<u>F&eIT Series</u>

Isolated High-Resolution
Analog Output Module

DAI16-4(FIT)GY

User's Manual

CONTEC CO.,LTD.

Check Your Package

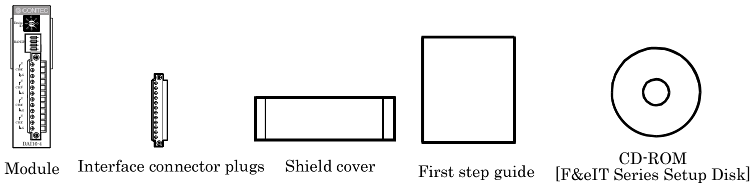
Thank you for purchasing the CONTEC product.

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[DAI16-4(FIT)GY] …1
- 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)

Table of Contents

1. Before Using the Product

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

About the Module

The DAI16-4(FIT)GY can control an external device by converting digital data into analog voltage or current signals.

This product can connect to F&eIT Series controller module <CPU-CAxx(FIT)GY, CPU-SBxx(FIT)GY) etc.> and construct the system.

The insulation between external signals and the Controller Module permits the use of the Controller Module without compromising the communications features of the latter. Please read this manual carefully to create application programs and configure the system, such as setting the switches and connecting it to external devices.

Features

- Analog output module providing high precision at a resolution of 16 bits.
- The output range is common to different channels, and can be selected from two output ranges: ±10V, and 0 - 20mA.
- A rotary switch allows you to set device IDs to help you keep track of device numbers.
- Flanged two-piece connector used to prevent disconnection from the connector on the controller module.
- Similar to other F&eIT series products, the system, in the module itself, incorporates a 35-mm DIN rail mounting mechanism as a standard item. 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 DAI16-4(FIT)GY can be connected to a variety of controllers.

Micro Controller 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

The functions and control of the DAI16-4(FIT)GY vary with the controller to which the DAI16-4(FIT)GYis connected.

Functions available with each controller connected

	CPU-SBxx(FIT)GY	CPU-CAxx(FIT)GY	SVR-MMF2(FIT)	SVR-MMF(FIT)GY
Software input range setting	o	o	*1	*1
D/A conversion with software command	o	o		
Continuous D/A conversion using the internal pacer clock	o *2			
Multi-channel simultaneous output function	o			
Interrupt function	o			
Device ID setting range	0 -7	0 -7	0 -7	0 -7

*1 For the function available, refer to the reference manual for the SVR-MMF2(FIT) and SVR-MMF(FIT)GY.

*2 Pacer clock setting: 10 - 1,073,741,824μsec

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			
	API-USBP(WDM)				
Control over the web				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 DAI16-4(FIT)GY 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 DAI16-4(FIT)GY can be accessed from the host computer over the network. The DAI16-4(FIT)GY 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 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

- Take the following precautions when handling this module.

- 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%, 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."
- Regarding “FCC PART 15 Class A Notice”
  This product has acquired the avobe-mentioned standard.
  However, a sufficient margin may not be secured for the standard. In this case, use a ferrite core (SEIWA E04SR200917 or a compatible product) for the interface cable or the power cable of the controller unit to which a ferrite core (SEIWA E04SR301334 or a compatible product) is connected.
  When attaching a ferrite core to the interface cable, just put the cable inside the case; when attaching a ferrite core to the connecting power cable of the controller unit, coil it around once near the connector while leaving it open, and then close it.

FCC PART 15Class 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

Humidity

10 - 90%RH (No condensation)

Corrosive gases

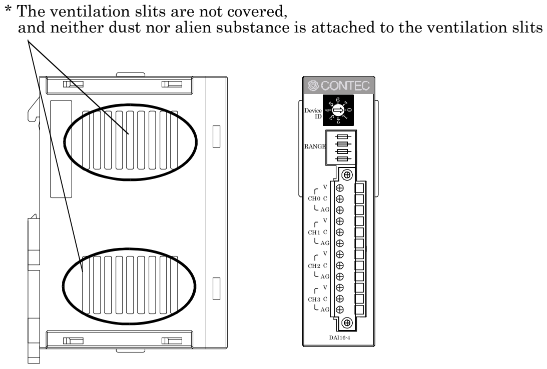
None

Floating dust particles

Not to be excessive

Inspection

Inspect the product periodically as follows to use it safely.


* The ventilation slits are not covered,
and neither dust nor alien substance is attached to the ventilation slits
CONTEC
Device ID
RANGE
CH0 V C AG
CH1 V C AG
CH2 V C AG
CH3 V C AG
DAI16-4

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.

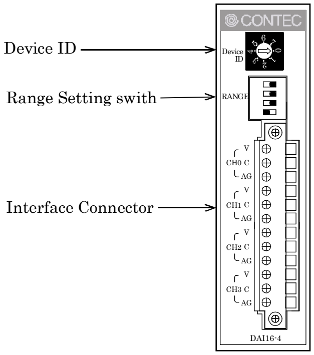


Figure 2.1. Names of module components

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.

Setup Method

A device ID can be set by turning the rotary switch on the device face.
To set a device ID, turn the switch knob.

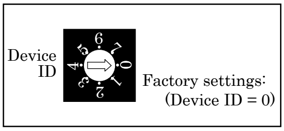


Figure 2.2. Setting a Device ID

Range Setting Switches

Set the voltage input or current input range setting depending on the measurement target. The input range is common to all channels; it cannot be set for each of them and must not be set to an invalid range.

Setup Method

To set the voltage or current input, use the corresponding DIP switch on the module face.

See the following sketches for reference to set each DIP switch

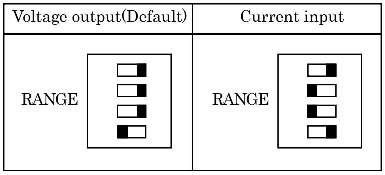


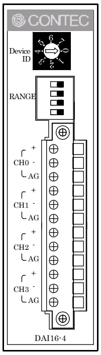
Figure 2.3. Setting a Output range

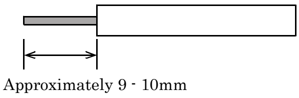
3. Connecting to an External Device

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 each terminal, strip the wire about 9 - 10 mm from the end and insert it into the opening. After inserting the wire, tighten the screw to fasten it. Compatible wires are AWG 28 - 16.





- Applicable connector
  3.81mm-pitch, 12-pin type of rated current 8A
  MC-1,5/12-GF-3,81 [Made by Phoenix Contact]

- Applicable plug(accessory bundled)
  Front-screw type with connector locking flange
  FRONT-MC 1,5/12-STF-3,81 [Made by Phoenix Contact]
  Applicable cable 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 (1 group) connector that is provided on the Module face.

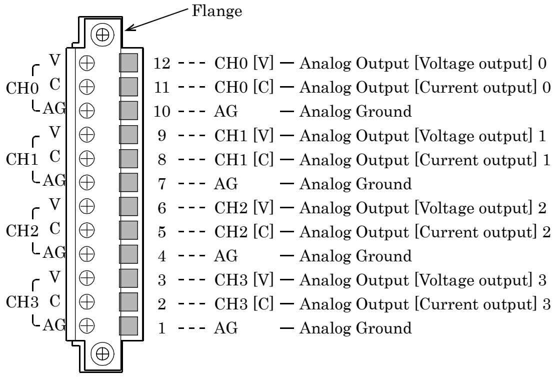

Flange
CH0 V
CH0 C
CH0 AG
CH1 V
CH1 C
CH1 AG
CH2 V
CH2 C
CH2 AG
CH3 V
CH3 C
CH3 AG
12 --- CH0 [V] — Analog Output [Voltage output] 0
11 --- CH0 [C] — Analog Output [Current output] 0
10 --- AG — Analog Ground
9 --- CH1 [V] — Analog Output [Voltage output] 1
8 --- CH1 [C] — Analog Output [Current output] 1
7 --- AG — Analog Ground
6 --- CH2 [V] — Analog Output [Voltage output] 2
5 --- CH2 [C] — Analog Output [Current output] 2
4 --- AG — Analog Ground
3 --- CH3 [V] — Analog Output [Voltage output] 3
2 --- CH3 [C] — Analog Output [Current output] 3
1 --- AG — Analog Ground

Figure 3.2. Signal layout on the interface connector <DAI16-4(FIT)GY>

Connecting an Analog Output Signal

Connecting a Voltage Output

Connecting with a Flat Cable

This example involves connecting the voltage output and the analog ground for each channel to the input and the ground of an external device, using a flat cable.

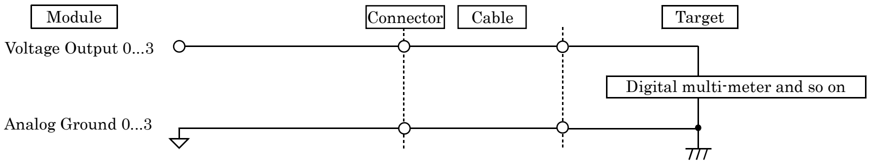


Figure 3.3. Connecting a Voltage Output (Flat Cable)

Connecting with a coaxial cable

A coaxial cable can be used in situations where the module is at a relatively large distance from the external device or when the noise immunity of the module must be improved. In this case, the voltage output and the analog ground for each channel are connected to the input and the ground, respectively, of the external device by using the core wire and the shield braid of the coaxial cable.

