Fieldbus

Hardware Developer's Guide for Fieldbus Round Card Users

August 1998 Edition
Part Number 322124A-01

Contents

About This Manual

Chapter 1
Introduction

Chapter 2
Hardware Installation

Chapter 3
Interfacing to Your Fieldbus Round Card

Chapter 4
Migrating from a Development Round Card to an OEM Round Card

Appendix A
Data Link Configuration Section Format

Appendix B
System Management Configuration Section Format

Appendix C
Dimensions

Appendix D
Specifications

Glossary

Figures

Tables

About This Manual

This manual contains instructions for installing, interfacing to, and programming the National Instruments Fieldbus Round Card.

The Round Card software is intended for use with Windows 3.*x*, Windows 95, or Windows NT. This manual assumes that you are already familiar with the Windows operating system you are using.

Organization of This Manual

This manual is organized as follows:

- Chapter 1, *Introduction*, provides a brief description of the Fieldbus Round Card hardware and available software.
- Chapter 2, *Hardware Installation*, contains instructions to help you install your Fieldbus Round Card.
- Chapter 3, *Interfacing to Your Fieldbus Round Card*, describes how to connect the Fieldbus Round Card to any external electronics, and how to develop your Field Device application to interface to the NI-FBUS Function Block Shell.
- Chapter 4, *Migrating from a Development Round Card to an OEM Round Card*, explains how to migrate from the development version of the Fieldbus Round Card to the OEM version of the Fieldbus Round Card.
- Appendix A, *Data Link Configuration Section Format*, explains how to structure the Data Link Configuration section of your Device Configuration `.ini` file.
- Appendix B, *System Management Configuration Section Format*, explains how to structure the System Management Configuration section of your Device Configuration `.ini` file.
- Appendix C, *Dimensions*, describes the dimensions of the Fieldbus Round Card and the GPIO Daughter Card.
- Appendix D, *Specifications*, describes the characteristics of the Fieldbus Round Card, the SP Daughter Card, and the GPIO Daughter Card.

Conventions Used in This Manual

The following conventions are used in this manual:

<>	Angle brackets enclose the name of a key on the keyboard—for example, <shift>. Angle brackets containing numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, DBIO<3..0>.
☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.
	This icon to the left of bold italicized text denotes a warning, which advises you of precautions to take to avoid being electrically shocked.
bold	Bold text denotes the names of menus, menu items, parameters, dialog box, dialog box buttons or options, icons, windows, Windows 95 tabs, or LEDs.
bold italic	Bold italic text denotes a note or warning.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept. This font also denotes text from which you supply the appropriate word or value, as in Windows 3.*x*.
italic monospace	Italic text in this font denotes that you must supply the appropriate words or values in the place of these items.
monospace	Text in this font denotes text or characters that you should literally enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and for statements and comments taken from programs.

Related Documentation

The following documents contain information that you may find helpful as you read this manual:

- *Fieldbus Foundation Specification*, which includes the following items:
  - *Fieldbus Foundation System Management Services*
  - *Function Block Application Process, Part 1*
  - *Function Block Application Process, Part 2*
- *NI-FBUS Communications Manager User Manual* from National Instruments, which explains how to use the interactive Fieldbus dialog system with your Fieldbus Round Card
- *NI-FBUS Configurator User Manual* from National Instruments, which explains how to use the NI-FBUS Configurator to configure your Fieldbus network
- *NI-FBUS Monitor User Manual* from National Instruments, which explains how to use the interactive NI-FBUS Monitor utility with your Fieldbus Round Card

Chapter 1

Introduction

This chapter provides a brief description of the Fieldbus Round Card hardware and available software.

Hardware Overview

The Fieldbus Round Card is a stand-alone card that allows you to interface to a network that complies with the Fieldbus Foundation H1 specification. The Fieldbus Round Card is powered from the Fieldbus. The Fieldbus Round Card uses the Motorola MC68331 embedded processor and a programmable 128 KB X 16 Flash to run the Stack Interface Library, Function Block Shell, and user applications. A 128 KB X 16 SRAM device on the card provides volatile memory.

The Fieldbus Round Card also has a 8 KB X 8 Serial ROM (SROM) that serves as non-volatile memory. This non-volatile memory is used to store the Fieldbus MIB parameters and block parameters.

The Fieldbus Round Card provides an interface to the Fieldbus. To run real-world applications, the Fieldbus Round Card can be connected to a Daughter Card. On the Daughter Card, you can implement DIO, A/D, D/A functionality. The Fieldbus Round Card provides the necessary interface signals via a 2 X 21 position header (W1) that can be connected to the Daughter Card. Refer to Figure 3-3, *Pin Assignments for the Daughter Card Interface Header, W1*, in Chapter 3, *Interfacing to Your Fieldbus Round Card*, for pin assignments of the W1 header.

The General Purpose I/O Daughter Card, henceforth referred to as the GPIO Daughter Card, can be used to connect devices such as valves, sensors, and actuators. The card has four digital inputs and four digital outputs. It has two analog input channels that can be used either in the voltage mode or the current mode. The GPIO Daughter Card also has one voltage analog output and one 4–20 mA analog output channel.

The Serial Programming Daughter Card, henceforth referred to as the SP Daughter Card, is primarily used for development purposes. The SP Daughter Card has a Background Debug Mode (BDM) port that you can

use for debugging the application during development. A serial port is also present on the SP Daughter Card. This port can be used either as a regular serial port or as a port over which the user can program the Flash on the Fieldbus Round Card with the user-level application. Refer to Figure 3-1, *Pin Assignments for the Fieldbus Round Card BDM Connector, W2*, in Chapter 3, *Interfacing to Your Fieldbus Round Card*.

When developing your application, you will use the development version of the Fieldbus Round Card. When you have finished developing and debugging your application, you need to migrate to the OEM version of the Round Card. Chapter 4, *Migrating from a Development Round Card to an OEM Round Card*, explains how to migrate from the development version to the OEM version of the Round Card.

Software Overview

The software available for the Fieldbus Round Card includes the NI-FBUS Function Block Shell, which is an Application Programmer's Interface (API) designed to simplify Fieldbus device development by providing a high-level interface to the Fieldbus communications stack. In addition, a serial driver API is provided to allow you to make use of the Round Card's interrupt-driven serial port. The serial driver supports standard HART commands, as well as generic access to the serial port to allow any other serial protocols. A linkable library version of the Fieldbus protocol stack is also available. Link your Function Block application with the Function Block Shell and the protocol stack before downloading it to your Fieldbus Round Card.

Chapter 2

Hardware Installation

This chapter contains instructions to help you install your Fieldbus Round Card.

