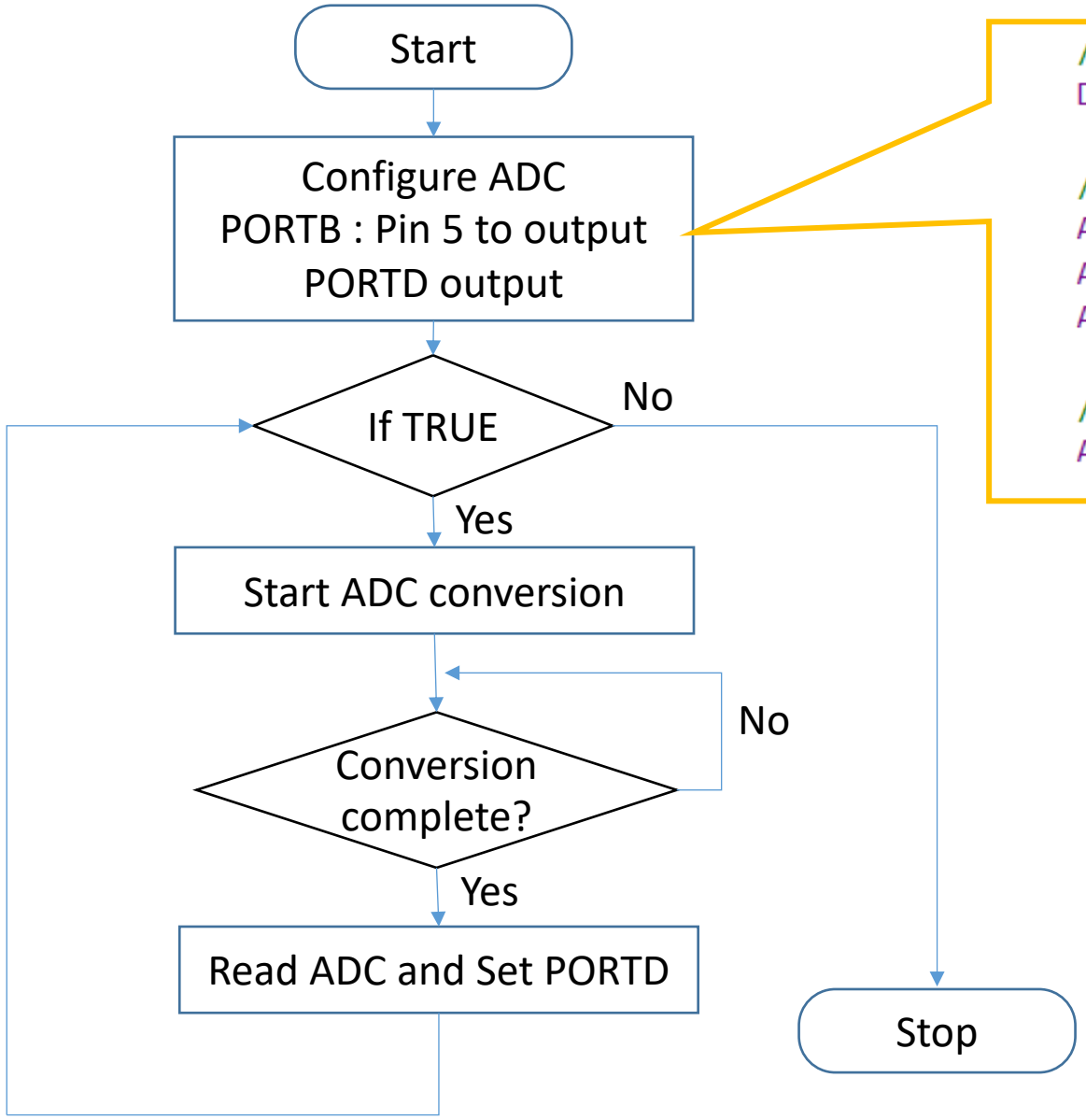
Figure 3.2

Flow Chart for A/D Converter Program



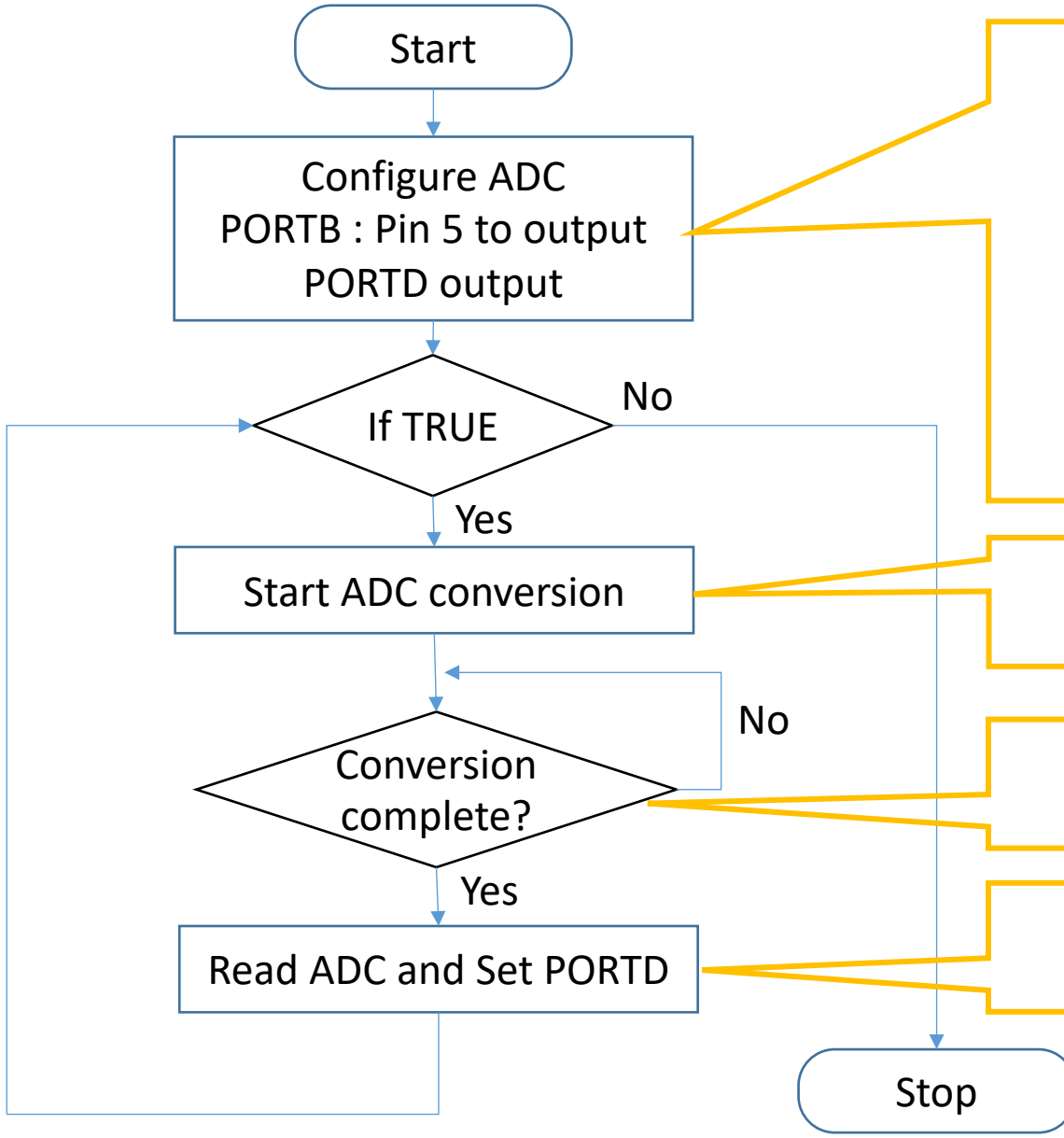
Single Conversion Mode

Flow Chart for A/D Converter Program




```
//Setting PORTD as output
DDRD = 0B11111111;

//Enable ADC
ADCSRA |= (1<<ADEN) ;  // Enable ADC
ADCSRA |= 0B00000111; // Selecting Pre-scalar value 128
ADMUX  &= 0B00111111; // Enable AREF,

// Make Four channel bits to 0 and then add channel number
ADMUX = (ADMUX & 0B11110000) | CHANNEL;
```

