OMRON

MEMS Flow Sensor

D6F-A7D/-AB71D

User's Manual

MEMS Flow Sensor

A302-E1-02

Table of Contents

1 Overview

This User's Manual shows how to use our MEMS digital output flow sensor (D6F-□□□□D-000-□), special instructions, etc. This document is a supplement to the product catalog, and should be used in combination with the product catalog in actual use.

2 Product lineup

The lineup of MEMS digital output flow sensors (D6F-□□□□D-000-□) are introduced in table 1.

Table 1 lineup

Flow range	Flow port type	Model
0~10 L/min	Quick Joint (P10)	D6F-10A7D-000-0
0~20 L/min		D6F-20A7D-000-0
0~50 L/min		D6F-50A7D-000-0
0~70 L/min	Quick Joint (P14)	D6F-70AB71D-000-0

The accessories (Sold separately) are introduced in Table 2.

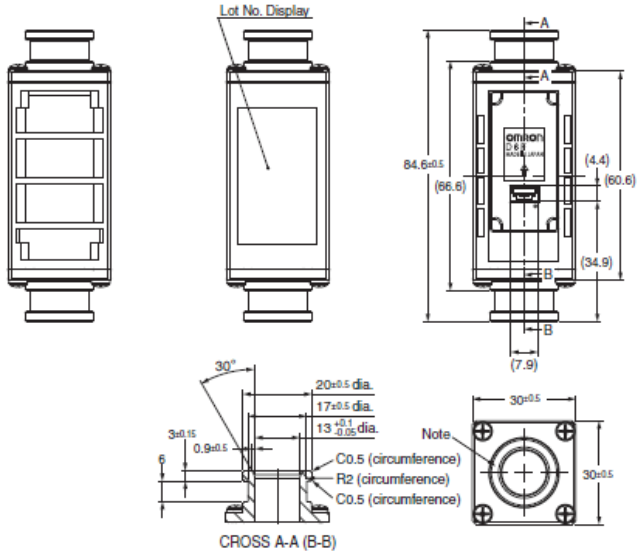
Table 2 Accessories (Sold separately)

Type	Model
Cable	D6F-CABLE3
Quick fastener	D6F-FASTENER-P10
Pipe fittings (P1－BambooΦ10mm)	D6F-PLG1

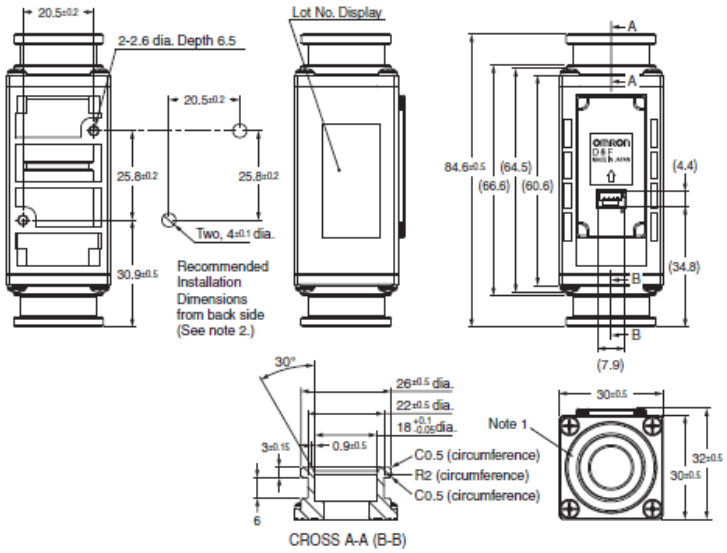
3 Dimensions

3.1 MEMS flow sensors

D6F-10A7D-000-0
D6F-20A7D-000-0
D6F-20A7D-000-0



D6F-70AB71D-000-0



*1 The port type of pipe fitting based on “Quick Joint P10 type “ or “Quick Joint P14 type”
P10 or P14 shows the name of an O-ring prescribed by JIS B 2401
The port of O-ring ditch is based on P10/P14 of JIS B 2401
Please obtain a male joint separately.

*2 To mount the sensor with 2.6-dia. holes, use P-type self-tapping screws with a nominal diameter of 3 mm and tighten them to a torque of 1.2 Nm max.
The screw threads must engage for 5.5 mm min.

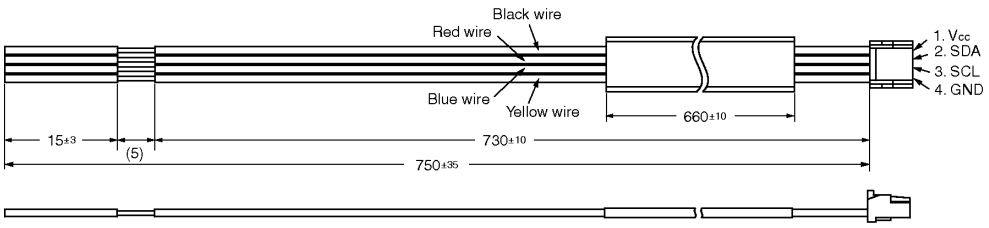
Use the following connectors to connect to the Sensor.

Housing	: GHR-04V-S（JST）
Terminals	: SSHL-002T-P0.2（JST）
Wires	: AWG26 to 30

3.2 Accessories

Cable

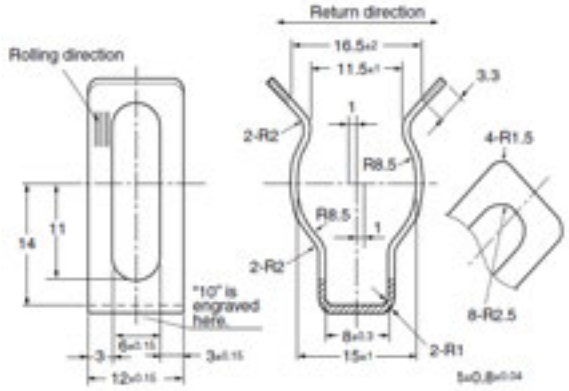
D6F-CABLE3



Contact :SSHL-002T-P0.2 (Manufactured by J.S.T. Mfg. Co., Ltd.)
Housing :GHR-04V-S (Manufactured by J.S.T. Mfg. Co., Ltd.)
Wire :AWG#28

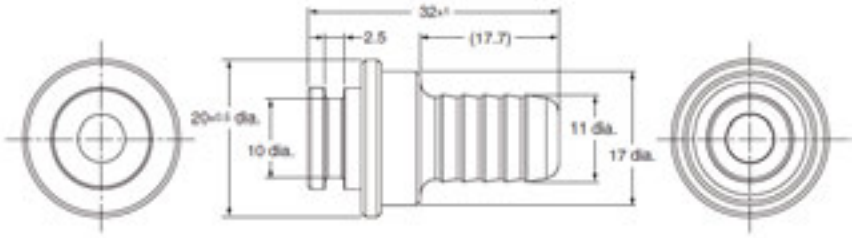
Quick fastener

D6F-FASTENER-P10



Pipe fittings

D6F-PLG1



4 Principle and Structure of flow sensor

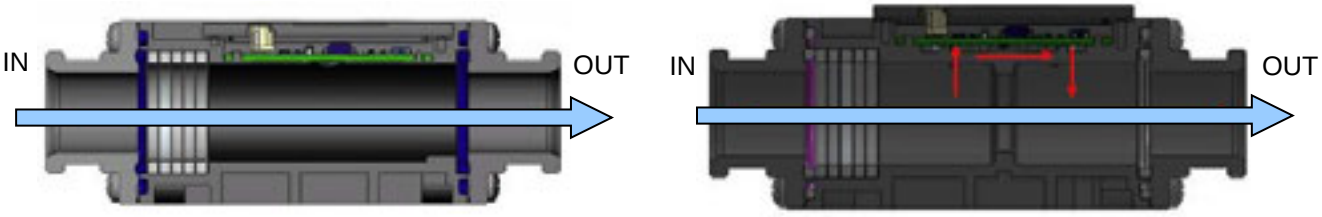
4.1 Principle

MEMS Digital Output Flow Sensor (D6F-□□□□D-000-□) is a thermal mass flow sensor.

A heater and thermopiles are make on a thin film formed on a silicon substrate which is used to measure the flow rate by capturing the heat transfer as the gas moves as a change in air flow.

4.2 Structure

The internal cross-sectional view of the MEMS flow sensor (D6F-□□□□D-000- □) is shown below. The flow of gas flows from the inlet toward the outlet, and the air flows on the surface of the MEMS flow sensor chip.



