



Engineering Standard

Smart Device Communication Standard

Version	Date	Initial	History
1.00	20th May 2019	Yingguang Wang	Initial release

Table of Contents

Overview.....	3
1. System Architecture.....	3
2. Packet Format	4
3. Communication Handshaking.....	5
4. Command Set Summary	6
4.1 Basic Information.....	6
4.2 Data Request	6
4.3 Data Set.....	7
4.4 Advanced Operations.....	7
Appendix I: CRC-16.....	9
Appendix II: SDCS Return Code.....	10
Appendix III: Data decoding examples	11
i. Get Sensor Format.....	11
ii. Get Data Pack	11
iii. Set Sensor Parameters	12
Appendix IV: Calibration Process	13

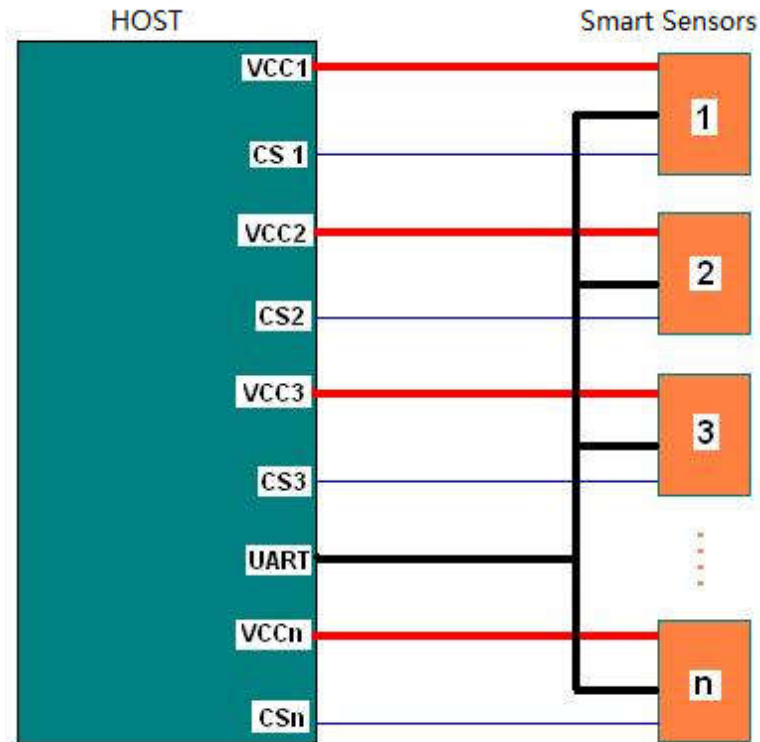
Overview

This document describes the '*Smart Device Communication Standard*', or SDCS, which enables the end user or 3rd party OEM to transfer remote control or raw data in real time between the smart device and the microcontroller.

Two communication modes – Responder mode (called Master/Slave mode) and Aloha mode (Initiative Reporting) are provided. At responder mode Unit won't initiate communication and only responds to commands from host, at Aloha mode Unit will transmit data packs by period or reaching one specific gas threshold.

1. System Architecture

This protocol is designed for Smart Sensors. The protocol works between Smart Sensors (SLAVE) and Host MCU(HOST). The communication is point-to-point, no source and destination address need.



Smart Sensor provide TTL communication interface to host MCU: Tx, Rx, and CS.

CS - chip select, falling edge (a high jump to low) will wake up the smart sensor asleep and activate the communication port, time delay from CS falling edge to normal UART communication will be no more than 20us; rising edge (a low jump to high) will de-activate the communication port of the smart sensor.

Tx – connect to Smart Sensor UART Tx port, transmit data to host

Rx – connect to Smart Sensor UART Rx port, receive data from host

2. Packet Format

The SDCS packet comprises of seven fields, as shown as below.