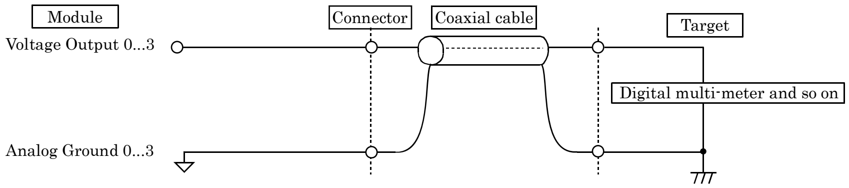


Figure 3.4. Connecting a Voltage Output (Coaxial Cable)

⚠ CAUTION

- When the power is turned on or the module is reset, the voltage output signal will be 0V.
- To avoid any malfunction, the voltage output signal should not be connected to the analog ground.
- To avoid any malfunction, the voltage output signal should not be connected to another analog output signal or the output signal of an external device.
- To avoid any malfunction, the connector plug should not be attached or detached when the power for the module or the external device is on.
- The maximum current capacity for a voltage output signal is ±5mA. To avoid any malfunction, do not connect an external device that generates a load exceeding this range.
- In situations where the connecting cable is subject to the effects of noise, the accurate voltage output can fail. The connecting cable should be installed away from any source of noise.
- In situations where the connecting cable is excessively long, the accurate voltage output can fail. Use a cable that is as short as possible.
- Because the D/A converter in the module does not contain a built-in deglitcher, a glitch can sometimes occur.

Connecting a Current Output

Example of Connecting a Negative Load Resistor

Two types of connection methods can be employed: fixed load and floating load. If the module is used as a load resistance, multiple current loops can be implemented by using the same power supply. The use of current output requires an external power supply (20 - 24V). In such a case, a power supply with a small ripple should be used in order to avoid an adverse impact on the conversion accuracy due to a large power supply ripple.

The load resistance R_L that is connected to the current output of each channel should be less than 400Ω, including the wire resistance.

Example of Connecting with a Flat Cable

Use a flat cable to connect the voltage output and the analog ground of each channel to the load resistance R_L.

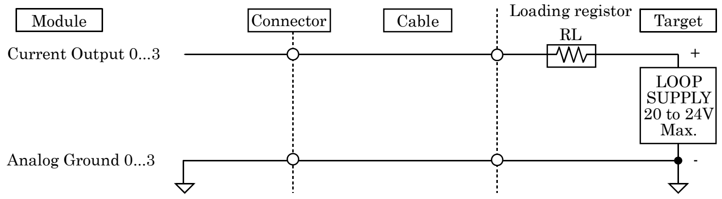


Figure 3.5. Connecting to a Floating Load (Flat Cable)

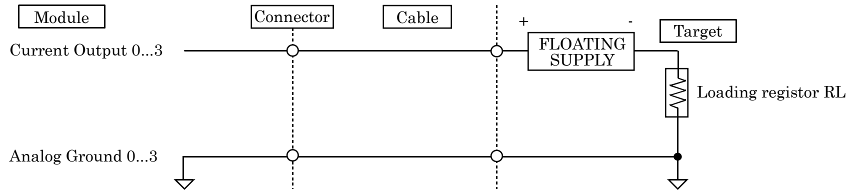


Figure 3.6. Connecting to a Fixed Load (Flat Cable)

Example of Connecting with a Coaxial Cable

A coaxial cable can be used in situations where the module is at a relatively large distance from the external device or when the noise immunity of the module must be improved. In this case, the current output and the analog ground for each channel are connected to the load resistance RL of the external device by using the core wire and the shield braid of the coaxial cable.

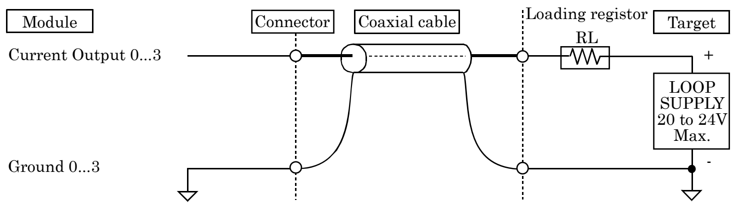


Figure 3.7. Connecting to a Floating Load (Coaxial Cable)

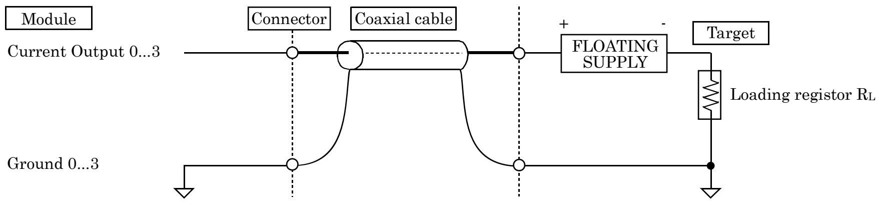


Figure 3.8. Connecting to a Fixed Load (Coaxial Cable)

⚠ CAUTION

- When the power is turned on or the module is reset, the current output signal will be 10 mA.
  If there is a problem with the current value output when the power is turned on, provide the module with an external relay.
- To avoid any malfunction, the current output signal should not be connected to the analog ground.
- To avoid any malfunction, the current output signal should not be connected to another analog output signal or the output signal of an external device.
- To avoid any malfunction, the connector plug should not be attached or detached when the power for the module or the external device is on.
- In situations where the connecting cable is subject to the effects of noise, the accurate current output can fail. The connecting cable should be installed away from any source of noise.

Fitting the Shield Cover

⚠ CAUTION

- Damage to internal devices may occur if the interface connector terminal is subject to a large static electric charge. Please fit the supplied shield cover to the connector if there is a risk of a large static electric charge.
- Fitting the supplied shield cover is not required if there is no risk of a large static electric charge occurring, such as when the connector is attached to the internal housing or similar.

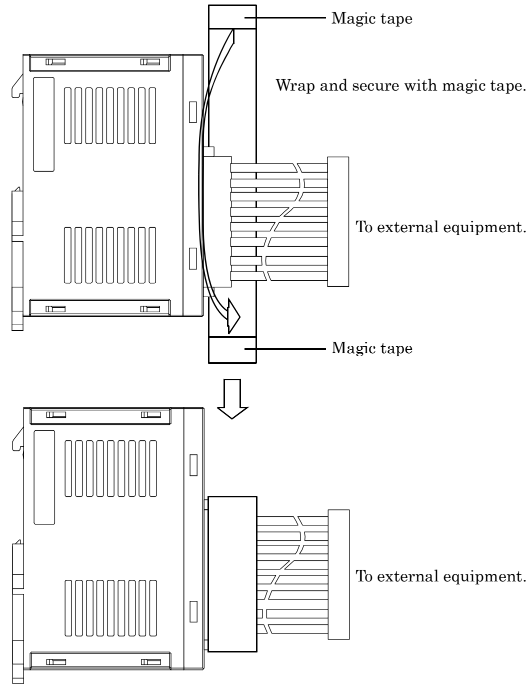


Figure 3.9. Fitting the Shield Cover

4. Using the I/O Address Map

Starting I/O Address

When connected to a CPU-SBxx(FIT)GY, the DAI16-4(FIT)GY 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 DAI16-4(FIT)GY.

Because the address bus on which the I/O address space is specified in not fully decoded in 16 bits, four starting I/O addresses exist, with one per 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.

Control examples are coded in C. Modify them in the appropriate language for the target OS or development system.

Table 4.1. List of Starting I/O Addresses

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

Input Port

Starting I/O address Input	D7	D6	D5	D4	D3	D2	D1	D0
	Products Category				**Revision Data**			
+0 (00h)	0	0	1	0	Revision Data 3	Revision Data 2	Revision Data 1	Revision Data 0
	Products ID Number							
+1 (01h)	ID Data7	ID Data6	ID Data5	ID Data4	ID Data3	ID Data2	ID Data1	ID Data0
	Interrupt Status							
+2 (02h)	Enable	Status	0	0	0	IRQ9	IRQ7	IRQ5
+3 (03h) – +21 (15h)	N/A							
	Analog Output Status 0							
+22 (16h)	0	0	Paccer Clock Error	Paccer Clock Input	Data Set Error	FIFO Memory Empty	Data Write Enable	Data Set Busy
	Analog Output Status 1							
+23 (17h)	0	0	0	FIFO Memory Flag	0	0	0	0
+24 (18h) – +31 (1Fh)	N/A							

