

# How to change DS18B20 temperature to 12bit resolution with using MERVIS environment

1. start new project
2. add 1Wire channel
3. assign DS18B20 thermometer
4. add new group in "channel" - name "twelvebit"

The screenshot shows the MERVIS environment with the 'channel' variable browser and the properties window for the 'twelvebit' group.

**Variable Browser - channel**

Name	Autogen Name	Enable	Comm. Value Mapping	ST Type	Array Length	Group	Read/Write Interval
1W-Thermometer		True					
Convert							00:00:00
save							00:00:00
Scratchpad							00:00:00
twelvebit							00:00:00
Convert_commblock			Bit	bool	0	Convert	00:00:00
Convert_commererror			Bit	bool	0	Convert	00:00:00
Convert_timestamp			Builtin	dt	0	Convert	00:00:00
save_commblock			Bit	bool	0	save	00:00:00
save_commererror			Bit	bool	0	save	00:00:00
save_timestamp			Builtin	dt	0	save	00:00:00
Scratchpad_commblock			Bit	bool	0	Scratchpad	00:00:00
Scratchpad_commererror			Bit	bool	0	Scratchpad	00:00:00
Scratchpad_timestamp			Builtin	dt	0	Scratchpad	00:00:00
twelvebit_commblock			Bit	bool	0	twelvebit	00:00:00
twelvebit_commererror			Bit	bool	0	twelvebit	00:00:00
twelvebit_timestamp			Builtin	dt	0	twelvebit	00:00:00
dosave			Bit	bool	0	save	00:00:00
Temperature			Builtin	int	0	Scratchpad	00:00:00
twelve			Builtin	lint	0	twelvebit	00:00:00
Attributes	True						
commererror			Bit	bool	0		00:00:00
priorityrefresh			Bit	bool	0		00:00:00
timestamp			Builtin	dt	0		00:00:00

**Properties - Group Properties**

Name: twelvebit

Assigned HW Properties: Multiple Values

DataPointSortOrder: 0

Id: 06271598-44F5-4bdc-e9be-6aee7...

Is Custom Box: True

Read/Write Interval: 0ms

Group Type: WriteOnly

Note:

PlatformIO group parameters:

Command: 78

Command Length: 1

Data Length: 3

Send as Broadcast: False

Enable CRC Check: False

Input filter mask [hex]:

Input expected mask [hex]:

Write Only On Change: True

Logical OR over Overlapped: False

Note:

5. add new group in "channel" - name "save"

The screenshot shows the MERVIS environment with the 'channel' variable browser and the properties window for the 'save' group.

**Variable Browser - channel**

Name	Autogen Name	Enable	Comm. Value Mapping	ST Type	Array Length	Group	Read/Write Interval
1W-Thermometer		True					
Convert							00:00:00
save							00:00:00
Scratchpad							00:00:00
twelvebit							00:00:00
Convert_commblock			Bit	bool	0	Convert	00:00:00
Convert_commererror			Bit	bool	0	Convert	00:00:00
Convert_timestamp			Builtin	dt	0	Convert	00:00:00
save_commblock			Bit	bool	0	save	00:00:00
save_commererror			Bit	bool	0	save	00:00:00
save_timestamp			Builtin	dt	0	save	00:00:00
Scratchpad_commblock			Bit	bool	0	Scratchpad	00:00:00
Scratchpad_commererror			Bit	bool	0	Scratchpad	00:00:00
Scratchpad_timestamp			Builtin	dt	0	Scratchpad	00:00:00
twelvebit_commblock			Bit	bool	0	twelvebit	00:00:00
twelvebit_commererror			Bit	bool	0	twelvebit	00:00:00
twelvebit_timestamp			Builtin	dt	0	twelvebit	00:00:00
dosave			Bit	bool	0	save	00:00:00
Temperature			Builtin	int	0	Scratchpad	00:00:00
twelve			Builtin	lint	0	twelvebit	00:00:00
Attributes	True						
commererror			Bit	bool	0		00:00:00
priorityrefresh			Bit	bool	0		00:00:00
timestamp			Builtin	dt	0		00:00:00

**Properties - Group Properties**

Name: save

Assigned HW Properties: Multiple Values

DataPointSortOrder: 0

Id: 548b3f1c-95ae-439a-981e-fabae...

