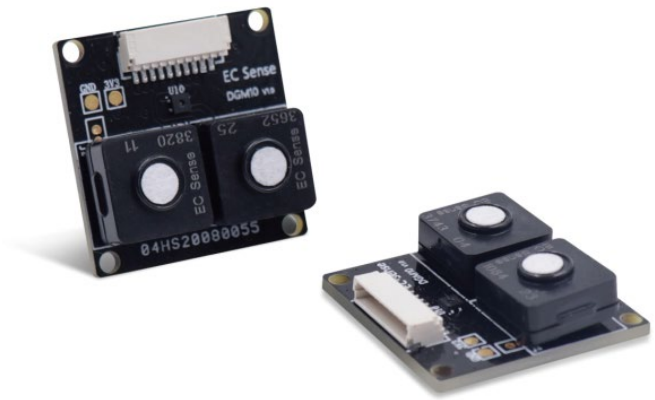
EC Sense
Easy Gas Sensors



DGM10 系列智能双气体传感器模组

UART Modbus-RTU 用户通信协议

I²C用户通讯协议

该文件详细介绍了DGM10双气体传感器模组的UART和I²C接口的通讯协议，用户在使用时主要的命令。

该产品的详细电气参数请查看技术规格书；

产品电气参数和使用方法请查阅产品使用说明书；

1、UART Modbus-RTU 用户通讯协议

1.1 通讯模式

DGM10模组采用UART(TTL)接口通信，通讯参数如下：

信号电压	波特率	数据位	停止位	校验位
3.3V	自适应	8位	1位	无

自适应波特率操作方式：上电5s后，发送0x7F，0x7F，两个字节的数据，随后按照通信协议发送具体指令即可。
(注意：如果上电后不发送0x7F，0x7F两个字节，或者发送的是其他数据，则有可能导致自适应波特率失败，无法正常通讯)

当前支持自适应波特率：115200，57600，19200，9600，2400

通讯协议：Modbus-RTU **出厂默认地址：**01 **数据读写时间间隔要求：**≥1s

本协议全部使用保持寄存器，即只使用三种功能码。

1）03 (0x03) 功能码，读多个保持寄存器

2）06 (0x06) 功能码，写单个保持寄存器

3）16 (0x10) 功能码，写多个保持寄存器

传感器模组采用了主从机问答式数据传输模式，每次数据传输由主机发送指令，从机根据指令传输对应数据。

1.2 修改Modbus地址

当修改传感器模组产品地址时，须单个产品的地址进行逐一修改，不能同时对多个传感器模组地址进行修改。

使用一组(两条)特殊指令修改Modbus地址,两条指令必须依次发送，中间间隔>=1S

指令中的[addr]为需要修改的modbus地址。范围:01(0x01)<=addr<=247(0xF7)，超出范围的值，默认为01(0x01)。

- 第一条指令

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x55	0x4E	0x4C	0x4F	0x43	0x4B	0x32	0x50

- 第二条指令

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]	Byte[8]	Byte[9]
0x4D	0x4F	0x44	0x49	0x46	0x59	0x00	addr	CRC_L	CRC_H

2、传感器类型代码表

可以通过读取保持寄存器中的传感器类型值，来识别当前模组上配置的传感器类型。传感器类型代码表：

气体名称	甲醛	有机挥发物	一氧化碳	氯气	氢气	硫化氢	氯化氢	氰化氢	氟化氢
气体分子式	HCHO	VOC	CO	Cl_2	H_2	H_2S	HCl	HCN	HF
数值	0x17	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F

气体名称	氨气	二氧化氮	氧气	臭氧	二氧化硫	溴化氢	溴气	氟气	磷化氢
气体分子式	NH_3	NO_2	O_2	O_3	SO_2	HBr	Br_2	F_2	PH_3
数值	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	0x28

