F&eIT Series

16Bits Isolated
Up-Counter Module
12 - 24VDC type

CNT16-8(FIT)GY

5VDC type

CNT16-8L(FIT)GY

User's Manual

CONTEC CO.,LTD.

Check Your Package

Thank you for purchasing the CONTEC product.

The product consists of the items listed below.

Check, with the following list, that your package is complete. If you discover damaged or missing items, contact your retailer.

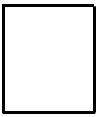
Product Configuration List

- Module (One of the following) ...1
  [CNT16-8(FIT)GY, or CNT16-8 L(FIT)GY]
- First Step Guide …1
- CD-ROM [F&eIT Series Setup Disk] *1 …1
- Interface connector plug ...1

*1 The CD-ROM contains various software and User's Manual (this manual)


Module


Interface connector plug


First step guide


CD-ROM
[F&eIT Series Setup Disk]

Copyright

Trademarks

Table of Contents

1. Before Using the Product

This chapter provides information you should know before using the product.

About the Module

This product is an expansion module (device module) that up-counts pulse signals input from an external device. The product is used in combination with the I/O controller module < CPU-CAxx(FIT)GY > (*1) or microcontroller unit < CPU-SBxx(FIT)GY > (*1) in the F&eIT Series.

Each < CNT16-8(FIT)GY > module can count a maximum of eight pulse signals. The pulse signals are input via opto-couplers. The external power supply can be in the range 12 - 24 VDC.
The < CNT16-8L(FIT)GY > has the same count function as the < CNT16-8(FIT)GY > but operates on a 5 VDC external power supply.

Please read this manual carefully to create application programs and configure the system, such as setting the switches and connecting it to external devices.

*1 The "x" in a model code represents a single digit (or no digit) indicating different products. (The same convention applies below).

Features

- Performs a 16-bit up-count for eight channels.
- Isolated from each other by an opto-coupler, offering good noise immunity.
- A digital filter is included to prevent miscounting due to chattering on the input pulses.
- A rotary switch allows you to set device IDs to help you keep track of device numbers.
- Like other F&eIT series products, the module has a 35mm DIN rail mounting mechanism as standard. A connection to a controller module can be effected on a lateral, stack basis in a unique configuration, which permits a simple, smart system configuration without the need for a backplane board.

Functions and control method by controller connected

The CNT16-8(FIT)GY and CNT16-8L(FIT)GY can be connected to a variety of controllers.

Supported controllers

Microcontroller Unit	: CPU-SBxx(FIT)GY
I/O Controller Module	: CPU-CAxx(FIT)GY
Monitoring & Control Server Unit	: SVR-MMF2(FIT)
Monitoring & Control Server Unit	: SVR-MMF(FIT)GY

Check each controller to which the module can be connected as well as the method of controlling the module when connected to that controller.

Connections to controllers

O: Permitted ×: Not permitted	CPU-SBxx(FIT)GY	CPU-CAxx(FIT)GY	SVR-MMF2(FIT)	SVR-MMF(FIT)GY
CNT16-8(FIT)GY	O	O	O	O
CNT16-8L(FIT)GY	O	O	O	O
Device ID setting range	0 - 7	0 - 7	0 - 7	0 - 7

Control method by controller connected

		CPU-SBxx(FIT)GY	CPU-CAxx(FIT)GY	SVR-MMF2(FIT)	SVR-MMF(FIT)GY
Control using the I/O address map		O			
Control using the memory address map			O		
Control via the Windows driver *	FIT Protocol		O		
	API-CAP(W32)		O		
	API-SBP(W32)	O			
Control over the web (as set from within the browser)				O	O

* The API-SBP(W32) is included in the development kit [DTK-SBxx(FIT)GY]; the other drivers are bundled with each controller.

Control using the I/O address map

When connected to the CPU-SBxx(FIT)GY, the module can receive I/O instructions directly from the controller module. For details, see Chapter 4 “Using the I/O Address Map”.

Control using the memory address map

When connected to the CPU-CAxx(FIT)GY, the module can be accessed from the host computer over the network. The module is assigned with its device ID in the memory managed by the controller module. The application running on the host computer controls the module by reading/writing the memory managed by the controller module. For details, see Chapter 5 “Using the Memory Address Map”.

Control via the Windows driver

For the functions and settings available when using the Windows driver, refer to the reference manual and online help for each module.

Control over the web

You can monitor collected data and manage the log over the web. You can use your familiar browser to easily make various settings. For details, refer to the reference manual for the SVR-MMF2(FIT), SVR-MMF(FIT)GY.

Customer Support

CONTEC provides the following support services for you to use CONTEC products more efficiently and comfortably.

Web Site

Japanese http://www.contec.co.jp/
English http://www.contec.com/
Chinese http://www.contec.com.cn/

Latest product information

CONTEC provides up-to-date information on products.
CONTEC also provides product manuals and various technical documents in the PDF.

Free download

You can download updated driver software and differential files as well as sample programs available in several languages.

Note! For product information

Contact your retailer if you have any technical question about a CONTEC product or need its price, delivery time, or estimate information.

Limited One-Year Warranty

CONTEC products are warranted by CONTEC CO., LTD. to be free from defects in material and workmanship for up to one year from the date of purchase by the original purchaser.
Repair will be free of charge only when this device is returned freight prepaid with a copy of the original invoice and a Return Merchandise Authorization to the distributor or the CONTEC group office, from which it was purchased.
This warranty is not applicable for scratches or normal wear, but only for the electronic circuitry and original products. The warranty is not applicable if the device has been tampered with or damaged through abuse, mistreatment, neglect, or unreasonable use, or if the original invoice is not included, in which case repairs will be considered beyond the warranty policy.

How to Obtain Service

For replacement or repair, return the device freight prepaid, with a copy of the original invoice. Please obtain a Return Merchandise Authorization number (RMA) from the CONTEC group office where you purchased before returning any product.

* No product will be accepted by the CONTEC group without the RMA number.

Liability

The obligation of the warrantor is solely to repair or replace the product. In no event will the warrantor be liable for any incidental or consequential damages due to such defect or consequences that arise from inexperienced usage, misuse, or malfunction of this device.

Safety Precautions

Understand the following definitions and precautions to use the product safely.

Safety Information

This document provides safety information using the following symbols to prevent accidents resulting in injury or death and the destruction of equipment and resources. Understand the meanings of these labels to operate the equipment safely.

⚠ DANGER	DANGER indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
⚠ WARNING	WARNING indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
⚠ CAUTION	CAUTION indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury or in property damage.

Handling Precautions

⚠ CAUTION

- Do not modify the module. CONTEC will bear no responsibility for any problems, etc., resulting from modifying this module.
- Do not use or store the equipment in a hot or cold place, or in a place that is subject to severe temperature changes.
  (Operating temperature range: 0 - 50°C)
- Do not use or store the equipment in a place subject to direct sunlight or near a heating device, such as a stove.
- Do not use or store the equipment in a dusty or humid place.
  (Operating humidity range: 10 - 90%RH, No condensation)
- As this product contains precision electronic components, do not use or store in environments subject to shock or vibration.
- Do not use or store the product near equipment generating a strong magnetic field or radio waves.
- If you notice any strange odor or overheating, please unplug the power cord immediately.
- In the event of an abnormal condition or malfunction, please consult the dealer from whom the equipment was purchased.
- To avoid electric shock, please do not touch the system with a wet hand.
- Do not open the module casing. CONTEC will disclaim any responsibility for equipment whose casing has been opened.
- To prevent damage, please do not subject the module to impact or bend it.
- To prevent contact malfunction, please do not touch the metallic pins on the external module connector.

- The module contains switches that need to be properly set. Before using the module, please check its switch settings.
- To avoid malfunction, please do not change the module switch settings in an unauthorized manner.
- Do not operate the device module when the power for the Controller Module is on.
  To avoid malfunction, please be sure to turn off the power for the Controller Module.

FCC PART 15 Class A Notice

NOTE

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in commercial environment.
This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference at his own expense.

WARNING TO USER

Change or modifications not expressly approved the manufacturer can void the user's authority to operate this equipment.

Environment

Use this product in the following environment. If used in an unauthorized environment, the module may overheat, malfunction, or cause a failure.

Operating temperature

0 - 50°C

Operating humidity

10 - 90%RH (No condensation)

Corrosive gases

None

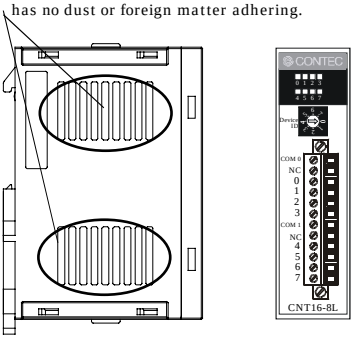
Floating dust particles

Not to be excessive

Inspection

Inspect the product periodically as follows to use it safely.

- Check that the ventilation slit has no obstruction and has no dust or foreign matter adhering.



Storage

When storing this product, keep it in its original packing form.

(1) Put the module in the storage bag.

(2) Wrap it in the packing material, then put it in the box.

(3) Store the package at room temperature at a place free from direct sunlight, moisture, shock, vibration, magnetism, and static electricity.

Disposal

When disposing of the product, follow the disposal procedures stipulated under the relevant laws and municipal ordinances.

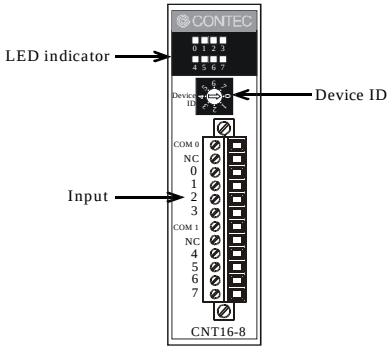
2. Module Nomenclature and Settings

Nomenclature of Module Components

Figure 2.1 shows the names of module components.

In the figure, the indicated switch settings represent factory settings.

CNT16-8(FIT)GY



CNT16-8L(FIT)GY

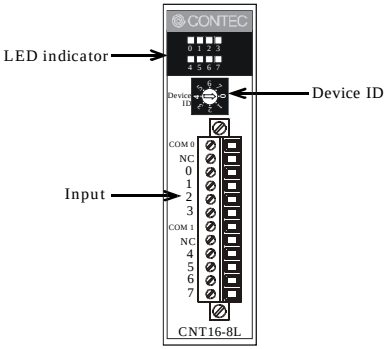


Figure 2.1. Names of Module Components

Setting a Device ID

The controller module distinguishes and keeps track of the modules that are connected to it by assigning device IDs to them. Each module, therefore, should be assigned a unique ID.