Field	1 st	2 nd	3 rd	4 th	5 th	6 th	7 th
<i>Item</i>	<i>SOP</i>	<i>Ver</i>	<i>Len</i>	<i>Cmd</i>	<i>Data</i>	<i>CRC</i>	<i>EOP</i>
<i>Bytes</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>0~n</i>	<i>2</i>	<i>1</i>

- Field 1 : *SOP*** **Description :** Start of packet
Content : 0x7B ‘{‘
- Field 2 : *Ver*** **Description :** Packet and command version
Content : 0x58
- Field 3 : *Len*** **Description :** Length of the rest of packet
Content : Total number of bytes for fields 4,5, 6 and 7
- Field 4 : *Cmd*** **Description :** Command Code
Content : Various
- Field 5 : *Data*** **Description :** Data requested by the command code
Content : Various, maxium 128 bytes, big-endian by default
- Field 6 : *CRC*** **Description :** CRC-16 (count from SOP to Data)
Content : Refer to Appendix I
- Field 7 : *EOP*** **Description :** End of packet
Content : 0x7D ‘}‘

Honeywell	Smart Device Communication Standard	
	Doc.No. 904-E800-142	Rev.No. W
	Date: 20/05/2019	Page 5 of 13

3. Communication Handshaking

The handshaking of the SDCS communication packet is defined as follows:

- The host always initiates the communication with a **Command Packet**
- The host will wait to receive an acknowledge packet before transmitting the next command packet to the same unit.
- If there is no error during transfer, with non-network connection, the slave will respond within 250 ms (If timeout 3 times, the slave will be taken as off line). The returned packet can be either a **Response Packet** or an **Error Packet**

- **Command Packet:** Sent from Host to Slave to request or to get something.
- **Response Packet:** Sent from Slave to Host after slave successfully finishes the operation requested by the Command Packet. The packet will include the information the Command Packet requested. The Packet Type for *Ver* and *CMD* in both Command Packet and Response Packet are the same. Len and Chk should be updated accordingly.

Examples:

Host send: *SOP Ver Len Cmd Chk EOP*

Slave response: *SOP Ver Len Cmd Data[0] ...Data[n] Chk EOP*

- **Error Packet:** If the slave receives a Command Packet, but can not perform the requested command or can not prepare the information, it will return an error packet indicating the reason for the failure. The error packet is a separated category.

The CMD is **ERROR_SLAVE** (0x71).

For the error code data, refer to appendix II.

Examples:

Host send: *SOP Ver Len Cmd Chk EOP*

Error packet response: *SOP Ver Len ERROR_SLAVE ErrorCode Chk EOP*

- If the Host receives **Error Packet**, it will not re-transmit.
- To protect settings, before Settings packet, host should set write-protect off, or settings will fail. Refer to command `WRITE_PROTECT` for details.
- One option is provided to enable or disable unit Aloha mode (initiative reporting) by command `ALOHA_CONFIG`. Once Aloha mode enabled, unit will transmit `ALOHA_DATA_PACKS` by period or reaching the gas threshold without requests from the host.

Honeywell	Smart Device Communication Standard	
	Doc.No. 904-E800-142	Rev.No. W
	Date: 20/05/2019	Page 6 of 13

4. Command Set Summary

Command Name	CMD	Command Data(Host)	Response Data(Slave)	Description
4.1 Basic Information				
GET_MODEL_NAME	0x10		DS[0-n]=ModelName	Model number ASCII format e.g. "ISN-1100"
GET_PROD_NAME	0x11		DS[0-n]=ProductName	Product name ASCII format e.g. "iSensor(7RPlus) PID"
GET_FW_VER	0x12		DS[0] = 'V' DS[1] = VER_MAJOR+'0' DS[2] = '.' DS[3] = VER_MINOR/10+'0' DS[4] = VER_MINOR%10+'0' DS[5] = PATCH	Firmware version, in ASCII format. e.g. "V1.01A" VER_MAJOR = 1 VER_MINOR = 1 PATCH = 'A'
GET_SEN_SN	0x13	DH[0]= type	DS[0-n] = SN	Sensor SN, ASCII format. e.g. "C00123456789" type: 0, for module 1~255 reserved
4.2 Data Request				
GET_DATA_PACK	0x30	DH[0]=Index 0x00-0x0F, single sensor request	DS[0]=Sensor Status DS[1]=Sensor Alarm DS[2~n+2]=Reading*10, n depends on data length n=4	Sensor Status: B0: B1: in warmup B2: B3: in calibration B4: B5: B6: B7: Sensor Alarm: B0: Over range B1: Reserved B2: Time not synchronized B3: High B4: Low B5: STEL B6: TWA B7: Drift Reading: instant reading, output from MSB to LSB , Unit: ppm
GET_SEN_PARA	0x33	DH[0]=Index 0x00-0x0F, single sensor request DH[1]=ParaMask.High DH[2]=ParaMask.Low	DS[0..n]=Parameter data; n depend on Data Length. LSB of ParaMask first out. DS[n+1...]: Repeated as <u>DS[0..n]</u> according to DH[1..2] bits.	Get sensor parameters ParaMask.Low: B0: Span*1000 B1: Low (Read-only) *1000 B2: High(Read-only) *1000 B3: reserved B4: OverRange (Read-only) *1000 B5: STEL(Read-only) *1000 B6: TWA(Read-only) *1000 B7: reserved ParaMask.High: reserved. B8: reserved B9: reserved B10: reserved B11: reserved B12: reserved B13: reserved B14: reserved B15: reserved