气体名称	砷化氢	硅烷	锗烷	乙硼烷	三氟化硼	硒化氢	恶臭气体	室内空气质量	室外空气质量
气体分子式	AsH_3	SiH_4	GeH_4	B_2H_6	BF_3	H_2Se	SMELL	IAQ	AQI
数值	0x29	0x2A	0x2B	0x2C	0x2D	0x31	0x32	0x33	0x34

气体名称	非甲烷总烃	硫化物	氮氧化物	一氧化氮	异丁烯	丙二醇	甲硫醇	苯乙烯	四氢噻吩
气体分子式	NMHC	SO_x	NO_x	NO	C_4H_8	$C_3H_8O_2$	CH_4S	C_8H_8	THT
数值	0x35	0x36	0x37	0x38	0x39	0x3A	0x3B	0x3C	0x3D

气体名称	三氯氧磷	光气	环氧乙烷	三甲胺	二甲胺	乙醇	二硫化碳	乙硫醇	二甲二硫醚
气体分子式	$POCl_3$	$COCl_2$	C_2H_4O	C_3H_9N	C_2H_7N	C_2H_6O	CS_2	C_2H_6S	$C_2H_6S_2$
数值	0x3E	0x3F	0x40	0x41	0x42	0x43	0x44	0x45	0x46

气体名称	乙烯	甲醇	苯	对二甲苯	甲苯	乙酸	二氧化氯	过氧化氢	联氨
气体分子式	C_2H_4	CH_3OH	C_6H_6	C_8H_{10}	C_7H_8	CH_3COOH	ClO_2	H_2O_2	N_2H_4
数值	0x47	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F

气体名称	偏二甲肼	三氯乙烯	三氯甲烷	三氯乙烷
气体分子式	$C_2H_8N_2$	C_2HCl_3	$CHCl_3$	$C_2H_3Cl_3$
数值	0x50	0x51	0x52	0x53

2.1 获取传感器模组测量参数

本组地址是用于输出气体浓度，温度，湿度，量程，传感器类型，运行灯开关控制，传感器寿命与性能状态。
以下S0表示：位置编号为S0的气体传感器　　S1表示：位置编号为S1的气体传感器
当传感器模组地址设置不正确时无法获取到以下信息。

地址	数据	是否可读写	数据名称	说明
0xF000	data[0]	可读可写	运行灯状态	1：打开　　0：关闭
0xF001 0xF002	data[1] data[2]	只读 只读	温度	类型：32位float型　(data[1]<<16)|data[2] 然后按照32位float数据类型转化　单位:℃
0xF003 0xF004	data[3] data[4]	只读 只读	湿度	类型：32位float型　(data[3]<<16)|data[4] 然后按照32位float数据类型转化　单位:%RH
0xF005 0xF006	data[5] data[6]	只读 只读	S0传感器 测量气体浓度	类型：32位float型　(data[5]<<16)|data[6] 然后按照32位float数据类型转化　单位:PPM(氧气:%)
0xF007 0xF008	data[7] data[8]	只读 只读	S0传感器 最大量程	类型：uint32　(data[7]<<16)|data[8] 单位:PPM(氧气:%)
0xF009	data[9]	只读	S0传感器类别	类型：uint16　具体对应数值请参照<表格1>
0xF00A 0xF00B	data[10] data[11]	只读 只读	S1传感器 测量气体浓度	类型：32位float型　(data[10]<<16)|data[11] 然后按照32位float数据类型转化　单位:PPM(氧气:%)
0xF00C 0xF00D	data[12] data[13]	只读 只读	S1传感器 最大量程	类型：uint32　(data[12]<<16)|data[13] 单位:PPM(氧气:%)
0xF00E	data[14]	只读	S1传感器类别	类型：uint16　具体对应数值请参照<表格1>
0xF00F	data[15]	只读	S0传感器性能状态	类型：uint16　2:传感器失效，更换新传感器 1:传感器接近失效，准备更换　0:传感器性能正常
0xF010	data[16]	只读	S1传感器性能状态	类型：uint16　2:传感器失效，更换新传感器 1:传感器接近失效，准备更换　0:传感器性能正常