A Device ID can be assigned in a 0 - 7 range, so that a maximum of eight modules can be distinguished.

The factory setting for the Device ID is [0].

Setup Method

A Device ID can be set by turning the rotary switch that is located on the module face.
A Device ID can be assigned by turning the switch.

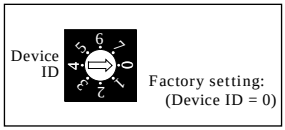


Figure 2.2. Setting a Device ID

LED Indicator

Table 2.1. LED Indicator

Name	Function	LED indicator
Status LED	Input indicator (0 - 7): GREEN	ON: Indicates current flowing in the input pin (Counting pulse signal)
		OFF: Indicates no current flowing in the input pin (Not counting pulse signal)

3. Connecting to an External Device

Interface Connector

How to Connect an Interface Connector

When connecting the Module to an external device, you can use the supplied connector plug.
To wire the Module, strip the sheath about 9 - 10mm from an end of the wire and insert the exposed wire into an opening. Tighten the screw to fasten the inserted wire. Applicable wires are AWG28 - 16.

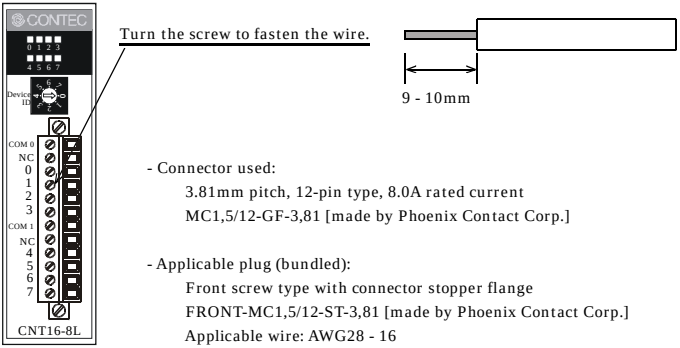


Figure 3.1. Connecting an Interface Connector and Connectors That Can Be Used

⚠ CAUTION

Removing the connector plug by grasping the cable can break the wire.

Signal Layout on the Interface Connector

The Module can be connected to an external device using a 12-pin connector that is provided on the Module face.

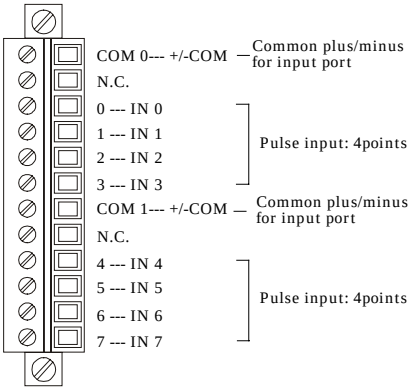


Figure 3.2. Signal Layout on the Interface Connector

External I/O Circuits

Input section

Figure 3.3 or 3.4 shows the input equivalent circuit for the interface section.
The signal input section consists of an opto-isolated input (compatible with both current sink output and current source output). An external power supply is therefore required to drive the input section of this module. The power requirement for the < CNT16-8(FIT)GY > is about 8 mA per input channel at 24 VDC (about 4 mA at 12 VDC) and for the < CNT16-8L(FIT)GY > is about 4 mA per input channel at 5 VDC.

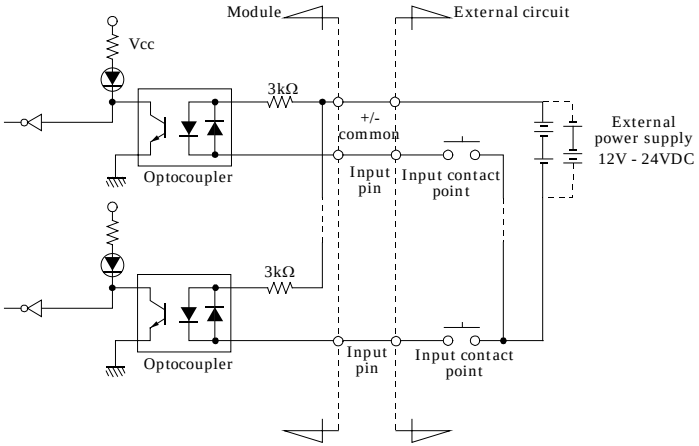


Figure 3.3. Input Circuit < CNT16-8(FIT)GY >

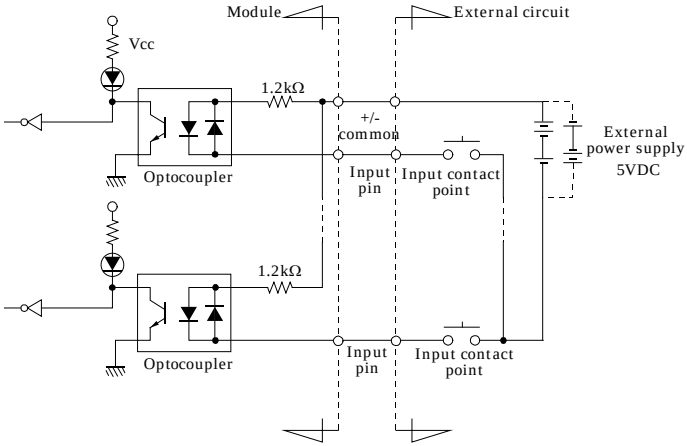


Figure 3.4. Input Circuit < CNT16-8L(FIT)GY >

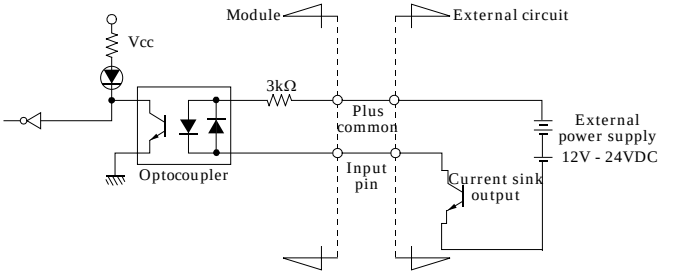


Figure 3.5. Example of a Connection to Current Sink Output < CNT16-8(FIT)GY >

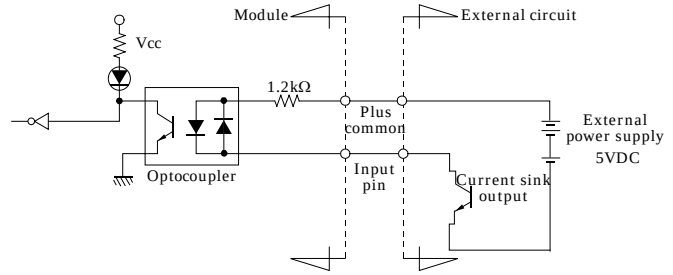


Figure 3.6. Example of a Connection to Current Sink Output < CNT16-8L(FIT)GY >

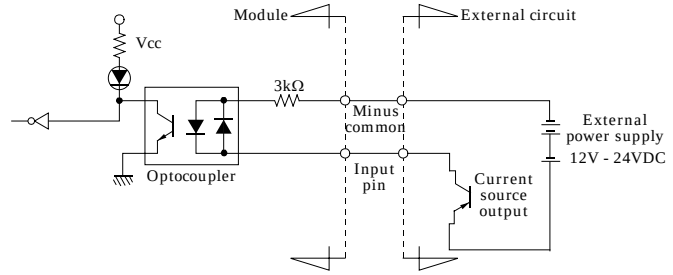


Figure 3.7. Example of a Connection to Current Source Output < CNT16-8(FIT)GY >

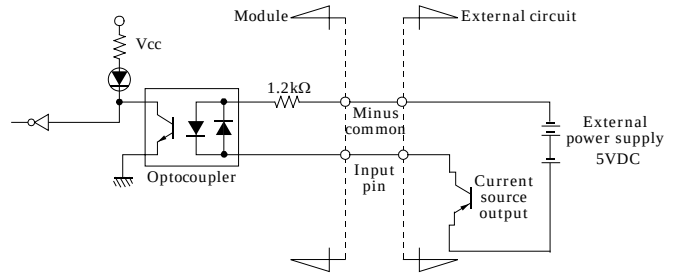


Figure 3.8. Example of a Connection to Current Source Output < CNT16-8L(FIT)GY >

4. Using the I/O Address Map

Starting I/O Address

When connected to a CPU-SBxx(FIT)GY, the Module can directly receive I/O commands from the controller module. Depending on how the Device ID is set, the I/O addresses indicated below will be used exclusively by the Module.

Because the address bus on which I/O address space is allocated is not fully decoded in 16 bits, four starting I/O addresses exist for each Device ID.

If the Device ID is set to 0h, one of the four addresses (0800h, 0840h, 0880h, or 08C0h) will be used as a starting I/O address.

Table 4.1. List of Starting I/O Addresses
< CNT16-8(FIT)GY, CNT16-8L(FIT)GY >

ID No.	Occupied I/O address			
0	0800h - 081Fh(recommended)	0840h - 085Fh	0880h - 089Fh	08C0h - 08DFh
1	1800h - 181Fh(recommended)	1840h - 185Fh	1880h - 189Fh	18C0h - 18DFh
2	2800h - 281Fh(recommended)	2840h - 285Fh	2880h - 289Fh	28C0h - 28DFh
3	3800h - 381Fh(recommended)	3840h - 385Fh	3880h - 389Fh	38C0h - 38DFh
4	4800h - 481Fh(recommended)	4840h - 485Fh	4880h - 489Fh	48C0h - 48DFh
5	5800h - 581Fh(recommended)	5840h - 585Fh	5880h - 589Fh	58C0h - 58DFh
6	6800h - 681Fh(recommended)	6840h - 685Fh	6880h - 689Fh	68C0h - 68DFh
7	7800h - 781Fh(recommended)	7840h - 785Fh	7880h - 789Fh	78C0h - 78DFh

For detailed specifications on the I/O space that is managed by the controller module, see the controller module manual.

