

F&eIT Series

Isolated Analog Output Module

DAI12-4(FIT)GY

User's Manual

CONTEC CO.,LTD.

Check Your Package

Thank you for purchasing the CONTEC product.

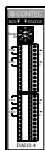
The product consists of the items listed below.

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

Product Configuration List

- Module ...1
- First Step Guide ...1
- CD-ROM [F&eIT Series Setup Disk] *1...1
- Interface connector plugs ...2

*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

Copyright 2001 CONTEC CO., LTD. ALL RIGHTS RESERVED

No part of this document may be copied or reproduced in any form by any means without prior written consent of CONTEC CO., LTD.

CONTEC CO., LTD. makes no commitment to update or keep current the information contained in this document.

The information in this document is subject to change without notice.

All relevant issues have been considered in the preparation of this document. Should you notice an omission or any questionable item in this document, please feel free to notify CONTEC CO., LTD.

Regardless of the foregoing statement, CONTEC assumes no responsibility for any errors that may appear in this document nor for results obtained by the user as a result of using this product.

Trademarks

F&eIT is a registered trademark or trademark of CONTEC CO., LTD. Other company and product names that are referred to in this manual are generally trademarks or registered trade trademark.

Table of Contents

Copyright.....	i
Trademarks.....	i
Product Configuration.....	ii
Table of Contents.....	iii
1. Introduction.....	1
Features.....	1
Functions and control method by controller connected.....	2
Limited One-Year Warranty.....	4
How to Obtain Service.....	4
Liability.....	4
Handling Precautions.....	5
About the Manual.....	6
2. Module Nomenclature and Settings.....	7
Nomenclature of Module Components.....	7
Setting a Device ID.....	8
Setup Method.....	8
LED Indicator.....	8
3. Connecting to an External Device.....	9
Interface Connector.....	9
How to Connect an Interface Connector.....	9
Signal Layout on the Interface Connector.....	10
Connecting an Analog Output Signal.....	11
Connecting a Voltage Output.....	11
Connecting a Current Output.....	13
4. Using the I/O Address Map.....	15
Starting I/O Address.....	15
List of I/O Address Maps.....	16
Specifications Common to F&eIT Products.....	18
Product Information.....	18
Overview of the D/A Conversion Function.....	20
Details on the Analog Output Status.....	28

List of Commands.....	31
Examples.....	32
Software Mode.....	32
Clock Mode (No Interrupts)	34
Clock Mode (with Interrupts)	37
5. Using the Memory Address Map.....	43
Module Settings Area.....	44
Module Information Area.....	47
Basic Output Data Area.....	49
Examples.....	51
6. System Reference	55
Block Diagram	55
Specifications.....	56
External Dimensions.....	58

1. Introduction

Congratulations on your recent purchase of an Insulator Digital Output Module.

The DAI12-4(FIT)GY can control external devices by converting digital data that is output from a, F&eIT-series controller module <CPU-CAxx(FIT)GY, CPU-SBxx(FIT)GY etc> into either voltage or current-based analog signals. The insulation between external signals and the Controller Module permits the use of the Controller Module without compromising the communications features of the latter.

By reading this manual carefully, please build a system by creating applications programs, setting the switches, and connecting the module to external devices.

Features

- The output range is common to different channels, and can be selected from five output ranges: 0 - 10V, 0 - 5V, $\pm 10V$, $\pm 5V$, and 0 - 20mA.
- A rotary switch that is provided allows you to set device IDs to help you keep track of device numbers.
- The system incorporates a screwless connector plug that allows you to easily attach and detach wires without using any special tools.
- Similar to other F&eIT series products, the system, in the module itself, incorporates a 35mm 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 DAI12-4(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
Isolated Analog Output Module for USB	: DAI12-4 (USB)GY
I/O Controller Module with USB	: CPU-CA10(USB)GY

The functions and control of the DAI12-4(FIT)GY vary with the controller to which the DAI12-4(FIT)GY is connected.

Functions available with each controller connected

	CPU-SBxx(FIT)GY	CPU-CAxx(FIT)GY	SVR-MMF2(FIT)	SVR-MMF(FIT)GY	DAI12-4(USB)GY	CPU-CA10(USB)GY
Software input range setting	○	○			○	○
D/A conversion with software command	○	○			○	○
Continuous D/A conversion using the internal pacer clock	○ *2		*1	*1	○ *3	
Multi-channel simultaneous output function	○				○	○
Interrupt function	○					
Device ID setting range	0 - 7	0 - 7	0 - 7	0 - 7	1 - 3	0 - 7

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

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

*3 Using the sampling timer built in the DAI12-4(USB)GY

The setting range is from 1000 to 1,073,741,000 microseconds.

Control method by controller connected

	CPU-SBxx(FIT)GY	CPU-CApp(FIT)GY	SVR-MMF2(FIT)	SVR-MMF(FIT)GY	DAI12-4(USB)GY	CPU-CA10(USB)GY
Control using the I/O address map	○					
Control using the memory address map		○				
Control via the Windows driver *	FIT Protocol		○			
	API-CAP(W32)		○			
	API-SBP(W32)	○				
	API-USBP(WDM)				○	○
Control over the web			○	○		

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

Control using the I/O address map

When connected to the CPU-SBxx(FIT)GY, the DAI12-4(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-CApp(FIT)GY, the DAI12-4(FIT)GY can be accessed from the host computer over the network.

The DAI12-4(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.

Limited One-Year Warranty

CONTEC F&eIT 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 product 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.

Handling Precautions

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%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.