- **示例** 读取0xF000~0xF010地址的指令

下行: (读取17个寄存器)

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x01	0x03	0xF0	0x00	0x00	0x11	0xB6(CRC_L)	0xC6(CRC_H)

返回:

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]	Byte[8]	Byte[9]	Byte[10]	Byte[11]	Byte[12]
0x01	0x03	0x22	0x00	0x01	0x41	0x9C	0x12	0x10	0x42	0x65	0xFB	0x00

Byte[13]	Byte[14]	Byte[15]	Byte[16]	Byte[17]	Byte[18]	Byte[19]	Byte[20]	Byte[21]	Byte[22]	Byte[23]	Byte[24]	Byte[25]
0x41	0xAC	0xF2	0x2A	0x00	0x00	0x00	0x32	0x00	0x22	0x41	0x10	0x87

Byte[26]	Byte[27]	Byte[28]	Byte[29]	Byte[30]	Byte[31]	Byte[32]	Byte[33]	Byte[34]	Byte[35]	Byte[36]	Byte[37]	Byte[38]
0x74	0x00	0x00	0x00	0x32	0x00	0x22	0x00	0x00	0x00	0x00	0xC9	0xD5

2.2 控制LED指示灯

气体传感器模组上有一个LED指示灯，用于显示传感器模组的工作状态，如果你需要降低传感器模组的功耗节省电源可以对其进行关闭，按照以下操作：

- **示例** 设置运行灯状态

下行: (关闭运行灯，将0写入0xF000)

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x01	0x06	0xF0	0x00	0x00	0x00	0xB6(CRC_L)	0xC6(CRC_H)

返回:

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x01	0x06	0xF0	0x00	0x00	0x00	0xB6(CRC_L)	0xC6(CRC_H)

2.3 获取软件版本号

传感器模组会因为增加功能或提升产品性能而进行产品升级，传感器模组程序有可能进行升级，如需获取该信息，通过读取以下地址中的数据，获取当前产品内部嵌入式软件版本号。

地址	数据	是否可读写	数据名称	说明
0xB000	'D' 'G'	只读	软件版本号	具体版本号，请以读取的数据为准，由于产品升级或功能改进，版本号将会进行更新。
0xB001	'M' '1'			
0xB002	'0' '.'			
0xB003	'1' '.'			
0xB004	'1' '.'			
0xB005	'1' '.'			
0xB006	'0' '.'			

地址	数据	是否可读写	数据名称	说明
0xB007	'2' '0'			
0xB008	'2' '0'			
0xB009	'.' '0'	只读	软件版本号	具体版本号，请以读取的数据为准，由于产品升级或功能改进，版本号将会进行更新。
0xB00A	'8' ',' ' '			
0xB00B	'2' '4'			

- **示例** 读取0xB000~0xB00B地址的指令

下行: (获取软件版本号)

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x01	0x03	0xB0	0x00	0x00	0x0C	(CRC_L)	(CRC_H)

返回:

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]	Byte[8]	Byte[9]	Byte[10]	Byte[11]	Byte[12]
0x01	0x03	0x18	0x44	0x47	0x4D	0x31	0x30	0x5F	0x31	0x2E	0x31	0x2E

Byte[13]	Byte[14]	Byte[15]	Byte[16]	Byte[17]	Byte[18]	Byte[19]	Byte[20]	Byte[21]	Byte[22]	Byte[23]	Byte[24]	Byte[25]
0x31	0x2E	0x30	0x5F	0x32	0x30	0x32	0x30	0x2E	0x30	0x38	0x2E	0x32

