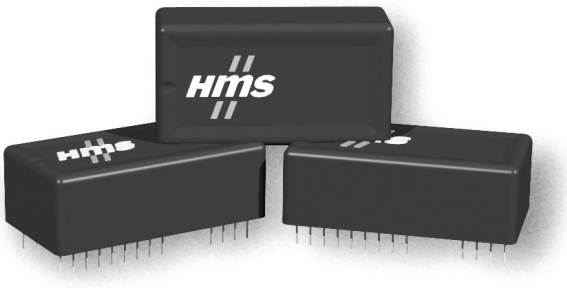


Design Guide
AnyBus®-IC

Rev. 1.33

HMS Industrial Networks AB

	☎	✉
Germany	+49- 721 - 96472 - 0	sales-ge@hms-networks.com
Japan	+81- 45 - 478 -5340	sales-jp@hms-networks.com
Sweden	+46- 35 - 17 29 20	sales@hms-networks.com
U.S.A	+1- 773 - 404 - 3486	sales-us@hms-networks.com

Table of Contents

Chapter 4 MIF Interface

Chapter 5 I/O Interface Data Mapping

Chapter 6 Parameters

Chapter 7 Initialisation

About This Document

This design guide is intended to provide a good understanding of the functionality shared by the AnyBus-IC range of communication modules. This document does not describe any of the fieldbus specific features available on the various versions of the AnyBus-IC module. (Please consult the separate Fieldbus Appendices for more information)

Important User Information

The data and illustrations found in this document are not binding. We, HMS Industrial Networks AB, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks AB. HMS Industrial Networks AB assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the application meets all performance and safety requirements including any applicable laws, regulations, codes, and standards.

Related Documentation

Document name	Author
AnyBus-IC Profibus DP Appendix	HMS
AnyBus-IC DeviceNet Appendix	HMS
AnyBus-IC Ethernet/IT Appendix	HMS
AnyBus-IC EtherNet/IP Appendix	HMS
Modbus Protocol Reference Guide	Modicon

Revision List

Revision	Date	Author	Chapter	Description
<1.20	-	-	-	(See previous revisions)
1.20	2002-10-11	PeP	3 8	Corrected shift register schematic Minor schematic update
1.25	2002-10-18	PeP	3 6	Updated automatic baudrate detection flowchart Corrected default value for parameter #9 Changed the name of parameter #9
1.30	2002-11-13	PeP	All 2 3 8	Minor cosmetic updates Updated switch schematics & info Updated RS485 schematic Updated schematic (DeviceNet shield connection)
1.31 -1.32	2003-10-15	ToT/PeP	All, 3, 6, 8	Minor corrections and adjustments
1.33	2004-04-27	PeP	All 6 1	Minor corrections Added information about FBNA bit (parameter #8) Improved pin layout table

Conventions used in this manual

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The term 'module' is used when referring to the AnyBus-IC
- The term 'application' is used when referring to the hardware that is connected to the application connector.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.

Support

Europe (Sweden)

E-mail:	support@hms-networks.com
Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com

HMS America

E-mail:	us-support@hms-networks.com
Phone:	+1-773-404-2271
Toll Free:	888-8-AnyBus
Fax:	+1-773-404-1797
Online:	www.anybus.com

HMS Germany

E-mail:	ge-support@hms-networks.com
Phone:	+49-721-96472-0
Fax:	+49-721-964-7210
Online:	www.anybus.com

HMS Japan

E-mail:	jp-support@hms-networks.com
Phone:	+81-45-478-5340
Fax:	+81-45-476-0315
Online:	www.anybus.com

About the AnyBus-IC

The AnyBus-IC integrates all analog and digital functionality required to communicate on a supported fieldbus network into a single chip. It can be used with intelligent as well as non-intelligent applications, using a serial communication interface, and/or using external shift-registers to form digital inputs and outputs.

It also features an easy to use monitor interface that can be used with most terminal emulation programs on a standard PC.

The module has a 32-pin dual in line case with a standard footprint, and requires only a single 5V DC power supply. Standardization of mechanical, electrical and software interfaces ensures that the different AnyBus-IC models are fully interchangeable. This also means that the same PCB layout can be used for the different fieldbus systems.

The AnyBus-IC is compatible with Profibus-DP, Devicenet and Ethernet. A separate version is available for each network type.

Key Features

Flexible Data Exchange Interfaces

Apart from the fieldbus interface, the module features two additional data exchange interfaces (SCI and SSC). These interfaces operate fully independently of each other and can be used simultaneously.

- **Serial Communication Interface (SCI)**

  Intelligent devices such as incremental encoders, sensors/actuators, operating terminals and motor control units normally have their own micro controller. The serial 2-wire TTL interface (SCI) connects the module to the micro controller of an intelligent device.

  This provides access to internal parameters of the module as well as I/O data exchange through the built in Modbus RTU protocol. By using this protocol, it is possible to address several AnyBus-IC modules on the same serial line.

- **Synchronous Serial Channel (SSC)**

  For non-intelligent devices, like valve terminals and modular I/O devices, the module features a clocked shift register interface (SSC) which has the capacity of connecting a maximum of 128 input signals and 128 output signals.

Flexible I/O Data Mapping

Using a flexible memory mapping system, data received on one interface can be mapped to another and vice versa.

Monitor Interface (MIF)

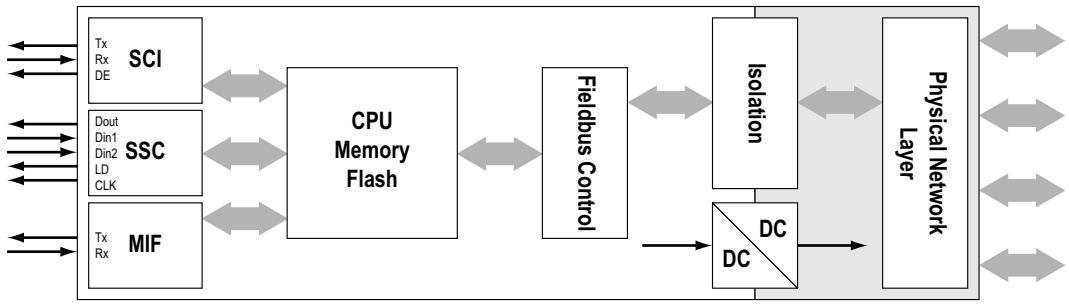
In addition to the serial data interface, the module features a second serial 2-wire interface used for configuration and monitoring using a standard PC terminal emulation program.

The use of this interface is optional, but it provides access to the various parameters of the module, as well as functions for data monitoring and configuration. It also provides a simple way of upgrading the internal firmware of the module.

Overview

Internals

Below is a schematic overview of the AnyBus-IC; its various communication interfaces, internal data-path, and the fieldbus interface.

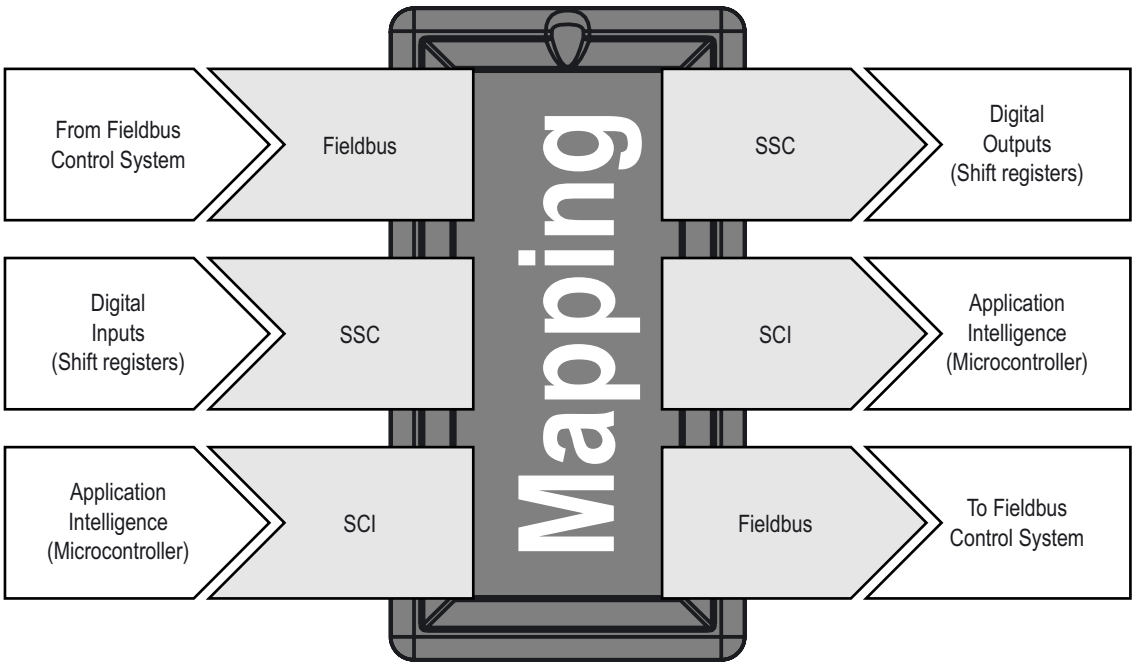


Data Mapping

The module features a flexible memory mapping system. Data received on any of the data exchange interfaces can be mapped (i.e connected) to another data exchange interface, thus forwarding this data from one interface to the other.

This way, the module can be used not only for fieldbus connectivity, but also to provide internal I/O inside the application. (i.e by mapping SSC I/O to the SCI interface and vice versa)

By mapping fieldbus I/O to the SSC interface, the digital inputs / outputs provided by this interface are instantly available from the fieldbus.



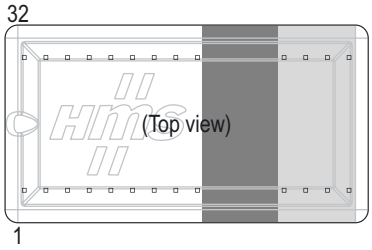
Application Connector

The module features a standard 32-pin DIL connector to connect to the host application.

Pins 13 - 20 on this connector are used for fieldbus specific signals which are specified in the separate fieldbus Appendices.

For mechanical information about the connector, see D-1 "Measurements".

For more information about signal levels, power requirements etc., see Appendix C-1 "Electrical Characteristics".



Pin	Signal	Description	Direction	Page
1	Vcc	+5V Power Supply	Input	C-2
2	/SSC Reset Out	SSC Reset signal (Active Low)	Output	2-1
3	/SSC LD	SSC Load signal (Active Low)	Output	2-1
4	SSC DO	SSC Data Output	Output	2-1
5	SSC DI2	SSC Data Input 2	Input	2-1
6	SSC DI1	SSC Data Input 1	Input	2-1
7	SSC CLK	SSC Clock	Output	2-1
8	/RESET	Module reset (Active Low)	Input	8-3
9	Vcc	+5V Power Supply	Input	C-2
10-12	NC	-	-	-
13	FB1	Fieldbus specific signal 1[a]	(Fieldbus specific)	See note[a]
14	FB2	Fieldbus specific signal 2[a]	(Fieldbus specific)	See note[a]
15	FB3	Fieldbus specific signal 3[a]	(Fieldbus specific)	See note[a]
16	FB4	Fieldbus specific signal 4[a]	(Fieldbus specific)	See note[a]
17	PE	Fieldbus specific signal PE[a]	(Fieldbus specific)	See note[a]
18	SHIELD	Fieldbus specific signal SHIELD[a]	(Fieldbus specific)	See note[a]
19	FB5	Fieldbus specific signal 5[a]	(Fieldbus specific)	See note[a]
20	FB6	Fieldbus specific signal 6[a]	(Fieldbus specific)	See note[a]
21-23	NC	-	-	-
24	GND	GND Power Supply	-	C-2
25	NC	-	-	-
26	/INT [BLE]	Interrupt (Active Low) & Boot loader enable switch	Output (Input)	8-2
27	MIF Tx	MIF Transmit signal	Output	4-1
28	MIF Rx	MIF Receive signal	Input	4-1
29	SCI DE [AUTO]	SCI DE signal (Auto Initialisation)	Output (Input)	3-1
30	SCI Tx	SCI Transmit signal	Output	3-1
31	SCI Rx	SCI Receive signal	Input	3-1
32	GND	GND Power Supply	-	C-2

a. Consult each separate fieldbus appendix for further information about these signals

Important! Different fieldbus systems have different electrical characteristics. The fieldbus specific pins should not be connected to a fieldbus type other than the one provided by the particular AnyBus IC model used. E.g. AnyBus-IC for Profibus-DP should not be connected to DeviceNet. Failure to observe this may damage the module and other equipment connected to the fieldbus interface pins.

SSC Interface

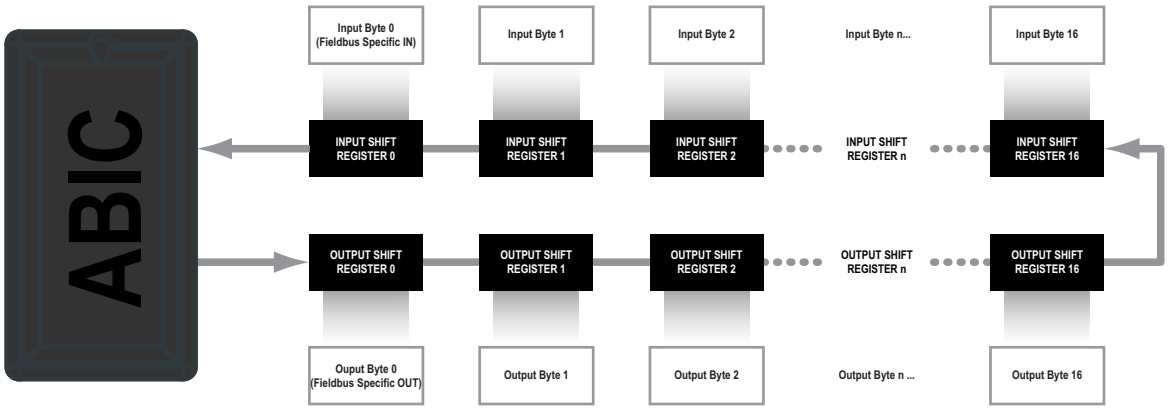
The SSC interface (Synchronous Serial Channel) is a clocked shift register interface similar to the Motorola SPI (Serial Peripheral Interface). It should be used for I/O data exchange and fieldbus specific input/output signals such as node address and led indications. It can be used in parallel with the SCI interface or stand alone mode, using auto initialisation (see 2-3 “Signal Descriptions” for more information.)

When using the module in stand alone /auto initialise mode, parameters can only be set using the MIF interface, see 4-1 “MIF Interface”.

Input / Output Registers

The SSC interface requires external shift registers to form digital inputs and outputs. The interface supports up to 17 input data shift registers (136 bits) and 17 output data shift registers (136 bits), of which 16 can be used for output data, and 16 for input data.

The first Input and Output Registers are by default reserved for fieldbus specific functions. The module can however be configured to use these registers for data exchange.



The module automatically detects the number of input- and output- registers during start-up. This information is stored in parameter #52 “SSC In Auto” and parameter #55 “SSC Out Auto”.

If required, the number of input / output registers can be configured manually using parameter #51 “SSC In Config” and #54 “SSC Out Config”. Parameter #8 “Configuration Bits” is used to determine whether the module should use the manually configured setting or auto detected setting.

FB Specific Input Registers

Input shift register 0 is by default reserved for fieldbus specific features, e.g node address configuration or baudrate settings. This function can be disabled by setting the FBNP-bit in parameter #8 "Configuration Bits". In this case this register is used for normal data exchange instead. (Note that this does not extend the number of possible inputs available for data exchange, i.e the maximum amount of input registers is 16)

The module supports both several types of switches. The type of switch used is configured using parameter #9 "Switch Coding".

Note: The Fieldbus Specific Input register can be used even if no SSC I/O data is present, e.g. if all data exchange is made through the SCI interface and only the node address is taken from the SSC interface.

FB Specific Output Registers

Output shift register 0 is by default reserved for fieldbus specific features, e.g to connect LEDs etc. This function can be disabled by setting the FBLP-bit in parameter #8 "Configuration Bits". In this case this register is used for normal data exchange instead. (Note that this does not extend the number of possible outputs available for data exchange, i.e the maximum amount of output registers is 16)

Note: The Fieldbus Specific Output register can be used even if no SSC I/O data is present, e.g. if all data exchange is made through the SCI interface and only the LEDs are used on the SSC interface.

Signal Descriptions

The SSC Interface consists of 6 signals:

- **SSC Reset Out (Pin 2)**
  This active low signal is used to reset the shift registers. (The use of this signal is optional)
  Note that this signal is not related to the module reset pin (Pin 8)
- **SSC LD (Pin 3)**
  Load signal to the shift registers. When this signal goes low, the contents of the shift register pins are loaded into the internal shift register.
- **SSC DO (Pin 4)**
  This signal contains the serial data output to the shift registers.
- **SSC DI2 (Pin 5)**
  Serial data input 1 from the shift registers.
- **SSC DI1 (Pin 6)**
  Serial data input 2 from the shift registers.
- **SSC CLK (Pin 7)**
  Clock signal to the shift registers.

SSC Input signals only

If only input shift registers are used, the SSC DO signal should be connected to the SSC DI1 signal. The module will then detect that no output shift registers are present and also the number of input shift registers.

SSC Output signals only

If only output shift registers are used, the SSC DI1 signal should be connected to the SSC DI2 signal. The module will then detect that no input shift registers are present and also the number of output shift registers.

SSC Input and Output signals

If both input and output shift registers are used, the SSC DI1 signal is connected on the serial data line between the output shift registers and the input shift registers. The module will then detect the number of input and output shift registers.

No SSC Inputs / Outputs

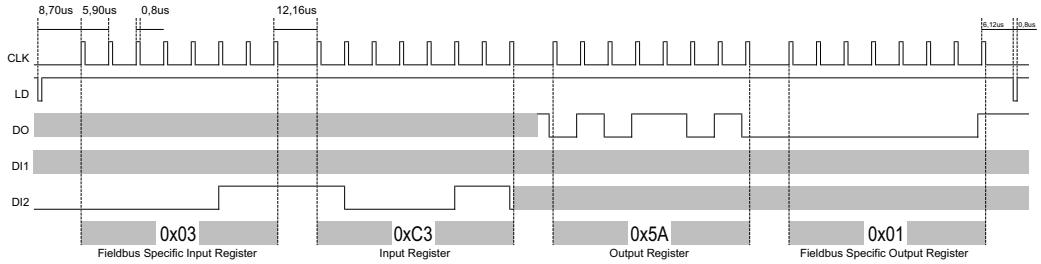
When this interface is not used, the SSC signals should be left unconnected.

