USB Signal Conditioner
Metrolog
SD20

User guide and technical reference

Metrolog SD20

User guide and technical reference

Version 2.0 – 03/19

For us2 with Metrolog SD20 devices with v2.0+ firmware

Safety information

1. Read all manual before using this equipment.

2. Turn it off before cleaning; use only slightly wet cloth for cleaning.

3. Do not install it close to heavy machinery with exposed running liquids.

4. Do not install it near strong heat sources.

5. To install or remove a connector hold firmly by its body. Never pull or apply force by the cable.

6. Except when described on this manual, do not try to repair this equipment. Opening it without the proper training and knowledge can lead to electrical accidents. Please inquire Metrolog about troubleshooting and preventive maintenance.

7. Unplug the equipment and send it for repairs if one of these situations occur:

    A. Cables were damaged or show burning/melting marks;
    B. It was exposed or submersed in liquid;
    C. It was exposed to the weather/rain;
    D. It does not perform as described on this manual;
    E. Damage to its enclosure was detected due to accidental fall or misuse.

CAUTION

RISK OF ELECTRICAL SHOCK
DO NOT OPEN

WARNING: TO AVOID EXPOSURE TO ELECTRICAL SHOCK DO NOT REMOVE OR OPEN THE EQUIPMENT ENCLOSURE.
INSIDE THE DEVICE THERE IS NOT ANY COMPONENTS THAT CAN BE REPLACED OR REPAIRED BY THE USER. REPAIRS MUST BE CONDUCTED ONLY BY QUALIFIED TECHNICIANS.

Warranty

This equipment has a warranty of 6 months from the invoice expedition date. During this period it may be repaired without any part or labor costs.

This warranty will be void if detected damage results of misused, fall, mechanical shock, incorrect cable connection, use by untrained personal, fire, floods or other accidents.

Only Metrolog's technical personnel will be eligible for analyzing any repair requests and void warranty, if applicable.

All warranty repairs will be conduct at Metrolog laboratory. Customer is responsible for shipment costs.

Information on this manual is subject to change without prior notice.

Summary

I. Technical Specifications

Communication Interface	• USB (Universal Serial Bus) 2.0
Linearization	• Look-up table linearization (LUT) up to 524288 reference points. Linear interpolation inter-points. • Curve modeling and data set generation by SD20ConfDiag software.
LVDT signal conditioner (SD20-LVDT model only)	• Low harmonic distortion sine wave oscillator (5kHz) for primary coil excitation. Factory adjustable amplitude based on transducer requirements (from 1.7 to $5V_{RMS}$) • Ratiometric signal processing with low thermal drift. • 500Hz bandwidth.
Analog to Digital converter	• High performance 24-bit A/D converter with high precision voltage reference. • User selectable primary sample rate from 6.8 to 880 samples per second.
Data transfer rate	• Variable data rate from 6.8 up to 880 read per second (based on primary and secondary filter parameters)
Signal threshold	• 2 user adjustable limits (lower and upper values).
Zero set	• Digital, set by software or digital signal.
Digital Input/output interface	• Input interface: 3 photo-coupled inputs with programmable functions. • Output interface: 2 open-collector outputs (with internal pull-up) with programmable functions.
Power supply	• 4.5 to 5.5V, 400mA, driven directly from the USB bus.
Temperature	• -10ºC to 70ºC (storage) • 10ºC to 50ºC (working conditions)
Protection class	• IP50
Dimensions	• 116 x 80 x 28 mm
Weight	• 115g

1. Connectors and electrical diagram

The SD20-LVDT and SD20-Analog have 3 frontal connectors, as show in Figure 1.

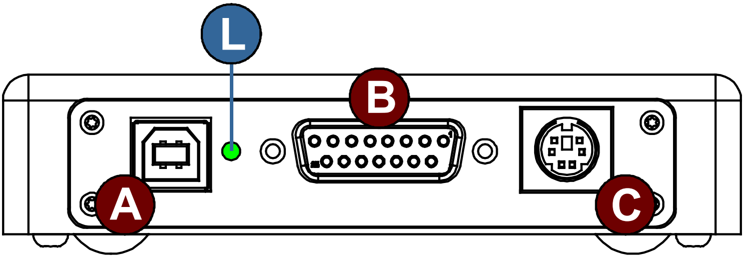

Figure 1 – Frontal view of SD20 conditioner – Connectors and status led

A – Female USB connector type “B”

B – Female DB15F connector– Transducer interface

C – 6 pins Mini-DIN connector – Digital Input/output ports

L – Status *Led*

1.1 USB Connector

Figure 1, highlight **A**, shows a “B” type USB connector used for data transmission and for powering the device.

To connect to a computer use the supplied type “AB” USB cable.

It is important to note that a 400mA capable USB port is required. Use a powered USB HUB or connect the device direct to the computer USB port. Do not use passive unpowered USB HUBs.

In case the SD20 is used without a computer (as a go/no-go device), use a USB compatible power supply (with type “A” connector, 5V, 500mA rating).

1.2 Transducer interface connector

Figure 1, highlight **B**, shows a female DIN DB15F connector used to connect an external transducer to the conditioner analog input port. The electrical diagram for the SD20-LVDT model is shown in Figure 2; the electrical diagram for SD20-Analog model is shown in Figure 3.

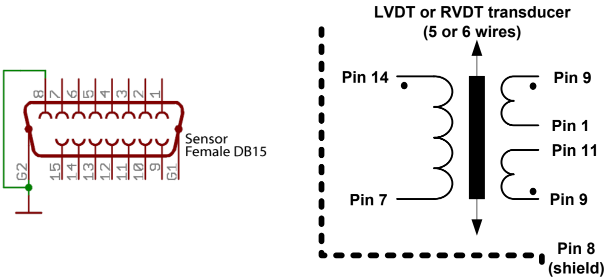

Figure 2 – Connection Diagram – SD20-LVDT model

Pin	Description
1	LVDT secondary winding (A)
2	Reserved (calibration resistor R_{OSC} pin A)
3	Connected to pin 9
4	Reserved (calibration resistor R_{GAIN} pin A)
5	N/C
6	N/C
7	LVDT primary winding
8	LVDT cable shield / GND
9	LVDT secondary winding (A/B, center-tap)
10	Reserved (calibration resistor R_{OSC} pin B)
11	LVDT secondary winding (B)
12	Reserved (calibration resistor R_{GAIN} pin B)
13	N/C
14	LVDT primary winding
15	+15VDC
Body	GND

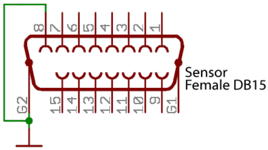

Figure 3 – Connection diagram–SD20-Analog model

Pin	Description
1	N/C
2	N/C
3	Connected to pin 9
4	Analog Input IN+
5	N/C
6	N/C
7	N/C
8	Cable Shield / GND
9	Analog Input IN-
10	N/C
11	N/C
12	N/C
13	N/C
14	N/C
15	+15VDC (transducer power supply)
Body	GND

1.3 Digital Input/output Port Connector

Figure 1, highlight **C**, shows a female Mini-DIN 6 pin connector used for digital input/output signal interface.

The device has 2 digital output ports, S1 and S2, and 3 digital input ports, E1, E2 and E3. Electrical diagram is shown in Figure 4.

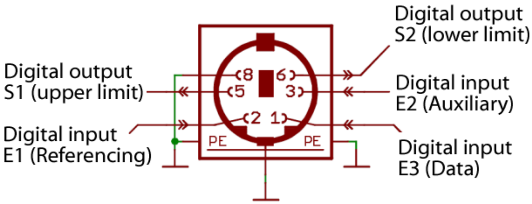

Figure 4 – I/O ports connection diagram– Female Mini-Din 6 pin connector

The I/O ports can be configured for different uses (see 4.12 Digital input/output ports configuration for details). Factory settings for the I/O ports are:

Output **S1**	Flags when the upper limit is reached
Output **S2**	Flags when the lower limit is reached
Input **E1**	Trigger a data transmission
Input **E2**	Set preset or zero value (referencing)
Input **E3**	Auxiliary command

Both digital outputs have an internal structure shown in Figure 5. Each output is photo-coupled and capable of driving small loads (< 30mA). Special care must be taken when connecting loads to these ports to avoid short-cuts or excessive current drain.

When interfacing with programmable logic controllers (PLC) direct connection can be made (if the PLC input can detect 15VDC as high level), or a pull-up resistor (typically 1kohm) to the +24VDC PLC power supply can be added.

Interface with low level voltage circuits (+3,3V, +5V, +10V) can be accomplish with a simple resistive voltage divider, transistor interface or level converter IC.

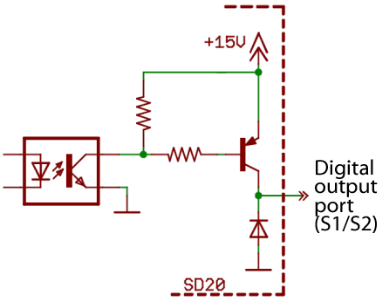

Figure 5 – Output ports internal structure

SD20's digital inputs have the internal structure shown in Figure 6. All digital inputs are photo-coupled, becoming active high when connect to ground (GND). Minimum current of 2mA must flow in order to active the photo-coupler interface.

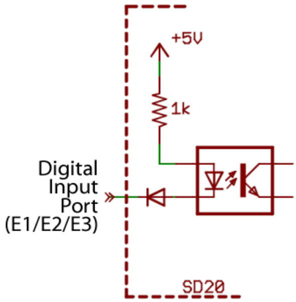

Figure 6 – Input ports internal structure

Interfacing the input ports to open-collector output ports of programmable logic controllers (PLC) can be done with direct connection (reverse voltage over 35V may damage the input port).

Interface with switches (dry contact) can be accomplished connecting the switch between the input pin and pin 8 (GND).

1.4 Status Led

Figure 1, highlight **L**, shows a communication and status led.

During boot-up (after USB cable connection and power supply) the status led will briefly become red and then turn green. This behavior indicates a successfully device initialization, with no hardware or software malfunctions detected.

During data transmission the status led will blink, indicating data flow to the computer.