About the Manual

This manual consists of the following chapters :

- | | |
|-----------|--|
| Chapter 1 | Introduction |
| Chapter 2 | Module Nomenclature and Settings
Explains the nomenclature of the components of the Module and their operations. |
| Chapter 3 | Connecting to an External Device
Explains interface connectors and provides tips on how to make a connection to an external device. |
| Chapter 4 | Using the I/O Address Map
Explains I/O port bit assignments and the definitions of the bits when the Module is used as a CPU-SBxx(FIT)GY module. |
| Chapter 5 | Using the Memory Address Map
Explains the module setting area, the information area, and the basic input data area when the Module is used as a CPU-CAXx(FIT)GY module. |
| Chapter 6 | System Reference
Explains module specifications and circuit block diagrams. |

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.

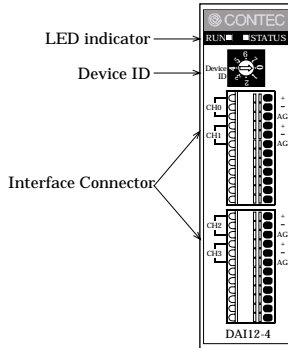


Figure 2.1. Names of Module Components

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.

To connect the DAI12-4(FIT)GY to the DAI12-4(USB)GY, assign a device ID between 1 and 3.

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

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.

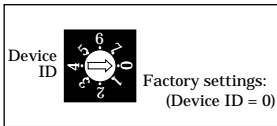


Figure 2.2. Setting a Device ID

LED Indicator

RUN: Indicates that the DAI12-4(FIT)GY can be controlled from the Controller Module (green).

STATUS: This light comes on when a D/A conversion error occurs (red).

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. When wiring the Module, strip off approximately 7 - 8 mm of the covering for the cable, and insert the bare wire by pressing the orange button on the connector plug. Releasing the orange button after the wire is inserted to fix the cable. Compatible wires are AWG 28 - 20.

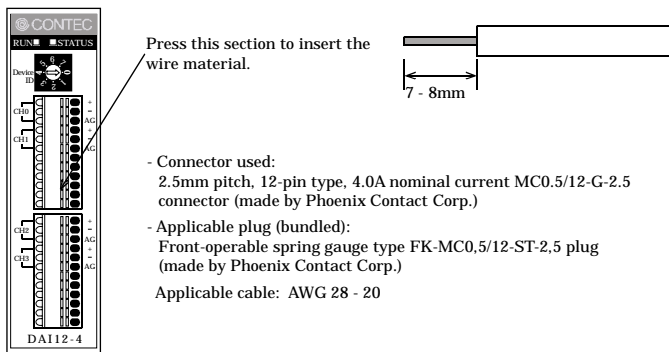


Figure 3.1. Connecting an Interface Connector and Connectors That Can Be Used

Note!

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 12pin (1 group) connector that is provided on the Module face.

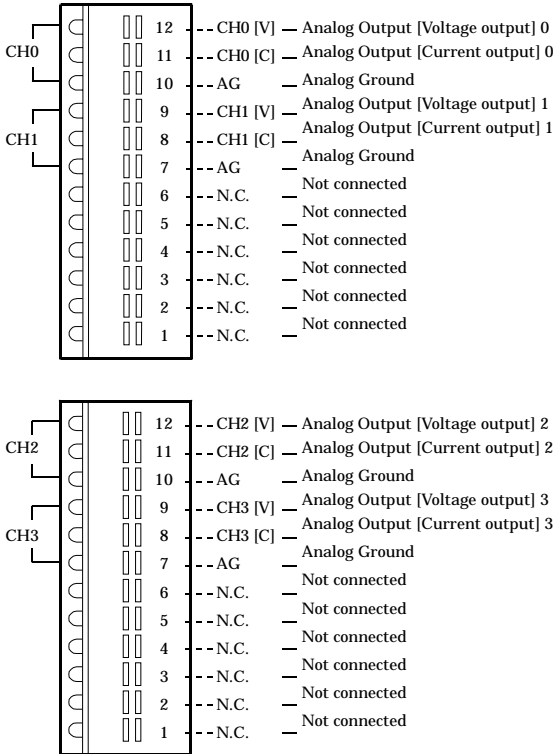


Figure 3.2. Signal Layout on the Interface Connector

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.

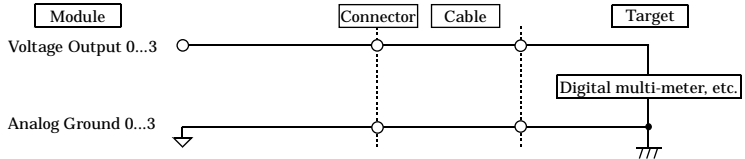


Figure 3.3. Connecting a Voltage Output (Flat Cable)

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.

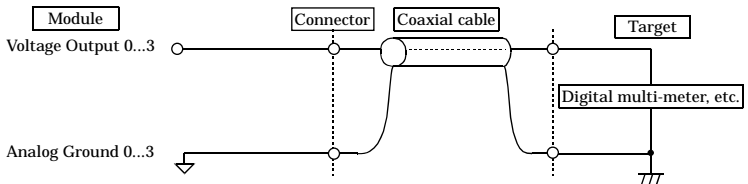


Figure 3.4. Connecting a Voltage Output (Coaxial Cable)

Notes!

- *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 $\pm 5\text{mA}$. 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 (10 - 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 500Ω , 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 .