D6F-□□A7D-000-0

D6F-70AB71D-000-0

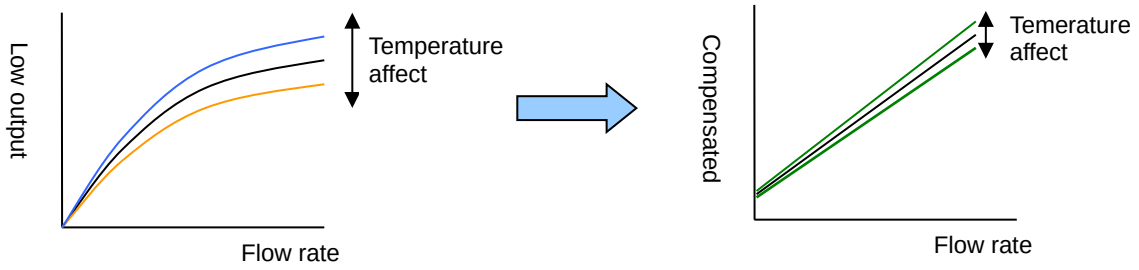
5 Feature

5.1 Liner output

The raw output voltage value from the flow sensor chip is approximately proportional to the square root of the gas flow rate. Also, the raw output voltage value of the flow sensor is affected by the ambient temperature.

D6F-□□□□D-000-0 implements the following correction function.

1) Output characteristics compensation
2) Ambient temperature compensation

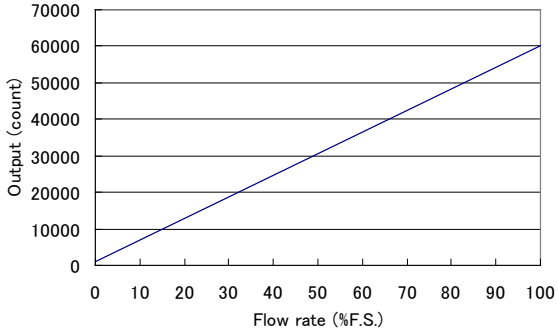


without compensation　　After compensation

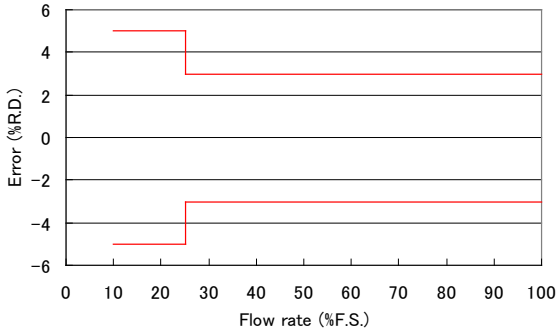
5.2 Accuracy

Output characteristics of D6F-□□□□D-000-0 is shown in below figure (left). This digital flow sensor outputs a 16-bit measured flow value through the I^2C interface. By using the above linear correction and temperature correction, the highly accurate flow error shown in the red line in the right figure is realized.

- +/- 3 %R.D ： 25 – 100 %F.S
- +/- 5 %R.D ： 10 – 25 %F.S



Output Characteristics



Accuracy

6 Characteristics / Performance

Model	D6F-10A7D-000-0	D6F-20A7D-000-0	D6F-50A7D-000-0	D6F-70AB71D-000-0
Flow Range (see note 1)	0 to 10 L/min	0 to 20 L/min	0 to 50 L/min	0 to 70 L/min
Calibration Gas (see note 2)	Air			
Flow Port Type	Quick Joint P10			Quick Joint P14
Electrical Connection	Four-pin connector			
Power supply	3.0 to 3.6 VDC			
Resolution	15bit			
Accuracy (See Note 3)	±5%RD (10%FS ≦ Flow rate < 25%FS) ±3%RD (25%FS ≦ Flow rate ≦ 100%FS)			±5%RD (10L/min ≦ Flow rate < 20L/min) ±3%RD (20L/min ≦ Flow rate ≦ 70L/min)
Response time	90ms or less			
Repeatability (See Note 4)	0.3%RD	0.3%RD	0.5%RD	1.3%RD
Interface	I²C			
Case	PPS			
Degree of Protection	IEC IP40 (Excluding tubing sections)			
Withstand Pressure	100kPa			
Pressure Drop (See Note 4)	0.034kPa	0.083kPa	0.28kPa	0.57kPa
Operating Temperature (See Note 6)	-10 to +60 ℃			
Operating Humidity (See Note 6)	35 to 85%RH			
Storage Temperature (See Note 6)	-30 to +80 ℃			
Storage Humidity (See Note 6)	35 to 85%RH			
Insulation Resistance	Between sensor outer cover and lead terminals: 20MΩ min. (at 500VDC)			
Dielectric Strength	Between sensor outer cover and lead terminals: 500VAC, 50/60Hz min, for 1min(leakage current: 1mA max.)			
Weight	57.3 g			64.4 g

Note 1: Volumetric flow rate at 0℃ , 101.3kPa

Note 2: Dry gas (must not contain large particles, e.g., dust, oil, or mist)

Note 3: -10℃ ≦ Operating Temperature ≦ 60℃

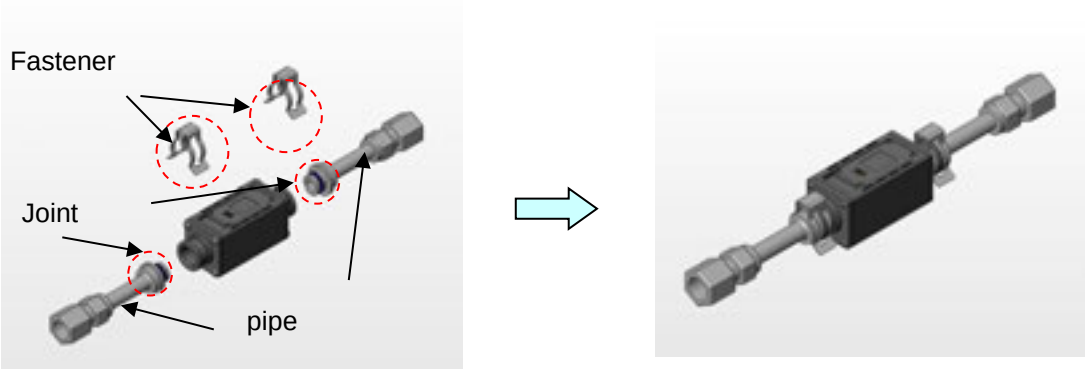
Note 4: Reference (typical)

Note 6: With no condensation or icing

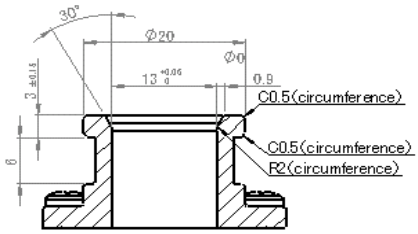
7 Setup

7.1 Piping connection of D6F-□□□□D-000-0

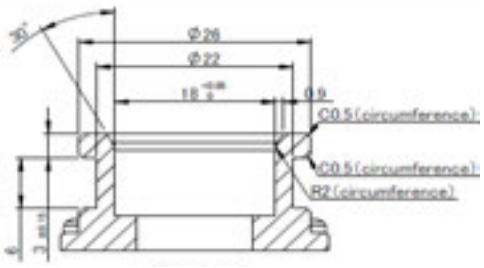
By using P10 or P14 quick joint, it is possible to easily connect with our digital flow sensor



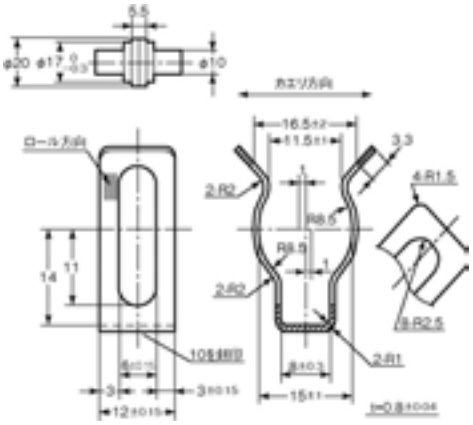
Quick joint can be used to connect the flow sensor and piping, which can be attached and detached by hand without using a tool. Currently, two types of quick joint model are available: P10 and P14. Please refer to each product data sheet to determine which type of quick fastener type to use.