List of I/O Address Maps

CNT16-8(FIT)GY

Starting I/O address Input	D7	D6	D5	D4	D3	D2	D1	D0
	Products Category				**Rivision Data**			
+0 (00h)	0	0	1	1	Rivision Data 3	Rivision Data 2	Rivision Data 1	Rivision Data 0
	Products ID Number							
+1 (01h)	0	0	0	0	0	0	0	1
	Interrupt Status							
+2 (02h)	Enable (0)	Status (0)				IRQ 9 (0)	IRQ 7 (0)	IRQ 5 (0)
+3 (03h) to +16 (10h)	Reserved							
+17 (11h)	**Data / Status**							
	Data 07	Data 06	Data 05	Data 04	Data 03	Data 02	Data 01	Data 00
+18 (12h) to +31 (1Fh)	Reserved							

Figure 4.1. Input Port < CNT16-8(FIT)GY >

Starting I/O address	D7	D6	D5	D4	D3	D2	D1	D0
Output +0 (00h) to +1 (01h)	N/A							
	Interrupt Status							
+2 (02h)	Enable (0)					IRQ 9 (0)	IRQ 7 (0)	IRQ 5 (0)
+3 (04h) to +15 (0Fh)	N/A							
	Command							
+16 (10h)	Command Data 07	Command Data 06	Command Data 05	Command Data 04	Command Data 03	Command Data 02	Command Data 01	Command Data 00
	Setting Data							
+17 (11h)	Setting Data 07	Setting Data 06	Setting Data 05	Setting Data 04	Setting Data 03	Setting Data 02	Setting Data 01	Setting Data 00
+18 (12h) to +31 (1Fh)	N/A							

Figure 4.2. Output Port < CNT16-8(FIT)GY >

CNT16-8L(FIT)GY

Starting I/O address Input	D7	D6	D5	D4	D3	D2	D1	D0
	Products Category				**Rivision Data**			
+0 (00h)	0	0	1	1	Rivision Data 3	Rivision Data 2	Rivision Data 1	Rivision Data 0
	Products ID Number							
+1 (01h)	0	0	0	0	0	0	1	0
	Interrupt Status							
+2 (02h)	Enable (0)	Status (0)				IRQ 9 (0)	IRQ 7 (0)	IRQ 5 (0)
+3 (03h) to +16 (10h)	N/A							
	Input Data							
+17 (11h)	Data 07	Data 06	Data 05	Data 04	Data 03	Data 02	Data 01	Data 00
+18 (12h) to +31 (1Fh)	Reserved							

Figure 4.3. Input Port < CNT16-8L(FIT)GY >

Starting I/O address	D7	D6	D5	D4	D3	D2	D1	D0
Output +0 (00h) +1 (01h)	N/A							
	Interrupt Status							
+2 (02h)	Enable (0)					IRQ 9 (0)	IRQ 7 (0)	IRQ 5 (0)
+3 (04h) +15 (0Fh)	N/A							
	Command							
+16 (10h)	Command Data 07	Command Data 06	Command Data 05	Command Data 04	Command Data 03	Command Data 02	Command Data 01	Command Data 00
	Setting Data							
+17 (11h)	Setting Data 07	Setting Data 06	Setting Data 05	Setting Data 04	Setting Data 03	Setting Data 02	Setting Data 01	Setting Data 00
+18 (12h) +31 (1Fh)	N/A							

Figure 4.4. Output Port < CNT16-8L(FIT)GY >

Specifications Common to F&eIT Products

The starting I/O addresses from +0h - +Fh are common to all modules in the F&eIT series.

Product Information

Starting I/O address Input	D7	D6	D5	D4	D3	D2	D1	D0
	Product Category				**Rivision Data**			
+0 (00h)	0	0	1	1	Revision Data3	Revision Data2	Revision Data1	Revision Data0
	Product ID Number							
+1 (01h)	0	0	0	0	0	0	0	1
	Interruput Status							
+2 (02h)	Enable(0)	Status(0)	0	0	0	IRQ 9 (0)	IRQ 7 (0)	IRQ 5 (0)

Figure 4.5. Product Information < CNT16-8(FIT)GY >

- Revision Data [D3 - D0]:
  This is product update information, subject to change without notice, that is managed by CONTEC.
- Product Category [D7 - D4]:
  This is a module function classification code. For the CNT16-8(FIT)GY and CNT16-8L(FIT)GY, the code is "1h".

Table 4.2. Product Category

Code	Function
0	Bus expansion
1	Digital input-output
2	Analog input-output
3	Counter
4	Serial communications
5	GPIB
6-F	Reserved

- Products ID Number [D7 - D0]:
  This is the product ID within the same product category.
  CNT16-8(FIT)GY : 1h
  CNT16-8L(FIT)GY : 2h

Following are examples of initialization coded in high-level languages:

Microsoft C
```
ProductID = inp( ADR+1 );
```

Microsoft QBASIC
```
ProductID = INP( ADR+1 )
```

* ADR is the starting I/O address for the CNT16-8(FIT)GY and CNT16-8L(FIT)GY.

Interrupt status

This is a common port on which the interrupt status requested by the Module can be verified. Information on interrupt sources varies from module to module.

Starting I/O address	D7	D6	D5	D4	D3	D2	D1	D0
Input	Interruput Status							
+2 (02h)	Enable (0)	Status (0)	0	0	0	IRQ 9 (0)	IRQ 7 (0)	IRQ 5 (0)

Figure 4.6. Interrupt Status

- Enable [D7]:

  This bit verifies the interrupt source enabled/disabled status.
  The value "1" indicates that a hardware interrupt on the controller module is enabled.

- Status [D6]:

  This bit indicates an interrupt request status in the module. When an interrupt event has occurred in the module and IRQ5, IRQ7, or IRQ9 is "1", this bit will also be "1".

- IRQ* [D2 - D0]:

  These bits allow you to verify the interrupt level that is currently set. The current interrupt level is indicated as "1".

Setting an interrupt level

Starting I/O address	D7	D6	D5	D4	D3	D2	D1	D0
Output	Interruput Status							
+2 (02h)	Enable (0)	Status (0)	0	0	0	IRQ 9 (0)	IRQ 7 (0)	IRQ 5 (0)

Figure 4.7. Setting an interrupt level

- Enable [D7]:

  This bit enables an interrupt source. Setting the bit to “1” enables hardware interrupts to the controller module.

- IRQ* [D2 - D0]:

  The interrupt level used by the module is set in these bits. Setting any of the IRQs to “1” makes the signal on that IRQ active upon reception of an interrupt request.

Bit Assignments for I/O Ports

The I/O ports for the counting function have command-section and data-section registers.

A register can be set as follows: First, a command is issued (OUT) to a port located at starting I/O address + 16 to make the register available for setting. For output, data is assigned (OUT) to the register at port +17; for input, the +17 port is read. In other words, for both input and output, a command is issued (OUT) to the output port +16 to make registers available for setting, and either data is issued (OUT) to the port +17 to set the register or the register is read at the +17 port.

When setting a register even when using the same command, the command should be output to the output port +16 each time.

Figures 4.8. and 4.9. show I/O port bit assignments.

Starting I/O address	D7	D6	D5	D4	D3	D2	D1	D0
+16 (10h)	Undetermined							
+17 (11h)	**Setting Data**							
	Setting Data 07	Setting Data 06	Setting Data 05	Setting Data 04	Setting Data 03	Setting Data 02	Setting Data 01	Setting Data 00

Figure 4.8. Input Port

Starting I/O address	D7	D6	D5	D4	D3	D2	D1	D0
+16 (10h)	**Command**							
	Command Data 07	Command Data 06	Command Data 05	Command Data 04	Command Data 03	Command Data 02	Command Data 01	Command Data 00
+17 (11h)	**Setting Data**							
	Setting Data 07	Setting Data 06	Setting Data 05	Setting Data 04	Setting Data 03	Setting Data 02	Setting Data 01	Setting Data 00

Figure 4.9. Output Port

Flow of Count Values

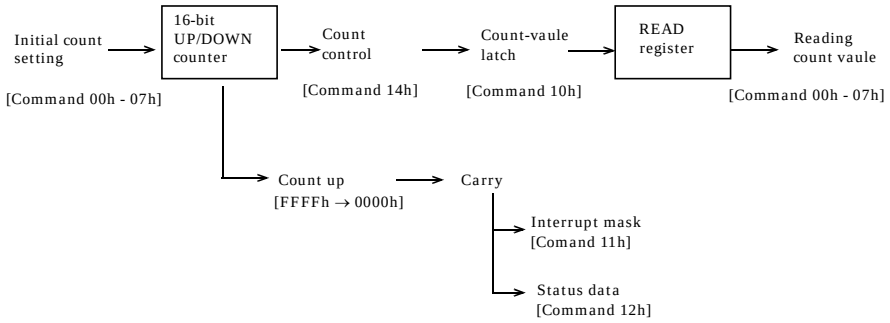


Figure 4.10. Flow of Count Values

Operation Commands

Table 4.3. Output Commands

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data Size
00h	CH0 initial count vaule								CH0 initial count vaule	16-bit
01h	CH1 initial count vaule								CH1 initial count vaule	16-bit
02h	CH2 initial count vaule								CH2 initial count vaule	16-bit
03h	CH3 initial count vaule								CH3 initial count vaule	16-bit
04h	CH4 initial count vaule								CH4 initial count vaule	16-bit
05h	CH5 initial count vaule								CH5 initial count vaule	16-bit
06h	CH6 initial count vaule								CH6 initial count vaule	16-bit
07h	CH7 initial count vaule								CH7 initial count vaule	16-bit
10h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Count data latching	8-bit
11h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Interrupt mask	8-bit
12h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Status reset	8-bit
13h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Counter initialization	8-bit
14h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Counter control	8-bit
15h	Not allowed			ST4	ST3	ST2	ST1	ST0	Digital filter	5-bit

Table 4.4. Input Commands

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data Size
00h	CH0 count vaule								CH0 count vaule	16-bit
01h	CH1 count vaule								CH1 count vaule	16-bit
02h	CH2 count vaule								CH2 count vaule	16-bit
03h	CH3 count vaule								CH3 count vaule	16-bit
04h	CH4 count vaule								CH4 count vaule	16-bit
05h	CH5 count vaule								CH5 count vaule	16-bit
06h	CH6 count vaule								CH6 count vaule	16-bit
07h	CH7 count vaule								CH7 count vaule	16-bit
11h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Interrupt mask	8-bit
12h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Status	8-bit
14h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Counter control	8-bit
15h	Not allowed			ST4	ST3	ST2	ST1	ST0	Digital filter	5-bit

Description of Input Commands

Reading a Count (command 00h - 07h)

By reading the contents of the READ register for a corresponding channel, this command reads a count value. By issuing the command to the output port +16 and reading the input port +17 two times, you can read low, middle, and high count values.

In the initial state, the contents of the READ register are undefined.

Following are programs that read a count value from CH0:

Microsoft C

```
outp( ADR+16, 0x0 );
LowerData = inp( ADR+17 );
 UpperData = inp( ADR+17 );
```

Microsoft QBASIC

```
OUT ADR+16, &H0
LowerData = INP( ADR+17 )
UpperData = INP( ADR+17 )
```

In this case, the count value latch for CH0 must be pre-set to [1].