Is Custom Box: True

Read/Write Interval: 0ms

Group Type: WriteOnly

Note:

PlatformIO group parameters:

Command: 72

Command Length: 1

Data Length: 0

Send as Broadcast: False

Enable CRC Check: False

Input filter mask [hex]:

Input expected mask [hex]:

Write Only On Change: True

Logical OR over Overlapped: False

Note:

6. add new "data point" in "channel" - name "twelve"

The screenshot shows the MERVIS environment with the 'channel' variable browser and the properties window for the 'twelve' data point.

**Variable Browser - channel**

Name	Autogen Name	Enable	Comm. Value Mapping	ST Type	Array Length	Group	Read/Write Interval
1W-Thermometer		True					
Convert							00:00:00
save							00:00:00
Scratchpad							00:00:00
twelvebit							00:00:00
Convert_commblock			Bit	bool	0	Convert	00:00:00
Convert_commererror			Bit	bool	0	Convert	00:00:00
Convert_timestamp			Builtin	dt	0	Convert	00:00:00
save_commblock			Bit	bool	0	save	00:00:00
save_commererror			Bit	bool	0	save	00:00:00
save_timestamp			Builtin	dt	0	save	00:00:00
Scratchpad_commblock			Bit	bool	0	Scratchpad	00:00:00
Scratchpad_commererror			Bit	bool	0	Scratchpad	00:00:00
Scratchpad_timestamp			Builtin	dt	0	Scratchpad	00:00:00
twelvebit_commblock			Bit	bool	0	twelvebit	00:00:00
twelvebit_commererror			Bit	bool	0	twelvebit	00:00:00
twelvebit_timestamp			Builtin	dt	0	twelvebit	00:00:00
dosave			Bit	bool	0	save	00:00:00
Temperature			Builtin	int	0	Scratchpad	00:00:00
twelve			Builtin	lint	0	twelvebit	00:00:00
Attributes	True						
commererror			Bit	bool	0		00:00:00
priorityrefresh			Bit	bool	0		00:00:00
timestamp			Builtin	dt	0		00:00:00

**Properties - Group Properties**

Name: twelve

Assigned HW Properties: Multiple Values

DataPointSortOrder: 0

Id: 2af7806d-6ac3-4359-ab14-4b348...

Is Custom Box: True

Read/Write Interval: 0ms

Group Type: WriteOnly

Comm. Value Mapped Type: Builtin

ST Type: lint

Transform: identity

Note:

Analog input configuration:

Autogen:

Enable SWAutogen: True

Target Project (SWAutogen):

Generated variable Name (SWAutogen): hw.\$1w-thermometer\_twelve\$

Retain (SWAutogen): True

Mapping:

IO => ST: \$hw.\$1w-thermometer\_twelve\$

PlatformIO datapoint parameters:

Write Offset Gap: 0

Data Offset (Parser): 0

Bit Offset (Parser): 0

MultiByte Length (Parser): 3

MultiByte Order (Parser): 17345678

## 7. add new "data point" in "channel" - name "doSave"

The Variable Browser shows the following variables for the 'channel' object:

