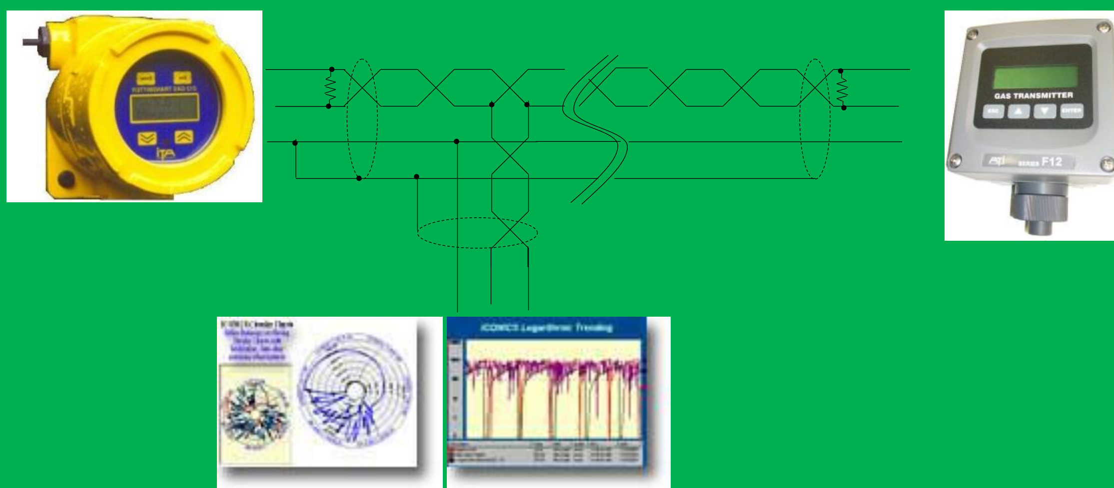
O&M Manual



D12/F12 Series
Modbus Interface

Home Office

Analytical Technology, Inc.
6 Iron Bridge Drive
Collegeville, PA 19426
Phone: 800-959-0299
610-917-0991
Fax: 610-917-0992
Email: sales@analyticaltechnology.com
Web: www.Analyticaltechnology.com

European Office

ATI (UK) Limited
Unit 1 & 2 Gatehead Business Park
Delph New Road, Delph
Saddleworth OL3 5DE
Phone: +44 (0)1457-873-318
Fax: + 44 (0)1457-874-468
Email: sales@atiuk.com

Table of Contents

1 Introduction

This manual describes the Modbus interface as implemented on D12 and F12 transmitters, equipped with the optional Modbus interface. The implementation of the interface is essentially identical on both transmitters. Throughout this manual, “D/F12” and “transmitter” are assumed to refer to either version, unless specifically noted.

The manual includes limited protocol information in Appendix B. Modbus Technical Overview, implementation, transmitter registers and subroutines, and RS485 networking practices and is intended for operators and system integrators who are familiar with RS232 point-to-point communications and RS485 networks. The illustrations and guidelines for setting up a communication network assume that an experienced installer will be capable of adapting them to a particular installation. ATI assumes no responsibility for any errors that may occur as a result of improper network installation.

The transmitter implements Modbus RTU protocol as defined in the “Modbus Protocol Specification”, available for download at http://www.modbus-ida.org/specs.php. Deviations from this guide are noted in the appropriate section. More information regarding Modbus, in general, may be viewed at: http://www.modbus-ida.org/

Modbus is a registered trademark of Modicon, Inc. Microsoft, Excel, Windows, and MS-DOS are a registered trademark of Microsoft Corporation. Citect is registered trademark of Ci Technologies Pty. Limited.

1.1 D12 Transmitter

The D12 Transmitter is used to monitor for gas leaks near storage cylinders, process piping, or gas feed equipment in virtually any type of industrial plant environment. The transmitter housing is explosion proof, and is rated for use in hazardous locations. The transmitter may be ordered with any of the sensor options listed in Table 1 and features, a non-intrusive four button user interface with a back-lighted graphics display, three level alarm system with three (optional) alarm relays, high-resolution 4-20mA current loop output, real-time clock, data-logger, and optional HART™ or Modbus™ communication interface.

Table 1 Sensor Options

H10 Gas Modules (D12 and F12)	Nearly 50 types available for detecting toxic gas mixtures, including Cl2, CO, H2S, NH3, and many more. Several H10 gas generators available for automatic timed testing of H10 sensors.
Infrared Gas Sensor (D12 Only)	Five types available for detecting toxic and explosive gas mixtures: HC (for LEL Methane, Propane, Butane, etc.), High HC (for Propane, Butane, etc.), CO2, High CO2, and N2O.
Catalytic Bead Sensor (D12 Only)	Used to detect explosive mixtures of Methane, Propane, Butane, etc.

1.2 F12 Transmitter

The F12 Gas Transmitter is used to monitor for gas leaks near storage cylinders, process piping, or gas feed equipment in many types of industrial environments. It is housed in NEMA 4X, polycarbonate enclosure (not suitable for explosive environments) and features an H10 sensor module, a non-intrusive four button user interface with a transflective graphics display, three level alarms with three (optional) alarm relays, a high-resolution 4-20mA current loop output, real-time clock, data-logger, and optional HART™ or Modbus™ communication interface. In addition, the transmitter offers several optional C18 gas generators for automatic, timed testing of H10 sensors.

2 Operation

The D/F12 transmitter implements the RTU version of the protocol, over an RS232 or RS485 connection, at baud rates up to 9600k. It is always a slave device and requires a Modbus master to poll its data.

The transmitter computes gas concentration and temperature every 200ms (5 times per second) and stores it in memory registers accessible via the Modbus interface. Other registers are used to read and write configuration settings.

The data and configuration memory of the transmitter has been enumerated as several hundred 16-bit registers; however, the transmitter does not permit writing directly to all of these registers. Instead, a subroutine selection register along with several data registers have been assigned for updating configuration registers and performing services, such as sensor calibration. The subroutines take care of such things as data range and status checks, calculating checksums, and updating error codes, and programming the EEPROM. A complete listing of registers and subroutines are included.

Modbus functions are encoded into the query message frame and designate specific operations such as read coil(s), write coil(s), read register(s), write register(s), read status, etc. Slave devices may not implement all of these functions, depending on their level of compliance. The transmitter most closely resembles the Modicon 884 PLC in that it supports the same functions (1-7, 15-17, but not 18) and can support message frames up to 256 bytes in length (no more than125 registers, no more than 2000 coils). It should be noted, however, that while the transmitter supports these functions it does not have coils and does not store data in the same registers as the 884. For example, function 1 – "Read Coil Status", and function 2 – "Read Input Status", always return 0s as the coil and input values.

In this document, the transmitter registers are referenced as though they reside in a slave Modicon PLC, that is, they are referred to as Holding Registers and are numbered from 40001 to 40999. These registers are accessed using functions 3-Read Holding Registers, 6-Preset Single Register, and 16-Preset Multiple Registers. Many high-level masters understand this relationship and allow you to build applications that accept register numbers, and even allow you to construct tag names. Other masters may require you to work closer to the protocol level and to supply addresses instead of register numbers. In this case, you would form an address by subtracting 40001 from the register number:

ADDRESS = REGISTER NUMBER - 40001

For example, the address of the main concentration reading in register 40043 would be 40043-40001=42 (2AH): Address Hi = 00H, Address Lo = 2AH

Modbus Functions Supported

The D/F12 transmitter attempts to implement protocol functions common to all Modicon PLCs. Since it is not a PLC, certain functions are implemented on a limited basis. In most cases, these limitations should not be a problem since the master is programmed to read and write specific data from each slave device. For example, there are no published discrete input (or coil) numbers; the transmitter and a master should never issue command 5 or 15 to force them on or off. Some master application programs may query outputs to coils or input status at startup as a means of detecting the presence of a slave device. In this case, the transmitter responds successfully to these queries without generating an exception, however, the returned status is always 0 (OFF). The table below summarizes the level of compatibility.

Function	Description	Y/N	D/F12 Transmitter Implementation
01	Read Coil Status	Y	Limited - no coil addresses defined, transmitter responds with 0 for every coil/bit requested.
02	Read Input Status	Y	Limited - no discrete inputs defined, transmitter responds with 0 for every coil/bit requested.
03	Read Holding Registers	Y	Full - valid register addresses are listed at the end of this document.
04	Read Input Registers	Y	Limited - no input registers defined, transmitter responds with 0 data for every register requested.
05	Force Single Coil	Y	Limited – no coil addresses defined, transmitter responds with exception code 02 – "Illegal Data Address"
06	Preset Single Register	Y	Full – valid register addresses are listed at the end of this document.
07	Read Exception Status	Y	Full - exception status codes are listed at the end of this document.
08	Diagnostics	Y	Limited – supported sub-functions listed at the end of this document.
09	Program 484	N	
10	Poll 484	N	
11	Fetch Comm Event Ctr.	N	
12	Fetch Comm Event Log	N	
13	Program Controller	N	
14	Poll Controller	N	
15	Force Multiple Coils	Y	Limited – no coil addresses defined, transmitter responds with exception code 02 – "Illegal Data Address"
16	Preset Multiple Registers	Y	Full – valid register addresses are listed at the end of this document.
17	Report Slave ID	Y	Full – slave id is 1100 (=ASC('D')*16+12)
18	Program 884/M84	N	
19	Reset Comm Link	N	
20	Read General Reference	N	
21	Write General Reference	N	
22	Mask Write 4X Register	N	
23	Read/Write 4X Register	N	
24	Read FIFO Queue	N	

2.1 Data Types

The transmitter organizes data into 16-bit holding registers, which are suitable for directly representing "integer" and "unsigned integer" data. The registers are also suitable for representing two 8-bit types, such as "byte" and "char". At least two registers are required for representing larger data types, such as "long" and "real".

Type	Bits	Range	Registers
Byte, Char	8	0 to +255	½
Integer	16	-32768 to +32767	1
Unsigned integer	16	0 to +65535	1
Long	32	-2,147,483,648 to +2,147,483,647	2
Unsigned Long	32	0 to +4,294,967,295	2
Real (IEEE 754)	32	+/-1.175e-38 to 3.40e+38	2

2.2 Endian

The term, "endian", is used to describe the byte order of a data type in memory. "Big endian" stores the most significant byte (MSB) of data in the lower memory address, followed by the less significant bytes in successively higher memory locations. "Little endian" stores the least significant byte (LSB) in low memory, and the more significant bytes in successively higher memory.

Example:
Memory Address=6048, Value=76543210 (hexadecimal) (=1,985,229,328 decimal)

Big Endian	Little Endian
6048: 76	6048: 10
6049: 54	6049: 32
6050: 32	6050: 54
6051: 10	6051: 76

Since endian can vary from slave to slave, the master must be able to arrange the data bytes before they are written, and after read back. Fortunately, many masters are capable of being configured to perform this action.

The D/F12 transmitter stores data in the little endian format, but then reverses register byte pairs during read/write operations - so that register data complies with the protocol.

2.2.1 Byte and Char

Byte and char data is stored in either the upper or lower 8-bits of a register, as noted in the register tables. When it is in the MSB, its value may be obtained by dividing the register by 256 or by shifting the register 8 bits to the right. When it is in the LSB, its value may be obtained by performing a bit-wise AND operation with 255 (FF hex).

MSB (8 bits) LSB (8 bits)	to extract…
[d d d d d d d d_ _ _ _ _ _ _ _]	divide by 256 or shift right 8 bits
[_ _ _ _ _ _ _ _d d d d d d d d]	AND with 255

2.2.2 Integer and Unsigned Integer

Since integer and unsigned integer data are both 16-bits wide, these data types may be represented by exactly one 16-bit register and require no extra math operations to extract or combine in order to obtain their values. The only consideration is with respect to overflow and preserving the sign of any operations performed on the data.
Example: Value=3210H
MSB: 32H
LSB: 10H
The transmitter returns this value as 3210H.

2.2.3 Long and Unsigned Long

Long and unsigned long data is stored in the transmitter as a single 32-bit value in little endian format, which is accessed as two contiguous 16-bit registers via the Modbus interface.
Example:
Value=76543210 hexadecimal (=1,985,229,328 decimal)
Registers=40003-40004 (memory address=02-05)

Memory
02: 10h
03: 32h
04: 54h
05: 76h

When the value is read (or written) via the Modbus interface:
Register 40003 = 3210h (data high=32h, data low=10h)
Register 40004 = 7654h (data high=76h, data low=54h)

To reconstruct the value, the Modbus server needs only to reorder the registers, not the bytes within the registers. Most Modbus servers permit you to specify this during device setup.

2.2.4 Real (Single Precision IEEE 754)

Single precision data is stored in the transmitter as a 32-bit IEEE 754[1] value in little endian format, which is accessed as two contiguous 16-bit registers via the Modbus interface.
Example:
Value=5000 (EB=45h, F0=9Ch, F1=40h, F2=00h)
Registers=40003-40004 (memory address=02-05)

Memory
02: 00h
03: 40h
04: 9Ch
05: 45h

When the value is read (or written) via the Modbus interface:
Register 40003 = 4000h (data high=40h, data low=00h)
Register 40004 = 459Ch (data high=45h, data low=9Ch)

[1] IEEE 754 format is [EB][F0][F1][F2], where EB is the biased exponent, and F0–F2 is the fraction from MSB to LSB.

To reconstruct the value, the Modbus server needs only to reorder the registers, not the bytes within the registers. Most Modbus servers permit you to specify this during device setup.

2.2.5 Strings

Strings are stored as pairs of byte swapped characters, with a terminating 0.
Example: "ATI D12 Transmitter".

```
LOW REGISTER
['T','A']
[' ','I']
['1','D']
[' ','2']
['r','T']
['n','a']
['m','s']
['t','i']
['e','t']
[ 0, 'r']
HIGH REGISTER
```

2.3 Transmitter Registers

Modbus™ registers overlay the transmitter's memory space and are listed by transmitter version in Appendix A. Every register listed is readable, however, only a small group of registers may be written to. Within the list are the register number, read/write access rights, tag name, data type, and description. While tag names are not part of the protocol, they are useful when building a database to be used for SCADA software that might run on a computer.

Network performance is improved by grouping registers that contain the most important data, so that they may be read as a block using one query, instead of two or more. This is true of the block of registers from 40035-40047, which contains the transmitter status, faults (if any), gas concentration, temperature, and current loop readings. This entire block may be read in one operation using function 3 – "Read Holding Registers."

The transmitter is available to decode and respond to query messages at the rate of five times per second, or once every 200ms. When a query arrives at the transmitter, it is buffered until the transmitter can decode it and form a response, which can take up to an additional 50ms. Therefore, the amount of time after a message arrives and before a response might begin to be transmitted may be as long as 250ms. At 9600 baud, the transmit time is 1.04ms per message byte. For the register block mentioned above, the total maximum response time would be 250ms + 1.04ms * 37 bytes (32 data + 5 overhead) = 289ms, or approximately 3½ queries per second, maximum.

2.4 Transmitter Subroutines

The transmitter does not permit writing directly to all registers. This is to protect the data integrity of the transmitter by restricting data values within an acceptable range, performing status checks prior to updating values, programming EEPROMs, calculating checksums, and updating error codes. Instead, transmitter setup values are updated by making subroutine calls. A subroutine call is simply a "write-sequence" to some special purpose registers.

Like other subroutines you may have encountered in computer programming, many of the subroutines expect data to be passed to them and are able to indicate if any problems occurred during execution. The first 14 registers are special purpose registers, used for performing subroutine calls. There are four 16-bit registers for accepting parameter data, a subroutine selection register, an error status register, and eight registers for providing return data (if any).

40001	Subroutine Register
40002	Error Status Register
40003-40006	Parameter Data Registers
40007-40014	Return Data Registers

Data types for parameter and return data can vary from one subroutine call to another. The table below lists the tags, data types, and Modbus register numbers for passing function parameters and results.

Name/Tag	Register	Data Type	No. of Regs
D12_MB_IDATA0	40003	INTEGER	1
D12_MB_IDATA1	40004	INTEGER	1
D12_MB_IDATA2	40005	INTEGER	1
D12_MB_IDATA3	40006	INTEGER	1
D12_MB_UDATA0	40003	UNSIGNED INT	1
D12_MB_UDATA1	40004	UNSIGNED INT	1
D12_MB_UDATA2	40005	UNSIGNED INT	1
D12_MB_UDATA3	40006	UNSIGNED INT	1
D12_MB_RDATA0	40003	REAL	2
D12_MB_RDATA1	40005	REAL	2
D12_MB_IRET0	40007	INTEGER	1
D12_MB_IRET1	40008	INTEGER	1
D12_MB_IRET2	40009	INTEGER	1
D12_MB_IRET3	40010	INTEGER	1
D12_MB_URET0	40007	UNSIGNED INT	1
D12_MB_URET1	40008	UNSIGNED INT	1
D12_MB_URET2	40009	UNSIGNED INT	1
D12_MB_URET3	40010	UNSIGNED INT	1
D12_MB_RRET0	40007	REAL	2
D12_MB_RRET1	40009	REAL	2

Attempting to write to registers above 40014 will result in an exception error 02, "Illegal Data Address" (see PI-MBUS-300).