During device programming the status led may briefly become red, returning to green. This behavior is normal and occurs during some of the internal software routines.

If status led stays red after boot-up, hardware malfunction or flash corruption was detected. Software updates and configuration software can diagnose several problems and apply fixes if required. Several auto diagnose routines are processed during boot-up and the device will not function if any anomaly is detected.

Please contact Metrolog technical assistance if needed.

2. Internal structure

2.1 Signal conditioner

The SD20 signal conditioner is available in two models, SD20-LVDT and SD20-Analog. The SD20-Analog model is a universal signal conditioner for DC type transducers and may be used with pressure sensors, load-cells, temperature sensors, inclinometers, among other transducers. The SD20-LVDT is a specialized version, with and additional hardware design to power and process signals from LVDT or RVDT sensors.

Its internal structure is composed of several analog and digital blocks. A structural design is shown in Figure 7.

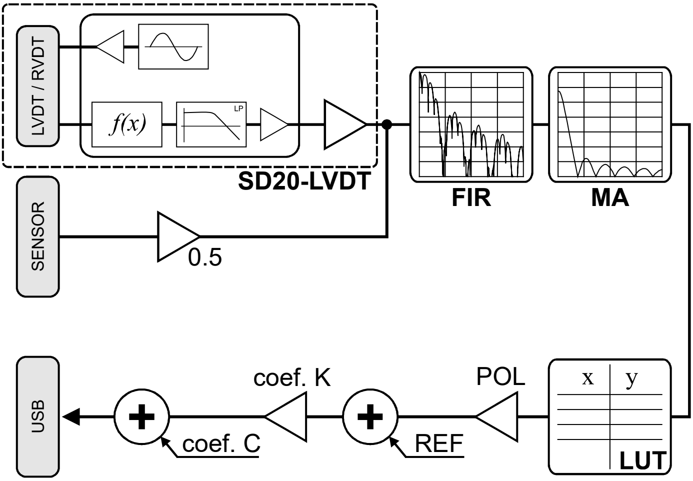

Figure 7 – SD20's internal structural design

Signal from the sensor is amplified and goes through a SINC^4 primary filter. This filter has high spectral selectivity and may be adjusted, for example, to attenuate (>80dB) frequencies from 50 to 60Hz. See details in section *2.4 Digital filters*.

After this primary filter the signal is processed by a secondary MA type filter (moving average) with high temporal selectivity. This filter may be adjusted with sample depth from 1 to 64, allowing aggressive signal aliasing. See details in section *2.4 Digital filters*.

Once filtered, the signal is forwarded to a Look-Up Table (LUT). This table has 524288 reference points stored in non volatile flash memory. The data set used for the LUT is created from the sensor's natural response curve, where data is acquired, processed and a mathematical model is used to interpolate all required data points.

This unique feature has the ability to compensate non-linearity errors, even if they are very small, increasing the final system accuracy.

The last conditioner block is responsible for data polarity inversion (POL), if required, digital gain (K coefficient, typically 1) and offset addition (C coefficient, typically 0).

The final value is then available for data transmission to the USB bus and for upper/lower limit threshold checks, if used.

2.2 Absolute and Relative measurements

The SD20 signal conditioner may be used to acquire absolute or relative measurements (with a nominal zero value defined by the user).

When using absolute measurements, all samples are processed by the LUT (see Figure 7), skip the sum block REF (the K and C coefficients still being applied, if used). This measuring mode can be used with absolute sensors, such as pressure sensors or load-cells, where the actual value is directly proportional transducer's magnitude.

When using relative measurements, all samples are processed by the LUT (see Figure 7) and then added to a reference value (offset) defined by the user. This measurement mode is applicable to incremental sensors or where variations from a set point must be observed.

Displacement sensors will usually be used with relative measurement mode. In these applications (usually with small displacement range available) the sensor only measures the displacement variations from a dimensional master.

Relative mode can be zeroed/set by the user using an input port (see *1.3 Digital Input/output Port Connector*) or by USB command requisition. When the system is zeroed/set, a new REF value is calculated and the final outputted value becomes the preset value defined by the user.

The transmission mode may be switched on the fly. It is important no note, however, that a referecing trigger (by input port signal or bus command) will automatically change the mode from absolute to relative.

2.3 Upper/Lower threshold limits

SD20 features two threshold limits, upper and lower control limits. These limits may be used to output signals when a measurement reaches maximum or minimum acceptable values.

In go/no-go systems it is possible to interface the digital outputs to activate, for example, luminous green/red indication, allowing the user to easily identify if the measurement is outside the acceptable range.

Additionally, if desired, the SD20 conditioner may be used without a computer. In this case the unit, previously programmed by user, only requires an external power supply to work (Metrolog FA20 or compatible).

2.4 Digital filters

SD20's digital conditioning circuits features two adjustable filters, allowing the user to set a specific sample rate and signal bandwidth.

The primary filter (FIR) has high spectral selectivity, with attenuation over 80dB for the selected cutoff frequencies and harmonics. The cutoff frequencies are directly correlated to the sample rate setting and may vary from 6.875 to 3520 samples per second (valid steps: 6.875; 13.75; 27.5; 55; 110; 220; 440 and 880 samples/s).

Figure 8 shows the primary cutoff frequency (fn) for each sample rate setting, and the typical filter response from 0 to 4fn.

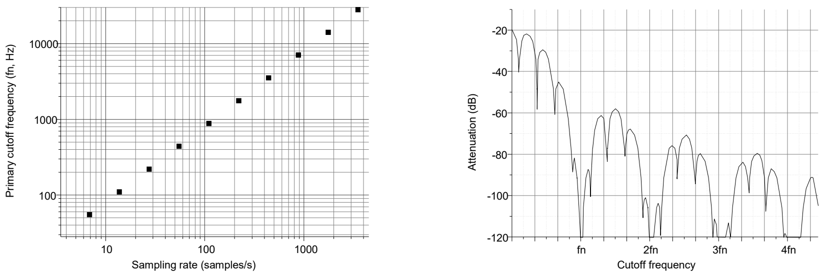

Figure 8 – Primary filter – Cutoff frequencies and signal attenuation versus sampling rate

Basically the primary filter with lower sample rates will offer better signal-to-noise ratio (SNR), but will limit the conditioner ability to observe transient signals. In opposite direction, higher sample rates will offer wider bandwidth, allowing the user to detect fast transients, however with increased noise floor.

The secondary filter (MA) has high temporal selectivity and can be used to suppress random noise generated by the sensor. It can be adjusted with values from 1 (disabled) to 64 (maximum depth), with cutoff frequency lower than 0.2Hz.

As a secondary filter, it receives the signal from the primary filter (FIR) and the final spectral response will be a combination of both filters. Figure 9 shows the primary cutoff frequency for various depth (1 to 64); each curve is represents the filter output from a specific sample rate set on the primary filter. The graph on the right shows the typical frequency filter response from 0 to 5fn.

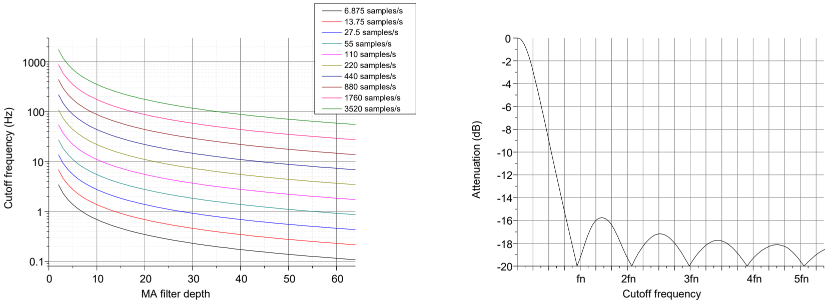

Figure 9 – Secondary filter MA
Cutoff frequency and attenuation versus filter depth
for various sample rates set in the primary filter

The secondary filter is especially useful to remove random noise or to smooth slow changing signals. If the primary filter is set to the lowest data rate (6.875 samples/s) and the secondary filter is set to its maximum depth (64 samples) the SD20 will provide maximum signal stability (at cost of response time – on these settings a step signal will reach 98% final value in about 10 seconds).

Reviewing all information, it is recommended:

- Manual measurements and data acquisition (typically limited to 5 samples/s): use primary filter at 27.5 samples/s (or lower) and secondary filter with 8 samples depth (or higher).
- Scanning application, such as run-out or conicity measurements: use primary filter with higher sample rate (110, 200 samples/s) and secondary filter disabled (1 sample depth).
- For transient analysis applications: signal shape must be analyzed and the primary filter data rate should be adjusted with the primary cutoff frequency 5 to 10 times higher than the highest frequency espected on the signal. The secondary filter may be applied to remove some signal noise, if present.

2.5 Data transmission and effective data rate

After completion of all calculations of the math block, a new value becomes available for transmission to the USB interface. Data transmission can occur in automatic mode (each new value is transmitted as soon it is available) or by request. Section *4.3 Data transmission* details the internal structure of data request and reply transmissions.

When continuous data transfer is used, a new value is transmitted just after data acquisition and mathematical processing. The effective data transfer is function of filter configurations and bus data rate.

Table 1 shows the effective data rate (reads per second) while using binary data transmission (IEE 754 float point coded added to 1 byte CRC-8). This table displays some combinations of primary filter setting (first column to the left) with secondary filter setting (table header line). The maximum data rate (2150 samples/s) is limited by the bus capacity (virtual UART port at 115,200bps). Lower data rates may occur at higher secondary filter depths due to the higher internal processing work load.

Table 1 – Effective USB data rate transmission, IEE754 coded, reads/s

Primary filter setting (samples/s) \ Secondary filter depth	**1**	**2**	**3**	**4**	**5**	**10**	**20**	**30**	**40**	**64**
3520	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)
1760	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)
880	847	847	847	847	847	847	847	847	847	847
440	435	435	435	435	435	435	435	435	435	435
220	220	220	220	220	220	220	220	220	220	220
110	111	110	110	110	110	110	110	110	110	110
55	55	55	55	55	55	55	55	55	55	55
27,5	27.5	27.5	27.5	27.5	27.5	27.5	27.5	27.5	27.5	27.5
13,75	13.75	13.75	13.75	13.75	13.75	13.75	13.75	13.75	13.75	13.75
6,875	6.875	6.875	6.875	6.875	6.875	6.875	6.875	6.875	6.875	6.875

(*) Not available in firmware v2.0+

3. Data acquisition – SD20 DataLogger

3.1 – Software Install – USB Driver

Before connecting SD20 to the computer for the first time it is necessary to install a USB driver. This driver will allow the operating system to identify new conditioners connected to the computer and to create new virtual communication port.

