Google Custom Search | Search

- home
- electronics
- howto
- tools
- knowledge
- forum

ADMUX - ADC Multiplexer Selection Register

This register is used to select reference voltage source, how the result should be stored (either left adjusted or right adjusted), analog port channel to be used for conversion.

Bit	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0
Read / Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Reference Selection Bits (REFS): There are three different ways of selecting reference voltage for AD conversion by ADC. REFS1 and REFS0 bits are used to determine what reference voltage source to be used for AD conversion.It can be either internal 2.56V or through external AREF pin (Analog Reference voltage).

REFS1	REFS0	Vref Selection
0	0	AREF used as VRef and internal VRef is turned off
0	1	AVCC with external capacitor at the AREF pin is used as VRef
1	0	This is a reserved bit
1	1	Internal reference voltage of 2.56V is used with an external capacitor at AREF pin for VRef

ADC Left Adjust Result (ADLAR): Once the conversion is complete, result is stored in two ADC data registers ADCH and ADCL. This result can be either left justified or right justified. If ADLAR bit is set, then it is left adjusted, and clearing it will right justify the result. By default, bit is cleared and right justified.

Right Justified

Bit	15	14	13	12	11	10	9	8
ADCH	-	-	-	-	-	-	10th bit	9th bit
Bit	7	6	5	4	3	2	1	0
ADCL	8th bit	7th bit	6th bit	5th bit	4th bit	3rd bit	2nd bit	1st bit

Left Justified

Bit	15	14	13	12	11	10	9	8
ADCH	10th bit	9th bit	8th bit	7th bit	6th bit	5th bit	4th bit	3rd bit
Bit	7	6	5	4	3	2	1	0
ADCL	2nd bit	1st bit	-	-	-	-	-	-

Left adjusting the result is a better choice if we just need 8-bit precision. This way, we can only read ADCH register and ignore ADCL register as ADCH gives out the first 8 bits required. Be informed that ADC data register is updated once ADCH register is read. Hence we need to read ADCL first and then ADCH if we need to read both results.

Voltage equivalent is derived using the formula:
VIn [V] = (ADCH*256+ADCL) * VRef [V] / 1024for 10 bit precision(1024 as 2^10 = 1024).

As per our previous example, the resultant binary equivalent was 1011100001(2)where ADCH = 10(2)and ADCL = 11100001(2). The decimal equivalent of ADCH is 2 and ADCL is 225.
Thus, VIn = ((2*256+225) * 5) / 1024 = 3.5986328125 Volts. Now if you check the exercise we did before, you see the same value derived.

For 8 bit precision, the formula shortens to VIn [V] = (ADCH)* VRef [V]/256 where ADCH = 10111000(2); 10111000(2)= 184
Therefore, VIn [V]= 184*5/256 = 3.59375 Volts.

Bits 3:0 – MUX3:0: Analog Channel Selection Bits

These bits are used to select particular analog input channel. The table shows bits to be set to enable any particular pin.

Note: Bit 4 is not used in Atmega8 as there are only 6 analog pins (or 8 in few variations). However, there are other AVR devices which have more than 8 analog pins for which the forth bit is utilized. All the pins in Atmega8 can be selected using only 3 bits (000 … 111). Also note that if bits are changed during a conversion, the change will not go in effect until conversion is complete.

Atmega8 has 6 analog ports ADC5...ADC0 in PDIP package. ADC7 and ADC6 are only available in TQFP and QFN/MLF Package.

MUX3	MUX2	MUX1	MUX0	Single Ended Input
0	0	0	0	ADC0
0	0	0	1	ADC1
0	0	1	0	ADC2
0	0	1	1	ADC3
0	1	0	0	ADC4
0	1	0	1	ADC5
0	1	1	0	ADC6 *
0	1	1	1	ADC7 *
1	0	0	0	Not used in Atmega8
1	0	0	1	Not used in Atmega8
1	0	1	0	Not used in Atmega8
1	0	1	1	Not used in Atmega8

1	1	0	0	Not used in Atmega8
1	1	0	1	Not used in Atmega8
1	1	1	0	Not used in Atmega8
1	1	1	1	Not used in Atmega8

* Only available in TQFP and QFN/MLF Package

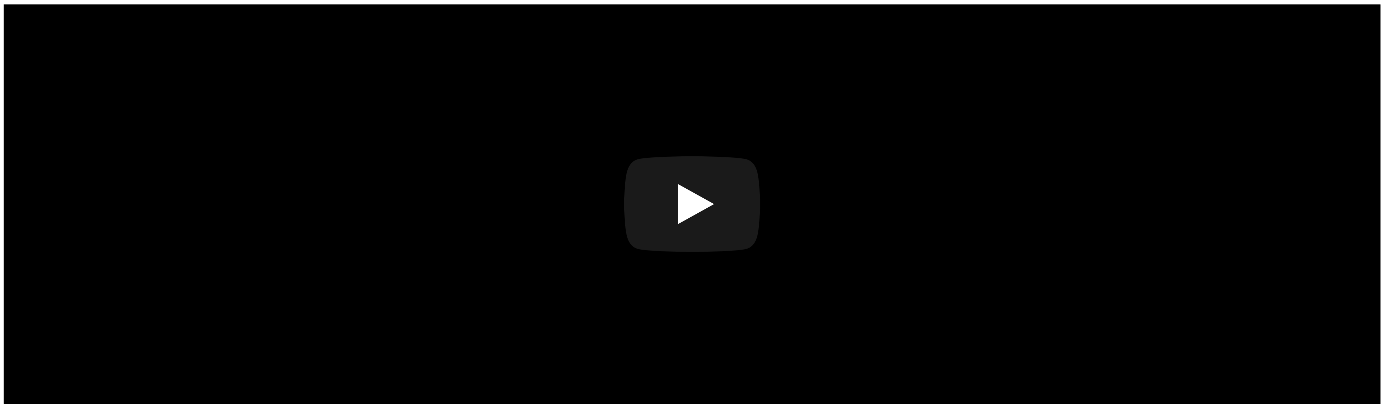
Note: If you check the datasheet of any AVR microcontroller, there are many places where you find “reserved bits”. In this case, keep reserved bits in their default value. If accessed, they should be written to "0 (zero)" so that they are compatible with future devices.

This ends a detailed explanation of ADC concepts. In the next section, we will see how to set up ADC in Atmega8 and also write a small program which shows the basic code required for ADC, which you can further extend based on your requirement.

Tutorial index:

«Previous - § - Next»

Featured Videos



Recent Articles

Atmega8 Development Board

A great step-by-step tutorial on building your own Atmel AVR based Atmega8 development board. The board is ideal for beginners with detailed explanation and pictures More...

L293D Motor Driver

For robots to do work, you need to know how to control a motor. L293D is a cleverly packed IC which can control two DC motors in both directions: forwards and reverse. Here is a detailed explanation of building a board based on L293D ICMore...

Hobby Servo Tutorial

Servo Motor is a device which uses error-sensing feedback signals to determine and control the position of a motor shaft. The term "servomechanism" closely relates to servo motors..More...

Blinking LED Tutorial

This is similar to what we achieve in any "Hello World" program. However, it is not just limited to blinking LED but scratches the surface of AVR-GCC programming... More...

Kindly Donate

If this site has helped you, then kindly consider a Donation to say "Thank You!!". Donation might help us keep all this information available for free and also pay for the resources.

If that is difficult, then a simple "Hi" in the forum would still do good :)

Five US Dollars $5.00 USD ▼

Pay Now