Serial Communications / Protocol in AirTest Products

General

The standard sensor has two different ways of serial communication with sensor's peripherals and the outside world. One is through the *UART (Universal Asynchronous Receiver and Transmitter)* and another through the *Microwire bus*. Basically, they differ in such a way that the sensor communicates via *UART* only when it is requested, but only *in-between* sensor measurements, whereas the *Microwire bus* is always used to send information after each measurement cycle as signalled by a *chip select* hardware wire. This latter bus is intended for communications with internal system functions and plug-in-cards in a *slave mode* using a *chip select* signal from the sensor microprocessor. One such plug-in-card is the *-LON* option, which connects an *Echelon neuron* and turns the system into a *LonMark* labelled *CO_2-Temp-Occupancy* triple sensor for advanced digital networks.

The *UART* communication protocol, general description

The intended UART communication protocol is a master/slave one. Communication can be initiated only by one node of the network, the master of the net. Sensor is always a slave. Baud rate is 9600, other settings are 8 data bits, 1 stop bit, no parity.

The *UART* protocol is intended for both RS232 and RS485 communication with other master. The high accuracy of the sensor requires the measurements to have the top priority. Hence, the total cycle is divided into two sequential time slots - measurement and communication. During the measurements the *UART* communication is disabled, and it is reinitialised at the beginning of communication time slot. Sensor can stay in measurement time slot for as long as 0.8 sec.

The communication protocol is written in such a way that any communication starts externally by sending to the sensor's *UART* one **Request To Send** (RTS) byte. Master must provide 8.2 *msec* silence on the communication bus **before** and **after** RTS in order to let slave to identify that the byte should be interpreted as RTS. In addition master shall give a sensor at least 21.5 *msec* after RTS to prepare answer.

Received during measurement time slot bytes are ignored.

As soon as the RTS byte is received and successfully identified, the slave will respond by sending a **Clear To Receive** (CTR) byte.

The master has to answer after a time-out period of minimum 8.2 msec counted from the last stop bit has been shifted into master's UART to the first start bit transition of shifting out byte by sending the **Message Request** (MR), that contains up to a maximum of 8 bytes. These8 bytes include task code and check sum. The time interval between bytes must not exceed 4 *msec* in order to let slave to distinguish RTS and data bytes. If more data is to be exchanged, a new RTS has to be sent for each 8 byte package.

If the sensor receives and understands the MR message, the sensor *UART* will respond by sending a **Data Code** (DC) containing the data requested, that contains up to a maximum of 8 bytes. These8 bytes include **Acknowledgement** (ACK) and check sum. In case of an execution command, only an **Acknowledgement** (ACK) will be sent after command execution. Slave must provide time interval between bytes to be less than 4 *msec* to prevent interpretation data bytes as RTS by any other slave in the RS485 network.

Any detected communication error (frame errors, check sum mismatch etc.) as well as detected application error (invalid task code, run time error of execution commands etc.) results in reinitialising of sensor communication without any report on the communication bus (no NACK). Master can detect the presence of sensor in the net due to CTR response. After execution command fail master shall check sensor status.

A new RTS can be sent immediately after the completion of a DC string (+8.2 msec). Because of the presence of measurement time slot with forbidden communication, the duration of a single communication session must be limited to 120 *msec*. Slave must not send CTR if there is no time to complete communication before measurements. Master has to try to send new RTS if it doesn't get CTR for this time period.

Because of the fact that measurement time slot can be as long as 0.8 sec, master has to try to send RTS several times (e.g. 10 attempts to establish communication by sending RTS with repetition period of 100 *msec*) before the conclusion of slave failure.

In addition to general communication sequence, described above, a special **broadcast command** is used to reset individual network address of the sensor to default value = 0. **Request To Send** (RTS) byte of broadcast command shall not be responded by CTR. Because of measurement time slot, broadcast command shall be repeated several times in order to ensure receiving by all sensors in the network. Usually network address reset is followed by procedure of reassigning of network addresses. See special chapter bellow for sensor algorithm description.