GET_SEN_NAME	0x35	DH[0]=Index 0x00-0x0F, single sensor request	DS[0..n]=sensor name string	e.g. "VOC", "CO", "H2S"; '\0' included
GET_SEN_PROD_DATE	0x37	DH[0]=Index 0x00-0x0F, single sensor request	DS[0]= year-2000 DS[1]= month DS[2]= day DS[3~4] = Warranty Days	Sensor produce date: Year(0~99, 0 means 2000) Month(1~12) Day(1~31) Sensor warranty Days: Warranty Days(0~65535)

4.3 Data Set

4.4 Advanced Operations

WRITE_PROTECT	0xA0	DH[0]=Operation 0: read 1: set If set then have the following byte: DH[1] = Data	If DH[0]=read, then have the following byte: DS[0]=Data	Before any settings, including advanced settings, should set write-protect off and it will keep the status for 300s, or response FAIL_NOWRITE error. Data: 1: ON 0: OFF
USER_CAL	0xA1	DH[0~1]=Sensor BitMap 1 – enable 0 – disable DH[2]=CalType DH[3]=Operation 0x01 – Prepare for calib 0x02 – Start user calib 0x03– Get calib result 0x00– Abort calib process If Operation = 0x02, then have the following bytes, calibration date&time DH[4]= year DH[5]= month DH[6]= day DH[7]= hour DH[8]= minute	If Operation = 0x00 and 0x01, no DS[] return needed; If Operation = 0x02, return calibration complete time cost in ms DS[0..1] = TimeCost If Operation = 0x03, return calib result DS[0~1] = Result Bit15..0: 1 – Success 0 – Fail	CalType, Calibration type. 0, Zero 1, Span 3~255, reserved. Calibration date&time, Year(0~99, 0 means 2000) Month(1~12) Day(1~31) Hour(0~23) Minute(0~59) Note: if calibration success, date will be set as the last calibration date Calibration process refer to appendix IV: Calibration process
SET_SEN_PARA	0x80	DH[0]=Index 0x00-0x0F, single sensor request DH[1]=ParaMask.High DH[2]=ParaMask.Low DH[3..n]=Parameter data n depend on Data Length. LSB of ParaMask first out. DH[n+1...]: Repeated as <u>DH[3..n]</u> according to DH[1..2] bits.		ParaMask.Low: B0: Span*1000 B1: readonly B2: readonly B3: reserved B4: readonly B5: readonly B6: readonly B7: reserved ParaMask.High: reserved. B8: reserved B9: reserved B10: reserved B11: reserved B12: reserved B13: reserved B14: reserved



Smart Device Communication Standard

Doc.No. 904-E800-142

Rev.No. W

Date: 20/05/2019

Page 8 of 13

				B15: reserved Data output from MSB to LSB, details refer to Appendix III: data decoding examples
--	--	--	--	---

Honeywell	Smart Device Communication Standard	
	Doc.No. 904-E800-142	Rev.No. W
	Date: 20/05/2019	Page 9 of 13