Interrupt mask (command 11h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
11h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Interrupt masuk	8-bit

This command allows you to monitor the status of the current interrupt mask that was set using the interrupt mask command. CH7 - CH0 are associated with channels. When these bits are [1], the interrupt is masked, and no interrupt signals are output.

You can monitor the masking status by issuing the value 15h to the output port +16 and by reading the input port +17.

0: Not masked
1: Masked

⚠ CAUTION

Initial state: FFH

Status (command 12h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
12h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Status	8-bit

Indicates a carry-up occurred for the count on each channel.

By issuing 12H to the output port +16 and reading the input port +17, you can monitor the status. The corresponding bit is set to "1" when a carry-up occurs on a channel (CH7 - CH0).

0 : No carry
1 : Carry

⚠ CAUTION

- Initial state: 00h
- If an interrupt is generated with the interrupt option set, and if one of the bits in CH7 - CH0 is set to [1], before another interrupt can be generated, the affected sense bit must be reset. For a description of how to reset a sense bit, see "Sense Reset" on output ports.

Count control (Command 14h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
14h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Count control	8-bit

Count operating status of each channel is displayed.

You can monitor the operaing status by issuing the value 14h to the output port +16 and by reading the input port +17.

0 : Stopping
1 : Operating

⚠ CAUTION

Initial state: 00h

Digital Filter (command 15h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
15h	0	0	0	ST4	ST3	ST2	ST1	ST0	Digital Filter	5-bit

This can be used to check the settings for the digital filter.

Output 15H to output port +16 and read input port +17.

Description of Output Commands

Initial Count Value (command 00h - 07h)

The command is issued to the output port +16, and an initial count value is set on the output port +17. Because count data consists of 16 bits, count values are output two times in low, middle, and high order, 8 bits each time. When the third (high 8 bits) count value is output, data consisting of 16 bits is loaded simultaneously by the counting process.

Following are program examples that set a count value 100(64h) on CH0:

Microsoft C

```
outp( ADR+16, 0x0 );
outp( ADR+17, 0x64 );
outp( ADR+17, 0x0 );
```

Microsoft QBASIC

```
OUT ADR+16, &H0
OUT ADR+17, &H64
OUT ADR+17, &H0
```

Count Data Latching (command 10h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
10h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Count data latching	8-bit

This command latches a count value on a corresponding channel to the READ register.

The command issues the value "10h" to the output port +16, and sets a data latch on the output port +17.

These bits are associated with the respective channels. The command latches the count value by setting the applicable bit to "1". All channels can be latched simultaneously by setting all applicable bits to "1".

⚠ CAUTION

The initial condition is no-latch (00h).

Interrupt Mask (command 11h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
11h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Interrupt Mask	8-bit

This command issues the value "11h" to the output port +16, and sets a mask on the output port +17. The generation of interrupt signals is disabled when an applicable bit is set to "1".

These bits are associated with the respective channels. Setting the value "1" to any of these bits disables a carry-up interrupt on the associated channel. Setting the value "0" resets the disabled condition.

⚠ CAUTION

- In the initial condition, all channels are timer-masked (FFh).
- Even in the masked state, on all channels the carry-up conditions change.

Status Reset (command 12h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
12h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Status Reset	8-bit

The interrupt sense sets an applicable bit to "1" when carry-up condition is detected on a given channel.

When the applicable bit is "1", an interrupt signal is not generated when another carry-up condition arises. Issuing the value "+12h" to the output port +16 and the value "1" to the applicable bit on the output port +17 clears the sense bit, and enables the generation of another interrupt signal.

Counter initialization (command 13h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
13h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Counter initialization	8-bit

This command resets the counter data of each channel to 0000h.

The command issues the value "13h" to the output port +16, and resets on the output port +17 and sets the applicable bit to "1".

Counter control (command 14h)

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
14h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	Counter control	8-bit

This command controls the corresponding count operation of channel. The command issues the value "14h" to the output port +16, and sets a count control on the output port +17. The command controls the count operation by setting the applicable bit to "1".

These bits are associated with the respective channels. Setting the value "1" to any of these bits starts counting. Setting the value "0" to any of these bits stops counting.

⚠ CAUTION

- The initial state is all channels stopped(00h).
- The count data remains unchanged while the count is stopped.

Digital Filter (command 15h)

Using this function digitally filters the signals at all input pins by means of hardware. Use it to prevent input signal noise and chattering from being detected by mistake.

The level is checked at each clock sampling time. If the signal level remains the same for longer than the time set for the digital filter, the signal is treated as a valid input and the level of the PC signal is changed.

Accordingly, if level changes occur at a higher frequency than the filter time, the level changes will not be detected.

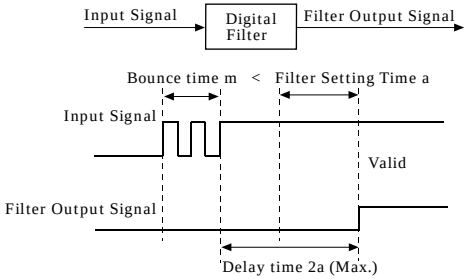


Figure 4.11. Digital Filter Operation Principle

Set Digital Filter Time

Command	D7	D6	D5	D4	D3	D2	D1	D0	Function	Data length
15h	Not used			ST4	ST3	ST2	ST1	ST0	Digital Filter	5-bit

Set Digital Filter Time

Output setting data "ST4 - ST0" to the "time setting" output port to set the digital filter time.

The table below lists the relationships between digital filter time and setting data.

Digital Filter Time [sec] = 2^n / (8 x 10^6)

n: Setting Data (0 - 20)

Table 4.5. Digital Filter Time and Setting Data

Setting Data (n)	Digital Filter Time	Setting Data (n)	Digital Filter Time	Setting Data (n)	Digital Filter Time
0 (00h) *1	Not used	7 (07h)	16µSec	14 (0Eh)	2.048mSec
1 (01h)	0.25µSec	8 (08h)	32µSec	15 (0Fh)	4.096mSec
2 (02h)	0.5µSec	9 (09h)	64µSec	16 (10h)	8.192mSec
3 (03h)	1µSec	10 (0Ah)	128µSec	17 (11h)	16.384mSec
4 (04h)	2µSec	11 (0Bh)	256µSec	18 (12h)	32.768mSec
5 (05h)	4µSec	12 (0Ch)	512µSec	19 (13h)	65.536mSec
6 (06h)	8µSec	13 (0Dh)	1024µSec	20 (14h)	131.072mSec

*1: Factory setting

⚠ CAUTION

- The default value for digital filter setting time is "Not used". The default setting is used when the power is turned on.
- The digital filter applies to all input channels; it cannot apply to specific input pins only.
- Do not set the setting data to any value other than the above. Doing so may result in a malfunction.

Examples

Pulse Count (no interrupts)

Flowchart

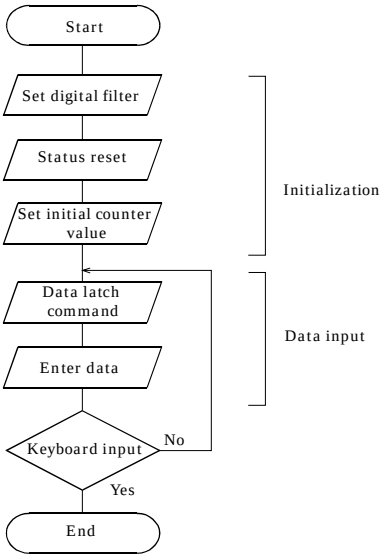


Figure 4.12. Pulse Count (No Interrupts)

Sample Program

```
/*
===========================================================================
   Sample program 1

          DEVICE ID:      0
          Initial Data:   0000H
          Channel:        0ch
          Digital Filter  1024 usec
          Interrupt:      N/A
===========================================================================
*/
#include <stdio.h>
#include <conio.h>

/* ----- Constant ------------------------------------------------------- */
#define   ADR    0x0800                          /* I/O address */
#define   CH     8                               /* channel */

/* ----- Declarations --------------------------------------------------- */
struct    REGS08 {                                /* 8-bit */
   unsigned char          lower;                 /* lower */
   unsigned char          upper;                 /* upper*/
};

struct    REGS16 {                                /* 16-bit */
   unsigned short         count;
};

union     ACCESS {
   struct REGS08 Byte[CH];                       /*  8-bit */
   struct REGS16 Whole[CH];                      /* 16-bit */
};

union     ACCESS CountData;


/* ----- Prototype ------------------------------------------------------ */
void      main( void );
void      Initialize( unsigned char );          /* initialize */
void      ReadData( unsigned char, unsigned short *, unsigned char * );
   /* read data */

/* ----- Initialize ----------------------------------------------------- */
void      Initialize( unsigned char ch )
{
   outp( ADR+0x10, 0x15 );                       /* digital filter set */
   outp( ADR+0x11, 0x0d );                       /* 1024us */
   outp( ADR+0x10, 0x12 );                       /* sense reset */
   outp( ADR+0x11, 0xff );
   outp( ADR+0x10, ch);                          /* initial data set */
   outp( ADR+0x11, CountData.Byte[ch].lower );   /* lower */
   outp( ADR+0x11, CountData.Byte[ch].upper );   /* upper */
}
```

```
/* ----- Read Data ------------------------------------------------------------
*/
void      ReadData( unsigned char ch, unsigned short *data, unsigned char *sts )
{
   outp( ADR+0x10, 0x10 );                        /* data latch */
   outp( ADR+0x11, 0xff );
   outp( ADR+0x10, ch );                          /* read data */
   CountData.Byte[ch].lower = inp( ADR+0x11 );  /* lower */
   CountData.Byte[ch].upper = inp( ADR+0x11 );  /* upper */
   *data = CountData.Whole[ch].count;
   outp( ADR+0x10, 0x12 );                        /* status*/
   *sts = (unsigned char)inp(ADR+0x11);
}

/* ------ main ------------------------------------------------------------- */
void      main( void )
{
   unsigned char ch, sts;
   unsigned short data;

   ch = 0;                                        /* channel 0 */
   CountData.Whole[ch].count = 0x0000;            /* count data */

   Initialize( ch );
   outp( ADR+0x10, 0x14 );                        /* count start */
   outp( ADR+0x11, (0x01 << ch));
   while( !kbhit() ) {
         ReadData( ch, &data, &sts );
         printf("%01dch   %08d   Status %02x \n", ch, data, sts );/* display */
   }
   outp( ADR+0x10, 0x14 );                        /* count stop */
   outp( ADR+0x11, 0x00 );
}

/* ---------------------------------------------------- End of file --- */
```

Pulse Count (with interrupts)