Warning: ***Electrostatic discharge can damage several components on your Fieldbus Round Card. To avoid such damage in handling the board, touch the antistatic plastic package to a metal part of your computer chassis before removing the board from the package.***

Perform the following steps to install the Fieldbus Round Card:

1. Connect the Fieldbus cable to terminal J1 on the Fieldbus Round Card. The positive (+) end of the cable should be connected to terminal F+ and the negative (–) end of the cable should be connected to terminal F– (see Figure 2-1). Ensure that the Fieldbus cable is properly terminated. Do not apply power to the Fieldbus Round Card.
2. Connect the SP Daughter Card to the Fieldbus Round Card by mating the W1 and W2 connectors on the Fieldbus Round Card to the J2 and J4 connectors on the SP Daughter Card. (See Figure 2-2 for the locations of the connectors on the SP Daughter Card.)
3. If you want to use the serial port, connect a serial cable from the serial port of the host PC to the J1 connector on the SP Daughter Card. This link can be used to download the application to the Flash on the Fieldbus Round Card.
4. You can debug the application in RAM by using the BDM port of the processor. Refer to Chapter 3, *Interfacing to Your Fieldbus Round Card*, for a description of the BDM mode. Connect the 3 V ICD cable that came with the software debugger from the parallel port of the host PC to connector W3 on the SP Daughter Card. Confirm that pin 1 of the 3 V ICD cable matches pin 1 on the W3 connector of the SP Daughter Card.
5. Connect a 12–24 DC power supply to connector J1 on the GPIO Daughter Card. The positive terminal of the power supply should be connected to the F+ terminal of connector J1 on the GPIO Daughter Card.

6. Connect the GPIO Daughter Card to the SP Daughter Card by mating the J2 and J4 connectors on the SP Daughter Card to the J2 and J3 connectors on the GPIO Daughter Card. (See Figure 2-3 for the locations of the connectors on the GPIO Daughter Card.)
7. Switch on both the Fieldbus and the GPIO Daughter Card power supplies.
8. Press the reset button on the Round Card, S1, to reset the processor.

☞ **Note:** ***Remember to order a 3 V ICD cable when you buy the software debugger. One suggested vendor for the ICD cable is: P&E Microcomputer Systems, Woburn, MA Web site: http://www.pemicro.com Low Voltage In-Circuit Debugger Cable (CPU16/CPU32).***

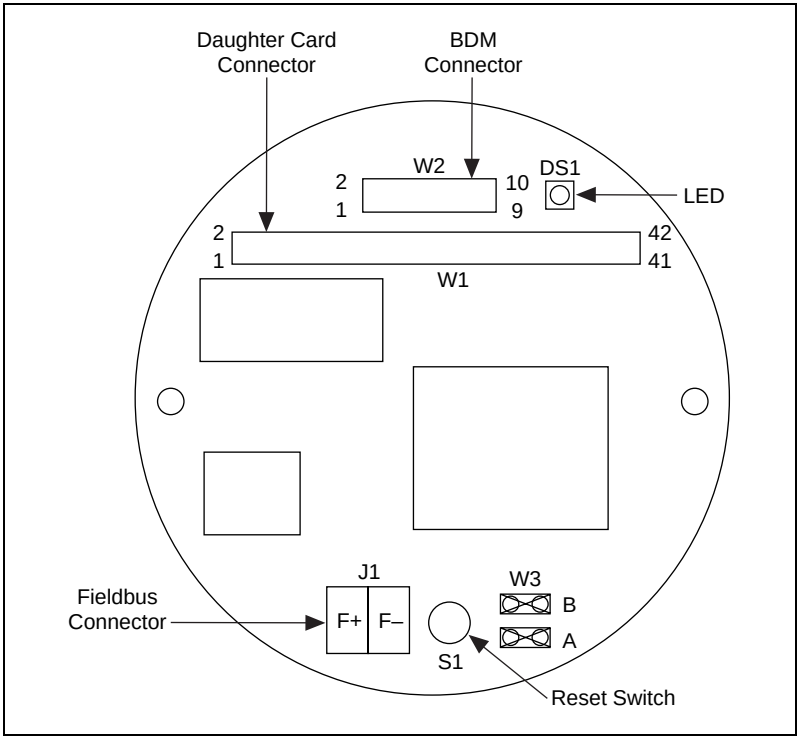

Figure 2-1. Top Side of Fieldbus Round Card

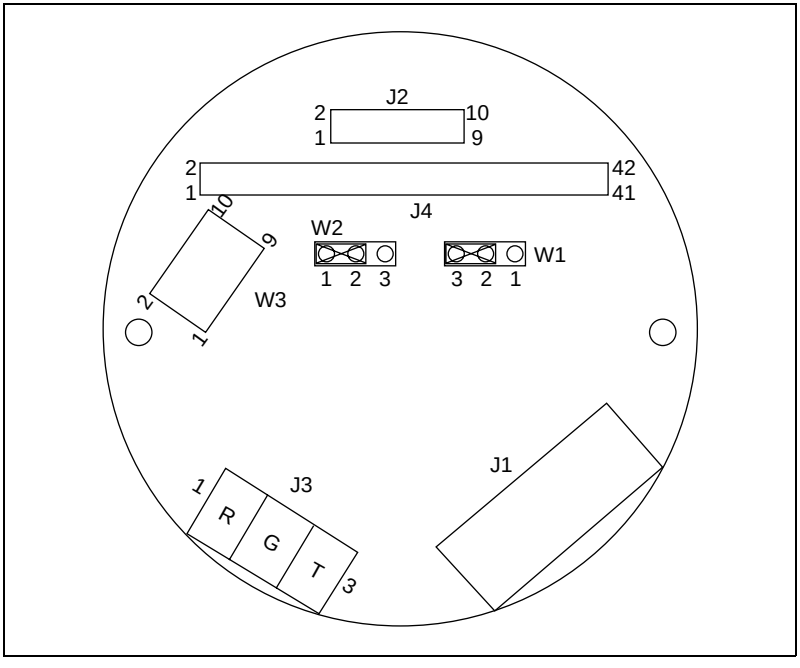

Figure 2-2. Top Side of SP Daughter Card

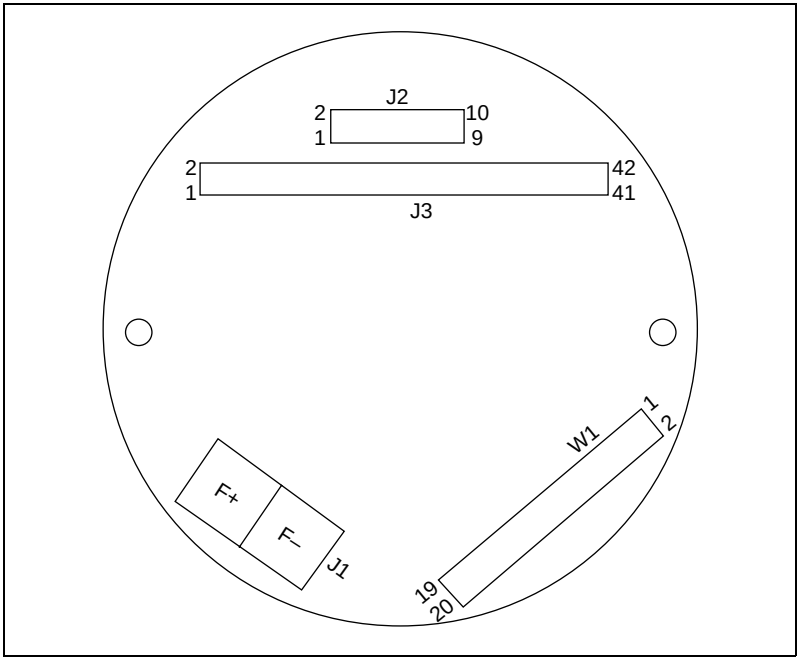

Figure 2-3. Top Side of GPIO Daughter Card

Your hardware is now installed.