Appendix I: CRC-16

CRC-16 is based on polynomial $x^{16} + x^{15} + x^2 + 1$, 0x8005

```

#define CRC16_POLY          0x8005
#define CRC16_INIT_REM     0x0000
#define CRC16_FINAL_XOR    0x0000
#define CRC_TABLE_SIZE     256

const unsigned short crc16Table[CRC_TABLE_SIZE] = {
    0x0000, 0x8005, 0x800F, 0x000A,
    0x801B, 0x001E, 0x0014, 0x8011,
    0x8033, 0x0036, 0x003C, 0x8039,
    0x0028, 0x802D, 0x8027, 0x0022,
    0x8063, 0x0066, 0x006C, 0x8069,
    0x0078, 0x807D, 0x8077, 0x0072,
    0x0050, 0x8055, 0x805F, 0x005A,
    0x804B, 0x004E, 0x0044, 0x8041,
    0x80C3, 0x00C6, 0x00CC, 0x80C9,
    0x00D8, 0x80DD, 0x80D7, 0x00D2,
    0x00F0, 0x80F5, 0x80FF, 0x00FA,
    0x80EB, 0x00EE, 0x00E4, 0x80E1,
    0x00A0, 0x80A5, 0x80AF, 0x00AA,
    0x80BB, 0x00BE, 0x00B4, 0x80B1,
    0x8093, 0x0096, 0x009C, 0x8099,
    0x0088, 0x808D, 0x8087, 0x0082,
    0x8183, 0x0186, 0x018C, 0x8189,
    0x0198, 0x819D, 0x8197, 0x0192,
    0x01B0, 0x81B5, 0x81BF, 0x01BA,
    0x81AB, 0x01AE, 0x01A4, 0x81A1,
    0x01E0, 0x81E5, 0x81EF, 0x01EA,
    0x81FB, 0x01FE, 0x01F4, 0x81F1,
    0x81D3, 0x01D6, 0x01DC, 0x81D9,
    0x01C8, 0x81CD, 0x81C7, 0x01C2,
    0x0140, 0x8145, 0x814F, 0x014A,
    0x815B, 0x015E, 0x0154, 0x8151,
    0x8173, 0x0176, 0x017C, 0x8179,
    0x0168, 0x816D, 0x8167, 0x0162,
    0x8123, 0x0126, 0x012C, 0x8129,
    0x0138, 0x813D, 0x8137, 0x0132,
    0x0110, 0x8115, 0x811F, 0x011A,
    0x810B, 0x010E, 0x0104, 0x8101,
    0x8303, 0x0306, 0x030C, 0x8309,
    0x0318, 0x831D, 0x8317, 0x0312,
    0x0330, 0x8335, 0x833F, 0x033A,
    0x832B, 0x032E, 0x0324, 0x8321,
    0x0360, 0x8365, 0x836F, 0x036A,
    0x837B, 0x037E, 0x0374, 0x8371,
    0x8353, 0x0356, 0x035C, 0x8359,
    0x0348, 0x834D, 0x8347, 0x0342,
    0x03C0, 0x83C5, 0x83CF, 0x03CA,
    0x83DB, 0x03DE, 0x03D4, 0x83D1,
    0x83F3, 0x03F6, 0x03FC, 0x83F9,
    0x03E8, 0x83ED, 0x83E7, 0x03E2,
    0x83A3, 0x03A6, 0x03AC, 0x83A9,
    0x03B8, 0x83BD, 0x83B7, 0x03B2,
    0x0390, 0x8395, 0x839F, 0x039A,
    0x838B, 0x038E, 0x0384, 0x8381,

```

```

0x0280, 0x8285, 0x828F, 0x028A,
0x829B, 0x029E, 0x0294, 0x8291,
0x82B3, 0x02B6, 0x02BC, 0x82B9,
0x02A8, 0x82AD, 0x82A7, 0x02A2,
0x82E3, 0x02E6, 0x02EC, 0x82E9,
0x02F8, 0x82FD, 0x82F7, 0x02F2,
0x02D0, 0x82D5, 0x82DF, 0x02DA,
0x82CB, 0x02CE, 0x02C4, 0x82C1,
0x8243, 0x0246, 0x024C, 0x8249,
0x0258, 0x825D, 0x8257, 0x0252,
0x0270, 0x8275, 0x827F, 0x027A,
0x826B, 0x026E, 0x0264, 0x8261,
0x0220, 0x8225, 0x822F, 0x022A,
0x823B, 0x023E, 0x0234, 0x8231,
0x8213, 0x0216, 0x021C, 0x8219,
0x0208, 0x820D, 0x8207, 0x0202
};

unsigned short crc16MakeTableMethod(unsigned short crc, const unsigned short *table,
                                   unsigned char *pbuffer, unsigned int length)
{
    while(length--)
        crc = table[((crc >> 8) ^ *pbuffer++) ^ (crc << 8)];    // normal
    return(crc ^ CRC16_FINAL_XOR);
}

unsigned char msg[10] = "123456789";
unsigned short length = 9;
crc16 = crc16MakeTableMethod(CRC16_INIT_REM, crc16Table, msg, length);
// the expected CRC: crc16=0xFEE8