Name	Autogen Name	Enable	Comm. Value Mapped	ST Type	Array Length	Group	Read/Write
1W-Thermometer		True	-	-	-	-	-
Convert	Convert	-	-	bool	0	Convert	00:00:00
save	save	-	-	bool	0	Convert	00:00:00
Scratchpad	Scratchpad	-	-	bool	0	Convert	00:00:00
twelvebit	twelvebit	-	-	bool	0	Convert	00:00:00
Convert_commblock	Convert_commblock	-	-	bool	0	Convert	00:00:00
Convert_commerror	Convert_commerror	-	-	bool	0	Convert	00:00:00
Convert_timestamp	Convert_timestamp	-	-	bool	0	Convert	00:00:00
save_commblock	save_commblock	-	-	bool	0	save	00:00:00
save_commerror	save_commerror	-	-	bool	0	save	00:00:00
save_timestamp	save_timestamp	-	-	bool	0	save	00:00:00
Scratchpad_commblock	Scratchpad_commblock	-	-	bool	0	Scratchpad	00:00:00
Scratchpad_commerror	Scratchpad_commerror	-	-	bool	0	Scratchpad	00:00:00
Scratchpad_timestamp	Scratchpad_timestamp	-	-	bool	0	Scratchpad	00:00:00
twelvebit_commblock	twelvebit_commblock	-	-	bool	0	twelvebit	00:00:00
twelvebit_commerror	twelvebit_commerror	-	-	bool	0	twelvebit	00:00:00
twelvebit_timestamp	twelvebit_timestamp	-	-	bool	0	twelvebit	00:00:00
doSave	doSave	-	-	bool	0	save	00:00:00
Temperature	Temperature	-	-	int	0	Scratchpad	00:00:00
twelve	twelve	-	-	int	0	twelvebit	00:00:00
Attributes	Attributes	True	-	-	-	-	-
commerror	commerror	-	-	bool	0	Scratchpad	00:00:00
priorityrefresh	priorityrefresh	-	-	bool	0	Scratchpad	00:00:00
timestamp	timestamp	-	-	dt	0	Scratchpad	00:00:00

The Properties window shows the following settings for the 'doSave' variable:

- Name: doSave
- Assigned HW Properties: Multiple Values
- DataPointSortOrder: 0
- Group: save
- Id: c81841b-d189-44e7-8dc5-1c2e6...
- Is Custom Box: True
- Read/Write Interval: 0ms
- Group Type: WriteOnly
- Comm. Value Mapped Type: Bit
- ST Type: bool
- Transform: identity
- Note:
- Autogen:
- Enable SWAutogen: True
- Target Project (SWAutogen):
- Generated variable Name (SWAutogen): hw.\$1w-thermometer\_dosave\$
- Reason (SWAutogen): True
- Mapping:
- IO > ST:
- ST > IO: \$hw.\$1w-thermometer\_dosave\$
- PlatformIO datapoint parameters:
- Write Offset Gap: 0
- Data Offset (Parser): 0
- Bit Offset (Parser): 0
- MultiByte Length (Parser): 0
- MultiByte Order (Parser): 12345678
- And Mask (hex): 0

## 8. build and run - after writing 127 the DS18B20 run in 12bit resolution mode / till switch OFF/

The Variable Browser shows the following variables for the 'channel' object:

Name	Namesp	Module	Type	Kind	PLC Value	PLC Last Comm
1W-Thermometer_Scratchpad_commblock	hw	smazat5	bool	Global	False	1/19/2018 1:58:00 PM
1W-Thermometer_Scratchpad_timestamp	hw	smazat5	dt	Global	dt#2018-01-19-12:58:00	1/19/2018 1:58:00 PM
1W-Thermometer_Scratchpad_commerror	hw	smazat5	bool	Global	False	1/19/2018 1:58:00 PM
1W-Thermometer_Convert_commblock	hw	smazat5	bool	Global	False	1/19/2018 1:58:00 PM
1W-Thermometer_Convert_timestamp	hw	smazat5	dt	Global	dt#2018-01-19-12:58:00	1/19/2018 1:58:00 PM
1W-Thermometer_Convert_commerror	hw	smazat5	bool	Global	False	1/19/2018 1:58:00 PM
1W-Thermometer_Temperature	hw	smazat5	real	Global	30.8125	1/19/2018 1:58:00 PM
1W-Thermometer_dosave	hw	smazat5	bool	Global	False	1/19/2018 1:57:08 PM
1W-Thermometer_twelve	hw	smazat5	int	Global	127	1/19/2018 1:56:58 PM
channel.1W-Thermometer.Temperature	HW_hidden	PLC	int	Global	493	1/19/2018 1:58:00 PM
tmp_real	HW_hidden	PLC	real	Global	30.8125	1/19/2018 1:58:00 PM
channel.1W-Thermometer.Temperature.transformation	HW_hidden	PLC	Linear	Global	{eno=0,en=1,k=0.0625,q=0}	1/19/2018 1:58:00 PM

The conversion table shows the relationship between decimal, hexadecimal, resolution, and conversion time:

dec	hex	resolution	conversion time
127	#7F	12 bit	93,75ms
95	#5F	11 bit	185,5ms
63	#3F	10 bit	375ms
31	#1F	9 bit	750ms