Figure 4.1. Input port

Output Port

Starting I/O address	D7	D6	D5	D4	D3	D2	D1	D0
Output +0 (00h)	N/A							
+1 (00h)	**Interrupt Status**							
+2 (02h)	Enable	N/A	N/A	N/A	N/A	IRQ9 Data	IRQ7 Data	IRQ5 Data
+3 (03h) … +15 (0Fh)	N/A							
	Analog Output Data(Lower)							
+16 (10h)	Conversion Data7	Conversion Data6	Conversion Data5	Conversion Data4	Conversion Data3	Conversion Data2	Conversion Data1	Conversion Data0(LSB)
	Analog Output Data(Upper)							
+17 (11h)	Conversion Data15 (MSB)	Conversion Data14	Conversion Data13	Conversion Data12	Conversion Data11	Conversion Data10	Conversion Data9	Conversion Data8
	Channel Data							
+18 (12h)	All Channel	End Channel	N/A	N/A	N/A	N/A	Channel Data1	Channel Data0
+19 (13h) … +21 (15h)	N/A							
	Status Reset 0							
+22 (16h)	N/A	N/A	Pacer Clock Error	Pacer Clock Input	Data Set Error	FIFO Memory Empty	Data Write Enable	N/A
	Status Reset 1							
+23 (17h)	N/A	N/A	N/A	FIFO Memory Flag	N/A	N/A	N/A	N/A
	Command							
+24 (18h)	Command Data7	Command Data6	Command Data5	Command Data4	Command Data3	Command Data2	Command Data1	Command Data0
+25 (19h) … +27 (1Bh)	N/A							
	Setting Data 0							
+28 (1Ch)	Setting Data07	Setting Data06	Setting Data05	Setting Data04	Setting Data03	Setting Data02	Setting Data01	Setting Data00
	Setting Data 1							
+29 (1Dh)	Setting Data15	Setting Data14	Setting Data13	Setting Data12	Setting Data11	Setting Data10	Setting Data09	Setting Data08
	Setting Data 2							
+30 (1Eh)	Setting Data23	Setting Data22	Setting Data21	Setting Data20	Setting Data19	Setting Data18	Setting Data17	Setting Data16
	Setting Data 3							
+31 (1Fh)	Setting Data31	Setting Data30	Setting Data29	Setting Data28	Setting Data27	Setting Data26	Setting Data25	Setting Data24

Figure 4.2. Output port

Specifications Common to F&eIT Products

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

Product Information

	D7	D6	D5	D4	D3	D2	D1	D0
Input	**Product Category**				**Revision Data**			
+0 (00h)	0	0	1	0	Rivision Data3	Rivision Data2	Rivision Data1	Rivision Data0
Input	**Product ID Number**							
+1 (01h)	ID Data7	ID Data6	ID Data5	ID Data4	ID Data3	ID Data2	ID Data1	ID Data0

Figure 4.3. Product information

- 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 DAI16-4(FIT)GY, the code is "2h ".

Table 4.2. Product Category

Code	Function
0	Extension BUS
1	Digital input-output
2	Analog input-output
3	Counter
4	Serial communication
5	GPIB
6-F	Reserved

- Product ID Number [D7 - D0] :
  This is the product ID within the same product category. For the DAI16-4(FIT)GY, the product ID is "3h".

Following are examples of the initialization that is performed:

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

*ADR is the starting I/O address for the DAI16-4(FIT)GY.

Interrupt Status

This is a common port on which the interrupt status requested by the Module can be verified. Although in this example values are assigned centered on the status concerning interrupt levels, information on interrupt sources varies from module to module.

	D7	D6	D5	D4	D3	D2	D1	D0
Input	Interrupt Status							
+2 (02h)	Enable	Status	0	0	0	IRQ9	IRQ7	IRQ5

Figure 4.4. Interrupt status

- Enable [D7] :

  This 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 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".

Following are examples of the initialization that is performed :

```
IrqStatus = inp( ADR+2 );
```

Setting an Interrupt Level

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Interrupt Data							
+2 (02h)	Enable	N/A	N/A	N/A	N/A	IRQ9 Data	IRQ7 Data	IRQ5 Data

Figure 4.5. Setting an interrupt level

- Enable [D7] :
  This bit enables an interrupt source.
- IRQ* [D2 - D0] :
  The interrupt level used by the module is set in these bits.

Following are examples of initialization settings that can be effected :

The interrupt level to be used is assigned to IRQ5.

```
outp( ADR+2, 0x81 );
```

FIFO Function Overview

The DAI16-4(FIT)GY has FIFO memory for 64 data items, capable of temporarily storing converted data. Check the following default settings for FIFO memory before use.

Analog output status

	D7	D6	D5	D4	D3	D2	D1	D0
Input	**Analog Output Status 1**							
+23 (17h)	0	0	0	FIFO Memory Flag	0	0	0	0

Figure 4.6. Analog output status

- FIFO Memory Flag [D4]:

  The status is set when the number of data items stored to FIFO memory changes from the number of data items set in the FIFO Flag register or greater to a smaller number. The status is not set unless the number of stored data items falls below the number of data items set in the FIFO Flag register from a greater number.

  The status is cleared by setting the FIFO memory flag status reset bit in the analog output status reset port to 1.

FIFO flag register

The FIFO Memory Flag status is set when FIFO memory stores the set number of data items.

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Command**							
+24 (18h)	0	0	0	0	0	1	1	1

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**FIFO Flag**							
+28 (1Ch)	0	Flag Data6	Flag Data5	Flag Data4	Flag Data3	Flag Data2	Flag Data1	Flag Data0

Figure 4.7. FIFO flag register

FIFO flag register setting examples are given below.

```
outp( ADR+24, 0x7 );
outp( ADR+28, FlagData );
```

Number of data items stored in FIFO memory

You can check the number of data items remaining in FIFO memory.

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Command							
+24 (18h)	0	0	0	0	1	0	0	0

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Number of FIFO Data							
+28 (1Ch)	0	Number of Data6	Number of Data5	Number of Data4	Number of Data3	Number of Data2	Number of Data1	Number of Data0

Figure 4.8. Number of data items stored in FIFO memory

The following example can be used to check the number of data items stored in FIFO memory.

```
outp( ADR+24, 0x8 );
NumberOfData = inp( ADR+28 );
```

FIFO memory clearance

Stored data is cleared.

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Command							
+24 (18h)	0	0	0	0	1	0	0	1

Figure 4.9. FIFO memory clear

The following example can be used to set the FIFO memory clear.

```
outp( ADR+24, 0x9 );
```

*Each status is also cleared to 0 by the following condition

- Initialization command output

Overview of the D/A Conversion Function

The analog output function produces specified 16-bit digital data from a specified channel as corresponding voltage or current analog output.
By setting a specific analog output mode, you can specify the method by which the analog output is to be produced.
Two analog output modes are supported:

- Transparent output mode
- Synchronous output mode

The transparent output mode updates the analog output from a specified channel immediately after output data is set.
The synchronous output mode pre-sets output data to the channels on which analog output is to be updated, and simultaneously updates analog output from multiple channel when the synchronous output command is issued.

Note that DSB and EOC represent the data set busy status and last channel converted.

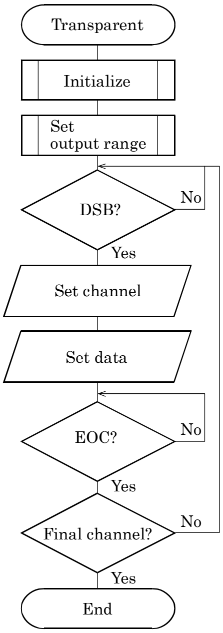


Figure 4.10 Transparent Output Mode

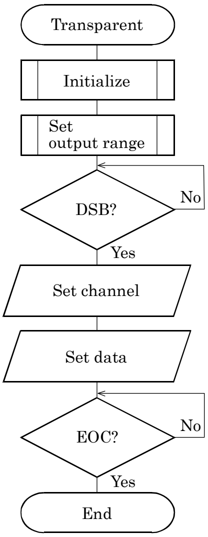


Figure 4.11 Synchronous Output Mode

Initialization

This step initializes the analog input function.

This command clears all settings and status, and returns the module to the "initialized state", which is the same as when the power is turned on and the RESET button is pressed.

During the initialization, the control port assumes the following state:

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Command**							
+24 (18h)	0	0	0	0	0	0	0	0

Figure 4.12. Initialization

Following are examples of the initialization that is effected :

```
outp( ADR+24, 0x0 );
```

Setting D/A Conversion Conditions

This step sets D/A conversion conditions.
In terms of procedures, first a D/A conversion condition setup command is issued, and then setting data is output.

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Command**							
+24 (18h)	0	0	0	0	0	0	1	0

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Conversion Setting**							
+28 (1Ch)	0	0	0	0	0	Output Mode	Pacer Clock	Coversion Mode

Figure 4.13. Setting D/A Conversion Conditions

- Output mode [D2]:
  This step sets the specific D/A conversion output method to be employed.
  Two output methods are available: "transparent mode", in which analog outputs from channels are updated in arbitrary, software-based timing; and "synchronous output mode", in which the analog outputs from multiple channels are simultaneously updated in arbitrary, software-based timing.

Output Mode	[0]	: Transparent	* Initialized state
	[1]	: Synchronous output	

- Pacer clock [D1]:
  This option is set only if the clock mode is selected as a D/A conversion mode.

Pacer Clock	[0]	: Internal Pacer Clock	* Initialized state
	[1]	: Reserved	

- D/A conversion mode [D0]:
  This step sets the timing for D/A conversion output.
  Two sampling modes are supported: "software mode", in which analog output is produced from a software-specified channel; and "clock mode", in which periodic D/A conversions are performed by means of clock signals.

