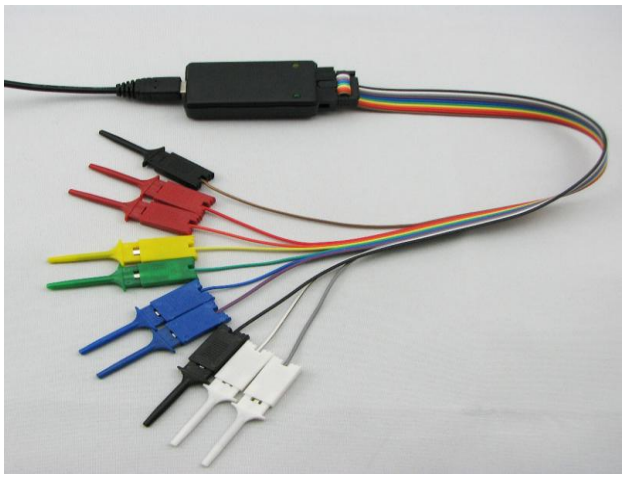
The Bus Pirate

SAMG **Nov 2015 by Peter Gheude**

Introduction to the Bus Pirate V3b hardware obtained from SandboxElectronics.com



The "Bus Pirate" created by Ian Lesnet is a universal bus interface, a Logic and Protocol Analyzer used to communicate with devices and IC's using a VT100 serial terminal.

The hardware is functionally identical with the official Dangerous Prototypes version 3b

The USB to serial converter chip in this design is a PL2303HX.

Many serial protocols are supported at 0-5.5volts and more can be added

1-Wire
I2C
SPI
JTAG
Asynchronous serial
MIDI
PC keyboard
HD44780 LCD
2- and 3-wire libraries with bitwise pin control
Scriptable binary bitbang, 1-Wire, I2C, SPI, and UART modes

Other Features

0-6volt measurement probe
1Hz-40MHz frequency measurement
1kHz – 4MHz pulse-width modulator, frequency generator
On-board multi-voltage pull-up resistors
On-board 3.3volt and 5volt power supplies with software reset
Macros for common operations
Bus traffic sniffers (SPI, I2C)
A bootloader for easy firmware updates
Transparent USB->serial mode10Hz-1MHz
SUMP compatible low-speed logic analyser
AVR STK500 v2 programmer clone
Supported in AVRDude programmer
Scriptable from Perl, Python, etc.

Realterm is a free terminal program compatible with ANSI- VT100 terminal protocol specially designed for capturing, controlling and debugging binary and other data streams. Serial Ports, USB Serial and TCP/IP & Telnet can be utilised. Also I2C, SPI, 1Wire chip control via BL233 / I2C2PC devices available from I2Cchip.com. The main port can be passed through or echoed to a TCP/IP or serial Echo Port. Realterm is used for testing the Bus Pirate on this occasion.