Timing

Note: An update sequence can be interrupted by other tasks, but a started sequence will always finish after the interruption in order to keep data consistency.

Normal Operation Sequence

The update sequence is run approximately every 6ms. The module starts with reading the Fieldbus Specific Input register followed by the input data registers. When the input data is read, the output data is shifted out using the output data registers, followed by the Fieldbus Specific Output register.

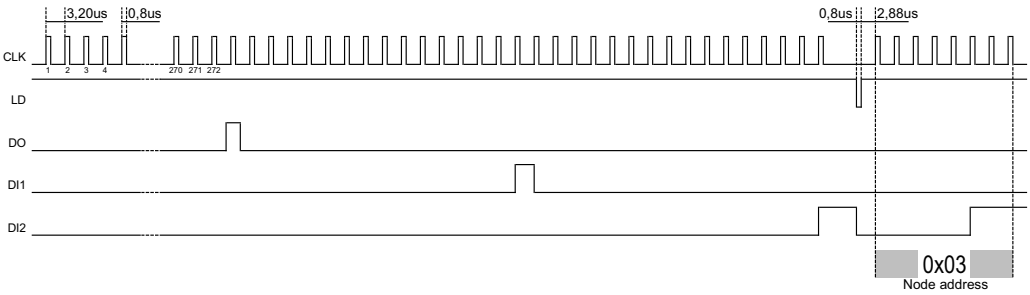


The example above reads 0x03 from the Fieldbus Specific Input register, 0xC3 from the first Input Register, and writes 0x5A to the first output register and 0x01 to the Fieldbus Specific Output Register.

Shift Register Initialisation Sequence

During start-up, the module calculates the number of attached shift registers. It starts with clearing the internal register in the shift registers by shifting zeroes through the registers the maximum number of times possible (17 Input Registers x 8 bit + 17 Output Registers x 8 bit = 272 times).

When the internal register is cleared, a 1 is shifted through the shift registers. When the bit reaches DI1 the number of output registers is calculated and when the bit reaches DI2 the number of input registers is calculated. The initialisation sequence ends with reading the Fieldbus Specific Input register.

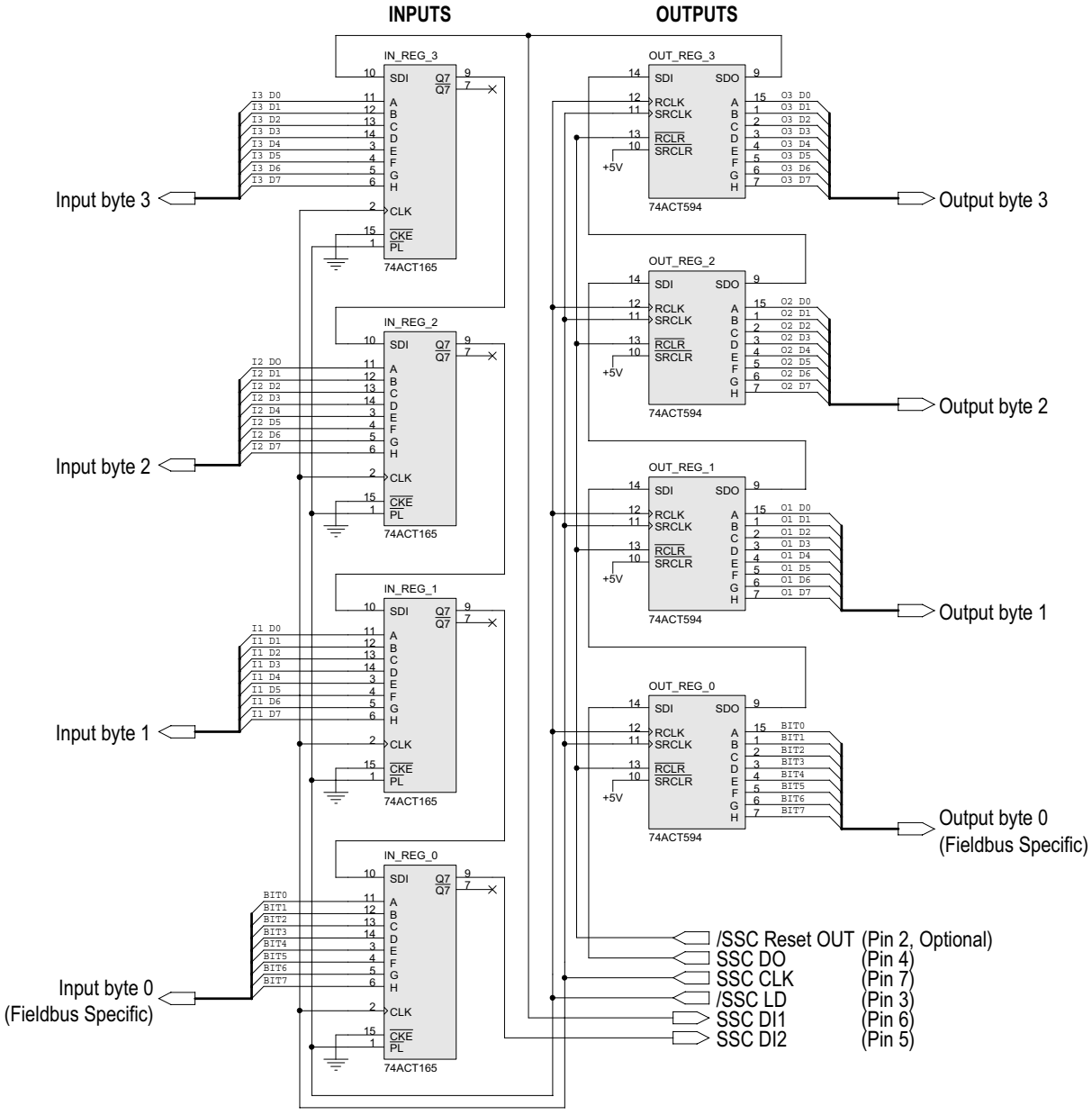


The example above reads 0x03 from the Fieldbus Specific Input register after detecting the number of shift registers.

Basic Shift Register Connection

In the schematic below, the fieldbus specific inputs / outputs are connected closest to the module in the shift register loop. The next stage of shift registers holds the first input / output byte. The last stage of shift registers are connected to SSC DI1.

The LSB of each input / output byte is available on input/output "A", and MSB on input/output "H".



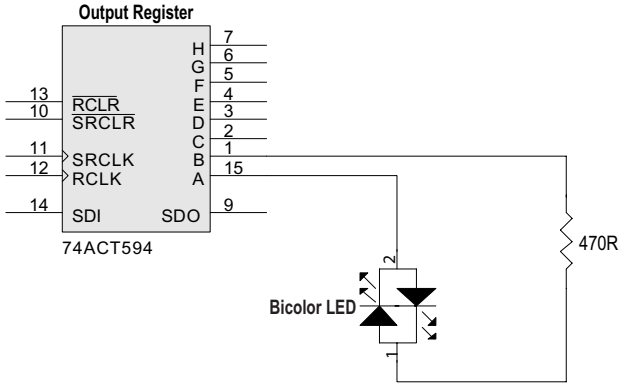
Fieldbus Specific Output, connection examples

The fieldbus specific output register is placed closest to the module in the shift register loop, and is used for fieldbus specific features, e.g led indicators.

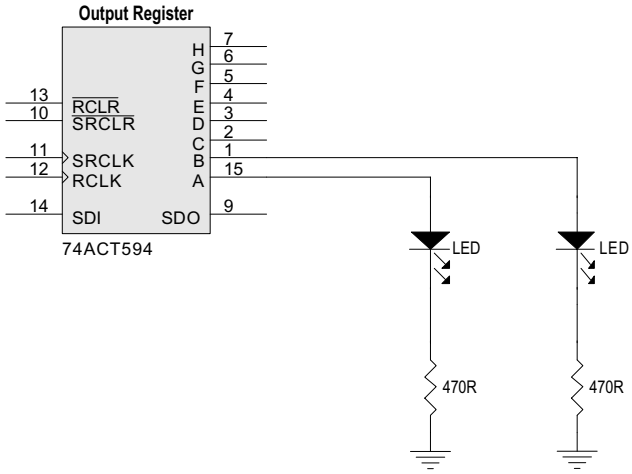
Direct LED connection, no external drivers

This type of connection requires shift registers of sufficient current capacity to drive the LEDs directly, e.g. 74ACTxxx.

Bi-colour LED.



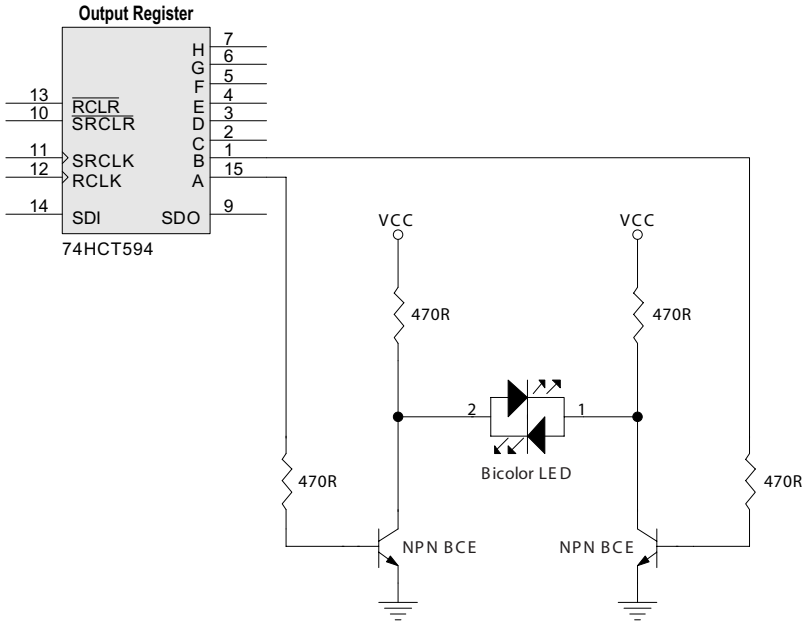
Single colour LED



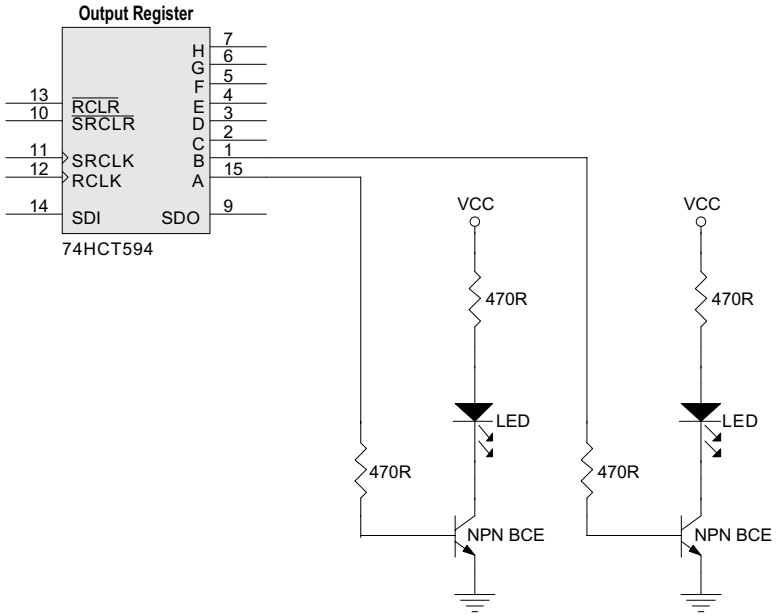
LED Connection using external driver circuit

This type of connection can be used with 74HCTxxx type shift registers.

Bi-colour LED



Single colour LEDs with external drivers



Fieldbus Specific Input, connection examples

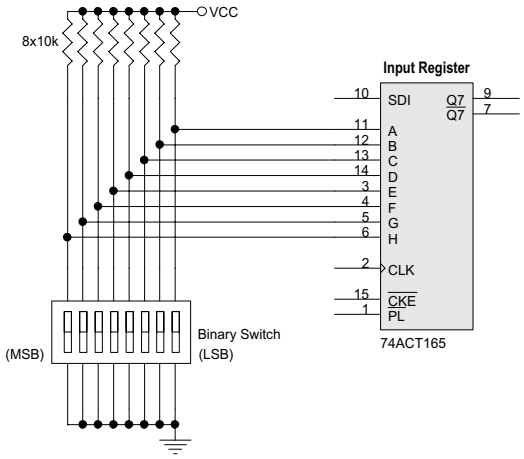
The fieldbus specific input register is placed closest to the module in the shift register loop, and is used for fieldbus specific features, e.g switches for fieldbus node address and baud rate configuration.

The module does not invert the bit-values of the switch; A low input voltage is interpreted as a logic zero (0), and a high input voltage is interpreted as a logic one (1).

The module supports both binary and BCD-coded switches. The type of switch used is configured in parameter #9 "Switch Coding". Note that different switches has different electrical characteristics and markings; be sure to select a switch suitable for the circuits shown below. Other switches may require a different type of connection.

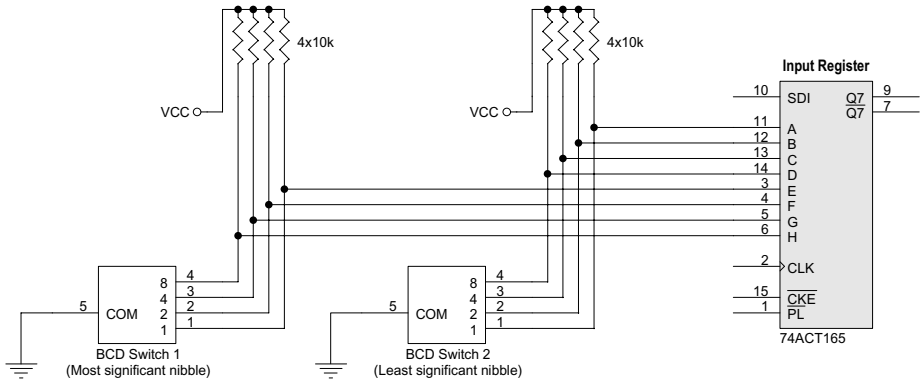
Binary Switch

A closed switch in the figure below, will be interpreted as a logic zero (0), and an open switch, will be interpreted as a logic one (1).



BCD-coded Switch

The BCD-coded switches in the figure below are of sinking kind (all bits are connected to common when the switch is in position zero. Connecting pull-up resistors to the switches as shown in the figure provides the true BCD code to the shift register.



SCI Interface

The SCI interface is a serial interface intended for intelligent applications, to connect the module to e.g. a micro processor. A simple Modbus based protocol is provided for configuration and data exchange. For more information about the Modbus implementation, see 9-1 "Modbus Protocol".

The application transfers the process image to the module during normal operation. This data is then be forwarded to the other data exchange interfaces according to the I/O mapping parameters. This works both ways, i.e data can also be forwarded from the other interfaces to the SCI interface.

The module supports up to 32 bytes of input data and 32 bytes of output data on this interface. This data is mapped to Modbus addresses, see 6-1 "Accessing Parameters using the SCI interface".

Note: If SCI_DE is connected to GND, this interface will be disabled.

Communication Settings

Parity and Databits

The port settings (parity) for this interface can be configured with parameter #16 (SCI Settings Config). The number of data bits is fixed to 8 bits. The default setting is no parity / 2 Stop bits.

The Modbus RTU specification states that 1 stop bit is used when parity is enabled, and 2 stop bits are used when parity is disabled.

Note that flow control is not supported by the AnyBus-IC module.

Note: When using automatic baudrate detection, these settings are detected by the module automatically.

Modbus RTU Address

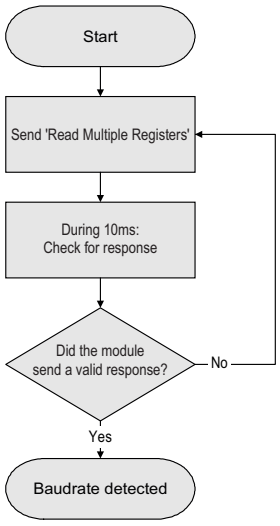
If more than one AnyBus-IC module is connected to the same interface, e.g. with a multi-drop RS485 or RS422 interface, it is important that each module is given a unique Modbus RTU address.

The Modbus RTU address is set using parameter #22 (Modbus RTU Address). Valid settings range from 0 to 247.

Baud rate

The baud rate for this interface can be altered using parameter #14 (SCI Rate Config). This interface also supports automatic baud rate detection, which is the default setting of this parameter.

This function is not a part of the Modbus specification and requires a specific start-up sequence. When the power is turned on, the application should send a "Read Multiple Registers" request to address 0x01 until the module responds with the correct baud rate (see flowchart), this should in normal applications happened within the 20 first attempts.



Note: In order for the automatic baud rate detection functionality to work, the module must be configured to modbus address 01h. This means that it is not possible to use automatic baudrate detection when several AnyBus-IC modules are connected to the same interface.

Signal descriptions

The SCI Interface consists of three signals:

- **SCI_DE (Pin 29)**

  When the SCI interface is used, this active-high signal is used for Data Enable when the interface is connected to e.g. a RS485 interface.

  If this pin is connected to GND before power-on, the module will initialise automatically and the SCI interface will be disabled

  (This signals is internally pulled-up with 10kΩ, thus it can be left unconnected when not used.)

- **SCI_Tx (Pin 30)**

  This signals is used to transmit data from the AnyBus-IC to the application.

  (This signals is internally pulled-up with 10kΩ, thus it can be left unconnected when not used.)

- **SCI_Rx (Pin 31)**

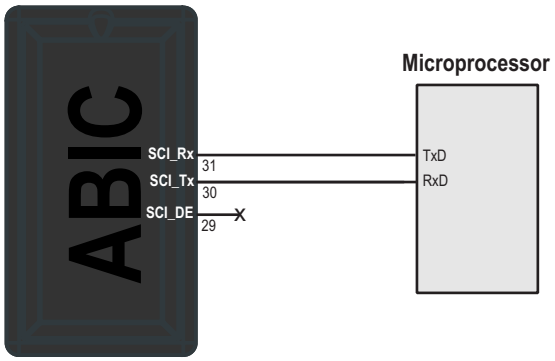
  This signal is used to transmit data from the application to the AnyBus-IC.

