Anybus® CompactCom™ C40

Implementation Guidelines

DESIGN GUIDE

HMSI-27–271 2.5 en-US ENGLISH

Important User Information

Disclaimer

Table of Contents

Page

1 Preface

1.1 About this Document

This document is intended to provide a good understanding of how to implement an application using the Anybus CompactCom C40. The document only describes the features that are specific to the Anybus CompactCom C40. For general information regarding the Anybus CompactCom 40, see the Anybus CompactCom 40 Software Design Guide, M40 Hardware Design Guide and B40 Hardware Design Guide. For information regarding the functionality of a specific fieldbus or industrial network, see the appropriate Anybus CompactCom 40 Network Guide.

The reader of this document is expected to be familiar with high level software design and industrial network communication systems in general.

For additional information, documentation, support etc., visit the support website at www.anybus.com/support.

1.2 Related Documents

Related documents

Document	Author	Document ID
Anybus CompactCom 40 Software Design Guide	HMS	HMSI-216-125
Anybus CompactCom M40 Hardware Design Guide	HMS	HMSI-216-126
Anybus CompactCom B40–1 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
Anybus CompactCom Network Guides (separate document for each supported fieldbus or network system)	HMS	

1.3 Document Conventions

Numbered lists indicate tasks that should be carried out in sequence:

1. First do this
2. Then do this

Bulleted lists are used for:

- Tasks that can be carried out in any order
- Itemized information

► An action

→ and a result

User interaction elements (buttons etc.) are indicated with bold text.

```
Program code and script examples
```

Cross-reference within this document: *Document Conventions, p. 3*

External link (URL): www.hms-networks.com

WARNING
Instruction that must be followed to avoid a risk of death or serious injury.

Caution
Instruction that must be followed to avoid a risk of personal injury.

! Instruction that must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.

Additional information which may facilitate installation and/or operation.

1.4 Document Specific Conventions

- The terms “Anybus” or “module” refers to the Anybus CompactCom module.
- The terms “host” or “host application” refer to the device that hosts the Anybus.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- Intel byte order is assumed unless otherwise stated.
- The terms “Anybus implementation” and “Anybus version” generally refers to the implementation in the Anybus module, i.e. network type and internal firmware revision.
- Unless something is clearly stated to be optional, it shall be considered mandatory.
- A byte always consists of 8 bits.
- A word always consists of 16 bits.

1.4.1 Abbreviations

Abbreviation	**Explanation**
ACI	Application Communication Interface
FE	Functional Earth
LE, -LE	Little-Endian
BE, -BE	Big-Endian
NA	Not applicable
MS	Module status
NS	Network status
eNVM	embedded Non-Volatile Memory
FSTM	Firmware stream
RLDS	RAMLoaderDS, C40 boot code
TCEP	Test Case Execution Protocol; Serial protocol used to communicate with the C40 tests
FWloaderDS	Firmware loader application/protocol used when programming product firmware to C40
C40	NP40 chip concept
NP40	HMS Network Processor 40 chip
HMS	HMS Networks
CIP	Common Industrial Protocol; the following fieldbus networks: EtherNet/IP, DeviceNet, CompoNet, ControlNet
hiff/HIFF	HMS Interchange File Format (used for example for C40 firmware files)

1.5 Trademark Information

Anybus® is a registered trademark of HMS Networks.

EtherNet/IP™ is a trademark of ODVA, Inc.

2 About the Anybus CompactCom C40

The Anybus CompactCom C40 provides a chip based solution for integrating network connectivity directly into your PCB design. The chip, NP40, is based on proven technology from Microsemi (the SmartFusion platform). Together with the included firmware, you can design a compact solution with all functionality for network connectivity, tailor made for your application.

Available networks are EtherNet/IP™, Modbus-TCP, EtherCAT, Ethernet POWERLINK, BACnet/IP, PROFINET, DeviceNet™, CC-Link, PROFIBUS and CANopen.

The NP40 chip and the firmware are used in the Anybus CompactCom M40 and B40 series. For more information see www.anybus.com.

2.1 Features

- NP40 processor and firmware for each network, bundled as a package
- Same chip used for all the supported network protocols
  - Network firmware is downloaded to the NP40 at final product manufacturing
  - One single hardware can be designed for all supported industrial Ethernet networks
- HMS Networks provides new versions of the firmware free of charge under the license
- Up to 1500 bytes of process data in each direction
- Event-based interface
- 8/16 bit parallel interface
- SPI interface
- Stand-alone shift register interface
- Serial UART (Universal Asynchronous Receiver-Transmitter), primarily used for firmware download, production test and service

2.2 Design Process

This implementation guide will assist you in your design process by showing:

- How to arrive at a complete schematic, including component selection
- How to arrive at a complete PCB
- How to implement a high quality production test system

3 NP40 Network Processor

The NP40 is a microprocessor with on-chip RAM and Flash, and built-in network controllers. It is designed for high-performance real-time industrial Ethernet slave/adapter applications, but is also well suited for high-performance industrial network slave applications.

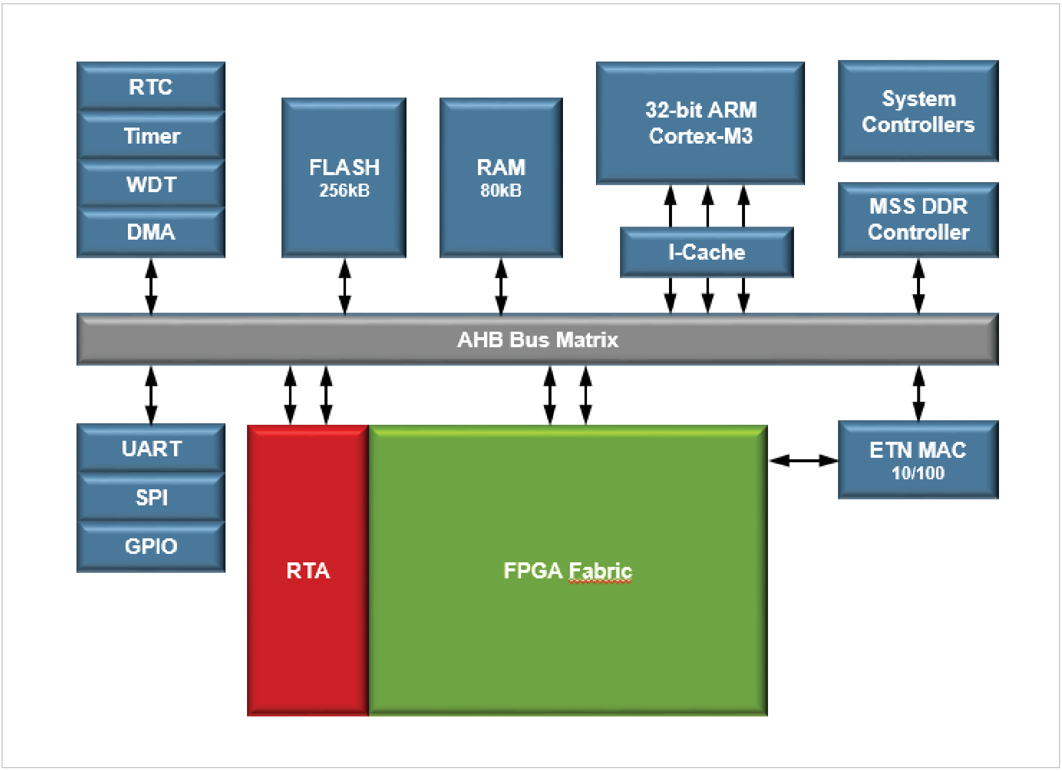


Fig. 1

Package	BGA VF400, RoHS compatible
Size	17*17 mm
Power Supply	3.3 V, 2.5 V, 1.2 V
Processor Core	ARM Cortex-M3
Operating Temperature	-40 °C to +85 °C
Junction Temperature	-40°C to +115°C
Host Application Interfaces	8-/16-bit parallel, SPI, Shift Register, UART

3.1 Network Processor Details

- High performance ARM® Cortex™-M3 with the flexibility of an FPGA
- Full network customizations
- Real Time Accelerator (RTA) architecture makes it possible to get “zero delay” data copy
- Real time network integration
- FPGA fabric for the various real-time Ethernet interfaces
- High performance with low power consumption
- Compact size
- For high-performance networks with support for sync applications
- Enabling profile support

3.2 Host Application Interfaces

- Parallel
  - 8/16-bit memory interface
  - 30ns access
  - Interrupt or polled
- Serial Peripheral Interface (SPI)
  - SPI slave interface
  - Cyclic frame with 32-bit CRC
  - Clock frequency up to 20MHz
  - 3-pin and 4-pin communication supported
- Stand-Alone Shift Register
  - Up to 32 bytes input and up to 32 bytes output
- Serial
  - Standard UART interface (8,n,1)
  - Configurable baud rates 19.2kbps – 625kbps

The host application interface is selected at startup from the settings of the OM[0..3] pins. For more information see the Anybus CompactCom 40 Software, M40 Hardware, and B40 Hardware Design Guides.