Bus modes, protocol libraries

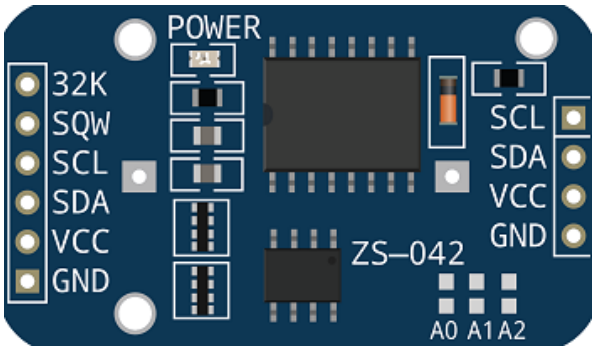
m <<<set mode command

There are many modes however we will use I2C for our example and other modes can be reviewed at another time.

```
HiZ>m
1. HiZ
2. 1-WIRE
3. UART
4. I2C
5. SPI
6. 2WIRE
7. 3WIRE
8. LCD
x. exit(without change)

(1)>4
Set speed:
 1. ~5KHz
 2. ~50KHz
 3. ~100KHz
 4. ~400KHz

(1)>4
Ready
I2C>W
POWER SUPPLIES ON
I2C>P
Pull-up resistors ON
I2C>

I2C>?
 General                                                Protocol interaction
 ------------------------------------------------------------------------------
 ?      This help                       (0)     List current macros
 =X/|X  Converts X/reverse X            (x)     Macro x
 ~      Selftest                        [       Start
 #      Reset the BP                            ]       Stop
 $      Jump to bootloader              {       Start with read
 &/%    Delay 1 us/ms                   }       Stop
 a/A/@  AUXPIN (low/HI/READ)            "abc"   Send string
 b      Set baudrate                    123
 c/C    AUX assignment (aux/CS)                 0x123
 d/D    Measure ADC (once/CONT.)        0b110   Send value
 f      Measure frequency               r       Read
 g/S    Generate PWM/Servo              /       CLK hi
 h      Commandhistory                  \       CLK lo
 i      Versioninfo/statusinfo          ^       CLK tick
 l/L    Bitorder (msb/LSB)              -       DAT hi
 m      Change mode                     _       DAT lo
 o      Set output type                         .       DAT read
 p/P    Pullup resistors (off/ON)       !       Bit read
 s      Script engine                   :       Repeat e.g. r:10
 v      Show volts/states               ;       Bits to read/write e.g. 0x55;2
 w/W    PSU (off/ON)            <x>/<x= >/<0>   Usermacro x/assign x/list all
```

```
I2C>i
Bus Pirate v3.b clone w/different PIC
Firmware v6.1 r1676  Bootloader v4.4
DEVID:0x044F REVID:0x3043 (24FJ64GA004 B5)
http://dangerousprototypes.com
CFG1:0xF9DF CFG2:0x3F7F
*----------*
Pinstates:
1.(BR)  2.(RD)  3.(OR)  4.(YW)  5.(GN)  6.(BL)  7.(PU)  8.(GR)  9.(WT)  0.(Blk)
GND     3.3V    5.0V    ADC     VPU     AUX     SCL     SDA     -       -
P       P       P       I       I       I       I       I       I       I
GND     3.25V   4.77V   0.00V   4.81V   H       H       H       H       H
POWER SUPPLIES ON, Pull-up resistors ON, Open drain outputs (H=Hi-Z, L=GND)
MSB set: MOST sig bit first, Number of bits read/write: 8
a/A/@ controls AUX pin
I2C (mod spd)=( 0 3 )
*----------*
```

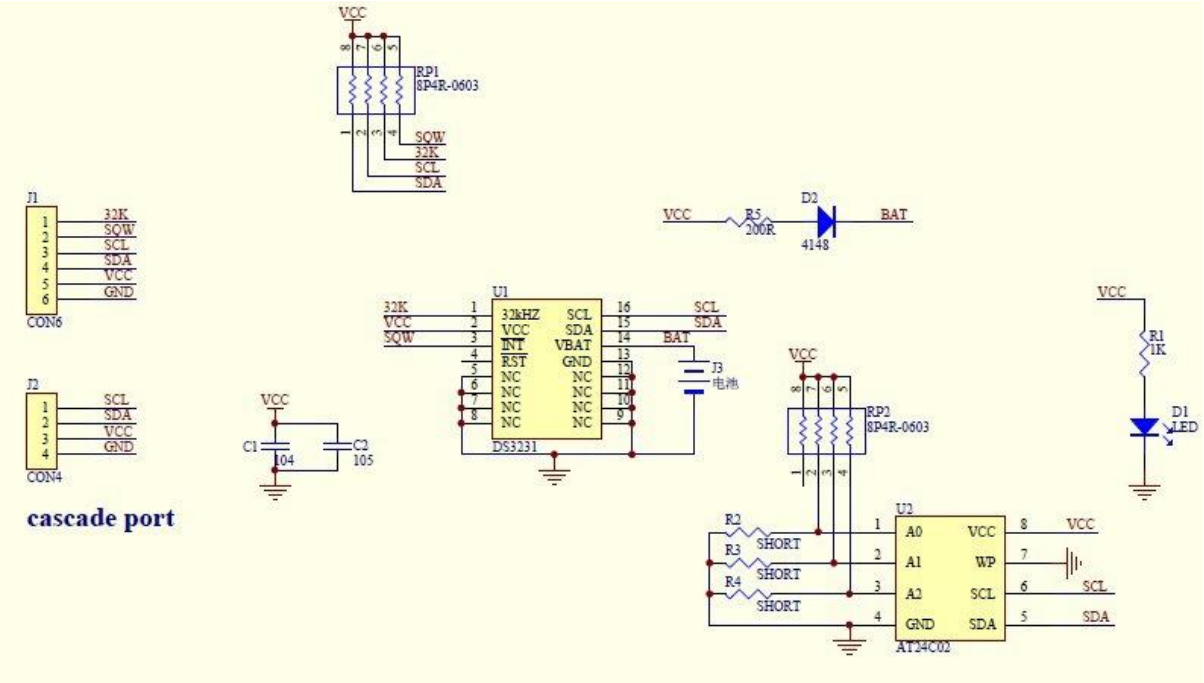
The DS3231 RTC Board and M24C02 2k-bit (256 x 8) eeprom taken from a old copier drum cartridge is used to demonstrate the Bus Pirate, making three devices on the I2C bus.