  (This signals is internally pulled-up with 10kΩ, thus it can be left unconnected when not used.)

Connection Examples

Direct Micro Processor Connection (TTL - TTL)

When connecting this interface directly to a microprocessor, no inverters or glue logic is needed. Only the SCI_Tx and SCI_Rx signals are used. SCI_DE should be left unconnected, not grounded.

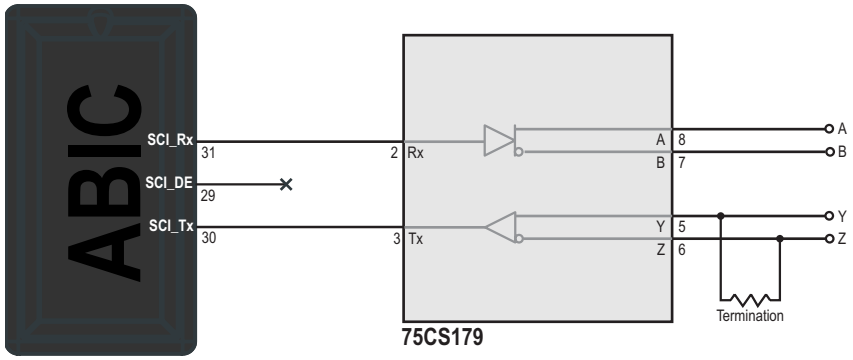


RS-422

RS-422 is designed for greater distances and higher Baudrates than RS-232. RS-422 uses balanced signals for data transmission, offering greater noise immunity.

When the SCI interface is connected to external equipment using RS-422, the signals must be converted from TTL to standard RS-422 levels.

In the example below, a 75CS179 transceiver is used for the signal conversion. The SCI_DE signal is intentionally left unconnected, not grounded.

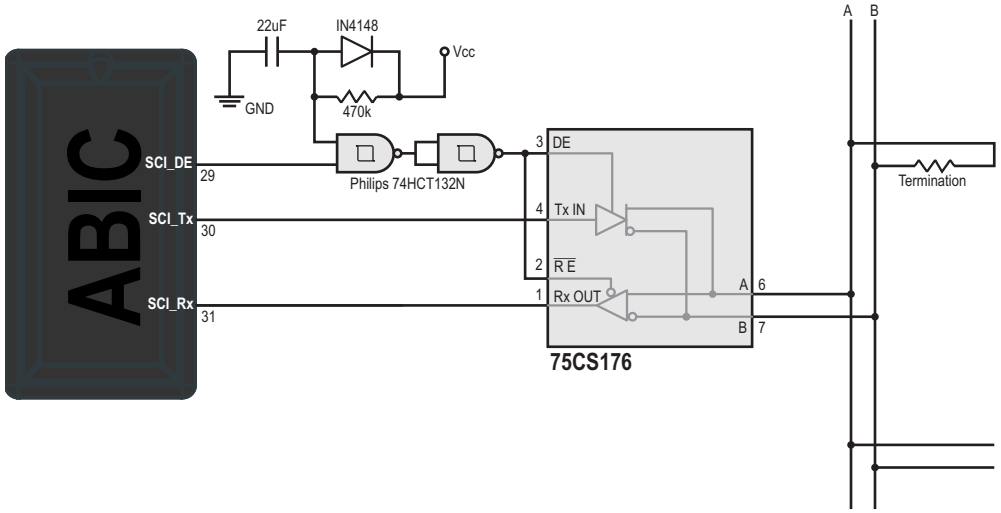


Note: It is recommended to use termination resistors to avoid reflections on the serial line.

RS-485 (Multidrop)

RS-485 is similar to RS-422 and offers greater noise immunity than RS-232. It is used for multipoint communications, i.e when several AnyBus-IC modules are connected to the same interface.

When the SCI interface is connected to external equipment using RS-485, the signals must be converted from TTL to standard RS-485 levels. In the example below, a 75CS176 is used for the conversion.



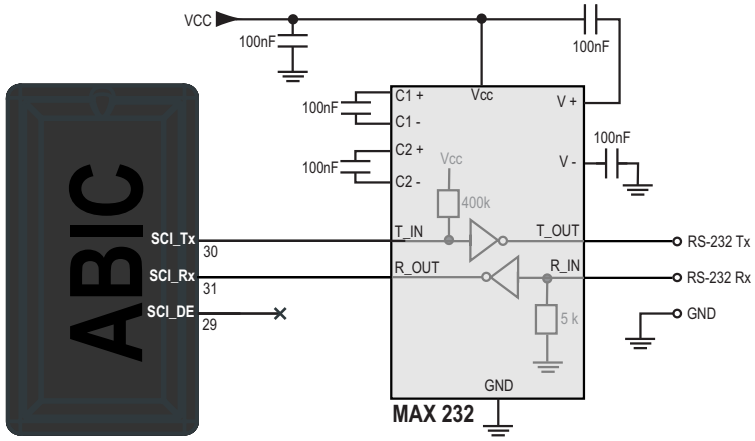
Depending on the type of fieldbus used, undefined data may be sent to the RS485 network during start-tup. To prevent this, use the circuit shown above for the DE signal. This will prevent the module from accessing the RS485 network for approximately 4 seconds after power on.

Note: It is recommended to use termination resistors to avoid reflections on the serial line

RS-232

When the SCI interface is connected to external equipment using RS-232, e.g a PC, the signals must be converted from TTL to standard RS-232 levels.

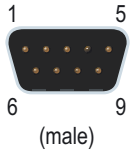
In the example below, a MAX232 transceiver is used for the signal conversion. The SCI_DE signal is intentionally left unconnected, not grounded.



RS232 DSUB 9 Pinout (DTE)

The full RS-232 standard specifies a 25-pin DSUB connector of which 22 pins are used. However, as most of these signals are not used in normal communications, most RS232 connectors provides a subset of the RS-232 standard using a 9 pin DSUB connector. (See pinout below).

Pin	Signal
1	Carrier Detect
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready
7	Request To Send
8	Clear To Send
9	Ring Indicator



Note: For longer cable lengths and/or electrically noisy environments it is recommended to use RS-422 or RS-485.

MIF Interface

The monitor interface provides an easy to use monitor / configuration interface using a standard PC terminal emulation program. Using this interface, it is easy to configure and diagnose parameters and monitor I/O data on the different data exchange interfaces. It is also used for firmware upgrades. The interface features a simple text-based menu system and requires no specific terminal emulation settings.

For information and examples about how to connect the monitor interface to a terminal, see 4-4 “Connection Example (RS-232)”.

Note: Although it is possible to use this interface as a configuration channel for the application, it is not recommended as menu entries may be added, changed, or even removed.

Communication Settings

The port settings (parity and stop bits) for the monitor interface are configured with parameter #20 (MIF Settings Config). The default port settings are no parity and 1 stop bit.

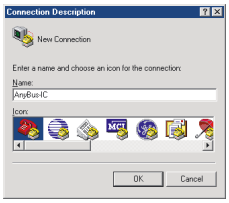
The number of data bits is fixed to 8 bits and cannot be altered. Also note that flow control is not supported by the module.

The baud rate for this interface can be altered using parameter #18 (MIF Rate Config). Default baud rate is 38,4 kbps. Unlike the SCI interface, the monitor interface does not feature automatic baud rate detection.

Terminal Configuration

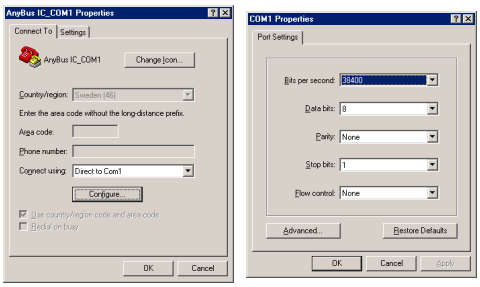
The example below requires the Windows HyperTerminal, but the procedure is usually similar in other terminal emulation programs.

1. Start the Windows HyperTerminal
2. Start a new connection in the File-menu. Name the new connection, e.g. “AnyBus-IC”



3. Select the ‘Properties’ entry in the ‘File’-menu.
4. Select the correct COM-port, speed, and port-settings.

   These settings must match the settings of the module.



Menu System

The MIF interface features an easy to use text-based user interface:

- To enter a sub menu or parameter, type the corresponding number and press <Enter>
- To enter a parameter value, enter the value and press <Enter>
- To return to a previous menu, or cancel a parameter input, press <ESC>
- To redraw the current menu, press <Enter>

Main menu

The main menu provides access to the various sub menus. It also displays the fieldbus type.

```
----------------------------------------
 AnyBus-IC - Main Menu
 Profibus-DP
----------------------------------------
  1 - Module Information
  2 - Parameters
  3 - Monitor
  4 - Firmware Upgrade
----------------------------------------
>
```

Sub Menus

1 - Module Information

This menu consists of two sub-menus, Software Versions which gives information about the current AnyBus-IC firmware, and Product Information, which gives information about the AnyBus-IC serial number and production date

```
----------------------------------------
 AnyBus-IC - Information
----------------------------------------
  1 - Software Versions
  2 - Product Information
----------------------------------------
>
```

2 - Parameters

This menu provides access to all parameters of the module. Note that these parameters can be altered using the SCI interface as well. See 6-1 “Parameters” for a specification of the various parameters.

```
----------------------------------------
AnyBus-IC - Parameters
----------------------------------------
  1 - AnyBus-IC
  2 - FB I/O Settings
  3 - SSC I/O Settings
  4 - SCI I/O Settings
  5 - Fieldbus Specific
----------------------------------------
>
```

3 - Monitor

All active I/O interface areas can be monitored in the monitor menu.

```
-----------------------------------------
 AnyBus-IC - I/O Areas
-----------------------------------------
  1 - Fieldbus Out
  2 - SSC In
  3 - SCI In
  4 - Fieldbus In
  5 - SSC Out
  6 - SCI Out
-----------------------------------------
>
```

Example:

```
--------------------------------
 AnyBus-IC - Fieldbus In
--------------------------------
 Byte #
   0     1100 0011   0xc3
   1     0000 0000   0x00
--------------------------------
```

Note: The I/O interface monitoring is not updated automatically. Press <Enter> to update the screen.

4 - Firmware Upgrade

This menu is used when upgrading the firmware of the module. For more information about firmware upgrades, see Appendix A-1 “Firmware Upgrade”.

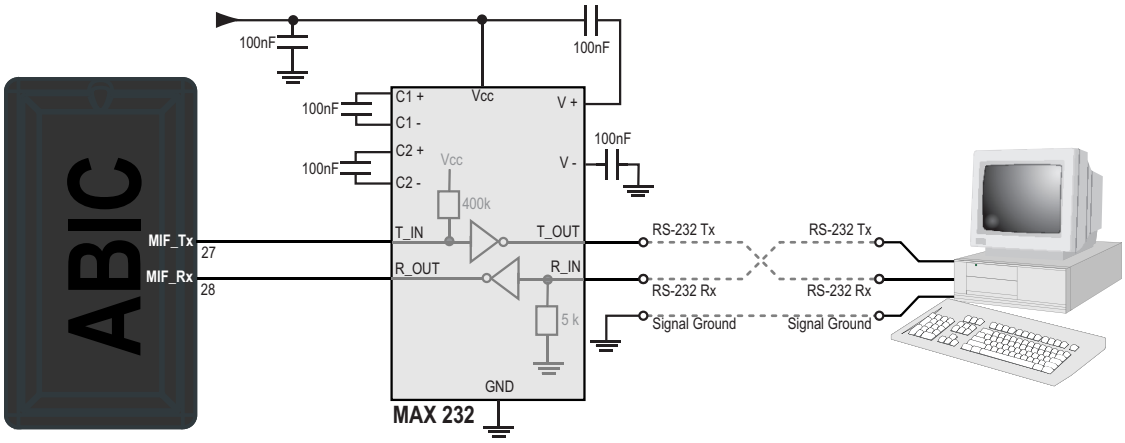
Signal Descriptions

The Monitor Interface consists of two signals:

- **MIF_Tx (Pin 27)**
  Transmit signal from AnyBus-IC to the terminal.
- **MIF_Rx (Pin 28)**
  Received signal from the terminal to the AnyBus-IC.

Connection Example (RS-232)

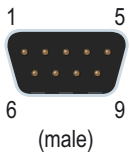
If a standard null-modem (crossed) cable is used, the connection to the PC can be implemented according to the figure below.



PC Serial port Pinout (DTE)

The full RS-232 standard specifies a 25-pin DSUB connector of which 22 pins are used. However, as most av these signals are not used in normal communications, most PC serial ports provides a subset of the RS-232 standard using a 9 pin DSUB connector. (See pinout below).

Pin	Signal
1	Carrier Detect
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready
7	Request To Send
8	Clear To Send
9	Ring Indicator



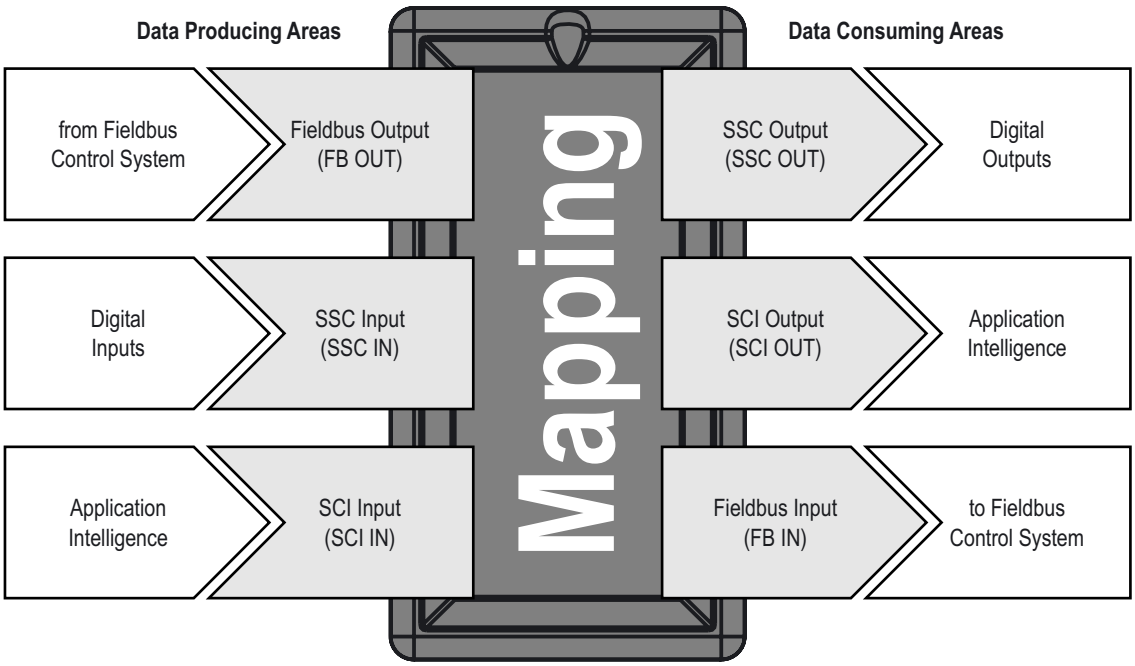
I/O Interface Data Mapping

Note: As in all communication systems, the terms Input and Output can be ambiguous because their meaning depend on which end of the link is being referenced. The convention in this chapter is that Input and Output are always being referenced to the master/scanner end of the link.

Introduction

The different data exchange interfaces in the module can be divided into two areas, one data producing area and one data consuming area.

Data produced by one interface can be mapped to any of the other interfaces, which consumes the data. The I/O parameters determines how the data will be mapped.

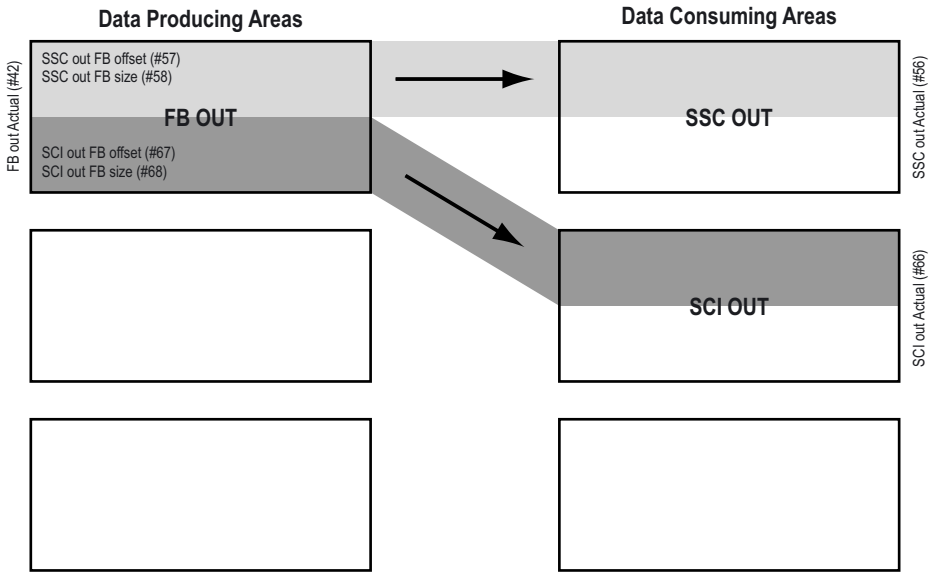


Produced data may be data from the fieldbus master (FB Out), data from the input shift registers (SSC In), or data from the SCI Interface (SCI In).

Data can **not** be mapped from one interface's producing data area to the same interface's consuming data area, i.e. "loop" the data on that interface. However, two data consuming areas can be configured to receive the same data from a data producing area.

FB Output Mapping

The data produced by the fieldbus control system resides in the FB Out area. This data can be mapped to both the SSC- and SCI- Out areas.