The typical subroutine call sequence involves three steps that must be carried out in order:

Step 1. Write the Parameter Data Registers (if any)
Step 2. Write the Subroutine Selection Register
Step 3. Read the Error Status (and Result Registers if applicable)

2.5 Master Device Interfaces

Your type of master device will determine the appearance of your subroutine calls. Some masters require you to work at a primitive level, where you must specify register numbers, or even a hexadecimal address in the slave, then decide which function to perform in order to transfer data. Other masters, such as a computer based SCADA system, maintain a database of register numbers and permit you to reference them in a high level language by their tag name, such as D12_MB_IDATA0 and D12_MB_SUB. This is the convention chosen for this manual when describing subroutine calls. For example, assume you wish to change the caution alarm set point to 10.0. This would appear as the following code sequence:

```
D12_MB_IDATA0=0;    //write 1st parameter to select the caution alarm
D12_MB_FDATA1=10.0; //write 2nd parameter to the new value of the set point
D12_MB_SUB=25;      //write the subroutine register to execute the function
If D12_MB_ERROR>0 Then Goto ErrorHandler
```

2.6 Subroutine Reference

The Modbus interface supports many subroutines for configuring and calibrating the transmitter. In the subroutine list that follows, the required sequence of writes is numbered in each table's left hand column. For the sake of clarity, read back of the Error Status is not listed, but should be done to ensure subroutines have been executed successfully (Error Status = 0).

2.6.1 #10 ZERO SENSOR

This subroutine forces the transmitter reading to zero and updates the sensor memory. Prior to executing this subroutine, make certain that the sensor is exposed to a zero gas atmosphere, or that proper zero calibration gas is flowing at the proper rate to the sensor's calibration cap. Zero calibration gas is specific to the sensor; follow the calibration guidelines in the transmitter user manual.

Data Required		*Type*	*Description of Value*
1	D12_MB_SUB	UINT	10
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 4=Cannot verify sensor memory 7=Sensor output too low 8=Sensor output too high 9=Power-up delay 10=Faults present 11=Executing auto-test
Associated Data		*Type*	*Description of Value*
	D12_SMARTS_GASNAME	STRING[16]	Gas name string
	D12_SMARTS_GASUNITS	STRING[8]	Gas units string

2.6.2 #11 SPAN SENSOR

This subroutine forces the transmitter reading to the concentration value specified while flowing gas to the sensor. Prior to executing this subroutine, make certain that the gas has been flowing at 500cc/min to the sensor's calibration cap for at least 5 minutes. Span calibration gas is specific to the sensor; follow the calibration guidelines in the transmitter user manual.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	Calibration gas concentration
2	D12_MB_SUB	UINT	11
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 4=Cannot verify sensor memory 7=Sensor output too low 8=Sensor output too high

			9=Power-up delay 10=Faults present 11=Executing auto-test 12=Supplied parameter too low 13=Supplied parameter too high
Associated Data		*Type*	*Description of Value*
	D12_SMARTS_GASNAME	STRING[16]	Gas name string
	D12_SMARTS_GASUNITS	STRING[8]	Gas units string
	D12_SMARTS_RANGEMAX	REAL	Concentration must be below this value * 1.2
	D12_SMARTS_RANGEMIN	REAL	Concentration must be above this value * 0.05

2.6.3 #14 CHANGE SENSOR RANGE

This subroutine changes the range of the sensor to the value in D12_MB_RDATA0. The 4-20mA current loop output is scaled between 0 and the sensor range value. The units of the sensor range, the upper and lower limits, and default value vary from sensor to sensor. Changing this parameter does not affect the alarm set points or blanking value.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	New sensor range value
2	D12_MB_SUB	UINT	14
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot verify sensor memory 12=Supplied parameter too low 13=Supplied parameter too high 18=Gas generator incompatible with sensor 19=Gas generator incompatible on sensor range
Associated Data		*Type*	*Description of Value*
	D12_SMARTS_RANGE	REAL	Current sensor range
	D12_SMARTS_RANGE_MIN	REAL	Minimum sensor range
	D12_SMARTS_RANGE_MAX	REAL	Maximum sensor range
	D12_SMARTS_RANGE_DEF	REAL	Default sensor range

2.6.4 #15 CHANGE SENSOR DAMPING

This subroutine changes the sensor damping setting to the value in D12_MB_RDATA0. The damping setting is in the range from 10 to 100 and may be scaled into an approximate first order time constant by: TC=20+setting/10. Typically, you would multiply the TC value by 3 to calculate the amount of time required to reach 95% of the final value.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	New damping setting (TC~=20+setting/10)
2	D12_MB_SUB	UINT	15
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot verify sensor memory 12=Supplied parameter too low 13=Supplied parameter too high

Associated Data		Type	Description of Value
	D12_SMARTS_AVERAGING	REAL	Current damping setting
	D12_SMARTS_AVERAGING_MIN	REAL	Minimum damping setting
	D12_SMARTS_AVERAGING_MAX	REAL	Maximum damping setting
	D12_SMARTS_AVERAGING_DEF	REAL	Default damping setting

2.6.5 #16 CHANGE SENSOR BLANKING

This subroutine changes the sensor blanking to the value in D12_MB_RDATA0. The sensor blanking value specifies the positive and negative limits of a band, centered at zero, within which the transmitter reading is forced to zero. This value, and its limits, is maintained internally as a ratio to the programmed full-scale range. The blanking value may be converted to concentration units by multiplying it by the sensor range.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	New blanking ratio
2	D12_MB_SUB	UINT	16
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot verify sensor memory 12=Supplied parameter too low 13=Supplied parameter too high
Associated Data		*Type*	*Description of Value*
	D12_SMARTS_BLANKING	REAL	Current blanking ratio
	D12_SMARTS_BLANKING_MIN	REAL	Minimum blanking ratio (0.0)
	D12_SMARTS_BLANKING_MAX	REAL	Maximum blanking ratio
	D12_SMARTS_BLANKING_DEF	REAL	Default blanking ratio

2.6.6 #17 CHANGE AUTOTEST STATE (n/a for most IR sensors)

This subroutine changes the auto-test state to the value in D12_MB_UDATA0.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	0=OFF …auto-test will not begin automatically 1=READY …auto-test may begin automatically 2=START …starts auto-test immediately 3=STOP …halts auto-test immediately
2	D12_MB_SUB	UINT	17
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot verify sensor memory 7=Sensor input too low 8=Sensor input too high 9=Power-up delay 10=Faults present 11=Executing auto-test 17=Gas generator not installed 18=Gas generator incompatible with sensor 19=Gas generator incompatible on sensor range 20=Concentration too high to begin autotest
Associated Data		*Type*	*Description of Value*
	D12_SYS_AUTOTSTATE- _DRVRIMAGE / 256	UINT	Current auto-test state

2.6.7 #18 CALIBRATE TEMPERATURE READING

This subroutine forces the temperature reading to the value specified in D12_MB_RDATA0, and updates the temperature calibration data. Error code 6 is returned on versions not equipped with a temperature sensor, such as the catalytic bead (combustible gas) version.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	Corrected temperature, -99.0 to 99.0 °C
2	D12_MB_SUB	UINT	18
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 6=Temperature reading not supported 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYS_CELSIUS	REAL	Current temperature reading in °C

2.6.8 #80 CHANGE ALTITUDE (DO sensor versions only)

This subroutine changes the altitude setting (in feet) to the value in D12_MB_IDATA0. The altitude data is used during calibration on the Dissolved Oxygen version only.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_IDATA0	INT	New altitude, -999 to 9999 ft., def = 0 ft
2	D12_MB_SUB	UINT	80
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 6=Altitude setting not supported 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_ALTITUDE	INT	Current altitude setting in feet

2.6.9 #81 CHANGE PRESSURE (DO sensor versions only)

This subroutine changes the air pressure setting (in mmHg) to the value in D12_MB_IDATA0. The air pressure data is used during calibration of the Dissolved Oxygen version only.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_IDATA0	INT	New pressure, 500 to 850 mmHg, def = 760 mmHg
2	D12_MB_SUB	UINT	81
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 6=Pressure setting not supported 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory

Associated Data		Type	Description of Value
	D12_SYSSETUP_PRESSURE	INT	Current pressure setting in mmHg

2.6.10 #82 CHANGE RELATIVE HUMIDITY (DO sensor versions only)

This subroutine changes the relative humidity setting to the value in D12_MB_IDATA0. The relative humidity data is used during calibration of the Dissolved Oxygen version only.

Data Required (in order shown)		Type	Description of Value
1	D12_MB_IDATA0	INT	New relative humidity, 0 to 100 %, def = 0%
2	D12_MB_SUB	UINT	82
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 6=Relative humidity setting not supported 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_RELATIVEHUM	INT	Current relative humidity setting in %

2.6.11 #83 CHANGE SALINITY (DO sensor versions only)

This subroutine changes the salinity concentration setting (in PPM) to the value in D12_MB_UDATA0. The salinity data is used during calibration of the Dissolved Oxygen version only.

Data Required (in order shown)		Type	Description of Value
1	D12_MB_UDATA0	UINT	New salinity setting, 0 to 60,000 PPM, def = 0 PPM
2	D12_MB_SUB	UINT	83
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 6=Salinity setting not supported 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_SALINITY	UINT	Current salinity setting in PPM

2.6.12 #29 CHANGE INHIBIT TIMEOUT PERIOD

This subroutine changes the alarm inhibit timeout period (in minutes) to the value in D12_MB_UDATA0. The alarm inhibit timeout period is the amount of time that alarms are held inactive immediately after sensor calibration, or when the transmitter is manually placed into the inhibit mode for maintenance.

Data Required (in order shown)		Type	Description of Value
1	D12_MB_UDATA0	UINT	New inhibit timeout, 0 to 5999 mins, def = 15 mins
2	D12_MB_SUB	UINT	29
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high

			14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_IHBTMOMINS	UINT	Current alarm inhibit period in minutes

2.6.13 #30 CHANGE ALARM INHIBIT STATE

This subroutine changes the alarm inhibit state according to the value in D12_MB_UDATA0. When D12_MB_UDATA0 is 1, alarms are inhibited (disabled) and a timer is loaded with the alarm inhibit timeout period. Alarms remain inhibited until the timer counts down to zero. When D12_MB_UDATA0 is 0, alarms are re-enabled and the timer is cleared to zero.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	1=Inhibit alarms 0=Cancel alarm inhibit
2	D12_MB_SUB	UINT	30
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_IHBTMOMINS	UINT	Current alarm inhibit period in minutes

2.6.14 #31 RESET ALARMS

This subroutine attempts to reset the manual-reset (latched) alarms specified in D12_MB_UDATA0, thereby clearing the respective alarm status bits in D12_SYS_STATUS. The alarm status bits are cleared only if the respective alarm condition has been removed. In most cases, this means the gas concentration triggering the alarm must reach the reset point. Note this also clears the latched over range condition on catalytic bead (combustible gas) sensor versions.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	BITS: x x x x x x x x x x x x x A W C Where, A=Alarm, W=Warning, C=Caution alarms
2	D12_MB_SUB	UINT	31
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 30=Cannot clear overrange (Cat. Bead Ver. Only)
Associated Data		*Type*	*Description of Value*
	D12_SYS_STATUS	UINT	Alarm status bits

2.6.15 #20 CHANGE ALARM SET POINT

This subroutine changes the value of either the Caution, Warning, or Alarm set points. The subroutine also changes the value of the corresponding reset point to the same value as the set point. The minimum, maximum, and default set point values are stored in the sensor memory as ratios to the sensor range. They may be converted to units of concentration by multiplying them by the sensor range. While the set point value is usually positive, it may be set to a negative value to activate when the sensor's un-blanked reading drifts below zero. The transmitter will activate a fault alarm when the un-blanked value reaches 20%FS below zero. Remember to configure the alarm's options so it will activate correctly above, or below, the set point.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	0=Caution 1=Warning 2=Alarm
2	D12_MB_RDATA1	REAL	New set point in units of concentration (i.e., PPM)
3	D12_MB_SUB	UINT	20
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot verify sensor memory 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_ALMSETUP_C_SETPOINT	REAL	Current Caution (alarm) set point
	D12_ALMSETUP_W_SETPOINT	REAL	Current Warning (alarm) set point
	D12_ALMSETUP_A_SETPOINT	REAL	Current Alarm (alarm) set point
	D12_SMARTS_CAUTIONSPMIN * D12_SMARTS_RANGE	REAL	Minimum Caution (alarm) set point value
	D12_SMARTS_WARNINGSPMIN * D12_SMARTS_RANGE	REAL	Minimum Warning set point value
	D12_SMARTS_ALARMSPMIN * D12_SMARTS_RANGE	REAL	Minimum Alarm set point value
	D12_SMARTS_CAUTIONSPMAX * D12_SMARTS_RANGE	REAL	Maximum Caution set point value
	D12_SMARTS_WARNINGSPMAX * D12_SMARTS_RANGE	REAL	Maximum Warning set point value
	D12_SMARTS_ALARMSPMAX * D12_SMARTS_RANGE	REAL	Maximum Alarm set point value
	D12_SMARTS_CAUTIONSPDEF * D12_SMARTS_RANGE	REAL	Default Caution set point value
	D12_SMARTS_WARNINGSPDEF * D12_SMARTS_RANGE	REAL	Default Warning set point value
	D12_SMARTS_ALARMSPDEF * D12_SMARTS_RANGE	REAL	Default Alarm set point value

When a new type of sensor is installed, the transmitter reads the three (C,W,A) set point ratio values from the sensor, converts them to units of concentration. After displaying each of them during sensor review, it copies the set points to the reset points and stores them in its own memory space.

2.6.16 #21 CHANGE ALARM RESET POINT

This subroutine changes the value of either the Caution, Warning, or Alarm reset points. The limits for this value depend on whether the alarm is active at or ABOVE the set point, or at or BELOW the set point.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	0=Caution 1=Warning 2=Alarm
2	D12_MB_RDATA1	REAL	New reset point in units of concentration (i.e., PPM) *For alarms active ABOVE set point …* Max=D12_[C,W,A]_SETPOINT Min=D12_SMARTS_[C,W,A]SPMIN*D12_SMARTS_RANGE Def=D12_SMARTS_[C,W,A]_SETPOINT *For alarms active BELOW set point …* Max=D12_SMARTS_[C,W,A]SPMAX*D12_SMARTS_RANGE Min=D12_[C,W,A]_SETPOINT Def=D12_SMARTS_[C,W,A]_SETPOINT
3	D12_MB_SUB	UINT	21
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot verify sensor memory 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory 15=Alarm disabled, cannot change reset point
Associated Data		*Type*	*Description of Value*
	D12_ALMSETUP_C_RESPOINT	REAL	Current Caution reset point
	D12_ALMSETUP_W_RESPOINT	REAL	Current Warning reset point
	D12_ALMSETUP_A_RESPOINT	REAL	Current Alarm reset point
	D12_SMARTS_CAUTIONSPMIN * D12_SMARTS_RANGE	REAL	Minimum Caution set point value
	D12_SMARTS_WARNINGSPMIN * D12_SMARTS_RANGE	REAL	Minimum Warning set point value
	D12_SMARTS_ALARMSPMIN * D12_SMARTS_RANGE	REAL	Minimum Alarm set point value
	D12_SMARTS_CAUTIONSPMAX * D12_SMARTS_RANGE	REAL	Maximum Caution set point value
	D12_SMARTS_WARNINGSPMAX * D12_SMARTS_RANGE	REAL	Maximum Warning set point value
	D12_SMARTS_ALARMSPMAX * D12_SMARTS_RANGE	REAL	Maximum Alarm set point value
	D12_SMARTS_CAUTIONSPDEF * D12_SMARTS_RANGE	REAL	Default Caution set point value
	D12_SMARTS_WARNINGSPDEF * D12_SMARTS_RANGE	REAL	Default Warning set point value
	D12_SMARTS_ALARMSPDEF * D12_SMARTS_RANGE	REAL	Default Alarm set point value

When a new type of sensor is installed, the transmitter reads the three (C,W,A) set point ratio values from the sensor, converts them to units of concentration. After displaying each of them during sensor review, it copies the set points to the reset points and stores them in its own memory space.