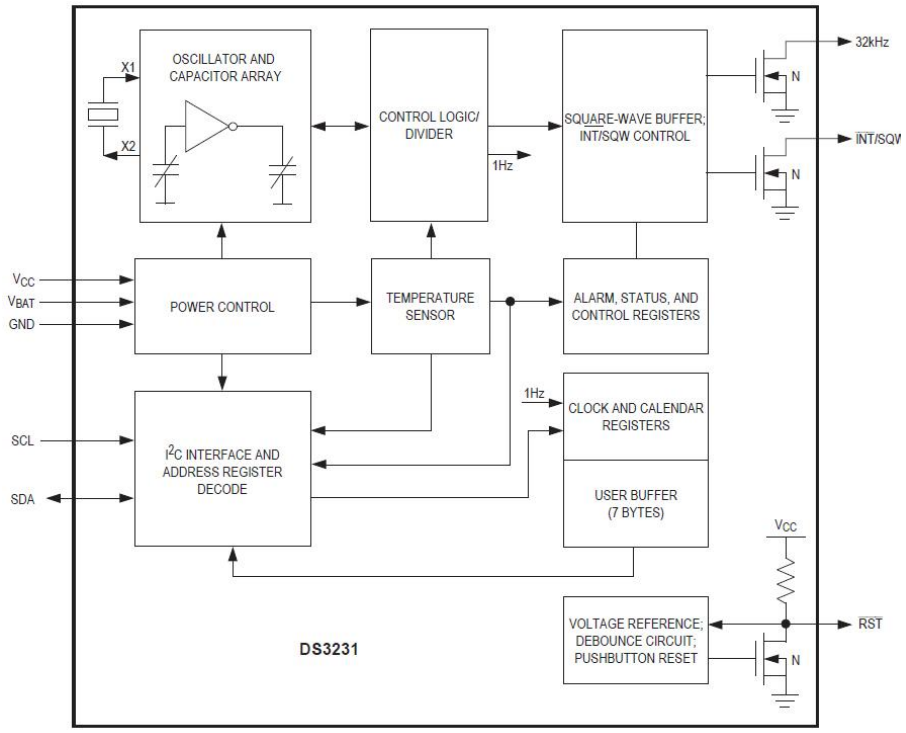


	HiZ	1-Wire	UART	I2C	SPI	JTAG
MOSI		OWD	TX	SDA	MOSI	TDI
CLK			SCL	CLK	TCK	
MISO			RX		MISO	TDO
CS					CS	TMS

AUX	Auxiliary I/O, freq. probe, PWM
Vpu	Input pull-up resistors (0-5V)
ADC	A/D converter,max. 6V,10bit,500ksps
5V,3.3V	Switchable supply, max. 150mA
GND	Ground to test circuit