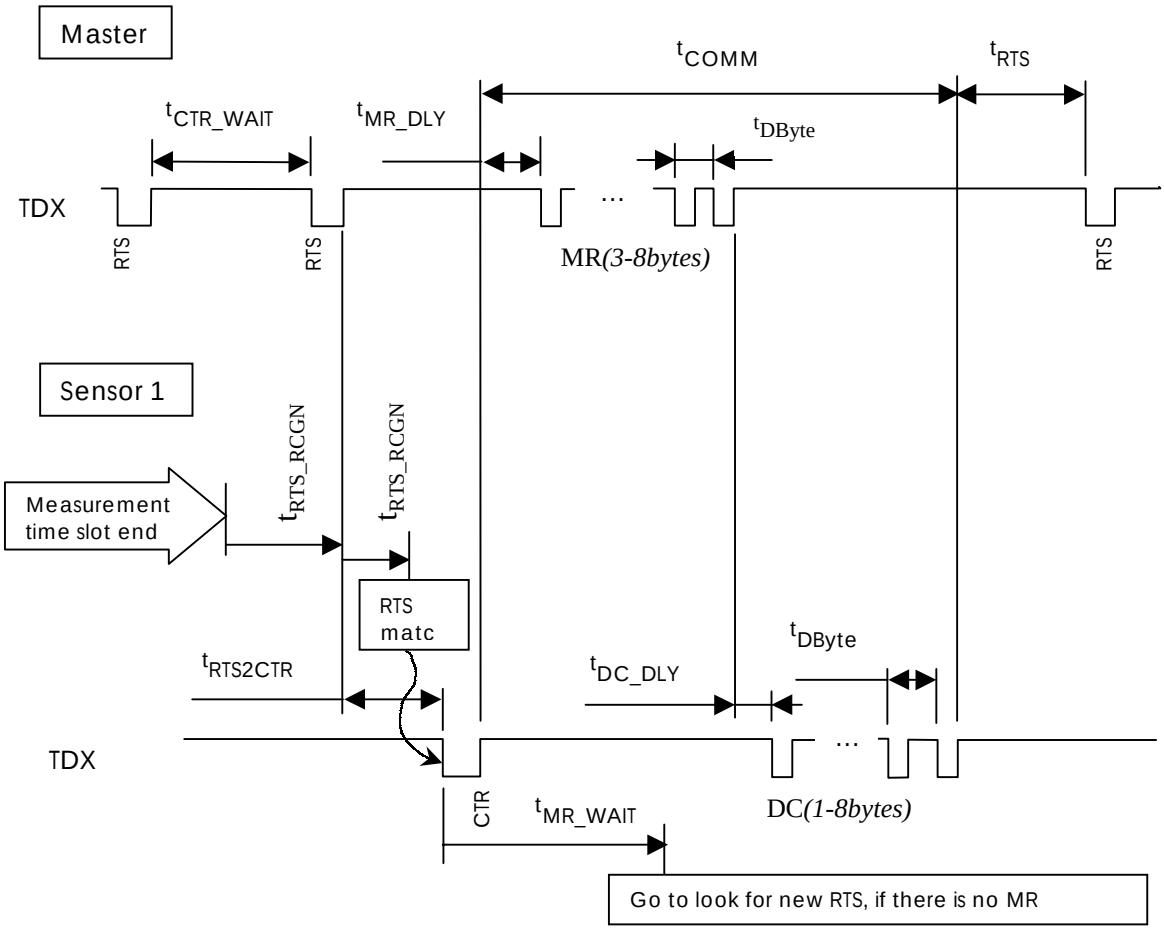
The *UART* communication protocol parameters summary

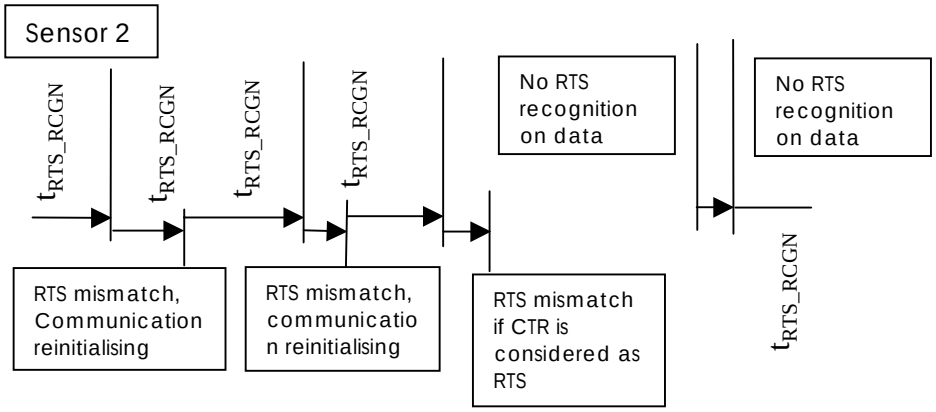
Parameter	minimum	typical	maximum	Notes
Baud rate		9600		
Data bits		8		
Stop bit		1		
Parity		No		
t_{CTR_WAIT}	$t_{RTS2CTR}$ maximum + 1.5 msec	100 msec		Typical value represents recommended one
t_{MR_DLY}	8.2 msec		20 msec	2
t_{RTS}	8.2 msec			
t_{COMM}			150 msec	3, WARNING 4
t_{DByte}			4 msec	Requirement for master.
$t_{RTS2CTR}$	8.2 msec		20 msec	
t_{DC_DLY}	8.2 msec		**80 msec**	1, WARNING 4 Maximum is changed in version 3.04
t_{RTS_RCGN}	4.1 msec	8.2 msec		Values for reference and sensor's operation explanation only
t_{MR_WAIT}		80 msec		Values for reference and sensor's operation explanation only. Sensor restarts to wait for new RTS if it doesn't receive total MR for t_{MR_WAIT}