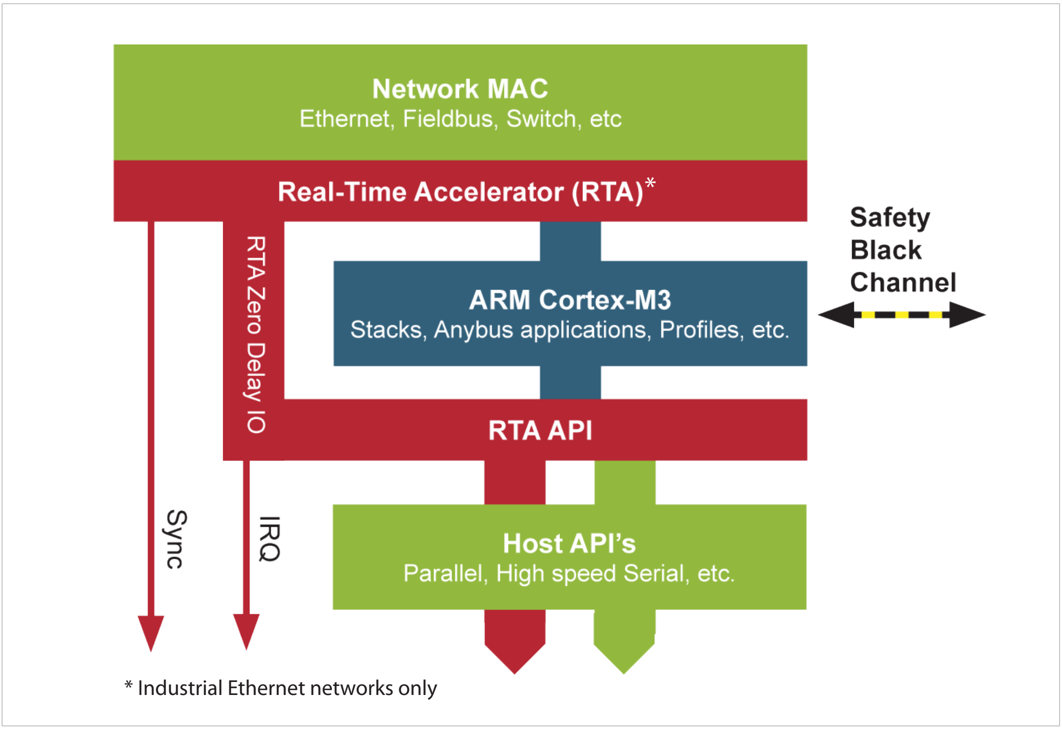

Network MAC
Ethernet, Fieldbus, Switch, etc
Real-Time Accelerator (RTA)*
Safety Black Channel
ARM Cortex-M3
Stacks, Anybus applications, Profiles, etc.
RTA Zero Delay IO
RTA API
Sync
IRQ
Host API's
Parallel, High speed Serial, etc.
* Industrial Ethernet networks only

Fig. 2

3.2.1 Safety Integration

A Safety Black Channel is implemented in the chip. This makes it possible to run safe communication, using an external safety module (for example IXXAT Safe T100). PROFIsafe, CIP Safety and FSoE are supported by the Anybus CompactCom 40 series.

3.2.2 Interface for Testing and Programming

The first time firmware is downloaded to the NP40 it requires the use of the serial UART (Rx and Tx pins). While future firmware updates can be done using the serial UART, there are also other ways to do this. See Anybus CompactCom 40 Software Design Guide for more information.

The production tests and services also require the use of the serial UART of the NP40.

! It is very important that the hardware is prepared for access to this interface from the beginning, since it is the only way to download production parameters and firmware in an empty NP40. Some downloads are large, and it is recommended to prepare the interface for the highest supported baud rate (625kbps) to speed up the procedure.

4 Components

It is recommended that all components are of industrial temperature range, -40 °C to +85 °C, to allow use of the NP40 in different environments and housings.

Bill of Materials (BOM) for the Anybus CompactCom M40 example designs is available from HMS.

! The C40 firmware is developed for a design using the components in the BOM. If other components are chosen, future firmware updates might fail.

4.1 Crystal

A 20 MHz crystal with ±25 ppm accuracy (including initial frequency offset and temperature drift, but not aging) or better is used to generate the external clock.

4.2 SRAM

In some implementations, an external SRAM memory must be used.

Signal	Function	Direction	Note
A0..XX	Address bus	NP40→SRAM	See schematics for more information
D0..15	Data bus	bidirectional	
OE_N	Output enable	NP40→SRAM	
WE_N	Write enable	NP40→SRAM	
LB_N	Lower byte enable	NP40→SRAM	
UB_N	Upper byte enable	NP40→SRAM	
CS_N	Chip select	NP40→SRAM	

An external SRAM is not used with PROFIBUS, DeviceNet, CC-Link and CANopen.

4.3 SPI Flash

The SPI flash memory holds configuration parameters, product specific data, and the communication firmware, that will be downloaded and used by the chip at the next restart. There is also an area for a file system.

Signal	Function	Direction	Note
FLASH_SCLK	Serial clock	NP40→FLASH	
FLASH_MOSI	Serial data input	NP40→FLASH	
FLASH_MISO	Serial data output	FLASH→NP40	
FLASH_SS_N	Chip select	NP40→FLASH	Pull up with 2,2 kΩ

4.4 NP40 Power Pins

In Anybus CompactCom M40 and B40 designs, each power rail is decoupled by a 10 µF bulk capacitor. Each power pin is decoupled by a 100 nF capacitor at close range.

A PLL filter is needed close to the PLL pins, to reduce jitter.

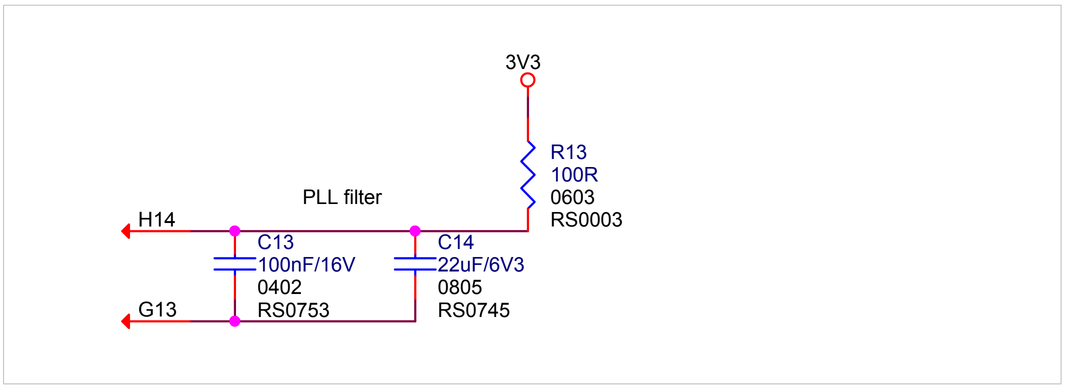


Fig. 3

Pin G13 is connected to GND on die.

4.5 FE Connection

The application as a whole will have to adapt FE (functional earth) connections to the requirements of the industrial network. See the Anybus CompactCom 40 Hardware Design Guide for more information.

Sometimes the term “protective earth” or “PE” is used instead of FE, but this should be avoided as these connections does not concern safety.

4.6 Ethernet PHY

The following Ethernet PHY circuits are compatible with C40. The PHYCLK phase delay strapping may differ depending on the PHY that is used. The strappings are applied to the signals TxD1 and TxD0.

		Supported PHYs			
		Broadcom BCM5241	National DP83848	National DP83630/ DP83640	Microchip KSZ8081M
Networks	EtherNet/IP	Yes	Does not fulfill Quick Connect and DLR requirements	Does not fulfill Quick Connect requirements	Has high latency which can affect highest performance applications
	PROFINET	Yes	Yes	Yes	
	POWERLINK	Yes	Yes	Yes	
	EtherCAT	Yes	Yes	Yes	
	Modbus-TCP	Yes	Yes	Yes	
PHYCLK phase [TxD1;TxD0]		[1;0]	[1;0]	[1;0]	[0;1]

Support for Quick Connect and DLR is not mandatory for an EtherNet/IP adapter. See the Anybus CompactCom 40 EtherNet/IP Network Guide for more information.

5 Hardware Design Considerations

5.1 Schematics

Complete schematics of the Anybus CompactCom M40 design and the pinning of the NP40 are available after signing the license agreement.