Interfacing to Your Fieldbus Round Card

Chapter 3

This chapter describes how to connect the Fieldbus Round Card to any external electronics, and how to develop your Field Device application to interface to the NI-FBUS Function Block Shell.

Interfacing to the Hardware

Description of the Fieldbus Round Card

The Fieldbus Round Card provides an interface to the Fieldbus. To run real-world applications, the Fieldbus Round Card can be connected to a Daughter Card. On the Daughter Card you can implement DIO, A/D, D/A functionality. The Fieldbus Round Card provides the necessary interface signals via a 2 X 21 position header (W1) that can be connected to the Daughter Card. Refer to Figure 3-3 for the pin assignments of the W1 header on the Fieldbus Round Card.

The Fieldbus Round Card is also capable of providing a 5 V at 4 mA power supply to power your electronics.

The Fieldbus Round Card can be placed in Simulate mode by removing jumper A of W3. In Simulate mode you can run a software simulation of your function blocks without doing any actual input/output. Under normal operating conditions, place jumper A of the W3 header on the Fieldbus Round Card. The software does not currently support this feature.

The MC68331 processor has a Background Debug Mode (BDM) port, as shown in Figure 2-1, *Top Side of Fieldbus Round Card*, in Chapter 2, *Hardware Installation*. The BDM port can be used for debugging your application in RAM. The BDM port has a serial interface that can be connected to the parallel port of the host computer running the software debugger. Over the BDM port you can monitor registers, read or write to memory, and single step through your code. The stack library has been developed and debugged with SDS SingleStep software, version 7.11.

To use your software debugger to debug your application, connect the SP Daughter Card to the Fieldbus Round Card. Connect the 3 V ICD cable from the parallel port of the host PC to the W3 connector on the SP Daughter Card. Connector W3 on the SP Daughter Card is connected to the BDM port on the processor.

Figure 3-1 shows the pin assignments for the SP Daughter Card BDM connector, W2 on the Fieldbus Round Card. These signals are directly routed to connector J2 on the SP Daughter Card.

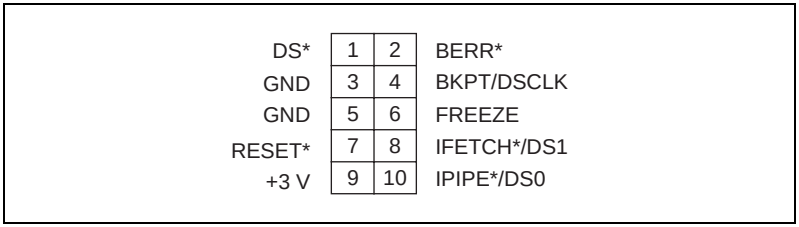

Figure 3-1. Pin Assignments for the Fieldbus Round Card BDM Connector, W2

Table 3-1 lists the signal descriptions of the BDM connector, W2, on the Fieldbus Round Card.

Table 3-1. Pinout of Fieldbus Round Card BDM Header, W2

Signal Name	Description
+ 3 V	3 V power supply output
BERR*	Bus error
BKPT/DSCLK	Hardware breakpoint signal or BDM clock source
DS*	Data strobe signal from processor
GND	Ground
FREEZE	Indicates processor has entered background mode
IFETCH*/DS1	Indicates instruction pipeline activity or BDM data input signal

Table 3-1. Pinout of Fieldbus Round Card BDM Header, W2 (Continued)

Signal Name	Description
IPIPE*/DS0	Indicates instruction pipeline activity or BDM data output signal
RESET*	Processor reset

The development version of the Fieldbus Round Card operates under one of two operating modes: PROG mode and RUN mode. When power is applied to the Fieldbus Round Card, the processor reads the position of jumper B of W3 to check the mode (see Figure 2-1, *Top Side of Fieldbus Round Card*, in Chapter 2, *Hardware Installation*, for the location of W3).

To select PROG mode, place jumper B of W3 on the Fieldbus Round Card. Connect the SP Daughter Card to the Fieldbus Round Card. Attach a standard serial port cable from the COM port of the host computer to connector J1 on the SP Daughter Card. When the card is powered up in PROG mode, the processor downloads the user application into the Flash, from the COM port of the host computer to the serial port of the processor.

To select RUN mode, remove jumper B of W3 from the Fieldbus Round Card. When the card is powered up in RUN mode, the processor begins executing the application that is loaded in the Flash.

The serial port of the processor can alternatively used to run the serial HART protocol or any other proprietary protocol.

Description of the GPIO Daughter Card

The GPIO Daughter Card can be used to interface to devices such as valves, sensors, and actuators. The GPIO Daughter Card implements DIO, A/D, and D/A functionality. The GPIO Daughter Card is powered from an external 12–24 VDC power supply. The Fieldbus Round Card can be connected to the GPIO Daughter Card via the daughter card interface connector, W1, on the Fieldbus Round Card. The GPIO Daughter Card has four 5 V compatible digital input ports and four 5 V compatible digital output ports. The digital input/output ports are fixed and cannot be reconfigured. The analog input interface consists of two 12-bit, 500 samples/s, voltage (0–10 V) or current (4–20 mA) input channels. You can use the two analog input channels either as voltage

or current channels. The GPIO Daughter Card also has one 12-bit, 0–10 V, analog voltage output channel, and one 12-bit, 4–20 mA analog current output channel. The analog output current channel can be connected to a 4–20 mA current loop. These I/O channels are available for use on the 2x10 position connector, W1, on the GPIO Daughter Card.

Figure 3-2 shows the pin assignments of the W1 connector on the GPIO Daughter Card.

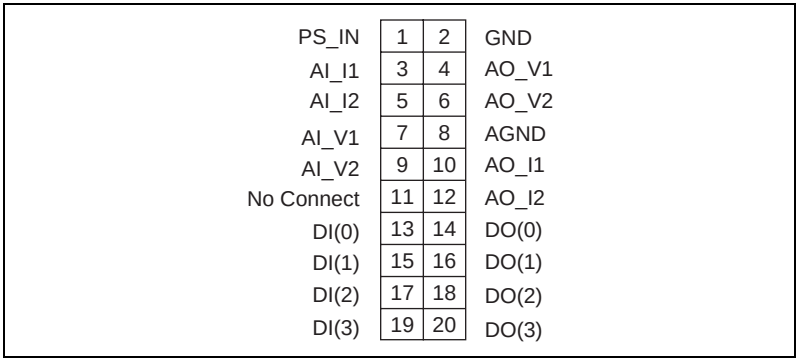

Figure 3-2. Pin Assignments for the W1 Connector on the GPIO Daughter Card

Table 3-2 lists the signal descriptions of the GPIO Daughter Card interface connector, W1 on the GPIO Daughter Card.

Table 3-2. Signal Descriptions of GPIO Daughter Card Connector, W1

Signal Name	Description
AGND	Analog ground on the GPIO Daughter Card
AI_I1	4–20 mA analog input current channel
AI_I2	4–20 mA analog input current channel
AI_V1	0–10 V analog input voltage channel
AI_V2	0–10 V analog input voltage channel
AO_I1	4–20 mA analog output current channel
AO_I2	4–20 mA analog output current channel

Table 3-2. Signal Descriptions of GPIO Daughter Card Connector, W1 (Continued)

Signal Name	Description
AO_V1	0–10 V analog output voltage channel
AO_V2	0–10 V analog output voltage channel
DI<3..0>	5 V–compatible digital input ports
DO<3..0>	5 V–compatible digital output ports
GND	Digital ground on the GPIO Daughter Card
PS_IN	12–24 VDC power supply to power the GPIO Daughter Card

Daughter Card Interface Signal Connections

The Fieldbus Round Card can easily be interfaced to your Daughter Card. Alternatively, you can design your own Daughter Card to meet your own design requirements. The Fieldbus Round Card to Daughter Card interface is over a 2 X 21-position header. Figure 3-3 shows the pin assignments for the W1 Daughter Card connector on the Fieldbus Round Card.