Flow Chart for A/D Converter Program




```
//Setting PORTD as output
DDRD = 0B11111111;

//Enable ADC
ADCSRA |= (1<<ADEN) ;  // Enable ADC
ADCSRA |= 0B00000111; // Selecting Pre-scalar value 128
ADMUX  &= 0B00111111; // Enable AREF,

// Make Four channel bits to 0 and then add channel number
ADMUX = (ADMUX & 0B11110000) | CHANNEL;
```

```
// Start Conversion
ADCSRA |= 0B01000000;
```

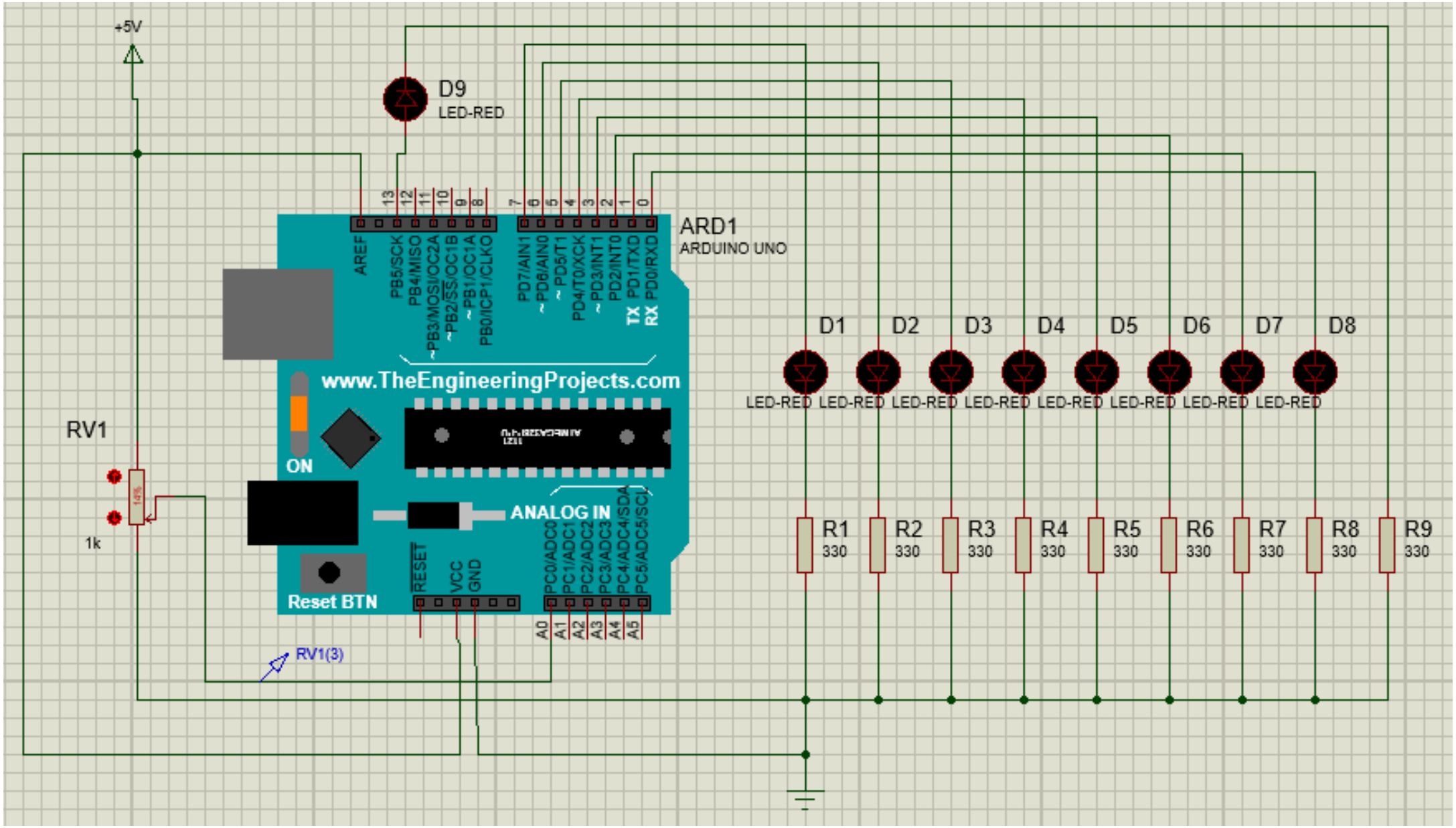
```
// Wait until conversion finishes
while (ADCSRA & (1<<ADSC));
```

```
// Set PORTD to the value of ADC
PORTD = ADC;
```

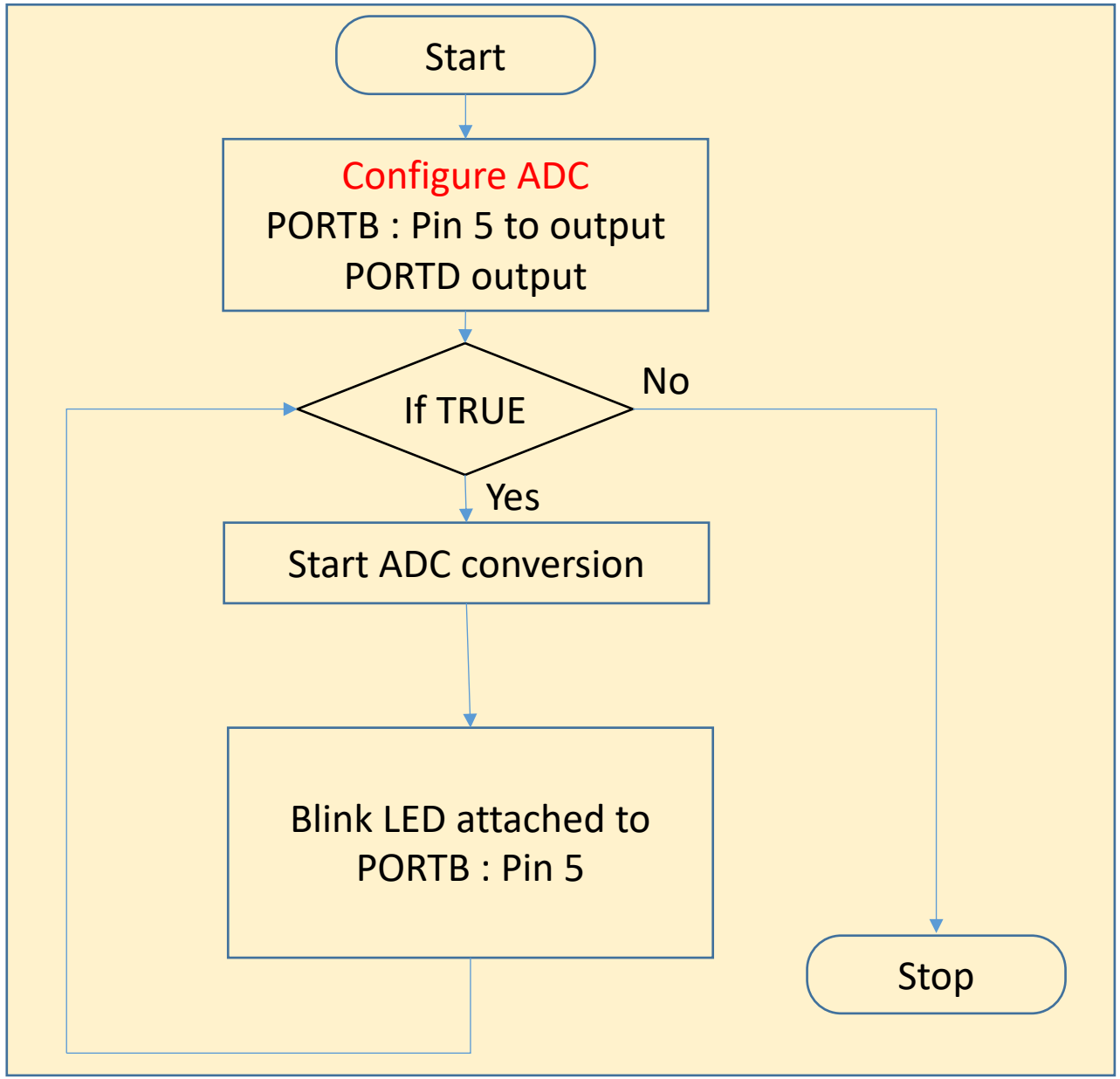
Flow Chart for A/D Converter Program

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#define CHANNEL 0B00000000

int main(void)
{
    //Setting PORTD as output
    DDRD = 0B11111111;