> **!** Unconnected pins in the example schematics must remain unconnected when designing a C40 device. Routing shortcuts via unconnected pins is not allowed.

5.1.1 Block Diagram (Ethernet)

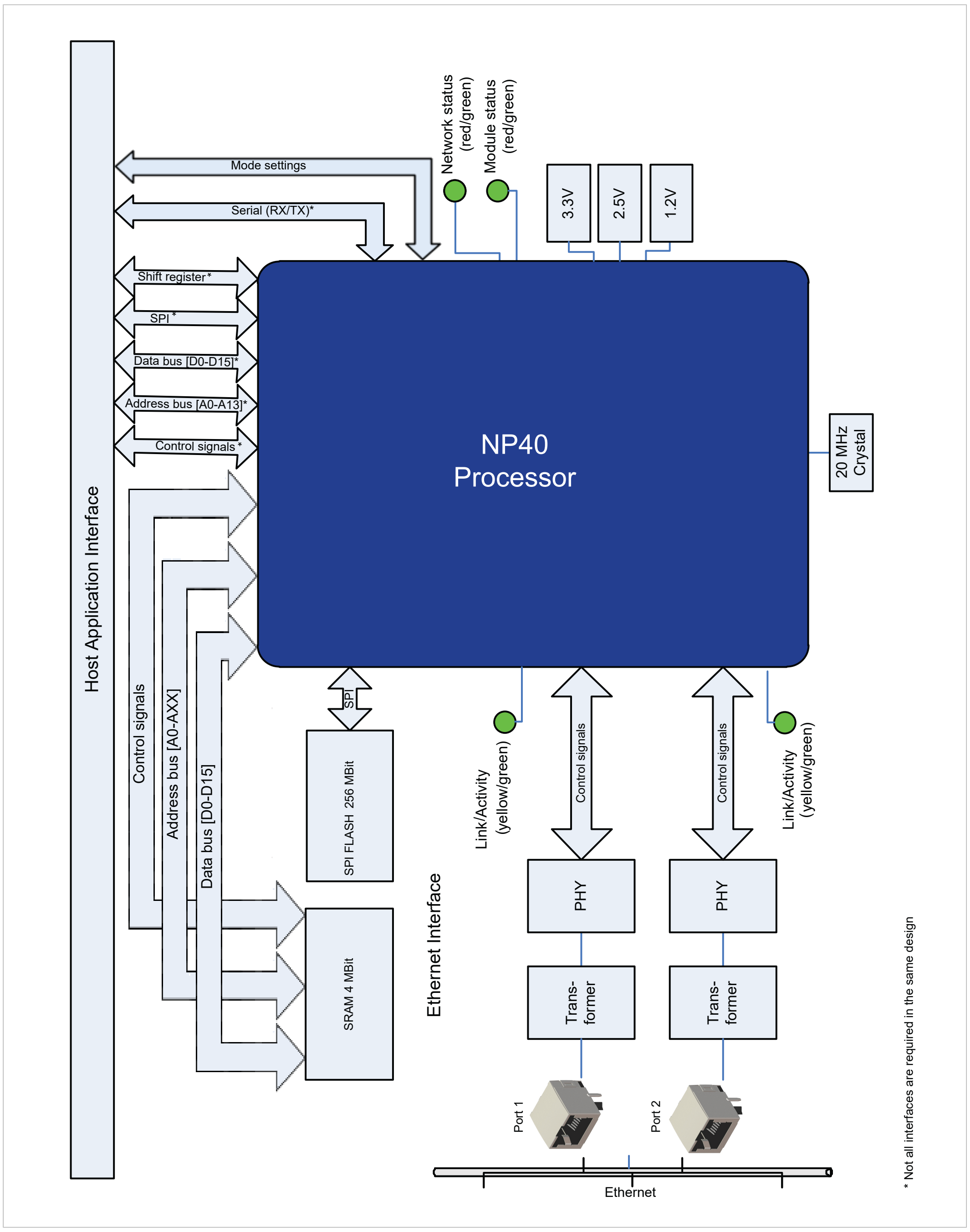


Fig. 4

5.1.2 LED Driver Outputs

A group of eight signals, the NW_LED signals, from the NP40 are used as direct LED drivers. They shall – in the extent that they are used – be connected as described in the examples in the pictures below. The resistor values should be chosen to get even light between different LEDs.

LED name	Signal Name	Default color	Default Functionality	
LED1	NW_LED1A	Green	Network status	
	NW_LED1B	Red		
LED2	NW_LED2A	Green	Module status	
	NW_LED2B	Red		
LED3	NW_LED3A	Green	All Industrial Ethernet Networks:	Link/Act for the network port (port A)
			Other:	Not used
	NW_LED3B	Yellow	EtherNet/IP, Modbus-TCP	10 Mbit Link/Act for Link/Act for the network port (port A)
			Other	Not used
LED4	NW_LED4A	Green	All Industrial Ethernet Networks:	Link/Act for the network port (port B)
			Other:	Not used
	NW_LED4B	Yellow	EtherNet/IP, Modbus-TCP	10 Mbit Link/Act for Link/Act for the network port (port B)
			Other	Not used

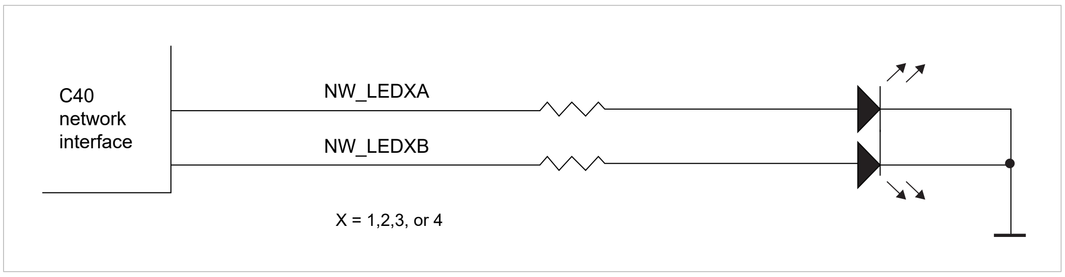


Fig. 5

See *Absolute Maximum Ratings, p. 21* for information about maximum current limitations for the LEDs.

5.1.3 Power Supply

It is recommended that all power supply pins from the host application, 3V3 and GND, are connected directly to the power planes of the PCB. That means that no filters should be used.

Apart from 3V3, the NP40 is powered by 1V2 (Core) and 2V5 (Memory) supplies. These supplies should be generated by regulators on board. To save power and reduce temperature issues, it is recommended to use switch mode DC-DC converters, and not LDOs, unless the power and temperature effect is small. The figure shows a recommended DC-DC converter solution.

The different power supply sources do not have to be synchronised, however it is important that all power levels are within specification when the reset is released.

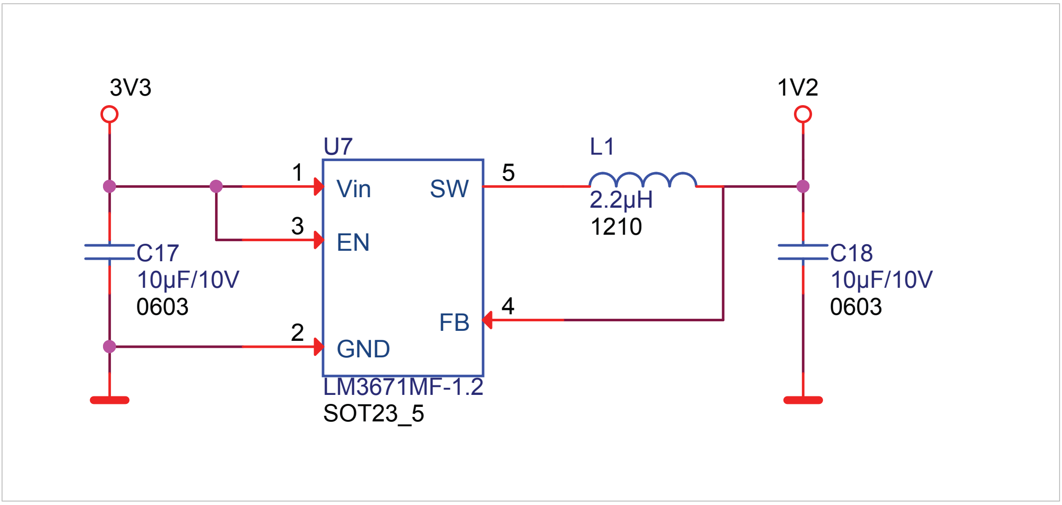


Fig. 6

See *DC Electrical Characteristics, p. 22* for more information.

5.1.4 Reset Circuitry

Assertion of a full reset is possible at any time.