Sampling Mode	[0]	: Software Command	* Initialized state
	[1]	: Clock	

Following are examples in which D/A conversion conditions are specified in high-level languages:

```
outp( ADR+24, 0x2 );
outp( ADR+28, ConditionData );
```

Setting an Output Range

The output range refers to the voltage range over which analog signals are output.

This setting is common to all channels, and involves the conversion of digital signals into analog output signals at a 16-bit resolution level.

The output range setup control port assumes the following state:

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Command							
+24 (18h)	0	0	0	0	0	0	1	1

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Range Setting							
+28 (1Ch)	Range Data7	Range Data6	Range Data5	Range Data4	Range Data3	Range Data2	Range Data1	Range Data0

Figure 4.14. Setting an Output Range

Table 4.3. Output Range and Setup Data

Range	Output range
00h	±10V
01 - BFh	Not decied
C0h	0 - 20mA
C1h or later	Not decied

Following are examples in which an output range is set :

The output range is set to -10 - 10V :

```
outp( ADR+24, 0x3 );
outp( ADR+28, 0x0 );
```

Setting an Internal Pacer Clock

When either the "clock mode" or the "internal pacer clock" is selected as a D/A conversion condition, this step specifies a clock cycle (clock data). In the initialized state, the clock data is undefined. When using an internal pacer clock, you must set the requisite clock data.
Clock data is specified in 250-nsec increments.
The allowable range is 10,000nsec to 1,073,741,824,000nsec (approximately 17 minutes 54 seconds), which corresponds to 39 to 4,294,967,295 setup data.

The relationship between clock cycles and setup data can be expressed in the following formula:

$$\textit{Clock data} = \frac{\textit{Pacer clock}}{250} - 1$$

The pacer clock is specified in units of nanoseconds.
The value of the pacer clock must satisfy the following expression:

$$\text{Pacer clock} \geq \underset{(10\mu\text{sec})}{10000\text{nsec}} \text{ x Number of specified channels}$$

D/A conversions in accurate cycles cannot be performed if the specified value is less than the conversion time for a specified number of channels.
The internal pacer clock-setting control port assumes the following state:

Output	D7	D6	D5	D4	D3	D2	D1	D0
	Command							
+24 (18h)	0	0	0	0	0	1	0	0

Output	D7	D6	D5	D4	D3	D2	D1	D0
	Timer Data 0							
+28 (1Ch)	Timer Data07	Timer Data06	Timer Data05	Timer Data04	Timer Data03	Timer Data02	Timer Data01	Timer Data00
	Timer Data 1							
+29 (1Dh)	Timer Data15	Timer Data14	Timer Data13	Timer Data12	Timer Data11	Timer Data10	Timer Data09	Timer Data08
	Timer Data 2							
+30 (1Eh)	Timer Data23	Timer Data22	Timer Data21	Timer Data20	Timer Data19	Timer Data18	Timer Data17	Timer Data16
	Timer Data 3							
+31 (1Fh)	Timer Data31	Timer Data30	Timer Data29	Timer Data28	Timer Data27	Timer Data26	Timer Data25	Timer Data24

Figure 4.15. Setting an Internal Pacer Clock

Following are examples in which an internal pacer clock is specified:

```
outp( ADR+24, 0x4 );
outp( ADR+28, ClockData0 );
outp( ADR+29, ClockData1 );
outp( ADR+30, ClockData2 );
outp( ADR+31, ClockData3 );
```

Setting a Conversion Channel

This step specifies the channels from which analog output is to be produced.
For byte access, conversion channels should be specified in the following order: channel specification → D/A conversion data (low bytes) → D/A conversion data (high bytes)

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Channel Data**							
+18 (12h)	All Channel	End Channel	N/A	N/A	N/A	N/A	Channel Data1	Channel Data0

Figure 4.16. Setting a Conversion Channel

- All channels [D7]:
  Assigning the value "1" to this bit causes the D/A conversion data specified in the following step to be output from all channels.

  Following are examples in which a conversion channel is specified:

```
outp( ADR+18, 0x80 );
outp( ADR+16, LowerData );
outp( ADR+17, UpperData );
```

- Final channel [D6]:
  This option is enabled when the synchronous output mode is on.
  Assigning the value "1" to this bit specifies the final channel for which the D/A conversion data to be specified in the following step is valid.
  Channels for which D/A conversion data can be valid are channels 0 through the final channel.

  Following are examples where data is output synchronously from channels 0 and 1:

```
outp( ADR+24, 0x2 );
outp( ADR+28, 0x5 );
outp( ADR+18, 0x00 );
outp( ADR+16, LowerData0 );
outp( ADR+17, UpperData0 );
outp( ADR+18, 0x41 );
outp( ADR+16, LowerData1 );
outp( ADR+17, UpperData1 );
```

Setting Conversion Data

Conversion data is specified in offset binary. The relationship between conversion data and analog output is indicated by the following formula:

$$Data = \frac{(voltage + offset)}{span} \times 2^{12}$$

Table 4.4. Output Range

Output	Offset	Span
-10V - +10V	10	20
0mA - +20mA	0	20

Table 4.5. Example of Conversion Data over a ±10V Range

Analog output value (±10V range)	16-bit conversion data
	Offset binary
+9.9997V	FFFFh
:	:
0.00030V	8001h
0.00000V	8000h
-0.00030V	7FFFh
:	:
-10.00000V	0000h

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Channel Data							
+18 (12h)	All Channel	End Channel	N/A	N/A	N/A	N/A	Channel Data1	Channel Data0

	D7	D6	D5	D4	D3	D2	D1	D0
Output	Analog Output Data(Lower)							
+16 (10h)	Conversion Data7	Conversion Data6	Conversion Data5	Conversion Data4	Conversion Data3	Conversion Data2	Conversion Data1	Conversion Data0 (LSB)
	Analog Output Data(Upper)							
+17 (11h)	Conversion Data15 (MSB)	Conversion Data14	Conversion Data13	Conversion Data12	Conversion Data11	Conversion Data10	Conversion Data9	Conversion Data8

	D7	D6	D5	D4	D3	D2	D1	D0
Input	Analog Output Status 0							
+22 (16h)	0	0	Pacer Clock Error	Pacer Clock Input	Data Set Error	FIFO Memory Empty	Data Write Enable	Data Set Busy

Figure 4.17. Output Port for Setting Conversion Data

The channel/conversion data output control procedure is described below.
Details on analog output and the interrupt status is given in the following section.
Following are examples in which a D/A conversion process is started.

For operation in clock mode, set data for two clock cycles once conversion has been started.

```
while( inp( ADR+22 &1) );
outp( ADR+18, 0x0 );
outp( ADR+16, LowerData );
outp( ADR+17, UpperData );
while( inp( ADR+22 & 2 );
```

Details on the Analog Output Status

The analog output status indicates the status of the D/A conversion operation.

Input	D7	D6	D5	D4	D3	D2	D1	D0
	Analog Output Status 0							
+22 (16h)	0	0	Pacer Clock Error	Pacer Clock Input	Data Set Error	FIFO Memory Empty	Data Write Enable	Data Set Busy
	Analog Output Status 1							
+23 (17h)	0	0	0	FIFO Memory Flag	0	0	0	0

Figure 4.18. Analog Output Status

- Dataset Busy (DSB) [D0]:
  This status bit is set to 1 when conversion data in FIFO memory is written to the register to the D/A converter. The bit is cleared upon completion of writing to the D/A converter.
- Data Write Enable (DWE) [D1]:
  This status bit is set to 1 when analog output data is updated with FIFO memory ready to read the next data from. The bit is cleared by setting the data write enable status bit in the analog output status reset port to 1.
- FIFO Memory Empty (FME) [D2]:
  This status bit is set to 1 when FIFO memory contains no data. The bit is cleared by setting the FIFO memory empty status bit in the analog output status reset port to 1.
- Data Set Error (DSE) [D3]:
  This status bit is set to 1 when conversion data fails to be written normally to the D/A converter. The error occurs in the following cases:
  - Conversion data stored in FIFO memory reaches 64 words.
  - FIFO memory contains no data to convert during timer-based cyclic conversion (in clock mode).
- Pacer Clock Input Status (PCI) [D4]:
  [1] is set to this status bit when a pacer clock is input after the timer start command is issued in the clock mode. This bit is cleared when [1] is set to the clock input status bit for the analog output status reset port. *
- Pacer Clock Error Status (PCE)[D5]:
  [1] is set to this status bit when a pacer clock is re-entered with the pacer clock input status bit being [1], during the operation of the timer in the clock mode.
  This bit is cleared when [1] is set to the pacer clock error status bit for the analog output status reset port. *
  For the FIFO Memory Flag, refer to FIFO Function Overview.

* These status bits are also [0] cleared under the following conditions:
- When the initialization command is issued
- When the D/A conversion condition-setting command is issued

Details on Resetting the Status

This step clears the analog output status.