## 9. for permanent change - set "doSave" to "True"

The Variable Browser shows the following variables for the 'channel' object:

Name	Namesp	Module	Type	Kind	PLC Value	PLC Last Comm
1W-Thermometer_Scratchpad_commblock	hw	smazat5	bool	Global	False	1/19/2018 1:59:12 PM
1W-Thermometer_Scratchpad_timestamp	hw	smazat5	dt	Global	dt#2018-01-19-12:59:12	1/19/2018 1:59:12 PM
1W-Thermometer_Scratchpad_commerror	hw	smazat5	bool	Global	False	1/19/2018 1:59:12 PM
1W-Thermometer_Convert_commblock	hw	smazat5	bool	Global	False	1/19/2018 1:59:12 PM
1W-Thermometer_Convert_timestamp	hw	smazat5	dt	Global	dt#2018-01-19-12:59:12	1/19/2018 1:59:12 PM
1W-Thermometer_Convert_commerror	hw	smazat5	bool	Global	False	1/19/2018 1:59:12 PM
1W-Thermometer_Temperature	hw	smazat5	real	Global	28.75	1/19/2018 1:59:12 PM
1W-Thermometer_dosave	hw	smazat5	bool	Global	True	1/19/2018 1:59:04 PM
1W-Thermometer_twelve	hw	smazat5	int	Global	127	1/19/2018 1:56:58 PM
channel.1W-Thermometer.Temperature	HW_hidden	PLC	int	Global	460	1/19/2018 1:59:12 PM
tmp_real	HW_hidden	PLC	real	Global	28.75	1/19/2018 1:59:12 PM
channel.1W-Thermometer.Temperature.transformation	HW_hidden	PLC	Linear	Global	{eno=0,en=1,k=0.0625,q=0}	1/19/2018 1:59:12 PM

The Port Monitor window shows the communication data for the 'channel' object. The 'doSave' variable is highlighted, showing a value of True.

The change is visible in "port monitor" view

## MEMORY

The DS18B20's memory is organized as shown in Figure 7. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers ( $T_H$  and  $T_L$ ) and configuration register. Note that if the DS18B20 alarm function is not used, the  $T_H$  and  $T_L$  registers can serve as general-purpose memory. All memory commands are described in detail in the *DS18B20 Function Commands* section.

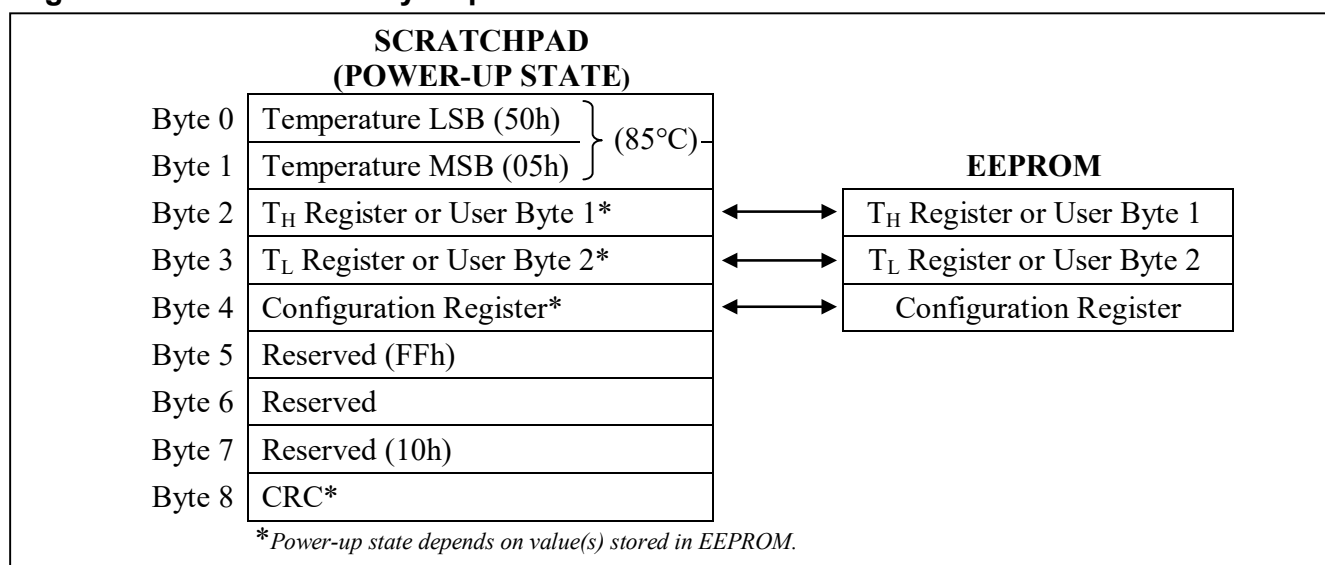
Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to  $T_H$  and  $T_L$  registers. Byte 4 contains the configuration register data, which is explained in detail in the *Configuration Register* section. Bytes 5, 6, and 7 are reserved for internal use by the device and cannot be overwritten.