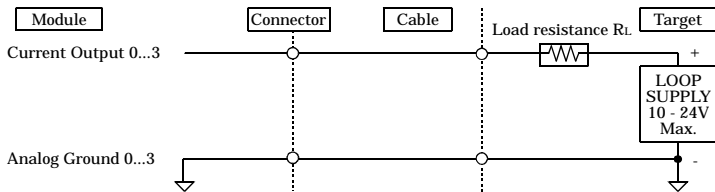


Figure 3.5. Connecting to a Floating Load (Flat Cable)

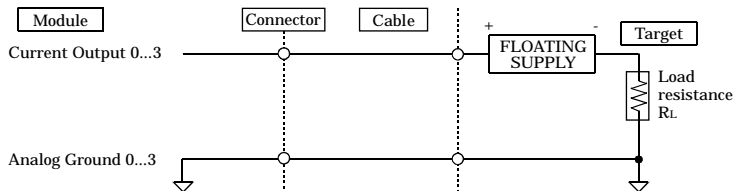


Figure 3.6. Connecting to a Fixed Load (Flat Cable)

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 R_L of the external device by using the core wire and the shield braid of the coaxial cable.

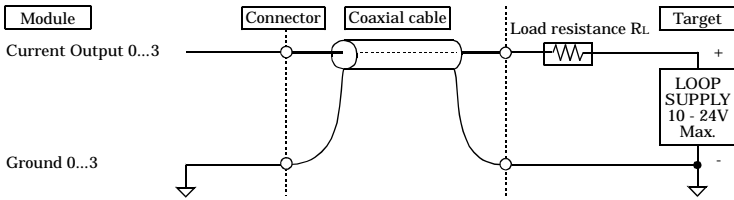


Figure 3.7. Connecting to a Floating Load (Coaxial Cable)

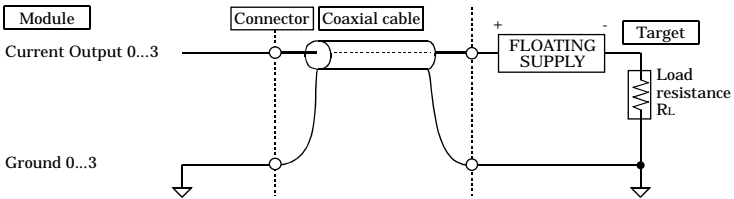


Figure 3.8. Connecting to a Fixed Load (Coaxial Cable)

Notes!

- When the power is turned on or the module is reset, the current output signal will be 0 mA.
- 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.

4. Using the I/O Address Map

Starting I/O Address

When connected to a CPU-SBxx(FIT)GY, the DAI12-4(FIT)GY can directly receive I/O commands from the controller module. Depending on how the Device ID is set, the occupied I/O address takes the values indicated below.

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

If the Device ID is set to 0, 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

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

		Input Port							
Starting I/O address	D07	D06	D05	D04	D03	D02	D01	D00	
input +0 (+00h)	Product Category				Revision Data				
	0	0	1	0	Revision Data3	Revision Data2	Revision Data1	Revision Data0	
+1 (+01h)	Product ID Number								
	0	0	0	0	0	0	0	1	
+2 (+02h)	Interrupt Status								
	Enable	Status	0	0	0	IRQ9	IRQ7	IRQ5	
+3 (+03h)	(Not available)								
⋮									
+21 (+15h)									
⋮									
+22 (+16h)	Analog Output Status								
	0	0	Pacer Clock Error	Pacer Clock Input	0	0	End of Conversion	Data Set Busy	
+23 (+17h)	(Not available)								
⋮									
+31 (+1Fh)									
⋮									

Figure 4.1. Input Port

Output Port

Starting I/O Address	D07	D06	D05	D04	D03	D02	D01	D00
+0 (+00h)	(Not allowed)							
⋮								
+1 (+01h)								
+2 (+02h)	Interrupt Data							
	Enable	N/A	N/A	N/A	N/A	IRQ9 Data	IRQ7 Data	IRQ5 Data
+3 (+03h)	(Not allowed)							
⋮								
+15 (+0Fh)								
+16 (+10h)	Analog Output Data (Lower)							
	Conversion Data 07	Conversion Data 06	Conversion Data 05	Conversion Data 04	Conversion Data 03	Conversion Data 02	Conversion Data 01	Conversion Data 00 (LSB)
+17 (+11h)	Analog Output Data (Upper)							
	N/A	N/A	N/A	N/A	Conversion Data 11 (MSB)	Conversion Data 10	Conversion Data 09	Conversion Data 08
+18 (+12h)	Channel Data							
	All Channel	End Channel	N/A	N/A	N/A	Channel Data 02	Channel Data 01	Channel Data 00
+19 (+13h)	(Not allowed)							
⋮								
+21 (+15h)								
+22 (+16h)	Status Reset							
	N/A	N/A	Pacer Clock Error	Pacer Clock Input	N/A	N/A	End of Conversion	N/A
+23 (+17h)	(Not allowed)							
⋮								
+24 (+18h)								
+24 (+18h)	Command Data							
	Command Data 07	Command Data 06	Command Data 05	Command Data 04	Command Data 03	Command Data 02	Command Data 01	Command Data 00
+25 (+19h)	(Not allowed)							
⋮								
+27 (+1Bh)								
+28 (+1Ch)	Setting Data 0							
	Setting Data 07	Setting Data 06	Setting Data 05	Setting Data 04	Setting Data 03	Setting Data 02	Setting Data 01	Setting Data 00
+29 (+1Dh)	Setting Data 1							
	Setting Data 15	Setting Data 14	Setting Data 13	Setting Data 12	Setting Data 11	Setting Data 10	Setting Data 09	Setting Data 08
+30 (+1Eh)	Setting Data 2							
	Setting Data 23	Setting Data 22	Setting Data 21	Setting Data 20	Setting Data 19	Setting Data 18	Setting Data 17	Setting Data 16
+31 (+1Fh)	Setting Data 3							
	Setting Data 31	Setting Data 30	Setting Data 29	Setting Data 28	Setting Data 27	Setting Data 26	Setting Data 25	Setting Data 24