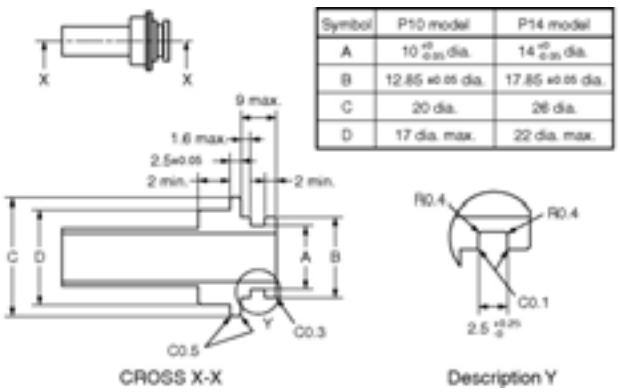
Joint cross section P10



Joint cross section P14



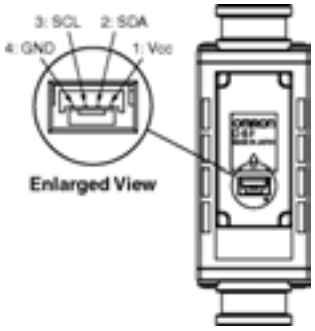
Quick fastener D6F-FASTENER-P10



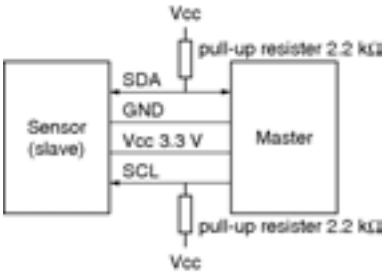
Male joint dimension

7.2 Electrical connection

D6F-□□□□D-000-□ is a digital I²C output method, pull-up resistors are required for the clock line (SCL) and data line (SDA) to communicate. As shown in the figure below, a 2.2 kΩ (recommended) pull-up resistor is required between VDD and each signal line.



Connecter layout



Connection

※ Attention

Compared to analog output type products, digital output type (I²C communication output) products have improved noise immunity, but errors in the environment may occur due to noise effects in the environment used by the customer. In that case, it is necessary to confirm the following points and to improve communication error.

Clock

Although this product supports SCL frequencies up to 400 kHz, if communication errors are likely to occur, we recommend that SCL frequencies be used at 100 kHz.

Wiring cable check

If the cable length between the customer control microcomputer and our flow sensor is long, it will be more susceptible to noise. In this case, it is recommended to use a shielded cable.

Check pull-up resistance

This product's I²C communication requires a pull-up resistor. The recommended resistance value is 2.2(kΩ), but please select the optimum resistance value according to the connection cable length between the customer side control microcomputer and our flow sensor.

In addition, if ACK is not returned from the sensor side, please judge as communication error. The ACK response time is one clock cycle of SCL, so if there is no ACK response for more than period, communication error will occur. In that case, you need to turn off the power.

8 Communication Specification

8.1 I²C Outline of I²C communication

Table 3 Basic functions of I²C communication

Item	Specification
Serial interface	I²C
SCL frequency	Max 400 kHz
Output format	Binary data (upper byte, lower byte)
Slave address	Bin 1101100b Hex 0x6C

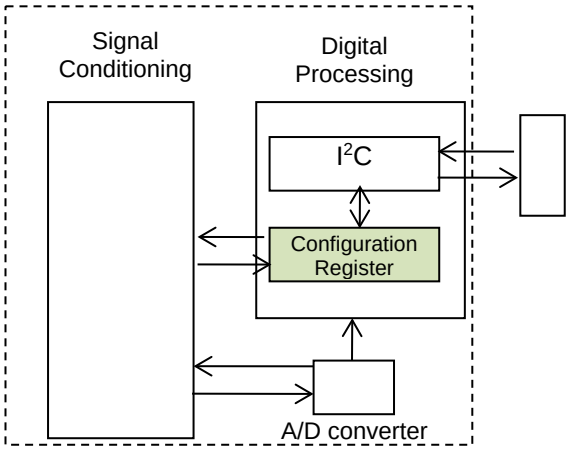
The I²C slave address is expressed as follows（0x6C）

Bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
								R/W
Value	1	1	0	1	1	0	0	1/0

Write：Set the LSB of the slave address to "0" and set it to D8h (1101_1000b)

Read：Set the LSB of the slave address to "0" and set it to D9h (1101_1001b)

8.2 Interface configuration resister

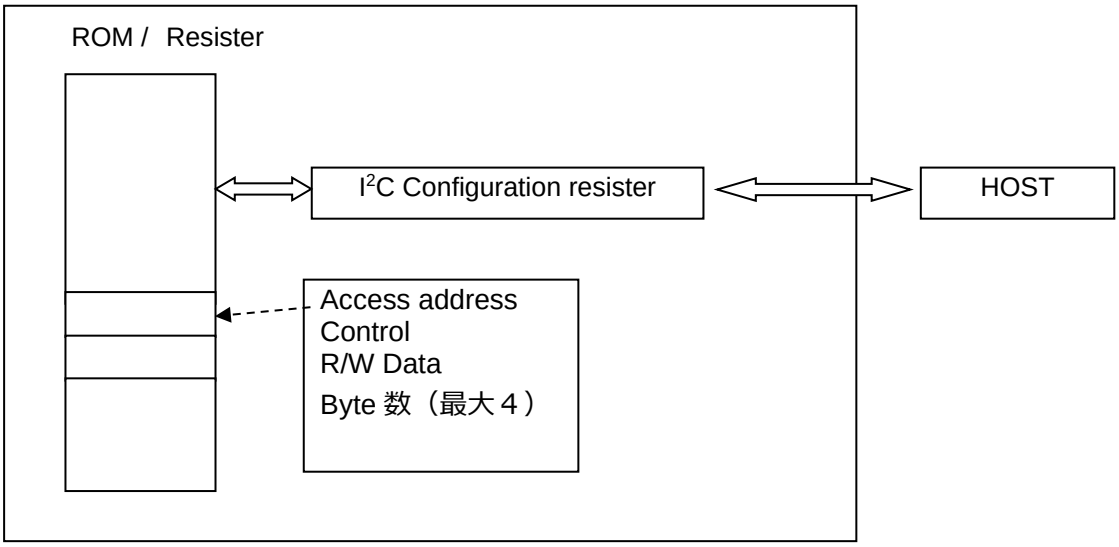


Schematic figure of the configuration resister

Table 4 Resiter map

Address	Name	Remark
D040h	SENS_CTRL	Sensor Control
D049h	INT_CTRL	CRC Operation contol
D051h	COMP_DATA1_H	Compensation flow rate value
D052h	COMP_DATA1_L	
D061h	TMP_H	Internal temperature
D062h	TMP_L	

The D6F-□□□□D-000-□ MCU is implemented via the interface configuration register.
To access the register, write the internal register address to be accessed in the interface configuration registers 00h and 01h.



I²C interface configuration resister image

Table 5 Interface configuration resister

Configuration Address	Function	Remark
00h	Access address 1 (upper byte)	Upper byte of access address
01h	Access address 2 (lower byte)	lower byte of access address
02h	Serial control	Write/Read access control
03h	Write buffer 0	Write data to access address
04h	Write buffer 1	Write data to access address+1
05h	Write buffer 2	Write data to access address+2
06h	Write buffer 3	Write data to access address+3
07h	Read buffer 0	Read data from access address
08h	Read buffer 1	Read data from access address+1
09h	Read buffer 2	Read data from access address+2
0Ah	Read buffer 3	Read data from access address+3
0Bh	Initialize	
0Dh	Power sequence	Hardware reset function control

Upper byte: Indicates the bits [15: 8] of the 16-bit data.
Lower byte: Indicates the bits [7:0] of the 16-bit data

8.2.1 Access Address Resister (00h – 01h)

The access address resister is used to access the internal register block including the sensor register map, ADC register map and internal memory map. This register specifies the start address because the address is automatically incremented when transferring multiple bytes.

Table 6 Access Address Resister

Address	MSB D7	D6	D5	D4	D3	D2	D1	LSB D0
00h	A15	A14	A13	A12	A11	A10	A9	A8
01h	A7	A6	A5	A4	A3	A2	A1	A0

8.2.2 Serial Control Resister (02h)

The serial control register has various bits to control serial access.

Table 7 Serial control resister (02h)

Address	MSB D7	D6	D5	D4	D3	D2	D1	LSB D0
02h	D_byte_ cnt[3]	D_byte_ cnt[2]	D_byte_ cnt[1]	D_byte_ cnt[0]	Req	R_WZ	Acc_ctl[1]	Acc_ctl[0]

- Acc_ctl [1：0] – Access control bit:
  0 0 = 16 bit access address (A15-A0) (Internal ROM or Resister)
  0 1 = reserved
  1 0 = reserved
  1 1 = reserved
- R_WZ – Read/ Writer selection bit
  0 = Write access
  1 = Read access
- Req- Request bit
  0 = Previous access request completed
  1 = New request.
  When the internal serial bus bridge control circuit completes request processing, this Req flag is cleared to “0”. In the case of a write access request, the internal bus bridge control circuit transfers the data in the write buffer to the register specified by the access address. For read access, the internal bus bridge control circuit stores data at the specified address in the read buffer.
- D_byte_cnt [3：0]
  Transfer byte number specification 1, 2, 3 and 4 byte transfer only are supported.

