ANALYT MESSTECHNIK



COMMUNICATION GUIDE

358 Series Modbus RTU and Modbus TCP/IP

Register map, commands and worked examples for reading and controlling 358 Series mass flow and pressure devices over Modbus RTU and Modbus TCP/IP.

AMTC-CG-0001

Rev. 2 · October 2026

About this guide

This guide explains how to read measurements from, and send commands to, 358 Series mass flow meters, mass flow controllers, pressure gauges and pressure controllers over Modbus. It covers both variants of the protocol: Modbus RTU on a serial line (RS-232 or RS-485) and Modbus TCP/IP on Ethernet. It contains the register map, every device command, the status and error codes, and worked examples you can follow with your own controller or PLC.

Each register and command states the firmware version that introduced it; a device with older firmware does not have it. Register numbering, hexadecimal values and 32-bit register pairs are explained under Conventions used in this guide (p. 13).

If you are connecting a device for the first time, start with Getting connected (p. 6). If you already know Modbus and only need addresses, go straight to the Command and register reference (p. 16).

Devices with Modbus RTU also accept the standard serial commands. Those commands are described in the *358 Series serial interface (RS-232/RS-485) — Communication guide* (AMTC-CG-0002), which is a companion to this guide. The guide describes the most common features and settings; if your application needs something that is not covered here, the Analyt-MTC application engineers will be glad to help.

Conventions in this guide:

- Commands you send and replies the device returns are set in monospace type, for example `SUCCESS`.

Abbreviations

Abbreviation	Meaning
CRC	Cyclic redundancy check
DHCP	Dynamic Host Configuration Protocol
IP	Internet Protocol
NaN	Not a number (IEEE 754 invalid float value)
PLC	Programmable logic controller
RTU	Remote terminal unit
SCADA	Supervisory control and data acquisition
TCP/IP	Transmission Control Protocol / Internet Protocol
TFT	Thin-film transistor (colour display)
TFTP	Trivial File Transfer Protocol

Safety notices in this guide

NOTICE

Practices not related to personal injury — damage to the product, the process or the sample.

Note

Information that helps understanding; never a hazard.

Contact and support

ANALYT-MTC Messtechnik GmbH

German Headquarters

Klosterrunsstraße 18, 79379 Müllheim
Germany

+49 7631 5545

info@analyt-mtc.de

French Subsidiary

175 Avenue d'Alsace, 68000 Colmar
France

+33 3 89 41 47 78

info@analyt-mtc.fr

www.analyt-mtc.com

Before you call

Have the following at hand so that Analyt-MTC can help you quickly:

- the model and serial number of the device (on its label);
- whether the device is Modbus RTU or Modbus TCP/IP, and its slave ID or IP address;
- the exact request you sent (function code, start register, quantity) and the response or error code you received.

Document information

Title	358 Series Modbus RTU and Modbus TCP/IP — Communication guide
Document number	AMTC-CG-0001
Revision	Rev. 2 — October 2026
Applies to	358 Series mass flow meters and controllers, pressure gauges and controllers with Modbus RTU or Modbus TCP/IP
Language	English

Revision history

Rev.	Date	Change
2	October 2026	Quality programme: new document type, number and identity pages.
1	October 2026	First issue.

Information in this guide is subject to change without notice. © 2026 ANALYT-MTC Messtechnik GmbH. All rights reserved.

Contents

1 **Interface and wiring** 6
Modbus RTU and Modbus TCP/IP 6
Getting connected 6
Modbus RTU: serial connection 6
Slave ID 8
Modbus TCP/IP: Ethernet connection 8
Setting the IP address 10

2 **Protocol conventions** 13
What Modbus gives you 13
Devices covered 13
Conventions used in this guide 13
How Modbus works on a 358 Series device 13
Function codes 13
Register numbers and addresses 14
Reading 32-bit values 14
Checking the byte order 14

3 **Command and register reference** 16
Register overview 16
Command registers 16
Device information 17
Standard readings 17
Reading information 19
Optimized readings 19
Legacy registers 21
Commands 22
Sending a command 22
Command status 23
Command list 23
Command flow 26
Data readings 26
Control 33
Control response analysis 43
Control optimisation 46
Administrative 49
Taring 50
Communications 56
Hardware 56
Set point 57
Gas mixing 57

4 **Examples** 59

How to read the examples 59
Example 1: tare the flow reading 59
Example 2: send a set point of 50.0 59
Example 3: read the mass flow of a controller 60
Example 4: create a gas mixture 60
Example 5: change the slave ID (Modbus RTU) 60

5 Error handling 62
Symptoms and remedies 62

A Statistics 64

B Engineering units 65

C Data sources 68

D Gas numbers 69

CHAPTER 1

Interface and wiring

Modbus RTU and Modbus TCP/IP

Both variants share the register map and the commands of Chapter 3 (p. 16). They differ as follows:

	Modbus RTU	Modbus TCP/IP
Physical link	Serial, RS-232 or RS-485	Ethernet, 10BaseT or 100BaseT
Connector	8-pin Mini-DIN (default), 6-pin locking, 9/15-pin D-sub, RJ45	Two RJ45 or M12 Ethernet ports with a built-in switch
Device address	Slave ID, 1–247	IP address
Standard serial commands	Also supported	Not stated
Legacy registers	Available (Legacy registers (p. 21))	Not available
Slave ID command	Available	Not applicable

Note
An RJ45 connector on a Modbus RTU device is only a serial connector, not Ethernet. Only devices ordered with Modbus TCP/IP can be connected to a network.

Getting connected

Modbus RTU: serial connection

A Modbus RTU device connects to the serial port of your master over RS-232 or RS-485. Depending on the order, the connector is the standard 8-pin Mini-DIN, a 6-pin locking connector, a 9-pin or 15-pin D-sub, or an RJ45 socket. For the pin assignment of the Mini-DIN connector, see Appendix D of the *358 Series serial interface (RS-232/RS-485) — Communication guide* (AMTC-CG-0002). For the receive and transmit pins of the other connectors, see the operating manual of the device.

In Modbus RTU, data travels serially in 8-bit bytes. Each exchange is a request from the master followed by a response from the addressed slave (Figure 1).

The master and every slave on the line must use identical serial settings. The usual settings of a 358 Series device are:

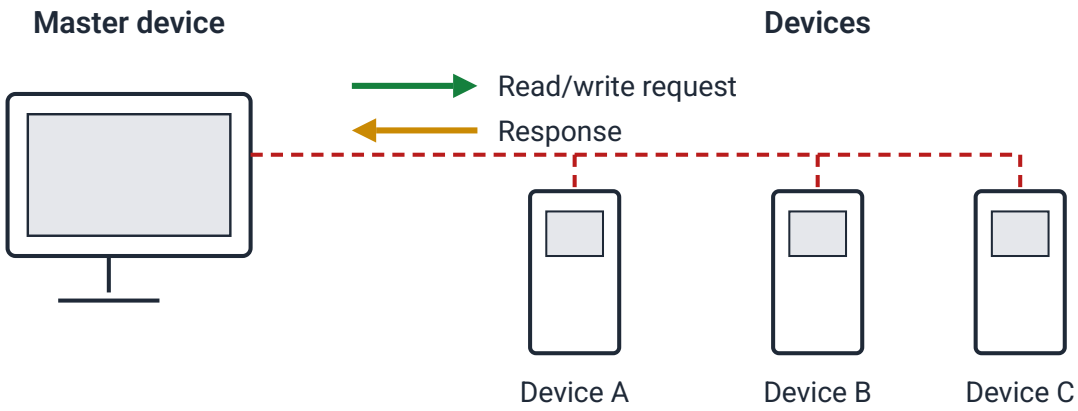


Figure 1. The master sends a read or write request; the addressed device sends the response.

Baud rate	19 200
Data bits	8
Parity	None
Stop bits	1
Flow control	None

The Modbus specification asks for even parity. A device needs special firmware to support it, and no parity, which the specification also recommends, is the normal setting.

On a device with a display, the serial settings are in *MENU > SETUP > RS-232* or *MENU > SETUP > RS-485*:

- **Modbus address** — the slave ID, 1 to 247 (see below).
- **Baud rate** — 19 200 by default. A new rate takes effect when you press SET.
- **Idle** — what a controller does when no valid request arrives for a set time: it goes to a zero set point or keeps the last one. The default is *Always Wait*, an infinite time; you can set up to 99 999.9 s.

The unit ID and the streaming interval of the standard serial commands do not apply to Modbus RTU.

NOTICE

When you change the baud rate of the device over the serial line, change the baud rate of your master port in time to read the confirmation. Otherwise the device switches, but its reply is not understood and you must set the master to the new rate to continue.

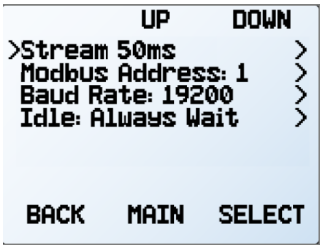


Figure 2. Serial settings: Modbus address, baud rate and idle behaviour.

Slave ID

Every device on a Modbus RTU line needs its own slave ID, a number from 1 to 247. The master puts this number in each request to say which device it is talking to. Two devices with the same ID on one line will both answer and corrupt each other's responses, so give each device a unique ID before you connect them together.

You can change the slave ID from the master with command 32767 (p. 23). Connect one device at a time, send the command with the new ID, then connect the next device. Legacy register 65 does the same; see Legacy registers (p. 21).

Modbus TCP/IP: Ethernet connection

A Modbus TCP/IP device has two Ethernet ports, both 10/100 Mbit/s, with a small switch built in. They are RJ45 sockets or, for a waterproof connection, M12 sockets. You can use either port to reach the device. The switch also passes traffic through, so you can chain devices one after another instead of running a cable from each one back to a central switch.

The direction in which the switch forwards packets can change only after a power cycle. For this reason, do not close the chain into a ring: a ring topology with failover is not supported.

Modbus TCP/IP wraps each Modbus RTU request or response into one TCP/IP packet and sends it over the Ethernet network.

RJ45 Ethernet ports

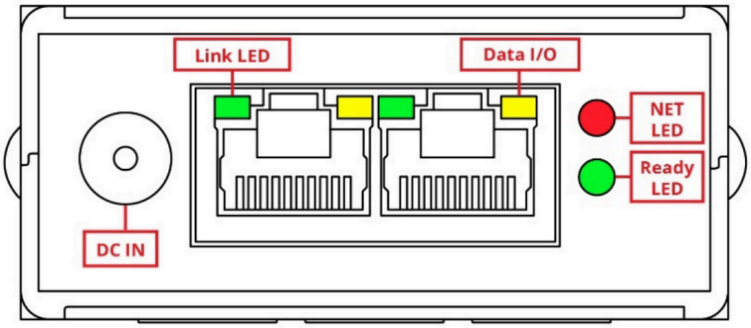


Figure 3. Device with Modbus TCP/IP and RJ45 ports.

Each RJ45 port has two LEDs. The green Link LED is lit while the port is connected to the network; if it is off, the device cannot connect to the network. The yellow Data I/O LED flashes as data packets are sent or received.

To the right of the two ports are the NET LED, nearer the back of the device, which shows whether the device is connected to a PLC, and the Ready LED, nearer the front, which shows the operating status.

LED state	Ready	NET
Off	Start-up	Start-up
Solid red	Invalid IP configuration	No active link
Flashing green	Device ready	—
Solid green	—	Link active

M12 Ethernet and M8 power ports

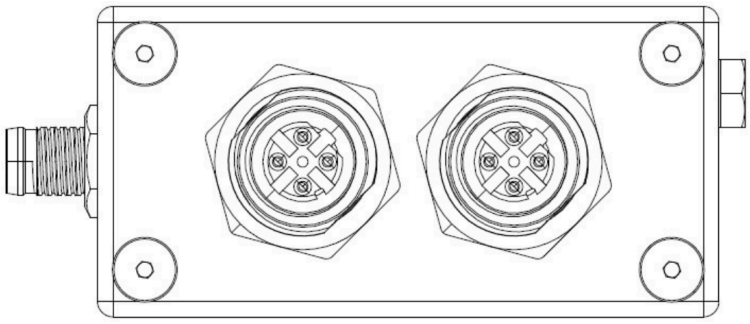


Figure 4. Device with Modbus TCP/IP and M12/M8 ports.

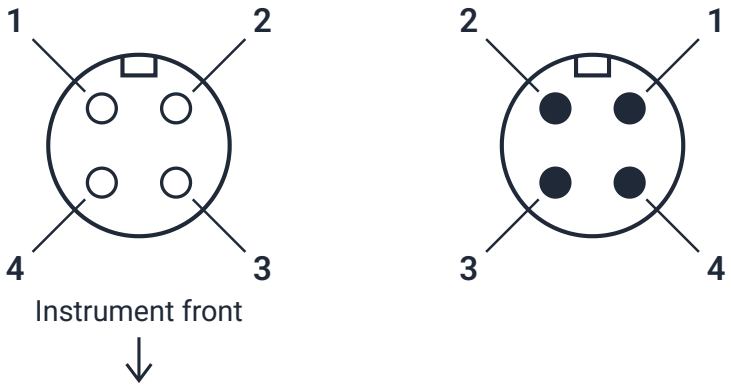


Figure 5. M12 Ethernet, seen from the top: female socket on the device (left), male plug on the cable (right).

Pin	Signal	Function
1	Tx+	Send data +
2	Rx+	Receive data +
3	Tx−	Send data −
4	Rx−	Receive data −

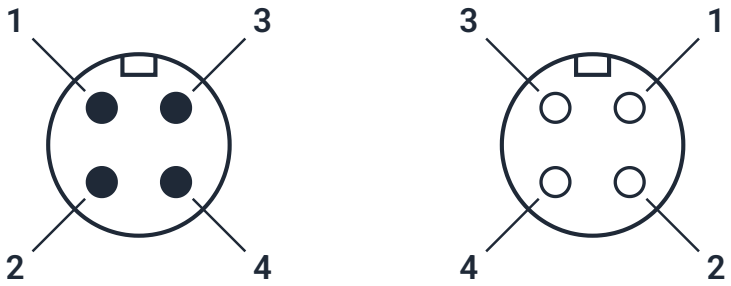


Figure 6. M8 power, seen from the side: male plug on the device (left), female socket on the cable (right).

Pin	Function
1	Reserved. Do not connect.
2	Power in: powers the device.
3	Reserved. Do not connect.
4	Ground: common ground for power and digital communication.

Setting the IP address

A new Modbus TCP/IP device asks a DHCP server for an IP address. In many plants that is all you need. If you want a fixed address, set it on the display if the device has one, or through the web server built into the device.

Device with a display

1. Choose *MENU > SETUP > Network*.
2. Select *ADR Source* and choose how the address is set: *Web Server Config* (the settings of the web server apply), *Static Address* (you enter the address on the display) or *DHCP* (a DHCP server assigns it).
3. Press SET. If the source changed, the device restarts its Ethernet interface, and Modbus cannot reach it for up to 15 seconds.
4. For a static address, select *ADR*, enter the IP address and press SET. The Ethernet interface restarts again, with the same wait of up to 15 seconds.
5. Enter the subnet mask (*MSK*) and the gateway (*GW*) in the same way, with values that suit your network.

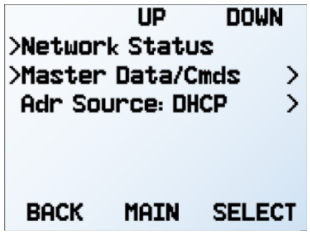


Figure 7. Network menu, address source DHCP.

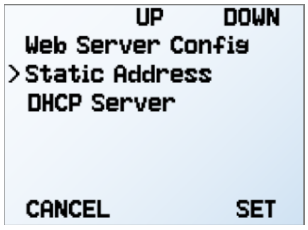


Figure 8. Options of *ADR Source*.

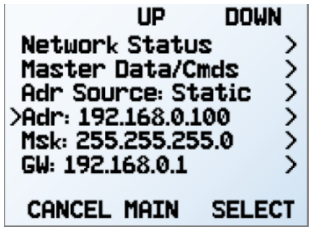


Figure 9. Static IP address, subnet mask and gateway.

