List of commands (public functions) of the ADS1115_WE library

Function	Parameters	what it does
void reset( )	none	Resets the ADS1115. Be careful this function uses the I2C general call reset. Therefore all devices listening to general calls will be resetted.
bool init ( )	none	Sets the Configuration Register to default values. Returns false if the ADS1115 is not connected
uint8_t isDisconnected ( )	none	Checks whether the I2C connection to the ADS1115 works. Returns the the return value of Wire.endTransmission().
void setAlertPinMode(*mode*)	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.
void setAlertLatch(*mode*)	ADS1115_LATCH_DISABLED ADS1115_LATCH_ENABLED	When the alert latch is enabled, the alert pin will be active until you call clearAlert or until you call getResult_V / getResult_mV. When disabled, the alert pin will be deactivated, if the results are within limits.
void setAlertPol(*polarity*)	ADS1115_ACTIVE_HIGH ADS1115_ACTIVE_LOW (default)	The alert pin can be set active-high or active-low
void setAlertModeAndLimit_V(*mode, upper limit, lower limit*)	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).
void setConvRate(*rate*)	ADS1115_*X*_SPS with *X* = 8, 16, 32, 64, 128, 250, 475, 860	Sets the conversion rate in number of samples per second (SPS).
convRate getConvRate( )	none	Returns the conversion rate as convRate, i.e. ADS1115_*X*_SPS
void setMeasureMode(*mode*)	ADS1115_CONTINUOUS ADS1115_SINGLE	Sets continuous or single-shot mode. In single-shot mode the measurements need to be triggered manually (startSingleMeasurement). After a single-shot measurement the device goes into power-down mode.
void setVoltageRange_mV(*range*)	ADS1115_RANGE_*X* with *X* = 6144, 4096, 2048, 1024, 0512, 0256	Sets voltage range in mV and therefore the gain amplifier. The range is alway 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.
void setAutoRange( )	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.
void setPermanentAutoRangeMode (*true /false*)	true / false	Sets the automatic voltage range permanantly, 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 setAutoRange().
void setCompareChannels( )	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
bool isBusy( )	none	Reads the Conversion Ready flag. Works only in single-shot mode.
void startSingleMeasurement( )	none	Triggers a single-shot measurement.
void getResult_V( ) / getResult_mV( )	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.
int getRawResult( )	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.
int getResultWithRange(*min, max*) int getResultWithRange(*min, max, maxVolt*)	min / max: minimum / maximum of the range you scale to maxVolt: maximum of the voltage range you scale to	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.
unsigned int getVoltageRange_mV( )	none	Returns the voltage range / gain you have chosen. In simple words it returns the XXXX in ADS1115_RANGE_XXXX.
void setAlertPinToConversionReady( )	none	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.
void clearAlert( )	none	When the alert latch is enabled, the alert will be active until you call this function or until you call getResult_V / getResult_mV.