2.6.17 #22 CHANGE ALARM SET DELAY

This subroutine changes the value of the Caution, Warning, or Alarm set delay period. The set delay period specifies the number of seconds that the concentration must be at the set point value before annunciating the alarm.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	0=Caution 1=Warning 2=Alarm
2	D12_MB_UDATA1	UINT	New set delay period, 0 to 10 secs, def=0 secs
3	D12_MB_SUB	UINT	22
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_ALMSETUP_C_SETDELAY	UINT	Current Caution set delay
	D12_ALMSETUP_W_SETDELAY	UINT	Current Warning set delay
	D12_ALMSETUP_A_SETDELAY	UINT	Current Alarm set delay

2.6.18 #23 CHANGE ALARM RESET DELAY

This subroutine changes the value of the Caution, Warning, or Alarm reset delay period. The reset delay period specifies the number of seconds that the concentration must have subsided below (if active above) or above (if active below) the reset point before de-activating the alarm.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	0=Caution 1=Warning 2=Alarm
2	D12_MB_UDATA1	UINT	New reset delay period, 0 to 7200 secs, def=0 secs
3	D12_MB_SUB	UINT	23
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_ALMSETUP_C_RESDELAY	UINT	Current Caution reset delay
	D12_ALMSETUP_W_RESDELAY	UINT	Current Warning reset delay
	D12_ALMSETUP_A_RESDELAY	UINT	Current Alarm reset delay

2.6.19 #27 CHANGE ALARM OPTIONS

This subroutine changes the value of either the Caution, Warning, or Alarm option bits. There are 5 bits that configure the specified alarm as follows:
a)Disabled, active above, or active below set point
b)Hold on fault, set on fault, and clear on fault
c)Manual reset or auto-reset

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	0=Caution 1=Warning 2=Alarm
2	D12_MB_UDATA1	UINT	New alarm options: 0000 0000 000R FFDD (16-bits) DD: 00=disabled, 01=active above SP, 10=active below SP FF: 00=hold on fault, 01=set on fault, 10=clear on fault R: 0=manual reset, 1=auto-reset Set all remaining bits to 0
3	D12_MB_SUB	UINT	27
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot verify sensor memory 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_ALMSETUP_C_OPTIONS	UINT	Current Caution alarm options
	D12_ALMSETUP_W_OPTIONS	UINT	Current Warning alarm options
	D12_ALMSETUP_A_OPTIONS	UINT	Current Alarm alarm options

2.6.20 #28 CHANGE RELAY CONFIGURATION

This subroutine changes the activation source and normal state of the specified relay.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	Relay selection 0=RL1 1=RL2 2=RL3
2	D12_MB_UDATA1	UINT	New relay configuration: 0000 0000 000N AAAA (16-bits) <u>AAAA is the activation source…</u> 0000=Caution 0001=Warning 0010=Alarm 0011=Fault 0100=Auto-clean (wet H2S sensor only) <u>N is the normal state of the relay…</u> 0=Normally de-energized 1=Normally energized (also known as failsafe)
3	D12_MB_SUB	UINT	28
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_R3CFG/256	UINT	Current configuration of RL3
	D12_SYSSETUP_R2CFG_R1CFG/256	UINT	Current configuration of RL2
	D12_SYSSETUP_R2CFG_R1CFG&0xff	UINT	Current configuration of RL1

2.6.21 #32 START ALARM RELAY TEST

This subroutine activates the relays associated with the alarms specified in D12_MB_UDATA0. Therefore, in order for a relay to activate, it must be configured for one of the alarms specified in D12_MB_UDATA0 (see CHANGE RELAY CONFIGURATION). If an alarm relay is configured as normally de-energized, this subroutine will energize it. Conversely, if an alarm relay is configured as normally energized (failsafe), this subroutine will de-energize it.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	Alarm bit image … 0000 0000 0000 FAWC F …Fault A …Alarm W …Warning C …Caution
2	D12_MB_SUB	UINT	32
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYS_STATUS D12_SYS_EXPSTATUS	UINT UINT	Alarm status bits will be set Bit 3 will be set

2.6.22 #33 STOP ALARM RELAY TEST

This subroutine ends any alarm relay test in progress.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_SUB	UINT	33
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors
	None		
Associated Data		*Type*	*Description of Value*
	D12_SYS_EXPSTATUS	UINT	Bit 3 will be cleared

2.6.23 #40 CHANGE INHIBIT MA LEVEL

This subroutine changes the level that is output on the current loop when the transmitter enters alarm inhibit mode.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	New inhibit mA level
2	D12_MB_SUB	UINT	40
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_LOOP_INHIBMA	REAL	Current inhibit mA level

2.6.24 #41 CHANGE FAULT MA LEVEL

This subroutine changes the level that is output on the current loop when a fault is detected.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	New fault mA level
2	D12_MB_SUB	UINT	41
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_LOOP_FAILMA	REAL	Current fault mA level

2.6.25 #42 CHANGE AUTOTEST MA LEVEL

This subroutine changes the level that is output on the current loop when the transmitter enters auto-test.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	New auto-test mA level
2	D12_MB_SUB	UINT	40
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_LOOP_ATMA	REAL	Current auto-test mA level

2.6.26 #47 FORCE MA OUTPUT

This subroutine places the transmitter into a forced mA output mode, and outputs the level specified in D12_MB_RDATA0 on the current loop. This forced output mode overrides the normal level proportional to gas concentration, as well as the levels associated with alarm inhibit mode, fault, and auto-test. The forced output mode ends by executing this subroutine with 0.0 in D12_MB_RDATA0. This subroutine should be executed prior to correcting either the 4mA level (sub #48), or the 20mA level (sub #49).

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	Forced mA level, 3.5-22.0 mA (Note: 0.0 ends forced output and returns loop to normal operation)
2	D12_MB_SUB	UINT	47
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYS_LOOPFIXEDMA	REAL	Fixed output active at this mA level (0.0 = disabled)

2.6.27 #48 CORRECT 4MA LEVEL

This subroutine corrects the 4mA current loop output based on the value received in D12_MB_RDATA0. If the value is greater than 4.00, the transmitter applies a negative correction to the 4mA calibration point. Conversely, the transmitter applies a positive correction if the supplied value is less than 4.00. The subroutine may need to be repeated to "zero-in" on the most accurate result. Prior to performing this subroutine, the transmitter output should be forced to 4.00mA using subroutine #47 above.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	Measured 4mA level
2	D12_MB_SUB	UINT	48
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory 16=Not in fixed current output mode
Associated Data		*Type*	*Description of Value*
	D12_SYS_LOOPFIXEDMA	REAL	Fixed output active at this mA level (0.0 = disabled)

2.6.28 #49 CORRECT 20MA LEVEL

This subroutine corrects the 20mA current loop output based on the value received in D12_MB_RDATA0. If the value is greater than 20.0, the transmitter applies a negative correction to the 20mA calibration point. Conversely, the transmitter applies a positive correction if the supplied value is less than 20.0. The subroutine may need to be repeated to "zero-in" on the most accurate result. Prior to performing this subroutine, the transmitter output should be forced to 20.0mA using subroutine #47 above.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	Measured 20mA level
2	D12_MB_SUB	UINT	49
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory 16=Not in fixed current output mode
Associated Data		*Type*	*Description of Value*
	D12_SYS_LOOPFIXEDMA	REAL	Fixed output active at this mA level (0.0 = disabled)

2.6.29 #50 CHANGE LOCK STATUS

This subroutine activates security based on the value passed in D12_MB_UDATA0.

Data Required (in order shown)		Type	Description of Value
1	D12_MB_UDATA0	UINT	New lock status, 0=Unlock, 1=Lock
2	D12_MB_SUB	UINT	50
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_USERLOCK	UINT	Current lock status
	D12_SYS_USECTIMER	UINT	Value > 0 indicates transmitter is unlocked

Note that the transmitter may be temporarily unlocked. This occurs when the operator enters the password to unlock, then elects to automatically relock after a specified period of time. This condition may be determined by polling both D12_SYSSETUP_USERLOCK and D12_SYS_USECTIMER.

D12_SYSSETUP_USERLOCK	D12_SYS_USECTIMER	Lock Status
0	*0*	*Not valid*
0	*>0*	*Unlocked*
1	*0*	*Locked*
1	*>0*	*Temporarily Unlocked*

2.6.30 #51 CHANGE PASSWORD

This subroutine changes the password to the value in D12_MB_UDATA0.

Data Required (in order shown)		Type	Description of Value
1	D12_MB_UDATA0	UINT	New password, 0 to 9999, def=0, master=680
2	D12_MB_SUB	UINT	51
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_USERPSWD	UINT	Current password value

2.6.31 #60 CHANGE SYSTEM CLOCK

This subroutine forces the system clock to the data supplied in the D12_MB_UDATA0-3 registers.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	00mmmmmm 000hhhhh (16-bits) m are the minutes digits, 0-59 h are the hours digits, 0-23
2	D12_MB_UDATA1	UINT	00000ddd 00ssssss (16-bits) d are the day of the week digits, 1-7(Mon-Sun) s are the seconds digits, 0-59
3	D12_MB_UDATA2	UINT	0000mmmm 000ddddd (16-bits) m are the month digits, 1-12 d are the date digits, 1-31
4	D12_MB_UDATA3	UINT	Year, 2000 to 2200
5	D12_MB_SUB	UINT	60
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high 14=Cannot verify CPU memory
Associated Data		*Type*	*Description of Value*
	D12_SYS_CLOCK_MIN_HR	UINT	Current clock minutes and hours (see format above)
	D12_SYS_CLOCK_DAY_SEC	UINT	Current clock day and seconds (see format above)
	D12_SYS_CLOCK_MO_DATE	UINT	Current clock month and date (see format above)
	D12_SYS_CLOCK_MIN_YEAR	UINT	Current clock year (see format above)

2.6.32 #61 INPUT KEYCODE

This subroutine simulates key switch actuations of the operator interface panel.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	Keycode: 00000000 001XCEDU (16-bits) where, U is the Up key D is the Down key E is the Enter key C is the Escape key X is used in conjunction with the Up or Down key to designate how long the key has been held: 0 = 1 count, 1 = 10 counts This is used during parameter edit for speeding up the increment and decrement of a value.
2	D12_MB_SUB	UINT	61
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 12=Supplied parameter too low 13=Supplied parameter too high
Associated Data		*Type*	*Description of Value*
	D12_SYSSETUP_USERPSWD	UINT	Current password value

2.6.33 #70 CHANGE DATALOG STATE

This subroutine changes the non-volatile state of the data log according to the value in D12_MB_UDATA0. The data log state may be changed to On, Off, or Clear. When On is specified, the data log begins sampling immediately at the programmed sampling rate. When the D12_MB_UDATA0 specifies clear, the data log remains in the clear state until the data has been cleared and then it automatically returns to its previous state – On or off.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	New data logger state, 0=Off,1=On,2=Clear
2	D12_MB_SUB	UINT	70
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high
Associated Data		*Type*	*Description of Value*
	D12_DLOG_STATE_ORGINDEX/256	UINT	Current data log state

2.6.34 #71 CHANGE DATALOG SAMPLING RATE

This subroutine changes the data logger sampling rate according to the value in D12_MB_UDATA0. If the new sampling rate is different from the previous rate, and there is data in the data logger memory, the data log memory will be cleared.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	New sampling rate index: 0=1/min (11 days) 1=2/min (22 days) 2=3/min (32 days) 3=4/min (43 days) 4=5/min (54 days) 5=6/min (64 days) 6=10/min (104 days) 7=12/min (124 days) 8=15/min (152 days) 9=20/min (196 days) 10=30/min (278 days) 11=60/min (474 days)
2	D12_MB_SUB	UINT	71
Data Returned		*Type*	*Description of Value*
	None		
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 13=Supplied parameter too high
Associated Data		*Type*	*Description of Value*
	D12_DLOG_STATE_ORGINDEX&0xff	UINT	Current sampling rate index

2.6.35 #72 GET DATALOG INFO

This subroutine retrieves information about the data log contents (required for reading actual data).

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_SUB	UINT	72
Data Returned		*Type*	*Description of Value*
	D12_MB_URET0	UINT	Sampling rate interval
	D12_MB_URET1	UINT	Samples per day
	D12_MB_URET2	UINT	Maximum number of days
	D12_MB_URET3	UINT	Sampling rate index
	D12_MB_URET4	UINT	Data log state
	D12_MB_URET5	UINT	Number of days in log
	D12_MB_URET6	UINT	Day number
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors
Associated Data		*Type*	*Description of Value*
	None		

2.6.36 #73 GET DATALOG RECORD

This subroutine retrieves the specified record from the data log. Record numbers run sequentially from oldest to youngest.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	Sample number
2	D12_MB_SUB	UINT	73
Data Returned		*Type*	*Description of Value*
1	D12_MB_URET0	UINT	00mmmmmm 000hhhhh (16-bits) m are the minutes digits, 0-59 h are the hours digits, 0-23
2	D12_MB_URET1	UINT	00000ddd 00ssssss (16-bits) d are the day of the week digits, 1-7(Mon-Sun) s are the seconds digits, 0-59
3	D12_MB_URET2	UINT	0000mmmm 000ddddd (16-bits) m are the month digits, 1-12 d are the date digits, 1-31
4	D12_MB_URET3	UINT	Year, 2000 to 2200
4	D12_MB_URET4	UINT	Sample value
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 14=Cannot retrieve specified sample from memory
Associated Data		*Type*	*Description of Value*
	None		

2.6.37 #12 GET CAL. HISTORY INFO

The calibration history is stored in the sensor EEPROM memory from which records may be retrieved both on screen, and via the Modbus interface. There are 63 zero and 63 span records. Each zero record contains a date and the concentration reading just prior to performing the calibration. Each span record contains a date and sensitivity deviation in percent. Once the history fills up (more than 63 user calibrations), the newest record overwrites the oldest record and the reported number of calibrations remains fixed at 63.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_SUB	UINT	12
Data Returned		*Type*	*Description of Value*
1	D12_MB_URET0	UINT	Number of zero calibrations, 0 to 63
2	D12_MB_URET1	UINT	Number of span calibrations, 0 to 63
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function
Associated Data		*Type*	*Description of Value*
	None		

2.6.38 #13 GET CALIBRATION RECORD

Reads a specific zero or span calibration record from the history.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_UDATA0	UINT	0 = request a zero record 1 = request a span record
2	D12_MB_UDATA1	UINT	Record number, 1 (oldest) to 63 (newest)
3	D12_MB_SUB	UINT	13
Data Returned		*Type*	*Description of Value*
1	D12_MB_URET0	UINT	Date, 1 to 31
2	D12_MB_URET1	UINT	Month, 1 to 12
3	D12_MB_URET2	UINT	Year, 2000 to 2200
3	D12_MB_URET3	REAL	*If requesting zero record…* Zero drift correction in units of concentration *If requesting span record…* Sensitivity deviation in percent
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 4=Cannot read/verify sensor memory 12=Supplied parameter too low 13=Supplied parameter too high
Associated Data		*Type*	*Description of Value*
	None		

2.6.39 #100 CONVERT CONCENTRATION FROM REAL TO %FS INTEGER

This subroutine converts real concentration values in units of PPM,PPB,%,and %LEL (according to the currently installed sensor) to integer concentration values in units of %FS. The supplied real concentration value is in IEEE-754 floating point format, and the result is a signed integer value that is scaled to the range of the sensor. The following formula is applied…

PCT_FS = (100 * CONCENTRATION_VALUE) / SENSOR_RANGE

This subroutine is suitable for Modbus masters that do not support IEEE-754 floating-point format. Use it to convert alarm set/reset points into %FS values.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_RDATA0	REAL	Concentration value in units of PPM,PPB,%,or %LEL
2	D12_MB_SUB	UINT	100
Data Returned		*Type*	*Description of Value*
1	D12_MB_IRET0	INT	Equivalent concentration in units of %FS
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function 12=Supplied parameter too low (result<-32768) 13=Supplied parameter too high(result>32767)
Associated Data		*Type*	*Description of Value*
	See subroutine #101.		

2.6.40 #101 CONVERT CONCENTRATION FROM %FS INTEGER TO REAL

This subroutine converts integer concentration values in units of %FS to real concentration values in units of PPM,PPB,%,or %LEL (according to the currently installed sensor). The supplied concentration value is a signed integer (-32768 to +32767), and the result is an IEEE-754 floating-point value that is scaled by the range of the sensor. The following formula is applied…

CONCENTRATION_VALUE= (PCT_FS / 100) * SENSOR_RANGE

This subroutine is suitable for Modbus masters that do not support IEEE-754 floating-point format. Use it to convert %FS values for setting alarm set/reset points.

Data Required (in order shown)		*Type*	*Description of Value*
1	D12_MB_IDATA0	REAL	Concentration value in units of %FS
2	D12_MB_SUB	UINT	100
Data Returned		*Type*	*Description of Value*
1	D12_MB_RRET0	INT	Equivalent concentration in units of PPM,PPB,%, or %FS.
Error Codes		*Type*	*Description of Value*
	D12_MB_ERROR	UINT	0=No errors 1=Sensor removed, cannot perform function
Associated Data		*Type*	*Description of Value*
	See subroutine #100.		

2.7 Transmitter Tables

Table 2 Error codes

1 - Sensor removed
2 – No data in data log
3 – Data log busy, cannot retrieve at this time
4 - Cannot update sensor calibration memory
5 – n/a
6 – Cannot perform this action
7 – Sensor output too low
8 – Sensor output too high
9 - Power up timeout active
10 - Sensor/transmitter fault present
11 - Autotest in progress
12 - Input parameter too small
13 - Input parameter too large
14 – Cannot verify CPU memory
15 – Alarm disabled, cannot change reset point
16 – Not in fixed current output mode

Table 3 Transmitter Status Bits

BIT 0 – Caution active
BIT 1 – Warning active
BIT 2 – Alarm active
BIT 3 – Fault active
BIT 4 – CWAF inhibit active
BIT 5 – Transmitter security active
BIT 6 – System data log active
BIT 7 – Current loop output fixed
BIT 8 – Temperature sensor input over range
BIT 9 – Temperature sensor input under range
BIT 10 – Gas sensor input over range
BIT 11 – Gas sensor input under range
BIT 12 – Data log checksum error
BIT 13 – Calibration history not initialized
BIT 14 – Sensor in Power On Delay
BIT 15 – Reserved

Table 4 Transmitter Expanded Status Bits

BIT0 – Gas generator installed
BIT1 – Gas generator type valid
BIT2 – Gas generator range valid
BIT3 – Alarm test active
BIT4 – Autotest in progress
BIT5 – Autotest pass
BIT6 – Cannot begin autotest
BIT7 – Auto-test failed
BIT8 – Auto-clean active
BIT9 – Auto-clean recovering
BIT 10 – Reserved
BIT 11 – Reserved
BIT 12 – Reserved
BIT 13 – Reserved
BIT 14 – Reserved
BIT 15 – Reserved

Table 5 Transmitter Fault Bits

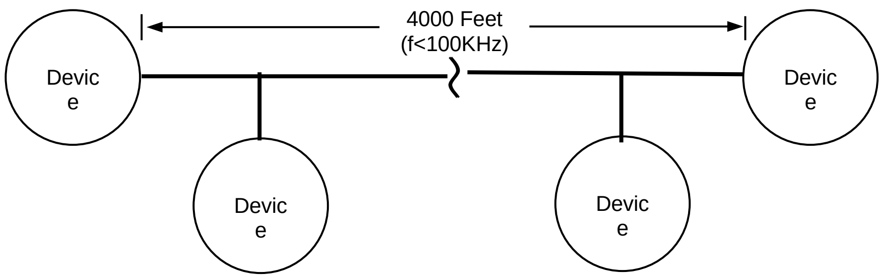
BIT 0 – Gas sensor ADC read fault
BIT 1 – LCD bus fault
BIT 2 – SPI bus fault
BIT 3 – Temperature ADC read fault
BIT 4 – Gas sensor input fault
BIT 5 – Gas sensor removed
BIT 6 – Gas sensor memory checksum fault
BIT 7 – Gas sensor configuration fault (or awaiting verification)
BIT 8 – Gas generator removed
BIT 9 – Gas generator configuration fault (type/range)
BIT 10 – System setup memory checksum fault
BIT 11 – Alarm memory checksum fault
BIT 12 – Operator interface memory checksum fault
BIT 13 – Hart memory checksum fault
BIT 14 – Autotest failure
BIT 15 – Relay option jumper installed but voltage not present