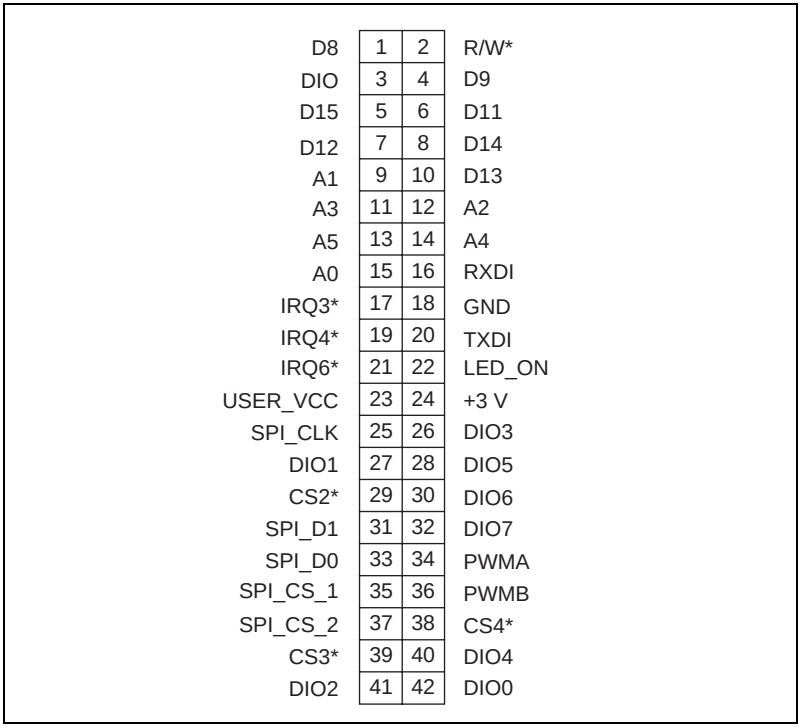

Figure 3-3. Pin Assignments for the Daughter Card Interface Header, W1

Table 3-3 lists the signal descriptions of the Daughter Card interface connector, W1 on the Fieldbus Round Card.

Table 3-3. Signal Descriptions of Daughter Card Connector, W1

Signal Name	Description
3 V	3 V power supply output
A<5..0>	Address lines driven by the MC68331 processor
CS<4..2>*	Programmable chip-select lines driven by the processor
D<15..8>	Upper data bus lines of the processor
DIO<7..0>	General-purpose digital I/O lines connected to the PORT GP of the processor

Table 3-3. Signal Descriptions of Daughter Card Connector, W1 (Continued)

Signal Name	Description
GND	Ground
IRQ<6,4,3>*	IRQ lines that provide IRQ signals to the processor
LED_ON	Output to drive an LED for monitoring status of the Fieldbus Round Card
PWMA	Output from the pulse width modulator on the processor
PWMB	Output from the pulse width modulator on the processor
R/W*	R/W* line driven by the processor
RXD1	3 V-compatible serial port line
SPI_CLK	Clock output from the SPI port of the processor
SPI_D0	Serial output from the SPI slave device
SPI_D1	Serial input to the SPI slave device
SPI_CS_1	SPI chip select driven by the processor to select the slave SPI devices
SPI_CS_2	SPI chip select driven by the processor to select the slave SPI devices
TXD1	3 V-compatible serial port line
USER_VCC	5 V at 4 mA power supply output to power user electronics

Using the Software

Overview

Most of the code that will be running on your Fieldbus Round Card has already been written for you. It includes the Fieldbus protocol stack, Function Block Shell, and Serial Driver, which are provided in the form of a linkable library. This library provides Fieldbus communications and an API (the Function Block Shell) designed to isolate your application as much as possible from the specifics of the Fieldbus.

In addition, the library contains an API to allow HART or direct serial access to the Round Card's serial port. This API facilitates communications with a HART or serial transducer external to the Round Card.

Developing Your Round Card Application

Complete the following steps to develop your application, after you have installed the hardware and software:

1. Write a device template for your device.
2. Convert the device template to C code using the Device Code Generator.
3. Write your Function Block Callbacks, algorithms, and device interface code.
4. Write your `userStart` function to register your callbacks.
5. Specify the system clock speed and RAM size.
6. Define and install any interrupt handlers you may need.
7. Write your Device Configuration.
8. Convert your Device Configuration to C code using the Configuration Code Generator.
9. Compile, link, and download your program on Flash for installation on the Fieldbus Round Card.
10. Burn your Flash on the Fieldbus Round Card.

These steps are documented in more detail in the following sections.

Writing Device Templates

You must create a device template to describe the network-visible structure of your device and the parameters of your function blocks to the Function Block Shell. The device template is an ASCII file that is divided into various sections containing numerical and string parameters. Sample device templates are in the `\samples` subdirectory of your installation directory for devices containing function blocks of the standard types.

The simplest way to create your device template is to modify a copy of one of the sample device templates using a text editor such as MS-DOS Edit or Windows Notepad. Choose the sample device template that most closely matches your device. For example, if the main function of your device is analog input, start with the AI Device Template. If you want your device to contain multiple function blocks, you need to paste several `BLOCK` sections from the sample files into your device template file.

The templates contain information about the device identification, the physical and function blocks in the device, and the device parameters. Your final device templates are converted to C code using the Device Code Generator, described in the next section.

Converting a Device Template to C Code

Before you compile your Function Block device, you must convert the device template to C code using the Device Code Generator. The Device Code Generator resides in the `\utils` subdirectory of your installation directory. The Device Code Generator takes the following command line arguments:

`codegen` *`deviceTemplate outputFile symbolFile`*

where *`deviceTemplate`* is the name of your device template file, and *`outputFile`* is the name that you want to call the output file. Make sure *`outputFile`* ends in `.c`.

If there are syntax errors in your device template, `codegen` tells you where they are. When you have corrected all syntax errors, *`outputFile`* is created. *`outputFile`* contains code representing the structure of your device. `outputFile` is used again when you compile and link your device application.

symbolFile contains a reference to DD information for the parameters defined in your DD and template file. If you are using only the standard function blocks and their parameters, *symbolFile* is the standard symbols file, `nifb.sym`, located in the `\samples` subdirectory. If you have defined your own blocks or parameters, *symbolFile* must be set to the output of the DD tokenizer.

Writing Function Block Callbacks

The callback functions that you must develop are responsible for the following main functions:

- Executing your function block algorithm
- Handling alarm confirmations and acknowledgments (if you are using alarms)

The Function Block Shell calls your execution callback whenever your Block is scheduled to execute. This callback performs whatever algorithm you want your function block to perform. The other callbacks, which are called after your device sends an alarm, allow you to perform device-specific processing upon alarm confirmations (notifications that the alarm was received) and alarm acknowledgments (notifications that a user has seen the alarm).