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

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
input +0 (00h)	Product Category				Revision Data			
	0	0	1	0	Revision Data3	Revision Data2	Revision Data1	Revision Data0
+1 (01h)	Product ID Number							
	0	0	0	0	0	0	0	1

Figure 4.3. Product Information

- Revision Data [D3 - D0]:
This is product update information, subject to change without notice, that is controlled by CONTEC.
- Product Category [D7 - D4]:
This is a module function classification code. For the DAI12-4(FIT)GY, the code is "2h".

Table 4.2. Product Category

Code	Function
0	Extention 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 DAI12-4(FIT)GY,), the product ID is "1h".

Following are examples of the initialization that is performed in high-level languages.

Microsoft C

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

* ADR denotes the starting I/O address for the DAI12-4(FIT)GY.

Microsoft QBASIC

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

* ADR denotes the starting I/O address for the DAI12-4(FIT)GY.

Interrupt Status

This is a 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.

Starting I/O Address	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 in high-level languages:

Microsoft C

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

Microsoft QBASIC

```
IrqStatus = INP ( ADR+2 )
```

Setting an Interrupt Level

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
output	Interrupt Data							
+2 (02h)	Enable	0	0	0	0	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 in high-level languages:

The interrupt level to be used is assigned to IRQ5.

Microsoft C

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

Microsoft QBASIC

```
OUT ADR+2, &H81
```


Overview of the D/A Conversion Function

The analog output function produces specified 12-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.

The following chart illustrates the basic conversion process.

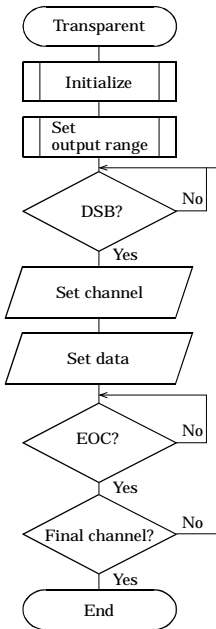


Figure 4.6.
Transparent Output Mode

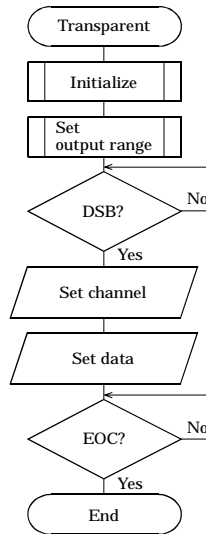


Figure 4.7.
Synchronous Output Mode

Initialization

This step initializes the analog output function.

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

The initialization control port assumes the following state:

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
output	Command							
+24 (18h)	0	0	0	0	0	0	0	0

Figure 4.8. Initialization

Following are examples where the initialization is effected in high-level languages:

Microsoft C

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

Microsoft QBASIC

```
OUT ADR+24, &H0
```

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.

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
output	Command							
+24 (18h)	0	0	0	0	0	0	1	0
output	Setting D/A Conversion Conditions							
+28 (1Ch)	0	0	0	0	0	Output Mode	Pacer Clock	Conversion Mode

Figure 4.9. 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:

Microsoft C

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

Microsoft QBASIC

```
OUT ADR+24, &H2  
OUT 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 12-bit resolution level.

The output range setup control port assumes the following state:

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
	Command							
+24 (18h)	0	0	0	0	0	0	1	1
output	D7	D6	D5	D4	D3	D2	D1	D0
	Range settings							
+28 (1Ch)	Range Data7	Range Data6	Range Data5	Range Data4	Range Data3	Range Data2	Range Data1	Range Data0

Figure 4.10. Setting an Output Range

Table 4.3. Output Range and Setup Data

Range	Output range
00h	±10V
01h	±5V
02 - 7Fh	Not decided
80h	0 - 10V
81h	0 - 5V
82 - BFh	Undecided
C0h	0 - 20mA
C1h and above	Not decided

Following are examples in which an output range is specified in high-level languages. The example involves setting an output range of 0 - 10V.

Microsoft C

```
outp( ADR+24, 0x3 );
outp( ADR+28, 0x80 );
```

Microsoft QBASIC

```
OUT ADR+24, &H3
OUT ADR+28, &H80
```

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 - 1,073,741,824,000nsec (approximately 17 minutes 54 seconds), which corresponds to 39 - 4,294,967,295 setup data.

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

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

The pacer clock is specified in units of nanoseconds.

The value of the pacer clock must satisfy the following expression:

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

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:

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
output	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 Data 07	Timer Data 06	Timer Data 05	Timer Data 04	Timer Data 03	Timer Data 02	Timer Data 01	Timer Data 00
	Timer Data 1							
+29 (1Dh)	Timer Data 15	Timer Data 14	Timer Data 13	Timer Data 12	Timer Data 11	Timer Data 10	Timer Data 09	Timer Data 08
	Timer Data 2							
+30 (1Eh)	Timer Data 23	Timer Data 22	Timer Data 21	Timer Data 20	Timer Data 19	Timer Data 18	Timer Data 17	Timer Data 16
	Timer Data 3							
+31 (1Fh)	Timer Data 31	Timer Data 30	Timer Data 29	Timer Data 28	Timer Data 27	Timer Data 26	Timer Data 25	Timer Data 24

Figure 4.11. Setting an Internal Pacer Clock

Following are examples in which an internal pacer clock is specified in high-level languages:

Microsoft C

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

Microsoft QBASIC