Byte[26]	Byte[27]	Byte[28]
0x34	0xC8	0x57

注：Byte[6] Byte[7]的CRC高低位值选择加校验ModbusCRC16模式自动生成。

2.4 休眠功能地址

用于设置气体检测模组进入睡眠，或者唤醒状态，达到省电的目的。

地址	数据	是否可读写	数据名称	说明
0xC000	data[0]	只写	睡眠状态	类型：uint16 1:进入休眠 0:退出休眠

- **示例** 写入休眠指令

下行: (写入休眠指令，进入休眠)

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x01	0x06	0xC0	0x00	0x00	0x01	0xB6(CRC_L)	0xC6(CRC_H)

返回:

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x01	0x06	0xC0	0x00	0x00	0x01	0xB6(CRC_L)	0xC6(CRC_H)

注：进入休眠状态后，发送其他指令将不被执行，视为无效指令，直到退出休眠。

2.5 获取传感器序列号

每个传感器都拥有一个独立的编号。DGM10为双传感器模组，拥有两个独立的传感器序列号。通过读取以下地址中的数据，获取当前模组中传感器序列号。

地址	数据	是否可读写	数据名称	说明
0xB100	'1' '2'	只读	S0 传感器序列号	ascii 码字符串 长度：16个字节 具体传感器序列号，请以读取的数据为准，每一个传感器模组在出厂时适配的传感器都具有唯一的序列号。
0xB101	'3' '4'			
0xB102	'5' '6'			
0xB103	'7' '8'			
0xB104	'9' 'A'			
0xB105	'B' 'C'			
0xB106	'D' 'E'			
0xB107	'F' '0'			
0xB108	'0' '1'	只读	S1 传感器序列号	ascii 码字符串 长度：16个字节 具体传感器序列号，请以读取的数据为准，每一个传感器模组在出厂时适配的传感器都具有唯一的序列号。
0xB109	'2' '3'			
0xB10A	'4' '5'			
0xB10B	'6' '7'			
0xB10C	'8' '9'			
0xB10D	'A' 'B'			
0xB10E	'C' 'D'			
0xB10F	'E' 'F'			

- **示例** 读取0xB100~0xB10F地址的指令

  下行: (获取传感器序列号)

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x01	0x03	0xB1	0x00	0x00	0x10	(CRC_L)	(CRC_H)

返回:

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]	Byte[8]	Byte[9]	Byte[10]	Byte[11]	Byte[12]
0x01	0x03	0x20	0x31	0x32	0x33	0x34	0x35	0x36	0x37	0x38	0x39	0x41

Byte[13]	Byte[14]	Byte[15]	Byte[16]	Byte[17]	Byte[18]	Byte[19]	Byte[20]	Byte[21]	Byte[22]	Byte[23]	Byte[24]	Byte[25]
0x42	0x43	0x44	0x45	0x46	0x30	0x30	0x31	0x32	0x33	0x34	0x35	0x36

Byte[26]	Byte[27]	Byte[28]	Byte[29]	Byte[30]	Byte[31]	Byte[32]	Byte[33]	Byte[34]	Byte[35]	Byte[36]
0x37	0x38	0x39	0x41	0x42	0x43	0x44	0x45	0x46	0xDD	0xCA

注：Byte[6] Byte[7]的CRC高低位值选择加校验ModbusCRC16模式自动生成。

2.6 用户标定功能

在某系特殊情况下，用户如果觉得有必要对标定参数进行修改，可以使用如下指令进行修改。注意标定0点和标定浓度需要分开操作，标定0点需要在没有被检测气体和干扰气体的情况下进行。标定浓度点时需要在确定气体浓度(并且没有干扰气体)环境中进行。