Writing userStart and Registering Callbacks

Your `userStart` function is called by the stack during startup. It is your chance to perform your own initialization tasks. `userStart` also initializes the Function Block Shell and registers your callbacks with the Function Block Shell.

Specifying the System Clock Speed and RAM Size

You must specify the system clock speed (4 MHz or 8 MHz) and RAM size (128 KB or 256 KB) you are using so that the protocol stack is set up correctly. To do so, you must declare and define two global variables called `SystemClockSpeed` and `RamSize`, both `unsigned char`, in your source code. Failing to declare these global variables may result in link time errors.

You must initialize these variables when you declare them. The valid values for these variables are as follows:

For `SystemClockSpeed`, choose `1` for a 4 MHz clock or `2` for an 8 MHz clock.

For `RamSize`, choose `1` for 128 KB or `2` for 256 KB.

If these variables are uninitialized or are initialized with any other values, a clock speed of 4 MHz and RAM size of 128 KB are assumed. Refer to Chapter 4, *Migrating from a Development Round Card to an OEM Round Card*, for more information on clock speed and RAM size.

Defining and Installing Interrupt Handlers

The Round Card provides interrupt pins at priority levels 3, 4, and 6, which are available to any interrupt sources on the Daughter Card. You may also use any internal modules of the microcontroller to generate interrupts that are necessary for you. (Notice, however, that you must not use the output compare OC1 of the general-purpose timer module to generate any interrupts. The protocol stack uses it for its timer handler routines.)

You must install interrupt handlers for the interrupts handled by you. To install an interrupt handler, `myinthandler`, at vector number *`vnum`*, your application must make the following stack function call:

`niInstallIntHandler(`*`vnum`*`,(unit32)myinthandler)`

where the value of `vnum` is one of the following: 27, 28, 30, or any number between 85 and 255, inclusive.

`niInstallInstHandler` returns zero upon success and nonzero upon failure. Notice that vector numbers 27, 28, and 30 are level 3, level 4, and level 6 interrupt autovectors, respectively. A vector number between 85 and 255 is for an interrupt whose priority level is encoded in the interrupt source.

Generating Your Device Configuration

The initial configuration of your device includes the configuration of items such as the starting node address, device identification, and, optionally, the function block schedules. You must specify the parameters in the standard Windows `.ini` file format in a configuration `.ini` file. Sample configuration files are included in the `\samples` subdirectory of your installation directory. You might want to start with one of these files and edit it according to your needs. The entries in the data link configuration and system management configuration sections of your configuration `.ini` file are described in Appendix A, *Data Link Configuration Section Format* and Appendix B, *System Management Configuration Section Format*.

After you have generated your configuration `.ini` file, you must run the Configuration Generator to create a C source file that contains your configuration. This step is described in the next section, *Converting a Device Template to C Code*.

Converting Your Device Configuration to C Code

To convert your Device Configuration to a compilable and linkable `.c` file, you must use the Configuration Generator utility. The Configuration Generator requires the following syntax:

`cfggen` *`iniFile cFile`*

where *`iniFile`* is the name of your configuration file, and *`cFile`* is the name of the C source file for the output C code. If your `.ini` file contains errors, the Configuration Generator halts and informs you where the errors are located. Otherwise, it creates a `.c` source file, which you use in the final step to create your binary file.

Compiling, Linking, and Downloading Your Program

During the development stage of your application, you should run your application in RAM by downloading the linker output file from your debugging environment. After the application is fully debugged, generate a binary image that can be downloaded to the Flash using the `niBurn` utility.

☞ **Note:** ***The*** `nistack.lib` ***library file, which your code must link to, was created with SDS CrossCode C/C++ Compiler, version 7.11. Use either the same compiler or a compatible compiler to ensure that your code works correctly with*** `nistack.lib`***. National Instruments recommends that you use SDS CrossCode C/C++ Compiler, version 7.11, for maximum compatibility with our library file.***

The first step in the final stage is compilation. The following files must be successfully compiled to `.o` format:

- The `.c` file generated by the Device Code Generator.
- The `.c` file generated by the Configuration Generator.
- Your own `.c` file that contains your `userStart` function, your callbacks, declaration of the global variables `SystemClockSpeed` and `RamSize`, and any interrupt handlers.

You should compile the files to `.o` format using compiler options to meet the following conditions:

- You must allow single-byte enumeration where possible. For the SDS CrossCode C/C++ compiler, version 7.11, this option is `-s enum=1`.
- You must set the size of the pointer to data and the size of the pointer to function to 4 each. For the SDS CrossCode C/C++ compiler, version 7.11, these options are `-s ptrd=4` and `-s ptrf=4`, respectively.

After you have compiled the files, you must link them with the National Instruments Round Card library, `nistack.lib`. This file contains the communications stack and Function Block Shell. Make sure to do the following when linking to `nistack.lib`:

- Specify case-sensitive public and external symbols. For the SDS CrossCode linker, do *not* use the `-u` option.
- Give the correct linker specification file (using the `-f` option) that maps different regions of your object files to the correct places in RAM and ROM. See the `\samples` directory for example linker specification (`.spc`) files for the SDS CrossCode linker.
- Do not use the `-l` option to specify `nistack.lib` to the linker, because `nistack.lib` is actually a relocatable object file. Instead, specify `nistack.lib` as you would any other object file when linking.

The following step is necessary only when you are ready to download your application to the Flash.

After successfully linking your application with `nistack.lib`, extract object bytes from the linker output file and convert them into a binary image that is ready to be downloaded into the memory of the target processor. To complete this step, you need a downloader utility such as SDS CrossCode's `down`.

The memory locations of the Flash, RAM, and Frontier-1 are shown in Table 3-4.

Table 3-4. Memory Map of Fieldbus Round Card

Component	Size	Memory Window (hex)
SRAM (DEV)	128 KB × 16	C0000–FFFFF
SRAM (OEM)	64 KB × 16	C0000–DFFFF
Flash	128 KB × 16	00000–3FFFF
Frontier-1	2 KB × 8	80000–807FF*
GPIO	2 KB × 8	90000–907FF

* The Frontier-1 needs a 32-byte window for accessing its registers, but the minimum memory window size that can be allocated is 2 KB, so the Frontier-1 registers are aliased within this 2 KB memory space.

Burn Your Flash

After you have run the downloader, you are ready to burn your Flash and test your program.

The Fieldbus Round Card provides a method for you to burn the Flash without an external PROM burner device. This method involves the use of the `niBurn` utility, an RS-232 port on the host, and the serial port on the SP Daughter Card. See the *Interfacing to the Hardware* section, earlier in this chapter, for the connections of the serial port on the SP Daughter Card.

The user program must be located at physical address 0x4000, because this is the address the `niBurn` utility jumps to when the board is reset in RUN mode, or when `niBurn` has successfully downloaded the user program into the Flash in PROG mode.

Complete the following steps to burn the Flash with the `niBurn` utility.

1. Power up the Round Card in PROG Mode (see the *Description of the Fieldbus Round Card* section, earlier in this chapter, for more information about PROG Mode).
2. Confirm that pins W2(1)–W2(2) and W1(3)–W1(2) are jumpered to select RS-232 level signals.
3. Make sure that the user program is located at physical address 0x4000, because this is the address the `niBurn` utility jumps to when the board is reset in RUN mode, or when `niBurn` has

successfully downloaded the user program into the Flash in PROG mode.