Table 6 Transmitter Expanded Fault Bits

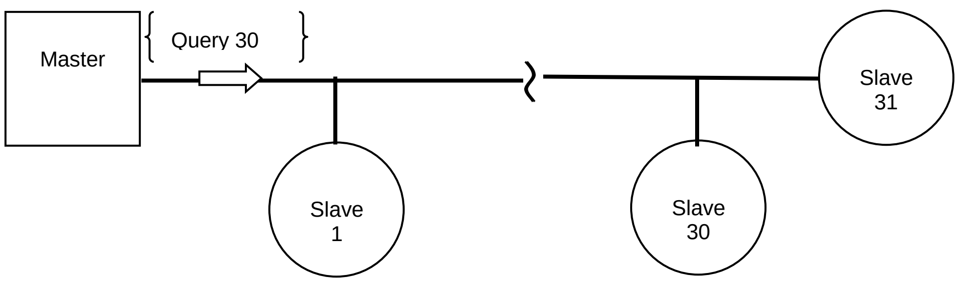
BIT 0 – Factory calibration fault
BIT 1 – Stack overflow

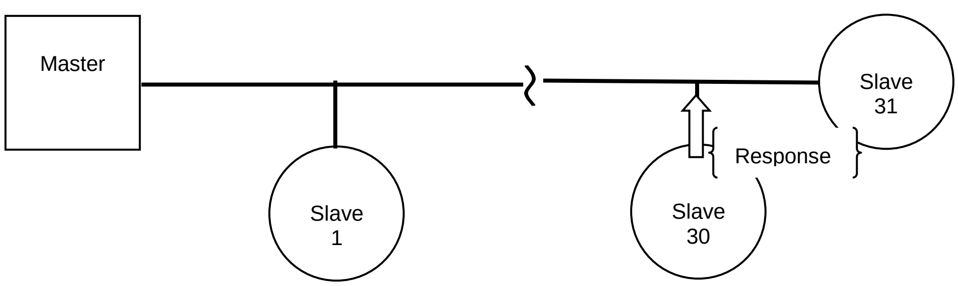
3 Introduction to RS485

The RS485 standard specifies a two-wire, half-duplex serial data bus for connecting up to 32 devices in parallel, at distances of up to 4000 feet at transmission rates at or below 100KHz. The RS485 standard allows the user to configure inexpensive local networks and multi-drop communications links using a twisted pair cable. A typical RS485 network can operate properly in the presence of reasonable ground differential voltages, withstand driver contentious situations, and provide reliable communications in electrically noisy environments with good common mode rejection.

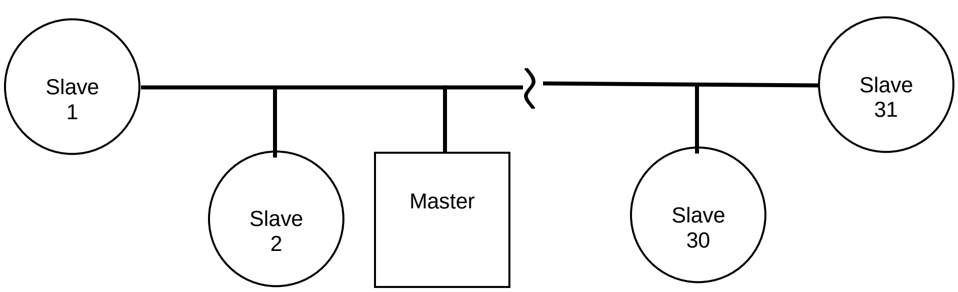


Half-duplex means outgoing messages share the same physical medium with incoming messages. Only one device may transmit at any given time. During any exchange of data communication, one device must act as master and one or more devices act as slaves. With no activity on the bus, the master device sends an addressed query to a slave and then gives up the bus. All slaves receive the message, but only the addressed slave responds.



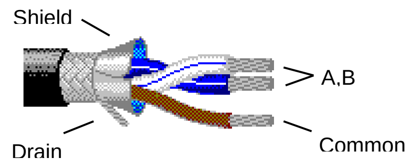


The master node may be located anywhere on the network, not just at one end.

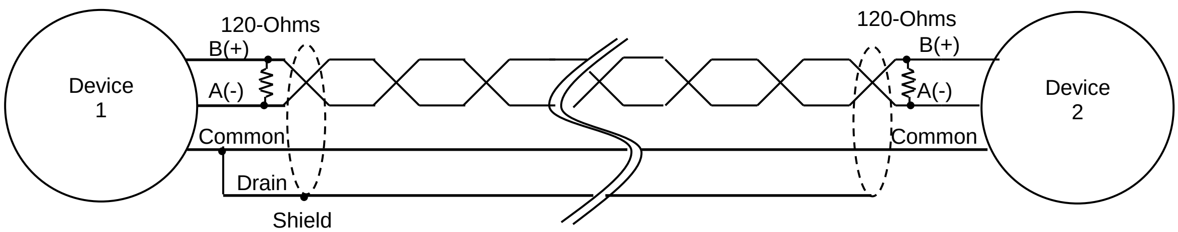


3.1 Bus Cable

The bus is a cable composed of a twisted pair of wires with a characteristic impedance of 120 ohms, and a 120-ohm termination resistor connecting the pair of wires at each end. Several manufacturers offer cables specifically designed for RS485, such as Belden's 3106A, which features one twisted pair, a separate signal common, a foil shield, and a drain wire in contact with the shield.

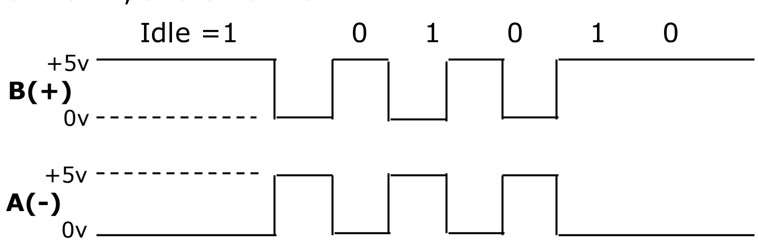


The twisted pair, labeled A and B (or – and +, respectively), form a differential transmission line capable of operating over a common mode voltage range from –7v to +12v (note 2). That is, the ground potential at each end of the network may differ by this amount. Connecting a signal common to each slave device will keep this potential to a minimum. The shield around the conductors provides protection from EMI (electromagnetic interference) and should be connected to common or ground at only one point to avoid circulating currents that might actually generate interference on the inner conductors. A schematic of the bus is shown below.



3.2 RS485 Line Drivers/Receivers

The differential lines, A and B, may be operated at TTL levels of 0 and 5 volts. The RS485 line driver outputs the logic high state (marking, or idle state) by driving 5 volts on B, and 0 volts on A. Conversely, the driver outputs the logic low state (spacing) by driving 5 volts on A, and 0 volts on B.



Over a distance of 4000 feet, the 5 volts applied to either line may be dropped significantly. This usually doesn't present a problem since RS485 receivers are specified to operate with a differential voltage of only 0.200 volts. In practice, however, the differential voltage should remain above 1.5v.

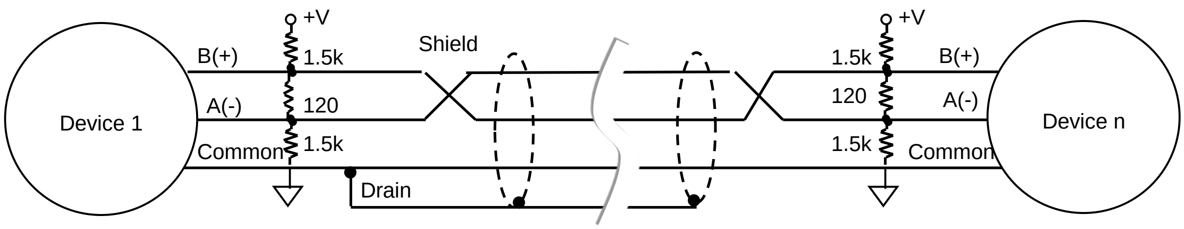
Logic State High	(Idle or Marking State):	(B – A) >= 200mV
Logic State Low	(Spacing State):	(A – B) >= 200mV

3.3 120 Ohm Termination

The two devices at the furthest end of the bus require termination resistors to cancel reflections. Intermediate devices do not. D/F12 transmitters have built in 120-ohm termination that is installed by way of a communication jumper block (see RS485 Jumper Plugs).

3.4 Bias

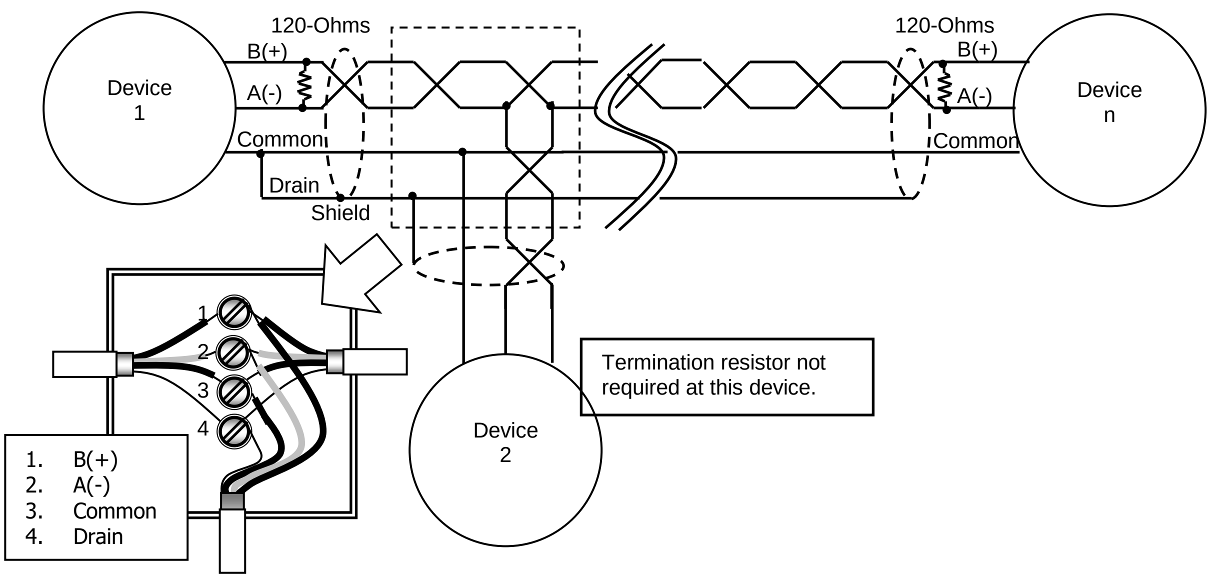
When there is no communication on the network, the A and B lines are floating. A small amount of noise could appear as the start of a message, which might interfere with the framing of valid messages. Biasing the transmission line keeps it in the idle state while it is not driven. The bias resistors maintain a differential of 200mV between the A and B lines.



D/F12 transmitters have built in 120-ohm termination, and 1.5k bias resistors that are installed by way of a communication jumper block (see RS485 Jumper Plugs, below).

3.5 Drops

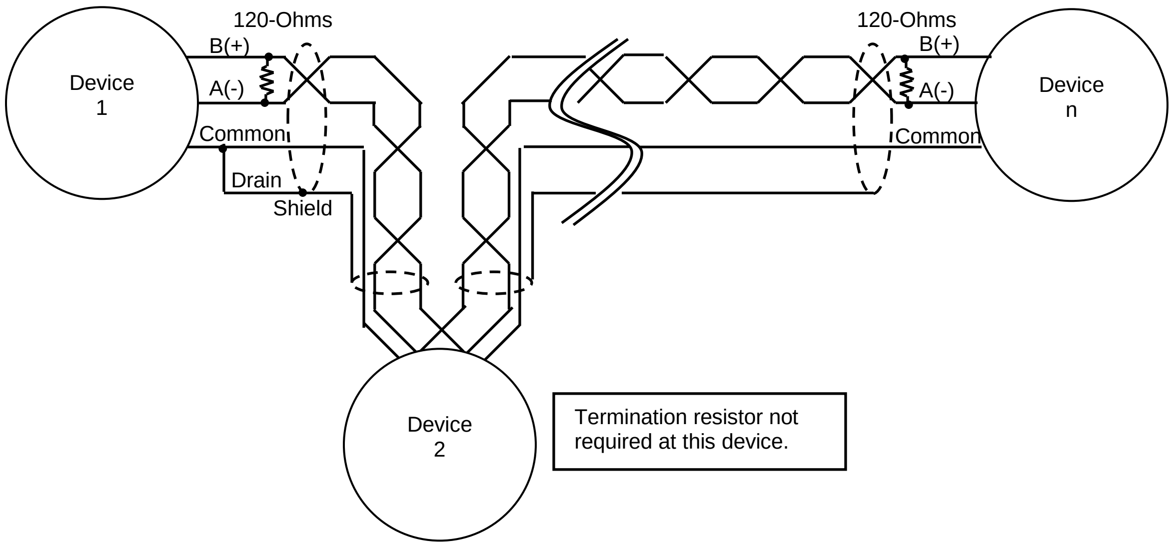
Often, a short length of cable is used at a junction box to form a branch, or "drop", from the bus to the device. These cables must very short as compared to the main trunk length of the bus so as to avoid signal reflections and require termination that would load the bus excessively. A rule of thumb is to not allow any single branch length to exceed 3% of the total trunk length. Again, only the devices at each end of bus require termination resistors, intermediate connections along the bus do not (bias resistors not shown for clarity).



Long branches requiring termination may be connected, however, a repeater must be used at a short distance from the connection. Star topologies should be avoided, since terminating each spoke will load the network excessively and reliable communications cannot be guaranteed.

3.6 Daisy Chaining

For devices not located at the ends of the bus, it may be possible to run the cable in and out of the device, a practice referred to as "daisy-chaining". Although this method eliminates the need for a separate drop wire, it will require more connections inside the transmitter housing and therefore consume more space.

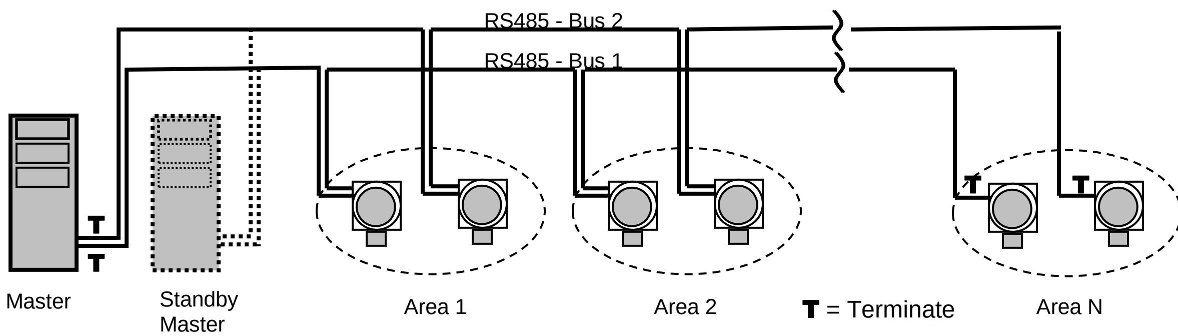


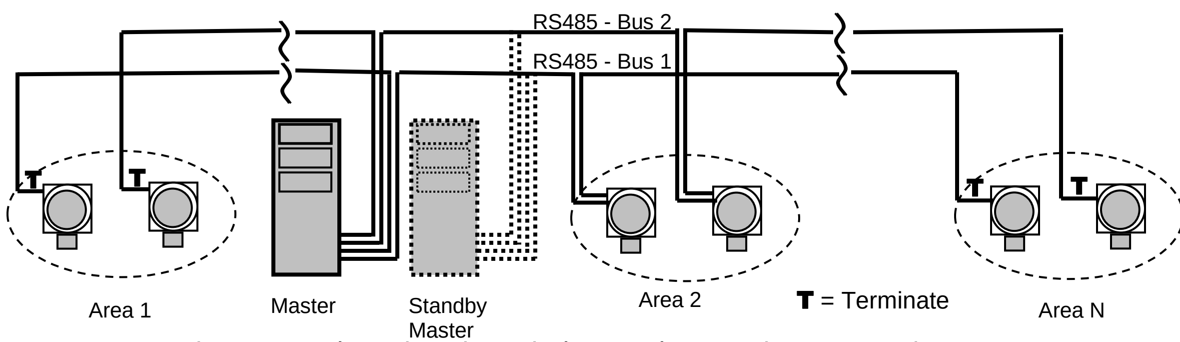
3.7 Isolation

Another consideration with large networks is isolation. Without isolation, severe damage can occur to your entire system, if a high voltage source is connected (accidentally or otherwise) to your communications lines. Your entire network could be damaged. With galvanic isolation the damage is generally limited to only one leg of the network, except in extreme cases of very high voltage (induced by lightening for example). While it goes against conventional wisdom, and can potentially cause a problem with circulating currents by grounding a shielded cable at both ends, this method is very effective at keeping induced lightening noise away from the communications lines. In the alternative, ground one end of the shield and connect the other end to ground through a bi-directional transient protector (from a few volts to a few hundred volts depending on the situation). You can find an extensive line of optical/transformer isolated repeaters and multi-port repeaters as well as a series of fiber optic products which provide very high isolation. These products are extremely effective in applications involving industrial control, large RS485 networks, outdoor data links between buildings, etc.