地址	数据	是否可读写	数据名称	说明
0xC100	data[0]	可读可写	S0用户零点启用标志	通过写入1来启动S0传感器用户零点标定 通过写入0来恢复S0出厂零点值
0xC101	data[1]	只写	S0写入零点值	通过写入1，将S0传感器当前值设置为零点，写入0不做操作 读取的值固定为0
0xC102	data[2]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC103	data[3]	只读	S0是否设置过 用户零点值	读取值为1时：S0传感器曾写入过用户零点值 读取值为0时：S0传感器没有写入过用户零点值
0xC104	data[4]	可读可写	S0用户跨度 启用标志	通过写入1来启动S0传感器用户标定跨度 通过写入0来恢复S0出厂标定跨度值
0xC105	data[5]	只读	S0是否设置过 用户跨度标定值	读取值为1时：S0写入过用户跨度标定值 读取值为0时：S0没有写入过用户跨度标定值
0xC106 0xC107	data[6] data[7]	只写	S0用户标定跨度值	写入S0当前浓度值，读取值固定为0 数据类型为float型，值必须大于0
0xC108 0xC109	data[8] data[9]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC10A 0xC10B	data[10] data[11]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC10C 0xC10D	data[12] data[13]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC10E 0xC10F	data[14] data[15]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC110	data[16]	可读可写	S1用户零点启用标志	通过写入1来启动S1传感器用户零点标定 通过写入0来恢复S1出厂零点值
0xC111	data[17]	只写	S1写入零点值	通过写入1，将S1传感器当前值设置为零点，写入0不做操作 读取的值固定为0
0xC112	data[18]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC113	data[19]	只读	S1是否设置过 用户零点值	读取值为1时：S1传感器曾写入过用户零点值 读取值为0时：S1传感器没有写入过用户零点值
0xC114	data[20]	可读可写	S1用户跨度 启用标志	通过写入1来启动S1传感器用户标定跨度 通过写入0来恢复S1出厂标定跨度值
0xC115	data[21]	只读	S1是否设置过 用户跨度标定值	读取值为1时：S1写入过用户跨度标定值 读取值为0时：S1没有写入过用户跨度标定值
0xC116 0xC117	data[22] data[23]	只写	S1用户标定跨度值	写入S1当前浓度值，读取值固定为0 数据类型为float型，值必须大于0
0xC118 0xC119	data[24] data[25]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC11A 0xC11B	data[26] data[27]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC11C 0xC11D	data[28] data[29]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效
0xC11E 0xC11F	data[30] data[31]	不可读，不可写	保留	保留值，如果读取，值固定为0，写入无效

3、I2C通讯模式

DGM10模组采用I2C接口通信，通讯参数如下：

信号电压	最高时钟
3.3V	100K HZ

I2C 7位默认地址：0x50 (默认)
数据读写时间间隔要求：>= 1s

4、修改I2C地址

通过模组上的UART接口，使用一组(三条)特殊指令修改I2C地址，两条指令必须依次发送，中间间隔>=1s。
I2C 7位地址范围：0x08～0x7F

- 第一条指令

Byte[0]	Byte[1]
0x7F	0x7F

- 第二条指令

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]
0x55	0x4E	0x4C	0x4F	0x43	0x4B	0x32	0x50

- 第三条指令

Byte[0]	Byte[1]	Byte[2]	Byte[3]	Byte[4]	Byte[5]	Byte[6]	Byte[7]	Byte[8]	Byte[9]
0x49	0x32	0x43	0x41	0x44	0x44	0x52	addr	CRC_L	CRC_H

5、传感器类型代码表

可以通过读取保持寄存器中的传感器类型值，来识别当前模组上配置的传感器类型。传感器类型代码表：

气体名称	甲醛	有机挥发物	一氧化碳	氯气	氢气	硫化氢	氯化氢	氰化氢	氟化氢	氨气
气体分子式	HCHO	VOC	CO	Cl_2	H_2	H_2S	HCl	HCN	HF	NH_3
数值	0x17	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F	0x20

气体名称	二氧化氮	氧气	臭氧	二氧化硫	溴化氢	溴气	氟气	磷化氢	砷化氢	硅烷
气体分子式	NO_2	O_2	O_3	SO_2	HBr	Br_2	F_2	PH_3	AsH_3	SiH_4
数值	0x21	0x22	0x23	0x24	0x25	0x26	0x27	0x28	0x29	0x2A