8.2.3 Write Buffer Resister (03h – 06h)

Address 03h – 06h is write buffers
The host microcomputer can write 1 byte to the register in the following two ways.

From host microcomputer to configuration register with 00h as start address
Data [0] is Address = A [15: 0] by burst write with A [15: 8], A [7: 0], 18h, data [0].
You can write to the following registers.

Alternatively, 1-byte write access is possible by executing the serial configuration register four times as follows.

- Write access address A [15: 8] to address 00h of serial configuration register
- Write access address A [7: 0] to address 01h of serial configuration register
- Write data [0] of write data to address 03h of serial configuration register
- Write 18h to address 02h of serial configuration register (1 byte new write)

[Supplement] When 02h in the serial configuration register is read, if bit [3] is “0”, write access is completed.
For details, refer to Section 8.2.7 I^2C Access Command Example.

8.2.4 Read Buffer Resister (07h – 0Ah)

Address 07h to 0Ah is Read bussers
The hos microcomputer can read out to the internal resister value by following two ways.

When the read request processing of the internal serial bus controller is completed, the request flag is cleared to “0”, and the value of the read target register is stored in the read buffer at address 07h – 0Ah of the configuration register.
In the case of 1-byte read, the host microcomputer issues a read request by performing a burst write with A [15: 8], A [7: 0], and 1Ch to the configuration register with 00h as the start address. After that, the host microcomputer waits for the request bit to be cleared to “0”, and reads the read buffer at address 07h of the configuration register to read the value of the read target register.

Alternatively, one byte can be read by executing the following four times in the serial configuration register.

- Write access address A [15: 8] to address 00h of serial configuration register
- Write access address A [7: 0] to address 01h of serial configuration register
- Write 1ch to address 02h of serial configuration register (1 byte new read)

[Supplement] When 02h in the serial configuration register is read, read access completion is indicated if bit [3] is “0”, and the value in the target register can be read by reading 07h.
For details, refer to Section 8.2.7 I^2C Access Command Example.

8.2.5 Initialize resister (0Bh)

After power on. It is required to load NVM trimming data.

- Write 00h to the initialization register (0Bh)

For details, refer to Section 8.2.7 I^2C Access Command Example.

8.2.6 Power sequence resister (0Dh)

Monitoring the status of the sensor.

Table 8 Power sequence resister

Address	MSB D7	D6	D5	D4	D3	D2	D1	LSB D0
0Dh	Hard_Reset	ADC_state	ADC_state	ADC_state	Pwr_seq_state5	Pwr_seq_state5	Pwr_seq_state5	Pwr_seq_state5

Table 9 State of Power sequence

Bit	Name	R/W	Description
[3:0]	Pwr_seq_state5	R	Indicates the state status value of the power sequence. h0(0000b): Idle h2(0010b): Active h9(1001b): Execute
[6:4]	ADC_state	R	Indicate the ADC control state
[7]	Hard_Reset	R/W	1 - > Perform the hardware reset （Auto clear after execution） 0 - > Don't perform the hardware reset

After initialization or at power reset: h0
Write 06h to D040 : h2
During execution: h9

When a hardware reset is performed, the hardware reset bit is automatically cleared to "0" after a reset, the internal register returns to the default value, and the internal trimming value is reloaded from nonvolatile memory. This hardware reset function is similar to a power reset.

Note:
Access to this reset bit does not require a password to unlock or access the device.
When using hardware reset, set bits 0 to 6 to 0.

8.2.7 I²C Access command example

The following shows an example of I²C access command. The blue colored ACK and read data parts are the output data of the flow sensor. The other uncolored parts are the parts controlled by the host microcomputer.

I²C Command Examples

・I²C Command：I²C write

Start address of configuration resister

Set address, 00h and 01h of the configuration resister

START	Slave Address	ACK	Access Address	ACK	Reg Address H	ACK	Reg Address L	ACK
S	D8h (6Ch (7b)+ 0)	A	00h	A	D0h	A	40h	A

Serial Ctrl	ACK	Write Data	ACK	STOP
18h	A	06h	A	P

Set data, 02h and 03h of the configuration resister

・I²C Command：I²C read

Address of the configuration resister (Read Buffer 0)

START	Slave Address	ACK	Access Address	ACK
S	D8h (6Ch (7b)+ 0)	A	07h	A

Configuration register Reading value from "07h" and "08h" (Read Buffer 0 / Read Buffer 1)

Re-Start	Slave Address	ACK	Read Data H	ACK	Read Data L	ACK	STOP
RS	D9h (6Ch (7b)+ 1)	A	xxh	A	xxh	NA	P

8.3 Resister detail information

The sensor module's internal memory and register access are accessed via configuration registers.

Note: Bits marked "None" should have the default value.

8.3.1 Sensor control resister (D040h)

Table 10 SENS_CTRL Resister

アドレス	MSB D7	D6	D5	D4	D3	D2	D1	LSB D0
D040h						MS	DV_PWR [1]	DV_PWR [0]
Write Access	None	None	None	None	None	Host＆MCU	Host＆MCU	Host＆MCU
Default	0	0	0	0	0	0	0	0

- DV_PWR[1：0] – Main device power mode setting
  0 0 = Stand by – All blocks are powered down
  1 0 = MCU ON – It is necessary to operate the MCU block. The power to the analog and memory units is turned on, and clock supply to the MCU is started.
  Supplement: Do not change the state of this register during measurement.
- MS – MCU Start – MCU mode is executed depending on the status of DV_PWR.
  0 = STOP The measurement processing sequence stops and each block is turned off.
  1 = START The MCU clock supply starts and the MCU mode according to the setting is executed.

8.3.2 CRC Operation control resister (D049h)

Table 11 INT_CTRL

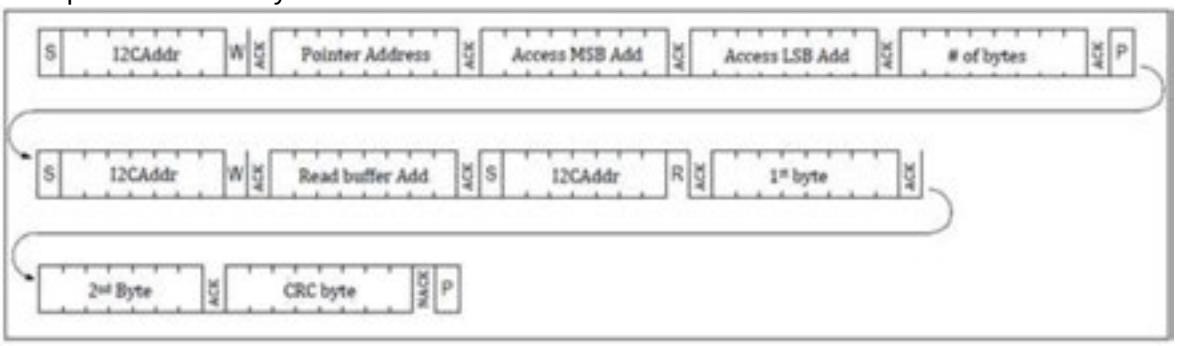
Address	MSB D7	D6	D5	D4	D3	D2	D1	LSB D0
D049h							CRC_EN	
Write Access	NONE	NONE	NONE	NONE	NONE	NONE	Host & MCU	NONE
Default	0	0	0	0	0	0	1	0

- CRC_EN – CRC check function selector （For details of CRC operation, refer to the following.）

  0 = Disable the CRC check function

  1 = Enable CRC check function

Supplementary explanation on CRC operation
CRC is used as an error detection method in data communication.
Omron uses CRC8 polynomial x^8+x^5+x^4+1 for our flow sensor. The following is an example of a 2-byte read I^2C access when the CRC function is enabled.