3.8 Interleaving Devices

When 16 or more transmitters are required, consideration should be given to running separate busses, and interleaving the coverage of multiple transmitters. For optimum safety, each bus should be run in it's own conduit and placed as far from the other as possible. Each bus should be connected to separate RS485 master channels, and if possible, to a standby-master device. The standby master guards against failures of the primary master.

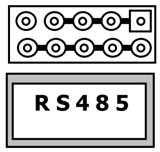
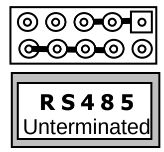


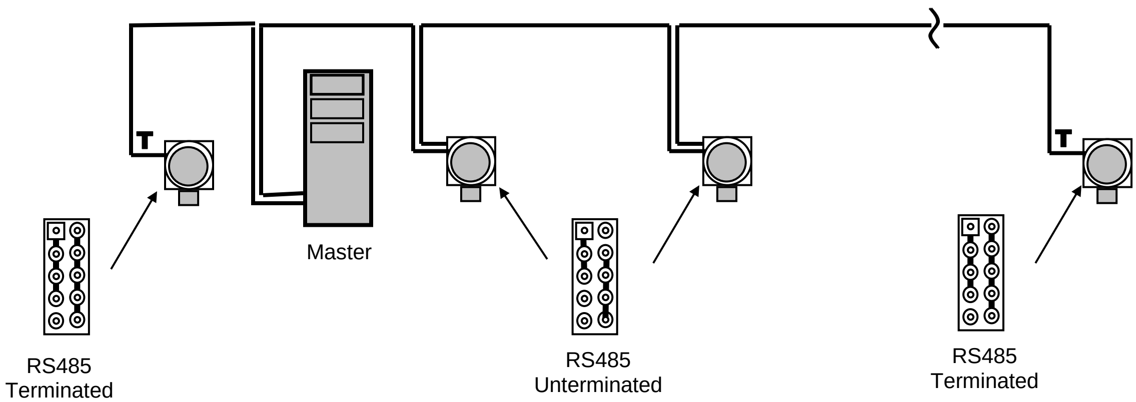


Large networks accumulate distributed electrical noise that can make communications unreliable. It is very important not to run communications wires in the same trough or conduit or in parallel with AC power cables. If there are unused conductors or pairs in the cable, terminate each end with a resistor to ground. Maintain as much distance as possible and cross any power cable at a right angle.

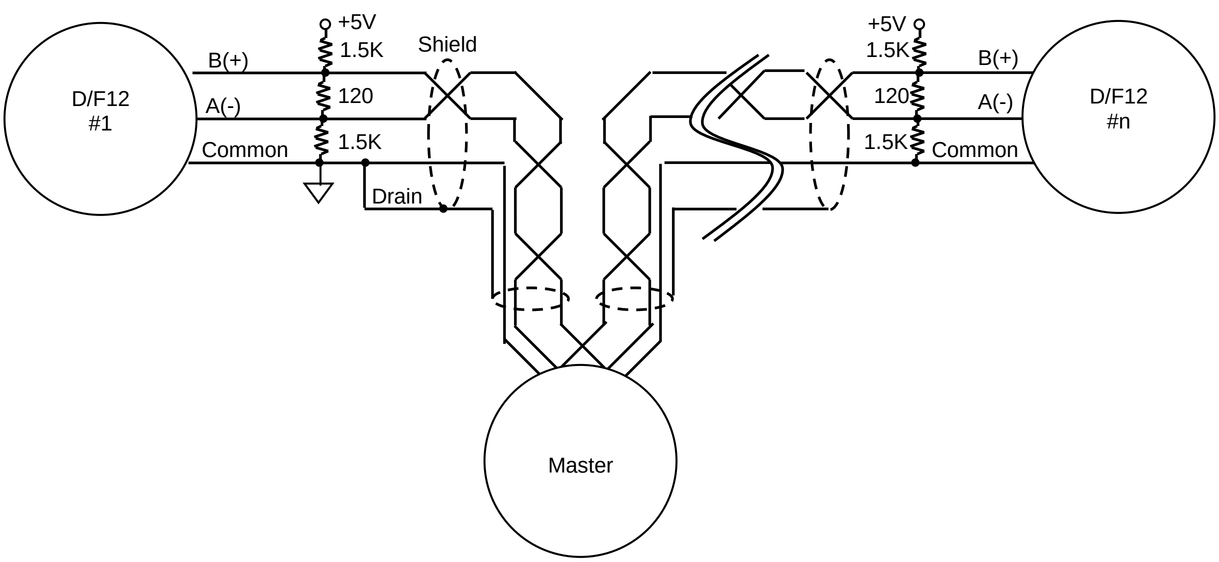
3.9 RS485 Jumper Plugs

D/F12 transmitters connected to RS485 networks require one of two jumper plugs.

RS485 (Terminated)	**RS485 Unterminated**
For use on transmitters requiring termination, i.e., at the ends of the bus cable.	For use on transmitters not requiring termination, i.e., between the ends of the bus cable.
	



The RS485 (terminated) jumper plug also connects bias resistors into the network, as shown in the schematic below. These bias resistors help keep noise from generating false start bits during idle times in the data stream.



3.10 Operator Interface Configuration

```
Menu>Setup>System>Commun

►Protocol=Modbus
 Interface=RS485
 Baud_Rate=9600
 Parametrs=N,8,1
```

Communication Setup page variables are used to configure the protocol and settings of the physical communication interface. The protocol selection is performed at the factory, and may not be changed. Settings for the physical communication interface may be changed for ASCII and Modbus protocols; however, they are restricted for the HART protocol).

Figure 1 Communication setup page

Table 7 Communication setup variables

Variables	Description
`Protocol`	The Protocol variable identifies the installed protocol driver: • None • ASCII (default) • Modbus (option) • HART (option)
`Interface`	The Interface variable selects the physical communication interface that the transmitter will control during transmit and receive functions: • RS232 (available for ASCII or Modbus, not for HART) • RS485 (available for ASCII or Modbus, not for HART) • MODEM (available for HART only) The transmitter must be wired in accordance with this selection. Refer to the transmitter's O&M manual for details.
`Baud_Rate`	The Baud_Rate variable is used to configure the baud rate of the transmitter's UART, and may be set to: 300,600,1200,2400,4800,9600,14.4k, or 28.8k The value is fixed at 1200 for HART protocol, and defaults to 9600 for Modbus and ASCII.
`Parametrs`	The Parametrs variable is used to configure parity, the number of data bits, and number of stop bits of the transmitter's UART: • N,8,1 …no parity, 8 data bits, 1 stop bits • N,8,2 …no parity, 8 data bits, 2 stop bits • E,8,1 …even parity, 8 data bits, 1 stop bit • O,8,1 …odd parity, 8 data bits, 1 stop bit The value is fixed at O,8,1 for HART protocol, and defaults to N,8,1 for Modbus and ASCII.

When the Modbus protocol driver is installed, a link to the Modbus Setup page appears on the main Setup page.

```
>Menu >Setup
  Sensor  DataLog
  Alarms  Display
  Auto-T  System
  4-20mA ►Modbus
```

```
>Menu >Setup >Modbus
 Modbus Setup
 ►Slave_Addr=1
  Time_Out=  3
```

Figure 2 Modbus setup page

Table 8 Modbus setup variables

Variables	Description
Slave_Addr	The Slave_Addr variable is the transmitter's slave address, which may set from 1 (default) to 247.
Time_Out	The Time_Out variable belongs to the data-link layer of the protocol and defines the number of 1.5 character timeouts used to frame messages. This variable is reserved for future use and changing it is not recommended.

4 Appendix A. Register Tables

4.1 D12 and F12 w/H10 Sensor; D12 with Catalytic Bead; D12 Wet H2S Sensors

Variable tag names beginning with D12_ apply to both the D12 and F12 versions of the transmitter, unless noted in the comments.

Register	Access	Tag Name	Type	Comments
40001	RW	D12_MB_SUB	UINT	//used for subroutine calls
40002	RW	D12_MB_ERROR	UINT	//used for returning transmitter specific error codes
40003	RW	D12_MB_IDATA0	UINT	//used for passing integer data to subroutines
40004	RW	D12_MB_IDATA1	UINT	//used for passing integer data to subroutines
40005	RW	D12_MB_IDATA2	UINT	//used for passing integer data to subroutines
40006	RW	D12_MB_IDATA3	UINT	//used for passing integer data to subroutines
40003	RW	D12_MB_UDATA0	UINT	//used for passing unsigned data to subroutines
40004	RW	D12_MB_UDATA1	UINT	//used for passing unsigned data to subroutines
40005	RW	D12_MB_UDATA2	UINT	//used for passing unsigned data to subroutines
40006	RW	D12_MB_UDATA3	UINT	//used for passing unsigned data to subroutines
40003	RW	D12_MB_LDATA0	REAL	//used for passing long data to subroutines
40005	RW	D12_MB_LDATA1	REAL	//used for passing long data to subroutines
40003	RW	D12_MB_RDATA0	REAL	//used for passing float data to subroutines
40005	RW	D12_MB_RDATA1	REAL	//used for passing float data to subroutines
40007	RW	D12_MB_IRET0	INT	//integer data return register
40008	RW	D12_MB_IRET1	INT	//integer data return register
40009	RW	D12_MB_IRET2	INT	//integer data return register
40010	RW	D12_MB_IRET3	INT	//integer data return register
40011	RW	D12_MB_IRET4	INT	//integer data return register
40012	RW	D12_MB_IRET5	INT	//integer data return register
40013	RW	D12_MB_IRET6	INT	//integer data return register
40014	RW	D12_MB_IRET7	INT	//integer data return register
40007	RW	D12_MB_URET0	UINT	//unsigned integer data return register
40008	RW	D12_MB_URET1	UINT	//unsigned integer data return register
40009	RW	D12_MB_URET2	UINT	//unsigned integer data return register
40010	RW	D12_MB_URET3	UINT	//unsigned integer data return register
40011	RW	D12_MB_URET4	UINT	//unsigned integer data return register
40012	RW	D12_MB_URET5	UINT	//unsigned integer data return register
40013	RW	D12_MB_URET6	UINT	//unsigned integer data return register
40014	RW	D12_MB_URET7	UINT	//unsigned integer data return register
40007	RW	D12_MB_RRET0	REAL	//real data return register
40009	RW	D12_MB_RRET1	REAL	//real data return register
40011	RW	D12_MB_RRET2	REAL	//real data return register
40013	RW	D12_MB_RRET3	REAL	//real data return register
40015	RW	D12_MB_LISTENONLY	UINT	//places transmitter into quiet mode

40033	R	D12_SYS_EXPFAULTS	UINT	//expanded fault bits
40034	R	D12_SYS_EXPSTATUS	UINT	//expanded status bits
40035	R	D12_SYS_FAULTS	UINT	//see fault bit defs above
40036	R	D12_SYS_STATUS	UINT	//see status bit defs above
40037	R	D12_SYS_CONC	REAL	//unblanked reading in sensor concentration units
40039	R	D12_SYS_CONCPCTFS	REAL	//unblanked reading as percent of full scale
40041	R	D12_SYS_CELSIUS	REAL	//temperature reading
40043	R	D12_SYS_CONCBL	REAL	//reading in sensor concentration units-blanked around zero
40045	R	D12_SYS_CONCPCTFSBL	REAL	//reading as percent of full scale-blanked around zero
40047	R	D12_SYS_LOOPMA	REAL	//current loop output derived from gas concentration
40049	R	D12_SYSLOOPFIXEDMA	REAL	//loop fixed at this value when greater than 0.0
40051	R	D12_SYS_LOOPPWM	UINT	//10 bit DAC value used to derive analog output
40052	R	D12_SYS_LOOPPWM_12BIT	UINT	//12 bit value representing concentration: 0=fault,300=inhibit,800=0%FS, 4000=100%FS
40053	R	D12_SYS_ADC0_RAW	LONG	//unfiltered adc reading for the concentration input
40055	R	D12_SYS_ADC0_LIGHT	LONG	//lightly filtered adc reading for the concentration input
40057	R	D12_SYS_ADC0_HEAVY	LONG	//heavily filtered adc reading for the concentration input
40059	R	D12_SYS_ADC0_DTC	REAL	//damping constant for concentration ADC
40061	R	D12_SYS_ADC1_RAW	LONG	//unfiltered adc reading for the temperature input
40063	R	D12_SYS_ADC1_LIGHT	LONG	//lightly filtered adc reading for the temperature input
40065	R	D12_SYS_ADC1_HEAVY	LONG	//heavily filtered adc reading for the temperature input
40067	R	D12_SYS_ADC1_DTC	REAL	//damping constant for temperature ADC
40069	R	D12_SYS_CONCINP_ADCOFFSET	LONG	//amplifier offset used during runtime
40071	R	D12_SYS_CONCINP_ADCGAIN	REAL	//amplifier gain used during runtime
40073	R	D12_SYS_CONCINP_RANGE	REAL	//concentration representing full scale
40075	R	D12_SYS_CONCINP_RNGFACT	REAL	//current sensor range divided by minimum sensor range
40077	R	D12_SYS_CONCINP_PWRUPGOTO	REAL	//output level during alarm inhibit, loop lock
40079	R	D12_SYS_CONCINP_ADCZERO	REAL	//ADC reading representing the zero input signal
40081	R	D12_SYS_CONCINP_ADCSPAN	REAL	//ADC reading representing the full scale
40083	R	D12_SYS_CONCINP_CORRADC	REAL	//corrected ADC reading...can be gain scaled and saved in smart sensor
40085	R	D12_SYS_CONCINP_BLANKING	REAL	//blank +/-X%fs around zero
40088	R	D12_SYS_CONCINP_FUTURE	UINT	//Modbus real register alignment, future use
40089	*R*	*D12_SYS_INTERNALADC0*	*INT*	*//D12 only: analog key-switch input*
40089	*R*	*F12_SYS_INTERNALADC0*	*INT*	*//F12 only: sensor reference voltage*
40090	*R*	*D12_SYS_INTERNALADC1*	*INT*	*//D12 only: temperature analog input*
40090	*R*	*F12_SYS_INTERNALADC1*	*INT*	*//F12 only: n/u*
40091	*R*	*D12_SYS_INTERNALADC2*	*INT*	*//D12 only: factory calibration jumper: OFF>512, ON<512*
40091	*R*	*F12_SYS_INTERNALADC2*	*INT*	*//F12 only: n/u*
40092	*R*	*D12_SYS_INTERNALADC3*	*INT*	*//D12 only: relay option jumper: OFF>512, ON<512*
40092	*R*	*F12_SYS_INTERNALADC3*	*INT*	*//F12 only: n/u*
40093	*R*	*D12_SYS_INTERNALADC4*	*INT*	*//D12 only: OFF>512, ON<512*
40093	*R*	*F12_SYS_INTERNALADC4*	*INT*	*//F12 only: factory calibration jumper; OFF>512, ON<512*
40094	*R*	*D12_SYS_INTERNALADC5*	*INT*	*//D12 only: OFF>512, ON<512*

40094	*R*	*F12_SYS_INTERNALADC5*	*INT*	*//F12 only: relay option jumper: OFF>512, ON<512*
40095	*R*	*D12_SYS_INTERNALADC6*	*INT*	*//D12 only: relay voltage input PRESENT>512, MISSING<512*
40095	*R*	*F12_SYS_INTERNALADC6*	*INT*	*//F12 only: remote alarm reset INACTIVE>512, ACTIVE<512*
40096	*R*	*D12_SYS_INTERNALADC7*	*INT*	*//D12 only: always 0*
40096	*R*	*F12_SYS_INTERNALADC7*	*INT*	*//F12 only: n/u*
40097	R	D12_SYS_USECTIMER	UINT	//0.2s resolution, not autoreset
40098	R	D12_SYS_FSECTIMER	UINT	//0.2s resolution, not autoreset
40099	R	D12_SYS_POWERONTIMER	UINT	//0.2s resolution, not autoreset
40100	R	D12_SYS_HOLDRDGTIMER	UINT	//0.2s resolution, not autoreset
40101	R	D12_SYS_INHIBITTIMER	UINT	//0.2s resolution, not autoreset
40102	R	D12_SYS_BLINKONTIMER	UINT	//0.2s resolution, autoreset
40103	R	D12_SYS_BLINKOFFTIMER	UINT	//0.2s resolution, autoreset
40104	R	D12_SYS_APPTIMER	UINT	//0.2s resolution, can be used by the application
40105	R	D12_SYS_AUTOTESTTIMER	UINT	//0.2s resolution, used for auto test
40106	R	D12_SYS_ATINTVTIMER	UINT	//1 minute resolution, used to time average autotest
40107	R	D12_SYS_AT_WAITS_FAILS	UINT	//number of autotest failures (no gas sensed) and autoTest waits (conc above 10%FS at start)
40108	R	D12_SYS_LOOPCONTROL_TIMERSTOP	UINT	//current loop control bits and timer control bits
40109	R	D12_SYS_AUTOTSTATE_DRVRIMAGE	UINT	//[MSB]used during auto test function //[LSB]image written out to comm and relay driver chips
40110	R	D12_SYS_RLYTESTEN_IMAGE	UINT	//[MSB]0=disable, 1=enable //[LSB]binary image: 0000 FAWC
40111	R	D12_SYS_CLOCK_MIN_HR	UINT	//[MSB]0..59 minute//[LSB]0..23 hour
40112	R	D12_SYS_CLOCK_DAY_SEC	UINT	//[MSB]1..7 day of the week //[LSB]0..59 second
40113	R	D12_SYS_CLOCK_MO_DATE	UINT	//[MSB]1..12 month //[LSB] 1..31 date
40114	R	D12_SYS_CLOCK_YEAR	INT	//0000..9999 binary
40115	R	D12_SYS_KEY_LAST_CURRENT	UINT	//[MSB]key code read last time//[LSB]current key code
40116	R	D12_SYS_KEY_CHGTO_HOLD	UINT	//[MSB]detects single key change//[LSB]counts current key code
40117	R	D12_SYS_KEY_HELDFOR_CHGFROM	UINT	//[MSB]how long key was held before change//[LSB]code of key before change
40118	R	D12_SYS_KEY_UPDN	INT	//0,1,10,-1,-10
40119	R	D12_SYS_ATPASSCONC	REAL	//concentration pass level
40161	R	D12_SYSSETUP_USERLOCK	UINT	//[MSB] =0 //[LSB] 0=off, 1=on
40162	R	D12_SYSSETUP_IHBTMOMINS	UINT	//inhibit time out period in minutes - updates sys.inhibitTimer
40163	R	D12_SYSSETUP_USERPSWD	UINT	//0-9999, 0 is default
40164	R	D12_SYSSETUP_LCDCONTRAST	UINT	//% contrast value
40165	R	D12_SYSSETUP_R2CFG_R1CFG	UINT	//[MSB]relay 2 cfg //[LSB]relay 1 cfg [BIT1,BIT0 is relay assignment 0-3; BIT4 is failsafe bit]
40166	R	D12_SYSSETUP_R3CFG	UINT	//[MSB]=back light enable; //[LSB] relay 3 cfg [BIT1,BIT0 is relay assignment 0-3; BIT4 is failsafe bit]
40167	R	D12_SYSSETUP_LOOP_DAC04	INT	//PWM duty cycle value representing exactly 4.00mA
40168	R	D12_SYSSETUP_LOOP_DAC20	INT	//PWM duty cycle value representing exactly 20.0mA
40169	R	D12_SYSSETUP_LOOP_ATMA	REAL	//mA output value during autotest
40171	R	D12_SYSSETUP_LOOP_INHIBMA	REAL	//mA output value during alarm inhibit
40173	R	D12_SYSSETUP_LOOP_FAILMA	REAL	//mA output value during trouble

