

List of commands (public functions) of the ADS1115_WE library

Function	Parameters	what it does
<code>void reset()</code>	none	Resets the ADS1115. Be careful this function uses the I2C general call reset. Therefore all devices listening to general calls will be resetted.
<code>bool init ()</code>	none	Sets the Configuration Register to default values. Returns false if the ADS1115 is not connected
<code>uint8_t isDisconnected ()</code>	none	Checks whether the I2C connection to the ADS1115 works. Returns the the return value of <code>Wire.endTransmission()</code> .
<code>void setAlertPinMode(mode)</code>	ADS1115_ASSERT_AFTER_X with X = 1, 2, 4; or: ADS1115_DISABLE_ALERT (default)	When alert limits are set, this function defines after how many out of range samples the alarm pin will assert. When the alert pin is set to conversion ready alert, you also need call this function with any parameter except ADS1115_DISABLE_ALERT. Which one you choose exactly doesn't matter.
<code>void setAlertLatch(mode)</code>	ADS1115_LATCH_DISABLED ADS1115_LATCH_ENABLED	When the alert latch is enabled, the alert pin will be active until you call <code>clearAlert</code> or until you call <code>getResult_V</code> / <code>getResult_mV</code> . When disabled, the alert pin will be deactivated, if the results are within limits.
<code>void setAlertPol(polarity)</code>	ADS1115_ACTIVE_HIGH ADS1115_ACTIVE_LOW (default)	The alert pin can be set active-high or active-low
<code>void setAlertModeAndLimit_V(mode, upper limit, lower limit)</code>	mode: ADS1115_MAX_LIMIT ADS1115_WINDOW limits: voltage [V]	In max limit mode the alert pin will be active when the max limit is exceeded. The pin will be deactivated again, when the the results are below the lower limit (if latch is not enabled) In the window mode the alert pin will be active when result are out of the window limits. It will be deactivated, when results are in the window limits again (if latch is not enabled).
<code>void setConvRate(rate)</code>	ADS1115_X_SPS with X = 8, 16, 32, 64, 128, 250, 475, 860	Sets the conversion rate in number of samples per second (SPS).
<code>convRate getConvRate()</code>	none	Returns the conversion rate as <code>convRate</code> , i.e. ADS1115_X_SPS
<code>void setMeasureMode(mode)</code>	ADS1115_CONTINUOUS ADS1115_SINGLE	Sets continuous or single-shot mode. In single-shot mode the measurements need to be triggered manually (<code>startSingleMeasurement</code>). After a single-shot measurement the device goes into power-down mode.
<code>ADS1115_MEASURE_MODE getMeasureMode()</code>	none	Returns the measure mode as ADS1115_MEASURE_MODE
<code>void setVoltageRange_mV(range)</code>	ADS1115_RANGE_X with X = 6144, 4096, 2048, 1024, 0512, 0256	Sets voltage range in mV and therefore the gain amplifier. The range is always from -X to +X. Voltages applied to the input pins shall not exceed VCC + 0.3 Volt. If you change the range the compare registers will be updated automatically.
<code>uint16_t getVoltageRange_mV()</code>	none	Returns the voltage range in millivolts.
<code>ADS1115_RANGE getRange()</code>	none	Returns the range as ADS1115_RANGE.
<code>void setAutoRange()</code>	none	Switches into the maximum range and continuous mode (if in single shot mode), measures the voltage and then switches to the smallest range in which the measured voltage is below 80% of the maximum of the range. If the ADS1115 was in single shot mode before it changes back again. You should only use this function if you expect non- or slow-changing voltages. The procedure takes several conversion times.
<code>void setPermanentAutoRangeMode (true / false)</code>	true / false	Sets the automatic voltage range permanently, but the range will only be changed if the vmeasured value is outside 30 - 80% of the maximum value of the current range. Therefore this method is faster than <code>setAutoRange()</code> .
<code>void setRememberChannelRanges (true / false)</code>	true / false	This function can be used in combination with <code>setPermanentAutoRangeMode</code> . If set, the optimized range for a channel will be saved before changing to another channel. When changing back the range will be set again. With this you can avoid unnecessary extra measurements to find the optimum range.
<code>void setCompareChannels()</code> <code>void setCompareChannels_nonblock()</code>	ADS1115_COMP_X with X = 0_1, 0_3, 1_3, 2_3, 0_GND, 1_GND, 2_GND, 3_GND	Sets the channels to be compared channels to be compared to GND. The nonblock version does not wait until data of the new channel is available - you have to take care yourself.
<code>bool isBusy()</code>	none	Reads the Conversion Ready flag. Works only in single-shot mode.
<code>void startSingleMeasurement()</code>	none	Triggers a single-shot measurement.
<code>void getResult_V() / getResult_mV()</code>	none	Returns the result currently available in the conversion register either in Volt or Millivolt. It does not wait for the current conversion to be completed.
<code>int getRawResult()</code>	none	Returns the raw result from the conversion registers which depends on the voltage as well as on the voltage range. E. g. if you choose ADS1115_RANGE_6144 a value of +32767 means 6144 mV; for ADS1115_RANGE_0256 a value of +32767 would mean 256 mV.

<pre>int getResultWithRange(min, max)</pre> <pre>int getResultWithRange(min, max, maxVolt)</pre>	<p>min / max: minimum / maximum of the range you scale to</p> <p>maxVolt: maximum of the voltage range you scale to</p>	<p>Best to be explained with an example: Condition: voltage range of the ADS1115 is ADS1115_6144 and you use getResultWithRange(-1023, +1023, +5000) Effect: the range of -32767 to +32767 is scaled down to -1023 to +1023. The third parameter (5000) leads to 5000 mV delivering a return value of +1023 -> like an Arduino UNO ADC with standard conditions. You have to take care that maxVolt is within the voltage range (ADS1115_RANGE_XXXX). If you don't use the third parameter, the voltage scale (maxVolt) will be equal to the voltage range XXXX in ADS1115_RANGE_XXXX.</p>
<pre>unsigned int getVoltageRange_mV()</pre>	<p>none</p>	<p>Returns the voltage range / gain you have chosen. In simple words it returns the XXXX in ADS1115_RANGE_XXXX.</p>
<pre>void setAlertPinToConversionReady()</pre>	<p>none</p>	<p>The alert pin will be active when a conversion is completed. You will also need to call the setAlertPinMode function. You can either use the alert pin for limit alerts or conversion ready alerts, not both in parallel.</p>
<pre>void clearAlert()</pre>	<p>none</p>	<p>When the alert latch is enabled, the alert will be active until you call this function or until you call getResult_V / getResult_mV.</p>