    //Enable ADC
    ADCSRA |= (1<<ADEN) ;  // Enable ADC
    ADCSRA |= 0B00000111; // Selecting Pre-scalar value 128
    ADMUX  &= 0B00111111; // Enable AREF,

    // Make Four channel bits to 0 and then add channel number
    ADMUX = (ADMUX & 0B11110000) | CHANNEL;

    while (1)
    {   // Start Conversion
        ADCSRA |= 0B01000000;
        // Wait until conversion finishes
        while (ADCSRA & (1<<ADSC));
        // Set PORTD to the value of ADC
        PORTD = ADC;
        _delay_ms(500);
    }
}
```

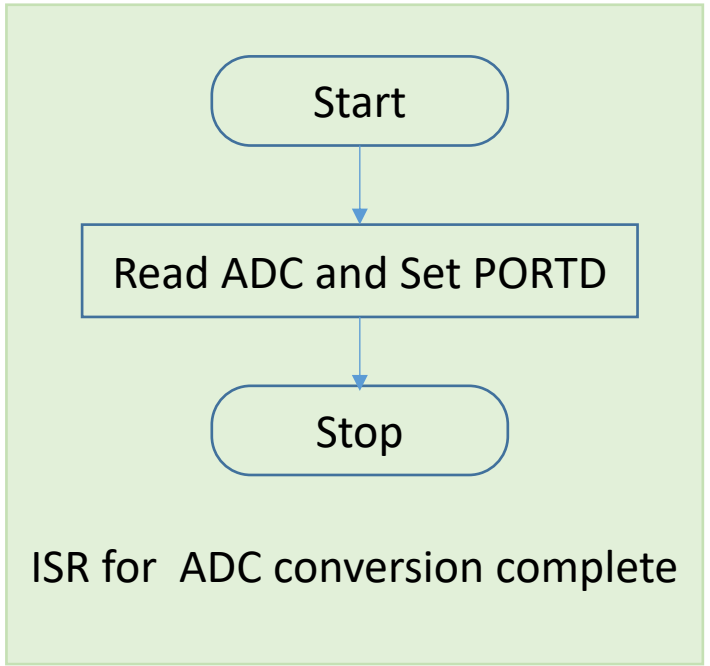
Simulate In Proteus



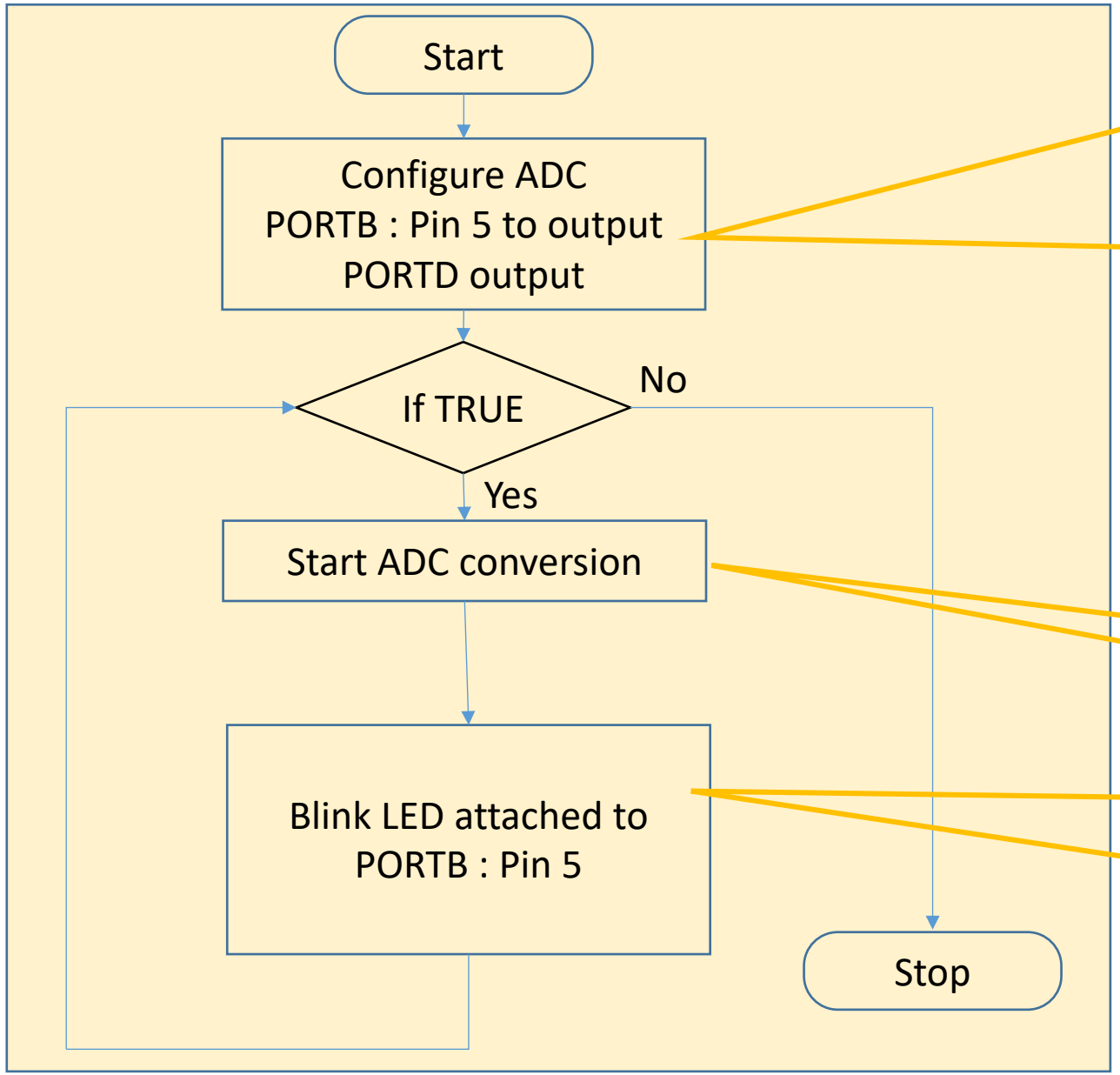
Flow Chart for A/D Converter Program With Interrupt



Single Conversion Mode with Interrupt



Coding A/D Converter Program With Interrupt

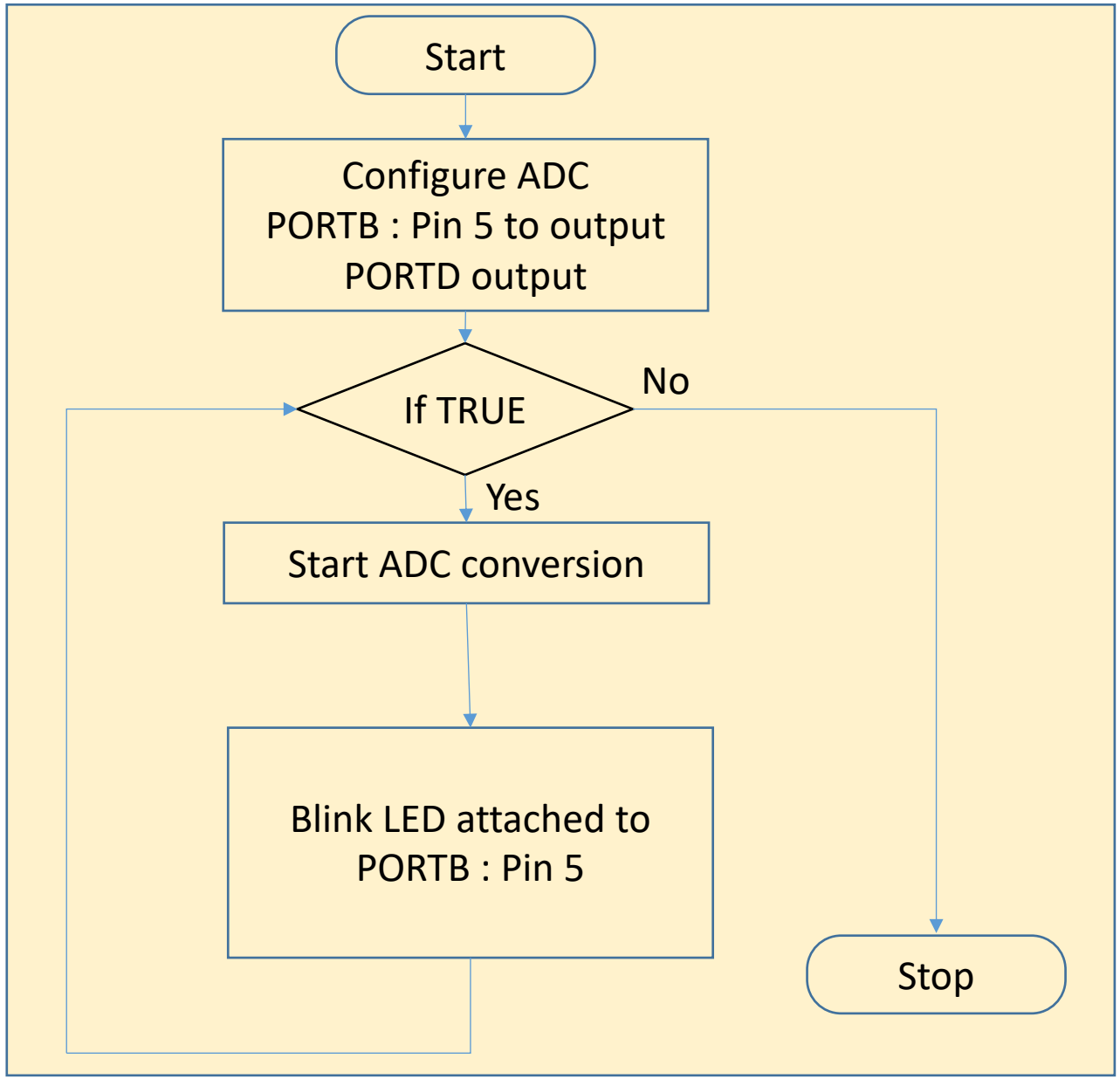


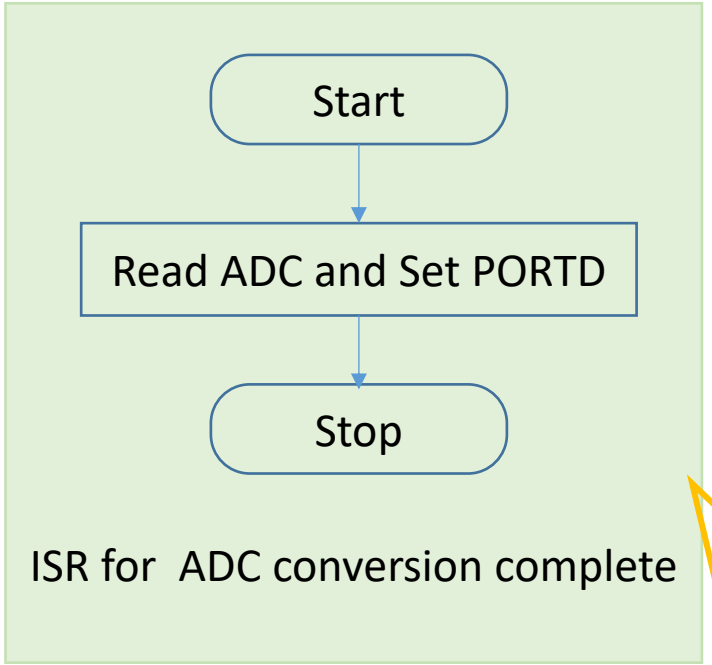