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Status Reset 0**							
+22 (16h)	0	0	Pacer Clock Error	Pacer Clock Input	Data Set Error	FIFO Memory Empty	Data Write Enable	0
	Status Reset 1							
+23 (17h)	0	0	0	FIFO Memory Flag	0	0	0	0

Figure 4.19. Status reset

- Data Write Enable Status Clear (DWE) [D1]:
  Setting the value [1] clears the data write enable status.
- FIFO Memory Empty Status (FME) [D2]:
  Setting the value [1] clears the FIFO memory empty status.
- Data Set Error Status (DSE) [D3]:
  Setting the value [1] clears the data set error status.
- Pacer Clock Input Status Clear (PCI)[D4]:
  Setting the value [1] clears the pacer clock input status.
- Pacer Clock Error Status Clear (PCE)[D5]:
  Setting the value [1] clears the pacer clock error status.
- FIFO Memory Flag Status Clear (FMF)[D4]:
  Setting the value [1] clears the FIFO memory flag status.

Interrupt Function

This option allows you to use the hardware interrupt function. For interrupt levels, a level that is set by the Module will be used. When using the interrupt function, you can pre-select one of the following status conditions as an interrupt source (multiple settings allowed):

Table 4.6. Interrupt function

Status	Explanation
Pacer Clock Input	When a pacer clock is input
Pacer Clock Error	When the pacer clock error status is set
Data Write Enable	When writing conversion data is enabled
Data Set Error	When writing conversion data fails to be completed normally
FIFO Memory Empty	When a FIFO memory empty status reset occurs
FIFO Memory Flag	FIFO Memory Flag status set

An interrupt request signal is generated simultaneously with the setting of the status that is specified as an interrupt source. If two or more interrupt sources are specified, you can specify a specific interrupt signal generation source by entering a status in the interrupt handler.

Setting an Interrupt Source

This option allows you to specify an interrupt signal generation source.

The control port that sets an interrupt source assumes the following state:

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Command**							
+24 (18h)	0	0	0	0	0	0	0	1

	D7	D6	D5	D4	D3	D2	D1	D0
Output	**Interrupt Mask0**							
+28 (1Ch)	1	1	Pacer Clock Error	Pacer Clock Input	Data Set Error	FIFO Memory Empty	Data Write Enable	1
	Interrupt Mask1							
+29 (1Dh)	1	1	1	FIFO Memory Flag	1	1	1	1

Figure 4.20. Interrupt Source Output Port

When the value [1] is output as an interrupt source, the control port is masked; when the value [0] is output, the port is set as an interrupt source.

[1] Masked * Initialized state
[0] Interrupt Request Enable

Following are examples in which a timer cycle is set:

```
outp( ADR+24, 0x1 );
outp( ADR+28, InterruptFactor );
```

List of Commands

Following is a list of DAI16-4(FIT)GY commands that are issued to "Output port +24":

Table 4.7. Command list

No.	HEX	Function	Data length
00	0	Initialization	0-bit
01	1	Masks interrupt source	8-bit
02	2	Sets D/A conversion	8-bit
03	3	Sets output range	8-bit
04	4	Sets internal pacer clock	32-bit
05	5	Starts timer	0-bit
06	6	Stops timer	0-bit
07	7	FIFO memory flag	8-bit
08	8	FIFO storage data number	8-bit
09	9	FIFO memory clear	0-bit

Examples

Software Mode

Flowchart

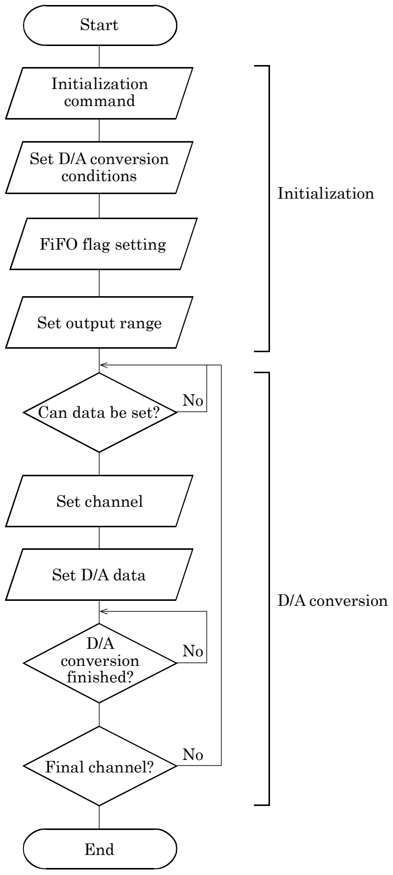


Figure 4.21. Software mode

Sample program

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

      DEVICE ID:         0
      Mode:              Software Mode, Partially Transparent
      Channel:           0 to 3ch
      Range:             -10 to 10V
      Internal Clock:    N/A
      Interrupt:         N/A

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

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

/* ----- Prototype -------------------------------------------------------- */
void  main( void );

/* ----- Main -------------------------------------------------------------
*/
void  main( void )
{
   unsigned char      UpperData = 0xc0;
   unsigned char      LowerData = 0x00;
   unsigned char      i;
   float           VDAT;

   outp( ADR+0x18, 0x00 );                      /* Initialize */
   outp( ADR+0x18, 0x02 );                      /* D/A Conversion Mode */
   outp( ADR+0x1c, 0x00 );                      /* Software */
   outp( ADR+0x18, 0x03 );                      /* Range */
   outp( ADR+0x1c, 0x00 );                      /* -10 to 10V */

   VDAT = (UpperData*0x100+LowerData)*20.0f/65536.0f-10.0f;

   for(i = 0; i < 4; i++) {
      while( ( (unsigned char)inp( ADR+0x16 ) ) & 0x01 );
      outp( ADR+0x12, 0x00+i );                 /* Set Channel */
      outp( ADR+0x10, LowerData );              /* Set Lower Data */
      outp( ADR+0x11, UpperData );              /* Set Upper Data */
      while( !((unsigned char)inp( ADR+0x16 ) & 0x02) );
      printf("%01dch  %02x%02x   %7.3f[V]\n", i, UpperData, LowerData, VDAT);
   }
}

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

Clock Mode (No Interrupts)

Flowchart

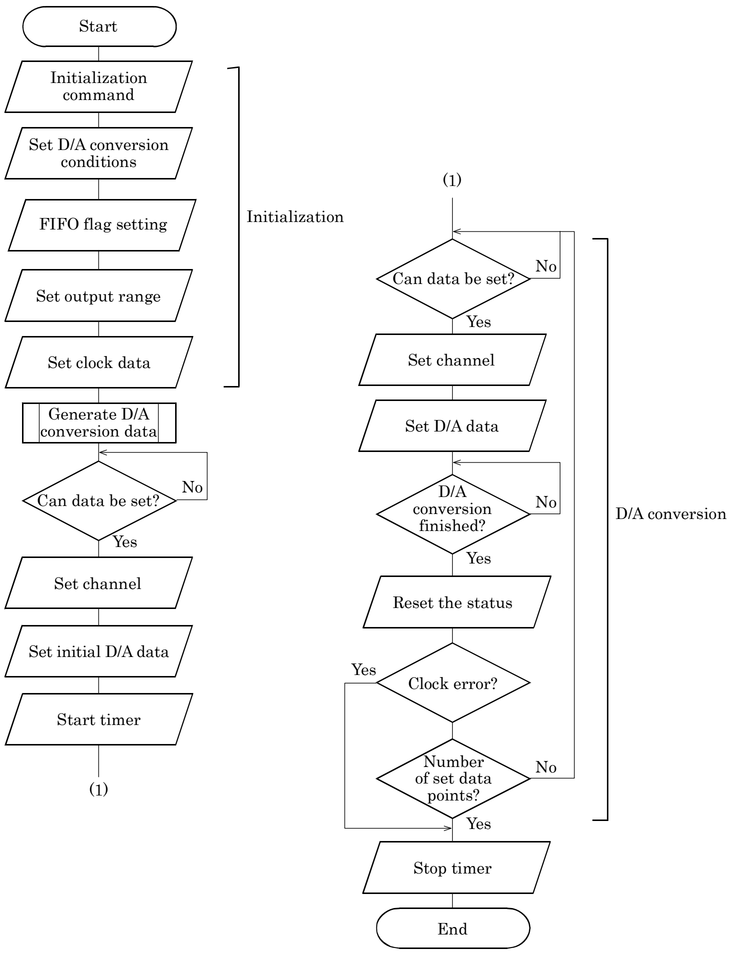


Figure 4.22. Clock Mode (No Interrupts)

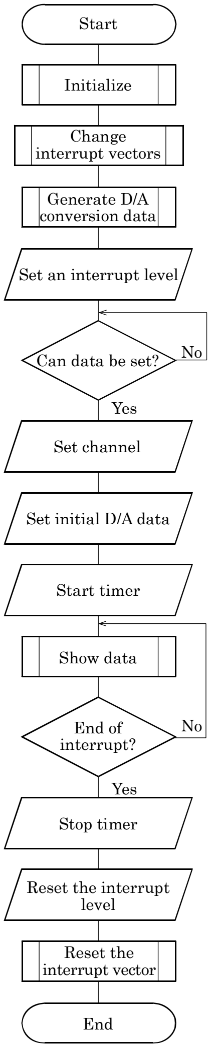
Note) Before starting timer, be sure to do the two-clocks or more date setting.