USB driver install is required just once. Administrator privileges are required for driver install.

The SD20 USB can be obtained in the device CD or directly on the web at:

http://www.metrolog.net/suporte/download.php?lang=en

There are driver for various operating systems. Please unpack and run the proper file as listed below:

<table>
<tr><td>Windows Server 2008 R2
Windows 7
Windows 7 x64
Windows Server 2008
Windows Server 2008 x64
Windows Vista
Windows Vista x64
Windows XP
Windows XP x64
Windows 2000
Windows Server 2003
Windows Server 2003 x64</td><td>\usb_driver\CDM20600.exe</td></tr>
<tr><td>Windows 98
Windows ME</td><td>\usb_driver\R10906.zip</td></tr>
<tr><td>Mac OS X</td><td>\usb_driver\FTDIUSBSerialDriver_v2_2_14.dmg</td></tr>
<tr><td>Linux
Linux x86_64</td><td>\usb_driver\ftdi_sio.tar.gz</td></tr>
</table>

After driver install connect the device to a USB port using the provided “AB” type USB cable. The operating system will identify the new equipment and create the proper communication port automatically.

3.2 – Install – SD20 DataLogger software

Inside the SD20 conditioner CD-ROM there is the SD20 DataLogger software. This software was developed to allow quick visualization and data acquisition. Also the software allows setup of all SD20 user parameters (filters, math coefficients, among others).

No install procedure is required. The software is portable and may be started from removable devices or from the user area. Only user privileges are required, so no special operating system permissions need to be granted (except permission to use the USB communication port).

It is recommended to copy the folder **\Metrolog_SD20_DataLogger** (within the product CD) to the user area.

After copying the folder run the file **SD20_DataLogger_v9_9.exe**. The software will scan the system for all SD20 available devices, and if required, request what device it should connect to.

Basic software RAM requirement is limited to 10Mb. Maximum data rate may be limited based on the computer hardware specifications.

3.3 – Data visualization and acquisition

3.3.1 – Main window

After startup the SD20 DataLogger software will show its main window, as shown in Figure 10.

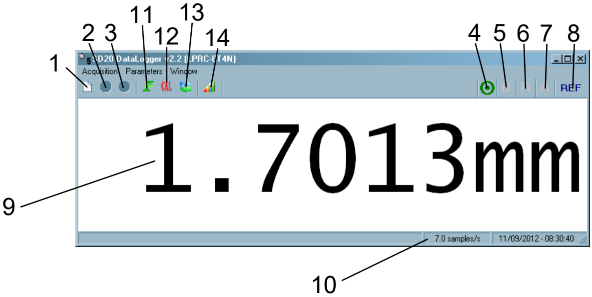

Figure 10 –SD20 DataLogger main window

Quick access buttons

(1) New dataset acquisition

(2) Start/Stop data acquisition

(3) Add new sample to the active data file

Indications

(4) Upper/lower limit indication

(5) Signal event in digit input port E3 (Data)

(6) Signal event in digit input port E1 (Referencing/Zero)

(7) Signal event in digit input port E2 (Auxiliary)

(8) Absolute/Relative measurement mode indication

Readout

(9) Instantaneous readout (or max/min/average/amplitude)

(10) Effective data transfer (sample per second)

Special functions

(11) Zero/Referencing button

(12) Calibration routine

(13) Clear of max/min/amplitude retentions

(14) Scope mode (graph visualization)

3.3.2 – Data acquisition

One of SD20 DataLogger features is data acquisition to a text file, allowing the user to export data to other software for further analysis.

To start a new data file click on the menu **Acquisition > New acquisition** (shortcut **Ctrl+N**) or click on the toolbar icon (indication "1" in Figure 10).

A setup window will be shown where the user can define the data file name, start and stop triggers and sampling method, as shown in Figure 11.

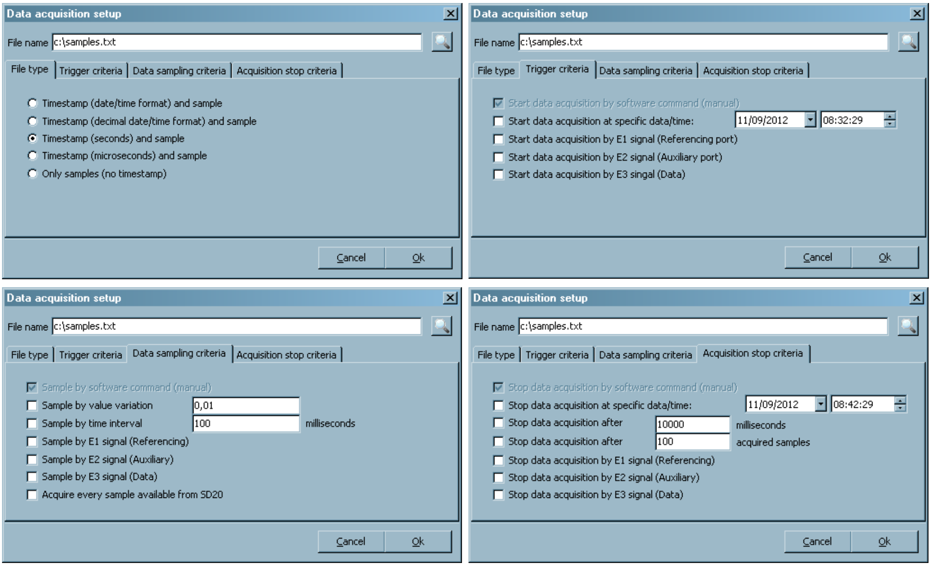

Figure 11 – Setup window for new data acquisition

Select a file name, sampling settings, trigger criteria, sampling criteria and stop criteria. Data sampling can be trigger by time intervals, value variation and/or by user request (direct on the software interface or by external signal applied to a digital input port).

After setup, the application will remain waiting for a trigger condition to start the acquisition process. Manual trigger can be set by pressing the start/stop icon on the software toolbar (indication "2" on Figure 10) or accessing menu **Acquisition > Start/Stop acquisition** (**F2** shortcut).

While data acquisitioning, it is possible to check running time and acquired sample count on the top status bar, as shown in Figure 12.

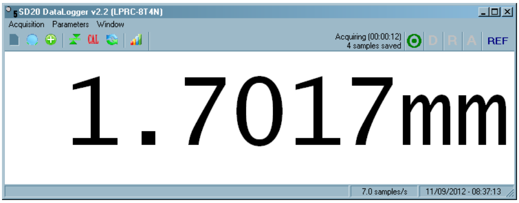

Figure 12 – SD20 DataLogger during data acquisition

New samples will be automatically added to the data file if any of the sampling criteria are meet. Manual sampling can be accomplished pressing the toolbar icon (indication "3" on Figure 10) or by menu **Acquisition > Capture Sample** (**F12** shortcut).

The acquisition process will end if any of the stop criteria are meet or by user intervention, by pressing the toolbar icon (indication "2" on Figure 10) or by menu **Acquisition > Start/Stop Acquisition** (**F2** shortcut).

To continue to acquire samples to an existing data file just create a new acquisition setup and set the filename to the existing file. Before the process starts the software will offer the option to overwrite or append to the existing data file.

All saved samples are ASCII coded, with the timestamp separated from the sample value by a tabulation character (**09H**). This data format can be directly imported by most commercial software, such as Matlab, Maple, Origin, Excel, among others.

3.4 – SD20 Internal parameters

All internal SD20 parameters, including digital filter setup, threshold limits and digital port configuration can be accessed by menu **Parameters > SD20 Internal parameters**.

On the first tab, as shown in Figure 13, it is possible to see all factory settings, including sensor information and serial numbers.

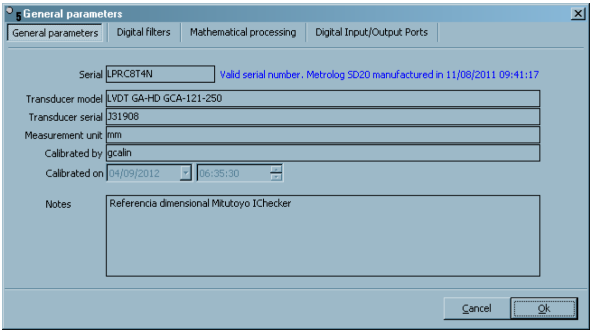

Figure 13 - SD20 internal parameters – Information tab

On the second tab, as shown in Figure 14, it is possible to change the primary and secondary digital filter settings. Factory settings are usually suitable for most application and are set based on the transducer characteristics. If required, change the settings based on the application requirements. Please revise guidelines on section *2.4 Digital filters* to configure the proper settings.

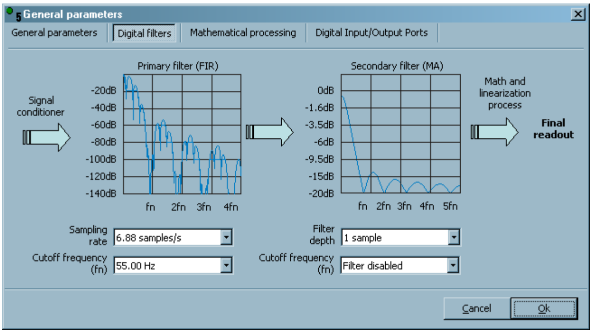

Figure 14 - SD20 Internal parameters – Digital filters settings

The thirds tab, as shown in Figure 15, gives access to all mathematical coefficients and upper/lower limit values.