```
void ConfigureADC(void)
{
    ADCSRA |= (1<<ADEN) ;  // Enable ADC
    ADCSRA |= 0B00000111; // Selecting Pre-scalar value 128
    ADMUX  &= 0B00111111; // Enable AREF,
    // Make Four channel bits to 0 and then add channel number
    ADMUX = (ADMUX & 0B11110000) | CHANNEL;

    //Enable interrupt
    ADCSRA |= (1<<ADIE);
    sei();

}
```

```
// Start Conversion
ADCSRA |= 0B01000000;
```

```
// Blink LED at PORTB:PIN5
PORTB |= 0B00100000;
_delay_ms(500);
PORTB &= 0B11011111;
_delay_ms(500);
```

Coding A/D Converter Program With Interrupt





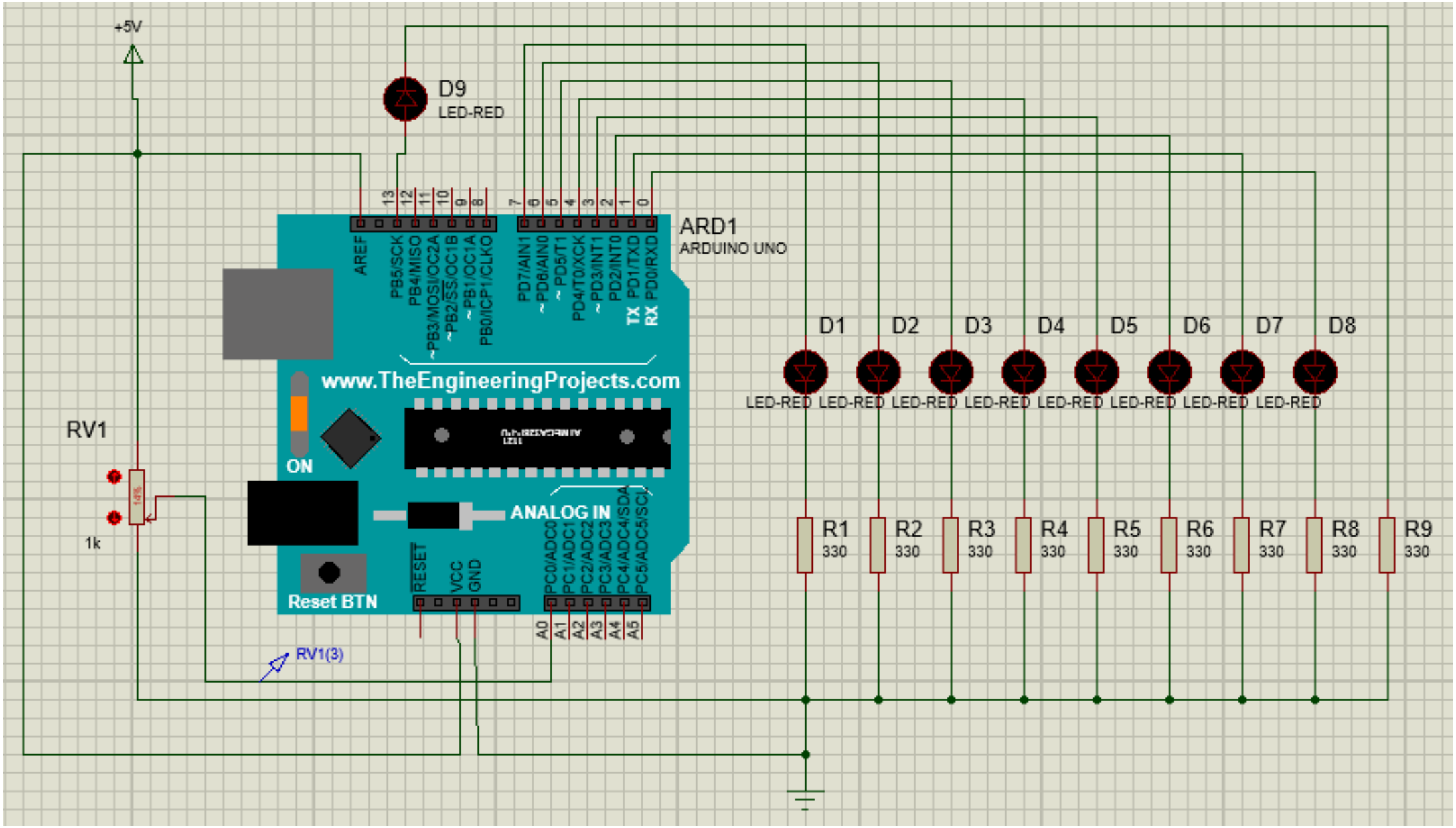