Sample program

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

      DEVICE ID:          0
      Mode:               Clock Mode, Simultaneous Conversion
      Channel:            0 to 3ch
      Range:              -10 to 10V
      Internal Clock:     250msec (250ns x 1,000,000)
      Interrupt:          N/A

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

/* ----- Constant ----------------------------------------------------------- */
#define   ADR      0x0800                        /* I/O address */
#define   CH       4                             /* Channels */
#define   NUM      11                            /* Conversion Times */

/* ----- Prototype ---------------------------------------------------------- */
void   main( void );

/* ----- Main ---------------------------------------------------------------
*/
void   main( void )
{
   unsigned char      UpperData[NUM], LowerData[NUM], sts;
   unsigned char      i, j;
   unsigned int    VDAT;
   float           Volt[NUM];

   outp( ADR+0x18, 0x00 );                       /* Initialize */
   outp( ADR+0x18, 0x02 );                       /* D/A Conversion Mode */
   outp( ADR+0x1c, 0x05 );                       /* Software */
   outp( ADR+0x18, 0x07 );                       /* FIFO Flag */
   outp( ADR+0x1c, 0x20 );
   outp( ADR+0x18, 0x03 );                       /* Range */
   outp( ADR+0x1c, 0x00 );                       /* -10 to 10V */
   outp( ADR+0x18, 0x04 );                       /* Timer Setting */
   outp( ADR+0x1c, 0x3f );                       /* 250ns x 1,000,000 */
   outp( ADR+0x1d, 0x42 );
   outp( ADR+0x1e, 0x0f );
   outp( ADR+0x1f, 0x00 );

   for(i = 0; i < NUM; i++) {                    /* D/A Conversion Data */
      VDAT = (unsigned int)(65536.0f/(NUM-1)*i);
      if( i == NUM-1) VDAT = 0xffff;
      UpperData[i] = (unsigned char)( ( VDAT & 0xff00 ) >> 8 );
      LowerData[i] = (unsigned char)( VDAT & 0xff );
      Volt[i] = (float)VDAT*20.0f/65536.0f-10.0f;
   }

   for(j = 0; j < 2 ; j++) {
      for(i = 0; i < CH; i++) {                  /* Set Initial Data */
         do {
            sts = (unsigned char)inp( ADR+0x16 );
         } while( sts & 0x01 );
         if ( i == 3 )   outp( ADR+0x12, 0x40+i );/* Set Last Channel */
         else outp( ADR+0x12, i );               /* Set Channel */
         outp( ADR+0x10, LowerData[i+j] );               /* Set Lower Data */
         outp( ADR+0x11, UpperData[i+j] );               /* Set Upper Data */
      }
   }
   outp( ADR+0x18, 0x05 );                     /* Timer Start */
   for(i = 0; i < NUM; i++) {
      do {
         sts = (unsigned char)inp( ADR+0x16 );
      } while( (sts & 0x02) != 0x02 );
      printf("0 to 3ch  %02x%02x   %7.3f[V]   ",
         UpperData[i], LowerData[i], Volt[i]);

      for(j = 0; j < CH; j++) {                /* Set Initial Data */
         do {
            sts = (unsigned char)inp( ADR+0x16 );
         } while( sts & 0x01 );
         if ( i+2  <= NUM) {
            if ( j == 3 )   outp( ADR+0x12, 0x40+j );/* Set Last Channel */
            else outp( ADR+0x12, j );                   /* Set Channel */
            outp( ADR+0x10, LowerData[i+2] );  /* Set Lower Data */
            outp( ADR+0x11, UpperData[i+2] );  /* Set Upper Data */
         }
      }

      sts = (unsigned char)inp( ADR+0x16 );
      outp( ADR+0x16, sts & 0x12 );            /* Status reset */

      if( sts & 0x20 ) {
         i = NUM;
         printf("Clock Error\n");
      } else printf("\n");
   }
   outp( ADR+0x18, 0x06 );                     /* Timer Stop */
}

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

Clock Mode (with Interrupts)

Flowchart



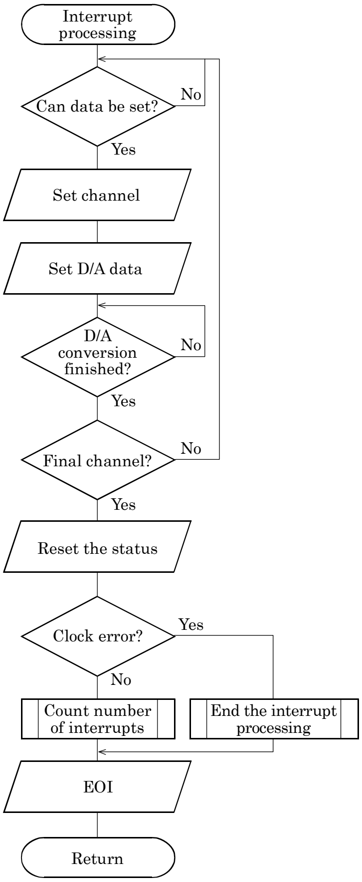


Figure 4.23. Clock Mode (with Interrupts) **Figure 4.24. Clock Mode (with Interrupts)**

Note) Before starting timer, be sure to do the two-clocks or more date setting.