```
OUT ADR+24, &H4
OUT ADR+28, ClockData0
OUT ADR+29, ClockData1
OUT ADR+30, ClockData2
OUT 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)

Starting I/O Address output	D7	D6	D5	D4	D3	D2	D1	D0
	Channel Data							
+18 (12h)	All Channel	End Channel	N/A	N/A	N/A	Channel Data 2	Channel Data 1	Channel Data 0

Figure 4.12. 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 in high-level languages:

Microsoft C

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

Microsoft QBASIC

```
OUT ADR+18, &H80
OUT ADR+16, LowerData
OUT 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:

Microsoft C

```
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 );
```

Microsoft QBASIC

```
OUT ADR+24, &H2
OUT ADR+28, &H5
OUT ADR+18, &H00
OUT ADR+16, LowerData0
OUT ADR+17, UpperData0
OUT ADR+18, &H41
OUT ADR+16, LowerData1
OUT 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 range	Offset	Span
-10V - +10V	10	20
-5V - +5V	5	10
0V - +10V	0	10
0V - +5V	0	5
0mA - +20mA	0	20

Table 4.5. Example of Conversion Data over a $\pm 10V$ Range

Analog output value (+/-10V range)	12-bit conversion data
	Offset binary
+9.995V	0FFF h
:	:
0.005V	0801 h
0.000V	0800 h
-0.005V	07FF h
:	:
-10.000V	0000 h

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
output	Channel Data							
+18 (12h)	All Channel	End Channel	N/A	N/A	N/A	Channel Data 02	Channel Data 01	Channel Data 00
output	Analog output Data (Lower)							
+16 (10h)	Conversion Data7	Conversion Data6	Conversion Data5	Conversion Data4	Conversion Data3	Conversion Data2	Conversion Data1	Conversion Data0(LSB)
+17 (11h)	Analog output Data (Upper)							
	0	0	0	0	Conversion Data11(MSB)	Conversion Data10	Conversion Data9	Conversion Data8
input	Analog Output Status							
+22 (16h)	0	0	Pacer Clock Error	Pacer Clock Input	0	0	End of Conversion	Data Set Busy

Figure 4.13. 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 in high-level languages.

Following are examples in which conversion data is output in high-level languages:

Microsoft C

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

Microsoft QBASIC

```
WHILE ( INP ( ( ADR+22 ) AND 1 ) = 1 ) : WEND
OUT ADR+18, &H0
OUT ADR+16, LowerData
OUT ADR+17, UpperData 2 ) : WEND
```


Details on the Analog Output Status

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

Starting I/O Address	D7	D6	D5	D4	D3	D2	D1	D0
input	Analog Output Status							
+22 (16h)	0	0	Pacer Clock Error	Pacer Clock Input	0	0	End of Conversion	Data Set Busy
output	Status Reset							
+22 (16h)	0	0	Pacer Clock Error	Pacer Clock Input	0	0	End of Conversion	0

Figure 4.14. Analog Output Status

Dataset Busy (DSB) [D0]:

When this bit is [0], the dataset is available for the output of D/A conversion data.
 When this bit is [1], the dataset is writing data to a D/A converter register in the module and is unavailable for writing any other D/A conversion data.

End of Conversion (EOC) [D2]:

When [1] is set to this status bit, the data that has been written to the D/A converter is converted into analog signals. This bit is cleared when [1] is set to the End of Conversion Status of the analog output status reset port.
 The end of conversion does not include analog signal setup time. *

Pacer Clock Input Status [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 [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. *

* 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

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
End of Conversion	When the writing of conversion data to the D/A converter is finished
Pacer Clock Input	When a pacer clock is input
Pacer Clock Error	When the pacer clock error status is 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:

Starting I/O Address output	D7	D6	D5	D4	D3	D2	D1	D0
	Command							
+24 (18h)	0	0	0	0	0	0	0	1
output	D7	D6	D5	D4	D3	D2	D1	D0
	Interrupt Source							
+28 (1Ch)	1	1	Pacer Clock Error	Pacer Clock Input	1	1	End of Conversion	1

Figure 4.15. 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 in high-level languages:

Microsoft C

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

Microsoft QBASIC

```
OUT ADR+24, &H1
OUT ADR+28, InterruptFactor
```

List of Commands

Following is a list of DAI12-4(FIT)GY commands that are issued to "Output port +24":

Table 4.7. List of Commands

No.	HEX	Function	Data size
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

Examples

Software Mode

Flowchart

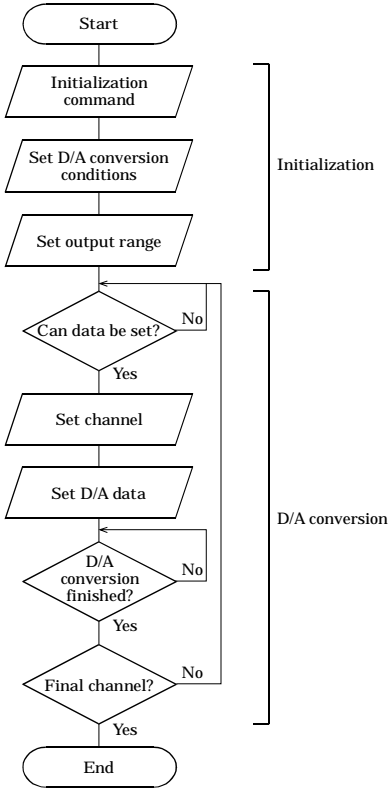


Figure 4.16. Software Mode

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 = 0x0c;
    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/4096.0f-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