The *Network Status* screen of the same menu helps when a connection fails. It shows the IP address (or *Configuring* during start-up, or *Waiting For Network* while no cable is connected or no DHCP address has arrived), the number of readings available, the link state (*Link Up* or *Link Down*) and whether a master is connected (*PLC Connected* or *PLC Not Connected*). The *Master Data/Cmds* screen behind it shows the diagnostic data exchanged with the master.

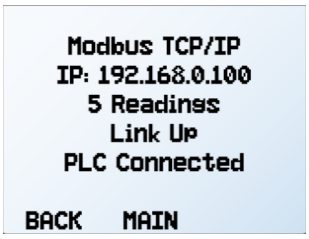


Figure 10. *Network Status* screen.

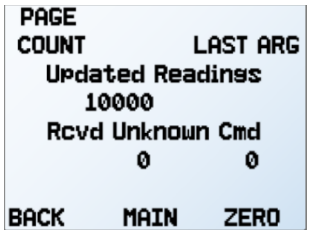


Figure 11. *Master Data/Cmds* diagnostics.

Device without a display

Without a display, assign a fixed address through the web server. The web server can only be reached once the device has an address, so the device must first obtain one by DHCP.

If you have no DHCP server on the network, you can set one up on your computer. The steps below use a Windows computer and a free DHCP server program, and connect the device directly to the computer.

1. On the computer, give an unused network interface a fixed address, for example `192.168.2.10` with subnet mask `255.255.255.0` (Figures 12 and 13).
2. Run a DHCP server on that interface, with a pool of addresses in the same range, for example starting at `192.168.2.100` (Figure 14).
3. Connect the device directly to that network interface and switch on its power.
4. Wait until the DHCP server shows that it has given an address to the device, and note the address (Figure 15).
5. Open a web browser and go to that address. The device web server opens on its *Home* tab.
6. Select the *Network Config* tab, clear the *Use DHCP* check box and enter the static IP address you want the device to use.
7. If your network needs it, enter or update the subnet mask and the gateway.
8. Click *Update*. The device reconfigures for about 15 seconds and cannot communicate during that time. Afterwards the web server answers at the new address and shows the new network settings.

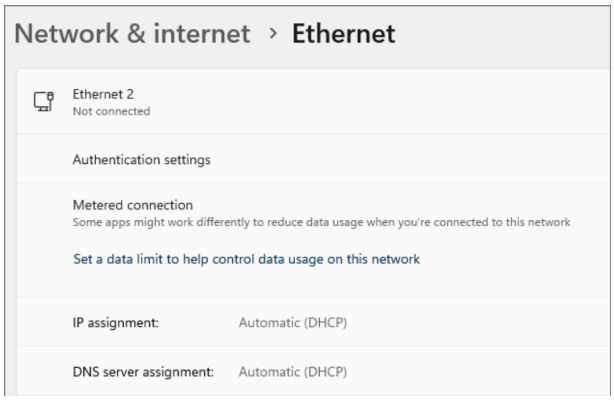


Figure 12. Windows network settings: the Ethernet interface.

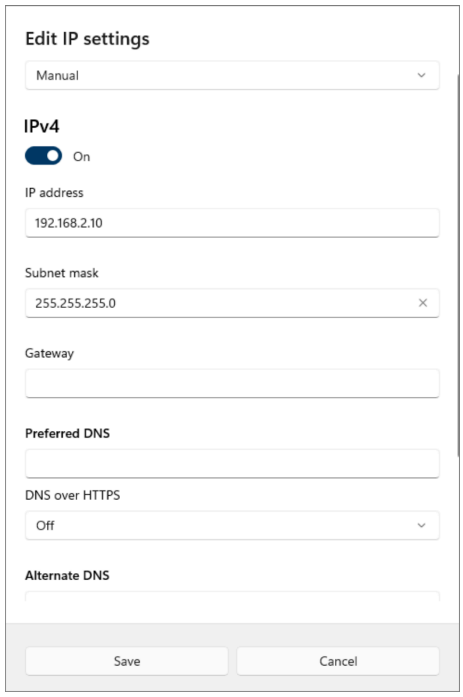


Figure 13. Windows dialog for the IP settings of the unused interface: manual address 192.168.2.10, subnet mask 255.255.255.0.

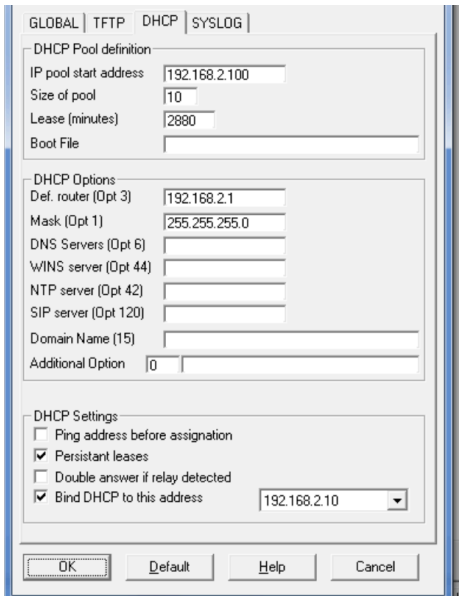


Figure 14. DHCP settings of the server program on the computer (a TFTP server): address pool starting at 192.168.2.100.

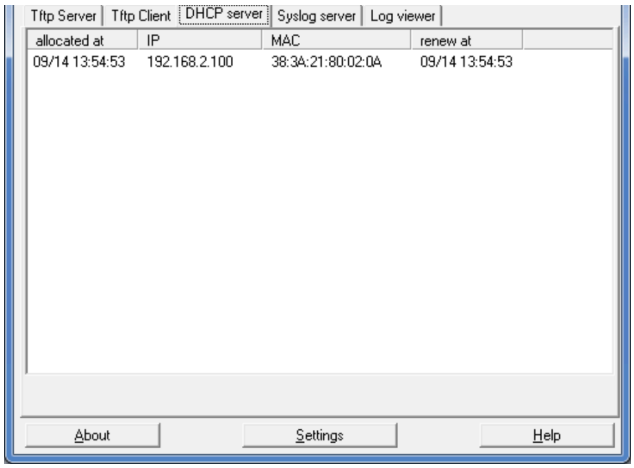


Figure 15. The DHCP server shows the address it gave to the device: 192.168.2.100.

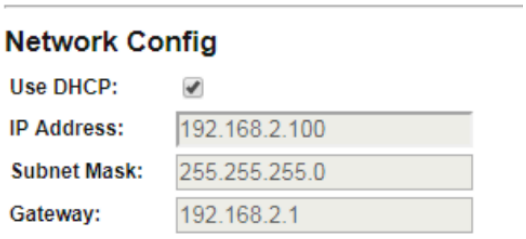


Figure 16. *Network Config* tab of the web server, with DHCP in use.

Note

The same web server also shows the statistics a device provides: open the *Data I/O* tab. See Typical reading assignments (p. 20).

CHAPTER 2

Protocol conventions

What Modbus gives you

Modbus is a messaging protocol that industrial controllers, PLCs and SCADA systems use to exchange data with field devices. A Modbus network has one master, which sends requests, and one or more slaves, which answer. The 358 Series device is the slave: your controller reads its measurements from numbered registers and writes set points and commands to others.

Over Modbus your controller can read the measurements of the device, send a set point, tare, select a gas and create gas mixtures.

Devices covered

This guide covers the four device types of the 358 Series that provide Modbus data:

- **Mass flow meters** — pressure, temperature, volumetric and mass flow, and an optional mass total.
- **Mass flow controllers** — the same readings plus a mass flow set point.
- **Pressure gauges** — pressure.
- **Pressure controllers** — pressure and a pressure set point.

Liquid flow devices are not covered. Each register and command in Chapter 3 (p. 16) states the firmware version that introduced it; a device with older firmware does not have it. The firmware version of a device is in registers 1090–1093 from firmware 10v07.0. A command that does not apply to a device, such as a set point on a meter, is ignored or answered with an error code (Chapter 3 (p. 22)).

Conventions used in this guide

Register numbers are decimal. Hexadecimal values are written with the prefix `0x`, for example `0x8002`. A register range such as `1203–1204` means two consecutive registers that together hold one 32-bit value. Values you type or send exactly as shown are set in a monospace font.

How Modbus works on a 358 Series device

The rules for addressing registers, reading numbers and choosing function codes. They apply to every register in this guide.

Function codes

The device organises its data in 16-bit registers and supports three Modbus function codes:

Code	Name
03	Read Holding Registers (Modbus RTU only)
04	Read Input Registers
16	Write Multiple Registers

Function 03 works on Modbus RTU only, so use function 04 for every read: it works on both variants, and the command procedure in Chapter 3 (p. 22) reads the command status with it. Every write uses

function 16, even when only one register changes. The examples in Chapter 4 (p. 59) use 04 for all reads.

The reply to a Write Multiple Registers message repeats the values the master sent, not the values in the registers after the write, as the Modbus specification requires. If a written value is not valid, for example a set point outside the range, the device adjusts it in the register but the reply still shows the original value; read the register to see the real value.

Reading a register that is not defined returns 65535, or an exception response with the code ILLEGAL DATA ADDRESS.

Register numbers and addresses

Register numbers in this guide follow the PLC convention, where the first register is number 1. The address that travels inside a Modbus message is zero-based, so it is one lower: register 1000 has address 999 (`0x03E7`) on the wire.

> **Note**
> Many master controllers and PLC libraries already subtract one for you, and others do not. If every value you read comes from the neighbouring register, your master uses zero-based numbers: subtract one from each register number in this guide.

Reading 32-bit values

Measurements, the set point and the device status are 32-bit values and take two consecutive registers. The higher-order half, bits 31 to 16, sits in the lower-numbered register; the lower-order half, bits 15 to 0, sits in the next one. This is the common big-endian word order, sometimes called "AB CD".

The tables of Chapter 3 (p. 16) give the data type of every register: 16-bit or 32-bit integer, 32-bit IEEE 754 single-precision float, or bit field. Within a 16-bit value, the most significant byte comes first.

Example: the number 50.0 as a 32-bit float

Register	Holds	Value
1010	Upper half (bits 31–16)	`0x4248`
1011	Lower half (bits 15–0)	`0x0000`

> **Note**
> If a value you read is nonsense but the register is right, swap the two halves in your master. Master software often offers this as a "word swap" option.

Checking the byte order

From firmware 10v19.0, registers 1088–1089 hold a fixed value that cannot be changed. Read both registers, combine them as your master does, and interpret the result as an integer or as a float. Find the value in the table below. If it is not listed, the wrong registers were read (add or subtract one from each register number) or another error occurred.

Hexadecimal	4-byte integer	Float	Meaning and action
`3F9E064B`	1067320907	1.234567	The order matches the device. Nothing to change.
`064B3F9E`	105594782	3.8226795e-35	Register order reversed: swap bytes 1 and 2 with bytes 3 and 4.
`9E3F4B06`	2654948102 / −1640019194	−1.012697e-20	Register order correct, bytes within each register reversed: swap bytes 1 and 2, and bytes 3 and 4.
`4B069E3F`	1258724927	8822335.0	Both reversed: swap bytes 1 and 4, and bytes 2 and 3.

Once the value read matches the device, test writing: write a 4-byte integer or a float to registers 1086–1087 and read it back with the now proven combination. These registers do not affect the device. If the device answers the write with an address error, add or subtract one from each register number.

CHAPTER 3

Command and register reference

Every register a 358 Series device offers and every command it accepts, with the firmware version that introduced it. The register number is used throughout; the register address is always one less.

Register overview

The registers fall into blocks. Standard readings keep the most common readings in the same registers whatever the configuration of the device; optimized readings pack the readings the device actually has into as few registers as possible. Registers not listed in this chapter cannot be used over Modbus.

3-1 Register blocks

Register	Contents	See
21–65	Legacy registers, Modbus RTU only	Legacy registers (p. 21)
1000–1009	Command registers	Command registers (p. 16)
1010–1011	Set point (float)	Optimized readings (p. 19)
1050–1059	Gas mix constituents	Command registers (p. 16)
1086–1099	Byte-order test, firmware version, serial number, dates	Device information (p. 17)
1199–1242	Optimized readings, float format	Optimized readings (p. 19)
1246–1289	Optimized readings, integer format	Optimized readings (p. 19)
1296–1321	Standard readings, integer format	Standard readings (p. 17)
1346–1371	Standard readings, float format	Standard readings (p. 17)
1400–1639	Information on optimized readings 1–20	Reading information (p. 19)
1640–1771	Information on the standard readings	Reading information (p. 19)
2041–2059	Legacy registers, Modbus RTU only	Legacy registers (p. 21)

Command registers

How to use these registers is described in Sending a command (p. 22).

3-2 Command and gas mix registers

Register	Access	Data type	Description	Firmware
1000	Write / Read	16-bit integer	Write: limited command ID. Read: ID of the last executed command.	6v17.0
1001	Write / Read	16-bit integer	Write: limited command argument. Read: result of the last executed command.	6v17.0
1002–1003	Write / Read	32-bit integer	Write: command ID. Read: ID of the command being executed or last executed.	10v07.0
1004–1005	Write / Read	32-bit integer	Write: command argument. Read: argument of the command being executed or last executed.	10v07.0
1006–1007	Read	32-bit integer	Status of the command being executed or last executed (command status (p. 23)).	10v07.0
1008–1009	Read	32-bit integer	Return value of the last executed command.	10v07.0

Register	Access	Data type	Description	Firmware
1050	Write	16-bit integer	Gas mix constituent 1: gas number.	6v17.0
1051	Write	16-bit integer	Gas mix constituent 1: fraction, 1 = 0.01 %, 10000 = 100 %.	6v17.0
1052	Write	16-bit integer	Gas mix constituent 2: gas number.	6v17.0
1053	Write	16-bit integer	Gas mix constituent 2: fraction, 1 = 0.01 %, 10000 = 100 %.	6v17.0
1054	Write	16-bit integer	Gas mix constituent 3: gas number.	6v17.0
1055	Write	16-bit integer	Gas mix constituent 3: fraction, 1 = 0.01 %, 10000 = 100 %.	6v17.0
1056	Write	16-bit integer	Gas mix constituent 4: gas number.	6v17.0
1057	Write	16-bit integer	Gas mix constituent 4: fraction, 1 = 0.01 %, 10000 = 100 %.	6v17.0
1058	Write	16-bit integer	Gas mix constituent 5: gas number.	6v17.0
1059	Write	16-bit integer	Gas mix constituent 5: fraction, 1 = 0.01 %, 10000 = 100 %.	6v17.0

Device information

3-3 Device information registers

Register	Access	Data type	Description	Firmware
1086–1087	Read/write	4 bytes	User test value; writing it does not affect the device. See Checking the byte order (p. 14).	10v19.0
1088–1089	Read	4 bytes	Fixed value: 1067320907 as a 4-byte integer, 1.234567 as a float. See Checking the byte order (p. 14).	10v19.0
1090	Read	16-bit integer	Firmware major version number	10v07.0
1091	Read	16-bit integer	Firmware minor version number	10v07.0
1092	Read	16-bit integer	Firmware custom version number	10v07.0
1093	Read	16-bit integer	Firmware internal version number	10v07.0
1094–1095	Read	32-bit integer	Serial number	10v19.0
1096	Read	16-bit integer	Manufacture month (most significant byte: value div 256) and day (least significant byte: value mod 256)	10v19.0
1097	Read	16-bit integer	Manufacture year	10v19.0
1098	Read	16-bit integer	Last calibration month (most significant byte: value div 256) and day (least significant byte: value mod 256)	10v19.0
1099	Read	16-bit integer	Last calibration year	10v19.0

Standard readings

Standard readings keep the most common readings in the same registers, whatever the configuration of the device. All standard reading registers were added in firmware 10v07.0. Each reading is available as a 4-byte integer and as a 4-byte float in engineering units. To convert an integer reading to engineering units, multiply it by 10^{-d}, where d is the decimal places register of the reading (Reading information (p. 19)).

A reading that is not valid for the current configuration reads as −2147483648 in integer format and as hexadecimal `ffffffff`, a quiet NaN, in float format.