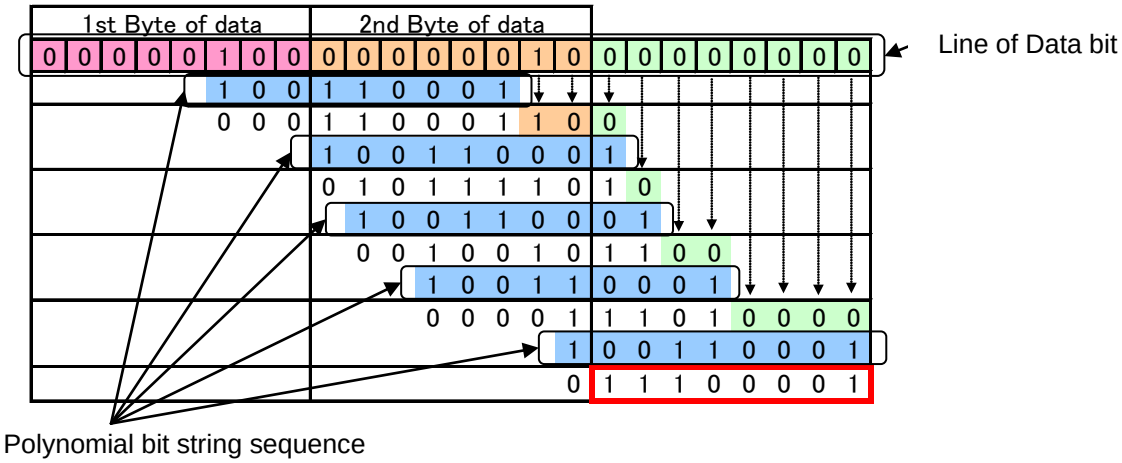


・Bit unit CRC-8 calculation method

1 Arrange the data bit string in a line.
2 Arrange the polynomial bit sequence below the data bit sequence.
3 If the leftmost bit of the data bit string is “0”, shift the polynomial bit string one bit to the right. If the leftmost bit of the data bit string is "1", an XOR operation is performed on the polynomial bit string. Then, shift the polynomial bit sequence one bit to the right.
4 Repeat steps 1 to 3 above until the polynomial bit string reaches the right end of the data bit string

The following is an example of CRC byte calculation based on XOR operation.

	hex	bin
1st Byte of data	04h	00000100
2nd Byte of data	02h	00000010
Polynomial (x^8+ x^5+ x^4+ 1)	131h	100110001
CRC-byte checksum	225h	11100001



8.3.3 Data resister group (D051h-D068h)

Table 12 16bit Data resister map

Address	Name	MSB D7	D6	D5	D4	D3	D2	D1	LSB D0	Remark
D051h	COMP_DATA1_H	DATA<15:8>								Compensated flow rate data
D052h	COMP_DATA1_L	DATA<7:0>								
D061h	TMP_H	DATA<15:8>								Internal temperature
D062h	TMP_L	DATA<7:0>								

See section 8. Output data description for details.

9 Description output data

Measured values are stored in pre-specified registers. Measured data is stored in a dedicated register. Each dedicated register stores the measurement target sensor output signal separately in the upper byte and the lower byte.
For example, the corrected flow rate value is 16 bits unsigned with COMP_DATA1_H and COM_DATA1_L concatenated.

9.1 Data alignment

Measurement data is expressed as 16-bit data. This data is also stored in the flow sensor register group in big endian format as continuous 2-byte data.
In big-endian format, the upper byte of data is stored in the lower address register and the lower byte of data is stored in the upper address register.

9.2 Contents of resister

・COMP_DATA1_H and COMP_DATA1_L [D051h – D052h] : Compensated flow rate (unsigned)

These registers store flow rate data for the selected correction method.

*Fc [L/min] = (Fv – 1024)*RANGE / 60000*

Fc is the converted flow data (L / min), and Fv is the value stored in the flow data register.

・TMP_H and TMP_L [D061h – D062h] : Temperature data (signed)

These registers contain the temperature data of the ASIC internal temperature sensor.
Use the following equation to convert the register readings to temperature data.

Tv [℃] = (Rv – 10214) / 37.39

Tv is the converted temperature data (° C), Rv is the value stored in the temperature data register

Attention: Please use temperature data only as a reference value. The temperature accuracy is not guaranteed in the specifications.

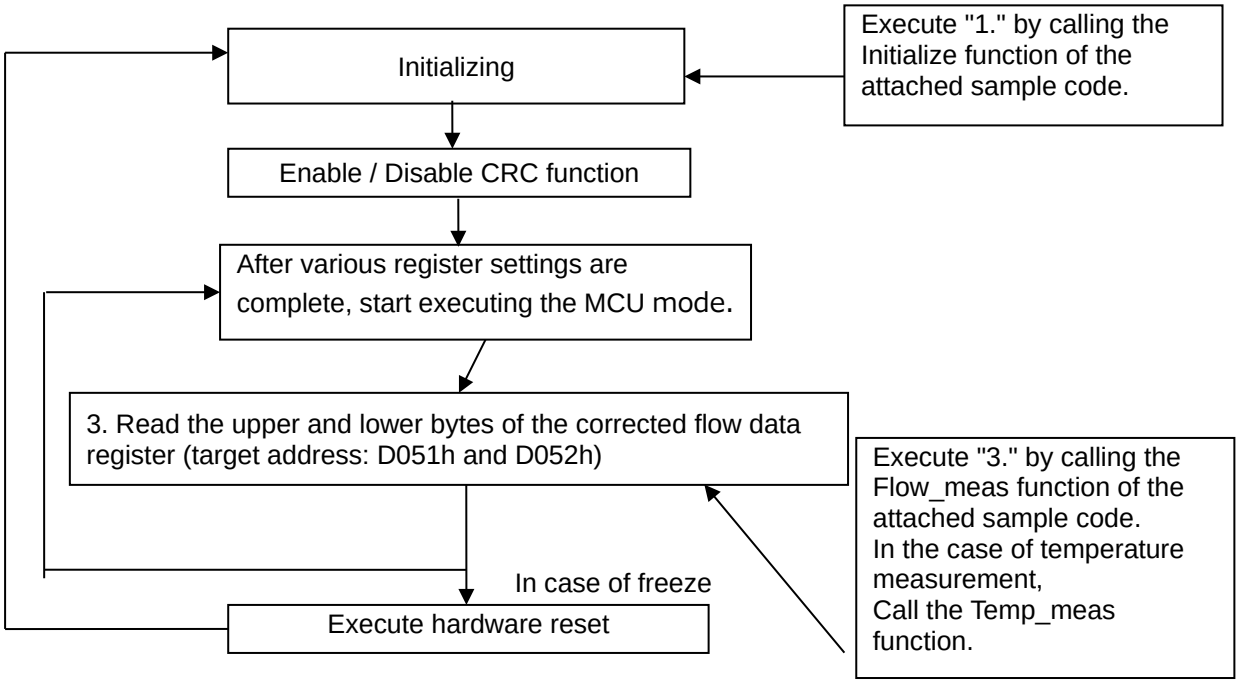
9.3 Example of Sensor data

The following table shows an example of flow sensor register readings at a given flow rate and temperature. The values listed in this table are assumed to be fully compensated by the device. (Ie no offset, no gain error)

Table 13 Temperature data register value and temperature value comparison

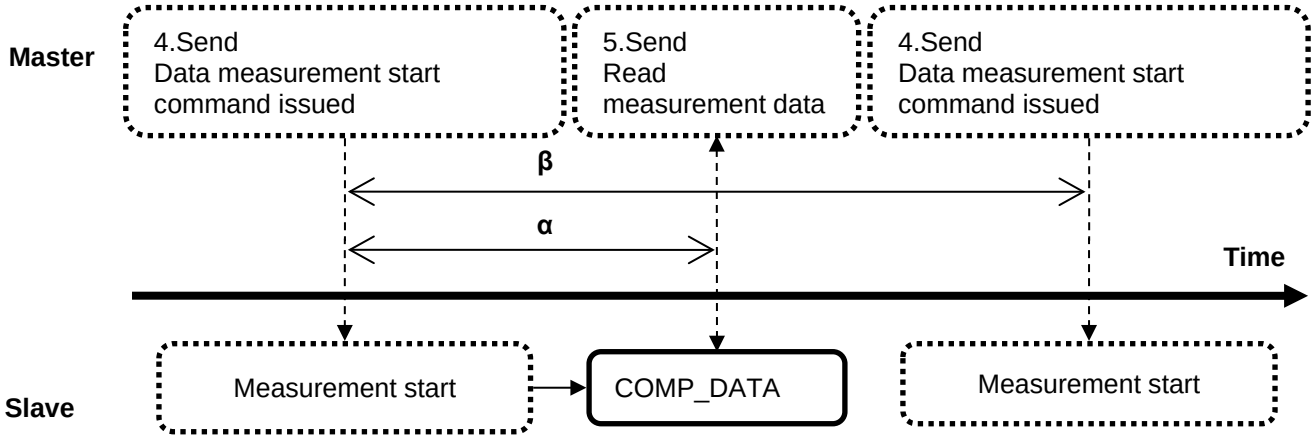
Resister address		resister value		temperature
TMP_H	TMP_L			
D061h	D062h	HEX	DEC	
2Bh	8Dh	2B8Dh	11149	25.0 ℃
2Eh	FFh	2EFFh	12031	48.6 ℃
26h	BBh	26BBh	9915	-8.0 ℃

10 Sensor operation flow chart



・Regarding communication waiting time

Items	Symbol	Remark
Response time	α	α ≧ 120 ms
Sampling period	β	β > α



10.1 I²C instruction in sensor operation

Initialization processing after power on

I²C command: Write 00h to the initialization register (0Bh) to load NVM trimming data. However, please keep the MCU in the non-reset state.
Execute this command 200 µs after power on.