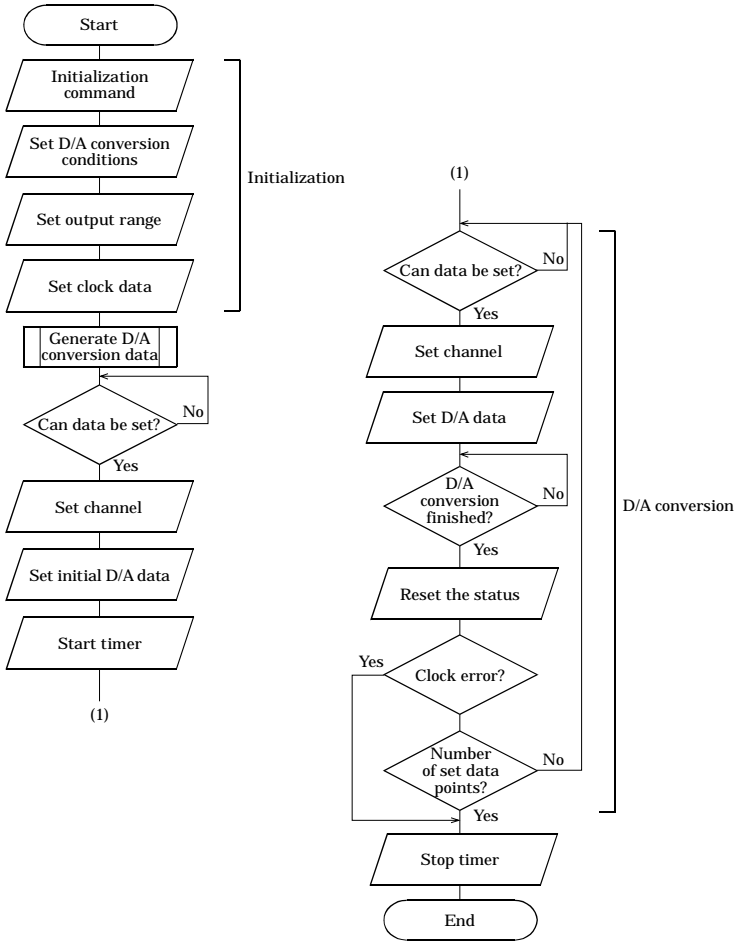


Figure 4.17. Clock Mode (No Interrupts)

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, 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)(4096.0f/(NUM-1)*i);
        if( VDAT > 0xffff ) VDAT = 0xffff;
        UpperData[i] = (unsigned char)( ( VDAT & 0xff00 ) >> 8 );
        LowerData[i] = (unsigned char)( VDAT & 0xff );
        Volt[i] = (float)VDAT*20.0f/4096.0f-10.0f;
    }
}

```



```

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] ); /* Set Lower Data */
    outp( ADR+0x11, UpperData[i] ); /* 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 ( j == 3 ) outp( ADR+0x12, 0x40+j );
            /* Set Last Channel */
        else outp( ADR+0x12, j );  /* Set Channel */
        outp( ADR+0x10, LowerData[i+1] ); /* Set Lower Data */
        outp( ADR+0x11, UpperData[i+1] ); /* 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

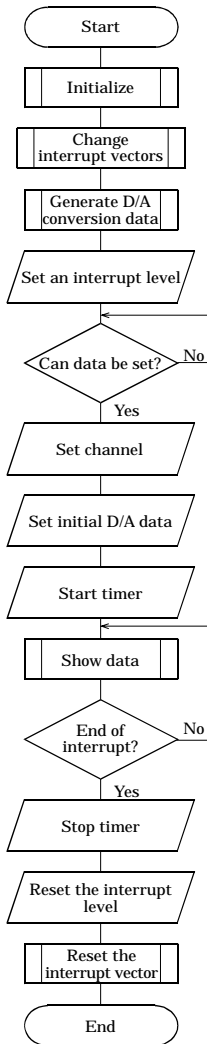
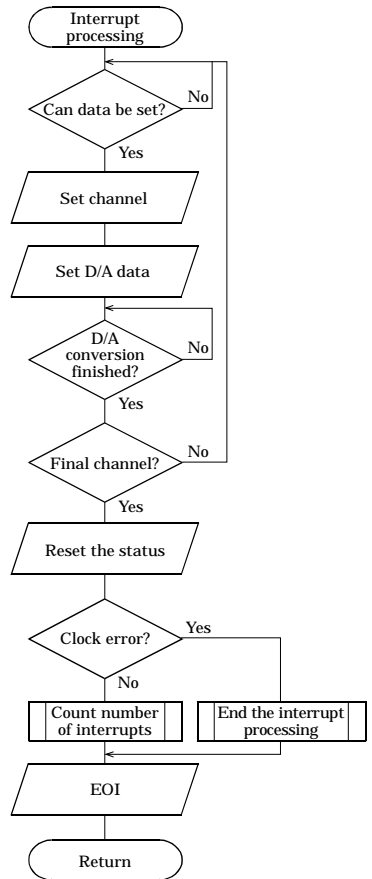


Figure 4.18. Clock Mode (with Interrupts)



Clock Mode (with Interrupts)

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 };
/* interrupt 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 );
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, 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( 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)(4096.0f/(NUM-1)*j);
            if( VDAT > 0xffff ) VDAT = 0xffff;
            DaData [i][j] = VDAT;
        }
    }
}

```

```

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

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/4096.0f-
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 DAI12-4(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 Module 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 DAI12-4(FIT)GY: the "module settings area", the "module information area", and the "basic output 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 Output Data Area

Basic output data is written into this area.