A capacitor can be mounted close to the NP40 reset input if the reset signal has influences from transients.

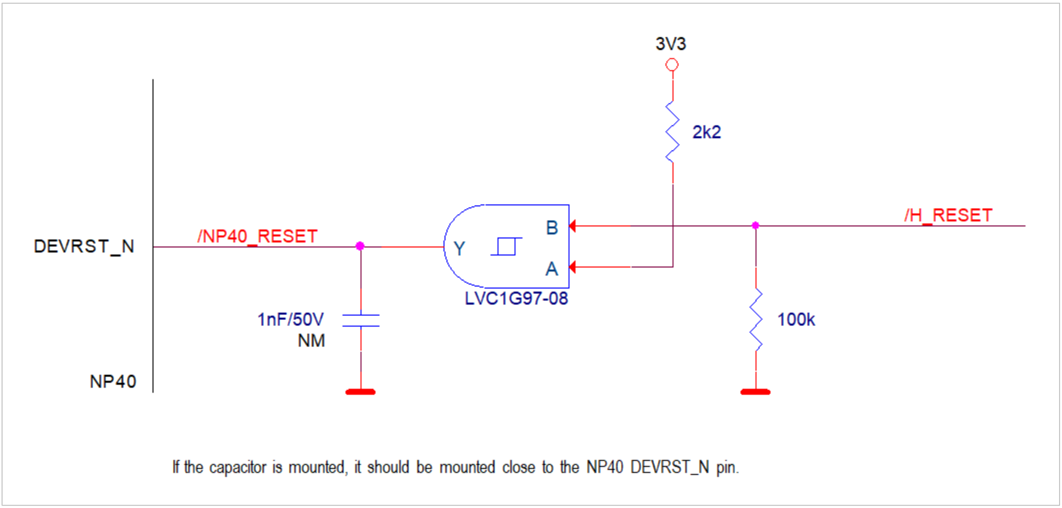

3V3
2k2
/H_RESET
DEVRST_N
/NP40_RESET
Y
B
A
LVC1G97-08
1nF/50V
NM
100k
NP40
If the capacitor is mounted, it should be mounted close to the NP40 DEVRST_N pin.

Fig. 7

If the reset signal is the result of more than one reset source with open drain output, a Schmitt trigger AND-gate or equivalent is used.

5.1.5 MI0/SYNC Circuitry

The Module Identification pins are read by the host application while the NP40 is in reset and expected to be a low impedance output. This means that an external pull-down resistor and a buffer is needed, as described below:

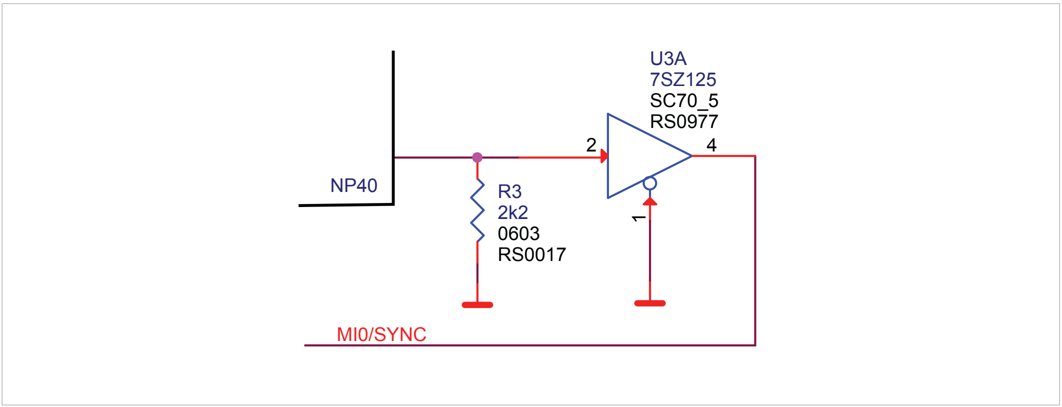


Fig. 8

5.2 PCB Layout

The PCB layout of the Anybus CompactCom M40 design is available as reference (in PDF format, no CAD files) after signing the license agreement.

5.2.1 PCB Layout Guidelines

It is recommended to design according to the following requirements:

- The PCB shall be designed according to IPC-2220.
- Via drill diameter is:
  - 0,2 mm hole in a 0,6 mm pad for through hole VIAs.
  - 0,2 mm hole in a 0,51 mm pad for through hole VIAs in special cases where other options can’t be applied. Pad on other planes (not BGA-side) could be 0,6mm, preferably used for NP40 power connections. Use as few vias with 0,51 mm pad as possible since the risk of manufacturing problems increases. Reduced via pads should only be used on one side of the PCB to allow PCB manufacturers to drill from that side.
  - In special cases where it is absolutely necessary to let conductors pass vertically or horizontally between vias that are distributed with 0,8 mm pitch (BGA pitch for NP40), it is possible to use vias of 0,2 mm hole in a 0,418 mm pad. The pads should be 0,6 mm in all layers that do not have passing conductors. Use as few vias with 0,418 mm pad as possible since the risk of manufacturing problems increases. Reduced via pads should only be used on one side of the PCB to allow PCB manufacturer to drill from that side.
  - 0,1 mm hole in 0,40 mm pad for micro VIA’s depending on the PCB density.
- Track width is minimum 5 mils.
- Spacing is minimum 5 mils.
- Ethernet signals from the PHY to the connectors should be routed with a 100 Ω differential impendance.

5.2.2 PCB Isolation Requirements

The requirements in this table are used for HMS Anybus CompactCom M40 and B40 modules. Note that other requirements may be listed for other applications. The recommendation is to use the same isolation distances.

Isolation barrier	**Working/transient voltage**		**Specified minimum distance**		**Notes**
	Clearance	**Creepage**	**Within layer**	**Between layers**	
Host to FE	B/S 700/2500 V	500 V	2,5 mm	0,4 mm	For PROFIBUS Shield is the same net as FE.
Host to Network	B/S 700/2500 V	500 V	2,5 mm	0,4 mm	-
Shield to FE	B/S 280/1500 V	296 V	1,4 mm	0,4 mm	-
Network to FE	B/S 280/1500 V	296 V	1,4 mm	0,4 mm	-
Network to Shield Ethernet	F 210/800 V	160 V	0,4 mm	0,1 mm	-
Network to Shield, others	B/S 280/1500 V	296 V	1,4 mm	0,4 mm	There is no guideline for the lower value. Network-dependent. A backwards calculation for PROFIBUS has given 200V.
Network A to Network B	B/S 280/1500 V	296 V	1,4 mm	0,4 mm	For two-port Ethernet products. No given guidelines
Shield A to Shield B	B/S 210/800 V	243 V	0,95 mm	0,4 mm	For two-port Ethernet products. No given guidelines

The information is based on EN60950-1:2006, secondary circuit, Material group IIIa and Pollution degree 2.

5.3 Soldering

5.3.1 Standard Reflow Profile for Lead-Free Packages

The reflow provided here, is for reference only. Users are advised to optimize their own board level parameters to get proper reflow outcome. Per package qualification maximum number of reflow that can be done on NP40 packages is 3.

Classification Temperature (T_C)

Package Thickness	Volume mm³ < 350	Volume mm³ 350 - 2000	Volume mm³ > 2000
< 1.6 mm	260 + 0 °C	260 + 0 °C	260 + 0 °C

Note:

- Tolerance: The device manufacturer/supplier shall assure process compatibility up to and including the stated classification temperature at the rated MSL level (this means peak reflow temperature + 0 °C. For example, 260 °C + 0 °C).
- At the discretion of the device manufacturer, but not the board assembler/user, the maximum peak package body temperature (T_p) can exceed the values specified in the table above and also in the table below (Classification Reflow Profile). The use of a higher T_p does not change the classification temperature (T_C).
- Package volume excludes external terminals (balls, bumps, lands, leads) and/or nonintegral heat sinks.
- The maximum component temperature reached during reflow depends on package thickness and volume. The use of convection reflow processes reduces the thermal gradients between packages. However, thermal gradients due to differences in thermal mass of SMD packages may still exist.
- Moisture sensitivity levels of components intended for use in a Pb-free assembly process shall be evaluated using the Pb-free classification temperatures and profiles defined above and in the table below (Classification Reflow Profile), whether or not Pb-free.
- SMD packages classified to a given moisture sensitivity level by using Procedures or Criteria defined within any previous version of J-STD-020, JESD22-A112 (rescinded), IPC-SM-786 (rescinded) do not need to be reclassified to the current revision unless a change in classification level or a higher peak classification temperature is desired.