```
ISR (ADC_vect)
{
    PORTD =ADC;
}
```

Code for A/D Converter Program With Interrupt

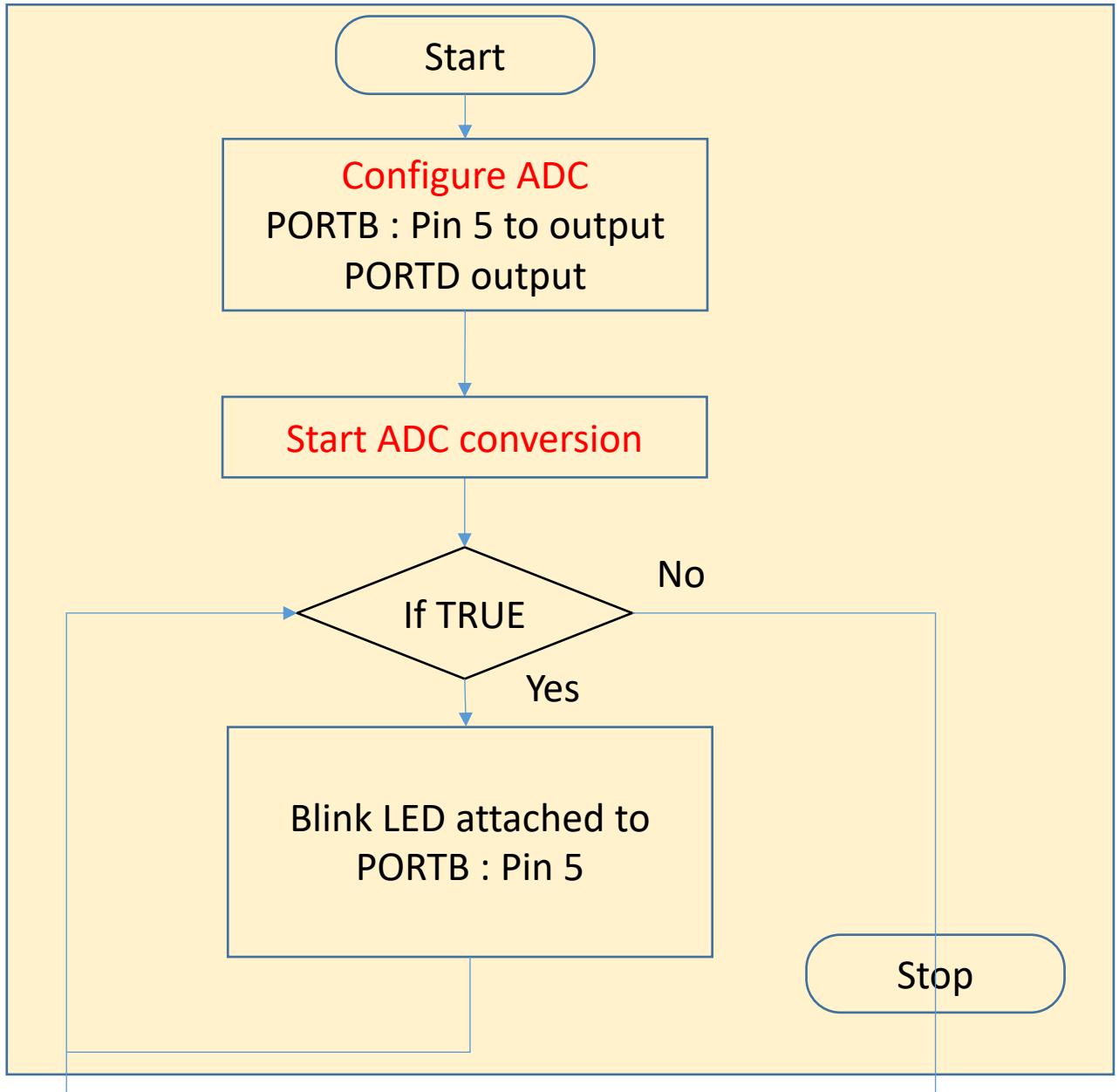
```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define CHANNEL 0B00000000

void ConfigureADC(void);

int main(void)
{   //Setting PORTD as output
    DDRD = 0B11111111;
    DDRB = 0B00100000;

    ConfigureADC();

    while (1)
    {
        // Start conversion
        ADCSRA |= 0B01000000; // Start Conversion

        // Blink LED at PORTB:PIN5
        PORTB |= 0B00100000;
        _delay_ms(500);
        PORTB &= 0B11011111;
        _delay_ms(500);
    }
}
```

```
void ConfigureADC(void)
{
    ADCSRA |= (1<<ADEN) ;  // Enable ADC
    ADCSRA |= 0B00000111; // Selecting Pre-scalar value 128
    ADMUX  &= 0B00111111; // Enable AREF,
    // Make Four channel bits to 0 and then add channel number
    ADMUX = (ADMUX & 0B11110000) | CHANNEL;

    //Enable interrupt
    ADCSRA |= (1<<ADIE);
    sei();