bus-pirate reference card, dangerousprototypes.com, V1.0





DS3231 RTC

Real-Time Clock Counts Seconds, Minutes, Hours, Day, Date, Month, and Year with Leap Year Compensation Valid Up to 2100
Contains a temperature compensated quartz tuning-fork crystal
Two Time-of-Day Alarms
Programmable Square-Wave Output
3.3V Operation
Digital Temp **Sensor**
Output: ±3°C Accuracy

Bus Interaction Syntax

A simple syntax is used to interact with chips. Syntax characters have the same general function in each bus mode, such as 'R' to read a byte of data.

[0x31 r:5]

This example syntax sends a bus start , the value 0x31, and then reads 5 bytes, followed by bus stop. Up to 255 characters of syntax may be entered into the Bus Pirate terminal at once, press enter to execute the syntax.

Terminal Control

The Bus Pirate understands some VT-100 (ANSI C0) terminal emulation.

Keyboard Key	Ctrl-Key	Action
[left arrow]	^B	Moves the cursor left one character
[right arrow]	^F	Moves the cursor right one character
[up arrow]	^P	Copies the previous command in the command history buffer to the command line
[down arrow]	^N	Copies the next command in the command history buffer to the command line
	^A	Moves the cursor to the beginning of the line

	^E	Moves the cursor to the end of the line
[backspace]	^H	Erases the character to the left of the cursor and moves the cursor left one character
[delete]	^D	Erases the character under (or to the right of) the cursor and moves the cursor left one character

Macros

I2C>(0)
0.Macro menu
1.7bit address search
2.I2C sniffer

I2C>(1)

Macros perform complex actions, like scanning for I2C addresses, interrogating a smart card, or probing a JTAG chain. Macros are numbers entered inside (). Macro (0) always displays a list of macros available in the current bus mode.

The 7bit address search macro below confirms that there are three I2C devices on the I2C bus. A M24C02 2k-bit (256 x 8) eeprom (0xA0) and a ZS-042 DS3231 (0xD0) and AT24C02 (0xAA). Note that 0xD0 is the Write Address and 0xD1 is the Read Address for the DS3231 RTC

```
I2C>(1)
Searching I2C address space. Found devices at:
0xA0(0x50 W) 0xA1(0x50 R) 0xAA(0x55 W) 0xAB(0x55 R) 0xD0(0x68 W) 0xD1(0x68 R)


I2C>[0xd0 0x0 [0xd1 r:19]
I2C START BIT
WRITE: 0b11010000 ACK
WRITE: 0b00000000 ACK
I2C START BIT
WRITE: 0b11010001 ACK
READ: 0b00011001  ACK 0b01000111  ACK 0b00011000  ACK 0b00000011  ACK 0b00010000
  ACK 0b00000001  ACK 0b00000000  ACK 0b00000000  ACK 0b00000000  ACK 0b00000000
  ACK 0b00000000  ACK 0b00000000  ACK 0b00000000  ACK 0b00000000  ACK 0b00011100
  ACK 0b10001000  ACK 0b00000000  ACK 0b00011100  ACK 0b10000000
NACK
I2C STOP BIT
I2C>
```