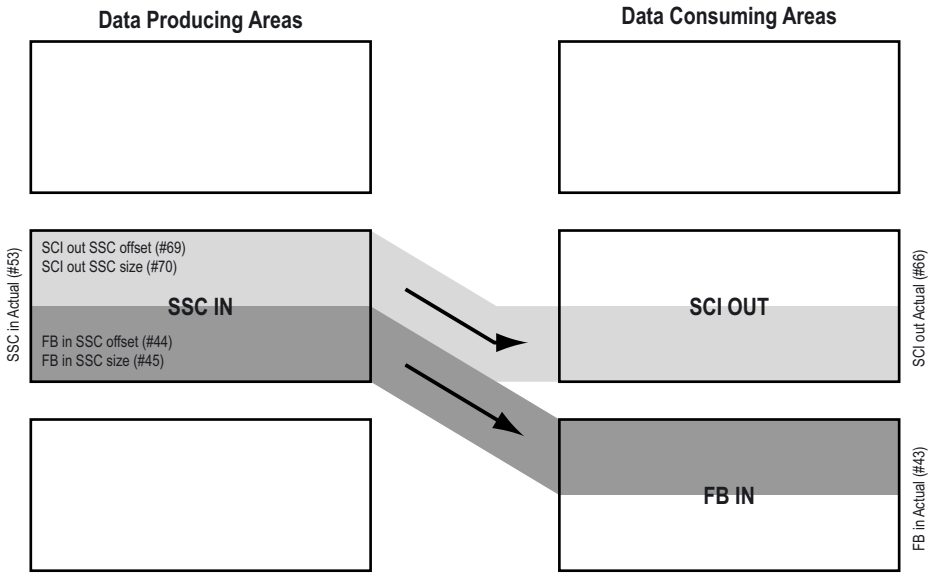


Parameters

Parameter	Description
FB Out Actual (#42)	This parameter specifies the total size of the FB Out area after initialisation. The value specified here is the maximum amount of data that can be mapped to other interfaces.
SSC Out FB Offset (#57)	This is the offset in the FB Out area where the mapping of data to the SSC Out area starts
SSC Out FB Size (#58)	This parameter specifies the amount of data that shall be mapped to the SSC interface.
SCI Out FB Offset (#67)	This is the offset in the FB Out area where the mapping of data to the SCI Out area starts
SCI Out FB Size (#68)	This parameter specifies the amount of data that shall be mapped to the SCI interface.

SSC Input Mapping

The data produced by the SSC input shift registers resides in the SSC In area. This data can be mapped to both the FB In and the SCI Out areas.

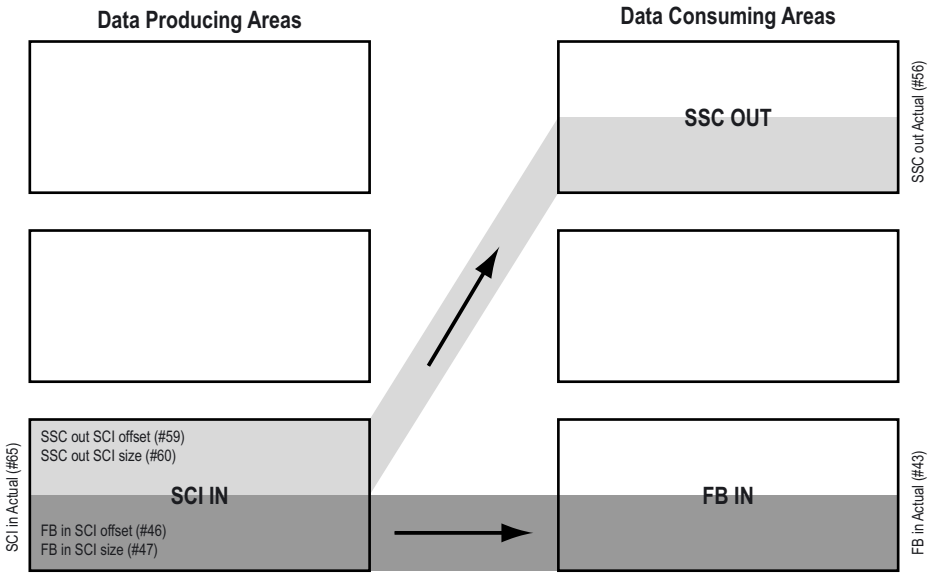


Parameters

Parameter	Description
SSC In Actual (#53)	This parameter specifies the total size of the SSC In area after initialisation. The value specified here is the maximum amount of data that can be mapped to other interfaces.
FB In SSC Offset (#44)	This is the offset in the SSC In area where the mapping of data to the FB In area starts
FB In SSC Size (#45)	This parameter specifies the amount of data that shall be mapped to the FB In area.
SCI Out SSC Offset (#69)	This is the offset in the SSC In area where the mapping of data to the SCI Out area starts
SCI Out SSC Size (#70)	This parameter specifies the amount of data that shall be mapped to the SCI interface.

SCI Input Mapping

The data produced by the application on the SCI interface resides in the SCI In area. This data can be mapped to both the SSC Out and FB In areas.



Parameters

Parameter	Description
SCI In Actual (#65)	This is the actual size of the SCI In area after initialisation. This is the maximum amount of data that can be mapped to the other interface's consumed data areas
FB In SCI Offset (#46)	This is the offset in the SCI In area where the mapping of data to the FB In area starts
FB In SCI Size (#47)	This parameter specifies the amount of data that shall be mapped to the FB In area.
SSC Out SCI Offset (#59)	This is the offset in the SCI In area where the mapping of data to the SSC Out area starts
SSC Out SCI Size (#60)	This parameter specifies the amount of data that shall be mapped to the SSC interface.

Parameters

The module can communicate on the three data exchange interfaces in many different ways. Parameters are used determine how the data should be treated by the module, and how the module should operate.

Parameters can be set both using the MIF and/or SCI interface. (If the SCI interface is disabled, i.e the SCI_DE pin is connected to GND, parameters must be set using the MIF interface)

The parameters are divided into three categories depending on their usage:

- **General AnyBus-IC Parameters**

  These parameters are used for configuration and status information, and are shared by all versions of the AnyBus-IC.

- **I/O Parameters**

  These parameters are used to specify how the data should be treated by the module, i.e how the data is mapped between the data exchange interfaces. These parameters are shared by all versions of the AnyBus-IC.

- **Fieldbus Specific Parameters**

  For information about Fieldbus Specific parameters, consult the separate Fieldbus Appendix.

Note: Parameters with a size of one byte are placed in the least significant byte of the word.

Accessing Parameters using the SCI interface

All parameters and SCI input/output data are mapped as Modbus addresses. The mapping is made according to the table below.

Modbus Register	Description
0x0000 -0x 001F	SCI Input data area (32 bytes)
0x0020 - 0x0FFF	(reserved)
0x1000 - 0x101F	SCI Output data area (32 bytes)
0x1020 - 0x5000	(reserved)
0x5001 -0x 5032	General AnyBus-IC Parameters, see 6-2 “General AnyBus-IC Parameters”
0x5033 -0x 5FFF	(reserved)
0x6000 - 0x6020	I/O Parameters, see 6-20 “I/O Parameters”
0x6021 - 0x6FFF	(reserved)
0x7000 -	Fieldbus Specific Parameters, see separate Fieldbus Appendix

Accessing Parameters using the MIF Interface

Parameters can also be set using the MIF interface. See 4-1 “MIF Interface” for more information. However, since this interface is designed for debugging and configuration purposes, this method should only be used when evaluating the various functions of the module.

General AnyBus-IC Parameters

These parameters are used to configure the basic settings of the module.

#	R/W	Name	Size	Default value	Modbus Address
1	R/W	Module Mode	2 bytes	-	0x5001
2	R	Module Status	2 bytes	-	0x5002
3	R	Module Type	2 bytes	-	0x5003
4	R	Fieldbus Type	2 bytes	-	0x5004
7	R	LED State	2 bytes	-	0x5007
8	R/W	Config Bits	2 bytes	0x0000	0x5008
9	R/W	Switch Coding	1 byte	Fieldbus dependant[a]	0x5009
10	R/W	Offline Action	1 byte	0x00	0x500A
11	R/W	Idle Action	1 byte	0x00	0x500B
12	R/W	Interrupt Config	2 bytes	0x0001	0x500C
13	R	Interrupt Cause	2 bytes	-	0x500D
14	R/W	SCI Rate Config	1 byte	0x00	0x500E
15	R	SCI Rate Actual	1 byte	-	0x500F
16	R/W	SCI Settings Config	1 byte	0x00	0x5010
17	R	SCI Settings Actual	1 byte	-	0x5011
18	R/W	MIF Rate Config	1 byte	0x04	0x5012
19	R	MIF Rate Actual	1 byte	-	0x5013
20	R/W	MIF Settings Config	1 byte	0x00	0x5014
21	R	MIF Settings Actual	1 byte	-	0x5015
22	R/W	Modbus RTU Address	1 byte	0x01	0x5016
23	R/W	Modbus CRC Disable	1 byte	0x00	0x5017
27	R/W	FB Fault Values	48 bytes	0x00	0x501B - 0x5032

a. See separate fieldbus appendix

Module Mode (Parameter #1)

This parameter is used to determine the current operating mode of the module.

Parameter number	1
Modbus Address	0x5001
Default value	-
Range	0x0000 - 0x0005
Size	2 bytes
Stored in NV RAM	No
Access	R/W

Valid Settings

- **0x0000 - Start-up Mode**

  This is the initial value of the parameter in all cases after power-up except when the module is automatically initialised.

- **0x0001 - Normal Operation Mode**

  When all parameters are updated with correct settings, the normal initialisation is triggered by writing this value. If the module is automatically initialised, the parameter automatically gets this value.

Note: If automatic baud rate detection on the SCI interface is enabled, it is not possible to initialise the module in normal operation mode via the monitor interface until the baud rate has been detected.

- **0x0002 - Fieldbus Specific Init**

  If the AnyBus-IC module supports fieldbus specific initialisation (fieldbus dependent) this value starts the initialisation. See fieldbus appendix for more information.

- **0x0003 - Reset Module**

  This value makes a reset of the module. The module needs to be re-initialised to start communicating. Note that this is not the same as mode 0x0004, see below.

- **0x0004 - Set Default**

  All configurable parameters will be set to their factory default value. Note that password protected parameters will not be affected unless the password is entered before sending “Set default”. Note that this is not the same as mode 0x0003, see above.

- **0x0005 - Hardware Self Test**

  By writing this value, an internal hardware test is performed on the AnyBus-IC module. For more information about how the hardware test is performed, see 8-4 “Hardware Self Test”.

Module Status (Parameter #2)

This parameter holds information about the current status of the module. This parameter is also used to indicate the status if a hardware test has been performed (See 8-4 "Hardware Self Test")

Parameter number	2
Modbus Address	0x5002
Default value	-
Range	Bit field
Size	2 bytes
Stored in NV RAM	No
Access	R

Bit layout

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RCF	FCF	ECF	-	-	SCIC	SSCC	FBC	-	-	-	-	-	-	SCI	SSC

- **SSC**
  - **1:** SSC Interface running - Shift registers are detected on the SSC interface and the auto initialised sizes are present in parameter #52 "SSC In Auto" and parameter #55 "SSC Out Auto".
  - **0:** SSC Interface stopped/not used - No shift registers are detected on the SSC interface.
- **SCI**
  - **1:** SCI Interface running
  - **0:** SCI Interface stopped/not used - Baud rate configuration error / baud rate not detected
- **FBC**
  - **1:** FB I/O Configuration Fault - I/O data mapping configuration error
  - **0:** FB I/O Configuration OK
- **SSCC**
  - **1:** SSC I/O Configuration Fault - I/O data mapping configuration error
  - **0:** SSC I/O Configuration OK
- **SCIC**
  - **1:** SCI I/O Configuration Fault - I/O data mapping configuration error
  - **0:** SCI I/O Configuration OK
- **ECF**
  - **1:** EEPROM memory check failed
  - **0:** EEPROM memory check OK
- **FCF**
  - **1:** FLASH memory check failed
  - **0:** FLASH memory check OK
- **RCF**
  - **1:** RAM check failed
  - **0:** RAM check OK

Module Type (Parameter #3)

This parameter can be used to determine what type of module that is mounted.

All standard AnyBus-IC modules has the same module type value.

Parameter number	3
Modbus Address	0x5003
Default value	-
Range	0x0000h - 0xFFFF
Size	2 bytes
Stored in NV RAM	Yes
Access	R

Values

- **0x0301 - Standard AnyBus-IC**

Fieldbus Type (Parameter #4)

This parameter can be used to determine the type of fieldbus interface that is provided by the module.

Parameter number	4
Modbus Address	0x5004
Default value	-
Range	0x0000 - 0xFFFF
Size	2 bytes
Stored in NV RAM	Yes
Access	R

Values

- **0x0001 - Profibus DP**
- **0x0025 - DeviceNet**
- **0x0082 - EtherNet/IP**

LED State (Parameter #7)

The state of the SSC LED register can be read using this parameter.

Note: This parameter is updated with the LED state, even if the SSC LED register is not used.

Parameter number	7
Modbus Address	0x5007
Default value	-
Range	Bit field
Size	2 bytes
Stored in NV RAM	No
Access	R

Bit layout

b15 b14	b13 b12	b11 b10	b9 b8	b7 b6	b5 b4	b3 b2	b1 b0
LED 8	LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1

LED1 is indicating the state of the least significant bit of the SSC LED register, LED2 is indicating the state of the second least bit and so on.

b(x)	b(x-1)	Description
0	0	LED is turned off
0	1	LED is turned on
1	0	LED is flashing 1Hz
1	1	Reserved (LED test - Fieldbus specific - See fieldbus appendix)

Configuration Bits (Parameter #8)

This parameter determines which values that will be valid for different initialisation parameters.

Parameter number	8
Modbus Address	0x5008
Default value	0x0000
Range	Bit field
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

Bit layout

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	-	-	-	-	FBNA	BR	NA	FBLP	FBNP	SSCO	SSCI

- **SSCI**

  1: Configured SSC input size is used. See parameter #51 "SSC In Config"

  0: Automatically initialised SSC input size is used. See parameter #52 "SSC In Auto"

- **SSCO**

  1: Configured SSC output size is used. See parameter #54 "SSC Out Config"

  0: Automatically initialised SSC output size is used. See parameter #55 "SSC Out Auto"

- **FBNP**

  (If there are no input shift registers mounted, this bit is automatically set to 1).

  1: Fieldbus Specific Input register is not present on the SSC interface

  0: Fieldbus Specific Input register is present on the SSC interface

- **FBLP**

  (If there are no output shift registers mounted, this bit is automatically set to 1)

  1: Fieldbus Specific Output register is not present on the SSC interface

  0: Fieldbus Specific Output register is present on the SSC interface

- **NA**[1]

  (If there are no output shift registers mounted, this bit is automatically set to 1).

  1: Node address determined by fieldbus specific node address parameters[2]

  0: Node address determined by the Fieldbus Specific Input register.

- **BR**

  (If BCD-coded switches are used, this bit will automatically be set to 1).

  1: Fieldbus baudrate is set using fieldbus specific baudrate parameter[2]

  0: Fieldbus baudrate is set via fieldbus, or via switches on the fieldbus specific input register.

- **FBNA**

  This bit determines the behaviour on systems where the node address can be received from the fieldbus. On fieldbus systems that does not feature this functionality, this bit has no function.

  1: Node address is received from the fieldbus. The value of the NA-bit will be ignored.

  0: Node address source is determined by the NA-bit, see above.

1. If the FBNA bit is set, the value of the NA bit will be ignored.
2. Consult each separate fieldbus appendix for further information.

Switch Coding (Parameter #9)

If the Fieldbus Specific Input register is enabled, this parameter determines how the value of the switches should be interpreted by the module. The default value of this parameter is fieldbus dependant, i.e. there is no guarantee that the same type of switch is used by default in another version of the AnyBus-IC.

Parameter number	9
Modbus Address	0x5009
Default value	Fieldbus dependant, see separate fieldbus appendix.
Range	0x00 - 0x01
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00 - BCD-coded switch**

  The BCD-code (**B**inary **C**oded **D**ecimal) represent each decimal digit (0-9) using four bits. Since the BCD-code only contains the decimal digits 0-9, the bit-combinations 1010, 1011, 1100, 1101, 1110 and 1111 are forbidden in the BCD-code. The table below shows all valid bit-combinations in the BCD-code.

  With the BCD-code it is possible to set values from 0-99 on the Fieldbus Specific Input register.

Decimal	BCD-Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

  Example

  The example below shows the representation of the decimal value 63 in BCD-code.

Decimal	BCD-Code	
63	0110	0011

- **0x01 - Binary switch**

  The switch represents the data in a binary format. In this mode it is possible to set values from 0-255 on the Fieldbus Specific Input register.

  Example

  The example below shows the representation of the decimal value 63 in binary code.

Decimal	Binary code
63	00111111

Offline Action Config (Parameter #10)

When the fieldbus goes from on-line to off-line or from idle to off-line, the fieldbus outputs can be configured to behave in different ways.

Note: The module must have been on-line once for the offline action to take affect.

Parameter number	10
Modbus Address	0x500A
Default value	0x00
Range	0x00 - 0x02
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00 - Clear**

  Fieldbus outputs are cleared when the fieldbus goes off-line.

- **0x01 - Freeze**

  Fieldbus outputs freeze in the state it has when the fieldbus goes off-line.

- **0x02 - Fault values**

  Fault values configured in parameter #27 "FB Fault Values" are copied to the fieldbus outputs when the fieldbus goes off-line.

Idle Action Config (Parameter #11)

When the fieldbus goes from on-line to idle or from off-line to idle, the fieldbus outputs can be configured to behave in different ways.

Note: The AnyBus-IC must have been on-line once for the idle action to take affect.

Parameter number	11
Modbus Address	0x500B
Default value	0x00
Range	0x00 - 0x02
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00 - Clear**

  Fieldbus outputs are cleared when the fieldbus goes to idle.

- **0x01 - Freeze**

  Fieldbus outputs freeze in the state it has when the fieldbus goes to idle.

- **0x02 - Fault values**

  Fault values configured in parameter #27 “FB Fault Values” are copied to the fieldbus outputs when the fieldbus goes to idle.

Interrupt Config (Parameter #12)

This parameter defines what events that are allowed to trigger an interrupt. See also parameter #13 "Interrupt Cause".