```

Appendix II: SDCS Return Code

Code	Return Code Name	Comment
0x31	FAIL_UNKNOWN	For all failure reasons other than in the following list
0x32	FAIL_INVALIDCMD	The command is not supported by this unit
0x33	FAIL_DATASIZE	The command is supported by this unit, but the data in the command packet is not supported
0x34	FAIL_INVALIDVALUE	The value to be set is not valid
0x39	FAIL_WRITEPROTECT	The value to be set is protected, can not be changed
0x3F	FAIL_OPERATION	Failed after execute command

Honeywell	Smart Device Communication Standard	
	Doc.No. 904-E800-142	Rev.No. W
	Date: 20/05/2019	Page 11 of 13

Appendix III: Data decoding examples

i. Get Sensor Format

Send data packet:

```

0x7B      SOP
0x58      Ver
0x05      Length, 5 bytes, (0x31 0x01 0x53 0x92 0x7D)
0x31      CMD
0x00      Data, sensor index=0, the first sensor
0xD3 0x97 CRC16 (0x7B 0x58 0x05 0x31 0x00)= 0xD397
0x7D      EOP

```

Recieve data packet:

```

0x7B      SOP
0x58      Ver
0x0B      Length = 11
0x31      CMD=0x31
0x03      Decimal Point(DP), DP = 3
0x04      Data Length(DL), DL = 4
0x00      unit code = 0, ppm
0x01 0xFF ResInt = 1, ResExp=-1 , resolution = 1*10^(-1)=0.1 ppm
0x80 0xFF Parameter mask=0x80FF, Span/Low/High/SPANH/OverRange/STEL/TWA/CF/Caltime
0xED 0x44 CRC16=0xED44
0x7D      EOP

```

ii. Get Data Pack

Send data packet:

```

0x7B      SOP
0x58      Ver
0x05      Length=5
0x30      CMD = 0x30
0x00      index=0, the first sensor
0x55 0x94 CRC16=0x5594
0x7D      EOP

```

Recieve data packet:

```

0x7B      SOP
0x58      Ver
0x0A      Length=10
0x30      CMD=0x30
0x02      status=0x02, in warmup
0x04      Error=0x04, sensor/lamp off
0x00 0x01 0xE2 0x6C reading =0x0001E26C=123500, so real reading = 123500/10^DP=123.500 ppm
0x44 0x90 CRC16 = 0x4490
0x7D      EOP

```

Honeywell	Smart Device Communication Standard	
	Doc.No. 904-E800-142	Rev.No. W
	Date: 20/05/2019	Page 12 of 13

iii. Set Sensor Parameters

Send data packet:

```

0x7B      SOP
0x58      Ver
0x0F      Length=15
0x80      CMD=0x80
0x00      index=0, the first sensor.
0x00 0x81 paraMask=0x0081, to set Span and CF, so followed by data and Span data first.
0x00 0x01 0x86 0xA0 Span data(length 4 bytes)=0x000186A0=100000, set Span=100000/10^DP=100.000 ppm
0x00 0x00 0x00 0x0F CF data(length 4 bytes) =0x0000000F=15, set CF=15/10=1.5
0x8B 0xD8 CRC16=0x8BD8
0x7D      EOP

```

Recieve data packet:

If write prettection is on, will receive error packet:

```

0x7B      SOP
0x58      Ver
0x05      Length=5
0x71      CMD=0x71, SLAVE_ERROR, followed by error code.
0x39      Error Code=0x39, FAIL_NORIGHT
0x53 0x07 CRC16=0x5307
0x7D      EOP

```

If write prettection is off and setting is OK, will receive normal packet:

```

0x7B      SOP
0x58      Ver
0x04      Length=4
0x80      CMD=0x80
0xC3 0xF7 CRC16=0xC3F7
0x7D      EOP

```

Appendix IV: Calibration Process