4. Wait for the DS1 LED to turn on.
5. Launch `niBurn`. The `niBurn` utility is located in the **NI Fieldbus Round Card** program group on the host. When you launch the `niBurn` utility, it prompts you for the name of your binary file, and the COM port to use.

   A standard serial cable leading to the SP Daughter Card must be attached to the COM port you specified. The `niBurn` utility contacts the Fieldbus Round Card and downloads your program into the Flash on the Fieldbus Round Card. The utility informs you when it has completed.
6. After the application has downloaded, remove jumper B of W3 before resetting the Fieldbus Round Card.

Migrating from a Development Round Card to an OEM Round Card

Chapter 4

This chapter explains how to migrate from the development version of the Fieldbus Round Card to the OEM version of the Fieldbus Round Card.

When the development and debugging phase of your application, described in the previous chapters, is complete, you are ready to move your application over to the OEM version of the Fieldbus Round Card. This move involves programming the Flash with your application during manufacturing.

Differences Between the Development and OEM Round Cards

Table 4-1 shows the differences between the development Round Card and the OEM Round Card.

Table 4-1. Features of the Development Round Card and OEM Round Card

Feature	Development Round Card	OEM Round Card
Operating Frequency	4 or 8 MHz	4 or 8 MHz
Operation At 4 MHz At 8 MHz	 9–32 V at 20 mA 9–32 V at 20 mA	 9–32 V at 10 mA 9–32 V at 20 mA
RAM	256 KB	128 KB
Flash	256 KB	256 KB
Serial ROM	8 KB	8 KB

The development Round Card has 256 KB RAM so that the entire application can be moved into the RAM and debugged.

System Clock Speed Selection

You can run the Fieldbus Round Card at two speeds: 4 MHz and 8 MHz. The reason to select one clock speed over the other is the power draw from the Fieldbus. A Fieldbus Round Card running at 8 MHz needs more current from the Fieldbus.

The Fieldbus driver circuitry on the Fieldbus Round Card governs the current draw from the Fieldbus. Refer to Table 4-1 for the current draw and speed specifications of the development and OEM versions of the Fieldbus Round Card.

The Fieldbus driver circuitry on the development Round Card is designed to draw current needed for the 8 MHz system clock. Selecting a slower clock speed on the development Round Card does not lower the current draw. However, when you are ready to migrate to the OEM Round Card, depending on your power budget, you need to order the OEM Round Card at the correct system clock speed.

Before you compile your application, you need to set the `SystemClockSpeed` variable to the speed at which you want to run the Round Card. When the Round Card initially powers up, it starts running at 4 MHz. During the execution of your program, your desired operating frequency is set. If the `SystemClockSpeed` variable is set to 8 MHz, the Round Card operates at 8 MHz. If the `SystemClockSpeed` variable is set to 4 MHz, the Round Card operates at 4 MHz. Refer to the *Specifying the System Clock Speed and RAM Size* section of Chapter 3, *Interfacing to Your Fieldbus Round Card*, for information on setting the `SystemClockSpeed` variable.

Data Link Configuration Section Format

Appendix
A

This appendix explains how to structure the Data Link Configuration section of your Device Configuration `.ini` file.

The Data Link Configuration section of your Windows `.ini` Device Configuration file must be converted to C code and linked with your application before the Round Card can communicate on the Fieldbus network. The code generated by running this file through the Configuration Generator automatically configures your board. When a parameter is changed over the Fieldbus, the parameter is updated in nonvolatile memory.

Following is a description of the format of the Data Link Configuration section.

The first line of the Data Link Configuration section is as follows:

```
[Data Link]
```

The general line format for all other lines is as follows:

`variable=value`

where the valid variable names and values are defined in Table A-1.

Table A-1. Valid Variable Names and Values for the Data Link Configuration

Variable Name	Valid Values	Default
`devClass`	BASIC LINKMASTER	none
`nodeAddress`	0x10–0xfb	none

`devClass` indicates whether the device functions as a basic device or a link master device.

`nodeAddress` is the address of the device on the Fieldbus network. It ranges from 0x10 to 0xff. According to the *Fieldbus Foundation Specification*, addresses between 0x10 and 0xf7 are fixed addresses. A

device with a fixed address can be in operational state. You will normally configure your device to have a fixed address. Addresses between 0xf8 and 0xfb are temporary addresses. A device with a temporary address on the bus will eventually be assigned a fixed address to be operational. Addresses between 0xfc and 0xff are visitor addresses. You should not assign your device a visitor address.

A Sample Data Link Configuration section follows:

```
[Data Link]
; Comments are allowed on lines starting with a
; semicolon
devClass=BASIC
nodeAddress=0x20
```

If you specify a node address in the range 0xf8 through 0xfb, your device may show up on the bus at any default address. You may use a system configurator such as NI-FBUS Configurator to assign a fixed address to your device.

System Management Configuration Section Format

Appendix
B

This appendix explains how to structure the System Management Configuration section of your Device Configuration `.ini` file.

The System Management Configuration section of your Windows `.ini` Device Configuration file must be converted to C code and linked with your application before the Round Card can communicate on the Fieldbus network. The code generated by running this file through the Configuration Generator automatically configures your card. When a parameter is changed over the Fieldbus, the parameter is updated in nonvolatile memory.

Following is a description of the format of the System Management Configuration section:

The names of the sections in System Management Configuration are as follows:

```
[MIB]
...
[FB Schedule 0]
...
[FB Schedule 1]
...
...
[FB Schedule N]
...
```

The general line format for all other lines is as follows:

`variable=value`

where the valid variable names and values are defined in Table B-1.

☞ **Note:** ***Most of the variables in Table B-1 are optional. In fact, only the*** `devID` ***is required. For other variables, default values will be used if values are not defined in the configuration file.***

Table B-1. Valid Variable Names and Values for the MIB Sections

Variable Name	Valid Values	Implied Units	Default
clockSyncInterval	0–255	seconds	10
macrocycleDuration	32-bit unsigned integer	1/32 ms	0x8000
primaryTimeMaster	0–255	n/a	none
devID	(ASCII string identifier of this device)	n/a	none
pdTag	(ASCII string tag for this device)	n/a	blank tag
T1	0–0xffffffff	1/32 ms	0x40000
T2	0–0xffffffff	1/32 ms	0x40000
T3	0–0xffffffff	1/32 ms	0x8000

Each [FB Schedule] section denotes a single entry in the FB Schedule.

Table B-2. Valid Variable Names and Values for the FB Schedule Sections

Variable Name	Valid Values	Implied Units	Default
offset	32-bit unsigned integer	1/32 ms	none
index	0–65535	n/a	none
vfdRef	32-bit unsigned integer	n/a	none

Dimensions

Appendix C

This appendix describes the dimensions of the Fieldbus Round Card and the GPIO Daughter Card.

All dimensions are in inches.