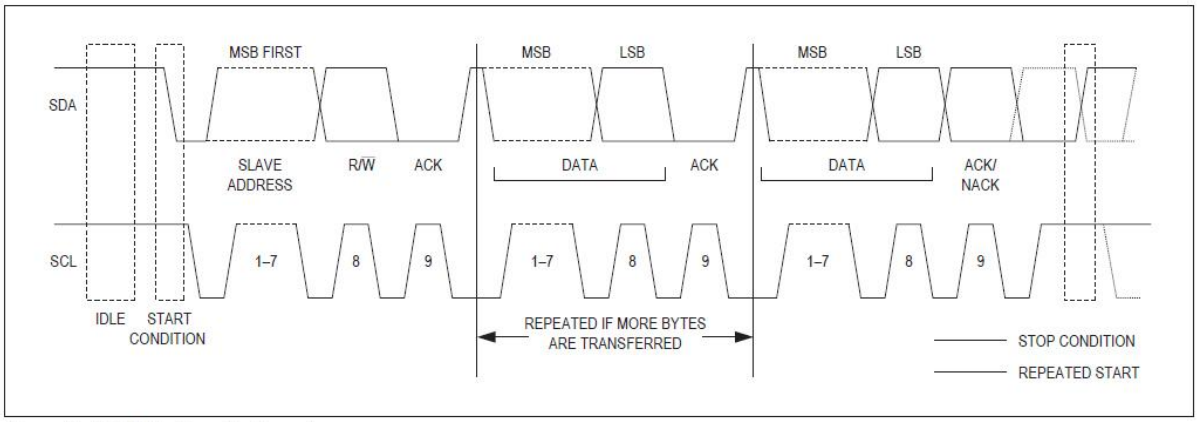


Figure 2. I²C Data Transfer Overview

I2C Serial Data Bus

The DS3231 supports a bidirectional I2C bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data is defined as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are slaves. The bus must be controlled by a master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS3231 operates as a slave on the I2C bus. Connections to the bus are made through the SCL input and open-drain SDA I/O lines. Within the bus specifications, a standard mode (100kHz maximum clock rate) and a fast mode (400kHz maximum clock rate) are defined. The DS3231 works in both modes.

The following bus protocol has been defined (Figure 2):

Data transfer may be initiated only when the bus is not busy.

During data transfer, the data line must remain stable whenever the clock line is high. Changes in the data line while the clock line is high are interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain high.

START data transfer: A change in the state of the data line from high to low, while the clock line is high, defines a START condition.

STOP data transfer: A change in the state of the data line from low to high, while the clock line is high, defines a STOP condition.

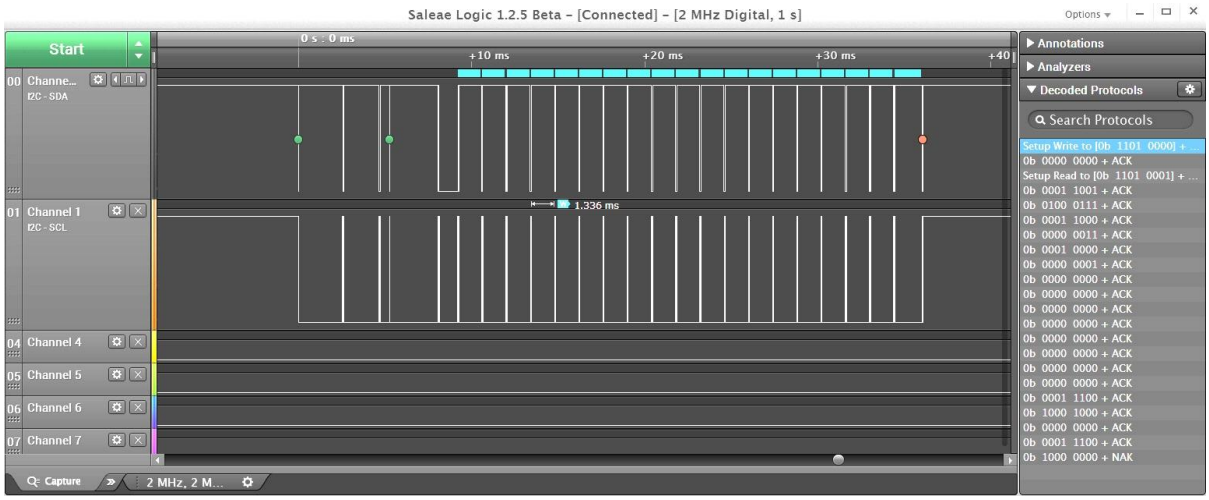
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the high period of the clock signal. The data on the line must be changed during the low period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between the START and the STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit.