NOTES:

1. For half duplex RS485 network minimum value implies requirement for master to leave bus for 8 msec after byte transmission is completed.
2. For half duplex RS485 network minimum value implies requirement do not enter the control of the bus for 8.2 msec. It is maximum time required to return sensor's RS485 transmitter into receive mode. Maximum value is a requirement for master and is not checked by sensor, see t_{MR_WAIT}
3. Maximum parameter shall be used to check communication operation.
4. WARNING! Generally, each command has its own execution time. Numbers in table cover most typical commands, but two exceptions; "Save sensor RAM fields in EEPROM" and "Reset TWA (Time Weighted Average) stack". They can have execution time as long as 3 seconds.

Transmit timing chart (except broadcast command):





Byte declaration (<hex> values in brackets):

RTS – Request To Send

<FE>	*Valid to all sensors, disregarding Network Address (INA) value.*	For use in RS 232 communication, **Never use in RS485 networks!!!**
<E0>	*Valid to all "un-named" sensors, which have default INA = 0.*	**Notice, that use of this address is risky in RS485 networks!!!**
<E1...FD>	*Valid only to an individual sensor.*	

The value <FE> holds for all sensors. In addition, each unit may be assigned an *individual network address* (INA) in the range *1...29*, in which case a *unit selective* RTS is possible by sending the byte value <E0+INA>. Sensors not assigned with an INA also respond to <E0>.

CTR – Clear To Receive

<DF>	Sensor is ready to receive MR	
<DD>	*Enter slave mode*	*not yet implemented*

MR – Message Request: <task code>,<address>,(<address>),(<data>,....,)<chksm>

The *message request* consists of 3 mandatory and one optional field. Message Request can be 3 to 8 bytes long, depending on the amount of data to be transferred.

A pure request for data (bit 7 in<task code> =1) is always 3 bytes long:

<task code>,<address>,<chksm>

Description of fields follows:

<task code> - one byte mandatory field

bit 7	bit 6	bit 5	bit 4	bit 3	Bit field 2..1..0	Check	Description
x	x	x	x	x	0 0 0	Reserved	Is interpreted as communication error by present version, no NACK
Standard Commands (read/write RAM)						bit 7	0 = *write* to RAM, 1 = *read* from RAM
						bit 6	0 = *lower* RAM , 1 = *upper* RAM
0	0	0	0	0	Number of data bytes to be written (not 0)	=< 5	Write to lower RAM
0	1	0	0	0			Write to higher RAM
1	0	0	0	0	Number of data bytes to be read (not 0)	=< 6	Read from lower RAM
1	1	0	0	0			Read from higher RAM

bit 7	bit 6	bit 5	bit 4	bit 3	Bit field 2..1..0	Check	Description
Special commands (read/write EEPROM)							
0	0	0	0	1	Number of data bytes to be written (not 0)	=< 4	Write to EEPROM.
1	0	0	0	1	Number of data bytes to be read (not 0)	=< 6	Read from EEPROM.