Module Settings Area

A module setting 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:

$$\text{Starting address} = 301000\text{h} + 80\text{h} \times (\text{Device ID})$$

Table 5.1. Module Settings Area

Address (h)	Area	Item	Size	Access type	Initial value (h)	Initial settings
Starting address +00	Module specific information	Module type (category)	1	R	02	DAI12-4(FIT)GY
Starting address +01		Module type (serial No.)	1	R	01	
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	0Ch	12-bit Analog output 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 DAI12-4(FIT)GY belongs to the analog module (02h) category.
- Module type (serial No.)
The DAI12-4(FIT)GY is an analog module with a serial No. 1 (01h).
- Supported functions
The DAI12-4(FIT)GY supports the basic output function (02h).
The basic output data takes analog output values.
- Number of basic input channels
The DAI12-4(FIT)GY does not have basic input data (00h).
- Basic input data size
The DAI12-4(FIT)GY does not have basic input data (00h).
- Number of basic output channels
The DAI12-4(FIT)GY has four basic output channels (04h).
Four analog output channels are provided.
- Basic output data size
The DAI12-4(FIT)GY has a basic output data size of 2 bytes (02h).
Of the 16-bit data area, the DAI12-4(FIT)GY uses 12 bits.
- Input channel settings address
The DAI12-4(FIT)GY does not have channel-specific settings. This field is provided for compatibility with other device modules.
- Input channel settings data size
The DAI12-4(FIT)GY does not have channel-specific settings. This field is provided for compatibility with other device modules.
- Output channel settings address
The DAI12-4(FIT)GY does not have channel-specific settings. This field is provided for compatibility with other device modules.
- Output channel settings data size
The DAI12-4(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 output resolution
The analog output resolution capacity of the DAI12-4(FIT)GY is fixed at 12 bits (0Ch).
- Analog output range
This field sets an analog output range.

Table 5.2. Setting an Analog Output Range

Setting value of analog output range (h)	Analog output range
00	-10V - +10V
01	-5V - +5V
32	0V - +10V
33	0V - +5V
64	0mA - +20mA

Channel settings

The DAI12-4(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:

$$\text{Starting address} = 300000\text{h} + 80\text{h} \times (\text{Device ID})$$

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	01
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	0Ch
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.

Note!

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:

$$\text{Starting address} = 305000\text{h} + 80\text{h} \times (\text{Device ID})$$

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	0	0	0	0	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:

$$\text{analog output value} = (\text{output voltage (V)} + \text{offset}) \times 2^{12} / \text{span}$$

$$\text{analog output value} = (\text{output current (mA)} + \text{offset}) \times 2^{12} / \text{span}$$

Table 5.6. Conversion Coefficients

Analog output range	Offset	Span
-10V - +10V	10	20
-5V - +5V	5	10
0V - +10V	0	10
0V - +5V	0	5
0mA - +20mA	0	20

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

Output voltage (V)	Analog output value (h)
+9.995	0FFF
:	:
0.005	0801
0.000	0800
-0.005	7FF
:	:
-10.000	0000

Notes!

- 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 DAI12-4(FIT)GY is installed at Device ID: 0.