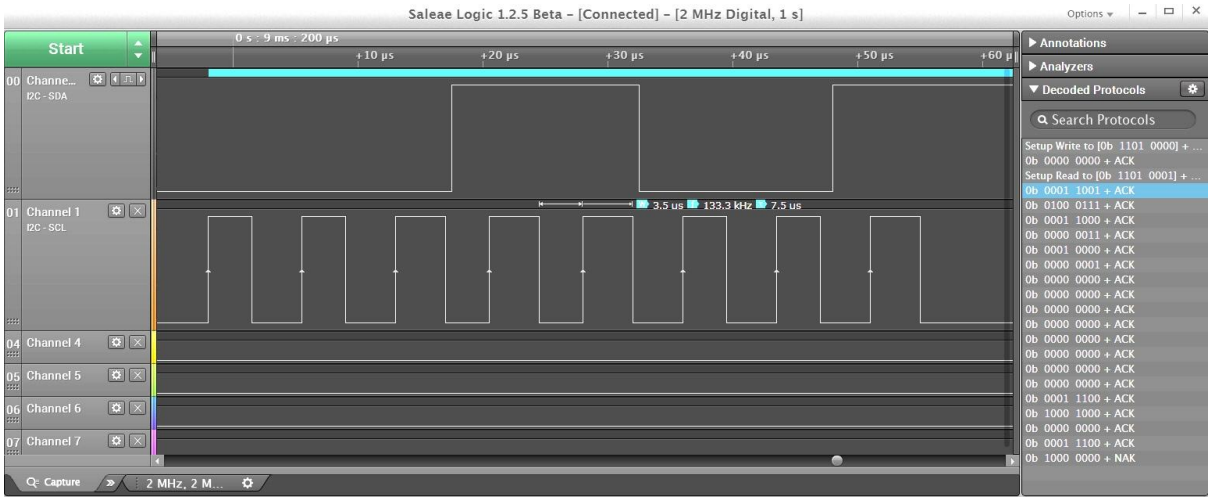
Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse, which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable low during the high period of the acknowledge-related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line high to enable the master to generate the STOP condition.

Captured IC2 logic time sequence diagrams – DS3231 19 Registers read



Captured IC2 logic time sequence diagrams – DS3231 Register 0x0 0b00011001 = 19 sec



ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

DS3231 – Extremely Accurate I2C-Integrated RTC/TCXO/Crystal – Registers

Registers (00h–10h)

Time and Date represented as Binary Coded Decimal format. (BCD)

Temperature Registers (11h–12h)

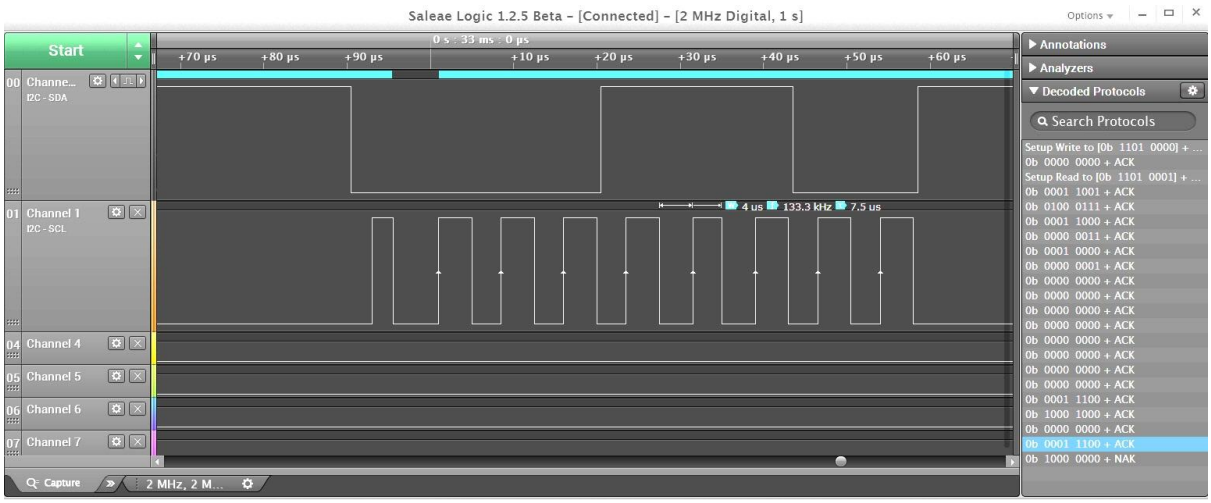
Temperature is represented as a 10-bit code with a resolution of 0.25°C and is accessible at location 11h and 12h. The temperature is encoded in two's complement format. The upper 8 bits, the integer portion, are at location 11h and the lower 2 bits giving four ¼ degree values, the fractional portion, are in the upper nibble at location 12h.