3-4 Standard reading registers

Reading	Integer	Float	Access	Information block
Alarm output status (bit field: +1 alarm 1 active, +2 alarm 2 active)	1296	1346	Read	—
Gas number (16-bit unsigned)	1297	1347	Read	—
Device status (32-bit bit field, status bits (p. 18))	1298–1299	1348–1349	Read	—
Set point	1300–1301	1350–1351	Read/write	1640–1651
Valve drive	1302–1303	1352–1353	Read	1652–1663
Pressure (primary)	1304–1305	1354–1355	Read	1664–1675
Secondary pressure	1306–1307	1356–1357	Read	1676–1687
Barometric pressure	1308–1309	1358–1359	Read	1688–1699
Temperature	1310–1311	1360–1361	Read	1700–1711
Volumetric flow	1312–1313	1362–1363	Read	1712–1723
Mass flow	1314–1315	1364–1365	Read	1724–1735
Totaliser 1	1316–1317	1366–1367	Read	1736–1747
Totaliser 2	1318–1319	1368–1369	Read	1748–1759
Humidity	1320–1321	1370–1371	Read	1760–1771

Writing the set point registers (1300–1301 or 1350–1351) requests a new set point; reading them returns the current set point. The secondary pressure is either measured by a second pressure sensor or calculated with the barometer, if the device has one.

Device status

The device status registers (1201–1202, 1248–1249, 1298–1299 and 1348–1349) all carry the same bit field. Bit 0 is the least significant bit. Several bits can be set at once. Other bits are reserved and may read 0 or 1.

3-5 Device status bits

Bit	Value	Condition
0	+1	Temperature over range
1	+2	Temperature under range
2	+4	Volumetric flow over range
3	+8	Volumetric flow under range
4	+16	Mass flow over range
5	+32	Mass flow under range
6	+64	Pressure over range
7	+128	Totaliser has reached its maximum
8	+256	Control loop held constant (valve drive on hold)
9	+512	Internal hardware error
10	+1024	Pressure above the user-set limit
11	+2048	Flow over range while totalising

Reading information

Each standard and each optimized reading has a block of 12 read-only registers that describe it. All were added in firmware 10v07.0. The same information is available with the query commands of Data readings (p. 26).

3-6 Layout of a reading information block

Offset	Data type	Description
+0	16-bit integer	Type: a statistic value from Appendix A (p. 64), or 0 if the reading is not available
+1	16-bit integer	Source: a value from Appendix C (p. 68), or 0 if the reading is not available
+2 … +3	32-bit integer	Minimum value, without over-range, as an integer (scale by the decimal places)
+4 … +5	32-bit integer	Maximum value, without over-range, as an integer (scale by the decimal places)
+6 … +7	32-bit float	Minimum value, without over-range, in engineering units
+8 … +9	32-bit float	Maximum value, without over-range, in engineering units
+10	16-bit integer	Units: a value from Appendix B (p. 65), or 0 if the reading is not available
+11	16-bit integer	Decimal places: digits after the implied decimal point in the current, minimum and maximum integer values

The block of a standard reading is in the last column of the standard reading table (p. 17): for example the pressure block starts at 1664, so the pressure units are in register 1674 and its decimal places in register 1675. The block of optimized reading n starts at register 1400 + 12 × (n − 1); it is listed in the optimized reading table (p. 19).

Optimized readings

Optimized readings place the readings the device supports in as few registers as possible, so that one read returns all of them. Which reading sits in which slot depends on the device type and its configuration; read the type register of the slot (Reading information (p. 19)) to find out. Invalid readings read as for the standard readings.

3-7 Optimized reading registers

Reading	Float	Integer	Information block	Firmware
Set point (read/write; write only before 10v07.0)	1010–1011	—	—	6v17.0
Alarm output status	1199	1246	—	10v07.0
Gas number	1200	1247	—	6v17.0 (float), 10v07.0 (int.)
Device status	1201–1202	1248–1249	—	6v17.0 (float), 10v07.0 (int.)
Reading 1	1203–1204	1250–1251	1400–1411	6v17.0 (float), 10v07.0 (int.)
Reading 2	1205–1206	1252–1253	1412–1423	6v17.0 (float), 10v07.0 (int.)
Reading 3	1207–1208	1254–1255	1424–1435	6v17.0 (float), 10v07.0 (int.)
Reading 4	1209–1210	1256–1257	1436–1447	6v17.0 (float), 10v07.0 (int.)
Reading 5	1211–1212	1258–1259	1448–1459	6v17.0 (float), 10v07.0 (int.)

Reading	Float	Integer	Information block	Firmware
Reading 6	1213–1214	1260–1261	1460–1471	6v17.0 (float), 10v07.0 (int.)
Reading 7	1215–1216	1262–1263	1472–1483	6v17.0 (float), 10v07.0 (int.)
Reading 8	1217–1218	1264–1265	1484–1495	6v17.0 (float), 10v07.0 (int.)
Reading 9	1219–1220	1266–1267	1496–1507	6v17.0 (float), 10v07.0 (int.)
Reading 10	1221–1222	1268–1269	1508–1519	6v17.0 (float), 10v07.0 (int.)
Reading 11	1223–1224	1270–1271	1520–1531	6v17.0 (float), 10v07.0 (int.)
Reading 12	1225–1226	1272–1273	1532–1543	6v17.0 (float), 10v07.0 (int.)
Reading 13	1227–1228	1274–1275	1544–1555	6v17.0 (float), 10v07.0 (int.)
Reading 14	1229–1230	1276–1277	1556–1567	6v17.0 (float), 10v07.0 (int.)
Reading 15	1231–1232	1278–1279	1568–1579	6v17.0 (float), 10v07.0 (int.)
Reading 16	1233–1234	1280–1281	1580–1591	6v17.0 (float), 10v07.0 (int.)
Reading 17	1235–1236	1282–1283	1592–1603	6v17.0 (float), 10v07.0 (int.)
Reading 18	1237–1238	1284–1285	1604–1615	6v17.0 (float), 10v07.0 (int.)
Reading 19	1239–1240	1286–1287	1616–1627	6v17.0 (float), 10v07.0 (int.)
Reading 20	1241–1242	1288–1289	1628–1639	6v17.0 (float), 10v07.0 (int.)

Note
Modbus RTU defines the registers of all 20 float readings, whatever the number of valid readings, and returns the invalid value for a reading that does not exist. Before firmware 10v07.0, Modbus TCP/IP returns an invalid address error for the register pair of a reading that does not exist.

Typical reading assignments

A device can output up to 20 configurable statistics, which are the values in registers 1203 onward. Which statistic sits in which slot depends on the device type and its configuration; the usual assignments are in the tables below. Unless stated otherwise, each is a 32-bit float.

The pressure reading can be absolute, gauge or differential pressure, depending on how your device is configured.

3-8 Mass flow controller

Register	Statistic
1203–1204	Pressure
1205–1206	Flow temperature
1207–1208	Volumetric flow
1209–1210	Mass flow
1211–1212	Mass flow set point
1213–1214	Mass total *

3-9 Mass flow meter

Register	Statistic
1203–1204	Pressure
1205–1206	Flow temperature
1207–1208	Volumetric flow
1209–1210	Mass flow
1211–1212	Mass total *

3-10 Pressure gauge

Register	Statistic
1203–1204	Pressure

3-11 Pressure controller

Register	Statistic
1203–1204	Pressure
1205–1206	Pressure set point

The register pairs are numbered readings 1 to 20; the register itself does not say what it holds. To see which statistics and units your own device actually outputs:

- **Modbus RTU:** send the query data frame command from the standard serial command set (for unit ID A, `A??D*`). It is described in the *358 Series serial interface (RS-232/RS-485) — Communication guide* (AMTC-CG-0002).
- **Modbus TCP/IP:** open the device's IP address in a web browser and select the *Data I/O* tab.

If you want a different set of statistics, talk to the Analyt-MTC application engineers.

* Mass total is only available on devices with the totaliser option.

Legacy registers

Modbus RTU still supports these registers, but all of them have been replaced by other registers that often do more. They were never available over Modbus TCP/IP.

3-12 Legacy registers (Modbus RTU only)

Register	Access	Data type	Description
21	Read/write	16-bit integer	Closed-loop proportional gain (P)
22	Read/write	16-bit integer	Closed-loop derivative gain (D)
23	Read/write	16-bit integer	Closed-loop integral gain (I)
24	Read/write	16-bit integer	Set point, in set point counts (64000 = 100 % of full scale)
46	Read/write	16-bit integer	Gas number
65	Read/write	16-bit integer	Slave address (Modbus slave ID, 1–247)
2041–2042	Read	32-bit float	Pressure (primary), in engineering units
2043–2044	Read	32-bit float	Temperature, in engineering units
2045–2046	Read	32-bit float	Volumetric flow, in engineering units
2047–2048	Read	32-bit float	Mass flow, in engineering units
2049–2050	Read	32-bit float	Set point, in engineering units
2051–2052	Read	32-bit float	Totaliser 1, in engineering units
2053	Read	16-bit integer	Slave address

Register	Access	Data type	Description
2054	Read	16-bit unsigned integer	Gas number
2055	Read	16-bit integer	Volumetric flow over range: 0 inactive, other value active
2056	Read	16-bit integer	Mass flow over range: 0 inactive, other value active
2057	Read	16-bit integer	Pressure over range: 0 inactive, other value active
2058	Read	16-bit integer	Temperature over range: 0 inactive, other value active
2059	Read	16-bit integer	Totaliser over range: 0 inactive, other value active

Commands

Sending a command

Commands can be sent in two ways. The limited procedure works with every firmware version from 6v17 but only for command IDs below 65535, with arguments 0–65535. The full procedure needs firmware 10v07.0 or later and accepts any command ID.

> **Note**
> The device runs a new command only when the command ID or the argument differs from the values already in the registers. Send command 0, No operation, between two commands, so that a repeated command is never missed.

Limited command, Modbus RTU

1. With function 16, Write Multiple Registers, write the command ID to register 1000 and the argument to register 1001 in one message. The device replies only when the command has finished. Most commands finish at once; some, such as a tare, take noticeably longer.
2. With function 04, Read Input Registers, read register 1001. A value above 32768 is an error code (command status (p. 23)); a value of 32767 or less is the return value of the command, and many commands return 0.

Limited command, Modbus TCP/IP

1. Write the command ID to register 1000 and the argument to register 1001 in one Write Multiple Registers message.
2. Read registers 1000–1001 with Read Input Registers. If register 1000 does not hold the ID you sent, the command is queued and not yet complete: read again. Once it matches, a value above 32768 in register 1001 is an error code (command status (p. 23)); otherwise it is the return value of the command.

Any command, Modbus RTU and Modbus TCP/IP (from 10v07.0)

These registers are paired for 4 bytes of data: write the two most significant bytes into the register with the lower number. Values read back are arranged the same way.

1. With Write Multiple Registers, write the command ID to registers 1002–1003 and the argument to registers 1004–1005. The command ID is always an unsigned integer. Many commands take an integer argument, but many take a single-precision float; the description of each command gives the format. The device replies at once, without waiting for the command to finish.
2. With Read Input Registers, read registers 1002–1009. If registers 1002–1005 do not match the command and argument you sent, the request is queued; if registers 1006–1007 hold 1 (`IN_PROGRESS`), the command is still running. In both cases read again.
3. Check the status in registers 1006–1007: a non-zero status is an error (command status (p. 23)); status 0 means the return value in registers 1008–1009 is valid; its format is given with each command.

Command status

3-13 Command status values

Status	Name	Meaning	Error code in register 1001*
0	SUCCESS	The last command completed successfully.	—
1	IN_PROGRESS	A command is being executed.	—
2	INVALID_ID	The ID of the last command is invalid.	32769
3	INVALID_ARGUMENT	The argument of the last command is invalid.	32770
4	UNSUPPORTED	The device does not support the last command.	32771
5	INVALID_MIX_IDX	The requested gas mix number is invalid.	32772
6	INVALID_MIX_GAS	A gas used in the mix does not exist on the device.	32773
7	INVALID_MIX_PCT	The gas mix fractions do not add up to 100 %.	32774

* For limited commands; the full procedure reports the status value itself in registers 1006–1007.

Command list

3-14 Commands

Command ID	Command	Firmware	Devices
COMMAND FLOW			
0	No operation (p. 26)	6v17.0	All devices
DATA READINGS			
28	Query reading type (p. 26)	10v07.0	All devices
27	Query reading source (p. 27)	10v07.0	All devices
65538	Query reading minimum (integer) (p. 27)	10v07.0	All devices
65536	Query reading minimum (float) (p. 27)	10v07.0	All devices
65539	Query reading maximum (integer) (p. 27)	10v07.0	All devices
65537	Query reading maximum (float) (p. 27)	10v07.0	All devices
29	Query reading engineering units (p. 28)	10v07.0	All devices
65300 + offset	Set reading engineering units (p. 28)	10v07.0	All devices
30	Query reading decimal places (p. 28)	10v07.0	All devices
36	Totaliser 1 flow (p. 29)	10v19.0	Flow devices
34	Totaliser 1 accumulation mode (p. 29)	10v19.0	Flow devices
38	Totaliser 1 action at limit (p. 29)	10v19.0	Flow devices
37	Totaliser 2 flow (p. 29)	10v19.0	Flow devices
35	Totaliser 2 accumulation mode (p. 30)	10v19.0	Flow devices
39	Totaliser 2 action at limit (p. 30)	10v19.0	Flow devices
65560	Standard flow reference pressure (integer) (p. 30)	10v19.0	Mass flow devices
65556	Standard flow reference pressure (float) (p. 30)	10v19.0	Mass flow devices
65562	Standard flow reference temperature (integer) (p. 31)	10v19.0	Mass flow devices
65558	Standard flow reference temperature (float) (p. 31)	10v19.0	Mass flow devices
65561	Normal flow reference pressure (integer) (p. 31)	10v19.0	Mass flow devices
65557	Normal flow reference pressure (float) (p. 31)	10v19.0	Mass flow devices
65563	Normal flow reference temperature (integer) (p. 32)	10v19.0	Mass flow devices
65559	Normal flow reference temperature (float) (p. 32)	10v19.0	Mass flow devices

Command ID	Command	Firmware	Devices
50	Flow averaging time constant (p. 32)	10v19.0	Flow devices
47	Pressure averaging time constant (p. 32)	10v19.0	All devices
48	Primary pressure averaging time constant (p. 33)	10v19.0	All devices
49	Secondary pressure averaging time constant (p. 33)	10v19.0	All devices
44	Reading zero band (p. 33)	10v19.0	All devices
CONTROL			
40	Auto-tare enable (p. 33)	10v19.0	Controllers
45	Auto-tare delay (p. 34)	10v19.0	Controllers
65548	Query set point limit (integer) (p. 34)	10v19.0	Controllers
65550	Set minimum set point limit (integer) (p. 34)	10v19.0	Controllers
65552	Set maximum set point limit (integer) (p. 34)	10v19.0	Controllers
65549	Query set point limit (float) (p. 34)	10v19.0	Controllers
65551	Set minimum set point limit (float) (p. 35)	10v19.0	Controllers
65553	Set maximum set point limit (float) (p. 35)	10v19.0	Controllers
12	Save power-up set point (p. 35)	7v05.0	Controllers
65546	Set point maximum ramp (saved) (p. 36)	10v07.0	Controllers
65547	Set point maximum ramp (temporary) (p. 36)	10v07.0	Controllers
42	Set point ramp enable (p. 37)	10v19.0	Controllers
6	Hold valve(s) (p. 37)	7v05.0	Controllers
15	Set active valve (p. 38)	7v16.0	Controllers
43	Loop control variable (p. 38)	10v19.0	Controllers
11	Set loop control variable (p. 38)	7v05.0	Controllers
13	Set loop control algorithm (p. 39)	7v08.0	Controllers
14	Read closed-loop control gain (p. 39)	7v08.0	Controllers
8	Set proportional gain (p. 39)	6v17.0	Controllers
10	Set integral gain (p. 39)	7v05.0	Controllers
9	Set derivative gain (p. 40)	6v17.0	Controllers
62	Control gain gas compensation (p. 40)	10v13.0	Mass flow controllers
16	Inverse pressure control (p. 40)	7v18.0	Controllers
51	Closed loop active at zero pressure set point (p. 41)	10v19.0	Controllers
65554	Closed-loop deadband (integer) (p. 41)	10v19.0	Controllers
65555	Closed-loop deadband (float) (p. 41)	10v19.0	Controllers
41	Closed-loop deadband mode (p. 42)	10v19.0	Controllers
65543	Query totaliser batch (integer) (p. 42)	10v07.0	Flow controllers
65540	Query totaliser batch (float) (p. 42)	10v07.0	Flow controllers
65544	Set totaliser batch 1 (integer) (p. 42)	10v07.0	Flow controllers
65541	Set totaliser batch 1 (float) (p. 42)	10v07.0	Flow controllers
65545	Set totaliser batch 2 (integer) (p. 43)	10v07.0	Flow controllers
65542	Set totaliser batch 2 (float) (p. 43)	10v07.0	Flow controllers
CONTROL RESPONSE ANALYSIS			
63	Check set point response: test set point (counts) (p. 43)	10v13.0	Controllers
65565	Check set point response: test set point (integer) (p. 43)	10v13.0	Controllers
65564	Check set point response: test set point (float) (p. 44)	10v13.0	Controllers