Table 14 Initialization

START	Slave Address	ACK	Access Address	ACK	Write Data	ACK	STOP
S	D8h (6Ch (7b)+ 0)	A	0Bh	A	00h	A	P

* If you use the CRC operation function, issue a control command referring to page 14.

MCU mode execution start after various register settings are completed

By writing 06h to the sensor control register (D040h), the MCU mode is executed with the various settings required to perform the corrections outlined in Section 6. By reading this register after writing 06h to the sensor control register, it is possible to read out the state of the MUX selected by the MCU. After processing, the MS bit is set to “0”.

[Caution]: Do not access the device while the MCU is operating. It is all right if write and read access are made after 120 msec.

I²C command: Write 06h to sensor control register (D040h) (MS = 1 & MCU_on)

Table 15 MCU mode execution processing

START	Slave Address	ACK	Access Address	ACK	Reg Address H	ACK	Reg Address L	ACK
S	D8h (6Ch (7b)+ 0)	A	00h	A	D0h	A	40h	A

Serial Ctrl	ACK	Write Data	ACK	STOP
18h	A	06h	A	P

Read the upper and lower byte data registers of the corrected flow register (D051h and D052h)

After initialization, the first data is not flow data. Please process invalid.

I²C command: Write 2Ch (2-byte read access) to the serial configuration register (address 2h) to read the corrected flow rate data registers (D051h and D052h).

Table 16 Read out compensated flow rate data（Step1）

START	Slave Address	ACK	Access Address	ACK	Reg Address H	ACK	Reg Address L	ACK
S	D8h (6Ch (7b)+ 0)	A	00h	A	D0h	A	51h	A

Serial Ctrl	ACK	STOP
2Ch	A	P

I²C command: Read 2 bytes of flow rate data register after correction through read buffer 0 (address 7h) and read buffer 1 (address 8h).

Table 17 Read out compensated flow rate data（Step2）

START	Slave Address	ACK	Access Address	ACK
S	D8h (6Ch (7b)+ 0)	A	07h	A

Re-Start	Slave Address	ACK	Read Data H	ACK	Read Data L	ACK	STOP
RS	D9h (6Ch (7b)+ 1)	A	xxh	A	xxh	NA	P

10.2 CRC operation function activation method

Set bit 1 of the interrupt control register to "1"

I²C command: Write 02h to the interrupt control register (D049h) (CRC_EN = 1)

Table 18 CRC Operation function activation

START	Slave Address	ACK	Access Address	ACK	Reg Address H	ACK	Reg Address L	ACK
S	D8h (6Ch (7b)+ 0)	A	00h	A	D0h	A	49h	A

Serial Ctrl	ACK	Write Data	ACK	STOP
18h	A	02h	A	P

10.3 Reset execution method

There is a way to reset if the sensor freezes for some reason.

MCU Reset

Set bit 2 of initialization register 0Bh to “1”. This stops the MCU execution and returns to the Active state. When the value of the power sequence register is 9 it may be solved by this process.

I²C command: Write 04h to the initialization register (0Bh)

Table 19 MCU Reset execution method

START	Slave Address	ACK	Access Address	ACK	Write Data	ACK	STOP
S	D8h (6Ch (7b)+ 0)	A	0Bh	A	04h	A	P

After MCU reset is executed, bit 2 of the initialization register is automatically cleared to "0".

Soft Reset

Set bit 3 of initialization register 0Bh to “1”. This returns the sensor to its power on state. When the value of the power sequence register is other than 9, 2 or 0, this process can return to Idle.

I²C command: Write 08h to initialization register (0Bh)

Table 20 Soft reset

START	Slave Address	ACK	Access Address	ACK	Write Data	ACK	STOP
S	D8h (6Ch (7b)+ 0)	A	0Bh	A	08h	A	P

After MCU reset is executed, bit 3 of the initialization register is automatically cleared to "0".

Hard Reset

Set bit 7 of the power sequence register to “1”. This returns the sensor to its power-on state.

I²C command: Write 80h to the power sequence register (0Dh) (Hard_Reset = 1)

Table 21 Hard reset

START	Slave Address	ACK	Access Address	ACK	Write Data	ACK	STOP
S	D8h (6Ch (7b)+ 0)	A	0Dh	A	80h	A	P

After MCU reset is executed, bit 7 of the initialization register is automatically cleared to "0".

10.4 I²C BUS

This bus is intended for communication between different ICs. There are two signal lines: bi-directional data signal (SDA) and clock signal (SCK). Both SDA and SCL signal lines must be connected to the supply voltage through pull-up resistors. The following protocols are defined:

- Data transfer can only be started when the bus is not busy.
- During data transfer, the data signal must be stable when the clock signal is “H”.
- If the data signal changes while the clock signal is "H", it is interpreted as a control signal.

Therefore, the following bus states are defined:

BUS non-busy state

Both data signal (SDA) and clock signal (SCL) are “H”.

Start of data transfer

The state in which the data signal (SDA) changes from "H" to "L" while the clock signal (SCL) is "H" is defined as the start condition.

Stop of data transfer

The state in which the data signal (SDA) changes from "L" to "H" while the clock signal (SCL) is "H" is defined as the stop condition.

Valid data

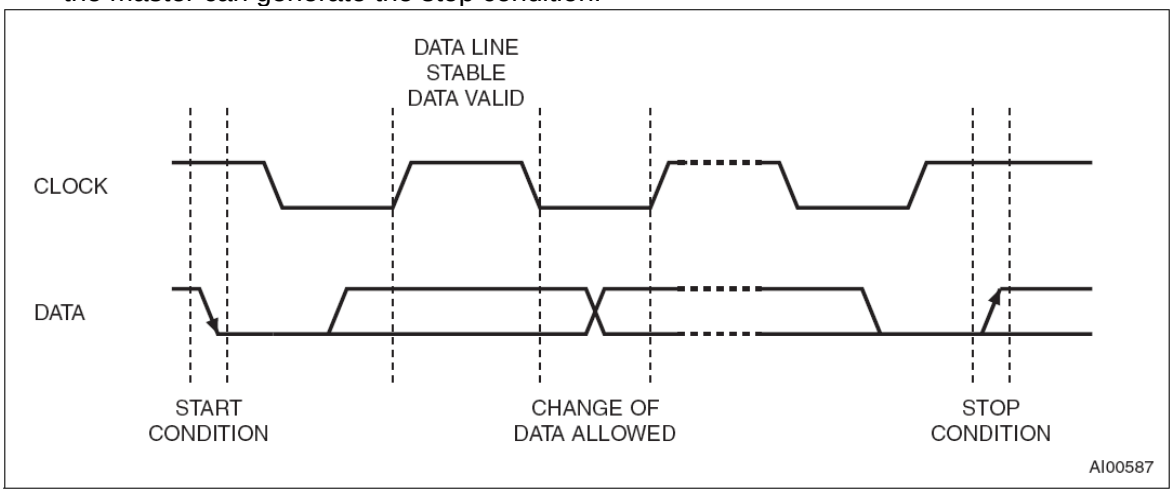
After the start condition, the data signal (SDA) becomes valid. The data signal is stable when the clock signal is “H”. The data signal will change while the clock signal is “L”. It is 1 bit data per clock. Each data transfer is initialized by the start condition and ended by the stop condition. There is no limit on the number of bytes transferred between the start and stop conditions. Each transfer data consists of byte unit data and receiver acknowledge as 9th bit data.

The side that sends a message is called the "transmitter" and the device that receives the message is defined as the "receiver". The side that controls the message is called the "master". Also, the side controlled by the master is called "slave".

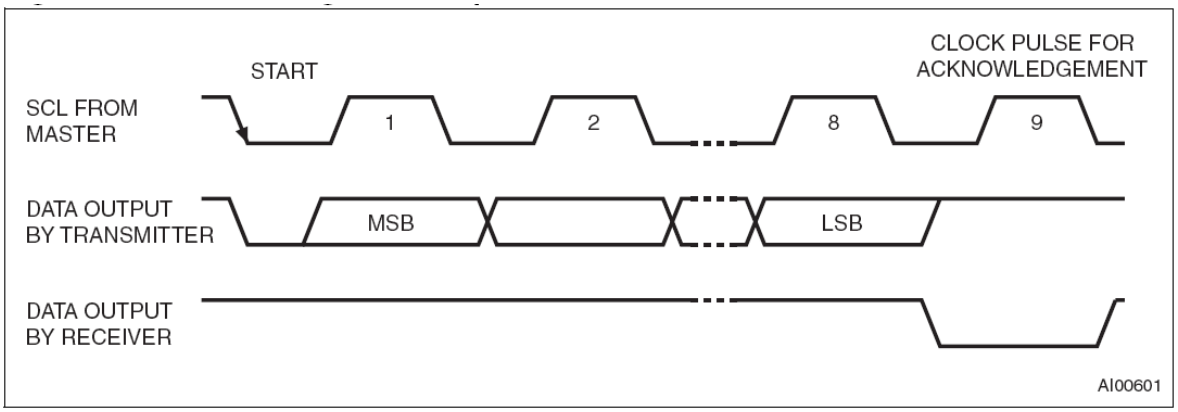
Acknowledge

Following each 8 bits or 1 byte, it becomes 1 acknowledge bit. This acknowledge bit is established by driving the bus level to “L” at the receiver while the master is generating clock pulses for the acknowledge bit. The designated slave receiver must issue an acknowledge bit after receiving each byte of clock synchronized data.