0b00=00 0b01=0.25 0b10=0.5 0b11=0.75

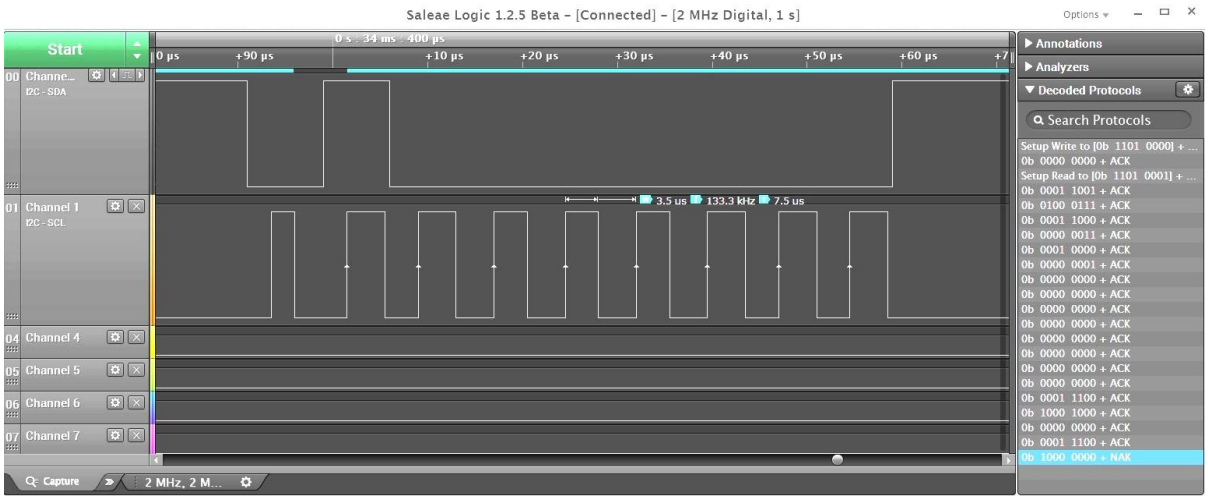
For example taken from captured data below,

Reg 11-> 0b00011100 + Reg 12->0b10000000 = +28. 5°C.

IC2 logic time sequence diagrams – DS3231 Register 0x11 0b00011100 = 6+8+4 = 28 Degs



Captured IC2 logic time sequence diagrams – DS3231 Register 0x12 0b10000000 = 0.5 deg



Information used in this tutorial / review sourced from Maximum Integrated DS3231 documentation, maximumintegrated.com, sandboxelectronics.com, dangerousprototypes.com websites and real-time operation using a Saleae logic analyser.

Note: An external 10k Pull-up resistor on the M24C02 pin 7 attached to Bus Pirate's AUX pin as input. Toggle auxiliary pin with Capital "A" sets AUX high, small "a" sets low and @ sets aux to input (high impedance mode) and reads the pin value.

Relevant Links;
http://dangerousprototypes.com/docs/Bus_Pirate
http://sandboxelectronics.com
http://www.maximintegrated.com
http://realterm.sourceforge.net