40175	R	D12_SYSSETUP_AT_STATE	UINT	//[MSB] autotest state 0=OFF, 1=ON //[LSB] gas generator on time in seconds
40176	R	D12_SYSSETUP_AT_INTERVAL	INT	//number of days between tests
40177	R	D12_SYSSETUP_AT_MIN_HR	UINT	//[MSB] minute of next autotest, 0..59//[LSB] hour of next autotest, 0..23
40178	R	D12_SYSSETUP_AT_DAY_SEC	UINT	//[MSB] day of next autotest, 1..7//[LSB] second of next autotest, 0..59
40179	R	D12_SYSSETUP_AT_MO_DATE	UINT	//[MSB] month of next autotest, 1..12//[LSB] date of next autotest, 1..31
40180	R	D12_SYSSETUP_AT_YEAR	INT	//year of next autotest, 0000..9999
40181	R	D12_SYSSETUP_AT_PASSVAL	REAL	//amount by which %FS conc must change in order to pass self test
40183	R	D12_SYSSETUP_LO_OFFSET	LONG	//ADC value obtained with 0.00000v applied
40185	R	D12_SYSSETUP_LO_GAIN	REAL	//measured gain of low gain amplifier (nominally 1)
40187	R	D12_SYSSETUP_HI_OFFSET	LONG	//ADC value obtained with 0.00000v applied
40189	R	D12_SYSSETUP_HI_GAIN	REAL	//measured gain of high gain amplifier (nominally 23)
40191	R	D12_SYSSETUP_FUTURE1	UINT	//future use
40193	R	D12_SYSSETUP_FUTURE2	UINT	//future use
40195	R	D12_SYSSETUP_UPKEYADC	INT	//upper adc limits for UP key
40196	R	D12_SYSSETUP_DNKEYADC	INT	//upper adc limits for DOWN key
40197	R	D12_SYSSETUP_ESCKEYADC	INT	//upper adc limits for ESCAPE key
40198	R	D12_SYSSETUP_ENTKEYADC	INT	//upper adc limits for ENTER key
40199	R	D12_SYSSETUP_ALTITUDE	INT	//altitude in feet, -1000 to +9999, def=0, required for DO altitude in feet (DO version)
40200	R	D12_SYSSETUP_PRESSURE	INT	//pressure in mm of Hg, 500 to 850, def=760(sea level), required for DO
40201	R	D12_SYSSETUP_RELATIVEHUM	INT	//%RH, 0 to 100, def=50, required for DO
40202	R	D12_SYSSETUP_SALINITY	UINT	//concentration in PPM, 0 to 60,000, def=0, required for DO
40203	R	D12_SYSSETUP_FUTURE3	UINT	//future use
40205	R	D12_SYSSETUP_INTERF_PROTO	UINT	//[MSB] 0=NONE,1=RS232,2=RS485,3=BELL202//[LSB] 0=NONE,1=ASCII,2=HART,3=MODBUS
40206	R	D12_SYSSETUP_BAUD_ADDR	UINT	//[MSB] baud rate index: 0=150,1=300,2=600,3=1200,4=2400,5=4800,6=9600,7=14.4k,8=28.8k //[LSB] slave address 0-255
40207	R	D12_SYSSETUP_GAP_PARITY	UINT	//[MSB] gap counter //[LSB] 0=NONE,1=ODD,2=EVEN
40241	R	D12_ALARM_STATE	UINT	//[MSB]=0000FAWC bits, 0=no chg, 1=chg//[LSB] = 0000FAWC bits, 0=inactive, 1=active
40242	R	D12_ALARM_C_SETTIMER	UINT	//timer used for caution alarm set delay
40243	R	D12_ALARM_W_SETTIMER	UINT	//timer used for warning alarm set delay
40244	R	D12_ALARM_A_SETTIMER	UINT	//timer used for highest alarm set delay
40245	R	D12_ALARM_C_RESETTIMER	UINT	//timer used for caution alarm reset delay
40246	R	D12_ALARM_W_RESETTIMER	UINT	//timer used for warning alarm reset delay
40247	R	D12_ALARM_A_RESETTIMER	UINT	//timer used for highest alarm reset delay
40248	R	D12_ALARM_C_MIN_HR	UINT	//[MSB] minute of last caution alarm, 0..59//[LSB] hour of last caution alarm, 0..23
40249	R	D12_ALARM_C_DAY_SEC	UINT	//[MSB] day of last caution alarm, 1..7 //[LSB] second of last caution alarm, 0..59
40250	R	D12_ALARM_C_MO_DATE	UINT	//[MSB] month of last caution alarm, 1..12//[LSB] date of last caution alarm, 1..31
40251	R	D12_ALARM_C_YEAR	INT	//year of last caution alarm, 0000..9999
40252	R	D12_ALARM_W_MIN_HR	UINT	//[MSB] minute of last warning alarm, 0..59//[LSB] hour of last warning alarm, 0..23

40253	R	D12_ALARM_W_DAY_SEC	UINT	//[MSB] day of last warning alarm, 1..7//[LSB] second of last warning alarm, 0..59
40254	R	D12_ALARM_W_MO_DATE	UINT	//[MSB] month of last warning alarm, 1..12//[LSB] date of last warning alarm, 1..31
40255	R	D12_ALARM_W_YEAR	INT	//year of last warning alarm, 0000..9999
40256	R	D12_ALARM_A_MIN_HR	UINT	//[MSB] minute of last highest alarm, 0..59 //[LSB] hour of last highest alarm, 0..23
40257	R	D12_ALARM_A_DAY_SEC	UINT	//[MSB] day of last highest alarm, 1..7//[LSB] second of last highest alarm, 0..59
40258	R	D12_ALARM_A_MO_DATE	UINT	//[MSB] month of last highest alarm, 1..12 //[LSB] date of last highest alarm, 1..31
40259	R	D12_ALARM_A_YEAR	INT	//year of last highest alarm, 0000..9999
40260	R	D12_ALARM_RESET	UINT	//[MSB]=0//[LSB] true when alarm acknowledged
40273	R	D12_ALMSETUP_C_SETPOINT	REAL	//caution alarm set point (in sensor concentration units)
40275	R	D12_ALMSETUP_W_SETPOINT	REAL	//warning alarm set point (in sensor concentration units)
40277	R	D12_ALMSETUP_A_SETPOINT	REAL	//highest alarm set point (in sensor concentration units)
40279	R	D12_ALMSETUP_C_RESPOINT	REAL	//caution alarm reset point (in sensor concentration units)
40281	R	D12_ALMSETUP_W_RESPOINT	REAL	//caution alarm reset point (in sensor concentration units)
40283	R	D12_ALMSETUP_A_RESPOINT	REAL	//caution alarm reset point (in sensor concentration units)
40285	R	D12_ALMSETUP_C_SETDELAY	UINT	//caution alarm set delay period in seconds
40286	R	D12_ALMSETUP_W_SETDELAY	UINT	//warning alarm set delay period in seconds
40287	R	D12_ALMSETUP_A_SETDELAY	UINT	//highest alarm set delay period in seconds
40288	R	D12_ALMSETUP_C_RESDELAY	UINT	//caution alarm reset delay period in seconds
40289	R	D12_ALMSETUP_W_RESDELAY	UINT	//warning alarm reset delay period in seconds
40290	R	D12_ALMSETUP_A_RESDELAY	UINT	//highest alarm reset delay period in seconds
40291	R	D12_ALMSETUP_C_OPTIONS	UINT	//bits=00000000000rffdd where, r=autoreset, ff=faultoverride,dd=direction
40292	R	D12_ALMSETUP_W_OPTIONS	UINT	//bits=00000000000rffdd where, r=autoreset, ff=faultoverride,dd=direction
40293	R	D12_ALMSETUP_A_OPTIONS	UINT	//bits=00000000000rffdd where, r=autoreset, ff=faultoverride,dd=direction
40321	R	D12_OP_PAGEINFO	UINT	//[MSB] page number,cursor,action,page stack pointer[LSB]
40323 [12]	R	D12_OP_PAGESTACK	UINT	//page stack
40335 [4]	R	D12_OP_EDITCLOCK	UINT	//used for editing time/date values
40339 [16]	R	D12_OP_EDITDATA	UINT	//used for editing various parameter values
40355 [5]	R	D12_OP_STOPROWS	UINT	//cursor stop row stack
40360 [5]	R	D12_OP_STOPCOLS	UINT	//cursor stop column stack
40365	R	D12_OP_CSTOPS_FLAGS	UINT	//cursor stops; flags
40385	R	D12_OP_DATEFORMAT	UINT	//[MSB]=0//[LSB] date format: 0=MM/DD/YYYY,1=DD MMM YY

40393	R	D12_SMARTS_RANGE	REAL	//currently programmed range
40395	R	D12_SMARTS_ZEROADC	REAL	//zero adc value
40397	R	D12_SMARTS_SPANADC	REAL	//span adc value
40399	R	D12_SMARTS_AVERAGING	REAL	//number of samples used to form average
40401	R	D12_SMARTS_BLANKING	REAL	//concentration below this value is forced to zero
40403	R	D12_SMARTS_CAUTIONSETPOINT	REAL	//caution alarm(FS ratio)
40405	R	D12_SMARTS_WARNINGSETPOINT	REAL	//warning alarm (FS ratio)
40407	R	D12_SMARTS_ALARMSETPOINT	REAL	//highest alarm (FS ratio)
40409 [2]	R	D12_SMARTS_CWAFLAGS	UINT	//alarm direction flags (0=aboveSP, 6= OFF, B=below SP) [0]=c,[1]=w,[2]=a,[3]=cs
40411 [2]	R	D12_SMARTS_DISPLAYFLAGS	UINT	//display flags [0]=#of decimal places to display,[1,2,3]=0
40413	R	D12_SMARTS_POWERUPDELAY	REAL	//number of seconds to inhibit on power up
40415	R	D12_SMARTS_ATPASSES	REAL	//number of successful autotests
40417	R	D12_SMARTS_ATFAILURES	REAL	//number of unsuccessful autotests
40419	R	D12_SMARTS_ATRETRIES	REAL	//number of autotest retries
40421	R	D12_SMARTS_CALINDEX	UINT	//[0]=no. of zero cals,[1]=no. of span cals,[2]=0,[3]=cs
40423	R	D12_SMARTS_TMPOFSCAL	REAL	//=24-bit ADC reading@25°C (approx.690mV w/2.5vref)
40425	R	D12_SMARTS_SERIALNUM	REAL	//serial number of smart sensor
40427	R	D12_SMARTS_HWREVLEVEL	REAL	//sensor hardware rev level
40429	R	D12_SMARTS_SWREVLEVEL	REAL	//sensor software rev level
40431	R	D12_SMARTS_GASNUM	REAL	//sensor gas species
40433 [8]	R	D12_SMARTS_GASNAME	STRING	//14 characters, 1 null, 1 checksum
40441 [4]	R	D12_SMARTS_GASUNITS	STRING	//6 characters, 1 null, 1 checksum
40445 [2]	R	D12_SMARTS_CS5522CFG	UINT	//adc configuration word
40447	R	D12_SMARTS_PWRUPGOTO	REAL	//transmitter output is forced to this value at power on
40449	R	D12_SMARTS_PARTNUM	REAL	//sensor part number
40451	R	D12_SMARTS_RANGEMIN	REAL	//minimum programmable sensor range
40453	R	D12_SMARTS_RANGEMAX	REAL	//maximum programmable sensor range
40455	R	D12_SMARTS_RANGEDEF	REAL	//default sensor range
40457	R	D12_SMARTS_ZEROADCMIN	REAL	//minimum adc value for zero
40459	R	D12_SMARTS_ZEROADCMAX	REAL	//maximum adc value for zero
40461	R	D12_SMARTS_ZEROADCDEF	REAL	//default adc value for zero
40463	R	D12_SMARTS_SPANADCMIN	REAL	//minimum adc value for span
40465	R	D12_SMARTS_SPANADCMAX	REAL	//maximum adc value for span
40467	R	D12_SMARTS_SPANADCDEF	REAL	//default adc value for span
40469	R	D12_SMARTS_AVERAGINGMIN	REAL	//minimum value for averaging
40471	R	D12_SMARTS_AVERAGINGMAX	REAL	//maximum value for averaging
40473	R	D12_SMARTS_AVERAGINGDEF	REAL	//default averaging value
40475	R	D12_SMARTS_BLANKINGMIN	REAL	//minimum value for blanking (FS ratio)
40477	R	D12_SMARTS_BLANKINGMAX	REAL	//maximum value for blanking (FS ratio)
40479	R	D12_SMARTS_BLANKINGDEF	REAL	//default blanking (FS ratio)
40481	R	D12_SMARTS_CAUTIONSPMIN	REAL	//minimum value for caution alarm (FS ratio)
40483	R	D12_SMARTS_WARNINGSPMIN	REAL	//maximum value for caution alarm (FS ratio)

40485	R	D12_SMARTS_ALARMSPMIN	REAL	//default caution alarm (FS ratio)
40487	R	D12_SMARTS_CAUTIONSPMAX	REAL	//minimum value for warning alarm (FS ratio)
40489	R	D12_SMARTS_WARNINGSPMAX	REAL	//maximum value for warning alarm (FS ratio)
40491	R	D12_SMARTS_ALARMSPMAX	REAL	//default warning alarm (FS ratio)
40493	R	D12_SMARTS_CAUTIONSPMAX	REAL	//minimum value for highest alarm (FS ratio)
40495	R	D12_SMARTS_WARNINGSPMAX	REAL	//maximum value for highest alarm (FS ratio)
40497	R	D12_SMARTS_ALARMSPMAX	REAL	//default highest alarm (FS ratio)
40499 [2]	R	D12_SMARTS_CWAFLAGSDEF	UINT	//default alarm direction flags (0=above SP, 6= OFF, B=below SP) [0]=c,[1]=w,[2]=a,[3]=cs
40501	R	D12_SMARTS_PWRUPDELAYMIN	REAL	//minimum value for power on delay in seconds
40503	R	D12_SMARTS_PWRUPDELAYMAX	REAL	//maximum value for power on delay in seconds
40505	R	D12_SMARTS_PWRUPDELAYDEF	REAL	//default power on delay in seconds
40553	R	D12_SMARTG_GASNUM	REAL	//type of gas generated
40555	R	D12_SMARTG_MAHRRATING	REAL	//milliamp-hour rating of gas generator
40557	R	D12_SMARTG_USEDMAHOURS	REAL	//milliamp hours used by generator
40559	R	D12_SMARTG_AVGINTERVAL	REAL	//average hours between autotest intervals
40561 [8]	R	D12_SMARTG_GASNAME	STRING	//16 character generator gas name string
40569 [4]	R	D12_SMARTG_GASUNITS	STRING	//8 character generator gas concentration units (ie, PPM,PPB,%,%LEL…)
40573	R	D12_SMARTG_OPERATINGMA	REAL	//current required by generator (milliamps)
40593	R	D12_DLOG_STATE_ORGINDEX	UINT	//[MSB] current state of data logger//[LSB] index to current sample rate
40594	R	D12_DLOG_NUMDAYS	INT	//total number of days stored so far
40595	R	D12_DLOG_NUMSAMPLES	INT	//testing only, not really required.
40596	R	D12_DLOG_DAY	INT	//current day index where samples are being stored
40597	R	D12_DLOG_CLOCK_MIN_HR	UINT	//[MSB] minute of last sample//[LSB] hour of last sample
40598	R	D12_DLOG_CLOCK_DAY_SEC	UINT	//[MSB] day of last sample//[LSB] second of last sample
40599	R	D12_DLOG_CLOCK_MO_DATE	UINT	//[MSB] month of last sample//[LSB] date of last sample
40600	R	D12_DLOG_CLOCK_YEAR	INT	//year of last sample
40601	R	D12_DLOG_CHECKSUM	UINT	//[MSB] checksum//[LSB]=0
40625 [32]	R	D12_LCD_CHARBUFFER_64	STRING	//lcd display text buffer
40657 [32]	R	D12_LCD_ATTRIBUTEBUFFER_64	UINT	//lcd display text attribute buffer
40689 [192]	R	D12_LCD_IMAGEBUFFER_384	UINT	//buffer for binary LCD dot data
40881	R	D12_LCD_SCROLLDIV_POSITION	UINT	//[MSB] scroll rate divider (=2)//[LSB] current position in scroll string
40883	R	D12_MEM_HWREVLEVEL	REAL	//revision level of transmitter hardware
40885	R	D12_MEM_SWREVLEVEL	REAL	//revision level of transmitter software

4.2 D12 with IR Sensor

This table applies only to the D12 transmitter with the IR sensor, and not to any version of the F12 transmitter.