Byte 8 of the scratchpad is read-only and contains the CRC code for bytes 0 through 7 of the scratchpad. The DS18B20 generates this CRC using the method described in the *CRC Generation* section.

Data is written to bytes 2, 3, and 4 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18B20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-Wire bus starting with the least significant bit of byte 0. To transfer the  $T_H$ ,  $T_L$  and configuration data from the scratchpad to EEPROM, the master must issue the Copy Scratchpad [48h] command.

Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall  $E^2$  [B8h] command. The master can issue read time slots following the Recall  $E^2$  command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done.

**Figure 7. DS18B20 Memory Map**



## CONFIGURATION REGISTER

Byte 4 of the scratchpad memory contains the configuration register, which is organized as illustrated in Figure 8. The user can set the conversion resolution of the DS18B20 using the R0 and R1 bits in this register as shown in Table 2. The power-up default of these bits is R0 = 1 and R1 = 1 (12-bit resolution). Note that there is a direct tradeoff between resolution and conversion time. Bit 7 and bits 0 to 4 in the configuration register are reserved for internal use by the device and cannot be overwritten.

**Figure 8. Configuration Register**

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

**Table 2. Thermometer Resolution Configuration**

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75ms	( $t_{\text{CONV}}/8$ )
0	1	10	187.5ms	( $t_{\text{CONV}}/4$ )
1	0	11	375ms	( $t_{\text{CONV}}/2$ )
1	1	12	750ms	( $t_{\text{CONV}}$ )

## CRC GENERATION

CRC bytes are provided as part of the DS18B20's 64-bit ROM code and in the 9<sup>th</sup> byte of the scratchpad memory. The ROM code CRC is calculated from the first 56 bits of the ROM code and is contained in the most significant byte of the ROM. The scratchpad CRC is calculated from the data stored in the scratchpad, and therefore it changes when the data in the scratchpad changes. The CRCs provide the bus master with a method of data validation when data is read from the DS18B20. To verify that data has been read correctly, the bus master must re-calculate the CRC from the received data and then compare this value to either the ROM code CRC (for ROM reads) or to the scratchpad CRC (for scratchpad reads). If the calculated CRC matches the read CRC, the data has been received error free. The comparison of CRC values and the decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS18B20 that prevents a command sequence from proceeding if the DS18B20 CRC (ROM or scratchpad) does not match the value generated by the bus master.

The equivalent polynomial function of the CRC (ROM or scratchpad) is:

$$\text{CRC} = X^8 + X^5 + X^4 + 1$$

The bus master can re-calculate the CRC and compare it to the CRC values from the DS18B20 using the polynomial generator shown in Figure 9. This circuit consists of a shift register and XOR gates, and the shift register bits are initialized to 0. Starting with the least significant bit of the ROM code or the least significant bit of byte 0 in the scratchpad, one bit at a time should be shifted into the shift register. After shifting in the 56th bit from the ROM or the most significant bit of byte 7 from the scratchpad, the polynomial generator will contain the re-calculated CRC. Next, the 8-bit ROM code or scratchpad CRC from the DS18B20 must be shifted into the circuit. At this point, if the re-calculated CRC was correct, the shift register will contain all 0s. Additional information about the Maxim 1-Wire cyclic redundancy check