The slave device must drive the SDA line stably to "L" in the "H" period of the clock at a clock cycle corresponding to the acknowledge bit. Of course, you need to consider the setup and hold time. The receiving master must issue a non-acknowledgement after the last byte output from the sending slave. In this case, the sending side should be kept “H” so that the master can generate the stop condition.



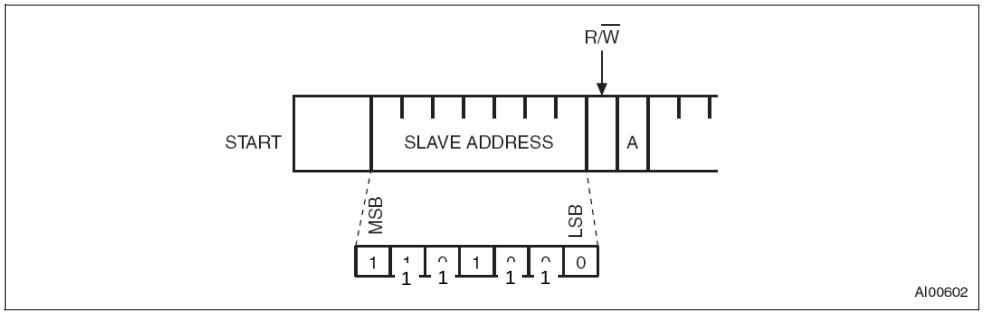
Serial bus data transfer sequence



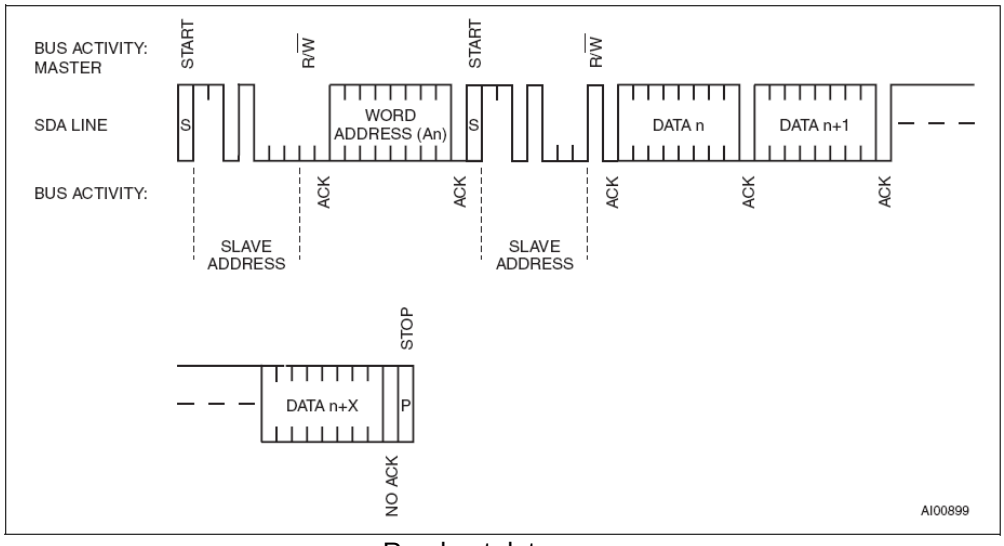
Acknowledge sequence

I²C Read out mode

In this mode, the master reads the digital flow sensor slave after setting the slave address (see below figure). Following the write mode control bit (R / W = 0) and the acknowledge bit, the word address "An" is set to the address pointer inside the sensor chip. Next, the start condition and slave address are issued again, followed by the read mode control bit (R / W = 1). At this point the master transmitter will be the master receiver. The byte data of the specified address is transmitted, and the master receiver issues an acknowledge to the slave transmitter. The address pointer is incremented only when the clock corresponding to the acknowledge bit is received. The digital flow sensor as a slave transmitter sends the data at address An + 1 to the data line, the master receiver receives this data and issues an acknowledge, and recognizes it as the next byte, and the address pointer is incremented to An + 2.



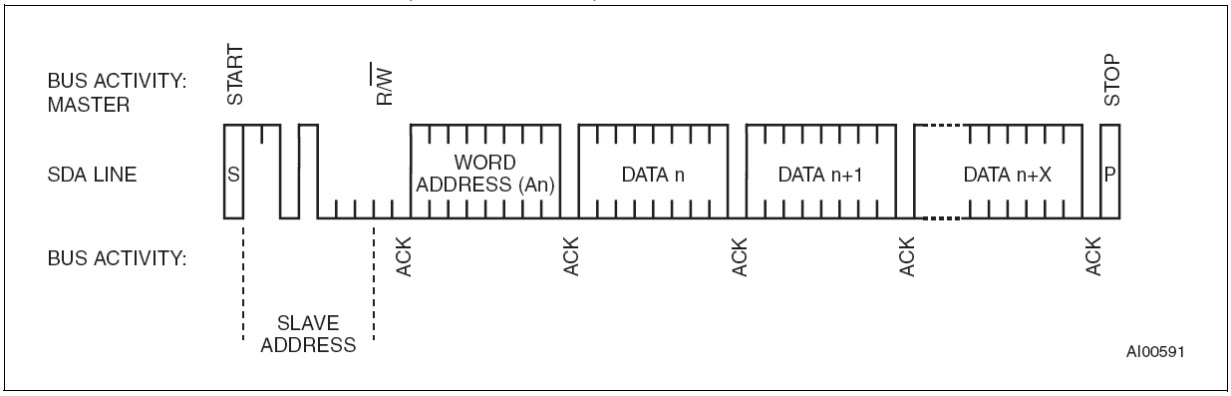
Slave address layout



Read out data sequence

I^{2}C Write mode

In this mode, the master transmitter transfers data to the digital flow sensor slave receiver. The bus protocol is shown in below figure. Following the start condition and slave address, put the bus in the logic "0" (R / W = 0) state. "An" is written to the internal address pointer of the device of the specified address by this. The word data to be written to the internal memory is then loaded onto the bus, and the internal address pointer is incremented to indicate the next address when the clock corresponding to the acknowledge bit is received. As a slave receiver, the digital flow sensor issues an acknowledge to the master transmitter each time a slave address, word address, or write data is received.



Read mode sequence

11 Word

- I^2C (Inter – Integrated Circuit)
  Serial communication protocol were proposed by Philips.
  It consists of only two bus lines: serial data line (SDA) and serial clock line (SCL). Each device connected to this bus has its own address (slave address), and based on that, it is possible to specify each device by software of the controlling host MCU.
  For more information on I^2C, please refer to the UM10204 document on NXP Semiconductors' website.

- Slave address
  Indicates the "address" of the other party in I^2C communication.
  It is usually expressed by 7 bits, and it is "110_1100b" or "6Ch" in our digital flow sensor. In actual communication, 1 bit is added to the least significant bit, which means write / read, to become 8 bits. When writing, “0” is added to “1101_1000b” or “D8h”. When reading, “1” is negatively added to “1101_1001b” or “D9h”. Whether the slave address is represented by 7 bits or 8 bits in software differs depending on the controlling host MCU used by the customer. Please confirm the I^2C communication software library specifications of the control side MCU to be used.

12 Sample source code

The following sample source code is for D6F-□□□□D-000-□ when using an STM32 microcontroller. The I²C control part needs to be modified according to the microcontroller you use.