Command ID	Command	Firmware	Devices
64	Check set point response: collection time (p. 44)	10v13.0	Controllers
66	Check set point response: run status (p. 44)	10v13.0	Controllers
67	Check set point response: result notes (p. 45)	10v13.0	Controllers
68	Check set point response: overshoot (counts) (p. 45)	10v13.0	Controllers
65567	Check set point response: overshoot (integer) (p. 45)	10v13.0	Controllers
65566	Check set point response: overshoot (float) (p. 46)	10v13.0	Controllers
69	Check set point response: timing results (p. 46)	10v13.0	Controllers
CONTROL OPTIMISATION			
70	Control optimisation maximum flow (counts) (p. 46)	10v13.0	Flow controllers
65569	Control optimisation maximum flow (integer) (p. 47)	10v13.0	Flow controllers
65568	Control optimisation maximum flow (float) (p. 47)	10v13.0	Flow controllers
71	Control optimisation speed mode (p. 47)	10v13.0	Flow controllers
72	Control optimisation goal time (p. 48)	10v13.0	Flow controllers
73	Control optimisation control algorithm (p. 48)	10v13.0	Flow controllers
74	Control optimisation run state (p. 48)	10v13.0	Flow controllers
75	Control optimisation result (p. 49)	10v13.0	Flow controllers
ADMINISTRATIVE			
4	Perform tare (p. 50)	7v05.0	All devices
31	Perform pressure sensor tare (p. 50)	10v07.0	All devices
32	Perform secondary pressure sensor tare (p. 51)	10v07.0	All devices
33	Perform flow tare (p. 51)	10v07.0	Flow devices
80	Reset totaliser (p. 51)	10v19.0	Flow devices
5	Reset totaliser 1 (p. 51)	7v05.0	Flow devices
1	Set selected gas (p. 52)	7v05.0	Mass flow devices
2	Create or update gas mix (p. 52)	7v05.0	Mass flow devices
52, 54, 56, 58, 60	Set gas mix constituent gas number (p. 52)	10v19.0	Mass flow devices
53, 55, 57, 59, 61	Set gas mix constituent fraction (p. 52)	10v19.0	Mass flow devices
3	Delete gas mix (p. 53)	7v05.0	Mass flow devices
24	Set relative humidity (p. 53)	9v04.0	Devices with humidity sensor
25	Set relative humidity reference temperature (p. 53)	10v07.0	Devices with humidity sensor
7	Lock or unlock display (p. 53)	7v05.0	All devices
20	Flash display (p. 53)	8v28.0	All devices
65570	Read serial number (p. 54)	10v19.0	All devices
81	Read partial serial number (p. 54)	10v19.0	All devices
82	Read manufacture date (p. 54)	10v19.0	All devices
83	Read calibration date (p. 54)	10v19.0	All devices
84	Read firmware version (p. 55)	10v19.0	All devices
17	Read configuration checksum (p. 55)	8v24.0	All devices
26	Restore factory settings (p. 55)	10v07.0	All devices
COMMUNICATIONS			
32767	Set Modbus slave address (p. 56)	7v05.0	Modbus RTU only
32765	Set communications watchdog timeout (p. 56)	8v24.0	Modbus RTU only

Command ID	Command	Firmware	Devices
HARDWARE			
18	Set primary analogue output value (p. 56)	8v24.0	All devices
19	Set secondary analogue output value (p. 57)	8v24.0	All devices

Each command below shows, on the left, its name, the firmware version that introduced it and the devices it applies to; on the right, the command ID, what it does, its argument and the response. Older firmware branches that also have the command are listed under the description.

Command flow

No operation

`6v17.0` All devices

> `Command ID` **`0`**

Does nothing. Send it between two identical commands, or before every command, so that a repeated command is never missed.

`Argument` Ignored.

Data readings

The information registers (Reading information (p. 19)) give the same data as these query commands: type, source, limits, units and decimal places of each reading.

Query reading type

`10v07.0` All devices

> `Command ID` **`28`**

Returns the type of a reading, a statistic value of Appendix A (p. 64).

`Argument` The reading, from the table below.

Value	Meaning
0	Set point reading
1	Valve drive reading
2	Pressure reading
3	Secondary pressure reading
4	Barometric pressure reading
5	Temperature reading
6	Volumetric flow reading
7	Mass flow reading
8	Totaliser 1 reading
9	Totaliser 2 reading
10	Humidity reading
1000–1019	Optimized reading n (1000 = reading 1 … 1019 = reading 20)

Response: `SUCCESS` with the type in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Query reading source

10v07.0 All devices

`Command ID` **27**

Returns the source of a reading, a value of Appendix C (p. 68).

`Argument` The reading, as for query reading type (p. 26).

Response: `SUCCESS` with the source in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Query reading minimum (integer)

10v07.0 All devices

`Command ID` **65538**

Returns the minimum of the reading, without over-range, as an integer. Multiply it by $10^{-\text{decimal places}}$ for engineering units.

`Argument` The reading, as for query reading type (p. 26).

Response: `SUCCESS` with the value in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Query reading minimum (float)

10v07.0 All devices

`Command ID` **65536**

Returns the minimum of the reading, without over-range, as a float.

`Argument` The reading, as for query reading type (p. 26).

Response: `SUCCESS` with the value in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Query reading maximum (integer)

10v07.0 All devices

`Command ID` **65539**

Returns the maximum of the reading, without over-range, as an integer. Multiply it by $10^{-\text{decimal places}}$ for engineering units.

`Argument` The reading, as for query reading type (p. 26).

Response: `SUCCESS` with the value in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Query reading maximum (float)

10v07.0 All devices

`Command ID` **65537**

Returns the maximum of the reading, without over-range, as a float.

`Argument` The reading, as for query reading type (p. 26).

Response: `SUCCESS` with the value in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Query reading engineering units

`10v07.0` All devices

`Command ID 29`

Returns the engineering units of a reading, a value of Appendix B (p. 65).

`Argument` The reading, as for query reading type (p. 26).

Response: `SUCCESS` with the units in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Set reading engineering units

`10v07.0` All devices

`Command ID 65300 + offset`

Changes the engineering units of a reading. Add the offset of the reading (table below) to the command ID 65300. Readings are not all independent: changing the units of the pressure reading also changes those of the secondary and barometric pressure.

`Argument` The value of the new units from Appendix B (p. 65).

Offset	Meaning
0	Set point reading
1	Valve drive reading
2	Pressure reading
3	Secondary pressure reading
4	Barometric pressure reading
5	Temperature reading
6	Volumetric flow reading
7	Mass flow reading
8	Totaliser 1 reading
9	Totaliser 2 reading
10	Humidity reading
100–119	Optimized reading n (100 = reading 1 … 119 = reading 20)

Response: `SUCCESS` with the new units in the return value; `UNSUPPORTED` if the offset is not a reading of the device; `INVALID_ARGUMENT` if the units do not suit the reading.

Query reading decimal places

`10v07.0` All devices

`Command ID 30`

Returns the number of digits after the implied decimal point in the current value, minimum and maximum of a reading.

`Argument` The reading, as for query reading type (p. 26).

Response: `SUCCESS` with the number of decimal places in the return value; `INVALID_ARGUMENT` if the device has no such reading.

Totaliser 1 flow

10v19.0 Flow devices

`Command ID` **36**

Reads or sets which flow totaliser 1 accumulates.

`Argument` A value from the table below.

Value	Meaning
0	Query; do not change the flow being accumulated.
1	Disable this totaliser.
4	Accumulate volumetric flow.
5	Accumulate mass flow.

Response: `SUCCESS`, and the totaliser accumulates the requested flow; `INVALID_ARGUMENT` if the value is not valid.

Totaliser 1 accumulation mode

10v19.0 Flow devices

`Command ID` **34**

Reads or sets how totaliser 1 accumulates flow.

`Argument` A value from the table below.

Value	Meaning
−1	Query; do not change the current mode.
0	Add positive flow, ignore negative flow.
1	Subtract negative flow, ignore positive flow.
2	Add positive flow and subtract negative flow.
3	Add positive flow; once flow stops, reset the volume when flow starts again.

Response: `SUCCESS`, and the totaliser accumulates in the requested mode; `INVALID_ARGUMENT` if the mode is not valid.

Totaliser 1 action at limit

10v19.0 Flow devices

`Command ID` **38**

Reads or sets what totaliser 1 does when it reaches the largest value it can hold.

`Argument` −1 queries the current action. For the values that select an action, contact Analyt-MTC.

Response: `SUCCESS`, and the totaliser performs the requested action at its limit; `INVALID_ARGUMENT` if the value is not valid.

Totaliser 2 flow

10v19.0 Flow devices

`Command ID` **37**

Reads or sets which flow totaliser 2 accumulates.

`Argument` A value from the table of Totaliser 1 flow (p. 29).

Response: `SUCCESS`, and the totaliser accumulates the requested flow; `INVALID_ARGUMENT` if the value is not valid.

Totaliser 2 accumulation mode

10v19.0 Flow devices

`Command ID` **35**

Reads or sets how totaliser 2 accumulates flow.

`Argument` A value from the table of Totaliser 1 accumulation mode (p. 29).

Response: `SUCCESS`, and the totaliser accumulates in the requested mode; `INVALID_ARGUMENT` if the mode is not valid.

Totaliser 2 action at limit

10v19.0 Flow devices

`Command ID` **39**

Reads or sets what totaliser 2 does when it reaches the largest value it can hold.

`Argument` −1 queries the current action. For the values that select an action, contact Analyt-MTC.

Response: `SUCCESS`, and the totaliser performs the requested action at its limit; `INVALID_ARGUMENT` if the value is not valid.

Standard flow reference pressure (integer)

10v19.0

Mass flow devices

`Command ID` **65560**

Reads or sets the reference pressure for standard flow (units that begin with S).

`Argument` 0 or a negative value queries. A positive integer sets the reference, in the units of the primary pressure, multiplied by $10^{\text{pressure decimal places}}$.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is too large; `UNSUPPORTED` if the device does not allow the reference to be set.

Standard flow reference pressure (float)

10v19.0

Mass flow devices

`Command ID` **65556**

Reads or sets the reference pressure for standard flow (units that begin with S).

`Argument` 0 or a negative value queries. A positive IEEE 754 single-precision float sets the reference, in the units of the primary pressure.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is not a valid pressure; `UNSUPPORTED` if the device does not allow the reference to be set.

Standard flow reference temperature (integer)

`10v19.0`

Mass flow devices

`Command ID` **65562**

Reads or sets the reference temperature for standard flow (units that begin with S).

`Argument` A value that is −500 or less once scaled to the temperature units queries. A positive integer sets the reference, in the temperature units, multiplied by $10^{\text{temperature decimal places}}$.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is not a valid temperature; `UNSUPPORTED` if the device does not allow the reference to be set.

Standard flow reference temperature (float)

`10v19.0`

Mass flow devices

`Command ID` **65558**

Reads or sets the reference temperature for standard flow (units that begin with S).

`Argument` −500 or less queries. A positive IEEE 754 single-precision float sets the reference, in the temperature units.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is not a valid temperature; `UNSUPPORTED` if the device does not allow the reference to be set.

Normal flow reference pressure (integer)

`10v19.0`

Mass flow devices

`Command ID` **65561**

Reads or sets the reference pressure for normal flow (units that begin with N).

`Argument` 0 or a negative value queries. A positive integer sets the reference, in the units of the primary pressure, multiplied by $10^{\text{pressure decimal places}}$.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is too large; `UNSUPPORTED` if the device does not allow the reference to be set.

Normal flow reference pressure (float)

`10v19.0`

Mass flow devices

`Command ID` **65557**

Reads or sets the reference pressure for normal flow (units that begin with N).

`Argument` 0 or a negative value queries. A positive IEEE 754 single-precision float sets the reference, in the units of the primary pressure.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is not a valid pressure; `UNSUPPORTED` if the device does not allow the reference to be set.

Normal flow reference temperature (integer)

10v19.0

Mass flow devices

`Command ID 65563`

Reads or sets the reference temperature for normal flow (units that begin with N).

`Argument` A value that is −500 or less once scaled to the temperature units queries. A positive integer sets the reference, in the temperature units, multiplied by $10^{\text{temperature decimal places}}$.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is not a valid temperature; `UNSUPPORTED` if the device does not allow the reference to be set.

Normal flow reference temperature (float)

10v19.0

Mass flow devices

`Command ID 65559`

Reads or sets the reference temperature for normal flow (units that begin with N).

`Argument` −500 or less queries. A positive IEEE 754 single-precision float sets the reference, in the temperature units.

Response: `SUCCESS` with the reference in the return value; `INVALID_ARGUMENT` if the reference is not a valid temperature; `UNSUPPORTED` if the device does not allow the reference to be set.

Flow averaging time constant

10v19.0 Flow devices

`Command ID 50`

Reads or sets the averaging time constant for flow readings.

`Argument` 65535 queries. Any other value sets the time constant, an integer in milliseconds.

Response: `SUCCESS` with the time constant in the return value; `INVALID_ARGUMENT` if it is negative or larger than 65535.

Pressure averaging time constant

10v19.0 All devices

`Command ID 47`

Reads or sets the averaging time constant for all pressure readings.

`Argument` 65535 queries. Any other value sets the time constant, an integer in milliseconds.

Response: `SUCCESS` with the time constant in the return value; `INVALID_ARGUMENT` if it is negative or larger than 65535.

Primary pressure averaging time constant

10v19.0 · All devices

`Command ID 48`

Reads or sets the averaging time constant for the primary pressure sensor.

`Argument` 65535 queries. Any other value sets the time constant, an integer in milliseconds.

Response: `SUCCESS` with the time constant in the return value; `INVALID_ARGUMENT` if it is negative or larger than 65535.

Secondary pressure averaging time constant

10v19.0 · All devices

`Command ID 49`

Reads or sets the averaging time constant for the secondary pressure sensor.

`Argument` 65535 queries. Any other value sets the time constant, an integer in milliseconds.

Response: `SUCCESS` with the time constant in the return value; `INVALID_ARGUMENT` if it is negative or larger than 65535.

Reading zero band

10v19.0 · All devices

`Command ID 44`

Reads or sets the band around zero in which all pressure and flow readings are reported as zero, on the digital outputs and on the display.

`Argument` 65535 queries. Otherwise the zero band, one count = 0.01 % of the full scale of the reading: 1 = no zero band, 638 = the largest zero band, 6.38 % of full scale.

Response: `SUCCESS` with the zero band in the return value; `INVALID_ARGUMENT` if the value is not valid.

Control

Commands for controllers only: set point, valves and totaliser batches.

Auto-tare enable

10v19.0 · Controllers

`Command ID 40`

Reads, enables or disables auto-tare.

`Argument` A value from the table below.

Value	Meaning
0	Query the current setting.
1	Disable auto-tare.
2	Enable auto-tare.

Response: `SUCCESS` with the current state in the return value; `INVALID_ARGUMENT` if the mode is not valid; `UNSUPPORTED` if the device is not a controller.

Auto-tare delay

10v19.0 Controllers

`Command ID` **45**

Reads or sets the delay between a zero set point and the start of an auto-tare.

`Argument` A value from the table below.

Value	Meaning
0	Query the current delay.
1–255	Set the delay; one count is 0.1 s, so 1 = 0.1 s and 255 = 25.5 s.