气体名称	锗烷	乙硼烷	三氟化硼	六氟化钨	四氟化硅	二氟化氙	四氟化钛	恶臭气体	室内空气质量
气体分子式	GeH_4	B_2H_6	BF_3	WF_6	SiF_4	XeF_2	TiF_4	SMELL	IAQ
数值	0x2B	0x2C	0x2D	0x2E	0x2F	0x30	0x31	0x32	0x33

气体名称	室外空气质量	非甲烷总烃	硫化物	氮氧化物	一氧化氮	异丁烯	丙二醇	甲硫醇	苯乙烯
气体分子式	AQI	NMHC	SO_X	NO_X	NO	C_4H_8	$C_3H_8O_2$	CH_4S	C_8H_8
数值	0x34	0x35	0x36	0x37	0x38	0x39	0x3A	0x3B	0x3C

气体名称	丁烷	乙烷	正己烷	环氧乙烷	三甲胺	二甲胺	乙醇	二硫化碳	乙硫醇	二甲二硫醚
气体分子式	C_4H_{10}	C_2H_6	C_6H_{14}	C_2H_4O	C_3H_9N	C_2H_7N	C_2H_6O	CS_2	C_2H_6S	$C_2H_6S_2$
数值	0x3D	0x3E	0x3F	0x40	0x41	0x42	0x43	0x44	0x45	0x46

气体名称	乙烯	甲醇	苯	对二甲苯	甲苯	乙酸	二氧化氯	过氧化氢	联氨	偏二甲肼
气体分子式	C_2H_4	CH_3OH	C_6H_6	C_8H_{10}	C_7H_8	CH_3COOH	ClO_2	H_2O_2	N_2H_4	$C_2H_8N_2$
数值	0x47	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F	0x50

气体名称	三氯乙烯	三氯甲烷	三氯乙烷
气体分子式	C_2HCl_3	$CHCl_3$	$C_2H_3Cl_3$
数值	0x51	0x52	0x53

6、I2C寄存器地址

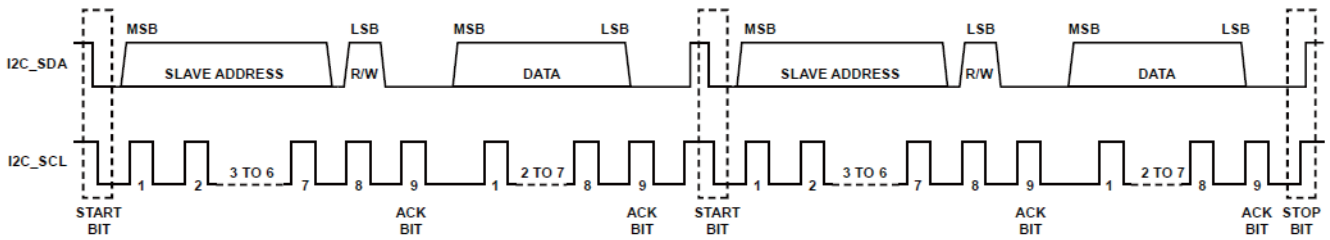
本组地址是用于输出气体浓度，温度，湿度，量程，传感器类型，运行灯开关控制，传感器当前状态。