Parameter number	12
Modbus Address	0x500C
Default value	0x0001
Range	Bit field
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

Bit layout

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	-	-	-	-	IDLE	RES	DEF	ACYC	FBOFF	FBON	START

- **START**[1]
  - **1:** An interrupt will be generated when the module has started from power-on and is ready to communicate on any of the interfaces.
  - **0:** This event will not cause an interrupt.
- **FBON**
  - **1:** An interrupt will be generated when the fieldbus goes from off-line to on-line.
  - **0:** This event will not cause an interrupt.
- **FBOFF**
  - **1:** An interrupt will be generated when the fieldbus goes from on-line to off-line.
  - **0:** This event will not cause an interrupt.
- **ACYC**[2]
  - **1:** An interrupt will be generated when new acyclic data is received from the fieldbus master.
  - **0:** This event will not cause an interrupt.
- **DEF**[2]
  - **1:** An interrupt will be generated when Set Default is received from the fieldbus master
  - **0:** This event will not cause an interrupt.
- **RES**[2]
  - **1:** An interrupt will be generated when Reset is received from the fieldbus master
  - **0:** This event will not cause an interrupt.
- **IDLE**[2]
  - **1:** An interrupt will be generated when the fieldbus goes from on-line to idle or from off-line to idle.
  - **0:** This event will not cause an interrupt.

1. If automatic baudrate detection is used, the interrupt is not generated until the correct baudrate is detected.
2. This bit is fieldbus dependant, i.e it may not be available on all versions of the AnyBus-IC

Interrupt Cause (Parameter #13)

This parameter indicates the event that has caused an interrupt. It is configured in parameter #12 "Interrupt Config" the events that shall generate an interrupt. The parameter is automatically cleared by the AnyBus-IC when it has been read by the application. See 8-2 "Interrupt (/INT) & Bootloader Enable (BLE)" for more information about the interrupt function.

Parameter number	13
Modbus Address	0x500D
Default value	-
Range	Bit field
Size	2 bytes
Stored in NV RAM	No
Access	R

Bit layout

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	-	-	-	-	IDLE	RES	DEF	ACYC	FBOFF	FBON	START

- **START**

  1: The AnyBus-IC module has started and is ready to communicate on any of the interfaces.
  0: This event has not caused an interrupt.

- **FBON**

  1: A transition from off-line to on-line on the fieldbus has occurred.
  0: This event has not caused an interrupt.

- **FBOFF**

  1: A transition from on-line to off-line on the fieldbus has occurred.
  0: This event has not caused an interrupt.

- **ACYC**

  1: New acyclic data is received from the fieldbus master.
  0: This event has not caused an interrupt.

- **DEF (Fieldbus Dependant)**

  1: Set Default has been received from the fieldbus master
  0: This event has not caused an interrupt.

- **RES (Fieldbus Dependant)**

  1: Reset has been received from the fieldbus master
  0: This event has not caused an interrupt.

- **IDLE (Fieldbus Dependant)**

  1: The fieldbus has gone from on-line to idle or from off-line to idle.
  0: This event has not caused an interrupt.

SCI Rate Config (Parameter #14)

This parameter is used to configure the baud rate of the SCI interface. A reset/power-cycle of the module is necessary in order for any changes to have effect.

Parameter number	14
Modbus Address	0x500E
Default value	0x00
Range	0x00h - 0x05
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00 -** Automatic baud rate detection. (Default setting)
- **0x01 -** 4,8 kbit/s
- **0x02 -** 9,6 kbit/s
- **0x03 -** 19,2 kbit/s
- **0x04 -** 38,4 kbit/s
- **0x05 -** 57,6 kbit/s

SCI Rate Actual (Parameter #15)

This parameter returns is the actual baud rate of the SCI interface.

Parameter number	15
Modbus Address	0x500F
Default value	-
Range	0x00 - 0x05
Size	1 byte
Stored in NV RAM	No
Access	R

Values

- **0x00 -** Baudrate is not set.
- **0x01 -** 4,8 kbit/s
- **0x02 -** 9,6 kbit/s
- **0x03 -** 19,2 kbit/s
- **0x04 -** 38,4 kbit/s
- **0x05 -** 57,6 kbit/s

SCI Settings Config (Parameter #16)

This parameter is used to configure the port settings of the SCI interface. A reset/power-cycle of the module is necessary in order for any changes to have effect.

This parameter has no effect when automatic baudrate detection is enabled.

Parameter number	16
Modbus Address	0x5010
Default value	0x00
Range	Bit field
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Bit layout

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	-	PAR1	PAR2

- **PAR2**
  1: Enable Parity
  0: Disable Parity
- **PAR1**
  (This bit has no effect if parity is disabled, see above)
  1: Odd Parity
  0: Even Parity

SCI Settings Actual (Parameter #17)

This parameter returns the actual port settings for the SCI interface. If automatic baudrate detection is enabled, the detected port settings are present in this parameter.

Parameter number	17
Modbus Address	0x5011
Default value	-
Range	Bit field
Size	1 byte
Stored in NV RAM	No
Access	R

Bit layout

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	-	PAR1	PAR2

- **PAR2**

  1: Enable Parity

  0: Disable Parity

- **PAR1**

  (This bit has no effect if parity is disabled, see above)

  1: Odd Parity

  0: Even Parity

MIF Rate Config (Parameter #18)

This parameter is used to configure the baud rate of the MIF interface. A reset/power-cycle of the module is necessary in order for any changes to have effect.

Note: Automatic baudrate detection is not supported on this interface.

Parameter number	18
Modbus Address	0x5012
Default value	0x04
Range	0x01 - 0x05
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x01 -** 4,8 kbit/s
- **0x02 -** 9,6 kbit/s
- **0x03 -** 19,2 kbit/s
- **0x04 -** 38,4 kbit/s (Default setting)
- **0x05 -** 57,6 kbit/s

MIF Rate Actual (Parameter #19)

This parameter returns the actual baud rate settings for the MIF interface.

Parameter number	19
Modbus Address	0x5013
Default value	-
Range	0x01 - 0x05
Size	1 byte
Stored in NV RAM	No
Access	R

Values

- **0x01 -** 4,8 kbit/s
- **0x02 -** 9,6 kbit/s
- **0x03 -** 19,2 kbit/s
- **0x04 -** 38,4 kbit/s
- **0x05 -** 57,6 kbit/s

MIF Settings Config (Parameter #20)

This parameter is used to configure the port settings of the MIF interface. A reset/power-cycle of the module is necessary in order for any changes to have effect.

Parameter number	20
Modbus Address	0x5014
Default value	0x00
Range	Bit field
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Bit layout

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	STOP	PAR1	PAR2

- **PAR2**

  1: Enable parity
  0: Disable parity

- **PAR1**

  (If parity is disabled (PAR2=0) this bit has no effect)

  1: Odd Parity
  0: Even Parity

- **STOP**

  1: 2 stop bits are used
  0: 1 stop bit is used

MIF Settings Actual (Parameter #21)

This parameter returns the actual port settings for the MIF interface.

Parameter number	21
Modbus Address	0x5015
Default value	-
Range	Bit field
Size	1 byte
Stored in NV RAM	No
Access	R

Bit layout

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	STOP	PAR1	PAR2

- **PAR2**
  1: Parity is enabled
  0: Parity is disabled
- **PAR1**
  (If parity is disabled (PAR2=0) this bit has no effect)
  1: Odd Parity
  0: Even Parity
- **STOP**
  1: 2 stop bits are used
  0: 1 stop bit is used

Modbus RTU Address (Parameter #22)

This parameter is used to configure the Modbus RTU Address used on the SCI interface.

Note: If auto baudrate detection is used, the Modbus RTU address must be 01h.

Parameter number	22
Modbus Address	0x5016
Default value	0x01
Range	0x01 - 0xF7
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Modbus CRC Disable (Parameter #23)

This parameter is used to disable / enable the Modbus CRC. Disabling this will force the module to skip the CRC field in all Modbus messages (both in query and response messages), i.e the CRC field will be completely removed from the message frame. Generally, it is not recommended to disable Modbus CRC checking. Use this function only in very special cases.

Note: This function requires firmware v1.10.03 or higher for AnyBus-IC PDP, and v1.01.10 or higher for AnyBus-IC DEV.

Parameter number	23
Modbus Address	0x5017
Default value	0x00
Range	0x00 - 0x01
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00 - Enable Modbus CRC checking.**

  Resulting Modbus message frame format:

Start				Address	Function	Data	CRC	End			
char	char	char		8bits	8bits	n*8bits	16bits	char	char	char	

- **0x01 - Disable Modbus CRC checking.**

  Resulting Modbus message frame format:

Start				Address	Function	Data	End			
char	char	char		8bits	8bits	n*8bits	char	char	char	

FB Fault Values (Parameter #27)

This parameter holds fault values that can be copied to the fieldbus output area. See parameter #10 “Offline Action” and parameter #11 “Idle Action” for more information.

Parameter number	27
Modbus Address	0x501B - 0x5032
Default value	0x00
Range	0x00 - 0xFF
Size	48 bytes
Stored in NV RAM	Yes
Access	R/W

I/O Parameters

Each data exchange interface features it's own set of I/O parameters. These are used to determine how the data should be mapped to the other data exchange interfaces.

These parameters must be configured by the application during initialisation (i.e before Module Mode) is set to 0x0001). Any changes made to these parameters requires a re-start (i.e re-initialisation) to have effect.

#	R/W	Name	Size	Default value	Modbus Address
40	R/W	FB Byte Order	1 byte	0x00	0x6000
41	R/W	FB Out Config	2 bytes	0x0000	0x6001
42	R	FB Out Actual	2 bytes	-	0x6002
43	R	FB In Actual	2 bytes	-	0x6003
44	R/W	FB In SSC Offset	2 bytes	0x0000	0x6004
45	R/W	FB In SSC Size	2 bytes	0x0000	0x6005
46	R/W	FB In SCI Offset	2 bytes	0x0000	0x6006
47	R/W	FB In SCI Size	2 bytes	0x0000	0x6007
50	R/W	SSC Byte Order	1 byte	0x00	0x600A
51	R/W	SSC In Config	2 bytes	0x0000	0x600B
52	R	SSC In Auto	2 bytes	-	0x600C
53	R	SSC In Actual	2 bytes	-	0x600D
54	R/W	SSC Out Config	2 bytes	0x0000	0x600E
55	R	SSC Out Auto	2 bytes	-	0x600F
56	R	SSC Out Actual	2 bytes	-	0x6010
57	R/W	SSC Out FB Offset	2 bytes	0x0000	0x6011
58	R/W	SSC Out FB Size	2 bytes	0x0000	0x6012
59	R/W	SSC Out SCI Offset	2 bytes	0x0000	0x6013
60	R/W	SSC Out SCI Size	2 bytes	0x0000	0x6014
63	R/W	SCI Byte Order	1 byte	0x00	0x6017
64	R/W	SCI In Config	2 bytes	0x0000	0x6018
65	R	SCI In Actual	2 bytes	-	0x6019
66	R	SCI Out Actual	2 bytes	-	0x601A
67	R/W	SCI Out FB Offset	2 bytes	0x0000	0x601B
68	R/W	SCI Out FB Size	2 bytes	0x0000	0x601C
69	R/W	SCI Out SSC Offset	2 bytes	0x0000	0x601D
70	R/W	SCI Out SSC Size	2 bytes	0x0000	0x601E

FB Byte Order (Parameter #40)

This parameter determines if the bytes in the fieldbus I/O area shall be byte swapped or not relative to the other I/O areas.

Note that in order for this function to work properly, the I/O length must be a multiple of 16 bits / 2 bytes.

Example

If data is mapped from the "FB Output area" to the "SSC Output area", parameter #40 "FB Byte Order" is 0x00, and parameter #50 "SSC Byte Order" is 0x01, the data written to the "FB Output area" will be swapped when it reaches the "SSC Output area". If the byte order parameters has the same value for both areas, no swap will be made.

Note: Both the FB Input area and the FB Output area is affected by this parameter.

Parameter number	40
Modbus Address	0x6000
Default value	0x00
Range	0x00 - 0x01
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00**

  Do not swap bytes in the fieldbus I/O area.

- **0x01**

  Swap bytes in the fieldbus I/O area.

FB Out Config (Parameter #41)

This parameter configures the size of the FB Out area. The size is specified in bytes.

Parameter number	41
Modbus Address	0x6001
Default value	0x0000
Range	0x0000 - 0x0030h
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

FB Out Actual (Parameter #42)

This parameter holds the actual size of the FB Out area after initialisation. The size is specified in bytes.

Parameter number	42
Modbus Address	0x6002
Modbus Address	
Default value	-
Range	0x0000 - 0x0030
Size	2 bytes
Stored in NV RAM	No
Access	R

FB In Actual (Parameter #43)

This parameter holds the actual size of the FB In area after initialisation. The value of this parameter is calculated using parameter #45 “FB In SSC Size” and parameter #47 “FB In SCI Size”. The size is specified in bytes.

Parameter number	43
Modbus Address	0x6003
Default value	-
Range	0x0000 - 0x0030
Size	2 bytes
Stored in NV RAM	No
Access	R

FB In SSC Offset (Parameter #44)

This parameter is used to set the source location for the SSC Input -> Fieldbus Input mapping.

Parameter number	44
Modbus Address	0x6004
Default value	0x0000
Range	0x0000h - 0x000F
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

FB In SSC Size (Parameter #45)

This parameter is used to specify how many bytes that will be mapped from the SSC Input area to the Fieldbus Input area.

Parameter number	45
Modbus Address	0x6005
Default value	0x0000
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

FB In SCI Offset (Parameter #46)

This parameter is used to set the source location for the SCI Input -> Fieldbus Input mapping.

Parameter number	46
Modbus Address	0x6006
Default value	0x0000
Range	0x0000 - 0x001F
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

FB In SCI Size (Parameter #47)

This parameter is used to specify how many bytes that will be mapped from the SCI Input area to the Fieldbus Input area.

Parameter number	47
Modbus Address	0x6007
Default value	0x0000
Range	0x0000 - 0x0020
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SSC Byte Order (Parameter #50)

This parameter determines if the bytes in the SSC I/O area shall be swapped or not relative to the other I/O areas.

Example

If data is mapped from the "SSC Input area" to the "FB Input area", parameter #50 "SSC Byte Order" is 0x00, and parameter #40 "FB Byte Order" is 0x01, the data written to the "SSC Input area" will be swapped when it reaches the "FB Input area". If the byte order parameters has the same value for both areas, no swap will be made.

Note that in order for this function to work properly, the I/O length must be a multiple of 16 bits / 2 bytes.

Note: The whole SSC I/O area is affected by the state of this parameter no matter of the contents.

Parameter number	50
Modbus Address	0x600A
Default value	0x00
Range	0x00 -0x 01
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00**

  Do not swap bytes in the SSC I/O area.

- **0x01**

  Swap bytes in the SSC I/O area.

SSC In Config (Parameter #51)

This parameter is used to configure the total size of the SSC In area (the FB specific input byte is not included in this size). The size is specified in bytes.

In order for the value of this parameter to be valid after initialisation, the SSCI-bit in parameter #8 "Configuration Bits" must be set.

Parameter number	51
Modbus Address	0x600B
Default value	0x0000
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SSC In Auto (Parameter #52)

This parameter returns the automatically configured size of the SSC In area (the FB specific input byte is not included in this size). The size is specified in bytes.

In order for the value of this parameter to be valid after initialisation, the SSCI-bit in parameter #8 "Configuration Bits" must be cleared.

Parameter number	52
Modbus Address	0x600C
Default value	-
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	No
Access	R

SSC In Actual (Parameter #53)

This parameter returns the actual size of the SSC In area after initialisation (the Fieldbus Specific Input byte is not included in this size). The size is specified in bytes.

Parameter number	53
Modbus Address	0x600D
Default value	-
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	No
Access	R

SSC Out Config (Parameter #54)

This parameter is used to configure the total size of the SSC Out area (the FB specific output byte is not included in this size). The size is specified in bytes.

In order for the value of this parameter to be valid after initialisation, the SSCO-bit in parameter #8 "Configuration Bits" must be set.

Parameter number	54
Modbus Address	0x600E
Default value	0x0000
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SSC Out Auto (Parameter #55)

This parameter returns the automatically configured size of the SSC Out area (the FB specific output byte is not included in this size). The size is specified in bytes.

In order for the value of this parameter to be valid after initialisation, the SSCO-bit in parameter #8 "Configuration Bits" must be cleared.

Parameter number	55
Modbus Address	0x600F
Default value	-
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	No
Access	R

SSC Out Actual (Parameter #56)

This parameter returns the actual size of the SSC Out area after initialisation (the FB specific output byte is not included in this size). The size is specified in bytes.

Parameter number	56
Modbus Address	0x6010
Default value	-
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	No
Access	R

SSC Out FB Offset (Parameter #57)

This parameter is used to set the source location for the Fieldbus Output -> SSC output mapping.

Parameter number	57
Modbus Address	0x6011
Default value	0x0000
Range	0x0000 - 0x002F
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SSC Out FB Size (Parameter #58)

This parameter is used to specify how many bytes that will be mapped from the Fieldbus Output area to the SSC Output area

Parameter number	58
Modbus Address	0x6012
Default value	0x0000
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SSC Out SCI Offset (Parameter #59)

This parameter is used to set the source location for the SCI Input -> SSC output mapping.

Parameter number	59
Modbus Address	0x6013
Default value	0x0000
Range	0x0000 - 0x001F
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SSC Out SCI Size (Parameter #60)

This parameter is used to specify how many bytes that will be mapped from the SCI Input area to the SSC Output area

Parameter number	60
Modbus Address	0x6014
Default value	0x0000
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SCI Byte Order (Parameter #63)

This parameter determines if the bytes in the SCI I/O area shall be swapped or not relative to the other I/O areas.

Example:

If data is mapped from the "SCI Input area" to the "FB Input area", parameter #63 "SCI Byte Order" is 0x00, and parameter #40 "FB Byte Order" is 0x01, the data written to the "SCI Input area" will be swapped when it reaches the "FB Input area". If the byte order parameters has the same value for both areas, no swap will be made.

Note that in order for this function to work properly, the I/O length must be a multiple of 16 bits / 2 bytes.

Note: The whole SCI I/O area is affected by the state of this parameter no matter of the contents.

Parameter number	63
Modbus Address	0x6017
Default value	0x00
Range	0x00 - 0x01
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Values

- **0x00**

  Do not swap bytes in the SCI I/O area.

- **0x01**

  Swap bytes in the SCI I/O area.

SCI In Config (Parameter #64)

This parameter configures the size of the SCI In area. The size is specified in bytes.