Register	Access	Tag Name	Type	Comments
40001	RW	D12_MB_SUB	UINT	//used for subroutine calls
40002	RW	D12_MB_ERROR	UINT	//used for returning transmitter specific error codes
40003	RW	D12_MB_IDATA0	UINT	//used for passing integer data to subroutines
40004	RW	D12_MB_IDATA1	UINT	//used for passing integer data to subroutines
40005	RW	D12_MB_IDATA2	UINT	//used for passing integer data to subroutines
40006	RW	D12_MB_IDATA3	UINT	//used for passing integer data to subroutines
40003	RW	D12_MB_UDATA0	UINT	//used for passing unsigned data to subroutines
40004	RW	D12_MB_UDATA1	UINT	//used for passing unsigned data to subroutines
40005	RW	D12_MB_UDATA2	UINT	//used for passing unsigned data to subroutines
40006	RW	D12_MB_UDATA3	UINT	//used for passing unsigned data to subroutines
40003	RW	D12_MB_LDATA0	REAL	//used for passing long data to subroutines
40005	RW	D12_MB_LDATA1	REAL	//used for passing long data to subroutines
40003	RW	D12_MB_RDATA0	REAL	//used for passing float data to subroutines
40005	RW	D12_MB_RDATA1	REAL	//used for passing float data to subroutines
40007	RW	D12_MB_IRET0	INT	//integer data return register
40008	RW	D12_MB_IRET1	INT	//integer data return register
40009	RW	D12_MB_IRET2	INT	//integer data return register
40010	RW	D12_MB_IRET3	INT	//integer data return register
40011	RW	D12_MB_IRET4	INT	//integer data return register
40012	RW	D12_MB_IRET5	INT	//integer data return register
40013	RW	D12_MB_IRET6	INT	//integer data return register
40014	RW	D12_MB_IRET7	INT	//integer data return register
40007	RW	D12_MB_URET0	UINT	//unsigned integer data return register
40008	RW	D12_MB_URET1	UINT	//unsigned integer data return register
40009	RW	D12_MB_URET2	UINT	//unsigned integer data return register
40010	RW	D12_MB_URET3	UINT	//unsigned integer data return register
40011	RW	D12_MB_URET4	UINT	//unsigned integer data return register
40012	RW	D12_MB_URET5	UINT	//unsigned integer data return register
40013	RW	D12_MB_URET6	UINT	//unsigned integer data return register
40014	RW	D12_MB_URET7	UINT	//unsigned integer data return register
40007	RW	D12_MB_RRET0	REAL	//real data return register
40009	RW	D12_MB_RRET1	REAL	//real data return register
40011	RW	D12_MB_RRET2	REAL	//real data return register
40013	RW	D12_MB_RRET3	REAL	//real data return register
40015	RW	D12_MB_LISTENONLY	UINT	//places transmitter into quiet mode
40033	R	D12_SYS_EXPFAULTS	UINT	//expanded fault bits

40034	R	D12_SYS_EXPSTATUS	UINT	//expanded status bits
40035	R	D12_SYS_FAULTS	UINT	//see fault bit defs above
40036	R	D12_SYS_STATUS	UINT	//see status bit defs above
40037	R	D12_SYS_CONC	REAL	//unblanked reading in sensor concentration units
40039	R	D12_SYS_CONCPCTFS	REAL	//unblanked reading as percent of full scale
40041	R	D12_SYS_CELSIUS	REAL	//temperature reading
40043	R	D12_SYS_CONCBL	REAL	//reading in sensor concentration units-blanked around zero
40045	R	D12_SYS_CONCPCTFSBL	REAL	//reading as percent of full scale-blanked around zero
40047	R	D12_SYS_LOOPMA	REAL	//current loop output derived from gas concentration
40049	R	D12_SYSLOOPFIXEDMA	REAL	//loop fixed at this value when greater than 0.0
40051	R	D12_SYS_LOOPPWM	UINT	//10 bit DAC value used to derive analog output
40052	R	D12_SYS_LOOPPWM_12BIT	UINT	//12 bit value representing concentration: 0=fault,300=inhibit,800=0%FS, 4000=100%FS
40053	R	D12_SYS_ADC0_RAW	LONG	//unfiltered pk-pk adc reading for the ref signal input
40055	R	D12_SYS_ADC0_LIGHT	LONG	//lightly filtered pk-pk adc reading for the ref signal input
40057	R	D12_SYS_ADC0_HEAVY	LONG	//heavily filtered pk-pk adc reading for the ref signal input
40059	R	D12_SYS_ADC0_DTC	REAL	//damping constant for the ref signal input
40061	R	D12_SYS_ADC1_RAW	LONG	//unfiltered pk-pk adc reading for the det signal input
40063	R	D12_SYS_ADC1_LIGHT	LONG	//lightly filtered pk-pk adc reading for the det signal input
40065	R	D12_SYS_ADC1_HEAVY	LONG	//heavily filtered pk-pk adc reading for the det signal input
40067	R	D12_SYS_ADC1_DTC	REAL	//damping constant for the det signal input
40083	R	D12_SYS_CONCINP_NORMABS	REAL	//normalized, zero t/c absorption reading
40089	R	D12_SYS_INTERNALADC0	INT	//analog key-switch input
40090	R	D12_SYS_INTERNALADC1	INT	//temperature analog input
40091	R	D12_SYS_INTERNALADC2	INT	//factory calibration jumper:OFF=val>512, ON=val<512
40092	R	D12_SYS_INTERNALADC3	INT	//relay option jumper:OFF=val>512, ON=val<512
40093	R	D12_SYS_INTERNALADC4	INT	//OFF=val>512, ON=val<512
40094	R	D12_SYS_INTERNALADC5	INT	//OFF=val>512, ON=val<512
40095	R	D12_SYS_INTERNALADC6	INT	//relay voltage input PRESENT=val>512, MISSING=val<512
40096	R	D12_SYS_INTERNALADC7	INT	//always 0
40097	R	D12_SYS_USECTIMER	UINT	//0.2s resolution, not autoreset
40098	R	D12_SYS_FSECTIMER	UINT	//0.2s resolution, not autoreset
40099	R	D12_SYS_POWERONTIMER	UINT	//0.2s resolution, not autoreset
40100	R	D12_SYS_HOLDRDGTIMER	UINT	//0.2s resolution, not autoreset

40101	R	D12_SYS_INHIBITTIMER	UINT	//0.2s resolution, not autoreset
40102	R	D12_SYS_BLINKONTIMER	UINT	//0.2s resolution, autoreset
40103	R	D12_SYS_BLINKOFFTIMER	UINT	//0.2s resolution, autoreset
40104	R	D12_SYS_APPTIMER	UINT	//0.2s resolution, can be used by the application
40105	R	D12_SYS_AUTOTESTTIMER	UINT	//0.2s resolution, used for auto test [n/a for most IR sensors]
40106	R	D12_SYS_ATINTVTIMER	UINT	//1 minute resolution, times average autotest [n/a for most IR sensors]
40107	R	D12_SYS_AT_WAITS_FAILS	UINT	//number of autotest failures (no gas sensed) and autoTest waits (conc above 10%FS at start) [n/a for most IR sensors]
40108	R	D12_SYS_LOOPCONTROL_TIMERSTOP	UINT	//current loop control bits and timer control bits
40109	R	D12_SYS_AUTOTSTATE_DRVRIMAGE	UINT	//[MSB]used during auto test function //[LSB]image written out to comm and relay driver chips
40110	R	D12_SYS_RLYTESTEN_IMAGE	UINT	//[MSB]0=disable, 1=enable //[LSB]binary image: 0000 FAWC
40111	R	D12_SYS_CLOCK_MIN_HR	UINT	//[MSB]0..59 minute//[LSB]0..23 hour
40112	R	D12_SYS_CLOCK_DAY_SEC	UINT	//[MSB]1..7 day of the week //[LSB]0..59 second
40113	R	D12_SYS_CLOCK_MO_DATE	UINT	//[MSB]1..12 month //[LSB] 1..31 date
40114	R	D12_SYS_CLOCK_YEAR	INT	//0000..9999 binary
40115	R	D12_SYS_KEY_LAST_CURRENT	UINT	//[MSB]key code read last time//[LSB]current key code
40116	R	D12_SYS_KEY_CHGTO_HOLD	UINT	//[MSB]detects single key change//[LSB]counts current key code
40117	R	D12_SYS_KEY_HELDFOR_CHGFROM	UINT	//[MSB]how long key was held before change//[LSB]code of key before change
40118	R	D12_SYS_KEY_UPDN	INT	//0,1,10,-1,-10
40119	R	D12_SYS_ATPASSCONC	REAL	//concentration pass level [n/a for most IR sensors]
40161	R	D12_SYSSETUP_USERLOCK	UINT	//[MSB] =0 //[LSB] 0=off, 1=on
40162	R	D12_SYSSETUP_IHBTMOMINS	UINT	//inhibit time out period in minutes - updates sys.inhibitTimer
40163	R	D12_SYSSETUP_USERPSWD	UINT	//0-9999, 0 is default
40164	R	D12_SYSSETUP_LCDCONTRAST	UINT	//% contrast value
40165	R	D12_SYSSETUP_R2CFG_R1CFG	UINT	//[MSB]relay 2 cfg //[LSB]relay 1 cfg [BIT1,BIT0 is relay assignment 0-3; BIT4 is failsafe bit]
40166	R	D12_SYSSETUP_R3CFG	UINT	//[MSB]=back light enable; //[LSB] relay 3 cfg [BIT1,BIT0 is relay assignment 0-3; BIT4 is failsafe bit]
40167	R	D12_SYSSETUP_LOOP_DAC04	INT	//PWM duty cycle value representing exactly 4.00mA
40168	R	D12_SYSSETUP_LOOP_DAC20	INT	//PWM duty cycle value representing exactly 20.0mA
40169	R	D12_SYSSETUP_LOOP_ATMA	REAL	//mA output value during autotest
40171	R	D12_SYSSETUP_LOOP_INHIBMA	REAL	//mA output value during alarm inhibit
40173	R	D12_SYSSETUP_LOOP_FAILMA	REAL	//mA output value during trouble
40175	R	D12_SYSSETUP_AT_STATE	UINT	//[MSB] autotest state 0=OFF, 1=ON //[LSB] gas gen on time in seconds [n/a for most IR sensors]
40176	R	D12_SYSSETUP_AT_INTERVAL	INT	//number of days between tests [n/a for most IR sensors]
40177	R	D12_SYSSETUP_AT_MIN_HR	UINT	//[MSB] minute of next autotest, 0..59//[LSB] hour of next autotest, 0..23 [n/a for most IR sensors]
40178	R	D12_SYSSETUP_AT_DAY_SEC	UINT	//[MSB] day of next autotest, 1..7//[LSB] second of next autotest, 0..59 [n/a for most IR sensors]
40179	R	D12_SYSSETUP_AT_MO_DATE	UINT	//[MSB] month of next autotest, 1..12//[LSB] date of next autotest, 1..31 [n/a for most IR sensors]
40180	R	D12_SYSSETUP_AT_YEAR	INT	//year of next autotest, 0000..9999 [n/a for most IR sensors]
40181	R	D12_SYSSETUP_AT_PASSVAL	REAL	//amount by which %FS conc must change in order to pass self test [n/a for most IR sensors]

40185	R	D12_SYSSETUP_SIG_RATIO	REAL	//transmitter gain error
40191	R	D12_SYSSETUP_TMP_YINT	LONG	//thermistor linearization correction constant
40193	R	D12_SYSSETUP_TMP_SLOPE	REAL	//thermistor linearization correction slope
40195	R	D12_SYSSETUP_UPKEYADC	INT	//upper adc limits for UP key
40196	R	D12_SYSSETUP_DNKEYADC	INT	//upper adc limits for DOWN key
40197	R	D12_SYSSETUP_ESCKEYADC	INT	//upper adc limits for ESCAPE key
40198	R	D12_SYSSETUP_ENTKEYADC	INT	//upper adc limits for ENTER key
40203	R	D12_SYSSETUP_DIGPOTSET	UINT	//msb=digital pot setting for det input, lsb=digital pot setting for ref input
40205	R	D12_SYSSETUP_INTERF_PROTO	UINT	//[MSB] 0=NONE,1=RS232,2=RS485,3=BELL202//[LSB] 0=NONE,1=ASCII,2=HART,3=MODBUS
40206	R	D12_SYSSETUP_BAUD_ADDR	UINT	//[MSB] baud rate index: 0=150,1=300,2=600,3=1200,4=2400,5=4800,6=9600,7=14.4k,8=28.8k //[LSB] slave address 0-255
40207	R	D12_SYSSETUP_GAP_PARITY	UINT	//[MSB] gap counter //[LSB] 0=NONE,1=ODD,2=EVEN
40241	R	D12_ALARM_STATE	UINT	//[MSB]=0000FAWC bits, 0=no chg, 1=chg//[LSB] = 0000FAWC bits, 0=inactive, 1=active
40242	R	D12_ALARM_C_SETTIMER	UINT	//timer used for caution alarm set delay
40243	R	D12_ALARM_W_SETTIMER	UINT	//timer used for warning alarm set delay
40244	R	D12_ALARM_A_SETTIMER	UINT	//timer used for highest alarm set delay
40245	R	D12_ALARM_C_RESETTIMER	UINT	//timer used for caution alarm reset delay
40246	R	D12_ALARM_W_RESETTIMER	UINT	//timer used for warning alarm reset delay
40247	R	D12_ALARM_A_RESETTIMER	UINT	//timer used for highest alarm reset delay
40248	R	D12_ALARM_C_MIN_HR	UINT	//[MSB] minute of last caution alarm, 0..59//[LSB] hour of last caution alarm, 0..23
40249	R	D12_ALARM_C_DAY_SEC	UINT	//[MSB] day of last caution alarm, 1..7 //[LSB] second of last caution alarm, 0..59
40250	R	D12_ALARM_C_MO_DATE	UINT	//[MSB] month of last caution alarm, 1..12//[LSB] date of last caution alarm, 1..31
40251	R	D12_ALARM_C_YEAR	INT	//year of last caution alarm, 0000..9999
40252	R	D12_ALARM_W_MIN_HR	UINT	//[MSB] minute of last warning alarm, 0..59//[LSB] hour of last warning alarm, 0..23
40253	R	D12_ALARM_W_DAY_SEC	UINT	//[MSB] day of last warning alarm, 1..7//[LSB] second of last warning alarm, 0..59
40254	R	D12_ALARM_W_MO_DATE	UINT	//[MSB] month of last warning alarm, 1..12//[LSB] date of last warning alarm, 1..31
40255	R	D12_ALARM_W_YEAR	INT	//year of last warning alarm, 0000..9999
40256	R	D12_ALARM_A_MIN_HR	UINT	//[MSB] minute of last highest alarm, 0..59 //[LSB] hour of last highest alarm, 0..23
40257	R	D12_ALARM_A_DAY_SEC	UINT	//[MSB] day of last highest alarm, 1..7//[LSB] second of last highest alarm, 0..59
40258	R	D12_ALARM_A_MO_DATE	UINT	//[MSB] month of last highest alarm, 1..12 //[LSB] date of last highest alarm, 1..31
40259	R	D12_ALARM_A_YEAR	INT	//year of last highest alarm, 0000..9999
40260	R	D12_ALARM_RESET	UINT	//[MSB]=0//[LSB] true when alarm acknowledged