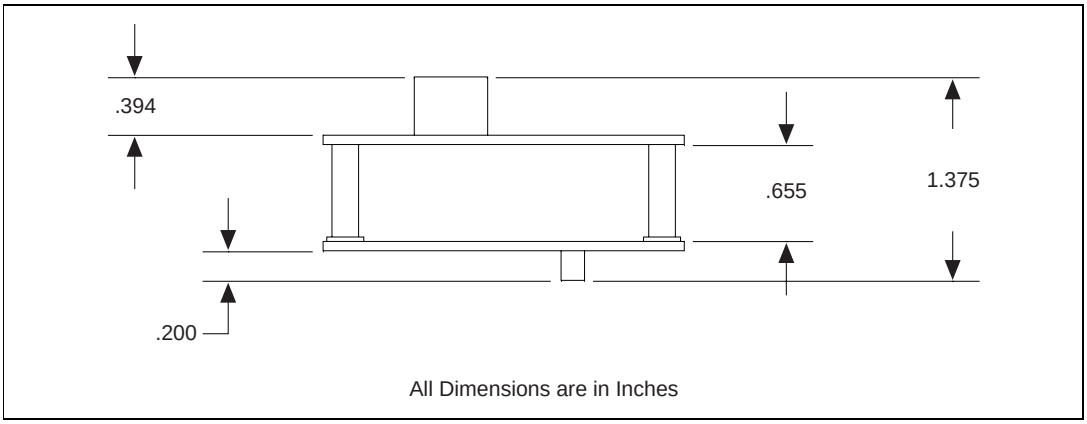

Figure C-1. Combined Dimensions of Fieldbus Round Card and GPIO Daughter Card

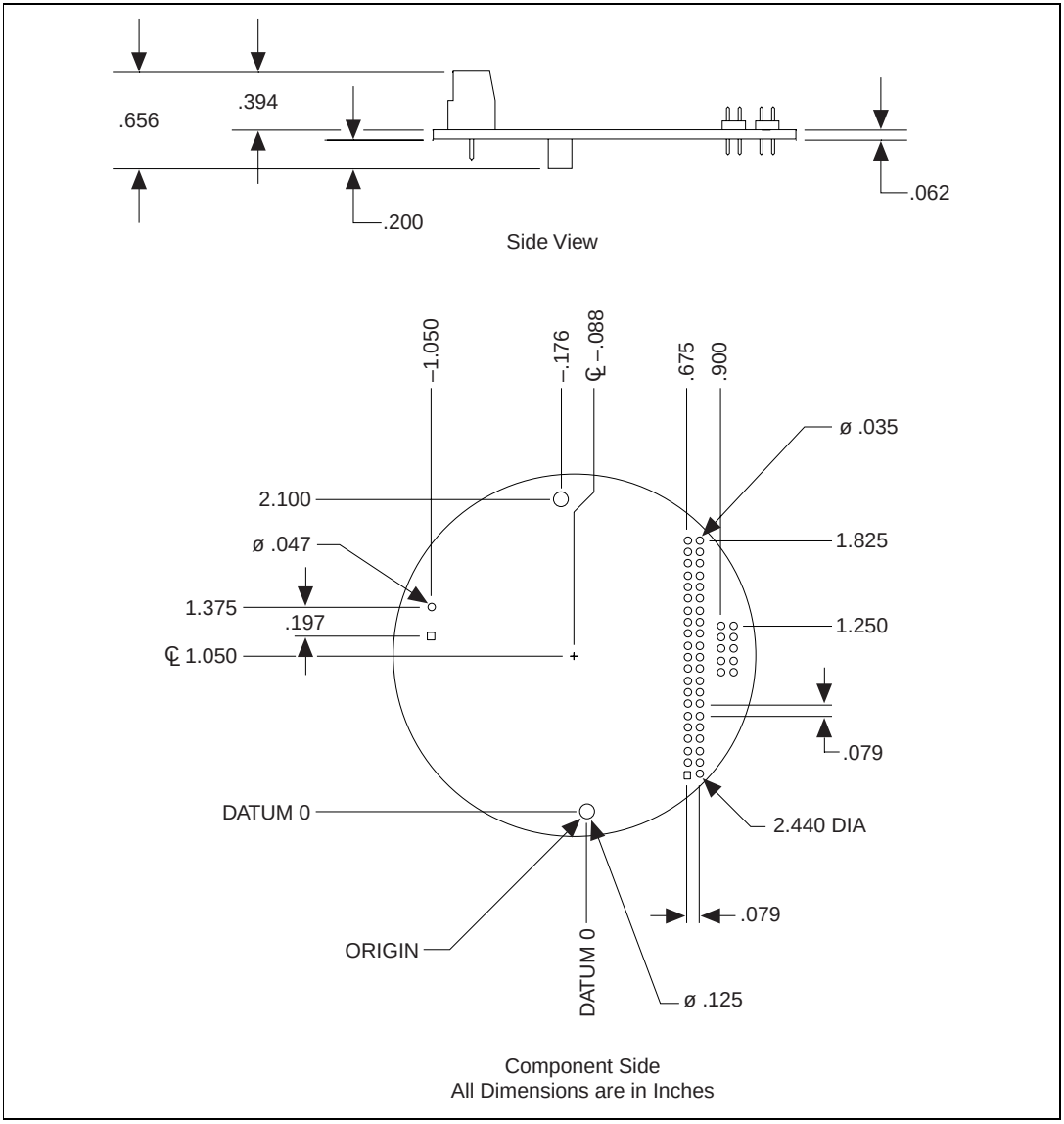

Figure C-2. Fieldbus Round Card Dimensions

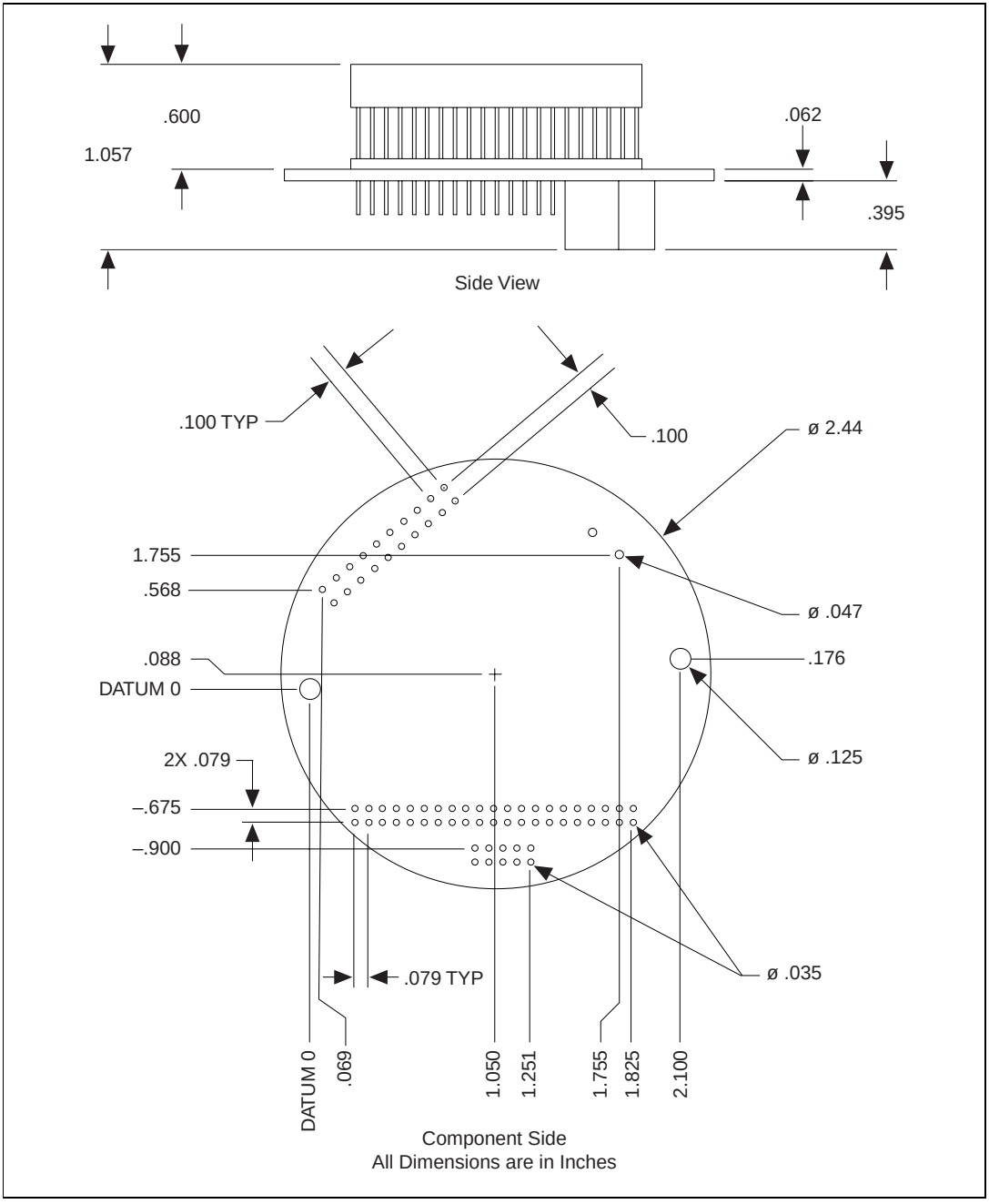

Figure C-3. GPIB Daughter Card Dimensions

Specifications

Appendix
D

This appendix describes the characteristics of the Fieldbus Round Card, the SP Daughter Card, and the GPIO Daughter Card.

Table D-1. Fieldbus Round Card Specifications

Characteristic	Specification
Dimensions	6.2 x 1.52 cm (2.44 x 0.6 in.)
Processor	Motorola MC68331, 8/4 MHz system clock
Crystal Clock Frequency	25 kHz
Fieldbus Interface	Fuji Electric Frontier-1, bus-powered, 31.25 kb/s
Fieldbus Power Supply	9–32 V at 10 mA at 4 MHz, 50 Ω terminated or 9–32 V at 20 mA at 8 MHz, 50 Ω terminated
Power Supply to External Electronics	5 V at 4 mA
Operating Environment Temperature Range Relative Humidity	 0° to 70° C 10% to 90% noncondensing
Storage Environment Temperature Range Relative Humidity	 –40° to 100° C 10% to 90% noncondensing

Table D-2. Fieldbus Round Card Components

Component	Location on Round Card
Fieldbus Connection	2-position terminal block JI
Daughter Card Interface	Header W1 (2 X 21 position)
BDM Port	Header W2 (2 X 5 position)
Reset Switch	Switch S1
Simulate Mode Jumper	Jumper A on W3
Program Mode Jumper	Jumper B on W3

Table D-3. Serial Programming Daughter Card Specifications and Components

Characteristic	Specification
Dimensions	6.2 x 1.52 cm (2.44 x 0.6 in.)
Daughter Card Interface	Header J4 (2 X 21 position)
Host PC to BDM Port Connector	Header W3 (2 X 5 position)
3 V/5 V/RS-232 Serial Port Connections	D-SUB 9 female connector J1 3-position screw terminal: Pin 1—Receive Pin 2—GND Pin 3—Transmit
Daughter Card BDM Port Interface	Header J2
3 V/5 V/RS-232 Transmit Signal Selector	Jumper W1 (1 X 3 position) Jumper across W1(1) and W1(2)—3 V/5 V tolerant transmit signal Jumper across W1(3) and W1(2)—RS-232 transmit signal

Table D-3. Serial Programming Daughter Card Specifications and Components (Continued)

Characteristic	Specification
3 V/5 V/RS-232 Receive Signal Selector	Jumper W2 (1 X 3 position) Jumper across W2(2) and W2(3)—3 V/5 V tolerant receive signal Jumper across W2(1) and W2(2)—RS-232 receive signal
Operating Environment Temperature Range Relative Humidity	 0° to 70° C 10% to 90% noncondensing
Storage Environment Temperature Range Relative Humidity	 –40° to 100° C 10% to 90% noncondensing