Response: `SUCCESS` with the current delay in the return value; `INVALID_ARGUMENT` if the delay is not valid; `UNSUPPORTED` if the device is not a controller.

Query set point limit (integer)

10v19.0 Controllers

`Command ID` **65548**

Returns the minimum or maximum set point limit as an integer.

`Argument` 0 queries the minimum set point limit, 1 the maximum.

Response: `SUCCESS` with the limit in the return value; `INVALID_ARGUMENT` if the device is not a controller.

Set minimum set point limit (integer)

10v19.0 Controllers

`Command ID` **65550**

Sets the minimum set point limit as an integer.

`Argument` The limit as an integer in the set point units (Appendix B (p. 65)), multiplied by $10^{\text{set point decimal places}}$.

Response: `SUCCESS` with the new limit in the return value; `INVALID_ARGUMENT` if the limit is too large or too small; `UNSUPPORTED` if the device is not a controller.

Set maximum set point limit (integer)

10v19.0 Controllers

`Command ID` **65552**

Sets the maximum set point limit as an integer.

`Argument` The limit as an integer in the set point units (Appendix B (p. 65)), multiplied by $10^{\text{set point decimal places}}$.

Response: `SUCCESS` with the new limit in the return value; `INVALID_ARGUMENT` if the limit is too large or too small; `UNSUPPORTED` if the device is not a controller.

Query set point limit (float)

10v19.0 Controllers

`Command ID` **65549**

Returns the minimum or maximum set point limit as a float.

`Argument` 0 queries the minimum set point limit, 1 the maximum.

Response: `SUCCESS` with the limit in the return value; `INVALID_ARGUMENT` if the device is not a controller.

Set minimum set point limit (float)

`10v19.0` Controllers

> `Command ID` **65551**

Sets the minimum set point limit; the command is documented as a float command.

`Argument` The limit as an integer in the set point units (Appendix B (p. 65)), multiplied by $10^{\text{set point decimal places}}$.

Response: `SUCCESS` with the new limit in the return value; `INVALID_ARGUMENT` if the limit is too large or too small; `UNSUPPORTED` if the device is not a controller.

Set maximum set point limit (float)

`10v19.0` Controllers

> `Command ID` **65553**

Sets the maximum set point limit; the command is documented as a float command.

`Argument` The limit as an integer in the set point units (Appendix B (p. 65)), multiplied by $10^{\text{set point decimal places}}$.

Response: `SUCCESS` with the new limit in the return value; `INVALID_ARGUMENT` if the limit is too large or too small; `UNSUPPORTED` if the device is not a controller.

Save power-up set point

`7v05.0` Controllers

> `Command ID` **12**

Stores the current set point; the controller uses it straight away the next time it is powered.

Also in 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6, 7v00.8, 7v00.9, 7v00.10, 7v00.11, 7v00.12, 7v00.15.

`Argument` Ignored.

Response: `SUCCESS`; the current set point is saved as the power-up set point.

Set point maximum ramp (saved)

10v07.0 Controllers

`Command ID` **65546**

Sets the maximum ramp rate of the set point and keeps it across power cycles. Do not send it more often than every few minutes.

`Argument` A negative value queries; 0 disables the maximum ramp. To set a ramp, take the change in percent of full scale per millisecond and multiply it by 10000000. Example: 1 % of full scale per second is 0.001 % per millisecond, which gives 10000. More values in the table below.

Value	Meaning
1000000000	100 % of full scale every millisecond
10000000	1 % of full scale every millisecond
1000000	100 % of full scale every second
16667	100 % of full scale every minute
10000	1 % of full scale every second
278	100 % of full scale every hour
167	1 % of full scale every minute
28	10 % of full scale every hour

Response: `SUCCESS` with the current maximum ramp in the return value; multiply it by 0.0000001 for percent of full scale per millisecond. `UNSUPPORTED` if the device does not support set point ramping.

Set point maximum ramp (temporary)

10v07.0 Controllers

`Command ID` **65547**

Sets the maximum ramp rate of the set point without saving it: it is lost when the device loses power.

`Argument` A value from the table of Set point maximum ramp (saved) (p. 36).

Response: `SUCCESS` with the current maximum ramp in the return value; multiply it by 0.0000001 for percent of full scale per millisecond. `UNSUPPORTED` if the device does not support set point ramping.

Set point ramp enable

`10v19.0` Controllers

`Command ID` **42**

Reads or sets when the maximum set point ramp applies.

`Argument` 65535 queries. To change the state, send a bit field from the table below.

Bit	Meaning
0	Set: the set point jumps to its initial value at power-up. Clear: it ramps there from 0.
1	Set: the set point jumps when it is commanded to 0. Clear: it ramps to 0.
2	Set: the set point jumps to a higher value. Clear: it ramps up to it.
3	Set: the set point jumps to a lower value. Clear: it ramps down to it.

Response: `SUCCESS` with the current enables in the return value; `INVALID_ARGUMENT` if the value is not valid; `UNSUPPORTED` if the device is not a controller or does not support ramping.

Hold valve(s)

`7v05.0` Controllers

`Command ID` **6**

Holds the valve(s) and stops further control of the process, or cancels a hold.

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` A value from the table below.

Value	Meaning
0	Cancel the valve hold and resume closed-loop control.
1	Hold all valves closed.
2	Hold the valves at their current positions.
3	Exhaust: close the upstream valve and open the downstream valve fully. Dual-valve controllers only.

Response: `SUCCESS` if the valves are set as requested; `UNSUPPORTED` if the device does not support that mode; `INVALID_ARGUMENT` if the value is not valid.

Set active valve

`7v16.0` Controllers

`Command ID` **15**

Only on stream-switching (multi-valve) controllers. Selects the valve that controls the process.

Also in 7v15.3.

`Argument` A value from the table below.

Value	Meaning
0	Upstream valve, or the only valve
1	Downstream valve, or an auxiliary valve wired in the downstream position
2	First auxiliary valve (from 10v11.0)
3	Second auxiliary valve (from 10v11.0)
65535	Query the active valve (from 10v11.0)

Response: `SUCCESS` if the active valve changes; the return value is the active valve (0 to 3) for argument 65535, else 0. `UNSUPPORTED` if the device is not a stream-switching (multi-valve) controller; `INVALID_ARGUMENT` if the value is not valid.

Loop control variable

`10v19.0` Controllers

`Command ID` **43**

Reads or changes the process variable that drives closed-loop control.

`Argument` 65535 queries. Otherwise a value from the table below.

Value	Meaning
0	Mass flow
1	Volumetric flow
2	Differential pressure
3	Absolute pressure
4	Gauge pressure
other	Any set point statistic of Appendix A (p. 64) (command 11: from 8v24.0)

Response: `SUCCESS` with the current variable in the return value; `INVALID_ARGUMENT` if the variable does not apply to the device; `UNSUPPORTED` if the device is not a controller.

Set loop control variable

`7v05.0` Controllers

`Command ID` **11**

Changes the statistic that the controller controls, so a mass flow controller can be set to control pressure or volumetric flow.

Also in 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6, 7v00.8, 7v00.9, 7v00.10, 7v00.11, 7v00.12, 7v00.15.

`Argument` A value from the table of loop control variable (p. 38).

Response: `SUCCESS` if the variable changes; `INVALID_ARGUMENT` if the statistic is not available on the device (for example mass flow on a pressure controller).

Set loop control algorithm

`7v08.0` Controllers

`Command ID 13`

Changes the active control algorithm.

Also in 7v07.1.

`Argument` A value from the table below.

Value	Meaning
1	PDF closed-loop control algorithm
2	PD²I closed-loop control algorithm

Response: `SUCCESS` if the algorithm is selected; `INVALID_ARGUMENT` if the value is not a valid algorithm.

Read closed-loop control gain

`7v08.0` Controllers

`Command ID 14`

Returns the current value of a gain of the control algorithm.

Also in 7v07.1.

`Argument` A value from the table below.

Value	Meaning
0	Proportional gain (P), PDF or PD²I
1	Derivative gain (D), PDF or PD²I
2	Integral gain (I), PD²I only

Response: `SUCCESS` with the gain, 0–65535, in the return value; `INVALID_ARGUMENT` if the gain does not exist for the active algorithm.

Set proportional gain

`6v17.0` Controllers

`Command ID 8`

Sets the proportional gain (P).

`Argument` A value from 0 to 65535.

Response: `SUCCESS` when the gain is set.

Set integral gain

`7v05.0` Controllers

`Command ID 10`

Sets the integral gain (I).

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` A value from 0 to 65535.

Response: `SUCCESS` when the gain is set.

Set derivative gain

6v17.0 Controllers

`Command ID` **9**

Sets the derivative gain (D).

`Argument` A value from 0 to 65535.

Response: `SUCCESS` when the gain is set.

Control gain gas compensation

10v13.0

Mass flow controllers

`Command ID` **62**

Reads or sets whether the closed-loop gains are compensated for the gas.

`Argument` A value from the table below.

Value	Meaning
0	Query only; do not change the gas compensation.
1	Disable gas compensation of the closed-loop gains.
2	Enable gas compensation of the closed-loop gains.
3	Enable gas compensation until the user changes a closed-loop gain.

Response: `SUCCESS` with the current mode in the return value; `UNSUPPORTED` if the device is not a controller or cannot set the gas properties; `INVALID_ARGUMENT` if the mode is not valid.

Inverse pressure control

7v18.0 Controllers

`Command ID` **16**

Reads, enables or disables inverse pressure control.

`Argument` A value from the table below.

Value	Meaning
0	Normal pressure control, not saved across power cycles
1	Inverse (usually back-pressure) control, not saved across power cycles
3	Normal pressure control, saved across power cycles
4	Inverse (usually back-pressure) control, saved across power cycles
65535	Query whether the control loop uses inverse action (from 10v19.0)

Response: `SUCCESS` if the mode changes; `INVALID_ARGUMENT` if the value is not available on the device or has no pressure control mode.

Closed loop active at zero pressure set point

`10v19.0` Controllers

`Command ID` **51**

Reads, enables or disables closed-loop control at a zero pressure set point. When flow is controlled, the valves always close at a zero set point. With the loop disabled, the valves behave as in the table below at a zero pressure set point.

`Argument` 65535 queries; 0 disables, 1 enables closed-loop control at a zero pressure set point.

Dual valve	Inverse pressure control	Negative pressure range	Valves at zero set point without active control
No	No	No	Close valve
No	No	Yes	Open valve
No	Yes	No	Open valve
No	Yes	Yes	Close valve
Yes	—	No	Open downstream valve
Yes	—	Yes	Open upstream valve

Response: `SUCCESS` with the current state in the return value; `INVALID_ARGUMENT` if the value is not valid; `UNSUPPORTED` if the device is not a controller.

Closed-loop deadband (integer)

`10v19.0` Controllers

`Command ID` **65554**

Reads or sets how far the process value may drift from the set point and stay in the deadband, as an integer. Control deadband is rarely used, except for pressure control with a dual-valve controller.

`Argument` −1 queries. Otherwise an integer in the set point units, multiplied by $10^{\text{set point decimal places}}$.

Response: `SUCCESS` with the new limit in the return value; `INVALID_ARGUMENT` if the limit is too large or too small; `UNSUPPORTED` if the device is not a controller.

Closed-loop deadband (float)

`10v19.0` Controllers

`Command ID` **65555**

Reads or sets how far the process value may drift from the set point and stay in the deadband, as an IEEE 754 single-precision float. Control deadband is rarely used, except for pressure control with a dual-valve controller.

`Argument` −1 queries. Otherwise a float in the set point units.

Response: `SUCCESS` with the new limit in the return value; `INVALID_ARGUMENT` if the limit is too large or too small; `UNSUPPORTED` if the device is not a controller.

Closed-loop deadband mode

`10v19.0` Controllers

`Command ID 41`

Reads or sets what the valves do inside the deadband.

`Argument` 0 queries; 1 closes the valves inside the deadband; 2 holds the valve positions inside the deadband.

Response: `SUCCESS` with the mode in the return value; `INVALID_ARGUMENT` if the mode is not valid; `UNSUPPORTED` if the device is not a controller.

Query totaliser batch (integer)

`10v07.0` Flow controllers

`Command ID 65543`

Returns the batch size of a totaliser as an integer; multiply it by $10^{-\text{totaliser decimal places}}$ for the totaliser units.

`Argument` 1 for totaliser 1, 2 for totaliser 2.

Response: `SUCCESS` with the batch size in the return value (0 if batching is disabled); `INVALID_ARGUMENT` if the totaliser is not enabled.

Query totaliser batch (float)

`10v07.0` Flow controllers

`Command ID 65540`

Returns the batch size of a totaliser as a float, in the totaliser units.

`Argument` 1 for totaliser 1, 2 for totaliser 2.

Response: `SUCCESS` with the batch size in the return value (0 if batching is disabled); `INVALID_ARGUMENT` if the totaliser is not enabled.

Set totaliser batch 1 (integer)

`10v07.0` Flow controllers

`Command ID 65544`

Sets the batch volume of totaliser 1 as an integer.

`Argument` The batch size multiplied by $10^{\text{totaliser decimal places}}$; 0 disables batching.

Response: `SUCCESS` with the batch size in the return value; `UNSUPPORTED` if totaliser 1 is not enabled; `INVALID_ARGUMENT` if the batch is larger than the totaliser can hold.

Set totaliser batch 1 (float)

`10v07.0` Flow controllers

`Command ID 65541`

Sets the batch volume of totaliser 1 as an IEEE 754 single-precision float.

`Argument` The batch size as a float in the totaliser units; 0 disables batching.

Response: `SUCCESS` with the batch size in the return value; `UNSUPPORTED` if totaliser 1 is not enabled; `INVALID_ARGUMENT` if the batch is larger than the totaliser can hold.

Set totaliser batch 2 (integer)

`10v07.0` Flow controllers

`Command ID` **65545**

Sets the batch volume of totaliser 2 as an integer.

`Argument` The batch size multiplied by $10^{\text{totaliser decimal places}}$; 0 disables batching.

Response: `SUCCESS` with the batch size in the return value; `UNSUPPORTED` if totaliser 2 is not enabled; `INVALID_ARGUMENT` if the batch is larger than the totaliser can hold.

Set totaliser batch 2 (float)

`10v07.0` Flow controllers

`Command ID` **65542**

Sets the batch volume of totaliser 2 as an IEEE 754 single-precision float.

`Argument` The batch size as a float in the totaliser units; 0 disables batching.

Response: `SUCCESS` with the batch size in the return value; `UNSUPPORTED` if totaliser 2 is not enabled; `INVALID_ARGUMENT` if the batch is larger than the totaliser can hold.

Control response analysis

A set point response check steps the set point to a test value, records the process for a set time and calculates overshoot and timing.

Check set point response: test set point (counts)

`10v13.0` Controllers

`Command ID` **63**

Reads or sets the test set point of the check, in set point counts.

`Argument` The test set point, 0–65534; 65535 queries.

Response: `SUCCESS` with the test set point in counts in the return value; `UNSUPPORTED` if the device is not a controller; `INVALID_ARGUMENT` if the value is outside 0–65535.

Check set point response: test set point (integer)

`10v13.0` Controllers

`Command ID` **65565**

Reads or sets the test set point as an integer; scale the return value by $10^{-\text{set point decimal places}}$.

`Argument` The test set point in set point units multiplied by $10^{\text{set point decimal places}}$; −2147483648 queries.

Response: `SUCCESS` with the test set point in the return value; `INVALID_ARGUMENT` if it is outside the valid range; `UNSUPPORTED` if the device is not a controller.

Check set point response: test set point (float)

10v13.0 Controllers

`Command ID` **65564**

Reads or sets the test set point as a single-precision float, in set point units.

`Argument` The test set point in set point units; −3.40282346639e+38 (hexadecimal `ff7fffff`) queries.

Response: `SUCCESS` with the test set point in the return value; `INVALID_ARGUMENT` if it is outside the valid range; `UNSUPPORTED` if the device is not a controller.

Check set point response: collection time

10v13.0 Controllers

`Command ID` **64**

Reads or sets how long the check collects data after the set point change.

`Argument` The collection time in seconds, 1–65535; 0 queries.