Parameter number	64
Modbus Address	0x6018
Default value	0x0000
Range	0x0000 - 0x0020
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SCI In Actual (Parameter #65)

This parameter holds the actual size of the SCI In area after initialisation. The size is specified in bytes.

Parameter number	65
Modbus Address	0x6019
Default value	-
Range	0x0000 - 0x0020
Size	2 bytes
Stored in NV RAM	No
Access	R

SCI Out Actual (Parameter #66)

This parameter holds the actual size of the SCI Out area after initialisation. The value of this parameter is calculated using parameter #68 “SCI Out FB Size” and parameter #70 “SCI Out SSC Size”. The size is specified in bytes.

Parameter number	66
Modbus Address	0x601A
Default value	-
Range	0x0000 - 0x001F
Size	2 bytes
Stored in NV RAM	No
Access	R

SCI Out FB Offset (Parameter #67)

This parameter is used to set the source location for the Fieldbus Output -> SCI output mapping.

Parameter number	67
Modbus Address	0x601B
Default value	0x0000
Range	0x0000 - 0x002F
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SCI Out FB Size (Parameter #68)

This parameter is used to specify how many bytes that will be mapped from the Fieldbus Output area to the SCI Output area.

Parameter number	68
Modbus Address	0x601C
Default value	0x0000
Range	0x0000 - 0x0020
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SCI Out SSC Offset (Parameter #69)

This parameter is used to set the source location for the SSC Input -> SCI output mapping.

Parameter number	69
Modbus Address	0x601D
Default value	0x0000
Range	0x0000 - 0x000F
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

SCI Out SSC Size (Parameter #70)

This parameter is used to specify how many bytes that will be mapped from the SSC Input area to the SCI Output area.

Parameter number	70
Modbus Address	0x601E
Default value	0x0000
Range	0x0000 - 0x0010
Size	2 bytes
Stored in NV RAM	Yes
Access	R/W

Fieldbus Specific Parameters

See specific fieldbus appendix for information about fieldbus specific parameters.

Initialisation

The AnyBus-IC can be initialised in three different modes; Automatic, Normal and Fieldbus Specific Initialisation. Each mode is intended for different configurations. This chapter will give examples of how to use the AnyBus-IC in different modes with different configurations.

- **Normal Initialisation**

  In this mode, the SCI is enabled and available for I/O data exchange and / or parameter configuration.

- **Automatic Initialisation / Stand Alone mode**

  In this mode, the SCI interface is disabled. Parameters must be set using the MIF interface.

- **Fieldbus Specific Initialisation**

  Some fieldbus specific features are available in this mode only.

Note: In the examples in this chapter assumes that all parameters has default values before initialisation starts. To reset the module to it's factory default settings, set parameter #1 ("Module Mode") to 0004h.

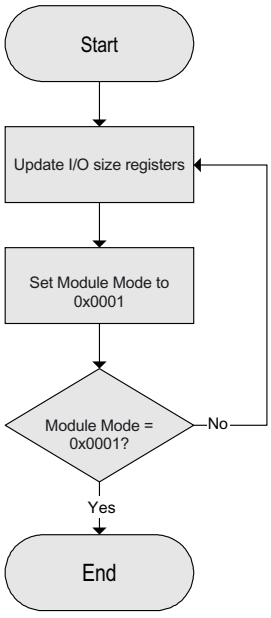
Normal Initialisation

When the AnyBus-IC is connected via the SCI interface to an application with a micro processor, the normal initialisation method is recommended. The AnyBus-IC is initialised with the normal initialisation by writing the "Normal Initialisation" value to parameter #1 "Module Mode".

Initialisation Sequence

The following steps should be followed when initialising the module using the SCI interface.

1. Start-up
2. If necessary, perform the automatic baudrate detection sequence for the SCI interface, see 3-1 "Baud rate".
3. Configure the I/O data sizes using the I/O parameters
4. Set parameter #1 ("Module Mode") to 0x0001 (Normal Mode)
5. Check if initialisation was successful
6. End of initialisation



Note: If required, the procedure above can also be performed using the MIF interface. However, since this interface is designed for debugging and configuration purposes, this initialisation method should only be used when evaluating the various functions of the module.

Automatic Initialisation (Stand alone)

By connecting the SCI DE [AUTO] signal (pin 29) to ground (GND), the module will be automatically initialised and run "stand alone". This initialisation method should be used in non-intelligent applications such as valve terminals and modular I/O devices.

In this mode, the SCI interface is disabled, and parameters can only be set using the MIF interface, see 4-1 "MIF Interface". The module will detect the SSC sizes automatically and map all data between the SSC interface and the fieldbus.

Note: If parameter #8 "Configuration Bits" is set to a value different than zero (0000h), the automatic initialisation may not work as expected (e.g. if the SSCI bit is 1, the SSC Input data size will be taken from parameter #51 "SSC In Config" instead of the automatically detected size). To reset the module to it's factory default settings, set parameter #1 ("Module Mode") to 0004h.

Fieldbus Specific Initialisation

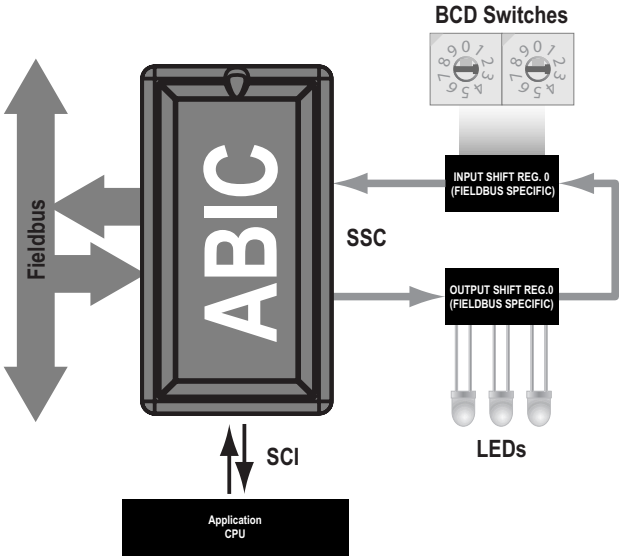
For advanced implementations, some versions of the AnyBus-IC supports fieldbus specific initialisation to be able to use all functions on a specific fieldbus. The fieldbus specific initialisation is started by writing the "Fieldbus Specific Initialisation" value to parameter #1 "Module Mode".

For information about the fieldbus specific initialisation, consult the separate fieldbus appendix.

Initialisation Examples, Normal Mode

Switches and LEDs on SSC, Data Exchange via SCI

- SCI interface connected to the application micro processor
- Automatic baudrate detection on SCI interface
- All I/O mapped between the SCI data area and the FB data area. (In this case 8 bytes in and 8 bytes out are used)
- LEDs on SSC interface used
- Switch on SSC interface used (BCD-coded)

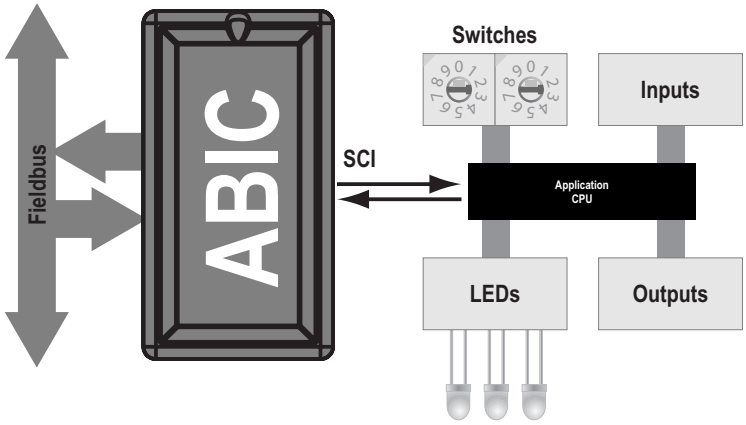


Parameter values for this initialisation

- #8 “Configuration Bits” - SSCI=1, SSCO=1
- #9 “Switch Coding” - 0x00
- #14 “SCI Rate Config” - 0x00 (default)
- #41 “FB Out Config” - 0x0008
- #45 “FB In SSC Size” - 0x0000 (default)
- #46 “FB In SCI Offset” - 0x0000 (default)
- #47 “FB In SCI Size” - 0x0008
- #51 “SSC In Config” - 0x0000 (default)
- #54 “SSC Out Config” - 0x0000 (default)
- #64 “SCI In Config” - 0x0008
- #67 “SCI Out FB Offset” - 0x0000 (default)
- #68 “SCI Out FB Size” - 0x0008
- #70 “SCI Out SSC Size” - 0x0000 (default)

Switches, LEDs and Data Exchange via SCI

- SCI interface connected to the application micro processor.
  (In this case 8 bytes in and 8 bytes out are used)
- LEDs and outputs handled by application
- Switches and inputs handled by application

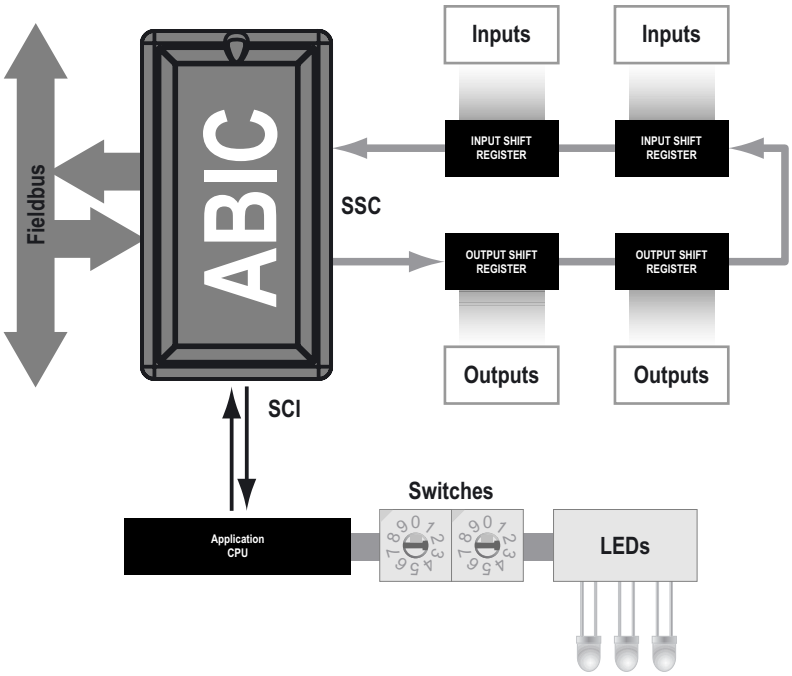


Parameter values for this initialisation

- #8 “Configuration Bits” - FBNP=1, FBLP =1, SSCI=1, SSCO=1
- #9 “Switch Coding” - 0x00
- #14 “SCI Rate Config” - 0x00 (default)
- #41 “FB Out Config” - 0x0008
- #45 “FB In SSC Size” - 0x0000 (default)
- #46 “FB In SCI Offset” - 0x0000 (default)
- #47 “FB In SCI Size” - 0x0008
- #51 “SSC In Config” - 0x0000 (default)
- #54 “SSC Out Config” - 0x0000 (default)
- #64 “SCI In Config” - 0x0008
- #67 “SCI Out FB Offset” - 0x0000 (default)
- #68 “SCI Out FB Size” - 0x0008
- #70 “SCI Out SSC Size” - 0x0000 (default)

Switches and LEDs on SCI, Data Exchange via SSC and SCI

- SCI interface connected to the application micro processor
- LEDs on application
- Switch on application
- Fieldbus output data is mapped to both SCI and SSC interfaces
- I/O data used on the SSC interface (2 byte SSC in / out)
- I/O data used on the SCI interface (6 byte SCI in / out)



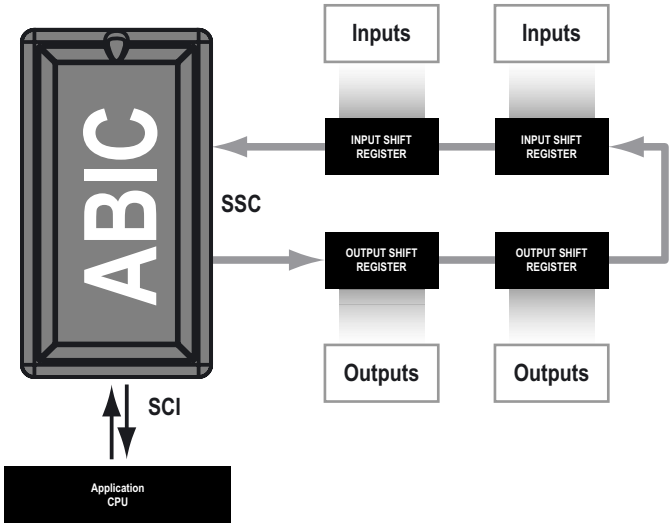
Parameter values for this initialisation

- #8 “Configuration Bits” - FBNP=1, FBLP =1, SSCI=1, SSCO=1
- #14 “SCI Rate Config” - 0x00
- #64 “SCI In Config” - 0x0006
- #67 “SCI Out FB Offset” - 0x0002
- #68 “SCI Out FB Size” - 0x0006
- #41 “FB Out Config” - 0x0008
- #45 “FB In SSC Size” - 0x0002
- #46 “FB In SCI Offset” - 0x0000
- #47 “FB In SCI Size” - 0x0006
- #44 “FB In SSC Offset” - 0x0000
- #51 “SSC In Config” - 0x0002
- #54 “SSC Out Config” - 0x0002
- #57 “SSC Out FB Offset” - 0x0000
- #58 “SSC Out FB Size” - 0x0002

SCI and SSC used for Data Exchange

This example illustrates that it is possible to communicate between the SSC and SCI interfaces only.

- SCI interface connected to the application micro processor
- I/O data used on the SSC interface
- I/O data mapped between SSC data area and SCI data area



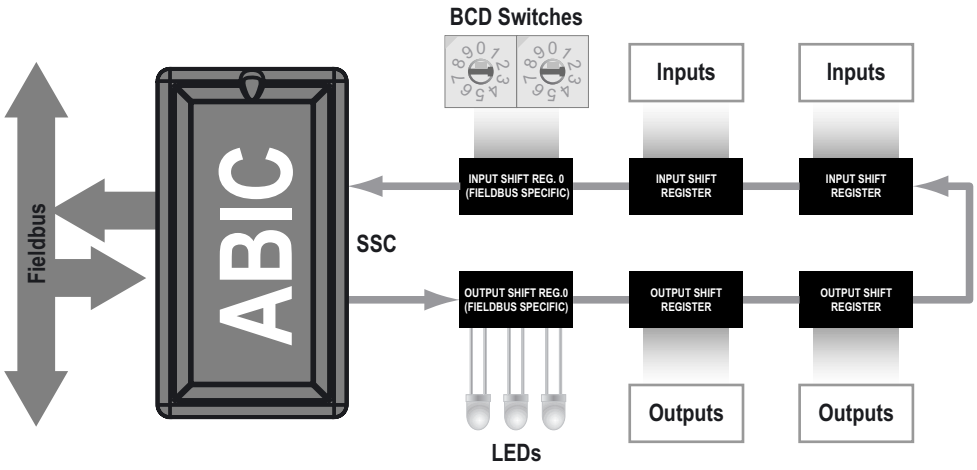
Parameter values for this initialisation

- #8 “Configuration Bits” - FBNP=1, FBLP =1, SSCI=1, SSCO=1
- #14 “SCI Rate Config” - 0x00
- #64 “SCI In Config” - 0x0002
- #41 “FB Out Config” - 0x0000
- #45 “FB In SSC Size” - 0x0000
- #46 “FB In SCI Offset” - 0x0000
- #47 “FB In SCI Size” - 0x0000
- #44 “FB In SSC Offset” - 0x0000
- #51 “SSC In Config” - 0x0002
- #54 “SSC Out Config” - 0x0002
- #60 “SSC Out SCI Size” - 0x0002
- #70 “SCI Out SSC Size” - 0x0002

Initialisation Examples, Automatic Mode

Switches and LEDs on SSC

- Automatically initialised SSC sizes used
- All I/O data is mapped between the SSC data area and the FB data area
- LEDs on SSC interface used (Fieldbus Specific output register)
- BCD switches on SSC interface used (Fieldbus Specific input register)



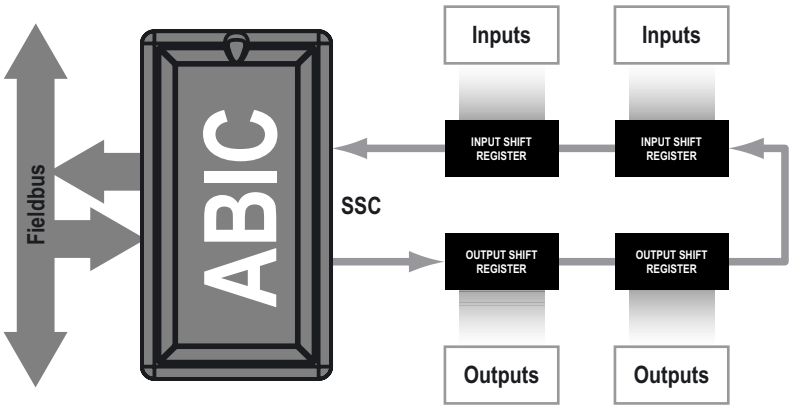
Parameter values for this initialisation

- #8 “Configuration bits” - 00h (default)
- #9 “Switch Coding” - 00h

Pre-configured Node address, no LEDs

In this example it is necessary to use the monitor interface to configure the node address.