Note:

1. Read/write EEPROM commands are used with 2-byte address. First address byte in MR represents high byte of address, second address byte in MR represents low byte of address.
2. EEPROM is byte-organised in pages of 16 byte each. First address of each page is given by XXX0h, last address of the page is given by XXXFh. Write command MUST write into ONE page at the time ONLY! Otherwise the first addresses of current page will be overridden.

bit 7	bit 6	bit 5	bit 4	bit 3	Bit field 2..1..0	Check	Description
Execution commands							
1	0	0	1	1	0 0 1	<address> = 0ADh	Deleted in 060xxx compared to 050xxx, no answer from sensor.
1	0	1	0	0	0 0 1	<address> = 0	Save sensor INA in EEPROM
1	0	0	1	0	0 0 1	<address> = 0	Copy EEPROM variables into RAM
1	0	1	0	1	0 0 1	<address> = AB	Lock EEPROM, see algorithm description in Appendix B.
1	0	1	1	0	0 0 1	<address> = A9	Unlock EEPROM, see algorithm description in Appendix B.
1	0	1	1	1	0 0 1	<address> = 01	Not implemented yet! Functions, number of function is written in address byte. 01 - Reset TWA (Time Weighted Average) stack.

Notes:

1. Any deviation of checked parameter results in recognition of communication error and sensor communication reinitialising. Sensor starts to look for new RTS.
2. Any unrecognised task code results in recognition of communication error and sensor communication reinitialising. Sensor starts to look for new RTS.

<address> - one byte mandatory field, *start address* to read/write. High (MSB) byte of address in 2-byte address for EEPROM read/write commands.

(<address>) - one byte optional field, *start address* to read/write. Low (LSB) byte of address in 2-byte address for EEPROM read/write commands.

(<data>,...) – optional field, *data*, 5 bytes maximum for one byte addressed commands, 4 bytes maximum for 2-byte addressed command write to EEPROM.

<chksm> - one byte mandatory field, *lsb* of a pure summation of the previous MR bytes

DC – Data Code <ACK>,(<data1>,(<data2>,....),<chksm>)

The DC string always starts with a communication status byte ACK. If data are requested, the DC string is 3 to 8 bytes long, depending on the amount of data. The DC code fields description is the following:

<ACK> - Acknowledgement, - one byte mandatory field, can be one of following two:
<06> - indicates that MR was successfully received and/or command was successfully executed.
<86> = <06> + bit 7 set = MR successfully received and/or command was successfully executed +*sensor event flag* is set

(<data1>,(<data2>,....) – optional field, *data*, 6 bytes maximum.

<chksm> - if data field presents, it is one byte mandatory field, *lsb* of a pure summation of the previous DC bytes.

Transmit example #1 Collecting the present sensor *temperature T and* CO_2 readings (*-hi/-lo* refers to *msb* and *lsb* of the 2-byte numbers):

external computer Tx---<FE>----------<84><14><98>----------
sensor *UART* Tx----------<DF>----------<06><Thi><Tlo><CO_2hi><CO_2lo><chksm>-----

The sensor hardware pin R/T will be in *low* logic level *(receive)* at all times, except for the short periods of *UART* transmit time when it will go *high*. Standard baud rate is 9600 b/s.

Transmit example #2 Collecting the present sensor CO_2 reading from a sensor that has detected some hardware error:

external computer Tx---<FE>----------<82><16><98>----------
sensor *UART* Tx----------<DF>----------<86><CO_2hi><CO_2lo><chksm>----------

UART communication options

accessory A232 - *hardware & software for temporal PC/computer-sensor connection*

This kit is primarily for use during sensor installations, reconfigurations, and maintenance and is included together with necessary PC software in the User Interface Program *VT (STRATEGY).*

option -485 add-on-PCB - *for fixed installation to external computer using the RS485*

Some useful *UART* commands	**MR - com. bytes**	**DC - sensor response**
Standard commands		
Collecting Temperature and CO_2 values	<84><14><98>	<ACK><Thi><Tlo><CO_2hi><CO_2lo><chksm>
Collecting CO_2 values	<82><16><98>	<ACK><CO_2hi><CO_2lo><chksm>
Collecting Temperature values	<82><14><96>	<ACK><Thi><Tlo><chksm>
Collecting Error Code	<81><43><A8>	<ACK><Error Code><chksm>
Collecting Individual Network Address	<C1><50><11>	<ACK><INA><chksm>
Assigning Individual Network Address	<41><50><INA> <chksm>	<ACK>
Reading Alarm Status byte, Bit #7 contains copy of DT active flag. This position of DT active flag is supposed to be supported for compatibility with future devices.	<C1><7F><40>	<ACK><AlarmStatus><chksm>
Special command		
Read from EEPROM		
Execution commands		
Save sensor INA in EEPROM	<A1><00><A1>	<ACK>
Save sensor RAM in EEPROM	<99><AD><46>	<ACK>
Copy EEPROM variables into RAM	<91><00><91>	<ACK>
Reset TWA (Time Weighted Average) stack	<B9><01><BA>	<ACK>

Note:

1. *In this chart Temperature and CO_2 values equals parameter 1 & 2 (channel 0 and channel 1) in the general representation.*

2. ***Notice, that immediately after powering up (or restarting after power fail) correct data is not available from the sensor. Reasonable data on measured parameters like temperature or carbon dioxide concentration appears in 3..7 seconds after power up and data of specified precision is available after specified warm-up time. Data on TWA of carbon dioxide becomes available in one minute after powering up.***

Appendix A: *UART* production commands and compatibility with old 050xxx rev 3.05 protocol.