Response: `SUCCESS` with the collection time in seconds in the return value; `INVALID_ARGUMENT` if it is outside 0–65535; `UNSUPPORTED` if the device is not a controller.

Check set point response: run status

10v13.0 Controllers

`Command ID` **66**

Reads the state of the check, and can start or abort it.

`Argument` A value from the first table below.

Value	Meaning
0	Query the state of the check; take no action.
1	Start a check: the device changes the set point (overriding an analogue set point), collects data for the set collection time, then returns to the normal set point.
2	Abort a running check; does nothing if none is running.

Value	Meaning
0	The last check completed successfully and the results are calculated.
1	No check has run since power-up.
2	A check could not start, because gain tuning or another check is active.
3	A check is running.
4	Data collection was aborted by the user; all returned values are zero.
5	Data collection stopped early because of oscillation; all returned values are zero.

Response: `SUCCESS` if the action was performed; the return value is the status of the running or last check (second table). `UNSUPPORTED` if the device is not a controller; `INVALID_ARGUMENT` if the value is not listed.

Check set point response: result notes

10v13.0 Controllers

`Command ID` **67**

Returns notes on unusual conditions during the last completed check that may affect the accuracy of the results. Several notes add up to one value.

`Argument` Always 0.

Value	Meaning
0	No notes: the process data showed typical patterns.
+1	The collection time was too short for reliable results; a longer time may change them.
+2	The process variable oscillated significantly; control was not stable and the results are not reliable.
+4	The process variable moved little compared with the noise, so the results may not be reliable. Most often the set point change was too small, or an oscillation was not detected.
+8	A closed valve opened during collection. The results are correct, but performance differs from moving an already open valve.
+16	A valve opened as far as it could. The results are correct, but performance is probably limited by the valve size, not the control settings.

Response: `SUCCESS` with the notes in the return value; `UNSUPPORTED` if the device is not a controller.

Check set point response: overshoot (counts)

10v13.0 Controllers

`Command ID` **68**

Returns the overshoot of the last completed check.

`Argument` Always 0.

Response: `SUCCESS` with the overshoot of the last completed check in set point counts in the return value; `UNSUPPORTED` if the device is not a controller.

Check set point response: overshoot (integer)

10v13.0 Controllers

`Command ID` **65567**

Returns the overshoot of the last completed check as an integer; scale it by $10^{-\text{set point decimal places}}$.

`Argument` Always 0.

Response: `SUCCESS` with the overshoot of the last completed check as an integer in the return value; `UNSUPPORTED` if the device is not a controller.

Check set point response: overshoot (float)

10v13.0 Controllers

`Command ID 65566`

Returns the overshoot of the last completed check as a single-precision float, in set point units.

`Argument` Always 0.

Response: `SUCCESS` with the overshoot of the last completed check as a float in the return value; `UNSUPPORTED` if the device is not a controller.

Check set point response: timing results

10v13.0 Controllers

`Command ID 69`

Returns one timing value of the last completed check.

`Argument` The timing value to read, from the table below; the return value is that value.

Value	Meaning
0	Time constant (T63) of the closed-loop response, in milliseconds; for a first-order-plus-dead-time (FOPDT) model.
1	Dead time (delay) of the system, in milliseconds; for a FOPDT model.
2	Rise time from 10 % to 90 % of the step, in milliseconds.
3	Estimated bandwidth, calculated from the rise time; multiply by 0.01 for hertz.

Response: `SUCCESS` with the requested value; `UNSUPPORTED` if the device is not a controller; `INVALID_ARGUMENT` if the value is not listed.

Control optimisation

Control optimisation drives the valves of a mass flow controller to find closed-loop gains for a chosen speed or response time.

Control optimisation maximum flow (counts)

10v13.0 Flow controllers

`Command ID 70`

Reads or sets the largest flow that optimisation will deliberately pass, in set point counts.

`Argument` 1–64000; 0 sets the limit to full-scale flow; 65535 queries.

Response: `SUCCESS` with the maximum flow in the return value; `INVALID_ARGUMENT` if the value is outside the range or below 20 % of full scale; `UNSUPPORTED` if the device is not a controller.

Control optimisation maximum flow (integer)

10v13.0 Flow controllers

`Command ID` **65569**

Reads or sets the largest flow that optimisation will deliberately pass, as an integer.

`Argument` The maximum flow in set point units multiplied by $10^{\text{set point decimal places}}$; 0 sets full-scale flow; a value below 0 queries.

Response: `SUCCESS` with the maximum flow in the return value; `INVALID_ARGUMENT` if the value is outside the range or below 20 % of full scale; `UNSUPPORTED` if the device is not a controller.

Control optimisation maximum flow (float)

10v13.0 Flow controllers

`Command ID` **65568**

Reads or sets the largest flow that optimisation will deliberately pass, as a float.

`Argument` The maximum flow in set point units, as a single-precision float; 0 sets full-scale flow; a value below 0 queries.

Response: `SUCCESS` with the maximum flow in the return value; `INVALID_ARGUMENT` if the value is outside the range or below 20 % of full scale; `UNSUPPORTED` if the device is not a controller.

Control optimisation speed mode

10v13.0 Flow controllers

`Command ID` **71**

Reads or sets the speed mode of control optimisation.

`Argument` A value from the table below.

Value	Meaning
0	Query the speed mode.
1	Most versatile use, slowest response.
2	Versatility before speed of response.
3	Speed of response before versatility.
4	Fastest response (smallest time constant); a little overshoot is allowed.

Response: `SUCCESS` with the current speed mode in the return value; `INVALID_ARGUMENT` if the mode is not listed; `UNSUPPORTED` if the device is not a controller.

Control optimisation goal time

10v13.0 Flow controllers

`Command ID` **72**

Reads or sets the response time constant that optimisation aims for. After optimisation the response is close to it for the tested set points, if the system allows. If it is shorter than the system can manage, optimisation finds the gains with the shortest time constant; if a gain reaches 1 and cannot get smaller, the result may be faster than requested.

`Argument` The time constant in milliseconds; 0 uses the fastest possible (as the fastest speed mode); 65535 queries.

Response: `SUCCESS` with the goal time in milliseconds in the return value (0 while a speed mode is active); `INVALID_ARGUMENT` if it is negative; `UNSUPPORTED` if the device is not a controller.

Control optimisation control algorithm

10v13.0 Flow controllers

`Command ID` **73**

Reads or sets the control algorithm for which optimisation determines the gains.

`Argument` A value from the table below.

Value	Meaning
0	Query only; do not change the configured algorithm.
1	Optimisation chooses the most suitable algorithm for the configuration and parameters; the choice may change between firmware versions.
2	Use the PDF algorithm. Speed modes are 2 to 3 times slower than with the other values.
3	Use the PD²I algorithm.

Response: `SUCCESS` with the current algorithm in the return value; `INVALID_ARGUMENT` if the value is not listed; `UNSUPPORTED` if the device is not a controller.

Control optimisation run state

10v13.0 Flow controllers

`Command ID` **74**

Reads the state of control optimisation, and can start, pause, resume or abort it.

`Argument` An action from the first table below.

Value	Meaning
0	Query the state only; take no action.
1	Abort any running optimisation.
2	Start a new optimisation with the current settings; a running one is aborted first.
3	As 2, but pause at every possible point and close the valves.
4	Pause at the next possible point and close the valves, for example to let a volume filled during optimisation bleed back to normal pressure. Does nothing if no optimisation is running.
5	Resume a paused optimisation.

Value	Meaning
0	Optimisation is not active.

Value	Meaning
1	Optimisation is driving the valves.
2	Optimisation is paused; the valves stay closed until it is resumed.
3	An abort was requested, but optimisation is still active.

Response: `SUCCESS` if the action was performed; the return value is the current activity (second table). `INVALID_ARGUMENT` if the action is not listed; `UNSUPPORTED` if the device is not a controller.

Control optimisation result

`10v13.0` Flow controllers

> `Command ID` **75**

Returns the result of the last completed optimisation.

`Argument` Always 0.

Value	Meaning
0	An optimisation is in progress.
1	The last optimisation completed successfully.
2	The last optimisation was aborted by the user or by an error.
3	An unexpected state occurred during the last optimisation.
4	The maximum flow allowed for optimisation is too small.
5	A valve selected for optimisation is not configured for closed-loop control.
6	No closed-loop control was configured at the time of the last optimisation.
7	Not enough flow to complete the last optimisation; check the inlet and outlet pressures.
8	The last optimisation found no gains with acceptable performance.

Response: `SUCCESS` with the result (table above) in the return value; `UNSUPPORTED` if the device is not a controller.

Administrative

Taring, totaliser reset, gas selection and gas mixes, display control, device information and factory settings.

Perform tare

`7v05.0` All devices

> `Command ID` **4**

Takes the current reading as zero. Gauge pressure tares need the sensor open to atmosphere; differential pressure tares need the same pressure on both sides; absolute pressure tares need a barometer. Flow tares need no flow through the device, best at or close to process pressure. See Taring (p. 50).

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` A value from the table below.

Value	Meaning
0	Tare gauge or differential pressure
1	Tare absolute pressure
2	Tare mass and/or volumetric flow

Response: `SUCCESS` if the tare is done; `UNSUPPORTED` if the tare does not apply (for example a mass flow tare on a pressure instrument).

Taring

A tare takes the present reading as zero. Prepare the device for the tare you choose:

- **Gauge pressure:** open the sensor to the atmosphere.
- **Differential pressure:** apply the same pressure to both sides.
- **Absolute pressure:** the device needs a barometer.
- **Flow:** make sure no flow passes through the device. The tare is best done at process pressure, or as close to it as possible.

If the tare does not apply to the device, for example a flow tare on a pressure instrument, the device answers with status `UNSUPPORTED`.

Perform pressure sensor tare

`10v07.0` All devices

> `Command ID` **31**

Takes the current gauge or differential pressure reading as zero; an absolute pressure tare uses the zero reading of the barometer. Gauge pressure tares need the sensor open to atmosphere; differential pressure tares need the same pressure on both sides; absolute pressure tares need a barometer.

`Argument` The tare time in milliseconds, 0–32767; 0 uses the default of 256 ms.

Response: `SUCCESS` if the tare is done; `INVALID_ARGUMENT` if the argument is below 0 or above 32767; `UNSUPPORTED` if the sensor cannot be tared, usually an absolute pressure tare without a barometer.

Perform secondary pressure sensor tare

10v07.0 · All devices

> Command ID **32**

Takes the current reading of the secondary pressure sensor as zero; an absolute pressure tare uses the zero reading of the barometer. Not every device has a secondary pressure sensor. Gauge pressure tares need the sensor open to atmosphere; differential pressure tares need the same pressure on both sides; absolute pressure tares need a barometer.

`Argument` The tare time in milliseconds, 0–32767; 0 uses the default of 256 ms.

Response: `SUCCESS` if the tare is done; `INVALID_ARGUMENT` if the argument is below 0 or above 32767; `UNSUPPORTED` if the sensor cannot be tared, usually an absolute pressure tare without a barometer, or no secondary sensor.

Perform flow tare

10v07.0 · Flow devices

> Command ID **33**

Takes the current flow reading as zero, for volumetric and mass flow.

`Argument` The tare time in milliseconds, 0–32767; 0 uses the default of 256 ms.

Response: `SUCCESS` if the tare is done; `INVALID_ARGUMENT` if the argument is below 0 or above 32767; `UNSUPPORTED` if the tare is not possible, usually because the device does not read volumetric or mass flow.

Reset totaliser

10v19.0 · Flow devices

> Command ID **80**

Resets the requested totaliser to zero.

`Argument` 1 for totaliser 1, 2 for totaliser 2.

Response: `SUCCESS` when the reset is complete; `INVALID_ARGUMENT` if the argument is not 1 or 2.

Reset totaliser 1

7v05.0 · Flow devices

> Command ID **5**

Resets totaliser 1 to zero.

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` None.

Response: `SUCCESS` when the reset is complete.

Set selected gas

`7v05.0`

Mass flow devices

`Command ID 1`

Changes the gas the device measures.

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` The gas number from Appendix D (p. 69).

Response: `SUCCESS` if the gas changes; `INVALID_ARGUMENT` if the gas number does not exist.

Create or update gas mix

`7v05.0`

Mass flow devices

`Command ID 2`

Creates or updates a gas mix from the constituents written to registers 1050–1059. See Gas mixing (p. 57).

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` 0 to use the first free gas number, counting down from 255; or a number 236–255 to create or overwrite that mix.

Response: `SUCCESS` with the gas number of the mix in the return value; `INVALID_MIX_IDX` if the number is outside 236–255; `INVALID_MIX_GAS` if a constituent does not exist; `INVALID_MIX_PCT` if the fractions do not add up to 100 %.

Set gas mix constituent gas number

`10v19.0`

Mass flow devices

`Command ID 52, 54, 56, 58, 60`

Sets the gas number of a gas mix constituent: 52 the first, 54 the second, 56 the third, 58 the fourth, 60 the fifth.

Also in 9v07.4.

`Argument` The gas number of the constituent.

Response: `SUCCESS` if the gas number is set; `INVALID_ARGUMENT` if the gas number does not exist.

Set gas mix constituent fraction

`10v19.0`

Mass flow devices

`Command ID 53, 55, 57, 59, 61`

Sets the fraction of a gas mix constituent: 53 the first, 55 the second, 57 the third, 59 the fourth, 61 the fifth.

Also in 9v07.4.

`Argument` The fraction, 0.01 % per count (10000 = 100 %).

Response: `SUCCESS` if the fraction is set; `INVALID_ARGUMENT` if it is above 10000 (100 %).

Delete gas mix

`7v05.0`

Mass flow devices

`Command ID 3`

Deletes a gas mix.

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` The gas number of the mix to delete.

Response: `SUCCESS` if the mix is deleted; `INVALID_MIX_IDX` if the mix does not exist.

Set relative humidity

`9v04.0`

Devices with humidity sensor

`Command ID 24`

Sets the relative humidity used for gas corrections. Firmware 9v04.0 on humidity builds only; all builds from 10v07.0.

`Argument` 0–10000, one count = 0.01 % humidity: 100 for 1 %, 10000 for 100 %.

Response: `SUCCESS` if the humidity is changed; `UNSUPPORTED` if it cannot be set on the device; `INVALID_ARGUMENT` if the value is outside 0–10000.

Set relative humidity reference temperature

`10v07.0`

Devices with humidity sensor

`Command ID 25`

Sets the reference temperature of the relative humidity used for gas corrections, in degrees Celsius. It should match the humidity percentage.

`Argument` −3000 to 10000, one count = 0.01 °C: −3000 is −30 °C, 10000 is 100 °C.

Response: `SUCCESS` if the temperature is changed; `UNSUPPORTED` if the humidity cannot be set on the device; `INVALID_ARGUMENT` if the value is outside −3000 to 10000.

Lock or unlock display

`7v05.0` All devices

`Command ID 7`

Locks or unlocks the display. A locked device still responds to key presses, but settings cannot be changed from the display.

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` 0 unlocks; any other value locks.

Response: `SUCCESS` if the display is locked or unlocked as requested.

Flash display

`8v28.0` All devices

`Command ID 20`

Flashes the backlight of the monochrome or TFT display; pressing any key stops it.

`Argument` 65535 flashes until stopped; 0 stops flashing; any other value flashes for that number of seconds.

Response: `SUCCESS` if the backlight flashes or stops as requested; `UNSUPPORTED` if no display is connected.

Read serial number

10v19.0 All devices

`Command ID 65570`

Returns the serial number of the device.

`Argument` Must be 0.

Response: `SUCCESS` with the serial number in the return value; `INVALID_ARGUMENT` if the argument is not 0.

Read partial serial number

10v19.0 All devices

`Command ID 81`

Returns part of the serial number, for protocols that support basic commands only.

`Argument` 0 for the lowest two bytes, 1 for the highest two bytes.

Response: `SUCCESS` with the requested part in the return value; `INVALID_ARGUMENT` if the argument is not 0 or 1.

Read manufacture date

10v19.0 All devices

`Command ID 82`

Returns part of the manufacture date.

`Argument` A value from the table below.

Value	Meaning
0	Year
1	Month
2	Day

Response: `SUCCESS` with the requested part in the return value; `INVALID_ARGUMENT` if the argument is not 0, 1 or 2.