Flowchart

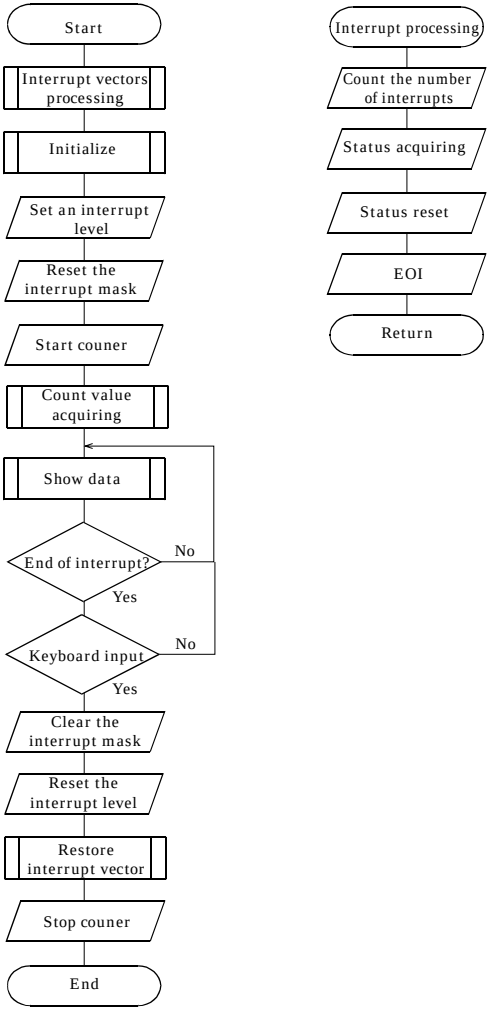


Figure 4.13. Pulse Count (with Interrupts)

Sample Program

```
/*
===========================================================================
   Sample program 2

          DEVICE ID:      0
          Initial Data:   0000H
          Channel:        0 to 3ch
          Digital Filter  1024 usec
          Interrupt:      IRQ5  10 times

===========================================================================
*/
#include <stdio.h>
#include <conio.h>
#include <dos.h>

/* ----- Constant ------------------------------------------------------- */
#define   ADR    0x0800                          /* I/O address */
#define   CH     4                               /* channel */
#define   IRQ5   0                               /* IRQ5 */
#define   IRQ7   1                               /* IRQ7 */
#define   IRQ9   2                               /* IRQ9 */


/* ----- Declarations --------------------------------------------------- */
struct    REGS08 {                                /* 8-bit */
   unsigned char          lower;                 /* lower */
   unsigned char          upper;                 /* upper*/
};

struct    REGS16 {                                /* 16-bit */
   unsigned short         count;
};

union     ACCESS {
   struct REGS08 Byte[CH];                       /*  8-bit */
   struct REGS16 Whole[CH];                      /* 16-bit */
};

union           ACCESS    CountData;
volatile  int   intcnt = 0;                                          /*
interrupt counter */
volatile  unsigned char intstatus = 0;          /* interrupt status */
volatile  int   IrqLevel = IRQ5;                /* interrupt level */
                int       OrgMasterImr, OrgSlaveImr;
                                                /* original IMR */
                unsigned char   IntVector[3] = { 0x0d, 0x0f, 0x71 };
                                                /* interruput vector */
                unsigned char   PicMask[3] = { 0xdf, 0x7f, 0xfd };
                                                /* mask bit */
                unsigned char   IsrClear[3] = { 0x65, 0x67, 0x61 };
                                                /* ISR clear */
                unsigned char   IntEnable[3] = { 0x81, 0x82, 0x84 };
                                                /* interrupt enable */


/* ----- Prototype -----------------------------------------------------
*/
void      main( void );
void      Initialize( unsigned char );          /* initialize */
void      ChgVect( void );                      /* change vector */
```

```
void      ReadData( void );                     /* read data */
void      Display( void );                      /* display */
void      ResVect( void );                      /* restore vector */
void      _interrupt _far inthandler( void );   /* interrupt handler */
void      ( _interrupt _far *OrgVect)();        /* original interrupt vector
*/


/* ----- Initialize ------------------------------------------------------
*/
void      Initialize( unsigned char ch )
{
   outp( ADR+0x10, 0x15 );                      /* digital filter set */
   outp( ADR+0x11, 0x0d );                      /* 1024us */
   outp( ADR+0x10, 0x12 );                      /* sense reset */
   outp( ADR+0x11, 0xff );
   outp( ADR+0x10, ch );                        /* initial data set */
   outp( ADR+0x11, CountData.Byte[ch].lower );  /* lower */
   outp( ADR+0x11, CountData.Byte[ch].upper );  /* upper */
}


/* ----- change vector ---------------------------------------------------
- */
void      ChgVect( void )
{
   OrgVect = _dos_getvect( IntVector[IrqLevel] );
   _disable();
   _dos_setvect( IntVector[IrqLevel], inthandler );
   if ( IrqLevel > IRQ7 ) {                     /* IMR and mask clear */
         outp( 0x21, ( OrgMasterImr = inp( 0x21 ) ) & 0xfb );
         outp( 0xa1, ( OrgSlaveImr = inp( 0xa1 ) ) & PicMask[IrqLevel] );
         outp( 0x20, 0x62 );                    /* ISR clear (master) */
         outp( 0xa0, IsrClear[IrqLevel] );      /* ISR clear (slave) */
   } else {
   /* IMR and mask clear */
         outp( 0x21, ( OrgMasterImr = inp( 0x21 ) ) & PicMask[IrqLevel] );
         outp( 0x20, IsrClear[IrqLevel] );      /* ISR clear */
   }
   _enable();
   /* enable */
}

/* ----- Read Data -------------------------------------------------------
*/
void      ReadData( void )

{
   unsigned char ch;
   outp( ADR+0x10, 0x10 );                                  /* data latch */
   outp( ADR+0x11, 0xff );

   for (ch=0; ch<CH; ch++) {
         outp( ADR+0x10, ch );                              /* read data */
         CountData.Byte[ch].lower = inp( ADR+0x11 );        /* lower */
         CountData.Byte[ch].upper = inp( ADR+0x11 );        /* upper */
   }
}

/* ----- display ---------------------------------------------------------
- */
void      Display( void )
{
```

```
   int          i;

   for (i = 0; i < CH; i++) {
         printf("%dch %04XH, ", i, CountData.Whole[i].count );
   }
   printf("interrupt count = %03d, status = %02X\n", intcnt, (int)intstatus);
}


/* ------ restore vector ------------------------------------------------------
- */
void      ResVect( void )
{
   _disable();                                    /* disable */
   if ( IrqLevel > IRQ7 ) {     /* restore IMR */
         outp( 0x21, OrgMasterImr );
         outp( 0xa1, OrgSlaveImr );
   } else
         outp( 0x21, OrgMasterImr );
   _dos_setvect( IntVector[IrqLevel], OrgVect );/* restore orgvect */
   _enable();                                     /* enable */
}


/* ----- interrupt handler ---------------------------------------------------
- */
void      _interrupt _far inthandler( void )
{
   _enable();                                     /* enable */

   intcnt++;                                      /* count interrupt */
   outp( ADR+0x10, 0x12 );                        /* status sence */
   intstatus = (unsigned char)inp(ADR+0x11);
   outp( ADR+0x10, 0x12 );                        /* sense reset */
   outp( ADR+0x11, 0xff );
   _disable();                                    /* disable */

   if ( IrqLevel > IRQ7 ) {                       /* EOI */
         outp( 0xa0, 0x20 );
         outp( 0xa0, 0x0b );
         if ( !inp( 0xa0 ) ) {
               outp( 0x20, 0x20 );
         }
   } else {
         outp( 0x20, 0x20 );
   }
}


/* ------ main ----------------------------------------------------------------
*/
void      main( void )
{
   unsigned char i;

   ChgVect();                                     /* change vector */

   for (i = 0; i < CH; i++) {
         CountData.Whole[i].count = 0x0000;      /* count data */
         Initialize(i);                          /* initialize */
   }

   outp( ADR+0x2, IntEnable[IrqLevel] );          /* interrupt level */
```

```
    outp( ADR+0x10, 0x11 );                     /* interrupt mask open */
    outp( ADR+0x11, 0x00 );

    outp( ADR+0x10, 0x14 );                     /* count start */
    outp( ADR+0x11, 0xff );

    while( intcnt < 10 && kbhit() == 0) {
          ReadData();                           /* read data */
          Display();                            /* display  */
    }

    Display();                                  /* display  */
    outp( ADR+0x10, 0x11 );                     /* interrupt mask close */
    outp( ADR+0x11, 0xff );
    outp( ADR+0x2, 0x0 );                       /* interrupt level */
    ResVect();                                  /* restore vector */
    outp( ADR+0x10, 0x14 );                     /* count stop */
    outp( ADR+0x11, 0x00 );

}

/* ------------------------------------------------------------ End of file --- */
```

5. Using the Memory Address Map

When connected to a CPU-CAxx(FIT)GY, the CNT16-8(FIT)GY, CNT16-8L(FIT)GY can be accessed by a host computer through a network. In addition, the Module can be allocated to the memory controlled by the Controller Module according to a given Device ID. Applications running on the host computer control the I/O modules by reading/writing the memory that is controlled by the Controller Module.

For detailed specifications on the memory controlled by the Controller Module, see the Controller Module manual.

Following is an explanation of the memory areas necessary for the use of this I/O module: the "module area", the "module information area", and the "basic data area".

Module Information area

This area controls the settings and how the module is started.

The module becomes available when the necessary settings are written into this area and the module activation option is set in the [module startup register].

Module information area

The current module settings are stored in this area.

When the Module is started, the contents of the Module Information Area are copied to the Module Information Area. By reading this area, you can verify the current module settings.

Basic I/O data area

Basic I/O data is accessed in this area.

Module Information Area

A module information area, which is a 128-byte (80h) area beginning with address 301000h and corresponding to a given Device ID, is where the settings for the given device are read and written.