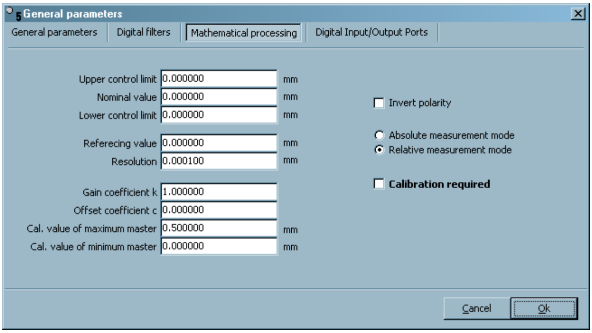

Figure 15 - SD20 Internal parameters – Mathematical processing

The forth and last tab, as shown in Figure 16, gives access to the input and output digital ports settings. To interface with external hardware please refer to section *1.3 Digital Input/output Port Connector*.

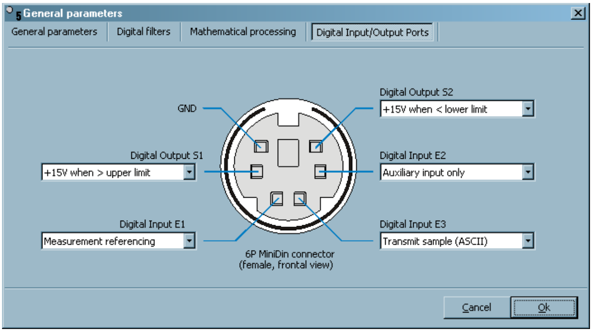

Figure 16 - SD20 Internal parameters – Input/output digital ports setup

4. Communication protocol

4.1 Notation and symbols

This section details the communication protocol used by the SD20 conditioner. The following notations and symbols are used to describe the transmission and reception data streams:

| 1F | = byte, hexadecimal notation

| 'X' | = byte, ASCII notation

| CR | = byte CR (*Carrier Return*), same as | 0D | in hexadecimal notation

| LF | = byte LF (*Line Feed*), same as | 0A | in hexadecimal notation

| LRC | = *Longitudinal Redundancy Check*

(please refer to section *4.21 LRC processing algorithm*)

| CRC | = *Cyclic Redundancy Check 8*, polynomial x^8+x^2+x+1

(please refer to section *4.19 Device serial,* factory information and parameters

From the firmware v2.0 it is possible to request all the information contained in the flash of the equipment, process equivalent to the use of the functions detailed in 4.17 Device serial number and factory information and 4.18 Functional parameters request:

Request: | 01 | A7 | 10 | 00 | 57 |

Answer: | A_0 | A_1 | A_2 | … | A_{1056} | LRC |

The contents of bytes A0 through A527 contain the factory and serial information of the equipment, as detailed in section 4.17 Device serial number and factory information.

The contents of bytes A528 to A1055 contain the functional parameters of the equipment, as detailed in section 4.18 Functional parameters request.

The last byte, A1056, contains the calculated LRC byte for the packet (veja *4.21 LRC*)

4.20 CRC-8 processing algorithm)

Notation of numbers within paragraphs may include the letter **H** for hexadecimal format (for example **10H**) or include the letter **d** to identify decimal format (for example **16d**). Numbers without any suffix can be either decimal integer or float type accordingly to the text scope.

4.2 Virtualized communication port

SD20's communication interface is mapped by the operating system (Windows, Linux, and Mac OS) as a standard serial port, allowing easy communication and compatibility with any software with RS232C interface capability.

From software stand point, the new mapped serial port will act as native RS232 port, where it should be opened, configured and transmission/reception processed in packets. There is no need to use control signals (DTR, RTS, DTR, CTS) or flow control.

The port should be initialized with 115200 bps baud rate, 8 data bit, no parity and 1 stop bit (**115200 8N1**). Other baud rates are not supported.

4.3 Data transmission

Data transmission from the SD20 may occur in ASCII format, binary or raw A/D reads. Single transmission can be performed by user request or continuous mode can be set.

4.3.1 Data transmission –ASCII format

Single value request:

Request: | ‘x’ | or | 78 |

Answer: | A_0 | A_1 | A_2 | … | A_{15} | CR | LF |

Request of multiple samples (continuous data transmission):

Request: | ‘X’ | or | 58 |

Answer: | A_0 | A_1 | A_2 | … | A_{15} | CR | LF | , | A_0 | A_1 | A_2 | … | A_{15} | CR | LF | , repeatedly

Data stream will always have 16 characters, followed by **CR** and **LF** terminators. The returned number is always right justified with empty characters filled with spaces.

For example, the value **16.3313827** will be returned as:

‘ ‘	‘ ‘	‘ ‘	‘ ‘	‘ ‘	‘ ‘	‘1‘	‘6‘	‘.‘	‘3‘	‘3‘	‘1‘	‘3‘	‘8‘	‘2‘	‘7‘	CR	LF

During continuous data transmission the device will transmit the 18-byte package whenever a new A/D sample is available and the final value has been processed (see section *2.4 Digital filters* for details about the sampling rate).

Continuous data transmission may be halted at any time by the command:

Transmission: | ‘0’ | or | 30 |

Transmission in ASCII format allows direct integration with commercial SPC software (VTB WinCep Online, Datamyte Applied Stats, among others), and direct data visualization with terminal software (Hyperterminal, Realterm, among others). If the user plans to develop new software it is recommended to use binary communication (see section *4.3.2 Data transmission – binary format*) due to the smaller data package and data integrity capability (CRC-8).

4.3.2 Data transmission – binary format

Single value request:

Request: | ‘f’ | or | 66 |

MSB			LSB	CRC

Answer: (above)

Request of multiple samples (continuous data transmission):

Request: | ‘F’ | or | 46 |

Answer:

MSB			LSB	CRC

,

MSB			LSB	CRC

, repeatedly

Binary data transmission follows the IEEE 754 standard for simple precision float type numbers (32-bits), where the transmission occurs from the most significant byte (MSB) to the least significant byte (LSB).

All packages have a fifth byte with a CRC-8 value calculated from the previous 4 bytes (see section *4.19 Device serial,* factory information and parameters

From the firmware v2.0 it is possible to request all the information contained in the flash of the equipment, process equivalent to the use of the functions detailed in 4.17 Device serial number and factory information and 4.18 Functional parameters request:

Request:

01	A7	10	00	57

Answer:

A_0	A_1	A_2	…	A_{1056}	LRC

The contents of bytes A0 through A527 contain the factory and serial information of the equipment, as detailed in section 4.17 Device serial number and factory information.

The contents of bytes A528 to A1055 contain the functional parameters of the equipment, as detailed in section 4.18 Functional parameters request.

The last byte, A1056, contains the calculated LRC byte for the packet (veja *4.21 LRC*)

4.20 CRC-8 processing algorithm for processing details).

The decimal number **16.336082458**, for example, will be transmitted as:

41	82	B0	4C	FC

During continuous data transmission in binary mode, special data packages may appear to signal events in the digital input ports. These special packages have the same 5-byte structure but the CRC-8 value, that is added by 1. Please refer to section *4.14 Signal events on input ports*, for details about this special package.

4.3.3 Data transmission – Raw A/D read

It is possible to request values directly from the 24-bit Analog to Digital converter. It is important to note, however, that all returned values are **not** processed by the linearization or math block and post processing may be required on the user software side:

Single value request:

Request: | ‘a’ | or | 61 |

Answer: | MSB | | | LSB | CRC |

Request of multiple samples (continuous data transmission):

Request: | ‘A’ | or | 41 |

Answer: | MSB | | | LSB | CRC | , | MSB | | | LSB | CRC | , repeatedly

Data transmission occurs in binary mode with coded unsigned 32-bit integer. Return will range from 0 to (2^{24}-1) or 0 to 16777215 counts.

All packages have a fifth byte with a CRC-8 value calculated from the previous 4 bytes (see section *4.19 Device serial,* factory information and parameters

From the firmware v2.0 it is possible to request all the information contained in the flash of the equipment, process equivalent to the use of the functions detailed in 4.17 Device serial number and factory information and 4.18 Functional parameters request:

Request: | 01 | A7 | 10 | 00 | 57 |

Answer: | A_0 | A_1 | A_2 | … | A_{1056} | LRC |

The contents of bytes A0 through A527 contain the factory and serial information of the equipment, as detailed in section 4.17 Device serial number and factory information.

The contents of bytes A528 to A1055 contain the functional parameters of the equipment, as detailed in section 4.18 Functional parameters request.

The last byte, A1056, contains the calculated LRC byte for the packet (veja *4.21 LRC*)

4.20 CRC-8 processing algorithm for processing details).

The A/D value **8409802**d, for example, will be transmitted as:

00	80	52	CA	55

During continuous data transmission in binary mode, special data packages may appear to signal events in the digital input ports. These special packages have the same 5-byte structure but the CRC-8 value, that is added by 1. Please refer to section *4.14 Signal events on input ports*, for details about this special package.

4.3.4 Data transmission – data packet (*firmware 2.0*+)

From firmware v2.0 it is possible to request the processed reading of the equipment (equivalent to the 'f' request), the raw ADC reading (equivalent to the 'a' request) and the status of the digital input and output (equivalent to the ‘d’ request) at once:

Single packet request:

Request: ‘p’ ou 70

Answer:

ADC MSB			ADC LSB	LUT MSB			LUT LSB	I/O STAT	CRC

Request of multiple packets (continuous data transmission):

Request: ‘P’ ou 50

Answer:

ADC MSB			ADC LSB	LUT MSB			LUT LSB	I/O STAT	CRC

, sucessivamente

Data transmission from 1st to the 4th bytes (ADC) occurs in binary mode with coded unsigned 32-bit integer. Return will range from 0 to (2^{24}-1) or 0 to 16777215 counts.

Data transmission from 5th to 8th bytes (LUT) follows the IEEE 754 standard for simple precision float type numbers (32-bits), where the transmission occurs from the most significant byte (MSB) to the least significant byte (LSB).

The 9th byte contains the status of the digital I/O ports as detailed in section 4.16 Input/output Status.