Classification Reflow Profile

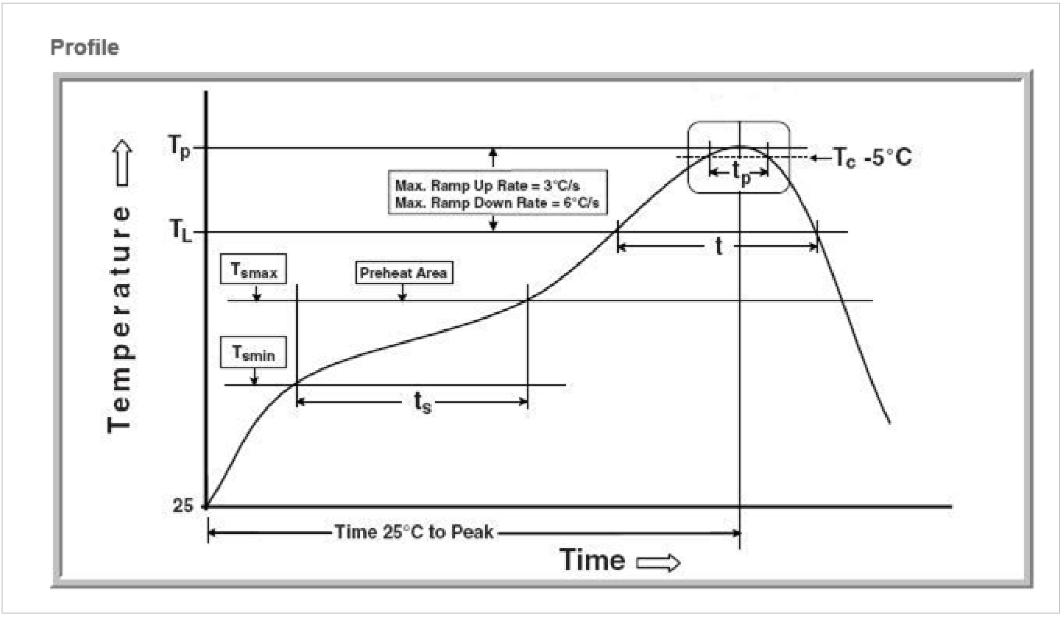


Fig. 9

Profile Feature	**Pb-Free Assembly**
Preheat and Soak Temperature minimum (T_{smin}) Temperature maximum (T_{smax}) Time (T_{smin} to T_{smax}) (t_s)	150 °C 200 °C 60 - 120 seconds
Average ramp-up rate (T_{smax} to T_p)	3 °C/second maximum
Liquidous temperature (T_L) Time at liquidous (t_L)	217 °C 60 - 150 seconds
Peak package body temperature (T_p)	See table above (Classification Temperature (T_C))
Time (t_p) within 5 °C of the specified classification temperature	30 seconds
Average ramp-down rate (T_p to T_{smax})	6 °C/second maximum
Time 25 °C to peak temperature	8 minutes maximum

Note:

- Tolerance for peak profile temperature (T_p) is defined as a supplier minimum and a user maximum.
- Tolerance for time at peak profile temperature (t_p) is defined as a supplier minimum and a user maximum.
- All temperatures refer to the center of the package, measured on the package body surface that is facing up during assembly reflow (for example live-bug). If parts are reflowed in other than the normal live-bug assembly reflow orientation (i.e. dead-bug), T_p shall be within ± 2 °C of the live-bug T_p and still meet the T_C requirements. Otherwise, the profile shall be adjusted to achieve the latter. To accurately measure actual peak package body temperatures, refer to JEP140 for recommended thermocouple use.

- Reflow profiles in this document are for classification/preconditioning and are not meant to specify board assembly profiles. Actual board assembly profiles should be developed based on specific process needs and board designs and should not exceed the parameters in the table above.

  For example, if T_C is 260 °C and time t_p is 30 seconds, this means the following for the supplier and the user:

  – For a supplier: The peak temperature must be at least 260 °C. The time above 255 °C must be at least 30 seconds.

  – For a user: The peak temperature must be at least 260 °C. The time above 255 °C must not exceed 30 seconds.

- All components in the test load shall meet the classification profile requirements.
- SMD packages classified to a given moisture sensitivity level by using Procedures or Criteria defined within any previous version of J-STD-020, JESD22-A112 (rescinded), IPC-SM-786 (rescinded) do not need to be reclassified to the current revision unless a change in classification level or a higher peak classification temperature is desired.

5.3.2 Storage of C40

Store in dry packing or cabinet with controlled humidity, JEDEC J-STD-020 level 3.

The product may be stored indefinitely once the dry pack is opened provided that the ambient humidity does not exceed 20% RH. Humidity controlled dry-air or dry-nitrogen cabinets are recommended for this purpose.

Floor life at 30°C/60% RH is 168 hours.

5.3.3 Baking process for C40 JEDEC J-STD-020 level 3

A re-bake is recommended if storage conditions exceed either 30°C/60% RH or the quoted floor life. The time and temperature of the bake depend on the package thickness as defined in the table below.

The floor life clock is reset to zero following the re-bake. However if the ambient conditions still do not meet those defined the product should be used immediately following the bake.

Default Baking Times for Trays with <150°C Tolerance	
Bake at 125°C	16 hours
Bake at 140°C	9 hours
Bake at 150°C	8 hours

5.4 Firmware

The network firmware is available after the license agreement is signed.

The network firmware and C40 production parameters should be downloaded to the NP40 at production of the C40 based device. See *Production Test Guidelines, p. 29* .

6 NP40 Technical Data

6.1 Package

The NP40 is packaged in a 400-pin VFPBGA (Very Fine Pitch Ball Grid Array).

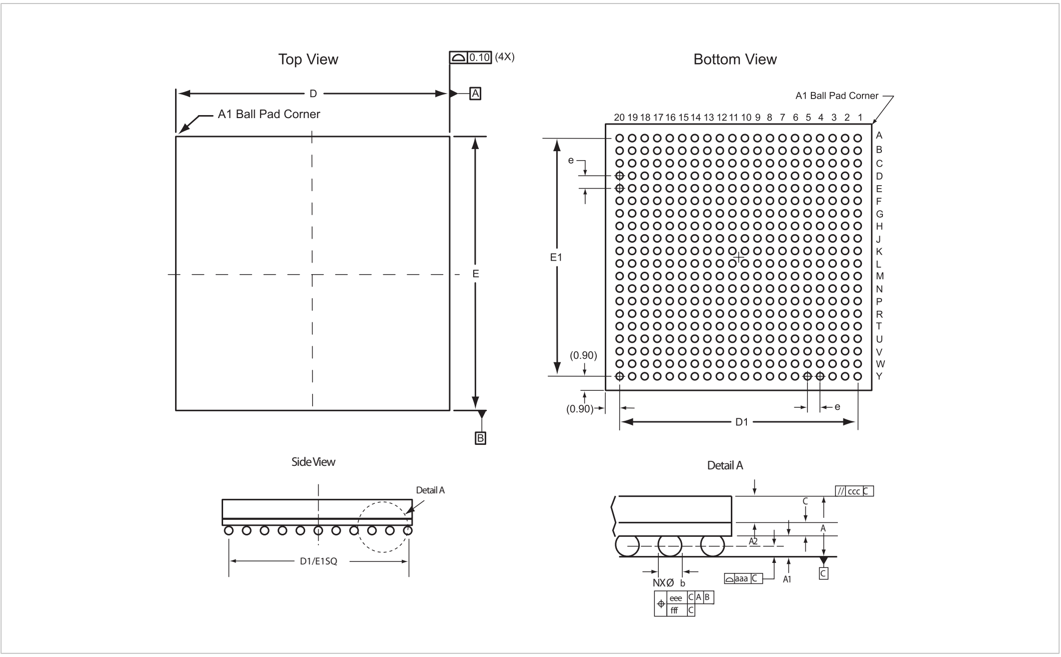


Fig. 10

Dimension (JEDEC equivalent)	Min (mm)	Nom (mm)	Max (mm)
A	1.31	1.41	1.51
A1	0.32	0.37	0.42
A2	0.65	0.70	0.75
aaa	0.12		
b	0.41	0.46	0.51
c	0.29	0.34	0.39
ccc	0.10		
D/E	17.00 BSC		
D1/E1	-	15.20	-
e	0.80 BSC		
eee	0.15		
fff	0.08		

6.2 Pin Signal Descriptions

Pin signal descriptions will be available upon signed license agreement.