The starting address can be determined according to the following expression:

Starting address = 301000h + 80h x (Device ID)

Table 5.1. Module Information Area < 1 / 3 >

Address(h)	Area	Item	Size	Access type	Initial value(h)	Initial settings
Starting address+00	Module-specific information	Module type (category)	1	R	03	CNT16-8(FIT)GY
Starting address+01		Module type (serial No.)	1	R	01 *1	
Starting address+02		System-reserved (revision No.)	1	R	None	
Starting address+03		Supported functions	1	R	01	Basic input
Starting address+04		Number of basic input channels	1	R	08	8channels
Starting address+05		Basic input data size	1	R	02	2bytes
Starting address+06		Number of basic output channels	1	R	00	
Starting address+07		Basic output data size	1	R	00	
Starting address+08		Input channel settings address	1	R	20	20h
Starting address+09		Input channel settings data size	1	R	08	8bytes
Starting address+0A		Output channel settings address	1	R	00	
Starting address+0B		Output channel settings data size	1	R	00	
Starting address+0C - Starting address+0F		Reserved	4	R	None	
Starting address+10	Common to modules	Module startup register	1	R/W	00	
Starting address+11		Error status	1	R	00	
Starting address+12		Counter resolution	1	R	10	16bits
Starting address+13		Digital filter setting	1	R/W	00	Not used
Starting address+14 - Starting address+1F		Reserved	12	R	None	

*1 "02" on the CNT16-8L(FIT)GY.

Table 5.1. Module Information Area < 2 / 3 >

Address(h)	Area	Item		Size	Access type	Initial value(h)	Initial settings
Starting address+20	Channel settings	CH0	Startup register	1	R/W	00	
Starting address+21			Status reset	1	R/W	00	
Starting address+22			Counter initialization	1	R/W	00	
Starting address+23 - Starting address+24			Preset counter value	2	R/W	0000	
Starting address+25 - Starting address+27			Reserved	3	R	None	
Starting address+28		CH1	Startup register	1	R/W	00	
Starting address+29			Status reset	1	R/W	00	
Starting address+2A			Counter initialization	1	R/W	00	
Starting address+2B - Starting address+2C			Preset counter value	2	R/W	0000	
Starting address+2D - Starting address+2F			Reserved	3	R	None	
Starting address+30		CH2	Startup register	1	R/W	00	
Starting address+31			Status reset	1	R/W	00	
Starting address+32			Counter initialization	1	R/W	00	
Starting address+33 - Starting address+34			Preset counter value	2	R/W	0000	
Starting address+35 - Starting address+37			Reserved	3	R	None	
Starting address+38		CH3	Startup register	1	R/W	00	
Starting address+39			Status reset	1	R/W	00	
Starting address+3A			Counter initialization	1	R/W	00	
Starting address+3B - Starting address+3C			Preset counter value	2	R/W	0000	
Starting address+3D - Starting address+3F			Reserved	3	R	None	
Starting address+40		CH4	Startup register	1	R/W	00	
Starting address+41			Status reset	1	R/W	00	
Starting address+42			Counter initialization	1	R/W	00	
Starting address+43 - Starting address+44			Preset counter value	2	R/W	0000	
Starting address+45 - Starting address+47			Reserved	3	R	None	
Starting address+48		CH5	Startup register	1	R/W	00	
Starting address+49			Status reset	1	R/W	00	
Starting address+4A			Counter initialization	1	R/W	00	
Starting address+4B - Starting address+4C			Preset counter value	2	R/W	0000	
Starting address+4D - Starting address+4F			Reserved	3	R	None	
Starting address+50		CH6	Startup register	1	R/W	00	
Starting address+51			Status reset	1	R/W	00	
Starting address+52			Counter initialization	1	R/W	00	
Starting address+53 - Starting address+54			Preset counter value	2	R/W	0000	
Starting address+55 - Starting address+57			Reserved	3	R	None	

Table 5.1. Module Information Area < 3 / 3 >

Address(h)	Area	Item		Size	Access type	Initial value(h)	Initial settings
Starting address+58		CH7	Startup register	1	R/W	00	
Starting address+59			Status reset	1	R/W	00	
Starting address+5A			Counter initialization	1	R/W	00	
Starting address+5B - Starting address+5C			Preset counter value	2	R/W	0000	
Starting address+5D - Starting address+5F			Reserved	3	R	None	
Starting address+60		Startup register mask		1	R/W	00	
Starting address+61		Status reset mask		1	R/W	00	
Starting address+62		Counter initialization mask		1	R/W	00	
Starting address+63		Preset counter value mask		1	R/W	00	
Starting address+64 - Starting address+7F		Reserved		28	R	None	

Module-specific information

- Module type (category)

  The CNT16-8x(FIT)GY belongs to the counter module (03h) category.

- Module type (serial No.)

  The CNT16-8(FIT)GY is a counter module with a serial No. 1 (01h).

  The CNT16-8L(FIT)GY is a counter module with a serial No. 2 (02h).

- Supported functions

  The CNT16-8x(FIT)GY supports the basic input function (01h).

  Basic input data takes count values.

  No basic output data.

- Number of basic input channels

  The number of basic input channels for the CNT16-8x(FIT)GY is 8 (08h).

  Eight counter channels are provided.

- Basic input data size

  The basic input data size for the CNT16-8x(FIT)GY is 2 (02h) byte.

  The count value consists of two bytes.

- Number of basic output channels

  The number of basic output channels for the CNT16-8x(FIT)GY is 0 (00h).

  No output channel.

- Basic output data size

  The basic output data size for the CNT16-8x(FIT)GY is 0 (00h) byte.

- Input channel settings address

  The addresses of the settings for each input channel are represented as offset addresses within the module setting area.

  On the CNT16-8x(FIT)GY, the locations of the input channel settings start from "20h".

- Input channel settings data size

  Indicates the size of the data area for each input channel setting.
  The data size for the CNT16-8x(FIT)GY input channel settings is 8 bytes (08h).

- Output channel settings address

  The CNT16-8x(FIT)GY does not have channel-specific settings. This field is provided for compatibility with other device modules.

- Output channel settings data size

  The CNT16-8x(FIT)GY does not have channel-specific settings. This field is provided for compatibility with other device modules.

Items common to modules

- Module startup register

  Setting the module startup option (01h) causes the device module to be started.

  Setting the module startup option when the module is being started causes the module to be restarted.

  The CNT16-8x(FIT)GY does not contain a module shutdown function.

  00h : No operation
  01h : Module startup

- Error status

  The error status bits, which are not reflected in the module information area, always remain [00h].

  The error status on a module is stored in the module information area.

- Counter resolution

  The counter resolution of the CNT16-8x(FIT)GY is 16 (10h) bit.

- Digital filter setting

  Sets the digital filter.

Table 5.2. Digital Filter Time and Setting Data

Setting Data (n)	Digital Filter Time	Setting Data (n)	Digital Filter Time	Setting Data (n)	Digital Filter Time
0 (00h) *1	Not used	7 (07h)	16µSec	14 (0Eh)	2.048mSec
1 (01h)	0.25µSec	8 (08h)	32µSec	15 (0Fh)	4.096mSec
2 (02h)	0.5µSec	9 (09h)	64µSec	16 (10h)	8.192mSec
3 (03h)	1µSec	10 (0Ah)	128µSec	17 (11h)	16.384mSec
4 (04h)	2µSec	11 (0Bh)	256µSec	18 (12h)	32.768mSec
5 (05h)	4µSec	12 (0Ch)	512µSec	19 (13h)	65.536mSec
6 (06h)	8µSec	13 (0Dh)	1024µSec	20 (14h)	131.072mSec

*1: Factory setting

⚠ CAUTION

- The default value for digital filter setting time is "Not used".
- Writing a setting data other than a "0" will apply this filter function to all input channels. This function cannot be applied to particular pins only but is applied to all input channels.
- Do not set the setting data to any value other than the above. Doing so may result in a malfunction.

Channel settings

- Startup register

  This command controls the corresponding count operation of the channel. The command controls the count operation by setting the data of corresponding channel to 01h.

  Setting "01h" to the setting for each channel starts counting on that channel. Setting "00h" stops counting.

  00h : Stop
  01h : Start

  Setting the module start setting (01h) to the module startup register sets the startup registers for all channels to "01h".

⚠ CAUTION

The count data does not change while the count is stopped.

- Status reset

  When a carry-up occurs on a channel count, the data for the corresponding channel in "module data area", "channel data", and "status" goes to "01h".

  The next carry-up does not occur if the "status" for a channel is "01h". Using status reset to output "01" to the corresponding channel clears the sense and permits the next count carry-up to occur.

  00h : No operation
  01h : Sense reset

  Changes to "00h" when sense reset processing completes.

- Counter initialization

  Resets the counter data of each channel to 0000h.

  00h : No operation
  01h : Counter initialization

  Changes to "00h" when counter initialization completes.

- Preset counter value

  By setting values, you can set a preset value to the count for each channel.
  The preset counter value is stored as a little endian value.

Table 5.3. Preset counter value

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

Changes to "0000h" when preset processing completes.

- Startup register mask

  Use this function when you want to start or stop channels simultaneously. Write the settings and mask for CH0 - CH7 in a single write operation.

Table 5.4. Startup register mask

	D7	D6	D5	D4	D3	D2	D1	D0
+60h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Update the "startup register" values for the channels corresponding to bits set to "1".

0 : Ignore

1 : Update "startup register" value.

- Status reset mask

  Use this function when you want to perform a status reset for multiple channels simultaneously. Write the settings and mask for CH0 - CH7 in a single write operation.

Table 5.5. Status reset mask

	D7	D6	D5	D4	D3	D2	D1	D0
+61h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Update the "Status reset" values for the channels corresponding to bits set to "1".

0 : Ignore

1 : Update "Status reset" value.

- Counter initialization mask

  Use this function when you want to initialize channels simultaneously. Write the settings and mask for CH0 - CH7 in a single write operation.

Table 5.6. Counter initialization mask

	D7	D6	D5	D4	D3	D2	D1	D0
+62h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Update the "Counter initialization" values for the channels corresponding to bits set to "1".

0 : Ignore

1 : Update "Counter initialization" value.

- Preset counter mask

  Use this function when you want to preset channels simultaneously. Write the settings and mask for CH0 - CH7 in a single write operation.

Table 5.7. Preset counter mask

	D7	D6	D5	D4	D3	D2	D1	D0
+63h	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Update the "Preset counter" values for the channels corresponding to bits set to "1".

0 : Ignore

1 : Update "Preset counter" value.

Module Information Area

The module information area, which is a 128-byte (80h) area beginning with address 300000h and corresponding to a given Device ID, is the area into which settings are read.