寄存器地址	寄存器值	读/写	数据名称	说明
0x00	Byte[0]	只读	保留	保留，固定为0
0x01	Byte[1]	可读可写	运行灯状态	1：打开　　0：关闭
0x02 0x03 0x04 0x05	Byte[2] Byte[3] Byte[4] Byte[5]	只读	温度	类型：32位float型 (data[2]<<24)|(data[3]<<16)|(data[4]<<8)|data[5] 然后按照32位float数据类型转化 单位：℃
0x06 0x07 0x08 0x09	Byte[6] Byte[7] Byte[8] Byte[9]	只读	湿度	类型：32位float型 (data[6]<<24)|(data[7]<<16)|(data[8]<<8)|data[9] 然后按照32位float数据类型转化 单位：%RH
0x0A 0x0B 0x0C 0x0D	Byte[10] Byte[11] Byte[12] Byte[13]	只读	传感器(0)浓度	类型：32位float型 (data[10]<<24)|(data[11]<<16)|(data[12]<<8)|data[13] 然后按照32位float数据类型转化 单位：PPM（氧气:%）
0x0E 0x0F 0x10 0x11	Byte[14] Byte[15] Byte[16] Byte[17]	只读	传感器(0)量程	类型：uint32 (data[14]<<24)|(data[15]<<16)|(data[16]<<8)|data[17] 单位：PPM（氧气:%）
0x12 0x13	Byte[18] Byte[19]	只读	传感器(0)类型	类型：uint16 (byte[18]<<8)|byte[19]
0x14 0x15 0x16 0x17	Byte[20] Byte[21] Byte[22] Byte[23]	只读	传感器(1)浓度	类型：32位float型 (data[20]<<24)|(data[21]<<16)|(data[22]<<8)|data[23] 然后按照32位float数据类型转化 单位：PPM（氧气:%）
0x18 0x19 0x1A 0x1B	Byte[24] Byte[25] Byte[26] Byte[27]	只读	传感器(1)量程	类型：uint32 (data[24]<<24)|(data[25]<<16)|(data[26]<<8)|data[27] 单位：　PPM（氧气:%）
0x1C 0x1D	Byte[28] Byte[29]	只读	传感器(1)类型	类型：uint16 (byte[28]<<8)|byte[29]

寄存器地址	寄存器值	读/写	数据名称	说明
0x1E	Byte[30]			
0x1F	Byte[31]			
0x20	Byte[32]			
0x21	Byte[33]			
0x22	Byte[34]			
0x23	Byte[35]			
0x24	Byte[36]			
0x25	Byte[37]			
0x26	Byte[38]			
0x27	Byte[39]			
0x28	Byte[40]			
0x29	Byte[41]	只读	软件版本号	软件版本号为ASCII码，基本格式如下: DGM10_1.1.1.0_2020.08.24
0x2A	Byte[42]			
0x2B	Byte[43]			
0x2C	Byte[44]			
0x2D	Byte[45]			
0x2E	Byte[46]			
0x2F	Byte[47]			
0x30	Byte[48]			
0x31	Byte[49]			
0x32	Byte[50]			
0x33	Byte[51]			
0x34	Byte[52]			
0x35	Byte[53]			
0x36	Byte[54]			
0x37	Byte[55]			
0x38	Byte[56]			
0x39	Byte[57]			
0x3A	Byte[58]			
0x3B	Byte[59]			
0x3C	Byte[60]			
0x3D	Byte[61]	只读	传感器(0)序列号	传感器序列号为ASCII码，基本格式如下: 1111111111111111
0x3E	Byte[62]			
0x3F	Byte[63]			
0x40	Byte[64]			
0x41	Byte[65]			
0x42	Byte[66]			
0x43	Byte[67]			
0x44	Byte[68]			
0x45	Byte[69]			
0x46	Byte[70]			
0x47	Byte[71]			
0x48	Byte[72]			
0x49	Byte[73]			
0x4A	Byte[74]			
0x4B	Byte[75]			
0x4C	Byte[76]			
0x4D	Byte[77]	只读	传感器(1)序列号	传感器序列号为ASCII码，基本格式如下: 1111111111111111
0x4E	Byte[78]			
0x4F	Byte[79]			
0x50	Byte[80]			
0x51	Byte[81]			
0x52	Byte[82]			
0x53	Byte[83]			
0x54	Byte[84]			
0x55	Byte[85]			
0x56	Byte[86]	可读可写	休眠模式	进入休眠模式：RX拉高，通过I2C写入1，并且在进入休眠后继续保持高电平，进如休眠I2C通道变为无效，直到退出休眠 退出休眠模式：RX拉低5ms，然后再拉高