Table D-4. GPIO Daughter Card Specifications and Components

Characteristic	Specification
Dimensions	6.2 x 1.52 cm (2.44 x 0.6 in.)
Power Supply	12–24 VDC at 50 mA at 4 MHz, 2-position terminal block J1
Digital Input	Four non-configurable 5 V CMOS inputs
Digital Output	Four non-configurable 5 V CMOS outputs
Analog Inputs	2 channels, 500 samples/s, single-ended Input voltage 0–10 V Input current 4–20 mA Accuracy ±0.5% FSR Resolution 12-bit

Table D-4. GPIO Daughter Card Specifications and Components (Continued)

Characteristic	Specification
Analog Outputs	4–20 mA current output channel Accuracy ±0.5% FSR Resolution 12-bit 0–10 V voltage output channel Accuracy ±0.5% FSR Resolution 12-bit
Input/Output Connector	Header W1 (2x10 position)
Power Supply Connector	2-position terminal block J1
Daughter Card Interface	Header J3 (2x21 position)
BDM Port Connector	Header J2 (2x5 position) (BDM signals not used on the card)
Operating Environment Temperature Range Relative Humidity	 0° to 70° C 10% to 90% noncondensing
Storage Environment Temperature Range Relative Humidity	 –40° to 100° C 10% to 90% noncondensing

Glossary

Prefix	Meanings	Value
m-	milli-	10^{-3}
c-	centi-	10^{-2}
k-	kilo-	10^{3}
M-	mega-	10^{6}

°	Degrees
Ω	Ohms
A	Amperes
A/D	Analog-to-digital
AI	Analog input
API	Application Programmer's Interface
ASCII	American Standard Code for Information Interchange
b	bytes
baud	The signaling rate of a line; bits per second
BDM	Background Debug Mode
b	bytes
bit	A binary digit; a digit (1 or 0) in the representation of a number in binary notation
byte	Eight related bits of data
C	Celsius

CMOS	complementary metal-oxide semiconductor
D/A	Digital-to-analog
DIO	Digital input/output
FB	Function Block
FSR	full-scale reading
FTP	File Transfer Protocol
GND	Ground
GPIO	General Purpose Input/Output
HART	HART Field Communications Protocol
hex	Hexadecimal
Hz	Hertz
in.	inches
I/O	Input/Output
KB	Kilobytes of memory
LED	Light-emitting diode
MB	megabytes of memory
OEM	Original equipment manufacturer
PC	Personal computer
PROM	Programmable read-only memory
RAM	Random-access memory
ROM	Read-only memory
RXD	Receive data
s	Seconds
snap	Read from the communications stack

SP	Serial Programming
SPI	Serial programming interface
SRAM	Serial random-access memory
SROM	Serial read-only memory
TXD	Transmit data
V	volts
VDC	volts direct current
VCR	Virtual Communication Relationship
VFD	Virtual Field Device