The starting address can be determined according to the following expression:

Starting address = 300000h + 80h x (Device ID)

Table 5.8. Module Information Area < 1 / 2 >

Address(h)	Area	Item	Size	Access type	Initial value (h)
Starting address+00	Module-specific information	Module type (category)	1	R	03
Starting address+01		Module type (serial No.)	1	R	01 *1
Starting address+02		System-reserved (revision No.)	1	R	None
Starting address+03		Supported functions	1	R	01
Starting address+04		Number of basic input channels	1	R	08
Starting address+05		Basic input data size	1	R	02
Starting address+06		Number of basic output channels	1	R	00
Starting address+07		Basic output data size	1	R	00
Starting address+08		Input channel settings address	1	R	20
Starting address+09		Input channel settings data size	1	R	08
Starting address+0A		Output channel settings address	1	R	00
Starting address+0B		Output channel settings data size	1	R	00
Starting address+0C - Starting address+0F		Reserved	4	R	None
Starting address+10	Common to modules	Module startup register	1	R	00
Starting address+11		Error status	1	R	00
Starting address+12		Counter resolution	1	R	10
Starting address+13		Digital filter	1	R	00
Starting address+14 - Starting address+1F		Reserved	12	R	None

*1 "02" on the CNT16-8L(FIT)GY.

Table 5.8. Module Information Area < 2 / 2 >

Address(h)	Area	Item		Size	Access type	Initial value (h)
Starting address+20	Channel settings	CH0	Startup register	1	R	00
Starting address+21			Status	1	R	00
Starting address+22 - Starting address+27			Reserved	6	R	None
Starting address+28		CH1	Startup register	1	R	00
Starting address+29			Status	1	R	00
Starting address+2A - Starting address+2F			Reserved	6	R	None
Starting address+30		CH2	Startup register	1	R	00
Starting address+31			Status	1	R	00
Starting address+32 - Starting address+37			Reserved	6	R	None
Starting address+38		CH3	Startup register	1	R	00
Starting address+39			Status	1	R	00
Starting address+3A - Starting address+3F			Reserved	6	R	None
Starting address+40		CH4	Startup register	1	R	00
Starting address+41			Status	1	R	00
Starting address+42 - Starting address+47			Reserved	6	R	None
Starting address+48		CH5	Startup register	1	R	00
Starting address+49			Status	1	R	00
Starting address+4A - Starting address+4F			Reserved	6	R	None
Starting address+50		CH6	Startup register	1	R	00
Starting address+51			Status	1	R	00
Starting address+52 - Starting address+57			Reserved	6	R	None
Starting address+58		CH7	Startup register	1	R	00
Starting address+59			Status	1	R	00
Starting address+5A - Starting address+5F			Reserved	6	R	None
Starting address+60 - Starting address+7F		Reserved		32	R	None

When the module is started, the contents of the module information area are stored in the module information area.

Items common to modules

- Module startup register

  This register stores the module operating status.

  00h : No operation
  01h : Module startup

- Error status

  This register stores the error status of the module.
  The error status register is reset when the module is restarted.

  00h : Normal status

- Counter resolution

  This can be used to check the counter resolution for the CNT16-8x(FIT)GY.

- Digital filter setting

  This can be used to check the digital filter setting.

Channel settings

- Startup register

  This register displays the count operating status of each channel.

  00h : Stopping

  01h : Operating

- Status

  Indicates a carry-up occurred for the count on each channel. The data corresponding to each channel is set to "01h" when a carry-up occurs on a channel.

  00h : No carry

  01h : Carry

  Once a carry-up has occurred for a channel, the next carry-up cannot be detected until the sense is reset for that channel. See “Status reset” in the “ module information area” for details of sense reset.

Basic I/O Data Area

The basic Input data area, which is a 128-byte (80h) area beginning with address 304000h and corresponding to a given Device ID

The starting address can be determined according to the following expression:

Starting address = 304000h + 80h x (Device ID)

Table 5.9. Basic Input Data area

Address(h)	Area	Item	Size	Access type
Starting address+00	CH0	Counter value	2	R
Starting address+02	CH1	Counter value	2	R
Starting address+04	CH2	Counter value	2	R
Starting address+06	CH3	Counter value	2	R
Starting address+08	CH4	Counter value	2	R
Starting address+0A	CH5	Counter value	2	R
Starting address+0C	CH6	Counter value	2	R
Starting address+0E	CH7	Counter value	2	R
Starting address+10 - Starting address+7F	Reserved		112	R

Digital input value

The preset counter value is stored as a little endian value.

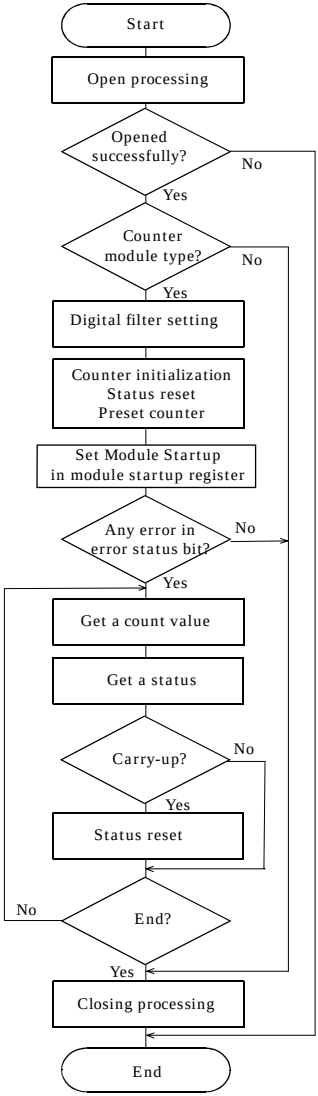
Table 5.10. Digital Input Values

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

Examples

Flowchart

The following flowchart illustrates an example where the CNT16-8x(FIT)GY is installed at device ID: 0.



Sample Program

```
/*=========================================================================
   F&eIT I/F Sample Program

         DEVICE ID:      0
         Channel:        0ch
=========================================================================== */
#include  <windows.h>
#include  <stdio.h>
#include  <stdlib.h>
#include  <conio.h>
#include  "Fit.h"

/* Address(common) */
#define   FIT_IO                              (0x00300000)
#define   FIT_IO_DEVICE_INFOR                 (0x0000)
#define   FIT_IO_DEVICE_CONFIG                (0x1000)
#define   FIT_IO_INPUT                        (0x4000)
#define   FIT_IO_OUTPUT                       (0x5000)

#define   FIT_IO_DEVICE_SIZE                  (0x0080)

#define   FIT_PRODUCT_CATEGORY                (0x00)

#define   FIT_MODULE_START                    (0x10)
#define   FIT_ERROR_STATUS                    (0x11)

/* Information(Common) */
#define   FIT_PRODUCT_DIGITAL                 (0x01)
#define   FIT_PRODUCT_ANALOG                  (0x02)
#define   FIT_PRODUCT_COUNTER                 (0x03)

#define FIT_MODULE_START_OFF                  (0x00)
#define   FIT_MODULE_START_ON                 (0x01)

/* Address(CNT) */
#define   FIT_CNT_BIT                         (0x12)
#define   FIT_CNT_D_FILTER                    (0x13)

#define   FIT_CNT_CH_START                    (0x00)
#define   FIT_CNT_CH_ST_RESET                 (0x01)
#define   FIT_CNT_CH_INITIAL                  (0x02)
#define   FIT_CNT_CH_PRESET                   (0x03)

/* Sample */
#define   FIT_SAMPLE_IP_ADDRESS               "172.17.8.151"
#define   FIT_SAMPLE_PORT                     (0x5007)
#define   FIT_SAMPLE_DEVICE_ID                (0)
#define   FIT_SAMPLE_CH                       (0)

int main(int argc, char* argv[])
{
   DWORD  dwIpAddress;
   DWORD  dwVaBase;
   DWORD  dwVaBase_Data;
   DWORD  dwVaOffset;
   DWORD  dwVaChOffset;
   DWORD  dwVaChOffset_Data;
   DWORD  dwChCarryCnt;
   WORD   hHandle;
   WORD   wStatus;
   BYTE   byCategory;
```

```
BYTE   byModuleStart;
BYTE   byData[0x80];
BYTE   byErrorStatus;
BYTE   byChStatus;

/* Open */
dwIpAddress = FIT_IpChenge((BYTE *)FIT_SAMPLE_IP_ADDRESS);
hHandle = FIT_Open((BYTE *)&dwIpAddress, FIT_SAMPLE_PORT, NULL);
if (hHandle == 0) {
      printf("Error! FIT_Open = %04X(H)\n", hHandle);
      return 1;
}

/* Offset Address */
dwVaOffset = FIT_IO_DEVICE_SIZE * FIT_SAMPLE_DEVICE_ID;

/* Read 'Category' */
dwVaBase = FIT_IO + FIT_IO_DEVICE_CONFIG;
wStatus = FIT_Read(hHandle, dwVaBase + dwVaOffset + FIT_PRODUCT_CATEGORY,
            0x01, &byCategory);
if (wStatus != 0) {
      printf("Error! FIT_Read = %04X(H)\n", wStatus);
      FIT_Close(hHandle);
      return 1;
}
if (byCategory != FIT_PRODUCT_COUNTER) {
      printf("Error! Category = %02X(H)\n", byCategory);
      FIT_Close(hHandle);
      return 1;
}

/* Digital Filter Set */
byData[0x00] = 0x0D;   /* 1.024msec */

/* Write 'Digital Filter Settings' */
wStatus = FIT_Write(hHandle, dwVaBase + dwVaOffset + FIT_CNT_D_FILTER,
            0x01, &byData[0]);
if (wStatus != 0) {
      printf("Error! FIT_Write = %04X(H)\n", wStatus);
}

/* Channel Offset */
dwVaChOffset  = 0x20 + 0x08 * FIT_SAMPLE_CH;

/* Read 'Channel Configuration' */
wStatus = FIT_Read(hHandle, dwVaBase + dwVaOffset + dwVaChOffset,
            0x08, &byData[0]);
if (wStatus != 0) {
      printf("Error! FIT_Read = %04X(H)\n", wStatus);
}

/* Configuration Data Set */
byData[FIT_CNT_CH_START]       = 0x00;       /* Ch Start */
byData[FIT_CNT_CH_ST_RESET]   = 0x01;        /* Ch Status Reset */
byData[FIT_CNT_CH_INITIAL]    = 0x01;        /* Ch Initial */
byData[FIT_CNT_CH_PRESET]     = 0x00;        /* Ch Preset Value (0)*/
byData[FIT_CNT_CH_PRESET + 1] = 0x00;

/* Write 'Channel Configuration' */
wStatus = FIT_Write(hHandle, dwVaBase + dwVaOffset + dwVaChOffset,
            0x08, &byData[0]);
if (wStatus != 0) {
      printf("Error! FIT_Write = %04X(H)\n", wStatus);
```

```
}

/* Write 'Module Start' */
dwVaBase = FIT_IO + FIT_IO_DEVICE_CONFIG;
byModuleStart = FIT_MODULE_START_ON;
wStatus = FIT_Write(hHandle, dwVaBase + dwVaOffset + FIT_MODULE_START,
            1, &byModuleStart);
if (wStatus != 0) {
      printf("Error! FIT_Write = %04X(H)\n", wStatus);
}

/* Read 'Error Status' */
dwVaBase = FIT_IO + FIT_IO_DEVICE_INFOR;
wStatus = FIT_Read(hHandle, dwVaBase + dwVaOffset + FIT_ERROR_STATUS,
            1, &byErrorStatus);
if (wStatus != 0) {
      printf("Error! FIT_Read = %04X(H)\n", wStatus);
      FIT_Close(hHandle);
      return 1;
}
if (byErrorStatus != 0x00) {
      printf("Error! Error Status = %02X(H)\n", byErrorStatus);
      FIT_Close(hHandle);
      return 1;
}

/* Read 'Counter Data' */
dwVaBase = FIT_IO + FIT_IO_DEVICE_INFOR;
dwVaOffset = FIT_IO_DEVICE_SIZE * FIT_SAMPLE_DEVICE_ID;
dwVaChOffset  = 0x08 * FIT_SAMPLE_CH;

dwVaBase_Data = FIT_IO + FIT_IO_INPUT;
dwVaChOffset_Data = 0x02 * FIT_SAMPLE_CH;

dwChCarryCnt = 0;

printf("- Hit Any Key:Stop -\n");
while (!kbhit()) {
      /* Read 'Counter Data' */
      wStatus = FIT_Read(hHandle, dwVaBase_Data + dwVaOffset +
                  dwVaChOffset_Data, 2, &byData[0]);
      if (wStatus != 0) {
            printf("Error! FIT_Read = %04X(H)\n", wStatus);
      }

      /* Read 'Status Data' (Carry Up)*/
      dwVaBase = FIT_IO + FIT_IO_DEVICE_INFOR;
      wStatus = FIT_Read(hHandle, dwVaBase + dwVaOffset + dwVaChOffset +
                  FIT_CNT_CH_ST_RESET + 0x20, 1, &byChStatus);
      if (wStatus != 0) {
            printf("Error! FIT_Read = %04X(H)\n", wStatus);
      }

      if( byChStatus == 0x01 ){
            /* Write 'Status Reset' (Carry Up Clear)*/
            dwChCarryCnt ++;
            dwVaBase = FIT_IO + FIT_IO_DEVICE_CONFIG;
            wStatus = FIT_Write(hHandle, dwVaBase + dwVaOffset +
                        dwVaChOffset + FIT_CNT_CH_ST_RESET + 0x20,
                        1, &byChStatus);
            if (wStatus != 0) {
                  printf("Error! FIT_Write = %04X(H)\n", wStatus);
            }
```

```
            }
            printf("Counter CH%d Data:%02X%02X CarryUpCount: %d\r",
                  FIT_SAMPLE_CH, byData[1], byData[0], dwChCarryCnt);

    }

    /* Close */
    FIT_Close(hHandle);

    return 0;
}
```