- Automatically initialised SSC sizes used
- All I/O mapped between the SSC data area and the FB data area
- Node address pre-configured (Fieldbus specific parameter), i.e. node address switch is not used
- LEDs not used

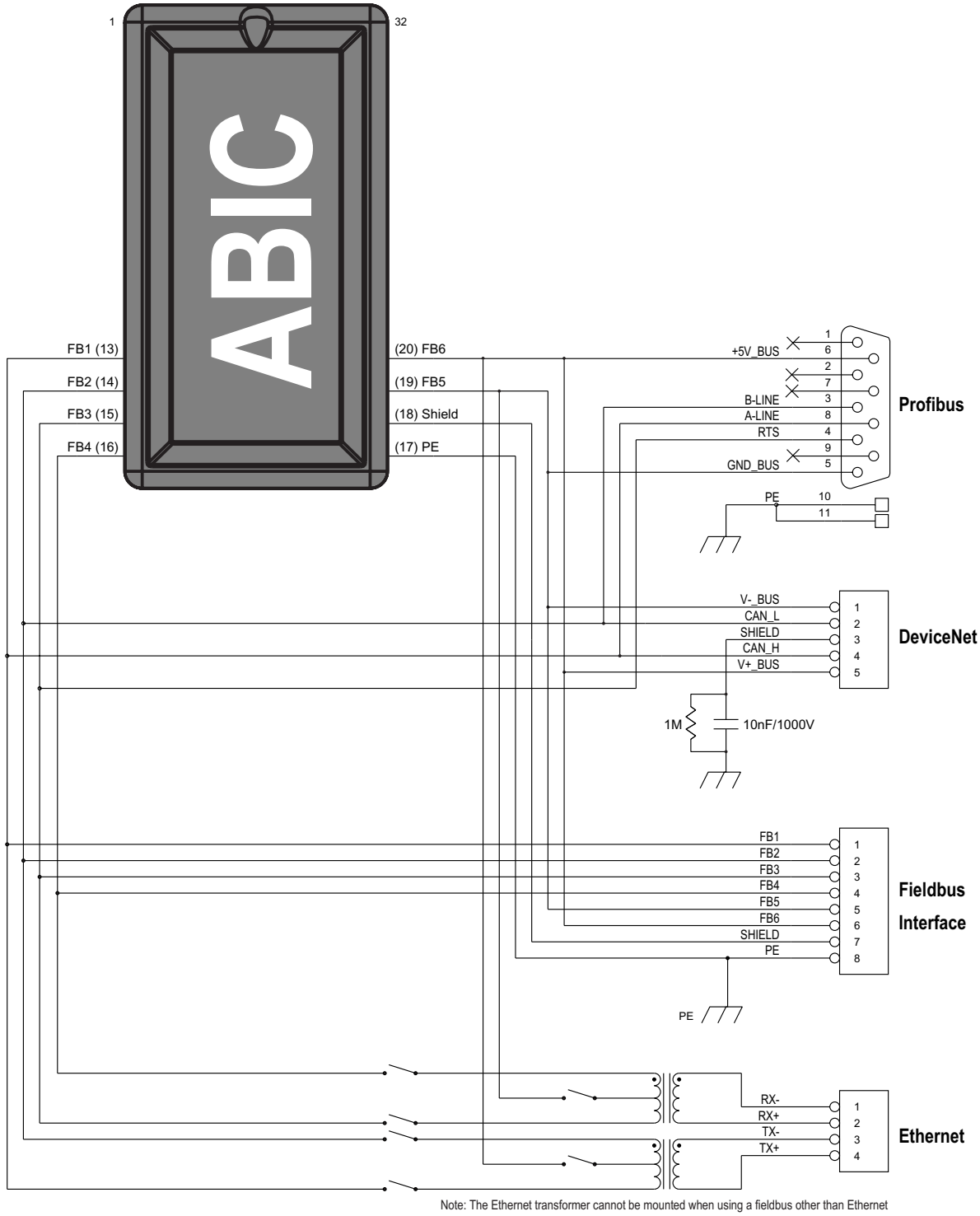


Parameter values for this initialisation

- #8 “Configuration Bits” - FBNP=1, FBLP =1
- Node address configured in fieldbus specific parameter

Miscellaneous

Fieldbus Interface



Note: The Ethernet transformer cannot be mounted when using a fieldbus other than Ethernet

Interrupt (/INT) & Bootloader Enable (BLE)

During power-on, this pin is used as a boot loader enable pin and after power-on, it serves as an interrupt pin.

Warning! Do not connect this pin to ground (GND) during normal operation.

Boot Loader Enable (BLE)

If this pin is connected to ground (GND) during power-on, the module will start in boot loader mode. This allows new firmware to be downloaded via the monitor interface (MIF).

Generally, this function should only be used if wrong data is accidentally downloaded into the on board flash. Normally, firmware upgrades should be performed via the Firmware Upgrade menu on the monitor interface (MIF).

Interrupt (/INT)

During normal operation, this pin acts as an active low interrupt output. Which events that should generate an interrupt can be configured using parameter #12 (“Interrupt Config”).

When an interrupt has occurred (i.e. when the interrupt pin has gone low), the cause of the interrupt can be read from parameter #13 (“Interrupt Cause”). The value of this register holds its value until it has been read by the application. When read, the interrupt is cleared and the interrupt pin goes high again.

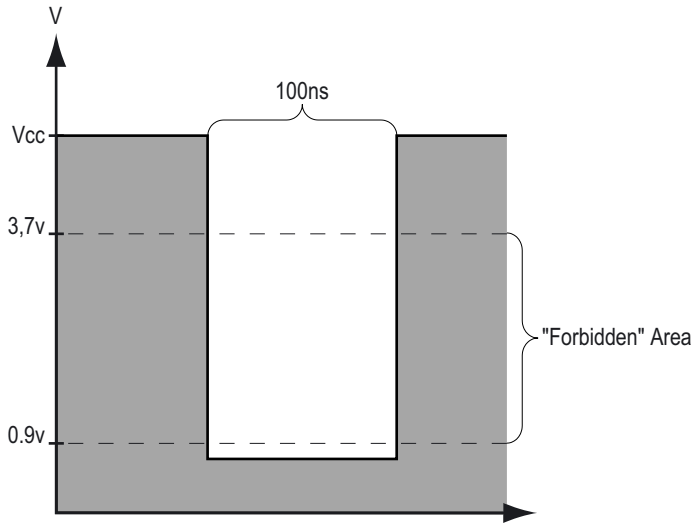
Reset

The /RESET pin is used to trigger a hardware reset of the module. If the reset signal is not used it is recommended to connect it directly to Vcc.

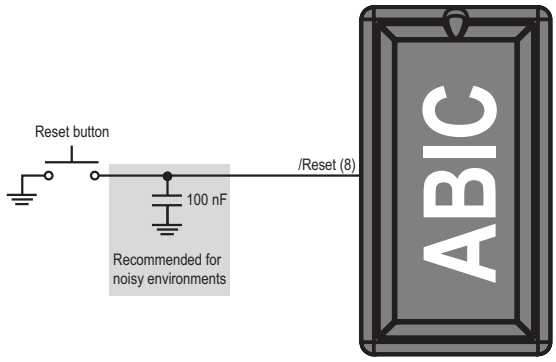
If the device is used in a noisy environment, it is recommended to connect a 100nF capacitor between the reset signal and ground (GND).

Timing

To trigger a hardware reset, a high-to-low transition with a minimum duration of 100ns is necessary. (See below)



Connection Example



Hardware Self Test

The application can instruct the module to perform a hardware self test using parameter #1 “Module Mode”. The result of the test is presented in parameter #2 “Module Status”.

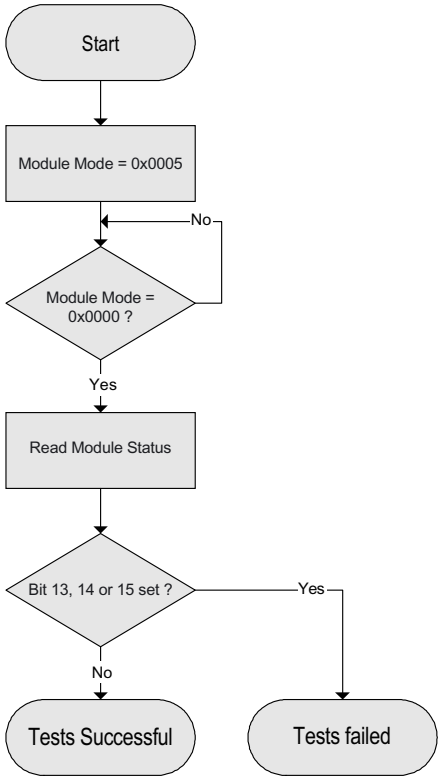
Note: This test must be performed before the module is initialized.

Test Sequence

1. Start the hardware tests by writing 0x0005 to parameter #1 (“Module Mode”)
2. During the test, parameter #1 (“Module Mode”) has the value 0x0005. When tests are finished, the value will be 0x0000.
3. Read parameter #2 (“Module Status”) to get information about the tests.

Test Evaluation

If bit 13, 14 or 15 is set, the corresponding test has failed. If the bit is cleared, the corresponding test passed successfully.



Modbus Protocol

This chapter gives a brief introduction to the Modbus RTU protocol. For more information, refer to the "Modbus Protocol Reference Guide" by Modicon.

Note: All information in this chapter is applicable on the AnyBus-IC, however, the Modbus specification may contain more functionality than described here.

Query / Response

The Modbus RTU protocol is a master / slave based communication protocol, using a Query / Response based messaging scheme. The AnyBus-IC module acts as a slave, and cannot Respond unless it has been addressed by the application using a Query.

Modbus RTU Message Frame Format

The modbus RTU message frame is divided into 6 different fields, see below.

Start				Address	Function	Data	CRC	End			
char	char	char		8bits	8bits	n*8bits	16bits	char	char	char	

- **Start**

  A Modbus RTU message is started with a silent interval of at least 3,5 characters.

- **Address field**

  This is the modbus address configured in parameter #22 "Modbus RTU Address". To broadcast the message to all devices, this field should be set to 0x00.

- **Function Code**

  This field contains the function code. In a Query, the Function Code is used to tell the AnyBus-IC module what kind of action to perform.

  In the Response generated by the module, the Function Code indicates either that everything went well by returning the original function code, an Exception to indicate that some kind of error occurred. In an Exception response, the module returns a code that is equivalent to the original function code with its most significant bit set. The Exception Code is returned in the data field of the response message.

- **Data**

  The data field is used to pass additional information related to the Function Code. For Exception responses, this field is used for the Exception Code, see next page.

- **CRC**

  This field is based on the Cyclical Redundancy Check error checking method. The CRC should be generated by the application. The module recalculates the CRC during upon receiving the message, and compares the calculated value to the actual value in the CRC field. If the two values are not equal, the module will not send a response message to the application, thus generating a timeout. The CRC should be placed with the least significant byte first, followed by the most significant byte. For example code, see Appendix B-1 "CRC Generation, Code Examples".

- **End**

  A Modbus RTU message ends with a silent interval of at least 3,5 characters.

Supported Exception Codes

The following exception codes are supported by the AnyBus-IC.

Code	Name	Meaning
0x01	Illegal Function	The function code in the query is not supported by the AnyBus-IC.
0x02	Illegal Data Address	The data address in the query is not allowed for the AnyBus-IC.
0x03	Illegal Data Value	The data value in the query is not valid.
0x04	Slave Device Failure	An unrecoverable error occurred while the slave was attempting to perform the requested action.

Supported Modbus Functions

The SCI interface of the AnyBus-IC supports the following Modbus functions.

Code	Name	Meaning
0x03	Read Multiple Registers	Reads the contents of a sequence of registers
0x04	Read Input Registers	Reads the contents of a sequence of registers. In the AnyBus-IC this function is the same as the “Read Multiple Registers”.
0x06	Write Single Register	Writes a value to a single register. When using broadcast, the same register in all modules will be affected.
0x10	Write Multiple Registers	Writes a value to a sequence of registers. When using broadcast, the same registers in all modules will be affected.
0x17	Read / Write Registers	Writes a value to a sequence of registers and reads a sequence of registers at the same time.

Read Multiple Registers (0x03)

This function will read the contents of a sequence of registers.

In the AnyBus-IC this function is the same as the "Read Input Registers".

Function Name	'Read Multiple Registers'
Function Code	0x03
Broadcast Supported	No

Example

The example below reads 2 registers in the beginning of the SCI Input data area (Modbus address 0x0000 - 0x0001) from an AnyBus-IC with address 0x01.

Query

Slave Address	0x01
Function Code	0x03
Start Address (High byte)	0x00
Start Address (Low byte)	0x00
Number of registers (High byte)	0x00
Number of registers (Low byte)	0x02
CRC	0xC40B

Response

Slave Address	0x01
Function Code	0x03
Byte Count	0x04
Data (High byte)	0x00
Data (Low byte)	0x00
Data (High byte)	0x00
Data (Low byte)	0x00
CRC	0xFA33

Error Response

Slave Address	0x01
Function Code	0x83
Exception Code	(Exception code, see 9-2 "Supported Exception Codes")
CRC	-

Read Input Registers (0x04)

This function will read the contents of a sequence of registers.

In the AnyBus-IC this function is the same as the "Read Multiple Registers".

Function Name	'Read Input Registers'
Function Code	0x04
Broadcast Supported	No

Example

The example below reads 2 registers in the beginning of the SCI Input data area (Modbus address 0x0000 - 0x0001) from an AnyBus-IC with address 0x01.

Query

Slave Address	0x01
Function Code	0x04
Start Address (High byte)	0x00
Start Address (Low byte)	0x00
Number of registers (High byte)	0x00
Number of registers (Low byte)	0x02
CRC	0x71CB

Response

Slave Address	0x01
Function Code	0x04
Byte Count	0x04
Data (High byte)	0x00
Data (Low byte)	0x00
Data (High byte)	0x00
Data (Low byte)	0x00
CRC	0xFB84

Error Response

Slave Address	0x01
Function Code	0x84
Exception Code	(Exception code, see 9-2 "Supported Exception Codes")
CRC	-

Write Single Register (0x06)

This function writes a value to a single register. When using broadcast, the same register in all modules will be affected.

Function Name	'Write Single Register'
Function Code	0x06
Broadcast Supported	Yes

Example

The example below writes to the beginning of the SCI Input data area (Modbus address 0x0000 - 0x0001) of an AnyBus-IC with address 0x01.

Query

Slave Address	0x01
Function Code	0x06
Start Address (High byte)	0x00
Start Address (Low byte)	0x00
Data (High byte)	0xAA
Data (Low byte)	0xFF
CRC	0xB72A

Response

Slave Address	0x01
Function Code	0x06
Start Address (High byte)	0x00
Start Address (Low byte)	0x00
Data (High byte)	0xAA
Data (Low byte)	0xFF
CRC	0xB72A

Error Response

Slave Address	0x01
Function Code	0x86
Exception Code	(Exception code, see 9-2 "Supported Exception Codes")
CRC	-

Write Multiple Registers (0x10)

This function writes data to a sequence of registers. When using broadcast, the same registers will be affected in all modules.

Function Name	'Write Multiple Registers'
Function Code	0x10
Broadcast Supported	Yes

Example

The example below will write 0xAABB / 0xCCDD to registers 0x0001 / 0x0002. Note that "Number of Registers" is specified in words, while "Byte Count" is specified in bytes.

Query

Slave Address	0x01
Function Code	0x10
Start Address (High byte)	0x00
Start Address (Low byte)	0x01
Number of registers (High byte)	0x00
Number of registers (Low byte)	0x02
Byte Count	0x04
Data (High byte)	0xAA
Data (Low byte)	0xBB
Data (High byte)	0xCC
Data (Low byte)	0xDD
CRC	0xF6C7

Response

Slave Address	0x01
Function Code	0x10
Start Address (High byte)	0x00
Start Address (Low byte)	0x01
Number of registers (High byte)	0x00
Number of registers (Low byte)	0x02
CRC	0x1008

Error Response

Slave Address	0x01
Function Code	0x90
Data (High byte)	(Exception code, see 9-2 "Supported Exception Codes")
CRC	-

Read / Write Registers (0x17)

This function writes and reads to a sequence of registers at the same time.

Function Name	'Read / Write Registers'
Function Code	0x17
Broadcast Supported	No

Example

The example below writes 0xAABB / 0xCCDD to registers 0x0010 and 0x0011, and reads 0x1122/0x3344/0x5566 from registers 0x0002 / 0x0003 / 0x0004.

Query

Slave Address	0x01
Function Code	0x17
Read Start Address (High byte)	0x00
Read Start Address (Low byte)	0x02
No of registers to read (High byte)	0x00
No of registers to read (Low byte)	0x03
Write Start Address (High byte)	0x00
Write Start Address (Low byte)	0x10
No of registers to write (High byte)	0x00
No of registers to write (Low byte)	0x02
Byte Count	0x04
Write Data 1 (High byte)	0xAA
Write Data 1 (Low byte)	0xBB
Write Data 2 (High byte)	0xCC
Write Data 2 (Low byte)	0xDD
CRC	0xAAEA

Response

Slave Address	0x01
Function Code	0x17
Byte Count	0x06
Read Data 1 (High byte)	0x11
Read Data 1 (Low byte)	0x22
Read Data 2 (High byte)	0x33
Read Data 2 (Low byte)	0x44
Read Data 3 (High byte)	0x55
Read Data 3 (Low byte)	0x66
CRC	0x2AE7

Error Response

Slave Address	0x01
Function Code	0x97
Data (High byte)	(Exception code, see 9-2 "Supported Exception Codes")
CRC	-

Object Messaging (0x5B)

Introduction

The modbus object messaging is an enhancement to the standard modbus protocol to be able to make object accesses. Originally, the modbus object messaging function is developed for Modbus/TCP. To fit the AnyBus-IC some changes/additions have been made to the original specification.

The modbus object messaging function must be used in the AnyBus-IC when e.g. acyclic data on the fieldbus shall be mapped to an object in the AnyBus-IC. See fieldbus appendix for information about how to map data via the object messaging function.

Function Name	'Object Messaging'
Function Code	0x5B
Broadcast Supported	No

Message Format

The modbus object messaging is based on dividing the standard data field in modbus into 7 sub-fields according to the figure below.

Message Frame

Address	Function	Sub field, see below	CRC

Query - Sub Field Format

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Data	Stuff byte
8 bits	8 bits	16 bits	16 bits	16 bits	n*16 bits	8 bits

Response - Sub Field Format

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Error Code	Data	Stuff byte
8 bits	8 bits	16 bits	16 bits	16 bits	16 bits	n*16 bits	8 bits

Fragment byte count

This field contains the number of bytes of the current object message (not including itself). The maximum number of bytes is 197.

Note: If a stuff byte is added by the end of the message, it is not included in the Fragment byte count.

Fragment Protocol

This field is used when sending fragmented messages. The bits are described below.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	FSN		-	-	-	LFI	FIPI

- **FIPI - Fragment In Process Indicator**
  1: This message is a fragment of a fragmented message.
  0: This message is not fragmented.
- **LFI - Last Fragment Indicator**
  1: This is the last fragment in a fragmented message.
  0: This not the last fragment in a fragmented message.
- **FSN - Fragment Sequence Number**
  This is a counter counting from 000_2 to 111_2. Each fragment in a fragmented message has a sequential number in this field.