All packages have a tenth byte with a CRC-8 value calculated from the previous 9 bytes (see section *4.19 Device serial,* factory information and parameters

From the firmware v2.0 it is possible to request all the information contained in the flash of the equipment, process equivalent to the use of the functions detailed in 4.17 Device serial number and factory information and 4.18 Functional parameters request:

Request:

01	A7	10	00	57

Answer:

A_0	A_1	A_2	…	A_{1056}	LRC

The contents of bytes A0 through A527 contain the factory and serial information of the equipment, as detailed in section 4.17 Device serial number and factory information.
The contents of bytes A528 to A1055 contain the functional parameters of the equipment, as detailed in section 4.18 Functional parameters request.

The last byte, A1056, contains the calculated LRC byte for the packet (veja *4.21 LRC*)

4.20 CRC-8 processing algorithm for processing details).

For example, reception

00	24	EA	70		40	C3	4D	A0		80	12

Is decoded as:

00 24 EA 70 = 2419312 value (raw ADC reading, without processing)

40 C3 4D A0 = 6.1032257 value (equipment reading, floating point)

80 = Status of digital equipment input and output ports

12 = CRC-8 from the 9 bytes

4.4 Absolute and relative values

During data transmission it is possible to request absolute (based on factory calibration) or relative values (with an offset value).

To switch data transmission to absolute mode:

Request: ‘b’ or 62

Answer: *(no acknowledgment)*

To switch data transmission to relative mode:

Request: ‘r’ or 72

Answer: *(no acknowledgment)*

To request value zeroing/referencing (value set by the user – please refer to section *4.6 Referencing value*):

Request: ‘z’ or 7A

Answer: *(no acknowledgment)*

Obs.: value referencing command will automatically change data transmission to relative mode.

4.5 Nominal value

4.5.1 – Setting the parameter

A nominal value may be saved by the user on SD20's flash memory. This value is just a reference for use by the client software (PC). It does not have any internal function, but can be a valuable reference to the user as an informative process parameter.

To set a new nominal value:

Request:

01	A5	09	MSB			LSB	CRC*

Answer:

'O'	'K'

The nominal value, a single precision float number, must be coded accordingly to IEEE 754 standard, where the most significant byte (MSB) is first transmitted. The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used).

For example, to save the nominal value parameter as 3.185 (=**404BD70AH**):

Request:

01	A5	09	40	4B	D7	0A	6D

Answer:

'O'	'K'

4.5.2 – Requesting the parameter

The request of the nominal value parameter, stored in the device flash, can be performed through the command:

Request:

01	A6	09	3F

Answer:

LSB			MSB	LRC

The data transmission follows the IEEE 754 standard for simple precision float type numbers (32-bits), where the transmission occurs from the **less** significant byte (LSB) to the **most** significant byte (MSB).

All packages have a fifth byte with a LRC value calculated from the previous 4 bytes (see section 4.21 LRC processing algorithm for processing details).

For example, the received packet:

00	00	80	C1	41

is decoded as the floating-point number -16.0 (=**C1800000H**).

4.6 Referencing value

4.6.1 – Setting the parameter

A reference value (or zero) may be saved on SD20's flash memory by the user and it is used during value zeroing/referencing command (trigger by signal at an input port or by command sent on the USB interface). After the referencing command the device will offset the actual value and the new output readout will be the reference value set by the user. If zero, the output will become zero. After referencing the device will transmit data in relative mode (see section *2.2 Absolute and Relative measurements* for details).

To set the referencing value:

Request:

01	A5	0A	MSB			LSB	CRC*

Answer:

'O'	'K'

The referencing value, a single precision float number, must be coded accordingly to IEEE 754 standard, where the most significant byte (MSB) is first transmitted. The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used).

For example, to set the referencing value to -16 (=**C1800000H**):

Request:

01	A5	0A	C1	80	00	00	6A

Answer:

'O'	'K'

4.6.2 – Requesting the parameter

The request of the referencing value parameter, stored in the device flash, can be performed through the command:

Request:

01	A6	0A	36

Answer:

LSB			MSB	LRC

The data transmission follows the IEEE 754 standard for simple precision float type numbers (32-bits), where the transmission occurs from the **less** significant byte (LSB) to the **most** significant byte (MSB).

All packages have a fifth byte with a LRC value calculated from the previous 4 bytes (see section 4.21 LRC processing algorithm for processing details).

For example, the received packet:

00	00	80	C1	41

is decoded as the floating-point number -16.0 (=**C1800000H**).

4.7 Upper and lower limits

4.7.1 – Setting the parameters

Two threshold limits may be saved on SD20's flash memory. These limits, upper and lower limits, are used as thresholds to set the digital outputs S1 and S2 (the digital output may have other functions; see section *4.12 Digital input/output ports configuration* for details).

These two thresholds are continuously checked against each new value obtained from the sensor. When the measurement is higher than the upper limit or lower than the lower limit, digital outputs are set. It is important to note that both digital outputs do not offer hysteresis or delay, and the output signals are synchronized and updated on each new conversion. This may cause fast switching transients when crossing the thresholds and should be handled by the user external hardware.

To set the upper limit:

Request:

01	A5	07	MSB			LSB	CRC*

Answer:

'O'	'K'

To set the lower limit:

Request:

01	A5	08	MSB			LSB	CRC*

Answer:

'O'	'K'

The upper/lower limit, a single precision float number, must be coded accordingly to IEEE 754 standard, where the most significant byte (MSB) is first transmitted. The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used).

For example, to set the upper limit to 10.21 (=**41235C29H**):

Request:

01	A5	07	41	23	5C	29	75

Answer:

'O'	'K'

4.7.2 – Requesting the parameters

The request of the value of the upper or lower tolerance limits, stored in the equipment flash, can be performed through the commands:

To request the upper limit:

Request:

01	A6	07	15

Answer:

LSB			MSB	LRC

To request the lower limit:

Request:

01	A6	08	38

Answer:

LSB			MSB	LRC

The data transmission follows the IEEE 754 standard for simple precision float type numbers (32-bits), where the transmission occurs from the **less** significant byte (LSB) to the **most** significant byte (MSB).

All packages have a fifth byte with a LRC value calculated from the previous 4 bytes (see section 4.21 LRC processing algorithm for processing details).

For example, the received packet:

29	5C	23	41	17

is decoded as the floating-point number 10.21 (=**41235C29H**).

4.8 Native resolution

4.8.1 – Setting the parameter

It is possible set a native resolution on the SD20 conditioner. This value does not have any internal function, but may serve on the client software (PC) to proper show the readout with adequate resolution and decimal places.

To set the native resolution:

Request:

01	A5	0B	MSB			LSB	CRC*

Answer:

‘O’	‘K’

The native resolution is coded as a fixed point value with 6 decimal points. The finest resolution possible is 0.000001 (coded as **00000001H)**. The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used).

For example, to set the native resolution to 0.05 (= **50000d** = **0000C350H**):

Request:

01	A5	0B	00	00	C3	50	DA

Answer:

‘O’	‘K’

4.8.2 – Requesting the parameter

The request of the native resolution parameter, stored in the device flash, can be performed through the command:

Request:

01	A6	0B	31

Answer:

LSB			MSB	LRC

The native resolution is coded as a fixed point value with 6 decimal points.

All packages have a fifth byte with a LRC value calculated from the previous 4 bytes (see section 4.21 LRC processing algorithm for processing details).

For example, the received packet:

50	C3	00	00	93

is decoded as the integer number 50000d (=**0000C350H**), which corresponds to the native resolution of 0.05.

4.9 Gain (K) and offset (C) coefficients

4.9.1 – Setting the parameters

There are two coefficients used in the math block: the gain coefficient (K) and the offset coefficient (C). These coefficients are applied in the final math processing, multiplying the value by K and adding an offset C, prior to transmission (see section *2.1 Signal conditioner* for details)

Typically the K coefficient is 1 (unitary) and the C coefficient 0 (no offset). When these two values are used, the final value after the LUT is then transmitted without any change.

In some cases the user may need to apply special coefficients to change gain or to offset the measurement by a constant value.

To set the K coefficient (gain):

Request:

01	A5	05	MSB			LSB	CRC*

Answer:

'O'	'K'

To set the C coefficient (*offset*):

Request:

01	A5	06	MSB			LSB	CRC*

Answer:

'O'	'K'

The coefficient value, a single precision float number, must be coded accordingly to IEEE 754 standard, where the most significant byte (MSB) is first transmitted. The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used).

For example, to set the K coefficient to 1.5 (=**3FC00000H**):

Request:

01	A5	05	3F	C0	00	00	1B

Answer:

'O'	'K'

4.9.2 – Requesting the parameters

The requisition of the gain and offset coefficients, stored in the equipment flash, can be performed through the commands:

To request the K coefficient (gain):

Request:

01	A6	05	1B

Answer:

LSB			MSB	LRC

To request the C coefficient (*offset*):

Request:

01	A6	06	12

Answer:

LSB			MSB	LRC

The data transmission follows the IEEE 754 standard for simple precision float type numbers (32-bits), where the transmission occurs from the **less** significant byte (LSB) to the **most** significant byte (MSB).

All packages have a fifth byte with a LRC value calculated from the previous 4 bytes (see section 4.21 LRC processing algorithm for processing details).

For example, the received packet:

00	00	C0	3F	FF

is decoded as the floating point number 1.5 (=**3FC00000H**).

4.10 Primary digital filter (FIR)

4.10.1 – Setting the parameter

The SD20 conditioner features two programmable digital filters, responsible to adjust input signal bandwidth and effective data sampling rate.

The primary digital filter, a FIR filter, may be adjusted with 8 different cutoff frequencies and sampling data rates. High sampling rate will allow acquisition of transient signals with higher frequencies, however with lower signal-to-noise ratio (SNR). Lower sampling rates are suitable for stable signals and will offer higher SNR.