6. System Reference

Block Diagram

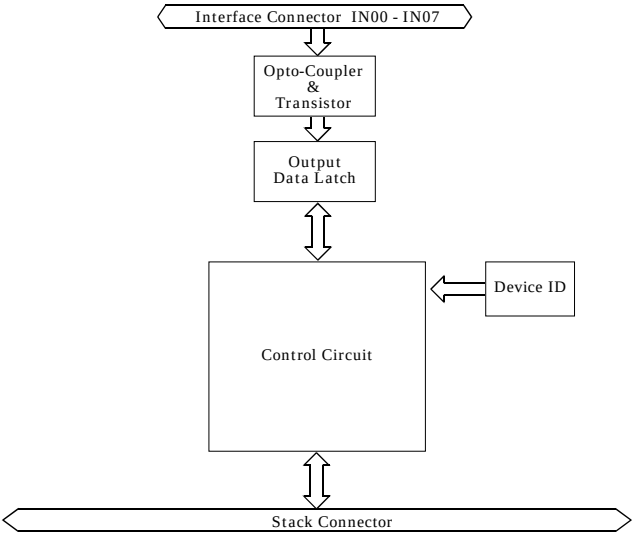


Figure 6.1. Circuit Block Diagram

Specifications

CNT16-8(FIT)GY

Table 6.1. Specifications

Item		Specification
Counter input section		
	Channel count	8 (8 points/common)
	Counting system	Up/down counting
	Max. count	FFFFH (binary data)
	Response frequency	5kHz (Max.) Duty 50% (Max.)
	Input format	Opto-isolated input (Compatible with both current sinking output and courrent source output)
	Input resistance	3kΩ
	Input ON /OFF current	3.4 mA(Min.) / 0.16 mA (Max.)
	External circuit power supply	12 - 24 VDC (±15%) (4 mA/12 V - 8 mA/24 V per channel)
	Digital filter	0.25μsec - 131.072μsec
	Interrupt	The interrupts for all eight channels are combined as a single interrupt output signal. An interrupt is generated when a count carry-up occurs.
Common section		
	Interrupt level	Using CPU-SBxx(FIT)GY: IRQ 5, 7, or 9
	Internal power consumption	5 VDC(±5%) 150 mA (Max.) *1
	Allowable distance of signal extension	Approx. 50m (depending on wiring environment)
	External dimensions (mm)	25.2 (W) x 64.7 (D) x 94.0 (H) (exclusive of protrusions)
	Weight of the module itself	100g
	Module connection method	Stack connection by the connector that is provided with the side of module
	Module installation method	One-touch connection to 35mm DIN rails (standard connection mechanism provided in the system)
	Applicable wire	AWG 28 - 16
	Applicable plug	FRONT-MC1,5/12-ST-3,81 (made by Phoenix Contact Corp.)

*1 The stack connector accepts currents of up to 3.0A (Max.).

Table 6.2. Installation Environment Requirements

Parameter	Requirement description
Operating temperature	0 - 50°C
Storage temperature	-10 - 60°C
Operating humidity	10 - 90%RH (No condensation)
Floating dust particles	Not to be excessive
Corrosive gases	None

⚠ CAUTION

When connecting the Module to a controller module, the internal power consumption should be taken into account. If the total current exceeds the capacity of the power supply unit, the integrity of the operation cannot be guaranteed. For further details, please see the Controller Module manual.

CNT16-8L(FIT)GY

Table 6.3. Specifications

Item		Specification
Counter input section		
	Channel count	8 (8 points/common)
	Counting system	Up/down counting
	Max. count	FFFFH (binary data)
	Response frequency	10kHz (Max.) Duty 50% (Max.)
	Input format	Opto-isolated input (Compatible with both current sinking output and courrent source output)
	Input resistance	1.2kΩ
	Input ON /OFF current	3.8 mA(Min.) / 0.16 mA (Max.)
	External circuit power	5 VDC (±10%) (4 mA per channel)
	Digital filter	0.25μsec - 131.072μsec
	Interrupt	The interrupts for all eight channels are combined as a single interrupt output signal. An interrupt is generated when a count carry-up occurs.
Common section		
	Interrupt level	Using CPU-SBxx(FIT)GY: IRQ 5, 7, or 9
	Internal power consumption	5 VDC(±5%) 150 mA (Max.) *1
	Allowable distance of signal extension	Approx. 50m (depending on wiring environment)
	External dimensions (mm)	25.2 (W) x 64.7 (D) x 94.0 (H) (exclusive of protrusions)
	Weight of the module itself	100g
	Module connection method	Stack connection by the connector that is provided with the side of module
	Module installation method	One-touch connection to 35mm DIN rails (standard connection mechanism provided in the system)
	Applicable wire	AWG 28 - 16
	Applicable plug	FRONT-MC1,5/12-ST-3,81 (made by Phoenix Contact Corp.)

*1 The stack connector accepts currents of up to 3.0A (Max.).

Table 6.4. Installation Environment Requirements

Parameter	Requirement description
Operating temperature	0 - 50°C
Storage temperature	-10 - 60°C
Operating humidity	10 - 90%RH (No condensation)
Floating dust particles	Not to be excessive
Corrosive gases	None

⚠ CAUTION

When connecting the Module to a controller module, the internal power consumption should be taken into account. If the total current exceeds the capacity of the power supply unit, the integrity of the operation cannot be guaranteed. For further details, please see the Controller Module manual.

External Dimensions

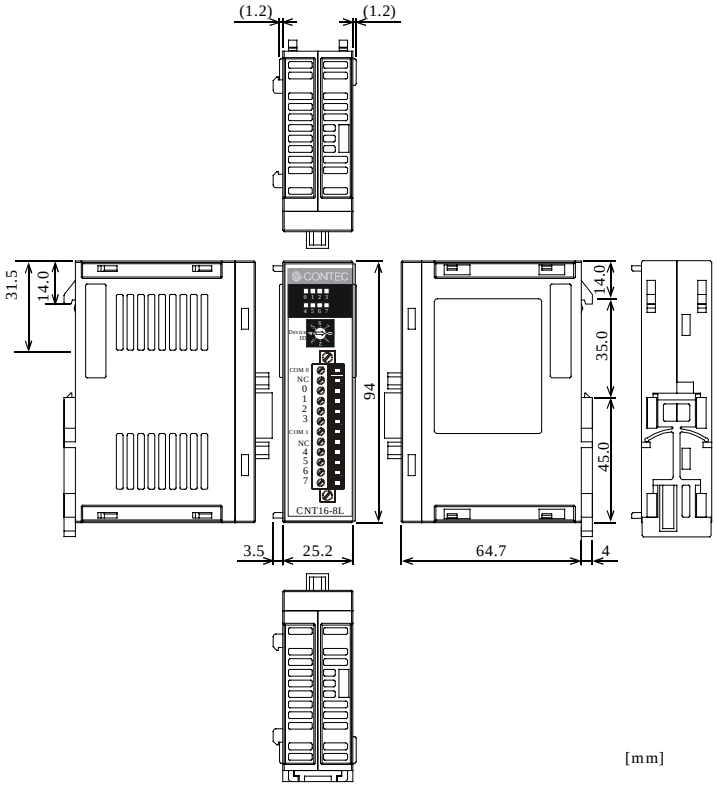


Figure 6.2. External Dimensions

CNT16-8(FIT)GY
CNT16-8L(FIT)GY
User's Manual

CONTEC CO., LTD.
November 2006 Edition

3-9-31, Himesato, Nishiyodogawa-ku, Osaka 555-0025, Japan
Japanese http://www.contec.co.jp/
English http://www.contec.com/
Chinese http://www.contec.com.cn/

[07262004]
[11152006_rev3]
Management No. A-40-668
Parts No. LYDQ282