6.3 Absolute Maximum Ratings

Temperature	Operating junction	-40 °C to +115 °C
	Programming junction	0 °C to +85 °C
	Storage	-65 °C to +150 °C
Allowable sink/source current per pin	ACI	8 mA
	LED	20 mA
	Network	8 mA
	SRAM	8 mA
	Flash	8 mA
Total current	3V3	100 mA
	2V5	150 mA
	1V2	600 mA

6.4 Thermal Characteristics

The absolute maximum ratings for temperature are given as junction values. Use the equations below to calculate the ambient, boards and case temperatures

$$\theta_{JA} = \frac{T_J - T_A}{P}$$

$$\theta_{JB} = \frac{T_J - T_B}{P}$$

$$\theta_{JC} = \frac{T_J - T_C}{P}$$

Fig. 11

θ_{JA} = Junction-to-air thermal resistance

θ_{JB} = Junction-to-board thermal resistance

θ_{JC} = Junction-to-case thermal resistance

T_J = Junction temperature

T_A = Ambient temperature

T_B= Board temperature (measured 1.0 mm away from the package edge)

T_C= Case temperature

P = Total power dissipated by the device

	θJA			**θJB**	**θJC**	**Units**
	Still Air	**1.0 m/s**	**2.5 m/s**			
Package thermal resistance	19.40	15.75	14.22	8.11	4.22	°C/W

6.5 DC Electrical Characteristics

T_A = -40 °C to +85 °C

V_{DD} = 1.2 V ± 5%, IOV_{CC} = 2.5 to 3.3 V ± 5%; all ground references 0 V

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
3V3	3.3 V DC supply voltage	T_A = -40 °C to +85 °C	3.15	3.3	3.45	V
2V5	2.5 V DC supply voltage		2.375	2.5	2.625	V
1V2	1.2 V DC supply voltage		1.14	1.2	1.26	V
V_{IH}	Logical 1 Input Voltage	3V3 supply voltage	2.0		3.45	V
V_{IL}	Logical 0 Input Voltage		-0.3		0.8	V
V_{IH}	Logical 1 Input Voltage	2V5 supply voltage	1.7		2.625	V
V_{IL}	Logical 0 Input Voltage		-0.3		0.7	V
V_{IH}	Logical 1 Input Voltage	LVPECL, differential standard			2.3	V
V_{IL}	Logical 0 Input Voltage		1.6			V
V_{ICM}	Input common mode voltage	LVPECL	0.3		2.8	
V_{IDIFF}	Input differential voltage		100	300	1000	mV
I_L	High impedance input leakage				10	?A
I_{OH}	Logical 1 Output Current	V_{OH} = VDDI - 0.4 V			-8 (ACI, network) -20 (LED)	mA
I_{OL}	Logical 0 Output Current	V_{OL} = 0.4 V			8 (ACI, network) 20 (LED)	mA
I_{OH}	Logical 1 Output Current, memory	V_{OH} = 2V5 -0.4,			-8	mA
I_{OL}	Logical 0 Output Current, memory	V_{OL} = 0.4 V			8	mA
Icc	Supply Current Active	V_{CC} = 3V3	Depending on the network protocol.		100	mA
		V_{CC} = 2V5			150	mA
		V_{CC} = 1V2			600	mA

6.6 Reset

The following requirements must be met by the reset regulator connected to the reset input signal.

6.6.1 Power Up

The reset input is active low. It must be connected to a host application controllable output pin in order to handle the power up sequence, voltage deviations and to be able to support network reset requests.

There is no internal reset regulation. To establish a reliable interface, the host application is solely responsible for resetting the application when the supply voltage is outside the specified range.

A fast rise time is required of the reset signal, preferably equal to the slew rate of typical logical circuits. A simple RC circuit is for example not sufficient to guarantee stable operation, as the slew rate from logic 0 to logic 1 is too slow.

! The rise time of the reset signal should be as fast as possible, and must not exceed 10 ns. The signal is not under any circumstances allowed to be left floating. Use a pull-down to prevent this. The minimum pulse length of the reset signal is 1 µs.

The following requirements must be met by the reset regulator connected to the reset input signal.

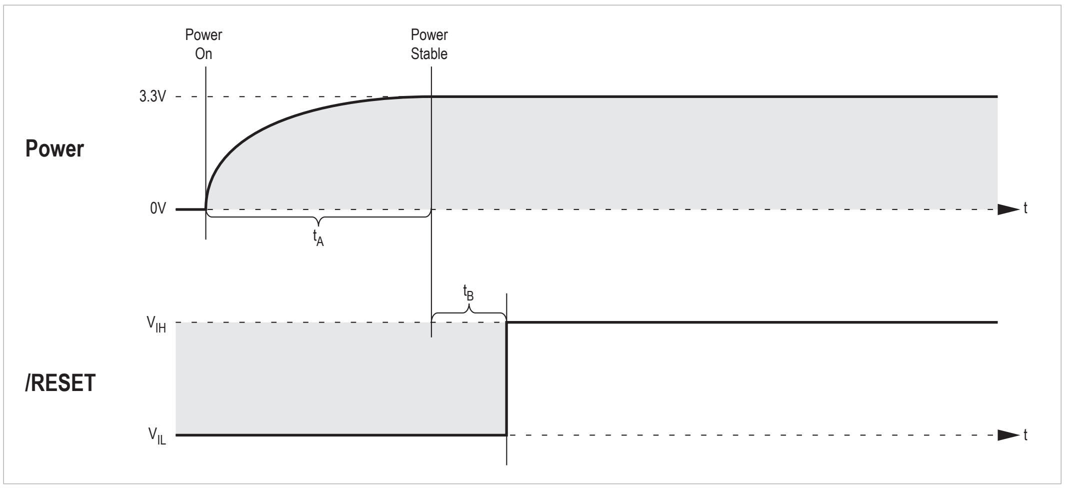


Fig. 12

Power up time limits are given in the table below:

Symbol	Min.	Max.	Definition
t_A	-	-	Time until the power supply is stable after power-on; the duration depends on the power supply design of the host application and is thus beyond the scope of this document.
t_B	1ms	-	Safety margin.

6.6.2 Restart

The reset pulse duration must be at least 10 µs in order for the NP40 to properly recognize a reset.

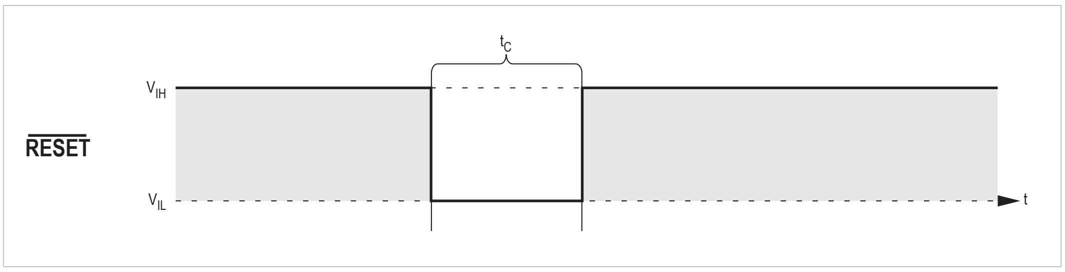


Fig. 13

Symbol	Min.	Max.	Definition
t_C	10 µs	-	Reset pulse width.

6.7 Operating Mode Signals

The OM[0..3] signals have to be stable for more than 10 µs before the rising edge of the reset signal. See the Anybus CompactCom 40 Software Design Guide for information on how to interpret the OM[0..3] signals.

6.8 ACI Parallel Timing

6.8.1 8-bit Memory Access

$\overline{CS}$	$\overline{WE}$	$\overline{OE}$	D[0..7]	Comment
1	X	X	High impedance	Module not selected
0	0	X	Data Input (Write)	Data on D[0..7] is written
0	1	0	Data Output (Read)	Data is available on D[0..7]
0	1	1	High impedance	Module is selected, but D[0..7] is in high impedance state

X = don’t care

6.8.2 16–bit Memory Access

$\overline{CS}$	$\overline{WEL}$	$\overline{WEH}$	$\overline{OE}$	D[0..15]	Comment
1	X	X	X	High impedance	Module not selected
0	0	0	X	Data Input (Write)	Data on D[0..15] is written
0	0	1	X	Data Input (Write)	Data on D[0..7] is written (8bit write)
0	1	0	X	Data Input (Write)	Data on D[8..15] is written (8bit write)
0	1	1	0	Data Output (Read)	Data is available on D[0..15]
0	1	1	1	High impedance	Module is selected, but D[0..15] is in high impedance state

X = don’t care

6.8.3 Read Timing

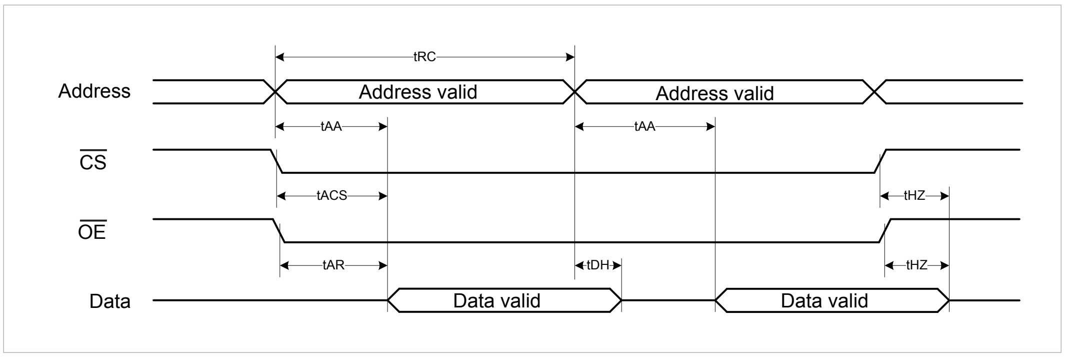


Fig. 14

Item	Description	Min	Max	Unit
tRC	Read cycle time	30	-	ns
tAA	Address valid to Data valid	-	30	
tACS	$\overline{\text{CS}}$ low to Data valid	-	30	
tAR	$\overline{\text{OE}}$ low to Data valid	-	15	
tHZ	$\overline{\text{CS}}$ or $\overline{\text{OE}}$ high to output high-Z	-	15	
tDH	Data hold time	0	-	

$\overline{\text{WE}}$ must remain high during a read access.