Commands to change the primary filter setting:

FIR 880 samples/s (f_c=7040Hz):	01	A5	01	00	00	00	18	2A
FIR 440 samples/s (f_c=3520Hz):	01	A5	01	00	00	00	20	82
FIR 220 samples/s (f_c=1760Hz):	01	A5	01	00	00	00	28	BA
FIR 110 samples/s (f_c=880Hz):	01	A5	01	00	00	00	30	F2
FIR 55 samples/s (f_c=440Hz):	01	A5	01	00	00	00	38	CA
FIR 27.5 samples/s (f_c=220Hz):	01	A5	01	00	00	00	40	A5
FIR 13.75 samples/s (f_c=110Hz):	01	A5	01	00	00	00	48	9D
FIR 6.875 samples/s (f_c=55Hz):	01	A5	01	00	00	00	78	0D

Answer: | 'O' | 'K' |

4.10.2 – Requesting the parameter

The request of the configuration of the primary filter (FIR), stored in the flash of the equipment, can be performed through the command:

Request:

01	A6	01	07

Answer:

FIR	00	00	00	LRC

The FIR byte indicates the active configuration of the filter:

- `18` = FIR 880 samples/s (f_c=7040Hz)
- `20` = FIR 440 samples/s (f_c=3520Hz)
- `28` = FIR 220 samples/s (f_c=1760Hz)
- `30` = FIR 110 samples/s (f_c=880Hz)
- `38` = FIR 55 samples/s (f_c=440Hz)
- `40` = FIR 27,5 samples/s (f_c=220Hz)
- `48` = FIR 13.75 samples/s (f_c=110Hz)
- `78` = FIR 6.875 samples/s (f_c=55Hz)

4.11 Secondary digital filter (MA)

4.11.1 – Setting the parameter

The SD20 conditioner features two programmable digital filters, responsible to adjust input signal bandwidth and effective sampling data rate.

The secondary digital filter, a moving average (MA) filter, have a sampling depth up to 64 samples and do not interfere with the primary filter data sampling. It computes the average value of the last n samples, providing high temporal selectivity and low spectral selectivity.

To change the secondary filter depth:

Request:

01	A5	02	00	00	00	MA	CRC*

Answer:

'O'	'K'

OBS*: The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used). MA value can range from **1**d to **64**d.

Command examples:

Depth MA 1 (disabled):	01	A5	02	00	00	00	01	C3
Depth MA 2 (minimum depth):	01	A5	02	00	00	00	02	CA
Depth MA 3:	01	A5	02	00	00	00	03	CD
Depth MA 4:	01	A5	02	00	00	00	04	D8
Depth MA 5:	01	A5	02	00	00	00	05	DF
Depth MA 8:	01	A5	02	00	00	00	08	FC
Depth MA 16:	01	A5	02	00	00	00	10	B4
Depth MA 32:	01	A5	02	00	00	00	20	24
Depth MA 48:	01	A5	02	00	00	00	30	54
Depth MA 64 (maximum depth):	01	A5	02	00	00	00	40	03

4.11.2 – Requesting the parameter

The request of the configuration of the secondary filter (MA), stored in the flash of the equipment, can be performed through the command:

Request:

01	A6	02	0E

Answer:

MA	00	00	00	LRC

The MA byte indicates the active configuration of the filter, for example:

03	00	00	00	03

Indicates that the filter is set to depth MA 3.

4.12 Digital input/output ports configuration

4.12.1 – Setting the parameters

The SD20 conditioner features 2 digital output ports and 3 digital input ports. Each port can be programmed with specific functions (for example for data request, value referencing, among others) or may be use as auxiliary ports.

To change the port function:

Request:

01	A5	03	00	00	IO1	IO0	CRC*

Answer:

'O'	'K'

OBS*: The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used).

IO1 byte (MSB) and IO0 byte (LSB) have specific flags detailed below:

Digital Input Port E1 (Data)

00	00

Sets port E1 to trigger a transmission in ASCII format (default)
(see *4.3.1 Data transmission –ASCII format* for details)

00	01

Sets port E1 to trigger a transmission in binary format
(see *4.3.2 Data transmission – binary format*4.3.2 Data transmission – for details)

00	02

Sets port E1 to trigger a transmission of raw A/D value
(see *4.3.3 Data transmission – Raw A/D read* for details)

00	04

No specific function (may be used by client software as an auxiliary input)
(see *4.14 Signal events on input ports* and
4.16 Input/output Status for application details)

Digital Input Port E2 (Referencing)

00	00

Sets port E2 to trigger readout zeroing/referencing (default)
(see *4.4 Absolute and relative values* for details)

00	08

No specific function (may be used by client software as an auxiliary input)
(see *4.14 Signal events on input ports* and
4.16 Input/output Status for application details)

Digital Input Port E3 (Auxiliary)

00	08

No specific function (may be used by client software as an auxiliary input)
(see *4.14 Signal events on input ports* and

4.16 Input/output Status for application details)

Digital Output Port S1 (Upper limit)

00	00

Sets port S1 to signal when upper limit is reached
(see *4.7 Upper and lower limits* for details)

02	00

Sets port S1 to signal when value is approved (within upper/lower limits)
(see *4.7 Upper and lower limits* for details)

04	00

No specific function (may be used by client software as auxiliary signal)
(see *4.15 Output port signaling* for details)

Digital Output Port S2 (Lower limit)

00	00

Sets port S2 to signal when lower limits is reached
(see *4.7 Upper and lower limits* for details)

10	00

Sets port S2 to signal when value is reproved (outside upper/lower limits)
(see *4.7 Upper and lower limits* for details)

20	00

No specific function (may be used by client software as auxiliary signal)
(see *4.15 Output port signaling* for details)

Multiple *flags* can be set within each of **IO0** and **IO1** bytes (byte overlaps with OR function).

4.12.2 – Requesting the parameters

The request of the I/O ports configuration, stored in the flash of the equipment, can be performed through the command:

Request:

01	A6	03	09

Answer:

IO 0	IO 1	00	00	LRC

Refer to section 4.12 Digital input/output ports configuration for decoding bytes IO0 and IO1.

4.13 System flags

4.13.1 – Setting the parameters

SD20 system flags allow readout polarity switch and transmission mode setup (absolute/relative modes).

Request:

01	A5	04	00	00	SF1	SF0	CRC*

Answer:

'O'	'K'

OBS*: The CRC-8 byte should compute from the third to the seventh byte (the first two bytes must not be used).

Bytes **SF1** (MSB) and **SF0** (LSB) have specific *flags*:

Polarity

00	00

Normal polarity

20	00

Inverse polarity (value output by the LUT block suffers signal inversion)
(see *2.1 Signal conditioner*2.1 for internal block details and processing)

Transmission mode

00	00

Absolute transmission mode
(see *2.2 Absolute and Relative measurements*2.2 Absolute and for details)

40	00

Relative transmission mode
(see *2.2 Absolute and Relative measurements*2.2 Absolute and for details)

Multiple *flags* can be set within each of **SF0** and **SF1** bytes (byte overlaps with OR function).

4.12.2 – Requesting the parameters

The request of the system flags configuration, stored in the flash of the equipment, can be performed through the command:

Request:

01	A6	04	1C

Answer:

SF0	SF1	00	00	LRC

Refer to section 4.13 System flags for decoding bytes SF0 and SF1.

4.14 Signal events on input ports

During continuous data transmission (binary mode or raw A/D mode only), special packages are created when signals are detected on input port E1, E2 or E3. Each time a high to low transition is detected a new event package it transmitted:

FF	FF	FF	STAT	CRC +1

where the STAT flags:

bit 0 (LSBit) = signal detected on input port E2 (Zero/Referencing)

bit 1 = signal detected on input port E1 (Data)

bit 2 = signal detected on input port E3 (auxiliary port)

bit 3..7 = reserved

Signal detection occurs when a signal transition from high (5 to 15VDC) to low (<1VDC) is applied to the input port. If the input port is being activated by a simple dry contact connected to GND (pin 8 or mini-DIN connector body), the signal detection will occurs when the contact is closed.

It is important to note the modified CRC-8 byte for this particular transmission package. The CRC-8 byte is calculated using the first 4 bytes, and then added by 1. This modified verification byte allows easy separation of regular value packages from the status package, without data transmission interruptions.

4.15 Output port signaling

The SD20 conditioner features two digital output ports that can be used to signal when upper/lower limits are reached, to signal approved/reproved status (when inside/outside the limits range) or even as auxiliary outputs set by the user.

When configured as auxiliary outputs (see section *4.12 Digital input/output ports configuration* for details how to change the port function), the S1 or S2 ports can be set or cleared using the commands:

To set S1 output:	'S'	or	53
To clear S1 output:	's'	or	73
To set S2 output:	'I'	or	49
To clear S2 output:	'i'	or	69

Answer: *(no acknowledgment)*

4.16 Input/output Status

The SD20 conditioner features 2 digital output ports and 3 digital input ports. Each port can be programmed with specific functions (for example for data request, value referencing, among others) or may be use as auxiliary ports.

During data communication it is possible to detect events in the input ports by a special data package, as detailed in section *4.14 Signal events on input ports*.

It is also possible to request port status asynchronously, by user request:

Request: | 'd' | or | 64 |

FF	FF	FF	STAT	CRC +1

Answer:

where the STAT flags:

bit 0 (LSBit) = **H** when E2 input port (Referencing) is in low logic level

bit 1 = **H** when E1 input port (Data) is in low logic level

bit 2 = **H** when input port E3 (auxiliary) is in low logic level

bit 3..5 = reserved

bit 6 = **H** when S2 output port (Lower limit) is set (high logic level)

bit 7 = **H** when S1 output port (Upper limit) is set (high logic level)

It is important to note the modified CRC-8 byte for this particular transmission package. The CRC-8 byte is calculated using the first 4 bytes, and then added by 1. This modified verification byte allows easy separation of regular value packages from the status package, without data transmission interruptions.

4.17 Device serial number and factory information

Detailed information about sensor model, serial numbers and device factory settings are saved on SD20’s flash memory and may be read by the user. The serial number, in special, is important in multi point measurements systems to enable identification of a specific SD20 unit without the need to rely on COM port number (that may change if USB cable is connected to another port).