}

ISR (ADC_vect)
{
    PORTD =ADC;
}
```

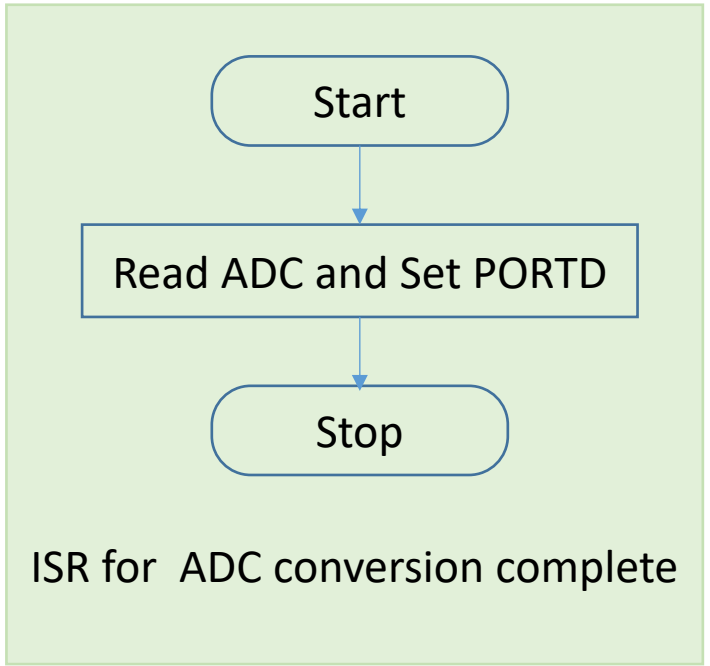
Simulate In Proteus



Flow Chart for A/D Converter Program With Interrupt



Free Running Mode with Interrupt



Coding A/D Converter Program With Interrupt

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
ccess	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

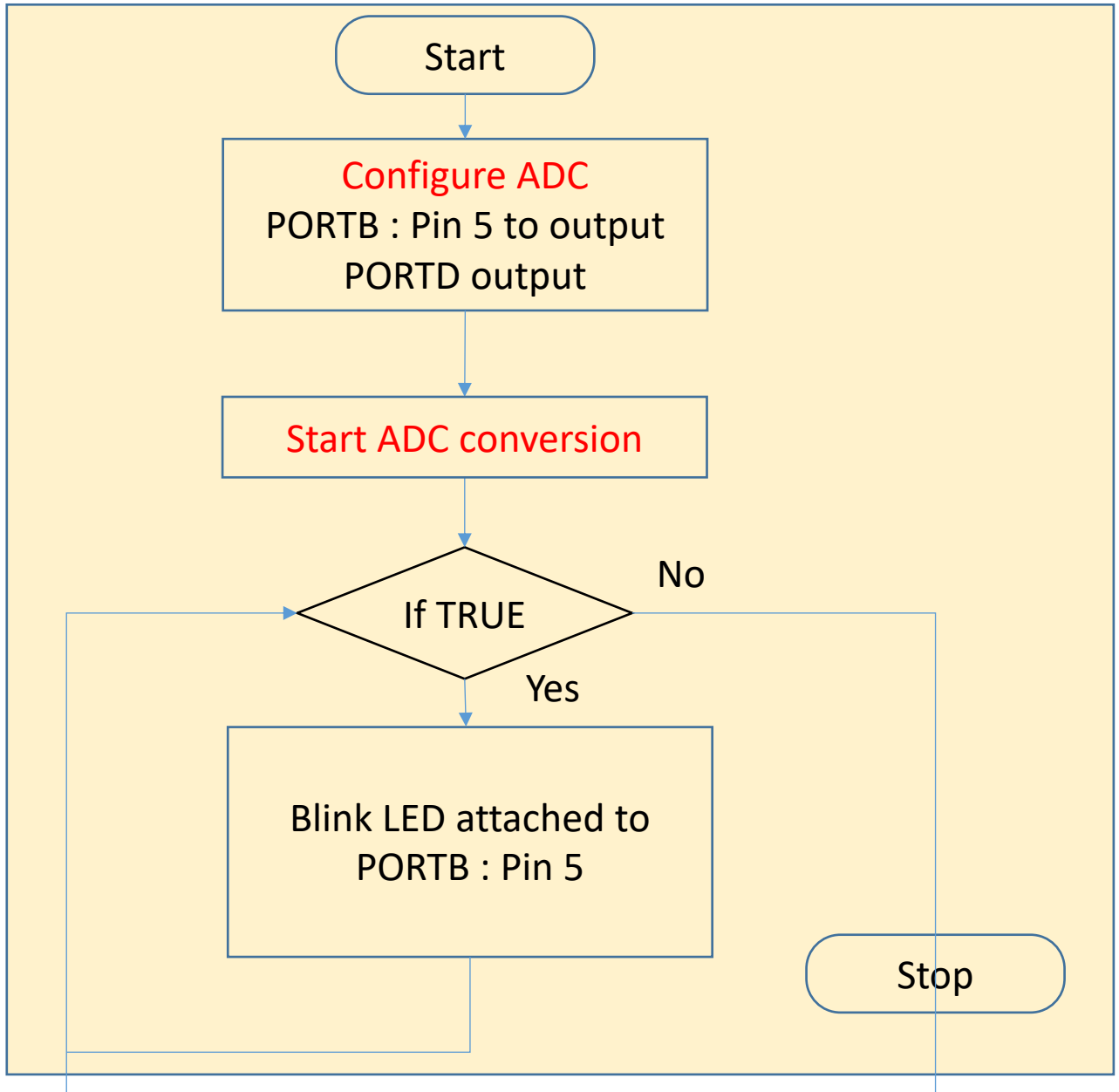
Name: ADCSRB
Offset: 0x7B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		ACME				ADTS2	ADTS1	ADTS0
Access		R/W				R/W	R/W	R/W
Reset		0				0	0	0