The timing diagram above shows a burst read but the timing applies for a single read as well.

The NP40 has no setup or hold requirements on the address bus relative to $\overline{\text{CS}}$ during read operations. The only limitation on read setup and hold-times is that the ping-pong and power up interrupt will be acknowledged if the value 3FFFh is present on the address bus for 10-15 ns or more while $\overline{\text{CS}}$ is low.

6.8.4 Write Timing

Writing to the memory can either be controlled by write enable (first figure below) or by chip select (second figure below)

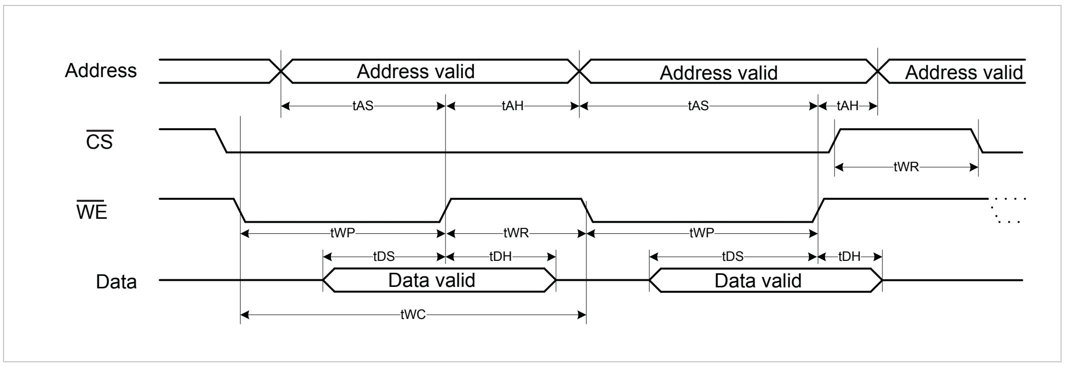


Fig. 15

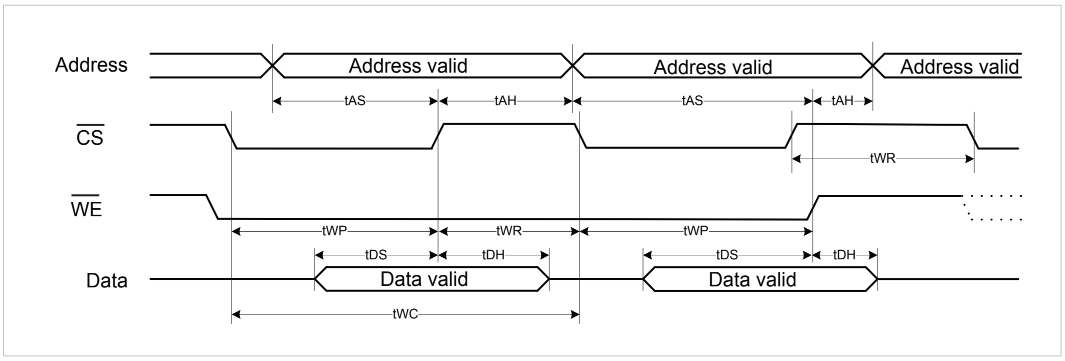


Fig. 16

Item	Description	Min	Max	Unit
tWC	Write cycle time	30	-	ns
tAS	Address valid before End-of-Write	15	-	
tAH	Address valid after End-of-Write	0	-	
tWP	$\overline{CS}$ and $\overline{WE}$ low pulse width	15	-	
tDS	Data valid before End-of-Write	15	-	
tDH	Data valid after End-of-Write	0	-	
tWR	Write recovery time	10	-	

$\overline{OE}$ is “don’t care” when $\overline{WE}$ is low.

In 16-bit mode, all timing requirements of $\overline{WE}$ applies to both $\overline{WEL}$ and $\overline{WEH}$.

The timing diagram above shows a burst write but the timing applies for a single write as well.

6.9 ACI SPI Timing

The C40 implements an SPI slave which means that MOSI is the data signal from the master SPI controller to C40 and MISO is the data signal from C40 to the SPI controller. The SPI master delivers the serial clock SCLK and the select signal $\overline{SS}$. C40 captures MOSI data on the rising edge of SCLK and propagates MISO data on the falling edge of SCLK. This is commonly known as SPI Mode 0.

The SPI interface can be operated in 4-wire mode, or in 3-wire mode where the C40 acts as slave on the SPI bus. The C40 samples the state of $\overline{SS}$ at power up. If $\overline{SS}$ is high then 4-wire mode is selected. If $\overline{SS}$ is low then three wire mode is selected.

In 4-wire mode the $\overline{SS}$ signal is used to indicate the start and stop of an SPI transfer. In this mode the SCLK signal is allowed to be either idle high or idle low. This mode also allows multiple SPI slaves on the same SPI bus, since C40 MISO is tristated when $\overline{SS}$ is high.

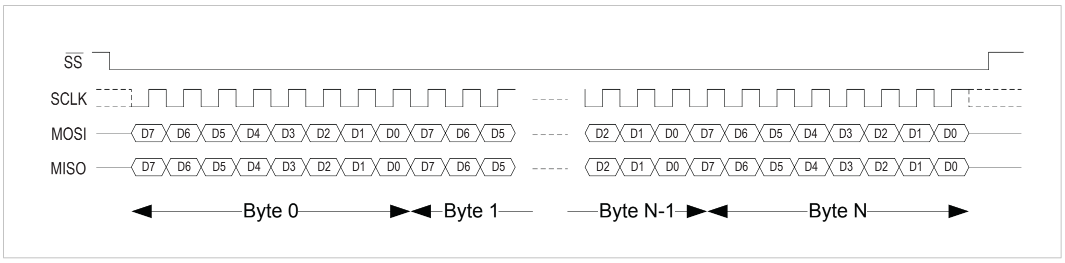

$\overline{SS}$
SCLK
MOSI
MISO
D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5
D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
Byte 0
Byte 1
Byte N-1
Byte N

Fig. 17

In 3-wire mode the $\overline{SS}$ signal must be tied low permanently, and the SCLK signal must be idle high. Multiple SPI slaves on the same bus are not possible in this mode. The C40 detects start and stop of a transfer by monitoring SCLK activity. There must be an idle period of at least 6 µs between two transfers in this mode. The SCLK must never remain at the high level for more than 5 µs during a transfer.