To request factory information:

Request:	01	A7	0F	8C	6E	
Answer:	A_0	A_1	A_2	…	A_{527}	LRC

The return package has 528 bytes length with the following data structure:

Byte 0 to 13	Fixed text "METROLOG SD20 "
Byte 14 to 22	Serial number (8 ASCII characters) + LRC
Byte 23 to 63	Paired sensor model (up to 40 characters) + LRC
Byte 64 to 104	Paired sensor serial number (up to 40 characters) + LRC
Byte 105 to 125	Measurement unit (up to 20 characters) + LRC
Byte 126 to 166	Calibrated by (name) (up to 40 characters) + LRC
Byte 167 to 186	Calibration date/time (dd/mm/yyyy hh:mm:ss format) (19 ASCII characters, 24H time format) + LRC
Byte 187 to 441	Additional observations (up to 254 characters) + LRC
Byte 442 to 527	Reserved
Byte 528	Verification byte LRC (see *4.21 LRC processing algorithm*)

For example, the following factory data would return from a SD20 device (in the diagram each fragment’s initial byte were marked with a thicker box and the check bytes LCR were highlighted in green):

4D	45	54	52	4F	4C	4F	47	20	53	44	32	30	20	4B	58	4B	59	54	48	34	4C	65	4C
42	42	33	31	35	50	41	2D	30	34	30	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	73	4A	31	34	35	35	33	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	7C	6D	6D	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	67	63	61	6C	69	6E	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0E	30
39	2F	30	33	2F	32	30	31	30	20	31	31	3A	31	30	3A	35	38	25	52	65	66	2E	20
64	65	20	63	61	6C	69	62	72	61	63	61	6F	20	6D	69	63	72	6F	6D	65	74	72	6F
20	6C	61	73	65	72	20	58	4C	53	34	30	2C	20	73	65	72	69	61	6C	20	41	58	38
33	35	32	34	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	79	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
FF																							

where,

4D	45	54	52	4F	4C	4F	47	20	53	44	32	30	20

= “**METROLOG SD20** ”

4B	58	4B	59	54	48	34	4C	65

= Serial “**KXKYTH4L**”

																							4C
42	42	33	31	35	50	41	2D	30	34	30	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	73	4A							

= Paired sensor model “**LBB315PA-040**”

																4A	31	34	35	35	33	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	7C															

= Paired sensor serial number “**J14553**”

									6D	6D	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00																		

= Measurement unit “**mm**”

						67	63	61	6C	69	6E	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0E	

= Calibrated by “**gcalin**”

																							30
39	2F	30	33	2F	32	30	31	30	20	31	31	3A	31	30	3A	35	38	25					

= Calibration date/time = “**09/03/2010 11:10:58**”

																			52	65	66	2E	20
64	65	20	63	61	6C	69	62	72	61	63	61	6F	20	6D	69	63	72	6F	6D	65	74	72	6F
20	6C	61	73	65	72	20	58	4C	53	34	30	2C	20	73	65	72	69	61	6C	20	41	58	38
33	35	32	34	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	79														

= Additional observations = “**Ref. de calibracao micrometro laser XLS40, serial AX83524**”

4.18 Functional parameters request

All functional parameters, such as limit thresholds, referencing value, among other, can be requested to the SD20 device as a single package. The request command is:

Request:

01	A7	0F	8D	69

Answer:

A_0	A_1	A_2	…	A_{527}	LRC

The returned package, with 528 bytes, has segments of 5 bytes, each one coding one parameter (4 bytes) and a check byte LRC.

Each 4 bytes parameter can codify a single precision float type, a integer value or a text segment (coded LSB to MSB):

Byte 0	Fixed value **53443230H** (watermark)
Byte 5	Primary filter configuration (FIR) (see *4.10 Primary digital filter (FIR)* for details)
Byte 10	Secondary filter configuration (MA) (see *4.11 Secondary digital filter (MA)* for details)
Byte 15	Digital input/output port configuration (see *4.12 Digital input/output ports configuration* for details)
Byte 20	System flags configuration (see *4.13 System flags* for details)
Byte 25	Coefficient K value (gain) (see *4.9 Gain (K) and* offset (C) coefficients for details)
Byte 30	Coefficient C value (*offset*) (see *4.9 Gain (K) and* offset (C) coefficients for details)
Byte 35	Upper limit value (see *4.7 Upper and lower limits* for details)
Byte 40	Lower limit value (see *4.7 Upper and lower limits* for details)
Byte 45	Nominal value (for user reference only) (see *4.5 Nominal value* for details)
Byte 50	Referencing value (see *4.6 Referencing value* for details)

Byte 55 Native resolution (user reference only)
(see *4.8 Native resolution* for details)

For example, for a SD20 device the following parameter data would return from the device (in the diagram each fragment LCR byte was highlighted in green):

53	44	32	30	15	40	00	00	00	40	03	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	80	3F	BF	00	00	00	00	00	29	5C	23	41	17	3D	0A	23	41	55	33	33	23
41	62	96	43	23	41	B7	64	00	00	00	64	6F	6C	6F	4D	65	74	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67	4D	65	74	72	6F	6C	6F	67
FF																							

where,

53	44	32	30	15	**53443230H** (watermark)
40	00	00	00	40	primary filter at 27.5 samples/s (f_c=220Hz)
03	00	00	00	00	secondary filter with depth set to 3 samples
00	00	00	00	00	input/output port configuration = **00000000H**
00	00	00	00	00	system flags = **00000000H**
00	00	80	3F	BF	coefficient K (gain) = **3F800000H** = 1.0 (single precision coded, accordingly to IEEE 754 standard)
00	00	00	00	00	coefficient C (offset) = **00000000H** = 0.0 (single precision coded, accordingly to IEEE 754 standard)
29	5C	23	41	17	upper limit value = **41235C29H** = 10.21 (single precision coded, accordingly to IEEE 754 standard)

3D	0A	23	41	55	lower limit = **41230A3DH** = 10.19 (single precision coded, accordingly to IEEE 754 standard)
33	33	23	41	62	nominal value = **41233333H** = 10.2 (single precision coded, accordingly to IEEE 754 standard)
96	43	23	41	B7	referencing value = **41234396H** = 10.204 (single precision coded, accordingly to IEEE 754 standard)
64	00	00	00	64	native resolution = **00000064H** = **100d** = 0.000100 (fixed point notation, coded as 6 decimal places)

4.19 Device serial, factory information and parameters

From the firmware v2.0 it is possible to request all the information contained in the flash of the equipment, process equivalent to the use of the functions detailed in 4.17 Device serial number and factory information and 4.18 Functional parameters request:

Request:

01	A7	10	00	57

Answer:

A_0	A_1	A_2	…	A_{1056}	LRC

The contents of bytes A0 through A527 contain the factory and serial information of the equipment, as detailed in section 4.17 Device serial number and factory information.

The contents of bytes A528 to A1055 contain the functional parameters of the equipment, as detailed in section 4.18 Functional parameters request.

The last byte, A1056, contains the calculated LRC byte for the packet (veja *4.21 LRC*)

4.20 CRC-8 processing algorithm

Many transmission packages described on this manual uses a CRC check byte (*Cyclic Redundancy Check*). This verification number is a ***hash*** function based on the polynomial function x^8+x^2+x+1 and is used as an integrity verification method to assure correct data transmission or reception.

The CRC-8 algorithm is not complex and may be implemented using various methods. Its computational method usually relays on a recursive CRC function call, processed byte by byte, where the previous result is used to process the next byte.

As a quick reference for developers, two CRC-8 implementations are listed, in C/C++ and Delphi/Pascal languages. Both routines relay on look-up table for maximum processing speed and lowest computational complexity.

4.20.1 CRC-8 implementation example – C/C++

```
/* (!) code fragment */
/* ---------------------------------------------------------------------------------------- */

    const unsigned char CRC8_TABLE[256] = {
              0x00,0x07,0x0E,0x09,0x1C,0x1B,0x12,0x15,0x38,0x3F,0x36,0x31,0x24,0x23,0x2A,0x2D,
              0x70,0x77,0x7E,0x79,0x6C,0x6B,0x62,0x65,0x48,0x4F,0x46,0x41,0x54,0x53,0x5A,0x5D,
              0xE0,0xE7,0xEE,0xE9,0xFC,0xFB,0xF2,0xF5,0xD8,0xDF,0xD6,0xD1,0xC4,0xC3,0xCA,0xCD,
              0x90,0x97,0x9E,0x99,0x8C,0x8B,0x82,0x85,0xA8,0xAF,0xA6,0xA1,0xB4,0xB3,0xBA,0xBD,
              0xC7,0xC0,0xC9,0xCE,0xDB,0xDC,0xD5,0xD2,0xFF,0xF8,0xF1,0xF6,0xE3,0xE4,0xED,0xEA,
              0xB7,0xB0,0xB9,0xBE,0xAB,0xAC,0xA5,0xA2,0x8F,0x88,0x81,0x86,0x93,0x94,0x9D,0x9A,
              0x27,0x20,0x29,0x2E,0x3B,0x3C,0x35,0x32,0x1F,0x18,0x11,0x16,0x03,0x04,0x0D,0x0A,
              0x57,0x50,0x59,0x5E,0x4B,0x4C,0x45,0x42,0x6F,0x68,0x61,0x66,0x73,0x74,0x7D,0x7A,
              0x89,0x8E,0x87,0x80,0x95,0x92,0x9B,0x9C,0xB1,0xB6,0xBF,0xB8,0xAD,0xAA,0xA3,0xA4,
              0xF9,0xFE,0xF7,0xF0,0xE5,0xE2,0xEB,0xEC,0xC1,0xC6,0xCF,0xC8,0xDD,0xDA,0xD3,0xD4,
              0x69,0x6E,0x67,0x60,0x75,0x72,0x7B,0x7C,0x51,0x56,0x5F,0x58,0x4D,0x4A,0x43,0x44,
              0x19,0x1E,0x17,0x10,0x05,0x02,0x0B,0x0C,0x21,0x26,0x2F,0x28,0x3D,0x3A,0x33,0x34,
              0x4E,0x49,0x40,0x47,0x52,0x55,0x5C,0x5B,0x76,0x71,0x78,0x7F,0x6A,0x6D,0x64,0x63,
              0x3E,0x39,0x30,0x37,0x22,0x25,0x2C,0x2B,0x06,0x01,0x08,0x0F,0x1A,0x1D,0x14,0x13,
              0xAE,0xA9,0xA0,0xA7,0xB2,0xB5,0xBC,0xBB,0x96,0x91,0x98,0x9F,0x8A,0x8D,0x84,0x83,
              0xDE,0xD9,0xD0,0xD7,0xC2,0xC5,0xCC,0xCB,0xE6,0xE1,0xE8,0xEF,0xFA,0xFD,0xF4,0xF3};

    unsigned char data[10] = {0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09};

    unsigned char i;
    unsigned char crc;

    crc = 0x00;
    for(i=0;i<10;i++)
        crc = CRC8_TABLE[(crc ^ data[i]) & 0xff];

/* ---------------------------------------------------------------------------------------- */
```

On this example, after processing all 10 bytes from **data[]** vector the CRC-8 value will be **39H**.