***UART* commands, (both 050xxx, 060xxx and 0700xx platforms).**	**MR - com. bytes**	**DC - sensor response**	**Status**
Standard commands			
Collecting Temperature and CO_2 values	<84><14><98>	<ACK><Thi><Tlo><CO_2hi><CO_2lo><chksm>	Not Changed!
Collecting CO_2 values	<82><16><98>	<ACK><CO_2hi><CO_2lo><chksm>	Not Changed!
Collecting Temperature values	<82><14><96>	<ACK><Thi><Tlo><chksm>	Not Changed!
Collecting AN1 analogue output values	<C2><45><07>	<ACK><AN1hi><AN1lo><chksm>	Address is changed!
Collecting AN2 analogue output values	<C2><47><09>	<ACK><AN2hi><AN2lo><chksm>	Address is changed!
Collecting Error Code	<81><43><C4>	<ACK><Error Code><chksm>	Address is changed!
Collecting Sensor PART Number	<81><41><C2>	<ACK><PART><chksm>	Not Changed!
Collecting Individual Network Address	<C1><50><11>	<ACK><INA><chksm>	Not Changed!
Assigning Individual Network Address	<41><50><INA><chksm>	<ACK>	Not Changed!
Reading Alarm Status byte, Bit #7 contains copy of DT active flag. This position of DT active flag is supposed to be supported for compatibility with future devices.	<C1><7F><40>	<ACK><AlarmStatus><chksm>	??? Is supposed to be implemented.
Reading DIG1 timer DT	<C1><3B><FC>	<ACK><DT><chksm>	???
Reading DT active flag (=bit #7 set)	<81><0E><8F>		Not available at old address, see “Reading alarm status byte”.

***UART* commands, (both 050xxx and 060xxx platforms).**	**MR - com. bytes**	**DC - sensor response**	**Status**
Special command			
Write to EEPROM	<09><00><10><01><1A>	<ACK>	Added and changed address mode
Read from EEPROM	<89><00><10><99>	<ACK><Sensor ID><chksm>	Added and changed address mode

***UART* commands, (both 050xxx, 060xxx and 0700xx platforms).**	**MR - com. bytes**	**DC - sensor response**	**Status**
Execution commands			
Save sensor INA in EEPROM	<A1><00><A1>	<ACK>	
Save sensor RAM in EEPROM	<99><AD><46>	<ACK>	**Not available in 060xxx or 0700xx!**
Copy EEPROM variables into RAM	<91><00><91>	<ACK>	
Lock EEPROM	<A9><AB><54>	<ACK>	Added in 060xxx and 0700xx, see algorithm bellow.
Unlock EEPROM	<B1><A9><5A>	<ACK>	Added in 060xxx and 070xx, see algorithm bellow.
Reset TWA (Time Weighted Average) stack	<B9><01><BA>	<ACK>	Not yet available in 060xxx or 0700xx!

***UART* commands, (both 050xxx, 060xxx and 0700xx platforms).**	**MR - com. bytes**	**DC - sensor response**	**Status**
Broadcast commandsd			
Reset network address	<DE>…<DE>… <DE>		**Not available in 060xxx or 0700xx!**

Appendix B: Lock/Unlock EEPROM algorithm description.

Generally, EEPROM itself is design to provide maximum data security. After power up it comes to write protected mode. But in order to have additional protection against accidental write to EEPROM due to, for example, uC program error, some portion of EEPROM can be protected from write by means of additional programming of EEPROM. Protected in such way portion can not be programmed even after ordinal write enable command, but additional reprogramming is required.

General EEPROM protection sequence:

1. After power-up uC writes LOCK command, protecting portion of EEPROM against write.
2. It repeats LOCK commands every 4 hours (to be specified).
3. For calibration purposes it is necessary to enable write operation. Computer have to send command UNLOCK. Notice, that internal LOCK command is random. It may happen at any time after computer has sent UNLOCK command. That's why it is necessary to read EEPROM back after write operation to check if write was accomplished.