寄存器地址	寄存器值	读/写	数据名称	说明
0x57	Byte[87]	可读	保留	保留字段
0x58	Byte[88]			
0x59	Byte[89]			
0x5A	Byte[90]			
0x5B	Byte[91]			
0x5C	Byte[92]			
0x5D	Byte[93]			
0x5E	Byte[94]			
0x5F	Byte[95]	可写	传感器(0)标定浓度写入	类型：32位float型 (data[10]<<24)|(data[11]<<16)|(data[12]<<8)|data[13] 然后按照32位float数据类型转化 单位：PPM (氧气:%) 连续写入4个寄存器，不能与其它寄存器同时写入
0x60	Byte[96]			
0x61	Byte[97]			
0x62	Byte[98]			
0x63	Byte[99]	可写	传感器(1)标定浓度写入	类型：32位float型 (data[10]<<24)|(data[11]<<16)|(data[12]<<8)|data[13] 然后按照32位float数据类型转化 单位：PPM (氧气:%) 连续写入4个寄存器，不能与其它寄存器同时写入
0x64	Byte[100]			
0x65	Byte[101]			
0x66	Byte[102]			
0x67	Byte[103]	可写	传感器(0)恢复出厂标定	1：恢复出厂标定　　其他：无效 单独写入，不能与其他寄存器同时写入
0x68	Byte[104]	可写	传感器(1)恢复出厂标定	1：恢复出厂标定　　其他：无效 单独写入，不能与其他寄存器同时写入

I²C时序



读写接口例程:

```
/************************************
功能：向一个寄存器中写入一个字节数据
dev_add:I2C 7位地址左移一位
register_addr:写入寄存器地址
data:写入数据
************************************/
void send_data(uint8_t dev_add, uint8_t register_addr, uint8_t data)
{
    I2C_Start();
    I2C_SendByte(dev_addr);
    if(I2C_CheckAck())
    {
        I2C_Stop();
        return;
    }
```

```
        I2C_SendByte(register_addr);
        if(I2C_CheckAck())
        {
            I2C_Stop();
            return;
        }
        I2C_SendByte(data);
        if(I2C_CheckAck())
        {
            I2C_Stop();
            return;
        }
        I2C_Stop();
}

/**********************************
功能:从register_addr寄存器开始读取read_len个字节
dev_add:I2C 7位地址左移一位
register_addr:写入寄存器地址
data:读取数据存储地址
read_len:读取长度
**********************************/
void read_data(uint8_t dev_add, uint8_t register_addr, uint8_t* data, uint8_t read_len)
{
    uint8 i = 0;
    I2C_Start();
    I2C_SendByte(dev_addr);
    if(I2C_CheckAck())
    {
        I2C_Stop();
        return;
    }
    I2C_SendByte(register_addr);
    if(I2C_CheckAck())
    {
        I2C_Stop();
        return;
    }

    I2C_ReStart();
        I2C_SendByte(dev_addr|0x01);
        if(I2C_CheckAck())
        {
            I2C_Stop();
            return;
        }

        datax[i] = I2C_ReceiveByte();
        i++;
        for(i<data_len;i++)
        {
            I2C_SendAck();
            datax[i] = I2C_ReceiveByte();
        }
        I2C_SendNAck();
        I2C_Stop();
}
```

德国研发生产中心

德国 EC Sense GmbH

Wangener Weg 3 | 82069 Hohenschäftlarn

座机：+49 (0)8178–99992–10

传真：+49 (0)8178–99992–11

邮箱：office@ecsense.com

网址：www.ecsense.com

亚太区 · 中国应用设计研发中心

宁波爱氪森科技有限公司

浙江 . 宁波市鄞州区金谷北路 228 号中物科技园 6 号楼

邮编：315100

座机：0574–88097236，88096372

邮箱：info@aqsystems.cn

网址：www.ecsense.cn