Read calibration date

10v19.0 All devices

`Command ID 83`

Returns part of the calibration date.

`Argument` A value from the table of read manufacture date (p. 54).

Response: `SUCCESS` with the requested part in the return value; `INVALID_ARGUMENT` if the argument is not 0, 1 or 2.

Read firmware version

`10v19.0` All devices

`Command ID` **84**

Returns part of the firmware version.

`Argument` A value from the table below.

Value	Meaning
0	Major part
1	Minor part
2	Custom part
3	Internal part

Response: `SUCCESS` with the requested part in the return value; `INVALID_ARGUMENT` if the argument is not 0, 1, 2 or 3.

Read configuration checksum

`8v24.0` All devices

`Command ID` **17**

Calculates a checksum of the calibration and configuration of the device. The calculation can take 300 ms; allow a timeout of more than 500 ms. Every setting kept across power cycles counts, and settings that may or may not be kept. Because calibration data is included, two devices with identical configurations can have different checksums; recalibration can change it, and a firmware change may. Put settings that change in normal operation, such as the set point, into a known state before you read the checksum.

Also in 8v22.2.

`Argument` Always 0.

Response: `SUCCESS` with the checksum, 0–65535, in the return value; `INVALID_ARGUMENT` if the argument is not 0.

Restore factory settings

`10v07.0` All devices

`Command ID` **26**

Returns all settings and the configuration to their state when the device last left the factory. Use it only while troubleshooting with Analyt-MTC support. All third-party calibrations are removed. Power-cycle the device after the restore.

NOTICE

Restoring factory settings removes all third-party calibrations. Use this command only while troubleshooting with Analyt-MTC support.

`Argument` 49374, to confirm the restore.

Response: `SUCCESS` when the restore is complete, readable only briefly before the restore clears the status; `INVALID_ARGUMENT` if the argument is not 49374; `UNSUPPORTED` if an error occurs during the restore.

Communications

Set Modbus slave address

`7v05.0` Modbus RTU only

`Command ID` **32767**

Sets the Modbus slave address of the device. See Slave ID (p. 8).

Also in 6v17.1, 6v17.2, 6v17.3, 6v20.2, 6v20.3, 6v20.4, 6v20.5, 6v20.6.

`Argument` The new address, 1–247.

Response: `INVALID_ARGUMENT` if the address is out of range.

Set communications watchdog timeout

`8v24.0` Modbus RTU only

`Command ID` **32765**

Sets the timeout of the communications watchdog. When more than this time has passed since the last successful Modbus communication, the device runs its disconnection logic: `COM` flashes on the display, the set point goes to 0 (unless configured otherwise) and a stream-switching (multi-valve) controller selects its primary valve (unless configured otherwise). The next successful communication resets the watchdog and normal operation resumes. A message with a checksum error, or addressed to another device, does not count; a correctly formatted message with a broadcast address, an unsupported function code, an invalid register address or an invalid register value does. For long-running commands, the timeout starts when the response begins to be sent.

Also in 8v22.2.

`Argument` The timeout, 0–65536, in 0.1 s: 100 is 10.0 s. 0 disables the watchdog.

> **Note**
> When the watchdog expires, a controller moves its set point to 0 and a stream-switching (multi-valve) controller selects its primary valve, unless configured otherwise. Normal operation resumes with the next successful communication.

Response: `SUCCESS` if the timeout is updated.

Hardware

Set primary analogue output value

`8v24.0` All devices

`Command ID` **18**

Sets the value of the primary analogue output.

Also in 8v22.2.

`Argument` 0–65535: 0 is the minimum of the device (for example 0 V or 4 mA), 64000 is full scale (for example 5 V or 20 mA).

Response: `SUCCESS` if the output is updated; `UNSUPPORTED` if this output is not available or not configured for this use; `INVALID_ARGUMENT` if the value is out of range.

Set secondary analogue output value

8v24.0 All devices

`Command ID` **19**

Sets the value of the secondary analogue output.

Also in 8v22.2.

`Argument` 0–65535: 0 is the minimum of the device (for example 0 V or 4 mA), 64000 is full scale (for example 5 V or 20 mA).

Response: `SUCCESS` if the output is updated; `UNSUPPORTED` if this output is not available or not configured for this use; `INVALID_ARGUMENT` if the value is out of range.

Set point

The set point can be written in three places: as a float in registers 1010–1011 or 1350–1351, or as an integer in registers 1300–1301, scaled by the set point decimal places. Always write both registers of a pair in one Write Multiple Registers message. A device without a controller ignores the set point. The reply to the write repeats the value you sent, not the value the device stored, and the device adjusts a set point outside the allowed range: read the register to see the value in force. Registers 1010–1011 can be read from firmware 10v07.0; before that, read the set point reading of the device (Typical reading assignments (p. 20)). Command 12 stores the current set point as the value the device starts with at power-up.

Note
Never write just one half of the set point. A write that covers only one register of the pair causes an error.

Gas mixing

The device can mix two to five gases into a single new gas. Mixing takes two steps: first describe the mixture in the mix registers, then run the mix gas command.

Each of the five slots in registers 1050–1059 is a pair: the gas index in one register, its percentage in the next. A percentage is a whole number in hundredths of a percent, and the percentages must add up to 100 %. So 50 % is written as `5000`, and a mixture of 60 %, 30 % and 10 % as `6000`, `3000` and `1000`. The device uses the first gases with a non-zero percentage, so write zero to every unused pair.

1. Write the index and percentage of each constituent gas to the mix registers, starting at register 1050.
2. Write command ID 2 to register 1000; the argument in register 1001 selects the mix index (see below).
3. Read the result as described for a limited command: on Modbus RTU read register 1001 with function 04; on Modbus TCP/IP read registers 1000–1001 until register 1000 holds 2 (see Limited command, Modbus TCP/IP (p. 22)). Register 1001 holds the mix index, or an error value if no mix was created.

3-15 Argument of the mix gas command

Argument	Result
`0`, or omitted	A new mix index is allocated, starting at 255 and counting down to 236. If all indices are in use, the mix is not created and an error is returned.
`236`–`255`	The mixture with that index is created, or updated if it already exists.
Anything else	An error is returned.

The mix is rejected, with the status codes in Command status (p. 23), when a constituent gas does not exist or the percentages do not add up to 100 %. The device names the new gas `M` followed by its index, for example `M255`. Delete a mixture you no longer need with command 3.

> **Note**
> Mixtures cannot be created or deleted on the device display. Existing mixtures appear in the display's gas selection.

CHAPTER 4

Examples

Five common tasks, shown as the Modbus messages your master sends. Use them to check your own configuration.

How to read the examples

Each example shows the Modbus *protocol data unit*: the function code followed by its data. Your master adds the slave ID and CRC for Modbus RTU, or the TCP header for Modbus TCP/IP, so those are not shown. All values are hexadecimal. Addresses are zero-based, one lower than the register number (see Register numbers and addresses (p. 14)), so register 1000 appears as `03E7`.

Example 1: tare the flow reading

Command 4 with argument 2 zeroes the mass and volumetric flow readings. Only do this with no flow through the device (see Taring (p. 50)). The command goes to registers 1000 and 1001 in one write.

Request: function 16, write registers 1000–1001

Field	Bytes	Meaning
Function code	`10`	Write Multiple Registers
Start address	`03E7`	Register 1000
Quantity	`0002`	Two registers
Byte count	`04`	Four data bytes follow
Register 1000	`0004`	Command 4, tare
Register 1001	`0002`	Argument 2, flow

On Modbus RTU, the reply comes when the tare is finished. To check the result, read register 1001 with function 04: request `04 03E8 0001`. A response of `04 02 0000` means the status is 0, success.

On Modbus TCP/IP, read registers 1000–1001 with function 04 (request `04 03E7 0002`) until register 1000 holds `0004`; register 1001 then holds the status (see Limited command, Modbus TCP/IP (p. 22)).

Example 2: send a set point of 50.0

The float 50.0 is `0x42480000`. The upper half goes to register 1010 and the lower half to register 1011, in one write. The reply repeats the values you sent.

Request: function 16, write registers 1010–1011

Field	Bytes	Meaning
Function code	`10`	Write Multiple Registers
Start address	`03F1`	Register 1010
Quantity	`0002`	Two registers
Byte count	`04`	Four data bytes follow
Register 1010	`4248`	Upper half of 50.0
Register 1011	`0000`	Lower half of 50.0

Example 3: read the mass flow of a controller

With the default statistics of a mass flow controller (Typical reading assignments (p. 20)), mass flow is in registers 1209–1210.

Request: function 04, read registers 1209–1210

Field	Bytes	Meaning
Function code	`04`	Read Input Registers
Start address	`04B8`	Register 1209
Quantity	`0002`	Two registers

The response `04 04 42C8 0000` contains four data bytes. The two registers `42C8` and `0000` make the float `0x42C80000`, which is **100.0** in the unit your device is configured for.

To read the gas number and the device status in one request, ask for three registers from register 1200: `04 04AF 0003`. A response of `04 06 0008 0000 0100` means gas number 8 and status `0x00000100`. Bit 8 is set, so the valve drive is on hold; see Device status (p. 18).

Example 4: create a gas mixture

This example mixes three gases at 60 %, 30 % and 10 %. The gas indices 5, 8 and 11 are placeholders; use the indices from the gas table in your device's operating manual.

1. Write the mix registers 1050–1059 in one request. The unused pairs, registers 1056–1059, are zero.
2. Run the mix gas command with argument 0, so that the device picks a free mix index.
3. Read register 1001 to learn the mix index.

Step 1: function 16, write registers 1050–1059

Register	Bytes	Meaning
`10 0419 000A 14` (function, start address 1050, ten registers, twenty data bytes)		
1050	`0005`	Gas 1: index 5
1051	`1770`	60.00 % (6000)
1052	`0008`	Gas 2: index 8
1053	`0BB8`	30.00 % (3000)
1054	`000B`	Gas 3: index 11
1055	`03E8`	10.00 % (1000)
1056–1059	`0000` × 4	Gases 4 and 5 not used

Step 2 is function 16 writing registers 1000–1001 with `10 03E7 0002 04 0002 0000`: command 2, argument 0. In step 3, a response of `04 02 00FF` shows that the mixture was stored as index 255, the first index to be allocated. On Modbus TCP/IP, read registers 1000–1001 (request `04 03E7 0002`) until register 1000 holds `0002`; register 1001 then holds the mix index (see Limited command, Modbus TCP/IP (p. 22)). The mixture now appears in the gas selection on the display.

Example 5: change the slave ID (Modbus RTU)

Command 32767 (`0x7FFF`) changes the slave ID. To give a device the new ID 12, write registers 1000–1001 with `10 03E7 0002 04 7FFF 000C`. Use the new ID in your next request, and change devices

one at a time so that two devices never share an ID on the line.

CHAPTER 5

Error handling

Common symptoms, their likely causes and what to do. If none of these fits, contact Analyt-MTC support with the information listed in About this guide (p. 2).

Symptoms and remedies

Symptom	Cause and remedy
CONNECTION	
No response on a device with an RJ45 connector	On a Modbus RTU device the RJ45 is a serial connector, not Ethernet. Connect it to a serial port, not to a network switch.
Modbus TCP/IP device cannot be reached by browser or master	The device takes its address from DHCP until you set a static one. Find the address its DHCP server gave it, or follow the steps in Setting the IP address (p. 10). On a device with a display, check the *Network Status* screen. After an address change, wait about 15 seconds.
No response on a serial line, or unreadable responses	Check that baud rate, data bits, parity and stop bits match on the master and the device (Modbus RTU: serial connection (p. 6)).
Garbled or mixed-up responses on a serial line	Two devices share the same slave ID. Disconnect all but one and change its ID with command 32767 (p. 23).
VALUES	
Every value comes from the neighbouring register	The master uses zero-based addresses. Subtract one from each register number in this guide (Register numbers and addresses (p. 14)).
A reading is nonsense although the register is right	The two 16-bit halves of the 32-bit value are in the wrong order. Swap the words in the master (Reading 32-bit values (p. 14)).
A statistic reads `0xFFFFFFFF`	That statistic slot is unused on this device (Modbus RTU). Check the configured statistics (Typical reading assignments (p. 20)).
Invalid register address error on a Modbus TCP/IP device	Most often, you read an unused statistic slot. Check the configured statistics in the web server's *Data I/O* tab. Reading any register that does not exist may also return 65535 or an illegal data address error.
Legacy registers (2041 onward) do not respond	Legacy registers exist on Modbus RTU only. Use the registers in Chapter 3 (p. 16) on a Modbus TCP/IP device.
COMMANDS	
Set point write gives an error	Both registers 1010 and 1011 must be written in a single Write Multiple Registers message. See Set point (p. 57).
The reply to a set point write shows a value you did not expect	The reply repeats what you sent, not what the device stored, and the device adjusts a set point outside the allowed range. Read the set point statistic of the device to see the value in force.
Set point has no effect	The device is a meter or gauge without a controller, which ignores set points.
A repeated command does nothing	The device runs a command only when the ID or the argument changes. Send command 0 between two identical commands (Sending a command (p. 22)).
Command status `INVALID_ID` (32769)	The command ID is not valid. Check register 1000 against the command list (p. 23).
Command status `INVALID_ARGUMENT` (32770)	The argument is not allowed for this command. Check register 1001.
Command status `UNSUPPORTED` (32771)	The device does not support the requested feature. Exhaust works only on dual-valve devices, display lock only on devices with a display, and a tare only if it applies to the device (for example, no flow tare on a pressure instrument).
Command status `INVALID_MIX_IDX`, `INVALID_MIX_GAS` or `INVALID_MIX_PCT` after a mix	The mix index must be 0 or 236–255 (`INVALID_MIX_IDX`, 32772), every gas index must exist (`INVALID_MIX_GAS`, 32773) and the percentages must add up to 100.00 %, that is 10000 (`INVALID_MIX_PCT`, 32774). See Gas mixing (p. 57).

Symptom	Cause and remedy
Mix gas fails although the data is correct	All indices from 236 to 255 are already in use. Delete a mixture you no longer need with command 3, or use an index you want to overwrite.

APPENDIX A

Statistics

Statistics are the readings and measurements a device provides. Their values are returned by the type registers of Reading information (p. 19) and by query reading type (p. 26), and are used by loop control variable (p. 38). Some statistics exist only on certain devices and configurations; ask Analyt-MTC about a statistic and how it works with your device.

A-1 Flow statistics

Flow statistic	Value	Notes
Mass flow	**5**	Current mass flow
Mass flow set point	**37**	Set point for mass flow
Total mass	**9**	Totalised mass
Total volume	**8**	Totalised volume, referenced to flow conditions
Volumetric flow	**4**	Volumetric flow, referenced to flow conditions
Volumetric flow set point	**36**	Set point for volumetric flow, referenced to flow conditions

A-2 Pressure statistics

Pressure statistic	Value	Notes
Pressure, absolute	**2**	Current absolute pressure
Pressure, absolute set point	**34**	Set point for absolute pressure
Pressure, barometric	**15**	Barometer reading
Pressure, differential	**7**	Current differential pressure
Pressure, differential set point	**39**	Set point for differential pressure
Pressure, gauge	**6**	Current gauge pressure
Pressure, gauge set point	**38**	Set point for gauge pressure
Pressure, second absolute	**344**	Absolute pressure of the second sensor, on devices with a second pressure sensor
Pressure, second absolute set point	**345**	Set point for the second absolute pressure
Pressure, second differential	**360**	Differential pressure of the second sensor, on devices with a second pressure sensor
Pressure, second differential set point	**361**	Set point for the second differential pressure
Pressure, second gauge	**352**	Gauge pressure of the second sensor, on devices with a pressure sensor in a second location
Pressure, second gauge set point	**353**	Set point for the second gauge pressure

A-3 Other statistics

Other statistic	Value	Notes
None	**1**	No statistic; usually an empty slot
Relative humidity	**25**	The fraction of complete saturation the device is using; with water vapour, this is the relative humidity
Temperature, stream	**3**	Current stream temperature
Valve drive	**13**	Valve drive signal
Valve drive set point	**45**	The set point drives the selected valve directly

APPENDIX B

Engineering units

Commands and registers that take or return a unit use the values below. Which table applies depends on the statistic: for example, use table B-3 for a totaliser that counts standard or normal volume. To change the units of a reading, use set reading engineering units (p. 28).