40273	R	D12_ALMSETUP_C_SETPOINT	REAL	//caution alarm set point (in sensor concentration units)
40275	R	D12_ALMSETUP_W_SETPOINT	REAL	//warning alarm set point (in sensor concentration units)
40277	R	D12_ALMSETUP_A_SETPOINT	REAL	//highest alarm set point (in sensor concentration units)
40279	R	D12_ALMSETUP_C_RESPOINT	REAL	//caution alarm reset point (in sensor concentration units)
40281	R	D12_ALMSETUP_W_RESPOINT	REAL	//caution alarm reset point (in sensor concentration units)
40283	R	D12_ALMSETUP_A_RESPOINT	REAL	//caution alarm reset point (in sensor concentration units)
40285	R	D12_ALMSETUP_C_SETDELAY	UINT	//caution alarm set delay period in seconds
40286	R	D12_ALMSETUP_W_SETDELAY	UINT	//warning alarm set delay period in seconds
40287	R	D12_ALMSETUP_A_SETDELAY	UINT	//highest alarm set delay period in seconds
40288	R	D12_ALMSETUP_C_RESDELAY	UINT	//caution alarm reset delay period in seconds
40289	R	D12_ALMSETUP_W_RESDELAY	UINT	//warning alarm reset delay period in seconds
40290	R	D12_ALMSETUP_A_RESDELAY	UINT	//highest alarm reset delay period in seconds
40291	R	D12_ALMSETUP_C_OPTIONS	UINT	//bits=00000000000rffdd where, r=autoreset, ff=faultoverride,dd=direction
40292	R	D12_ALMSETUP_W_OPTIONS	UINT	//bits=00000000000rffdd where, r=autoreset, ff=faultoverride,dd=direction
40293	R	D12_ALMSETUP_A_OPTIONS	UINT	//bits=00000000000rffdd where, r=autoreset, ff=faultoverride,dd=direction
40321	R	D12_OP_PAGEINFO	UINT	//[MSB] page number,cursor,action,page stack pointer[LSB]
40323 [12]	R	D12_OP_PAGESTACK	UINT	//page stack
40335 [4]	R	D12_OP_EDITCLOCK	UINT	//used for editing time/date values
40339 [16]	R	D12_OP_EDITDATA	UINT	//used for editing various parameter values
40355 [5]	R	D12_OP_STOPROWS	UINT	//cursor stop row stack
40360 [5]	R	D12_OP_STOPCOLS	UINT	//cursor stop column stack
40365	R	D12_OP_CSTOPS_FLAGS	UINT	//cursor stops; flags
40385	R	D12_OP_DATEFORMAT	UINT	//[MSB]=0//[LSB] date format: 0=MM/DD/YYYY,1=DD MMM YY
40393	R	D12_IR_RANGE	REAL	//currently programmed range
40395	R	D12_IR_SPANTMPCO	REAL	//linear comp factor for span drift over temperature
40397	R	D12_IR_SPANABS	REAL	//nominal absorption at 100% full scale (20mA)
40399	R	D12_IR_AVERAGING	REAL	//number of samples used to form average
40401	R	D12_IR_BLANKING	REAL	//concentration below this value is forced to zero
40403	R	D12_IR_CAUTIONSETPOINT	REAL	//caution alarm(FS ratio)
40405	R	D12_IR_WARNINGSETPOINT	REAL	//warning alarm (FS ratio)
40407	R	D12_IR_ALARMSETPOINT	REAL	//highest alarm (FS ratio)
40409	R	D12_IR_CWAFLAGS	UINT	//alarm direction flags (0=above SP, 6= OFF, B=below SP) [0]=c,[1]=w,[2]=a,[3]=cs
40411	R	D12_IR_DISPLAYFLAGS	UINT	//display flags [0]=#of decimal places to display,[1,2,3]=0

40413	R	D12_IR_POWERUPDELAY	REAL	//number of seconds to inhibit on power up
40425	R	D12_IR_SERIALNUM	REAL	//serial number of smart sensor
40427	R	D12_IR_HWREVLEVEL	REAL	//sensor hardware rev level
40429	R	D12_IR_SWREVLEVEL	REAL	//sensor software rev level
40431	R	D12_IR_GASNUM	REAL	//sensor gas species
40433	R	D12_IR_GASNAME	STRIN G	//14 characters, 1 null, 1 checksum
40441	R	D12_IR_GASUNITS	STRIN G	//6 characters, 1 null, 1 checksum
40447	R	D12_IR_PWRUPGOTO	REAL	//transmitter output is forced to this value at power on
40449	R	D12_IR_PARTNO	REAL	//part number of sensor
40451	R	D12_IR_RANGEMIN	REAL	//minimum programmable sensor range
40453	R	D12_IR_RANGEMAX	REAL	//maximum programmable sensor range
40455	R	D12_IR_RANGEDEF	REAL	//default sensor range
40457	R	D12_IR_LEL	REAL	//lower explosive limits for current gas selection in %V/Air
40459	R	D12_IR_UEL	REAL	//upper explosive limits for current gas selection in %V/Air
40461	R	D12_IR_CALINFO	UINT	//MSB=group,status,span gas,checksum
40463	R	D12_IR_SPANCONC	REAL	//%V/V of gas used at last span
40465	R	D12_IR_SPANTEMP	REAL	//temperature at last span
40467	R	D12_IR_SPANFACTOR	REAL	//factor calculated during span calibration
40469	R	D12_IR_AVERAGINGMIN	REAL	//minimum value for averaging
40471	R	D12_IR_AVERAGINGMAX	REAL	//maximum value for averaging
40473	R	D12_IR_AVERAGINGDEF	REAL	//default averaging value
40475	R	D12_IR_BLANKINGMIN	REAL	//minimum value for blanking (FS ratio)
40477	R	D12_IR_BLANKINGMAX	REAL	//maximum value for blanking (FS ratio)
40479	R	D12_IR_BLANKINGDEF	REAL	//default blanking (FS ratio)
40481	R	D12_IR_CAUTIONSPMIN	REAL	//minimum value for caution alarm (FS ratio)
40483	R	D12_IR_WARNINGSPMIN	REAL	//maximum value for caution alarm (FS ratio)
40485	R	D12_IR_ALARMSPMIN	REAL	//default caution alarm (FS ratio)
40487	R	D12_IR_CAUTIONSPMAX	REAL	//minimum value for warning alarm (FS ratio)
40489	R	D12_IR_WARNINGSPMAX	REAL	//maximum value for warning alarm (FS ratio)
40491	R	D12_IR_ALARMSPMAX	REAL	//default warning alarm (FS ratio)
40493	R	D12_IR_CAUTIONSPMAX	REAL	//minimum value for highest alarm (FS ratio)
40495	R	D12_IR_WARNINGSPMAX	REAL	//maximum value for highest alarm (FS ratio)
40497	R	D12_IR_ALARMSPMAX	REAL	//default highest alarm (FS ratio)
40499	R	D12_IR_CWAFLAGSDEF	UINT	//default alarm direction flags (0=above SP, 6= OFF, B=below SP) [0]=c,[1]=w,[2]=a,[3]=cs

40501	R	D12_IR_GASLIN_NEGA	REAL	//linearization exponent factor, -a
40503	R	D12_IR_GASLIN_INVB	REAL	//linearization exponent power, 1/b
40505	R	D12_IR_INFO_00	UINT	//msb=number of gasses; lsb=type
40506	R	D12_IR_INFO_01	UINT	//msb=units code;lsb=gas index
40507	R	D12_IR_INFO_02	UINT	//serial number (0-65535)
40508	R	D12_IR_INFO_03	UINT	//msb=number of spans, lsb=number of zeros
40509	R	D12_IR_INFO_04	UINT	//msb=future;lsb=lamp freq
40510	R	D12_IR_INFO_05	UINT	//msb=digital pot setting for det signal; lsb=digital pot setting for ref signal
40511	R	D12_IR_INFO_06	UINT	//msb=adc read delay for det signal;lsb=adc read delay for ref signal
40512	R	D12_IR_INFO_07	UINT	//msb=unassigned; lsb=checksum for previous 15 bytes
40513	R	D12_IR_ZEROCONC	REAL	//%V/V of gas used at last zero
40515	R	D12_IR_ZEROTEMP	REAL	//temperature at last zero
40517	R	D12_IR_REFZERO	REAL	//reference peak-to-peak ADC value stored during zero
40519	R	D12_IR_DETZERO	REAL	//detector peak-to-peak ADC value stored during zero
40521	R	D12_IR_ZEROPOSTMPCO	REAL	//tcomp factor applied to normalized transmissivity when above 25C
40523	R	D12_IR_ZERONEGTMPCO	REAL	//tcomp factor applied to normalized transmissivity when below 25C
40525	R	D12_IR_TMPCORYINT	REAL	//thermistor offset error (y-intercept of error line)
40527	R	D12_IR_TMPCORSLOPE	REAL	//thermistor scale error (slope of error line, computed between TMPCAL0 and TMPCAL1)
40529	R	D12_IR_TMPPOLY3A	REAL	//poly3 coefficients of thermistor circuit (x is 10-bit ADC reading)
40531	R	D12_IR_TMPPOLY3B	REAL	//poly3 coefficients of thermistor circuit (x is 10-bit ADC reading)
40533	R	D12_IR_TMPPOLY3C	REAL	//poly3 coefficients of thermistor circuit (x is 10-bit ADC reading)
40535	R	D12_IR_TMPPOLY3D	REAL	//poly3 coefficients of thermistor circuit (x is 10-bit ADC reading)
40553	R	D12_SMARTG_GASNUM	REAL	//type of gas generated [n/a for most IR sensors]
40555	R	D12_SMARTG_MAHRRATING	REAL	//milliamp-hour rating of gas generator [n/a for most IR sensors]
40557	R	D12_SMARTG_USEDMAHOURS	REAL	//milliamp hours used by generator [n/a for most IR sensors]
40559	R	D12_SMARTG_AVGINTERVAL	REAL	//average hours between autotest intervals [n/a for most IR sensors]
40561[8]	R	D12_SMARTG_GASNAME	STRIN G	//16 character generator gas name string [n/a for most IR sensors]
40569[4]	R	D12_SMARTG_GASUNITS	STRIN G	//8 character generator gas concentration units (ie, PPM,PPB,%,%LEL…) [n/a for most IR sensors]
40573	R	D12_SMARTG_OPERATINGMA	REAL	//current required by generator (milliamps) [n/a for most IR sensors]
40593	R	D12_DLOG_STATE_ORGINDEX	UINT	//[MSB] current state of data logger//[LSB] index to current sample rate
40594	R	D12_DLOG_NUMDAYS	INT	//total number of days stored so far
40595	R	D12_DLOG_NUMSAMPLES	INT	//testing only, not really required.
40596	R	D12_DLOG_DAY	INT	//current day index where samples are being stored
40597	R	D12_DLOG_CLOCK_MIN_HR	UINT	//[MSB] minute of last sample//[LSB] hour of last sample
40598	R	D12_DLOG_CLOCK_DAY_SEC	UINT	//[MSB] day of last sample//[LSB] second of last sample
40599	R	D12_DLOG_CLOCK_MO_DATE	UINT	//[MSB] month of last sample//[LSB] date of last sample

40600	R	D12_DLOG_CLOCK_YEAR	INT	//year of last sample
40601	R	D12_DLOG_CHECKSUM	UINT	//[MSB] checksum//[LSB]=0
40625 [32]	R	D12_LCD_CHARBUFFER_64	STRIN G	//lcd display text buffer
40657 [32]	R	D12_LCD_ATTRIBUTEBUFFER_64	UINT	//lcd display text attribute buffer
40689 [192]	R	D12_LCD_IMAGEBUFFER_384	UINT	//buffer for binary LCD dot data
40881	R	D12_LCD_SCROLLDIV_POSITION	UINT	//[MSB] scroll rate divider (=2)//[LSB] current position in scroll string
40883	R	D12_MEM_HWREVLEVEL	REAL	//revision level of transmitter hardware
40885	R	D12_MEM_SWREVLEVEL	REAL	//revision level of transmitter software

5 Appendix B. Modbus Technical Overview (Brief)

Modbus protocol is a messaging structure, widely used to establish master-slave communication between intelligent devices. A message sent from a master to a slave contains a one-byte slave address, a one-byte command, data bytes (depending on command), and a two byte CRC. The protocol is independent of the underlying physical layer and is traditionally implemented using RS232, RS422, or RS485 over a variety of media (e.g. fiber, radio, cellular, etc.).

The protocol comes in 2 flavors – ASCII and RTU. The formats of messages are identical in both forms, except that the ASCII form transmits each byte of the message as two ASCII hexadecimal characters. Therefore, ASCII messages are twice as long as RTU messages. The main advantage of the RTU mode is that it achieves higher throughput, while the ASCII mode allows time intervals of up to 1 second to occur between characters without causing an error. As stated earlier, the transmitter uses the RTU form and does not support the ASCII form.

The basic structure of an RTU frame is shown below:

[ADDRESS][FUNCTION][DATA][CRC]

The address field of a message frame contains an eight-bit slave device address in the range of 0 ... 247 decimal. The individual slave devices are assigned addresses in the range of 1 ... 247, and address 0 is reserved as a broadcast address. A master addresses a slave by placing the slave address in the address field of the message. When the slave sends its response message, it places its own address in this address field of the response to let the master know which slave is responding. All slaves accept broadcast messages (address 0) as though they were addressed specifically to them, but do not transmit a response message.

The function code field of a message frame contains an eight-bit code in the range of 1 ... 255 decimal. When a query message is sent from the master, the function code field tells the slave device what kind of action to perform. Examples include reading the contents of a group of registers, writing to a single register, writing to a group of registers, and reading the exception status.

When the slave device responds to the master, it uses the function code field to indicate either a normal (error-free) response or that some kind of error occurred (called an exception response). For a normal response, the slave simply echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most significant bit set to logic 1.

The data field is constructed of one or more bytes and contains additional information, which the slave must use to take the action defined by the function code. This can include items like discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field.

If no error occurs, the data field of a response from a slave to a master contains the data requested. If an error occurs, the field contains an exception code that the master application can use to determine the next action to be taken.

The data field can be nonexistent (of zero length) in certain kinds of messages. For example, in a request from a master device for a slave to respond with its communications event log (function code 0B hexadecimal), the slave does not require any additional information. The function code alone specifies the action.

Messages are terminated with a 16-bit CRC value that is computed from all of the bytes of the message. The two byte CRC is superior to just simple checksums because it can help reject more types of errors.

5.1 Registers and Coils

Modbus protocol was originally designed to transfer data to and from PLCs (Programmable Logic Controllers), which organize data into groups of registers and coils. PLC registers containing i/o information are called input registers and are numbered 30001 to 39999, while registers containing data or the results of calculations are known as holding registers and are numbered from 40001 to 49999. The term coils, on the other hand, refers to discrete (0 or 1) inputs and outputs. Traditionally, these are inputs from such things as switch closures and outputs to the coils of relays, which are under the control of the PLC.

All registers are 16 bit values, which may be read or written to individually, or in blocks by using specific functions. Likewise for coils, which are one bit values. Since register functions transfer 16 bits and discrete (coil) functions transfer only one, it is usually more efficient to use register functions, which reduces the number of messages required to transfer data. For this reason, the transmitter organizes all of its data into registers only, or more specifically, data is organized into the holding registers starting at 40001.

The protocol specifies which registers to access by the value of the function code embedded into the message. For example, to read one or more holding registers in a slave device, the master must use function 3 – "Read Holding Register". Similarly, the master must use function 4 – "Read Input Register" to read one or more of the input registers. The same is true for coils, function 1 – "Read Coil Status" reads the status of one or more discrete outputs (bits), and function 2 – "Read Input Status" reads the status of one or more discrete inputs.

For more detailed information on the protocol, please refer to the "Modicon Modbus Protocol Reference Guide" at http://www.modicon.com/techpubs/toc7.html or, "Modbus Protocol Specification", available for download at http://www.modbus-ida.org/specs.php. Deviations from this guide are noted in the appropriate section. More information regarding Modbus, in general, may be viewed at: http://www.modbus-ida.org/

PRODUCT WARRANTY

Analytical Technology, Inc. (Manufacturer) warrants to the Customer that if any part(s) of the Manufacturer's equipment proves to be defective in materials or workmanship within the earlier of 18 months of the date of shipment or 12 months of the date of start-up, such defective parts will be repaired or replaced free of charge. Inspection and repairs to products thought to be defective within the warranty period will be completed at the Manufacturer's facilities in Collegeville, PA. Products on which warranty repairs are required shall be shipped freight prepaid to the Manufacturer. The product(s) will be returned freight prepaid and allowed if it is determined by the manufacturer that the part(s) failed due to defective materials or workmanship.

This warranty does not cover consumable items, batteries, or wear items subject to periodic replacement including lamps and fuses.

Gas sensors carry a 12 months from date of shipment warranty and are subject to inspection for evidence of misuse, abuse, alteration, improper storage, or extended exposure to excessive gas concentrations. Should inspection indicate that sensors have failed due to any of the above, the warranty shall not apply.

The Manufacturer assumes no liability for consequential damages of any kind, and the buyer by acceptance of this equipment will assume all liability for the consequences of its use or misuse by the Customer, his employees, or others. A defect within the meaning of this warranty is any part of any piece of a Manufacturer's product which shall, when such part is capable of being renewed, repaired, or replaced, operate to condemn such piece of equipment.

This warranty is in lieu of all other warranties (including without limiting the generality of the foregoing warranties of merchantability and fitness for a particular purpose), guarantees, obligations or liabilities expressed or implied by the Manufacturer or its representatives and by statute or rule of law.

This warranty is void if the Manufacturer's product(s) has been subject to misuse or abuse, or has not been operated or stored in accordance with instructions, or if the serial number has been removed.

Analytical Technology, Inc. makes no other warranty expressed or implied except as stated above.

WATER QUALITY MONITORS

Dissolved Oxygen
Free Chlorine
Combined Chlorine
Total Chlorine
Residual Chlorine Dioxide
Potassium Permanganate
Dissolved Ozone
pH/ORP
Conductivity
Hydrogen Peroxide
Peracetic Acid
Dissolved Sulfide
Residual Sulfite
Fluoride
Dissolved Ammonia
Turbidity
Suspended Solids
Sludge Blanket Level
MetriNet Distribution Monitor

GAS DETECTION PRODUCTS

NH_3	Ammonia
CO	Carbon Monoxide
H_2	Hydrogen
NO	Nitric Oxide
O_2	Oxygen
CO	Cl2 Phosgene
Br_2	Bromine
Cl_2	Chlorine
ClO_2	Chlorine Dioxide
F_2	Fluorine
I_2	Iodine
H_X	Acid Gases
C_2H_4O	Ethylene Oxide
C_2H_6O	Alcohol
O_3	Ozone
CH_4	Methane (Combustible Gas)
H_2O_2	Hydrogen Peroxide
HCl	Hydrogen Chloride
HCN	Hydrogen Cyanide
HF	Hydrogen Fluoride
H_2S	Hydrogen Sulfide
NO_2	Nitrogen Dioxide
NO_x	Oxides of Nitrogen
SO_2	Sulfur Dioxide
H_2Se	Hydrogen Selenide
B_2H_6	Diborane
GeH_4	Germane
AsH_3	Arsine
PH_3	Phosphine
SiH_4	Silane
HCHO	Formaldehyde
$C_2H_4O_3$	Peracetic Acid
DMA	Dimethylamine