Class ID

ID of the object class associated with the service.

Instance ID

ID of the instance associated with the service.

Error Codes

In the response from the AnyBus-IC, the following error codes may be used.

- **General Errors**

Code	Name	Description
0x0000	Success	
0x0001	Invalid Service Code	The requested service is not implemented or defined for this object.
0x0002	Invalid Service Code Parameter	The parameters required to perform the service are invalid.
0x0003	Invalid Attribute	The specified attribute is not supported in this object.
0x0004	Attribute out of range	Set value is out of range for the attribute.
0x0005	Not valid in this state	The object cannot perform the requested service in it's current state.
0x0006	Fragmentation error	Error in message fragmentation.

- **Manufacturer Specific Errors**

Code	Name	Description
0x0101	Class Not Supported	The class specified are not implemented in this device.
0x0104	Instance Not Supported	The specified instance does not exist.
0x0105	Instance Already Exist	The requested instance to be created already exists.
0x0107	Attribute Not Settable	A request to modify a non-modifiable attribute.
0x0109	Not Enough Data	The message provides less data than was needed for the attribute.
0x010A	Too Much Data	The message provides more data than was needed for the attribute.
0x010C	Resource Unavailable	Resources needed for the object to perform the requested service were unavailable.
0x010D	Device State Conflict	The device's current mode/state prohibits the execution of the requested service.

Service Code

Code	Name	Description
0x0001	Get Attribute	
0x0002	Get Attribute Response	
0x0003	Set Attribute	
0x0004	Set Attribute Response	
0x0005	Create	
0x0006	Create Response	
0x0007	Remove	
0x0008	Remove Response	
0x0009	Reset	
0x000A	Reset Response	
0x000B	Start	
0x000C	Start Response	
0x000D	Stop	
0x000E	Stop Response	
0x000F	Save	
0x0010	Save Response	
0x0011	Restore	
0x0012	Restore Response	
0x0013	Nop	
0x0014	Nop Response	
0x0015 - 0x007F	-	Not used
0x0080 - 0x00FF	-	Class specific services

Data

The first word in the data field is used for Attribute data. If an attribute is not used in the service, this word must be 0x0000.

Stuff Byte

The length of the standard data field in modbus must always be a multiple of 16 bits. If it is not, the stuff byte is added by the end of the fragment to ensure this. The stuff byte does not contain any meaningful data and is not included in the Fragment byte count.

Firmware Upgrade

The firmware of the module can be upgraded using the MIF interface. The new firmware should be transferred to the module using the standard Xmodem protocol.

The module offers two ways of upgrading the firmware, see below.

Standard Firmware Upgrade

This is the "standard" procedure to use when upgrading the firmware of the module. To reach the firmware download menu, select "4 - Firmware Upgrade" in the main menu.

The module will ask you to confirm the firmware download. Press 'Y' to confirm, 'N' to cancel.

```
Do you want to download new firmware (Y/N)? y
```

The module then waits for the terminal program to initiate the transfer. Press Control+X to cancel.

```
Start XMODEM transfer from your terminal program
Hit ^X to cancel the transfer
```

Note: Do not disconnect or turn the power off during firmware transfer. Failure to observe this may case corruption of the information stored in the flash and render the module inoperable.

Firmware Upgrade using Bootloader Switch

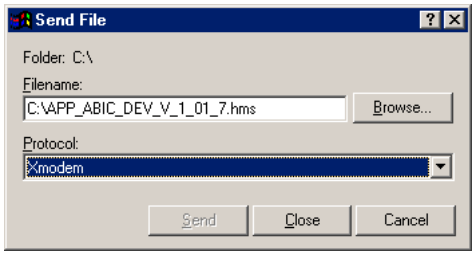
This function should be used only when wrong data has been accidentally downloaded into the flash, or if the transfer for some reason has been interrupted.

To use this mode, pin 26 (/INT [BLE]) should be connected to ground during start up. By doing so, the module will enter a "fail-safe" firmware download mode. The downloading procedure is then identical to the "Standard Firmware Upgrade"-procedure.

Example using the Windows HyperTerminal

Prepare the module for firmware download using one of the methods described above. To transfer the new firmware to the module, follow the steps below.

1. Select the 'Send File' entry in the 'Transfer' menu.
2. Click 'Browse...' in the 'Send File' dialog and select the file containing the new firmware.
3. Select 'Xmodem' in the in the 'Protocol' selection menu
4. Click 'Send' to initiate the firmware download.



CRC Generation, Code Examples

CRC Calculation Basics

The CRC calculation is started by first preloading a 16-bit register to all 1's. Then a process begins of applying successive eight-bit bytes of the message to the current contents of the register.

During generation of the CRC, each eight-bit character is XORed with the current register contents. The result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB is a 1, the register is XORed with a preset, fixed value. If the LSB was a 0, no XOR takes place. This process is repeated until eight shifts have been performed.

After the last (eighth) shift, the above process repeats for the next byte in the message. The final contents of the register, after all bytes of the message have been applied, is the CRC value.

Step by Step:

1. Load a 16-bit register with 0xFFFF (all 1's). Call this the CRC register.
2. XOR the first eight-bit byte of the message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.
3. Shift the CRC register one bit to the right (toward the LSB), zerofilling the MSB. Examine the LSB that was just shifted out from the register.
4. If the LSB is 0, repeat Step 3 (another shift). If the LSB is 1, Exclusive OR the CRC register with the polynomial value 0xA001 (1010 0000 0000 0001).
5. Repeat Steps 3 and 4 until eight shifts have been performed. When this is done, a complete eight-bit byte will have been processed.
6. Repeat Steps 2 … 5 for the next eight-bit byte of the message. Continue doing this until all bytes have been processed.

- **Result**

  The final contents of the CRC register is the CRC value. When the CRC is placed into the message, its upper and lower bytes must be swapped as described in chapter 9-1 “Modbus Protocol” (CRC).

Example 1

The following example calculates the CRC using the method described earlier.

Note: This function performs the swapping of the high/low CRC bytes internally. Therefore the CRC value returned from the function can be directly placed into the message for transmission.

The function returns the CRC as a type UINT16, and takes two arguments:

- **UINT8 *pabMessage;**
  A pointer to the message buffer containing binary data to be used for generating the CRC.
- **UINT16 iLength;**
  The quantity of bytes in the message buffer.

Typedefs

UINT8	= Unsigned 8 bit (e.g. unsigned char)
UINT16	= Unsigned 16 bit (e.g. unsigned short)

Source Code

```
UINT16 GenerateCrc( UINT8* pabMessage, UINT16 iLength )
{

   UINT16 iCRCReg = 0xFFFF;
   UINT8 bByteCount;
   UINT8 bBitCount, bCarryFlag;

   for( bByteCount = 0 ; bByteCount < iLength ; bByteCount++ )
   {

     *((UINT8*)&iCRCReg + 1) = pabMessage[ bByteCount ] ^ ( iCRCReg & 0x00FF );

      for( bBitCount = 0; bBitCount < 8 ; bBitCount++ )
       {

         bCarryFlag = iCRCReg & 0x0001;
         iCRCReg = iCRCReg >> 1;

         if( bCarryFlag != 0 )
         {
            iCRCReg ^= 0xA001;
         }
      }
   }

   return( iCRCReg << 8 | iCRCReg >> 8 );

}/* end GenerateCrc */
```

Example 2

This example uses another approach to calculate the CRC; All of the possible CRC values are pre-loaded into two arrays, which are simply indexed as the function increments through the message buffer. One array contains all of the 256 possible CRC values for the high byte of the 16-bit CRC field, and the other array contains all of the values for the low byte.

Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

Note: This function performs the swapping of the high/low CRC bytes internally. Therefore the CRC value returned from the function can be directly placed into the message for transmission.

The function returns the CRC as a type UINT16, and takes two arguments:

- **UINT8 *pabMessage;**
  A pointer to the message buffer containing binary data to be used for generating the CRC.
- **UINT16 iLength;**
  The quantity of bytes in the message buffer.

Typedefs

UINT8	= Unsigned 8 bit (e.g. unsigned char)
UINT16	= Unsigned 16 bit (e.g. unsigned short)

Source Code

```
UINT16 GenerateCrc( UINT8* pabMessage, UINT16 iLength )
{
   static UINT8 abCrcHi[] =
   {  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
      0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
      0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
      0x00, 0xC1, 0x81, 0x40
   };

   static UINT8 abCrcLo[] =
   {
      0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
      0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
      0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
      0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
      0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
      0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
      0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
      0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
      0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
      0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
      0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
      0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
      0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
      0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
      0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
      0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
      0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
      0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
      0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
      0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
      0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
      0x41, 0x81, 0x80, 0x40
   };

   UINT8 bCrcHi = 0xFF;
   UINT8 bCrcLo = 0xFF;
   UINT16 iIndex;

    while( iLength-- )
    {
        iIndex = bCrcHi ^ *pabMessage++;
        bCrcHi = bCrcLo ^ abCrcHi[iIndex];
        bCrcLo = abCrcLo[iIndex];
    }

    return( bCrcHi << 8 | bCrcLo );

}/* end GenerateCrc */
```

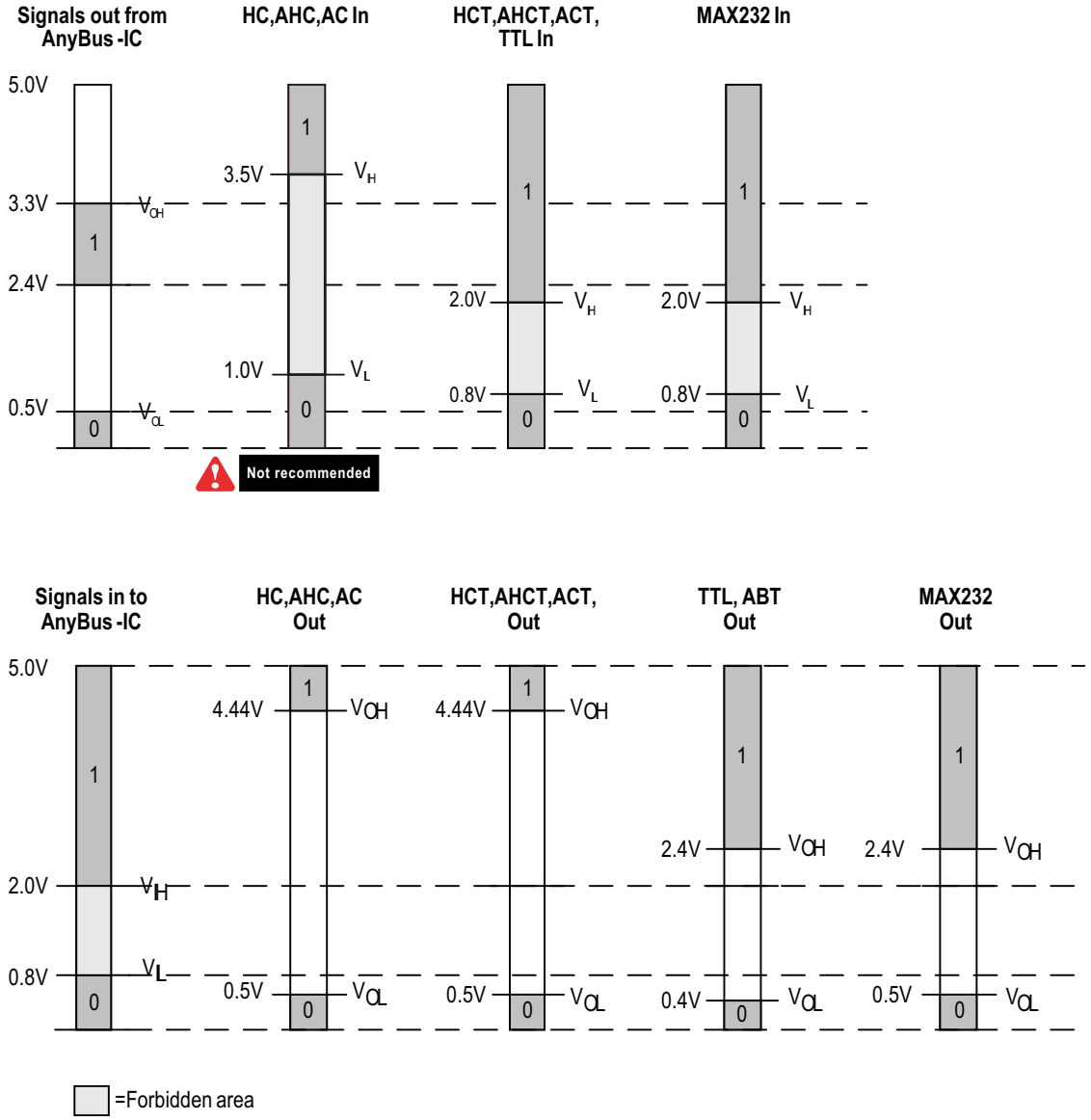
Electrical Characteristics

Signal Levels

It is recommended to use 74HCTxxx / 74ACTxxx type TTL circuits. 74HCxxx, 74ACxxx, and 74AHCxxx type circuits are not recommended, and may not work properly as the "forbidden area" of these types of circuits is not within the boundaries allowed by the AnyBus IC. (See figure below)

For RS232 communication on the SCI / MIF interfaces, it is recommended to the MAX202 transceiver from Maxim.

When connecting current demanding components (e.g leds) directly to the outputs of the SSC shift registers, it is required to use shift registers of sufficient current capacity, e.g. 74ACTxxx.



Note: For exact signal levels, consult the data sheet for the used logic circuit. Signal levels may depend on temperature and supply voltage.

Power

Supply Voltage

The module requires a regulated +5V ±5% DC power supply.

Power Consumption

The maximum power consumption may vary between different models of the AnyBus IC module. Please consult each separate fieldbus appendix.

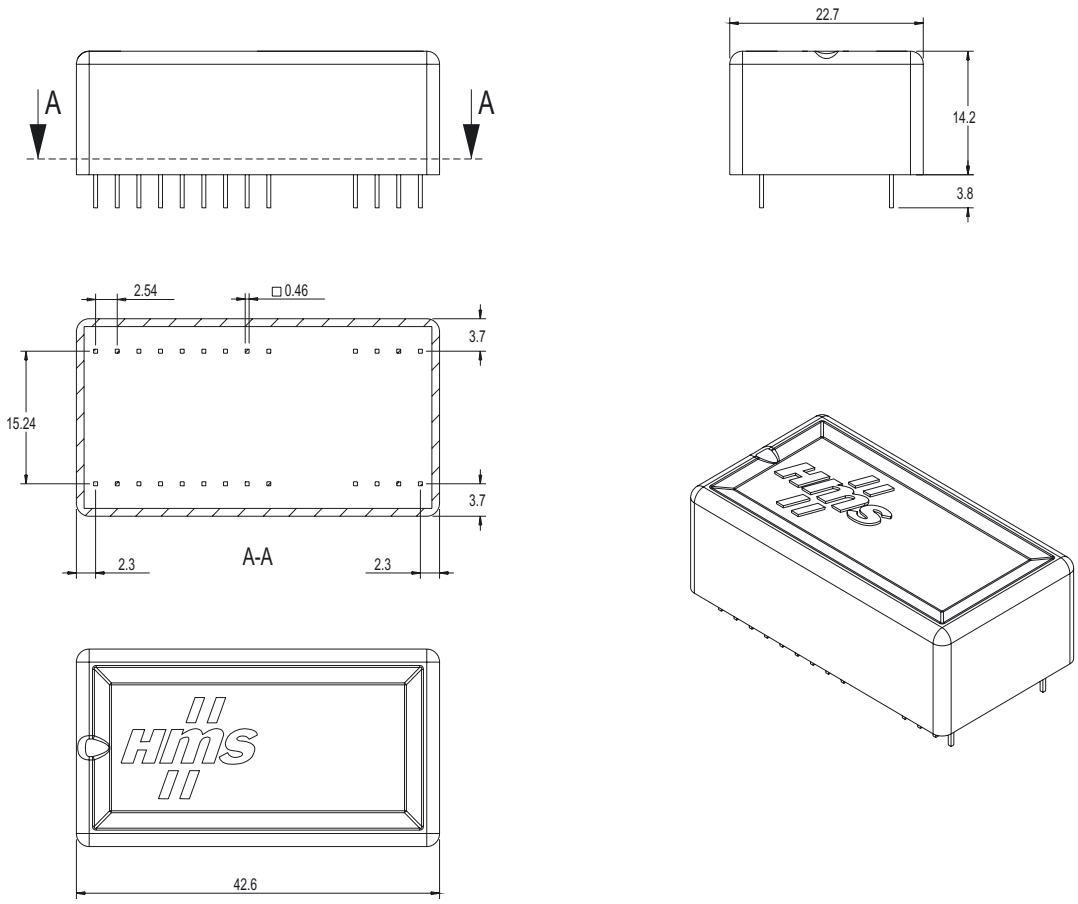
Protective Earth

PE requirements are fieldbus dependant. Please consult the separate fieldbus appendix for details.

Mechanical

Measurements

All measurements are in millimetres, tolerance is ±0,1mm unless otherwise stated.



Mounting

It is recommended to use standard DIL sockets instead of soldering the module directly to the circuit-board.

Depending on shock / vibration conditions, it may be wise to use a cable tie or similar around to ensure electrical contact between the AnyBus-IC and it's socket.

Environmental Specification

Temperature

Operating

-10°C to +70°C

Test performed according to IEC-68-2-1 and IEC 68-2-2.

Non Operating

-25°C to +85°C

Test performed according to IEC-68-2-1 and IEC 68-2-2.

Wave Soldering

Temperature: 320°C

Velocity: 10mm/s

Humidity

The product is designed for a relative humidity of 5 to 95% non-condensing.

Test performed according to IEC 68-2-30.

EMC compliance

Emission

According to EN 50 081-2:1993

Tested per EN 55011:1998, class A, radiated

Immunity

According to EN 61000-6-2:1999

Tested per:

EN 61000-4-2:1995, ±8kV VCP

EN 61000-4-3:1996, 10V/m

EN 61000-4-4:1995, bus ±2kV, power ±2kV

EN 61000-4-5:1995, bus ±1kV, power N/A

EN 61000-4-6:1996, 10Vrms