B-1 Standard and normal flow units

Value	Label	Description
0		Unit not specified; use default values.
1	`---`	Unknown unit; no conversions are performed to other units. If calibrated with different units, the value is displayed in those units.
2	`SµL/m`	Standard microlitre per minute
3	`SmL/s`	Standard millilitre per second
4	`SmL/m`	Standard millilitre per minute
5	`SmL/h`	Standard millilitre per hour
6	`SL/s`	Standard litre per second
7	`SLPM`	Standard litre per minute
8	`SL/h`	Standard litre per hour
11	`SCCS`	Standard cubic centimetre per second
12	`SCCM`	Standard cubic centimetre per minute
13	Scm^3/h	Standard cubic centimetre per hour
14	Sm^3/m	Standard cubic metre per minute
15	Sm^3/h	Standard cubic metre per hour
16	Sm^3/d	Standard cubic metre per day
17	Sin^3/m	Standard cubic inch per minute
18	`SCFM`	Standard cubic foot per minute
19	`SCFH`	Standard cubic foot per hour
21	`SCFD`	Standard cubic foot per day
20	`kSCFM`	1000 standard cubic feet per minute
32	`NµL/m`	Normal microlitre per minute
33	`NmL/s`	Normal millilitre per second
34	`NmL/m`	Normal millilitre per minute
35	`NmL/h`	Normal millilitre per hour
36	`NL/s`	Normal litre per second
37	`NLPM`	Normal litre per minute
38	`NL/h`	Normal litre per hour
41	`NCCS`	Normal cubic centimetre per second
42	`NCCM`	Normal cubic centimetre per minute
43	Ncm^3/h	Normal cubic centimetre per hour
44	Nm^3/m	Normal cubic metre per minute
45	Nm^3/h	Normal cubic metre per hour
46	Nm^3/d	Normal cubic metre per day
62	`Count`	Set point count, 0–64000
63	`%`	Percent of the full scale

B-2 True mass flow units

Value	Label	Description
64	mg/s	Milligram per second
65	mg/m	Milligram per minute
66	g/s	Gram per second
67	g/m	Gram per minute
68	g/h	Gram per hour
69	kg/m	Kilogram per minute
70	kg/h	Kilogram per hour
71	oz/s	Ounce per second
72	oz/m	Ounce per minute
73	lb/m	Pound per minute
74	lb/h	Pound per hour

B-3 Total standard and normal volume units

Value	Label	Description
0		Unit not specified; use default values.
1	---	Unknown unit; no conversions are performed to other units. If calibrated with different units, the value is displayed in those units.
2	SµL	Standard microlitre
3	SmL	Standard millilitre
4	SL	Standard litre
6	Scm^3	Standard cubic centimetre
7	Sm^3	Standard cubic metre
8	Sin^3	Standard cubic inch

Value	Label	Description
9	Sft^3	Standard cubic foot
10	$kSft^3$	1000 standard cubic feet
32	NµL	Normal microlitre
33	NmL	Normal millilitre
34	NL	Normal litre
36	Ncm^3	Normal cubic centimetre
37	Nm^3	Normal cubic metre

B-4 Volumetric flow units

Value	Label	Description
0		Unit not specified; use default values.
1	`---`	Unknown unit; no conversions are performed to other units. If calibrated with different units, the value is displayed in those units.
2	`µL/m`	Microlitre per minute
3	`mL/s`	Millilitre per second
4	`mL/m`	Millilitre per minute
5	`mL/h`	Millilitre per hour
6	`L/s`	Litre per second
7	`LPM`	Litre per minute
8	`L/h`	Litre per hour
9	`US GPM`	US gallon per minute
10	`US GPH`	US gallon per hour
11	`CCS`	Cubic centimetre per second
12	`CCM`	Cubic centimetre per minute
13	cm^3/h	Cubic centimetre per hour
14	m^3/m	Cubic metre per minute
15	m^3/h	Cubic metre per hour
16	m^3/d	Cubic metre per day
17	in^3/m	Cubic inch per minute
18	`CFM`	Cubic foot per minute
19	`CFH`	Cubic foot per hour
21	`CFD`	Cubic foot per day
62	`count`	Set point count, 0–64000
63	`%`	Percent of full scale

B-5 Total volume units

Value	Label	Description
0		Unit not specified; use default values.
1	---	Unknown unit; no conversions are performed to other units. If calibrated with different units, the value is displayed in those units.
2	µL	Microlitre
3	mL	Millilitre
4	L	Litre
5	US GAL	US gallon
6	cm^3	Cubic centimetre
7	m^3	Cubic metre
8	in^3	Cubic inch
9	ft^3	Cubic foot
61	µP	Micropoise, a measure of viscosity; no conversions are performed to or from other units

B-6 Pressure units

Value	Label	Description
0		Unit not specified; use default values.
1	---	Unknown unit; no conversions are performed to other units. If calibrated with different units, the value is displayed in those units.
2	Pa	Pascal
3	hPa	Hectopascal
4	kPa	Kilopascal
5	MPa	Megapascal
6	mbar	Millibar
7	bar	Bar
8	g/cm^2	Gram force per square centimetre
9	kg/cm^2	Kilogram-force per square centimetre
10	PSI	Pound-force per square inch
11	PSF	Pound-force per square foot
12	mTorr	Millitorr
13	torr	Torr
14	mmHg	Millimetre of mercury at 0 °C
15	inHg	Inches of mercury at 0 °C
16	mmH_2O	Millimetre of water at 4 °C (NIST conventional)
17	mmH_2O	Millimetre of water at 60 °F
18	cmH_2O	Centimetre of water at 4 °C (NIST conventional)
19	cmH_2O	Centimetre of water at 60 °F
20	inH_2O	Inch of water at 4 °C (NIST conventional)
21	inH_2O	Inch of water at 60 °F
22	atm	Atmosphere (absolute pressure only)

Value	Label	Description
61	V	Volt; no conversions are performed to or from other units (intended only for log-linear absolute pressure sensors)
62	count	Set point count, 0–64000
63	%	Percent of full scale

B-7 Temperature units

Value	Label	Description
0		Unit not specified; use default values.
1	---	Unknown unit; no conversions are performed to other units. If calibrated with different units, the value is displayed in those units.
2	°C	Degree Celsius
3	°F	Degree Fahrenheit
4	°K	Kelvin
5	°Ra	Degree Rankine

APPENDIX C

Data sources

The source registers of Reading information (p. 19) and query reading source (p. 27) return one of these values.

C-1 Data sources

Data source	Value	Notes
Instant display	**1**	Current readings, smoothed for the display
Instant serial	**2**	Current readings, smoothed for serial values
Totaliser 2	**25**	Readings taken from the second totaliser

APPENDIX D

Gas numbers

Gas numbers are used by set selected gas (p. 52), the gas mix registers 1050–1059 and the gas mix commands, and are read in the gas number registers (1200, 1247, 1297, 1347). Numbers 236 to 255 are reserved for your own gas mixes (p. 57).

#	Short name	Long name
0	Air	Air (Clean Dry)
1	Ar	Argon
2	CH_4	Methane
3	CO	Carbon Monoxide
4	CO2	Carbon Dioxide
5	C_2H_6	Ethane
6	H_2	Hydrogen
7	He	Helium
8	N_2	Nitrogen
9	N_2O	Nitrous Oxide
10	Ne	Neon
11	O_2	Oxygen
12	C_3H_8	Propane
13	nC_4H_{10}	Normal Butane
14	C_2H_2	Acetylene
15	C_2H_4	Ethylene (Ethene)
16	iC_4H_{10}	Isobutane
17	Kr	Krypton
18	Xe	Xenon
19	SF_6	Sulfur Hexafluoride[1]
20	C-25	25 % CO_2, 75 % Ar
21	C-10	10 % CO_2, 90 % Ar
22	C-8	8 % CO_2, 92 % Ar
23	C-2	2 % CO_2, 98 % Ar
24	C-75	75 % CO_2, 25 % Ar
25	He-25	25 % He, 75 % Ar
26	He-75	75 % He, 25 % Ar
27	A1025	90 % He, 7.5 % Ar, 2.5 % CO_2
28	Star29	Stargon CS (90 % Ar, 8 % CO_2, 2 % O_2)
29	P-5	5 % CH_4, 95 % Ar
30	NO	Nitric Oxide[2]
31	NF_3	Nitrogen Trifluoride[2]
32	NH_3	Ammonia[2]
33	Cl_2	Chlorine[2]
34	H_2S	Hydrogen Sulfide[2]
35	SO_2	Sulfur Dioxide[2]
36	C_3H_6	Propylene[2]
80	1Buten	1-Butylene[2]
81	cButen	Cis-Butene (cis-2-Butene)[2]
82	iButen	Isobutene[2]
83	tButen	Trans-2-Butene[2]
84	COS	Carbonyl Sulfide[2]
85	DME	Dimethylether (C_2H_6O)[2]
86	SiH_4	Silane[2]
100	R-11	Trichlorofluoro-methane (CCl_3F)[2,3]
101	R-115	Chloropentafluoro-ethane (C_2ClF_5)[2,3]
102	R-116	Hexafluoroethane (C_2F_6)[2]
103	R-124	Chlorotetrafluoro-ethane (C_2HClF_4)[2,3]
104	R-125	Pentafluoroethane (CF_3CHF_2)[2,3]
105	R-134A	Tetrafluoroethane (CH_2FCF_3)[2,3]
106	R-14	Tetrafluoromethane (CF_4)[2]
107	R-142b	Chlorodifluoroethane (CH_3CClF_2)[2,3]
108	R-143a	Trifluoroethane ($C_2H_3F_3$)[2,3]
109	R-152a	Difluoroethane ($C_2H_4F_2$)[2]
110	R-22	Difluoromonochloro-methane ($CHClF_2$)[2,3]
111	R-23	Trifluoromethane (CHF_3)[2,3]
112	R-32	Difluoromethane (CH_2F_2)[2,3]
113	R-318	Octafluorocyclobutane (C_4F_8)[2]
114	R-404A	44 % R-125, 4 % R-134A, 52 % R-143A[2,3]
115	R-407C	23 % R-32, 25 % R-125, 52 % R-134A[2,3]
116	R-410A	50 % R-32, 50 % R-125[2,3]
117	R-507A	50 % R-125, 50 % R-143A[2,3]
140	C-15	15 % CO_2, 85 % Ar
141	C-20	20 % CO_2, 80 % Ar
142	C-50	50 % CO_2, 50 % Ar
143	He-50	50 % He, 50 % Ar
144	He-90	90 % He, 10 % Ar
145	Bio5M	5 % CH_4, 95 % CO_2
146	Bio10M	10 % CH_4, 90 % CO_2
147	Bio15M	15 % CH_4, 85 % CO_2
148	Bio20M	20 % CH_4, 80 % CO_2
149	Bio25M	25 % CH_4, 75 % CO_2
150	Bio30M	30 % CH_4, 70 % CO_2
151	Bio35M	35 % CH_4, 65 % CO_2
152	Bio40M	40 % CH_4, 60 % CO_2
153	Bio45M	45 % CH_4, 55 % CO_2
154	Bio50M	50 % CH_4, 50 % CO_2
155	Bio55M	55 % CH_4, 45 % CO_2
156	Bio60M	60 % CH_4, 40 % CO_2
157	Bio65M	65 % CH_4, 35 % CO_2
158	Bio70M	70 % CH_4, 30 % CO_2
159	Bio75M	75 % CH_4, 25 % CO_2
160	Bio80M	80 % CH_4, 20 % CO_2
161	Bio85M	85 % CH_4, 15 % CO_2
162	Bio90M	90 % CH_4, 10 % CO_2
163	Bio95M	95 % CH_4, 5 % CO_2
164	EAN-32	32 % O_2, 68 % N_2
165	EAN-36	36 % O_2, 64 % N_2
166	EAN-40	40 % O_2, 60 % N_2
167	HeOx20	20 % O_2, 80 % He
168	HeOx21	21 % O_2, 79 % He
169	HeOx30	30 % O_2, 70 % He
170	HeOx40	40 % O_2, 60 % He
171	HeOx50	50 % O_2, 50 % He
172	HeOx60	60 % O_2, 40 % He
173	HeOx80	80 % O_2, 20 % He

#	Short name	Long name
174	HeOx99	99 % O_2, 1 % He
175	EA-40	Enriched air, 40 % O_2
176	EA-60	Enriched air, 60 % O_2
177	EA-80	Enriched air, 80 % O_2
178	Metab	Metabolic Exhalant (16 % O_2, 78.04 % N_2, 5 % CO_2, 0.96 % Ar)
179	LG-4.5	4.5 % CO_2, 13.5 % N_2, 82 % He
180	LG-6	6 % CO_2, 14 % N_2, 80 % He
181	LG-7	7 % CO_2, 14 % N_2, 79 % He
182	LG-9	9 % CO_2, 15 % N_2, 76 % He
183	HeNe-9	9 % Ne, 91 % He
184	LG-9.4	9.4 % CO_2, 19.25 % N_2, 71.35 % He
185	SynG-1	40 % H_2, 29 % CO, 20 % CO_2, 11 % CH_4
186	SynG-2	64 % H_2, 28 % CO, 1 % CO_2, 7 % CH_4
187	SynG-3	70 % H_2, 4 % CO, 25 % CO_2, 1 % CH_4
188	SynG-4	83 % H_2, 14 % CO, 3 % CH_4
189	NatG-1	93 % CH_4, 3 % C_2H_6, 1 % C_3H_8, 2 % N_2, 1 % CO_2
190	NatG-2	95 % CH_4, 3 % C_2H_6, 1 % N_2, 1 % CO_2
191	NatG-3	95.2 % CH_4, 2.5 % C_2H_6, 0.2 % C_3H_8, 0.1 % C_4H_{10}, 1.3 % N_2, 0.7 % CO_2
192	CoalG	50 % H_2, 35 % CH_4, 10 % CO, 5 % C_2H_4
193	Endo	75 % H_2, 25 % N_2
194	HHO	66.67 % H_2, 33.33 % O_2
195	HD-5	LPG: 96.1 % C_3H_8, 1.5 % C_2H_6, 0.4 % C_3H_6, 1.9 % n-C_4H_{10}
196	HD-10	LPG: 85 % C_3H_8, 10 % C_3H_6, 5 % n-C_4H_{10}
197	OCG-89	89 % O_2, 7 % N_2, 4 % Ar
198	OCG-93	93 % O_2, 3 % N_2, 4 % Ar
199	OCG-95	95 % O_2, 1 % N_2, 4 % Ar
200	FG-1	2.5 % O_2, 10.8 % CO_2, 85.7 % N_2, 1 % Ar
201	FG-2	2.9 % O_2, 14 % CO_2, 82.1 % N_2, 1 % Ar
202	FG-3	3.7 % O_2, 15 % CO_2, 80.3 % N_2, 1 % Ar
203	FG-4	7 % O_2, 12 % CO_2, 80 % N_2, 1 % Ar
204	FG-5	10 % O_2, 9.5 % CO_2, 79.5 % N_2, 1 % Ar
205	FG-6	13 % O_2, 7 % CO_2, 79 % N_2, 1 % Ar
206	P-10	10 % CH_4, 90 % Ar
210	D-2	Deuterium

[1] Sulfur hexafluoride is a highly potent greenhouse gas monitored under the Kyoto Protocol.

[2] Corrosion-resistant devices only.

[3] Under the Montreal Protocol and its Kigali Amendment, the production and consumption of these ozone-depleting substances is being, or has been, phased out. Make sure you comply with these regulations before using these gases; the same applies to R-113, R-123 and R-141b.

ANALYT-MTC Messtechnik GmbH

Sales, service and calibration

German Headquarters
Klosterrunsstraße 18, 79379 Müllheim
Germany
+49 7631 5545
info@analyt-mtc.de

French Subsidiary
175 Avenue d'Alsace, 68000 Colmar
France
+33 3 89 41 47 78
info@analyt-mtc.fr

www.analyt-mtc.com

AMTC-CG-0001 · Rev. 2 · 2026-10

© 2026 ANALYT-MTC Messtechnik GmbH. All rights reserved. Subject to technical changes without notice.