Sample program

```
/*============================================================================
=
   Sample program 3

      DEVICE ID:         0
      Mode:              Clock Mode, Partially Simultaneous Conversion
      Channel:           0 to 3ch
      Range:             -10 to 10V
      Internal Clock:    1sec (250ns x 4,000,000)
      Interrupt:         IRQ5  11 times

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

/* ----- Constant ------------------------------------------------------------ */
#define   ADR      0x0800                       /* I/O address */
#define   CH       4                            /* Channels */
#define   NUM      11                           /* Conversion Times */
#define   IRQ5     0                            /* IRQ5 */
#define   IRQ7     1                            /* IRQ7 */
#define   IRQ9     2                            /* IRQ9 */

volatile  unsigned int DaData[CH][NUM];         /* D/A Data */
volatile  int          intcnt = 0;              /* interrupt counter */
volatile  int          IrqLevel = IRQ5;         /* interrupt level */
volatile  int          Irqsts;                  /* interrupt level */
int                    OrgMasterImr, OrgSlaveImr;
                                                /* original IMR */
unsigned char          IntVector[3] = { 0x0d, 0x0f, 0x71 };
                                                /* interuput 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( void );                       /* initialize */
void  ChgVect( void );                          /* change vector */
void  ResVect( void );                          /* restore vector */
void  _interrupt _far inthandler( void );       /* interrupt handler */
void  ( _interrupt _far *OrgVect)();            /* original interrupt vector
*/


/* ----- Initialize ----------------------------------------------------------
*/
void  Initialize( void )
{
   outp( ADR+0x18, 0x00 );                      /* Initialize */
   outp( ADR+0x18, 0x02 );                      /* D/A Conversion Mode */
   outp( ADR+0x1c, 0x05 );                      /* Software */
   outp( ADR+0x18, 0x07 );                      /* FIFO Flag */
   outp( ADR+0x1c, 0x20 );
   outp( ADR+0x18, 0x03 );                      /* Range */
   outp( ADR+0x1c, 0x00 );                      /* -10 to 10V */
   outp( ADR+0x18, 0x04 );                      /* Timer Data */
   outp( ADR+0x1c, 0xff );                      /* 250ns x 4,000,000 */
   outp( ADR+0x1d, 0x08 );
   outp( ADR+0x1e, 0x3d );
   outp( ADR+0x1f, 0x00 );
   outp( ADR+0x18, 0x01 );                /* Interrupt Factor */
   outp( ADR+0x1c, 0xef );                /* Sampling Clock Input Mask OFF */
}


/* ----- 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 */
}


/* ----- 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 )
{
   unsigned char      i;
   unsigned char      UpperData, LowerData, sts;

   _enable();                                   /* enable */
   intcnt++;
   do {
      sts = (unsigned char)inp( ADR+0x16 );
   } while( sts & 0x01 );

   for(i = 0; i < 4; i++) {
      do {
         sts = (unsigned char)inp( ADR+0x16 );
      } while( sts & 0x01 );
      if ( intcnt  <= NUM) {
            if( i == 3 ) outp( ADR+0x12, 0x40+i ); /* Set Last Channel */
            else  outp( ADR+0x12, i );                    /* Set Channel */
            UpperData = (unsigned char)( ( DaData[i][intcnt] >> 8 ) & 0xff );
            LowerData = (unsigned char)( DaData[i][intcnt] & 0xff );
            outp( ADR+0x10, LowerData );                  /* Set Lower Data */
            outp( ADR+0x11, UpperData );                  /* Set Upper Data */
        }
    }

    sts = (unsigned char)inp( ADR+0x16 );
    outp( ADR+0x16, sts );                       /* Status reset */
    if( sts & 0x20 )intcnt = 32767;

    _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 int    i, j;
    unsigned int    VDAT;
    float           Volt;
    unsigned char       UpperData, LowerData, sts;
    int             intcntnow = 0;

    for(j = 0; j < NUM; j++) {                   /* D/A Conversion Data */
        for(i = 0; i < CH; i++) {
            VDAT = (unsigned int)(65536.0f/(NUM-1)*j);
            if( j == NUM-1) VDAT = 0xffff;
            DaData [i][j] = VDAT;
        }
    }

    Initialize();                                        /* initialize */
    ChgVect();                                   /* change vector */

    for(j = 0; j < 2; j++) {
        for(i = 0; i < 4; i++) {                 /* Set Initial Data */
            do {
                sts = (unsigned char)inp( ADR+0x16 );
            } while( sts & 0x01 );
            if( i == 3 ) outp( ADR+0x12, 0x40+i ); /* Set Last Channel */
            else  outp( ADR+0x12, i );                    /* Set Channel */
            UpperData = (unsigned char)( ( DaData[i][0] >> 8 ) & 0xff );
            LowerData = (unsigned char)( DaData[i][0] & 0xff );
            outp( ADR+0x10, LowerData );                  /* Set Lower Data */
            outp( ADR+0x11, UpperData );                  /* Set Upper Data */
        }
    }
    outp( ADR+0x2, IntEnable[IrqLevel] );        /* interrupt level */
    outp( ADR+0x18, 0x05 );                      /* Timer Start */

    while( intcntnow <= NUM ) {
        printf("interrupt count=%02d   ", intcntnow);
        if ( intcntnow > 0 ) {
          for(i = 0; i < CH; i++) {
             Volt = DaData[i][intcntnow-1]*20.0f/65536.0f-10.0f;
             printf("%01dch %7.3fV  ", i, Volt);
          }
       }
       printf("\n");
       intcntnow = intcnt;
    }

    outp( ADR+0x18, 0x06 );                         /* Timer Stop */
    outp( ADR+0x2, 0x0 );                           /* interrupt level */
    ResVect();                                      /* restore vector */

    printf("\n\n");
    if( intcnt == 32767 )  printf("Pacer Clock Error\n");
}

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

5. Using the Memory Address Map

When connected to a CPU-CAxx(FIT)GY, the DAI16-4(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 the DAI16-4(FIT)GY: the "module settings area", the "module information area", and the "basic input data area".

Module Settings 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 Settings Area are copied to the Module Information Area. By reading this area, you can verify the current module settings.

Basic Input Data Area

Basic input data is read in this area.

Module Settings Area

A module settings 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 written.

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

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

Table 5.1. Starting address

Address (h)	Area	Item	Size	Access type	Initial value (h)	Initial settings
Starting address +00	Module specific information	Module type (category)	1	R	02	DAI16-4(FIT)GY
Starting address +01		Module type (serial No.)	1	R	03	
Starting address +02		System-reserved (revision No.)	1	R	None	
Starting address +03		Supported functions	1	R	02	Basic Output
Starting address +04		Number of basic input channels	1	R	00	0 channel
Starting address +05		Basic input data size	1	R	00	0 bytes
Starting address +06		Number of basic output channels	1	R	04	4 channels
Starting address +07		Basic output data size	1	R	02	2 byte
Starting address +08		Input channel settings address	1	R	20	20h
Starting address +09		Input channel settings data size	1	R	06	6 bytes
Starting address +0A		Output channel settings address	1	R	50	50h
Starting address +0B		Output channel settings data size	1	R	06	6 bytes
Starting address +0C - Starting address +0F		Reserved	4	R	None	
Starting address +10	Item common to channels	Module startup register	1	R/W	00	
Starting address +11		Error status	1	R	00	
Starting address +12 - Starting address +19		Reserved	8	R	None	
Starting address +1A		Analog output resolution	1	R	10h	16-bit Analog input resolution
Starting address +1B		Analog output range	1	R/W	00	-10V - +10V
Starting address +1C - Starting address +1F		Reserved	4	R	None	
Starting address +20 - Starting address +7F	Channel settings	Reserved	96	R	None	

Module-specific information

- Module type (category)
  The DAI16-4(FIT)GY belongs to the analog module (02h) category.
- Module type (serial No.)
  The DAI16-4(FIT)GY is an analog module with a serial No. 3 (03h).
- Supported functions
  The DAI16-4(FIT)GY supports the basic output function (02h).
  The basic output data takes analog input values.
- Number of basic input channels
  The DAI16-4(FIT)GY does not take basic input data (00h).
- Basic input data size
  The DAI16-4(FIT)GY does not take basic input data (00h).
- Number of basic output channels
  The number of basic output channels for the DAI16-4(FIT)GY is 4 (04h).
  Eight analog output channels are provided.
- Basic output data size
  The basic output data size for the DAI16-4(FIT)GY is 2 (02h) bytes.
  This is a 16-bit data area, of which 16 bits are used by the DAI16-4(FIT)GY.
- Input channel settings address
  The DAI16-4(FIT)GY does not have channel-specific settings.
  This field is provided for compatibility with other device modules.
- Input channel settings data size
  The DAI16-4(FIT)GY does not have channel-specific settings.
  This field is provided for compatibility with other device modules.
- Output channel settings address
  The DAI16-4(FIT)GY does not have channel-specific settings.
  This field is provided for compatibility with other device modules.
- Output channel settings data size
  The DAI16-4(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.
  00h: No operation
  01h: Module startup
- Error status
  The error status bits, which are not reflected in the module settings area, always remain [00h].
  The error status on a module is stored in the module information area.
- Analog input resolution
  The analog input resolution capacity of the DAI16-4(FIT)GY is fixed at 16 bits (10h).
- Analog input range
  This field sets an analog input range.

Table 5.2. Analog output range

Setting value of analog output range (h)	Analog input range
00	-10V - +10V
64	0mA - 20mA

Channel settings

The DAI16-4(FIT)GY does not have channel-specific settings.
This field is provided for compatibility with other device modules.

Module Information Area

The module information area is a 128-byte (80h) area beginning with address 300000h and corresponding to a given Device ID.

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

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

Table 5.3. Module information area

Address (h)	Area	Item	Size	Access type	Initial value (h)
Starting address +00	Module specific information	Module type (category)	1	R	02
Starting address +01		Module type (serial No.)	1	R	03
Starting address +02		System-reserved (revision No.)	1	R	None
Starting address +03		Supported functions	1	R	02
Starting address +04		Number of basic input channels	1	R	00
Starting address +05		Basic input data size	1	R	00
Starting address +06		Number of basic output channels	1	R	04
Starting address +07		Basic output data size	1	R	02
Starting address +08		Input channel settings address	1	R	20
Starting address +09		Input channel settings data size	1	R	06
Starting address +0A		Output channel settings address	1	R	50
Starting address +0B		Output channel settings data size	1	R	06
Starting address +0C - Starting address +0F		Reserved	4	R	None
Starting address +10	Item common to channels	Module startup register	1	R/W	00
Starting address +11		Error status	1	R	00
Starting address +12 - Starting address +19		Reserved	8	R	None
Starting address +1A		Analog output resolution	1	R	10h
Starting address +1B		Analog output range	1	R/W	00
Starting address +1C - Starting address +1F		Reserved	4	R	None
Starting address +20 - Starting address +7F	Channel settings	Reserved	96	R	None

When the module is started, the contents of the module setting area are stored in the module information area, with the exception of the [Module Startup Register] and the [Error Status].

- Module startup register
  This register holds the module operating status.
  Therefore, the fact that the module is shut down simply indicates that the module has not been started.

00h	Module shutdown
01h	Module operating

- 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
21h	Module timeout

  The module timeout status (21h) is an error status that does not usually occur, and indicates that an error occurred during an A/D conversion process. This status will be reset when the module is restarted.

⚠ CAUTION

Writing an analog output value when the module is shut down can generate a module timeout (21h) condition. Writing operations involving analog output values should be performed when the module is running.

Basic Output Data Area

The basic output data area, which is a 128-byte (80h) area beginning with address 305000h, corresponds to a given Device ID.

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

Starting address = 305000h + 80h × (Device ID)

Table 5.4. Basic Output Data Area

Address (h)	Area	Item	Size	Access type
Starting address + 00 - Starting address + 01	CH0	Analog output value	2	R/W
Starting address + 02 - Starting address + 03	CH1	Analog output value	2	R/W
Starting address + 04 - Starting address + 05	CH2	Analog output value	2	R/W
Starting address + 06 - Starting address + 07	CH3	Analog output value	2	R/W
Starting address + 08 - Starting address + 7F	Reserved		120	R

- Analog output value
  Analog output values are stored as Little Endians.

Table 5.5. Analog Output 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

The D/A conversion is performed on a channel-by-channel basis, with a timing that writes analog output values; the results are stored two bytes at a time. For this reason although there is data compatibility between the high and low bytes, there is no synchronization between channels.

Conversion formula:

analog output value = (output voltage (V) + offset) × 2^{16} / span

analog output value = (output current (mA) + offset) × 2^{16} / span

Table 5.6. Conversion Coefficients

Analog output range	Offset	Span
-10V - +10V	10	20
0mA - +20mA	0	20

Table 5.7. Example of a Conversion in the Analog Output Range -10V - +10V

Analog output value (±10V range)	16-bit conversion data
	Offset binary
+9.9997V	FFFFh
:	:
0.00030V	8001h
0.00000V	8000h
-0.00030V	7FFFh
:	:
-10.00000V	0000h

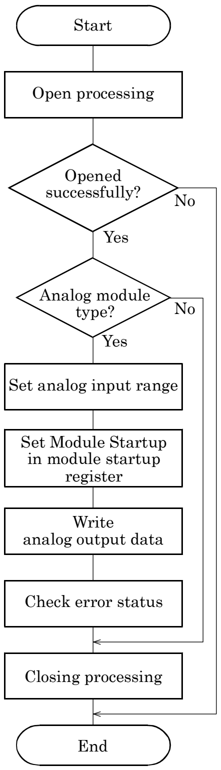
⚠ CAUTION

- For analog output values, valid data is output during the operation of the module. When the module is shut down, the analog input values are undefined.
- An analog output value is 2 bytes per channel. In order to maintain compatibility between the high and low bytes, data should be output in a single WRITE operation.

Examples

Flowchart

Following is an example in which the DAI16-4(FIT)GY is installed at Device ID :0:



Address: Determine category by reading address:301000h. For analog-type module, category is "02h".

Set analog output range.
Address: Analog output range is written to address: 30101Bh. Default is "00h". (Analog output range: 10V to +10V) This value need not be written when the default is used.

Address: Module is started by writing "01h" to address: 301010h. When started, the analog output data becomes valid.

Address: By writing 2 bytes to address: 305000h, the module can output data from CH0. In subsequent operations, the module can write analog output data with 2 bytes at a time, up to CH3.

Address: Error status can be checked by reading address: 300011h.

Figure 5.1. Basic Output Data Area

Sample Program

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

      DEVICE ID:   0
      Channel:     0 to 3ch
      Range:       -10 to 10V
==========================================================================
*/
#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(AIO) */
#define   FIT_AIO_AI_BIT          (0x12)
#define   FIT_AIO_AI_RANGE        (0x13)
#define   FIT_AIO_AI_MODE         (0x14)
#define   FIT_AIO_AO_BIT          (0x1A)
#define   FIT_AIO_AO_RANGE        (0x1B)

/* Information(AIO)*/
#define   FIT_AIO_RANGE_PM10      (0)
#define   FIT_AIO_RANGE_PM5       (1)
#define   FIT_AIO_RANGE_P10       (50)
#define   FIT_AIO_RANGE_P5        (51)
#define   FIT_AIO_RANGE_P20MA     (100)

/* Sample */
#define   FIT_SAMPLE_IP_ADDRESS   "192.168.132.211"
#define   FIT_SAMPLE_PORT         (0x5007)
#define   FIT_SAMPLE_DEVICE_ID    (0)

int main(void)
{
   DWORD  dwIpAddress;
   DWORD  dwVaBase;
   DWORD  dwVaOffset;
```

```
WORD   hHandle;
WORD   wStatus;
BYTE   byCategory;
BYTE   byRange;
BYTE   byModuleStart;
BYTE   byData[0x80];
BYTE   byChCount;
BYTE   byErrorStatus;

/* 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,
             1, &byCategory);
if (wStatus != 0) {
   printf("Error! FIT_Read = %04X(H)\n", wStatus);
   FIT_Close(hHandle);
   return 1;
}
if (byCategory != FIT_PRODUCT_ANALOG) {
   printf("Error! Category = %02X(H)\n", byCategory);
   FIT_Close(hHandle);
   return 1;
}

/* Write 'D/A Range' */
byRange = FIT_AIO_RANGE_PM10;      /* Range:-10 to 10V */
wStatus = FIT_Write(hHandle, dwVaBase + dwVaOffset + FIT_AIO_AO_RANGE,
             1, &byRange);
if (wStatus != 0) {
   printf("Error! FIT_Write = %04X(H)\n", wStatus);
}

/* Write 'Module Start' */
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);
}

/* Write 'D/A Data' */
for (byChCount = 0; byChCount < 4; byChCount++) {
   /* Output Data:10V */
   byData[byChCount * 2 + 1] = 0x0F;
   byData[byChCount * 2] = 0xFF;
   printf("D/A CH%d Data:%02X%02X\n", byChCount, byData[byChCount * 2 + 1],
      byData[byChCount * 2]);
}
dwVaBase = FIT_IO + FIT_IO_OUTPUT;
wStatus = FIT_Write(hHandle, dwVaBase + dwVaOffset, 2 * 4,
             (BYTE *)&byData[0]);
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);
    }
    if (byErrorStatus != 0x00) {
        printf("Error! Error Status = %02X(H)\n", byErrorStatus);
    }

    /* Close */
    FIT_Close(hHandle);

    return 0;
```