4.20.2 CRC-8 implementation example – Delphi/Pascal

```
{ (!) code fragment }
{ ------------------------------------------------------------------------------------------ }

    CRC8_TABLE: array [0..255] of byte = (
             $00,$07,$0E,$09,$1C,$1B,$12,$15,$38,$3F,$36,$31,$24,$23,$2A,$2D,
             $70,$77,$7E,$79,$6C,$6B,$62,$65,$48,$4F,$46,$41,$54,$53,$5A,$5D,
             $E0,$E7,$EE,$E9,$FC,$FB,$F2,$F5,$D8,$DF,$D6,$D1,$C4,$C3,$CA,$CD,
             $90,$97,$9E,$99,$8C,$8B,$82,$85,$A8,$AF,$A6,$A1,$B4,$B3,$BA,$BD,
             $C7,$C0,$C9,$CE,$DB,$DC,$D5,$D2,$FF,$F8,$F1,$F6,$E3,$E4,$ED,$EA,
             $B7,$B0,$B9,$BE,$AB,$AC,$A5,$A2,$8F,$88,$81,$86,$93,$94,$9D,$9A,
             $27,$20,$29,$2E,$3B,$3C,$35,$32,$1F,$18,$11,$16,$03,$04,$0D,$0A,
             $57,$50,$59,$5E,$4B,$4C,$45,$42,$6F,$68,$61,$66,$73,$74,$7D,$7A,
             $89,$8E,$87,$80,$95,$92,$9B,$9C,$B1,$B6,$BF,$B8,$AD,$AA,$A3,$A4,
             $F9,$FE,$F7,$F0,$E5,$E2,$EB,$EC,$C1,$C6,$CF,$C8,$DD,$DA,$D3,$D4,
             $69,$6E,$67,$60,$75,$72,$7B,$7C,$51,$56,$5F,$58,$4D,$4A,$43,$44,
             $19,$1E,$17,$10,$05,$02,$0B,$0C,$21,$26,$2F,$28,$3D,$3A,$33,$34,
             $4E,$49,$40,$47,$52,$55,$5C,$5B,$76,$71,$78,$7F,$6A,$6D,$64,$63,
             $3E,$39,$30,$37,$22,$25,$2C,$2B,$06,$01,$08,$0F,$1A,$1D,$14,$13,
             $AE,$A9,$A0,$A7,$B2,$B5,$BC,$BB,$96,$91,$98,$9F,$8A,$8D,$84,$83,
             $DE,$D9,$D0,$D7,$C2,$C5,$CC,$CB,$E6,$E1,$E8,$EF,$FA,$FD,$F4,$F3);

    data: array [0..9] of byte = ($00,$01,$02,$03,$04,$05,$06,$07,$08,$09);
    i: integer;
    crc: byte;

    crc := $00;
    for i:=0 to 9 do
        crc := CRC8_TABLE[(crc XOR data[i]) AND $ff];

{ ------------------------------------------------------------------------------------------ }
```

On this example, after processing all 10 bytes from **data[]** vector the CRC-8 value will be **39H**.

4.21 LRC processing algorithm

Many transmission packages described on this manual uses a LRC check byte (*Longitudinal Redundancy Check*). This simple verification code is used to check transmission/reception data integrity and has very low computational complexity.

The LRC check byte is simple and only requires the use of XOR (exclusive OR) binary operator byte by byte.

As a quick reference for developers, two LRC implementations are listed, in C/C++ and Delphi/Pascal languages.

4.21.1 LRC implementation example – C/C++

```
/* (!) code fragment */
/* ------------------------------------------------------------------------------- */

    unsigned char data[10] = {0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09};

    unsigned char i;
    unsigned char lrc;

    lrc = 0x00;
    for(i=0;i<10;i++)
        lrc ^= data[i];

/* ------------------------------------------------------------------------------- */
```

After processing the **lrc** variable will have the value **01H**, the LCR coded value for all 10 bytes of **data[ ]** vector.

4.21.2 LRC implementation example – Delphi/Pascal

```
{ (!) code fragment }
{ -------------------------------------------------------------------------------- }

    data: array [0..9] of byte = ($00,$01,$02,$03,$04,$05,$06,$07,$08,$09);
    i: integer;
    lrc: byte;

    lrc := $00;
    for i:=0 to 9 do
        lrc := lrc XOR data[i];

{ -------------------------------------------------------------------------------- }
```

After processing the **lrc** variable will have the value **01H**, the LCR coded value for all 10 bytes of **data[ ]** vector.

Appendix A – ASCII Table

Dec	Oct	Hex	Binary	Value
000	000	000	00000000	NUL
001	001	001	00000001	SOH
002	002	002	00000010	STX
003	003	003	00000011	ETX
004	004	004	00000100	EOT
005	005	005	00000101	ENQ
006	006	006	00000110	ACK
007	007	007	00000111	BEL
008	010	008	00001000	BS
009	011	009	00001001	HT
010	012	00A	00001010	LF
011	013	00B	00001011	VT
012	014	00C	00001100	FF
013	015	00D	00001101	CR
014	016	00E	00001110	SO
015	017	00F	00001111	SI
016	020	010	00010000	DLE
017	021	011	00010001	XON
018	022	012	00010010	DC2
019	023	013	00010011	XOFF
020	024	014	00010100	DC4
021	025	015	00010101	NAK
022	026	016	00010110	SYN
023	027	017	00010111	ETB
024	030	018	00011000	CAN
025	031	019	00011001	EM
026	032	01A	00011010	SUB
027	033	01B	00011011	ESC
028	034	01C	00011100	FS
029	035	01D	00011101	GS
030	036	01E	00011110	RS
031	037	01F	00011111	US
032	040	020	00100000	SP
033	041	021	00100001	!
034	042	022	00100010	"
035	043	023	00100011	#
036	044	024	00100100	$
037	045	025	00100101	%
038	046	026	00100110	&
039	047	027	00100111	'
040	050	028	00101000	(
041	051	029	00101001	)
042	052	02A	00101010	*
043	053	02B	00101011	+
044	054	02C	00101100	,
045	055	02D	00101101	-
046	056	02E	00101110	.
047	057	02F	00101111	/
048	060	030	00110000	0
049	061	031	00110001	1
050	062	032	00110010	2
051	063	033	00110011	3
052	064	034	00110100	4
053	065	035	00110101	5
054	066	036	00110110	6
055	067	037	00110111	7
056	070	038	00111000	8
057	071	039	00111001	9
058	072	03A	00111010	:
059	073	03B	00111011	;
060	074	03C	00111100	<
061	075	03D	00111101	=
062	076	03E	00111110	>
063	077	03F	00111111	?

Dec	Oct	Hex	Binary	Value
064	100	040	01000000	@
065	101	041	01000001	A
066	102	042	01000010	B
067	103	043	01000011	C
068	104	044	01000100	D
069	105	045	01000101	E
070	106	046	01000110	F
071	107	047	01000111	G
072	110	048	01001000	H
073	111	049	01001001	I
074	112	04A	01001010	J
075	113	04B	01001011	K
076	114	04C	01001100	L
077	115	04D	01001101	M
078	116	04E	01001110	N
079	117	04F	01001111	O
080	120	050	01010000	P
081	121	051	01010001	Q
082	122	052	01010010	R
083	123	053	01010011	S
084	124	054	01010100	T
085	125	055	01010101	U
086	126	056	01010110	V
087	127	057	01010111	W
088	130	058	01011000	X
089	131	059	01011001	Y
090	132	05A	01011010	Z
091	133	05B	01011011	[
092	134	05C	01011100	\
093	135	05D	01011101	]
094	136	05E	01011110	^
095	137	05F	01011111	_
096	140	060	01100000	`
097	141	061	01100001	a
098	142	062	01100010	b
099	143	063	01100011	c
100	144	064	01100100	d
101	145	065	01100101	e
102	146	066	01100110	f
103	147	067	01100111	g
104	150	068	01101000	h
105	151	069	01101001	i
106	152	06A	01101010	j
107	153	06B	01101011	k
108	154	06C	01101100	l
109	155	06D	01101101	m
110	156	06E	01101110	n
111	157	06F	01101111	o
112	160	070	01110000	p
113	161	071	01110001	q
114	162	072	01110010	r
115	163	073	01110011	s
116	164	074	01110100	t
117	165	075	01110101	u
118	166	076	01110110	v
119	167	077	01110111	w
120	170	078	01111000	x
121	171	079	01111001	y
122	172	07A	01111010	z
123	173	07B	01111011	{
124	174	07C	01111100	|
125	175	07D	01111101	}
126	176	07E	01111110	~
127	177	07F	01111111	DEL

Metrolog Controles de Medição Ltda
Rua Sete de Setembro, 2656 – Centro
13560-181 – São Carlos – SP - Brazil
Phone: +55 (16) 3371-0112 – +55 (16) 3372-7800
Web: **www.metrolog.net – www.metrolog.ind.br**
E-mail: **metrolog@metrolog.net**

Distribuidor

Brasil e América do Sul

CONTATO

Endereço
Rua Sete de Setembro, **2656** - Centro
13560-181 - São Carlos - SP - Brasil

Telefone
+ 55 (16) 3371-0112
+ 55 (16) 3372-7800

Internet
www.metrolog.net
metrolog@metrolog.net

Metrolog Controles de Medição