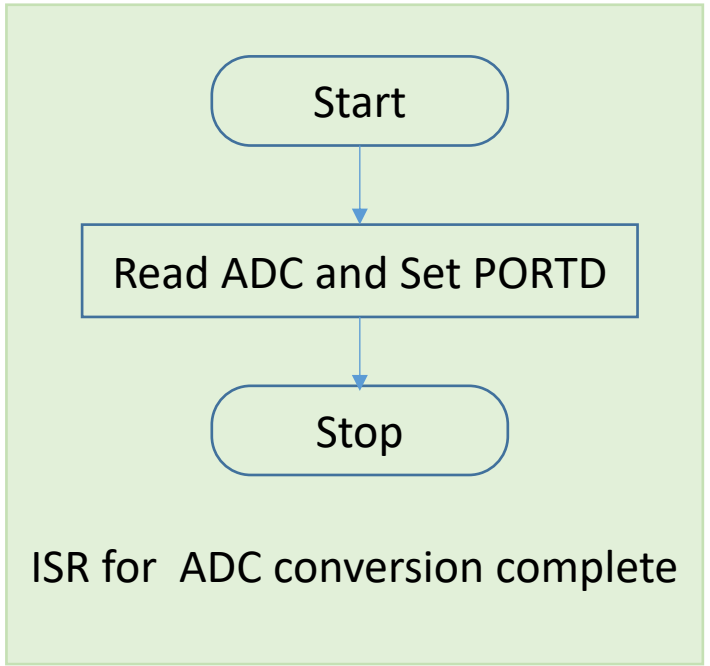
Table 28-6. ADC Auto Trigger Source Selection

ADTS[2:0]	Trigger Source
000	Free Running mode
001	Analog Comparator
010	External Interrupt Request 0
011	Timer/Counter0 Compare Match A
100	Timer/Counter0 Overflow
101	Timer/Counter1 Compare Match B
110	Timer/Counter1 Overflow
111	Timer/Counter1 Capture Event

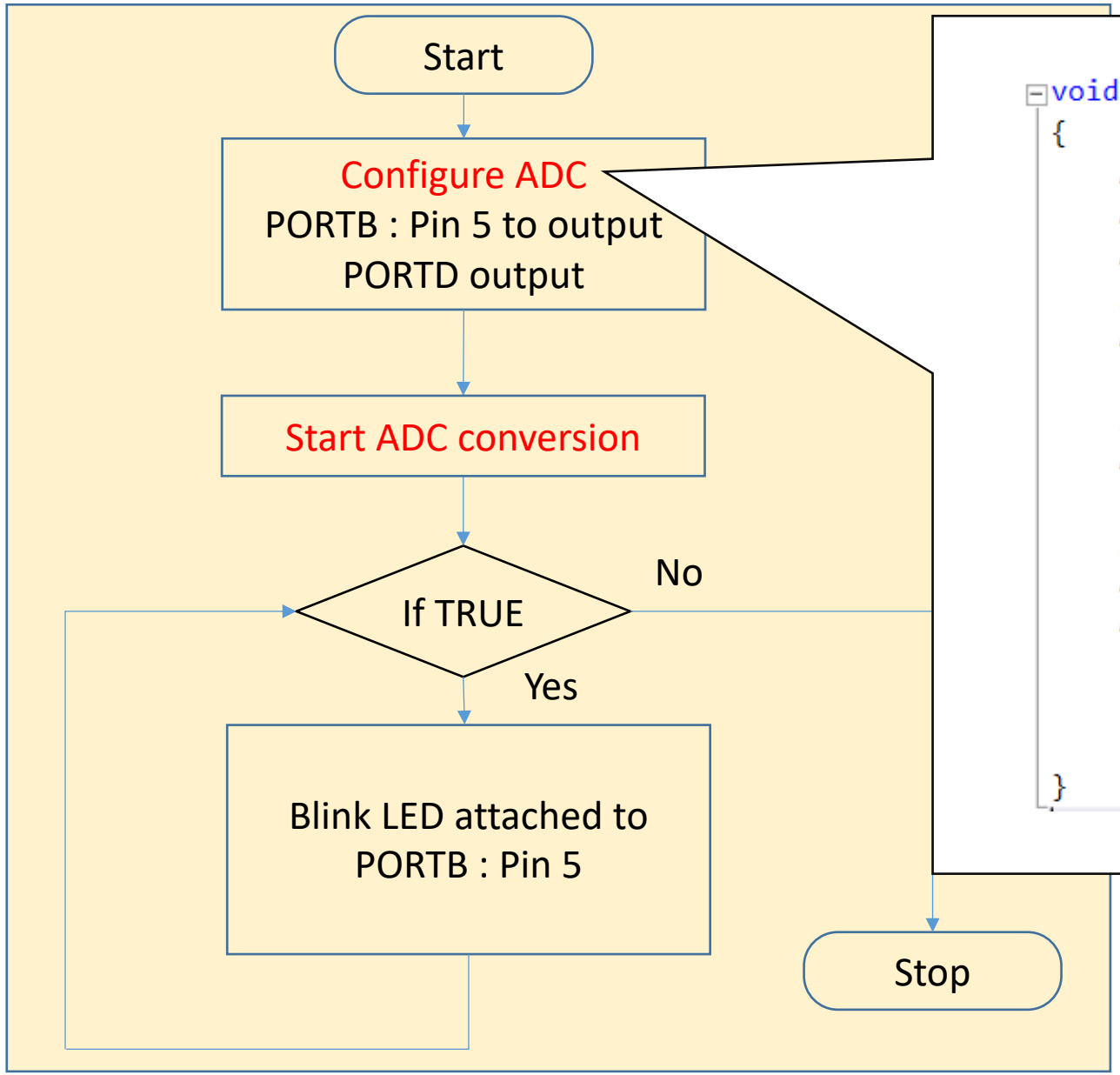
Flow Chart for A/D Converter Program With Interrupt



Free Running Mode with Interrupt

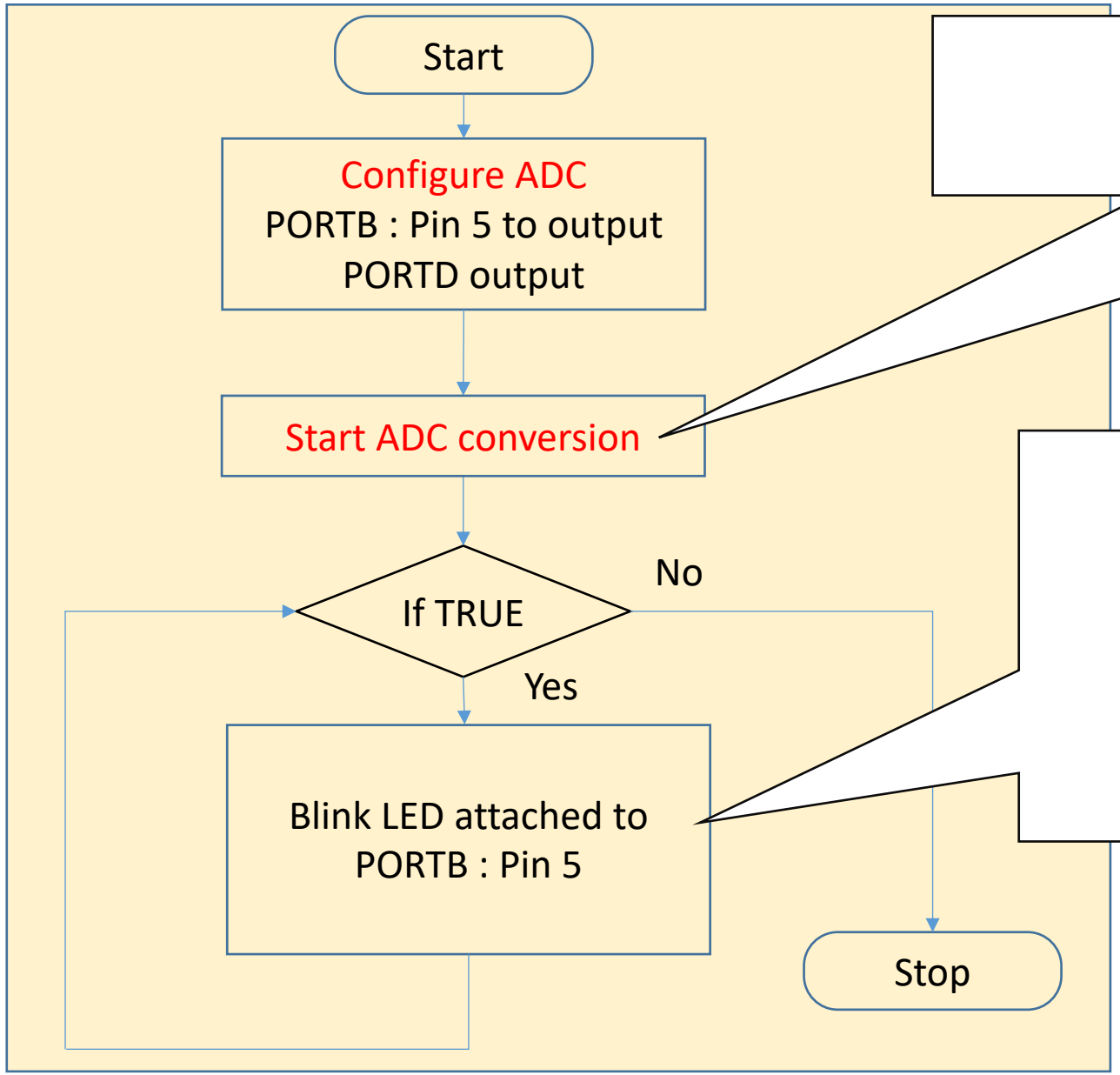


Flow Chart for A/D Converter Program With Interrupt