6. System Reference

Block Diagram

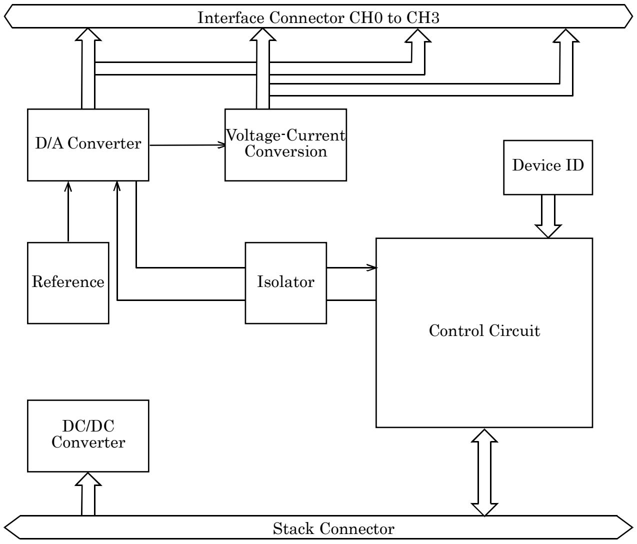


Figure 6.1. Circuit block diagram

Specifications

Table 6.1. Specifications

Item	Specifications	
	Voltage output	Current output
Analog output section		
Output format	Bus-isolated voltage output	Bus-isolated current output
Output range	Bipolar ±10V (Output current ±5mA)	0 - 20mA
Output impedance	Voltage range: 10Ω (Max.)	----
Output channel	4 channels	
Resolution	16 Bits	
Conversion accuracy *1	±18LSB(±0.027% of FSR)	
Settling time	10□sec/ch	20μsec/ch
Data buffer	64-Word	
Interrupt	Either IRQ5 or IRQ7 or IRQ9 *2	
Internal Paser timer	10μsec - 1,073,741,824μsec *2	
Common section		
Internal power consumption	5VDC±5% 500mA(Max.)	
Maximum distance of signal extension	1.5m	
Physical dimensions (mm)	25.2(W) x 64.7(D) x 94.0(H) (exclusive of protrusions)	
Weight (module itself)	100 g	
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)	
Compatible wires	AWG 28 - 16	
Connectors	FRONT-MC 1,5/12-STF-3,81 (mfd by PHOENIX CONTACT Corp.) 3.81 mm-pitch, nominal current : 8A (Max.)	

*1 The Conversion accuracy means an error of approximately 0.1% occurs over the maximum range at 0°C and 50°C ambient temperature.

*2 Available only when the DAI16-4(FIT)GYs connected to the CPU-SBxx(FIT)GY.

⚠ CAUTION

- When connecting one of the modules 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.
- Current output requires an external power supply. With large external power supply fluctuations (ripple), the conversion precision indicated in the specifications may not be attainable. If this problem occurs, please use a low-ripple power supply.
- Depending upon the specific controller module that is used, some of the functions are not supported

Table 6.2. Installation Environment Requirements

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

Physical Dimensions

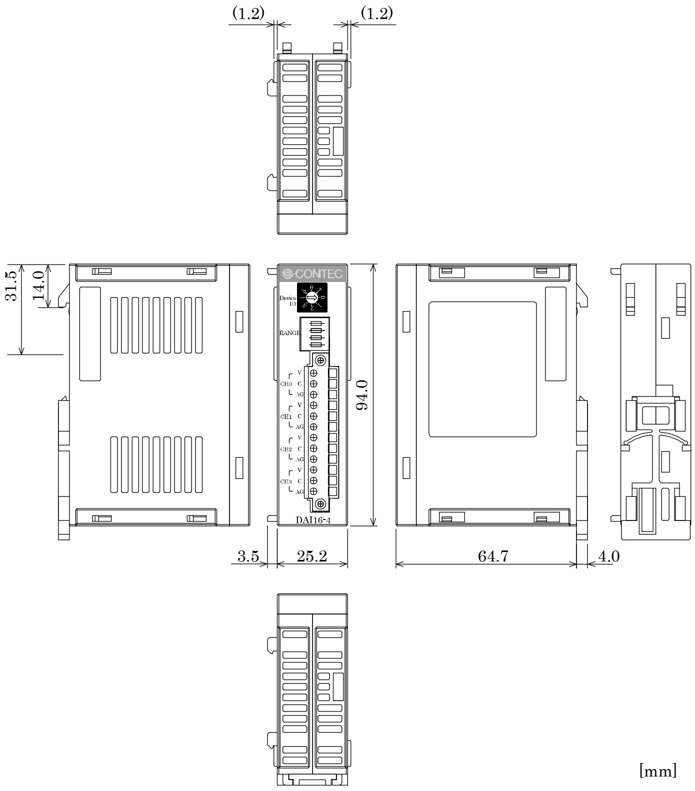


Figure 6.2. Physical dimensions

DAI16-4(FIT)GY

User’s Manual

CONTEC CO., LTD. August 2008 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/

[11262003] Management No. A-40-658

[08222008_rev5] Parts No. LYCS494