12.1 D6F_D_Sample.h

```
/*=============================================*/
/* D6F Digital Type Flow Sensor Header file (using STM32)
  * :Copyright: (C) OMRON Corporation, Microdevice H.Q.
  * :Auther     : Kenichi Handa
  * :Revision:   $Rev$
  * :Id:         $Id$
  * :Date:       $Date$
  *
  * All Rights Reserved
  * OMRON Confidential
  * OMRON Proprietary Right
  *=============================================*/
/*==   for General          ==*/
#define SA_7          0x6F // for 7bit Slave Address
#define RANGE_MODE 100    // Full Range 100[L/min]
/*==   for Measure Mode ==*/
#define F          1    // Flow rate mode
#define T          2    // Temperature mode
/*==   for CRC & HW_reset ==*/
#define ON         1    // CRC ON / HW_reset execute
#define OFF        2    // CRC OFF / HW_reset release

/* Function prototypes -----------------------------------------------------*/
void    Initialize( void );
short Flow_meas( void );
short Temp_meas( void );
int     CRC_enable(char mode);
void    HW_reset(char mode);
unsigned char GetCRC8(unsigned char data[], unsigned char num);
/*    Private Functions -------------------------------------------------------*/
int     I2C_WR(unsigned char add, char *dbuf, unsigned char n);
uint8_t I2C_RD_8(unsigned char add, char *dbuf, unsigned char n);
short I2C_RD_16(unsigned char add, char *dbuf, unsigned char n);
unsigned short I2C_RD_u16(unsigned char add, char *dbuf, unsigned char n);
```

12.2 D6F_D_Sample.c

```
/*=============================================*/
/* D6F Digital Flow Sensor Sample Code (using STM32)
  * :Copyright: (C) OMRON Corporation, Microdevice H.Q.
  * :Auther     : Kenichi Handa
  * :Revision:   $Rev$
  * :Id:            $Id$
  * :Date:         $Date$
  *
  * All Rights Reserved
  * OMRON Confidential
  * OMRON Proprietary Right
  *=============================================*/
#include "stm32f10x_i2c.h"
#include "D6F_D_Sample.h"

#define I2C1_SCL_PIN                        GPIO_Pin_6
#define I2C1_SDA_PIN                        GPIO_Pin_7
#define   POLYNOMIAL_CRC8 0x131

typedef unsigned char     uint8;
typedef unsigned short    uint16;
typedef unsigned long     uint32;

              short    RD_FIFO;       /* 16bit data width */
unsigned short uRD_FIFO;       /* 16bit data width */
              uint8_t RD_REG;       /*    8bit data width */
              char    setting_done_flag = 0;
              char    CRC_on = 0;    /* CRC function flag */
              uint8_t det_crc;

// Dummy wait routine
void adc_wait(volatile unsigned long delay)
{
             while(delay) delay--;
}

/*=============================================*/
/* Initialize Function                                    */
/* Usage             : Initialize( void )                 */
/* Argument        : Null                                      */
/* Return value : Null                                    */
/*=============================================*/
void Initialize( void )
{
```

```
    /* EEPROM Control <= 00h */
    char send1[] = {0x0B, 0x00};
    I2C_WR(SA_7, send1, 2);
}

/*=====================================================*/
/* Flow rate measure Function                          */
/* Usage             : Flow_meas( void )               */
/* Argument          : NULL                            */
/* Return value : Compensated Flow rate value(short)   */
/*=====================================================*/
short Flow_meas(void)
{
    short rd_fifo;
    short rd_flow;
    unsigned long   wait_time;
    unsigned char   calc_crc8;
    uint8_t rd_data[2] = {0, 0};

    calc_crc8 = 0; /* initialize */
    /* [D040] <= 06h */
    char send2[] = {0x00, 0xD0, 0x40, 0x18, 0x06};
    I2C_WR(SA_7, send2, 5);

    /* wait time depend on resolution mode */
    wait_time     = 120; /* 120msec wait */
    adc_wait(wait_time);

    /* [D051/D052] => Read Compensated Flow value */
    char send3[] = {0x00, 0xD0, 0x51, 0x2C, 0x07};
    uRD_FIFO = I2C_RD_u16(SA_7, send3, 5);

    if (CRC_on){
       rd_data[0] = (uRD_FIFO >> 8) & 0xFF;
       rd_data[1] = (uRD_FIFO       ) & 0xFF;
       calc_crc8 = GetCRC8(rd_data, 2);

       if(calc_crc8 != det_crc){ /* CRC value mismatch !! */
          /* Please type your error routine */
       }
    }
    rd_flow = ((uRD_FIFO - 1024) * RANGE_MODE *10/ 60000); /* convert to [L/min](x10) */
    return rd_flow;
}
```

```
/*=====================================================*/
/* Temperature measure Function                        */
/* Usage            : Temp_meas()                      */
/* Argument         : NULL                             */
/* Return value : x10 Temperature                      */
/*=====================================================*/
short Temp_meas(void)
{
    short       rd_temp;
    unsigned long wait_time;

    /* [D040] <= 06h */
    char send2[] = {0x00, 0xD0, 0x40, 0x18, 0x06};
    I2C_WR(SA_7, send2, 5);

    /* wait time depend on resolution mode */
    wait_time    = 50 + 15; /* 65msec wait */
    adc_wait(wait_time);

    /* [D061/D062] => Read TMP_H/TMP_L value */
    char send3[] = {0x00, 0xD0, 0x61, 0x2C, 0x07};
    RD_FIFO = I2C_RD_16 (SA_7, send3, 5);

    rd_temp = ((RD_FIFO -10214)*1000 / 3739); // convert to degree-C(x10)
    return rd_temp;
}

/*=====================================================*/
/* CRC Control Function                                */
/* Usage            : CRC_enable(ON)                   */
/* Argument1        : ON    : CRC function enable      */
/*                    : OFF : CRC function disable     */
/* Return value :                                      */
/*=====================================================*/
int CRC_enable(char mode)
{
    uint8_t       int_reg_rd;
    uint8_t       int_reg_mod;

    /* [D049] => Pre-Read INT_CTRL Reg value */
    char send1[] = {0x00, 0xD0, 0x49, 0x1C, 0x07};
    int_reg_rd = I2C_RD_8(SA_7, send1, 5);

    switch(mode){
       case ON:int_reg_mod = int_reg_rd | 0x02; /* set INT_CTRL[1] */
```

```
                /* [D049][2] <= 1:CRC ON Mode */
                char send2[] = {0x00, 0xD0, 0x49, 0x18, int_reg_mod};
                I2C_WR (SA_7, send2, 5);
                CRC_on = 1;
        break;
        case OFF:int_reg_mod = int_reg_rd & 0xFD; /* clear INT_CTRL[1] */
                /* [D049][2] <= 0:CRC OFF Mode */
                char send3[] = {0x00, 0xD0, 0x49, 0x18, int_reg_mod};
                I2C_WR (SA_7, send3, 5);
                CRC_on = 0;
        break;
        default:
                return 1; /* illegal argument detect */
        break;
    }
    return 0; /* normal finish */
}

/*=====================================================*/
/* HW reset Control Function                           */
/* Usage            : HW_reset(ON)                     */
/* Argument1        : ON    : HW reset execute         */
/*                  : OFF : HW reset release           */
/* Return value :                                      */
/*=====================================================*/
void HW_reset(char mode)
{
    uint8_t      pow_reg_mod;

    switch(mode){
        case ON :pow_reg_mod = 0x80;/* Power Sequence Register <= 80h */
        break;
        case OFF:pow_reg_mod = 0x00;/* Power Sequence Register <= 00h */
        break;
        default :return 1; /* illegal argument detect */
        break;
    }

    char send1[] = {0x0D, pow_reg_mod};/* Power Sequence Register <= 80h */
    I2C_WR(SA_7, send1, 2);
}
```

```
/*=====================================================*/
/* CRC8 calculation Control Function                   */
/* Usage            : GetCRC8(data, num)               */
/* Argument1        : data : taget data for CRC8 calculate */
/* Argument2        : num  : size of data              */
/* Return value : CRC value of data                    */
/*=====================================================*/
unsigned char GetCRC8(unsigned char data[], unsigned char num)
{
   unsigned char crc = 0;
   unsigned char bytcnt;
   unsigned char bitcnt;

   for (bytcnt = 0; bytcnt < num; bytcnt++){
      crc ^= (data[bytcnt]);

      for (bitcnt = 8; bitcnt > 0; bitcnt--){
         if (crc & 0x80){
            crc = (crc << 1) ^ POLYNOMIAL_CRC8;
         }
         else{
            crc = (crc <<1);
         }
      }
   }
   return crc;
}
```

Please check each region's Terms & Conditions by region website.

OMRON Corporation

Electronic and Mechanical Components Company

Regional Contact

Americas
https://www.components.omron.com/

Europe
http://components.omron.eu/

Asia-Pacific
https://ecb.omron.com.sg/

China
https://www.ecb.omron.com.cn/

Korea
https://www.omron-ecb.co.kr/

Japan
https://www.omron.co.jp/ecb/

© OMRON Corporation 2019-2024 All Rights Reserved.
In the interest of product improvement, specifications are subject to change without notice.

Cat. No. A302-E1-02
1024 (0919)