```
void ConfigureADC(void)
{
    ADCSRA |= (1<<ADEN) ;  // Enable ADC
    ADCSRA |= 0B00000111; // Selecting Pre-scalar value 128
    ADMUX  &= 0B00111111; // Enable AREF,
    // Make Four channel bits to 0 and then add channel number
    ADMUX = (ADMUX & 0B11110000) | CHANNEL;

    //Enable interrupt
    ADCSRA |= (1<<ADIE);

    //Enable Free Running Mode
    ADCSRA |= (1<<ADATE);
    ADCSRB &= 0B11111000;

    sei();

}
```

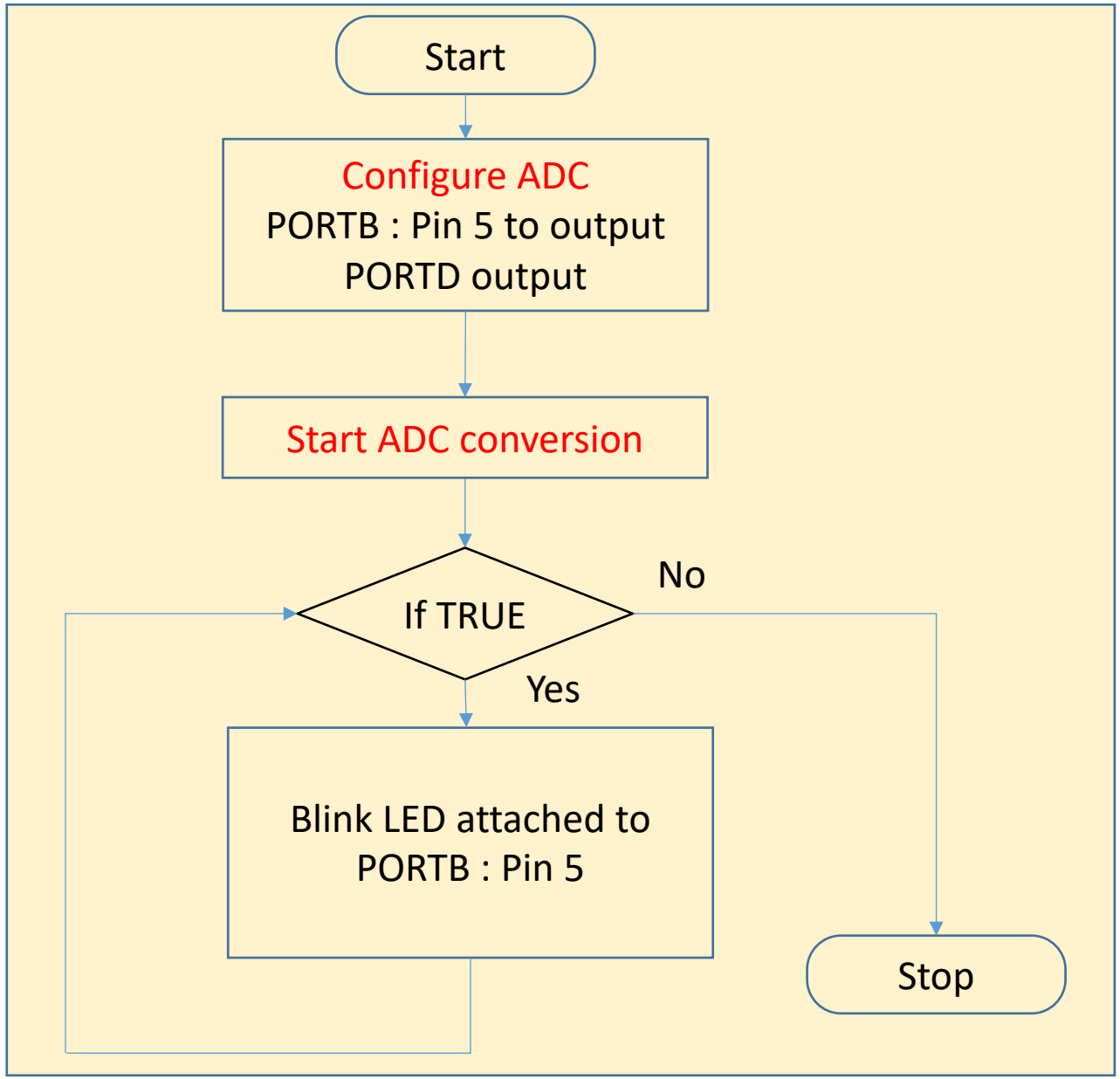
Flow Chart for A/D Converter Program With Interrupt

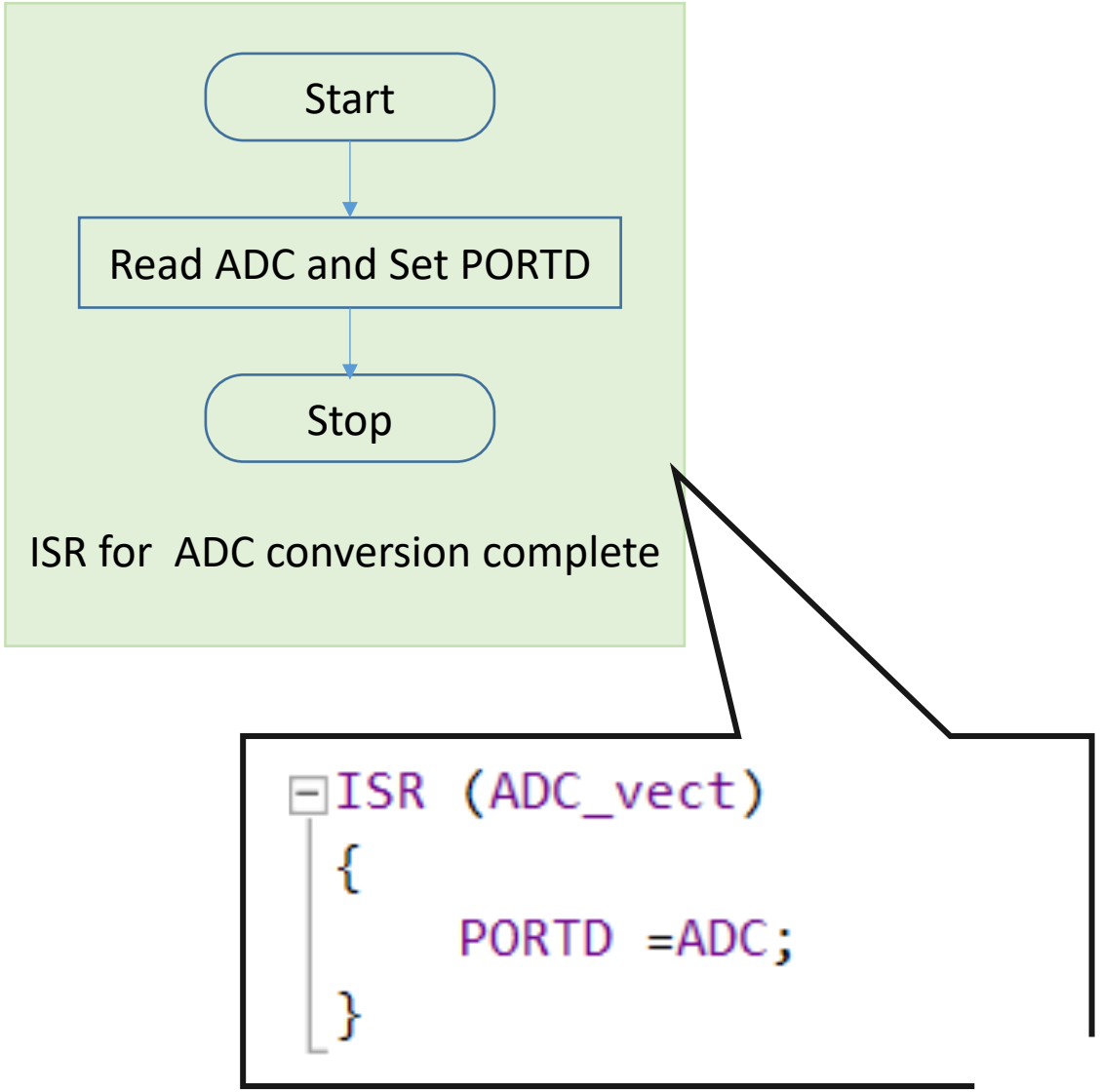



```
// Start conversion
ADCSRA |= 0B01000000; // Start Conversion
```

```
// Blink LED at PORTB:PIN5
PORTB |= 0B00100000;
_delay_ms(500);
PORTB &= 0B11011111;
_delay_ms(500);
```

Flow Chart for A/D Converter Program With Interrupt





Code for A/D Converter Program With Interrupt

```c
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define CHANNEL 0B00000000

void ConfigureADC(void);

int main(void)
{   //Setting PORTD as output
    DDRD = 0B11111111;
    DDRB = 0B00100000;

    ConfigureADC();

    // Start conversion
    ADCSRA |= 0B01000000; // Start Conversion

    while (1)
    {
        // Blink LED at PORTB:PIN5
        PORTB |= 0B00100000;
        _delay_ms(500);
        PORTB &= 0B11011111;
        _delay_ms(500);
    }
}
```

```c
void ConfigureADC(void)
{
    ADCSRA |= (1<<ADEN) ;  // Enable ADC
    ADCSRA |= 0B00000111; // Selecting Pre-scalar value 128
    ADMUX  &= 0B00111111; // Enable AREF,
    // Make Four channel bits to 0 and then add channel number
    ADMUX = (ADMUX & 0B11110000) | CHANNEL;

    //Enable interrupt
    ADCSRA |= (1<<ADIE);

    //Enable Free Running Mode
    ADCSRA |= (1<<ADATE);
    ADCSRB &= 0B11111000;

    sei();

}

ISR (ADC_vect)
{
    PORTD =ADC;
}
```

Simulate In Proteus