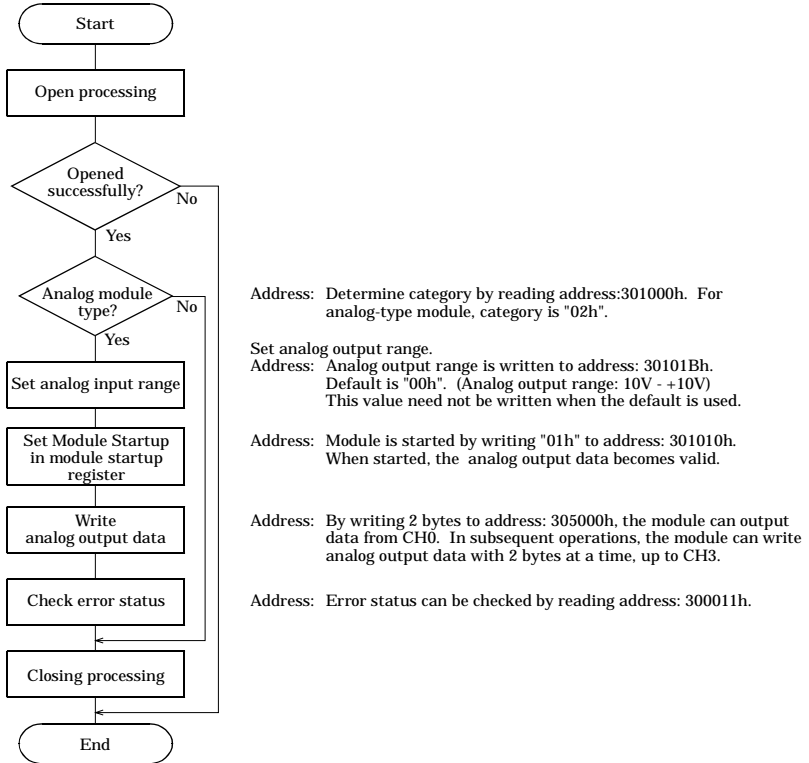


Figure 5.1. Basic Output Data Area

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

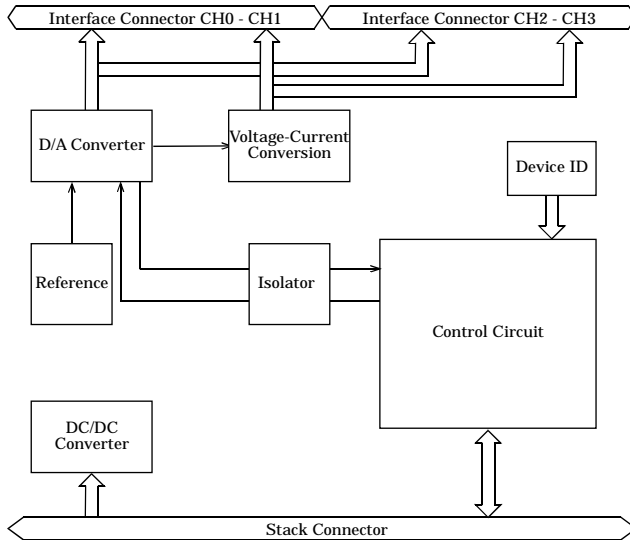


Figure 6.1. Circuit Block Diagram

Specifications

Table 6.1. Specifications

Item	Specification
Analog output section	
Output format	Bus-isolated voltage/current output
Output range	Voltage: Bipolar $\pm 10V$, $\pm 5V$ Unipolar 0 - 10V, 0 - 5V (Output voltage $\pm 5mA$) Current: 0 - 20mA
Output impedance	Voltage range: 10 Ω (Max.)
Output channel	4 channels
Resolution	12 Bits
Conversion accuracy *1	Voltage range: $\pm 3LSB$, Current range: $\pm 5LSB$
Settling time	Voltage range: 10 μsec /ch, Current range: 20 μsec /ch
Interrupt	Either IRQ5 or IRQ7 or IRQ9 *2
Internal sampling timer	10 μsec - 1,073,741,824 μsec *1 *3
Common section	
Internal power consumption	5VDC $\pm 5\%$ 400mA(Max.)
Maximum distance of signal extension	1.5m
External dimensions (mm)	25.2(W) x 64.7(D) x 94.0(H) (exclusive of protrusions)
Weight (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 - 20
Applicable plug	FK-MC 0,5/12-ST-2,5 (made by PHOENIX CONTACT Corp.)

*1 When the environment temperature is near 0°C or 50°C, the non-linearity error may become larger.

*2 Available only when the DAI12-4(FIT)GY is connected to the CPU-SBxx(FIT)GY.

*3 If connected to a DAI12-4(USB)GY, uses the internal timer in the DAI12-4(USB)GY.

The setting range is from 1000 to 1,073,741,000 μsec .

Notes!

- 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
Line-Noise resistance	Line-noise	AC line/2kV, Signal line/1kV (IEC1000-4-4Level 3, EN61000-4-4Level 3)
	Static electricity resistance	Contact discharge/4kV (IEC1000-4-2Level 2, EN61000-4-2Level 2) Atmospheric discharge/8kV (IEC1000-4-2Level 3, EN61000-4-2Level 3)
Vibration resistance	Sweep resistance	10 - 57Hz/semi-amplitude 0.15mm, 57 - 150Hz/2.0G 80minutes each in X, Y, and Z directions (JIS C0040-compliant, IEC68-2-6-compliant)
Impact resistance		15G, half-sine shock for 11ms in X, Y, and Z directions (JIS C004-compliant, IEC68-2-27-compliant)

External Dimensions

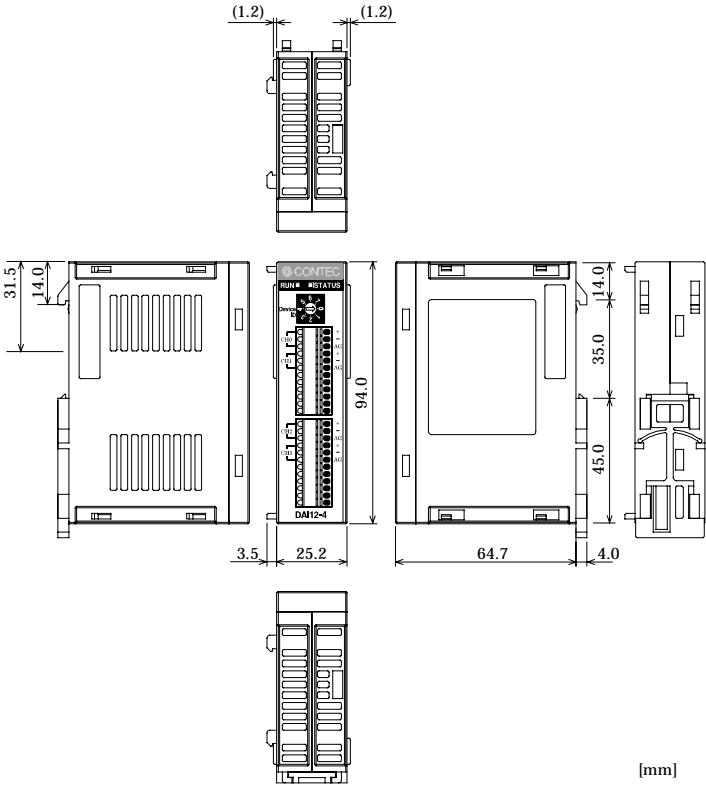


Figure 6.2. External dimensions

DAI12-4(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/>

No part of this document may be copied or reproduced in any form by any means without prior written consent of CONTEC CO., LTD. [11152006]

[08062001]

Management No. A-40-608

[1115006_rev7]

Parts No. LZU3862