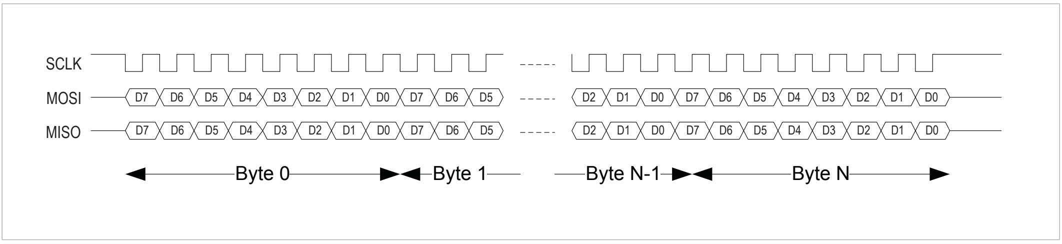

SCLK
MOSI
MISO
D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5
D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
Byte 0
Byte 1
Byte N-1
Byte N

Fig. 18

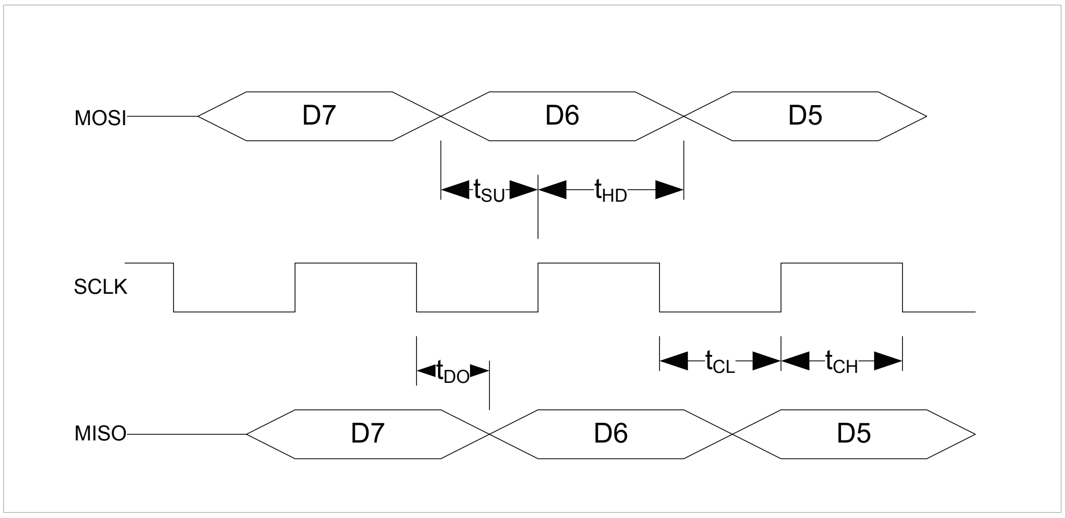


Fig. 19

SPI bit timing for 3-wire mode is shown in the picture below.

Item	Description	Min Value	Max Value
tSU	MOSI setup before SCLK rising edge	8 ns	-
tHD	MOSI hold after SCLK rising edge	8 ns	-
tDO	MISO change after SCLK falling edge	4 ns	21 ns
tCL	SCLK low period	20 ns	-
tCH	SCLK high period	20 ns	-
tCL+tCH	SCLK period	50 ns	-

6.10 Shift Register Mode Timing

The Anybus CompactCom C40 operates in 12.5 MHz in shift register mode.

6.10.1 Timing Diagram

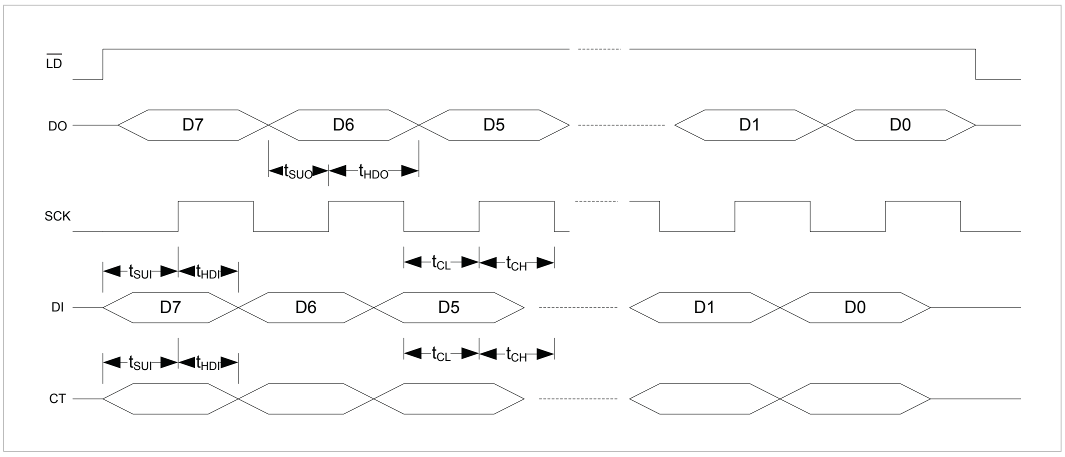


Fig. 20

Abbreviations from the diagram above, explained, and timing details:

Item	Description	Min Value
tSUO	DO setup before SCLK rising edge	20 ns
tHDO	DO hold after SCK rising edge	20 ns
tSUI	DI/CT setup before SCK rising edge	10 ns
tHDI	DI/CT hold after SCK rising edge	0 ns
tCH	SCK high half period	35 ns
tCL	SCK low half period	35 ns
tCH+tCL	SCK period	78 ns

The idle time between two transfers, i.e. the $\overline{\text{LD}}$ signal is low, has to be at least 1 µs.

In shift register mode, the C40 always transfers 32 bytes input and 32 bytes output, regardless of the actual number of connected shift registers.

7 Production Test Guidelines

7.1 Introduction

To produce a C40 unit, a sequence of tests, configurations, and other actions must be performed on the embedded target. The tests are provided in two different HIFF files that are loaded via the serial UART protocol to the preprogrammed boot code in the NP40, where it is verified and, if approved, executed in the internal RAM. The contents of these HIFF files are signed by HMS and the NP40 boot code will not execute the loaded content if the encapsulated signature mismatches. This guarantees that HMS is the originator of the loaded test applications and firmware.

You should use the test cases during the production of your C40 based product, to program mandatory data segments inside the on board SPI flash, to perform physical hardware tests, and to program firmware and production data.

HMS provides production test sequences for the C40 for each supported industrial network. You need to perform these to produce a C40 unit.

> **!** When the licensing agreement has been signed, HMS will deliver firmware HIFF files and production release information for the licensed industrial network or networks. To optimize prototyping, this file or these files should be copied to the /firmware/ folder in the NP40 licensing package, renaming them the same as the placeholder files already present. The file ABCC_40_PIR_7258_1_01_01.hiff should for example be renamed ABCC_40_PIR_7258_X_YY_ZZ.hiff, which is the file name used in the delivered test scripts.

> **!** Always check the production release information document of each firmware release for the correct production parameter settings.

The production test sample package contains three methods to perform the tests:

- Developer GUI interface

  A Windows GUI loads the embedded target HIFF files, that contain the test cases, and executes them by the manipulation fields in the GUI.

  The GUI also allows you to select a firmware file to be downloaded to the C40 unit if desired.

  The method is described in more detail in *Developer GUI Interface, p. 48*

- Automated production script

  A production script is executed by the sample application that automates the manual steps that are performed in the previous method.

  It is recommended to use this option during the integration stages of the C40 product, when the need to produce early prototypes is the focus.

  HMS provides example production scripts that can be used to execute the required test sequences for each supported C40 industrial network type.

  Four example files are provided per industrial network:

  – Full production, generic: Generic test sequences that are common to several industrial networks or fieldbuses.

  – Full production, locked: Test sequences that are locked to a specific fieldbus or industrial network

  – Re-test and firmware download

  – Firmware download only

  The method is described in more detail in *Automated Production Scripts, p. 53*.

- DLL API

  A public DLL, that can be included in a production test application at the user's side, is provided. Everything that can be achieved by the previously described methods are possible to perform with the API. One advantage is that you can more smoothly integrate the HMS production API into the production environment of your own.

  HMS provides all source code to the public DLL API. This makes it possible to integrate/port the API to a different platform.

  The method is described in more detail in *DLL API, p. 61*.

7.2 C40 Production Package Contents

The C40 Production Package consists of the following directories and files:

Directories	
dllapi	Windows console example using the C40 DLL API to produce a unit based on the C40.
firmware	Directory for the network firmware files and the 'Golden Image' to be programmed during the production procedure. **Note:** The initial package contains empty network firmware files that must be replaced with real files before programming.
proddata	Examples of production data to be programmed during the production procedure. **Note:** The production data must be generated uniquely for every produced unit. The enclosed python script can be used to read and generate .bin-files.
scripts	Example scripts for different networks according to the C40 Production Test Sequences.
source	C40 ProdTest Sample App: Source code for the Windows C40 Production Test Sample App using the C40 DLL API. C40 ProdTest API: Source code for the DLL API.
testapps	Test firmware and test specifications for the different tests required during the production procedure.

Files	
7013-RLDS_App-ABCC40_CommonTests_X_YY_ZZ.hiff	Common tests (descriptions can be found in the testapps-directory)
7013-RLDS_App-ABCC40_ProductFirmwareTests_X_YY_ZZ.hiff	Product firmware tests (descriptions can be found in the testapps-directory)
C40 ProdTest Sample App.exe	The executable 'Developer GUI interface'
C40ProdTestApi.dll	DLL with functions used to produce a unit based on the C40 using the DLL API

7.3 Connecting the NP40 for Production Test

The tests are performed using the serial UART channel of the NP40. Refer to the picture below for how to connect the NP40 for serial mode.

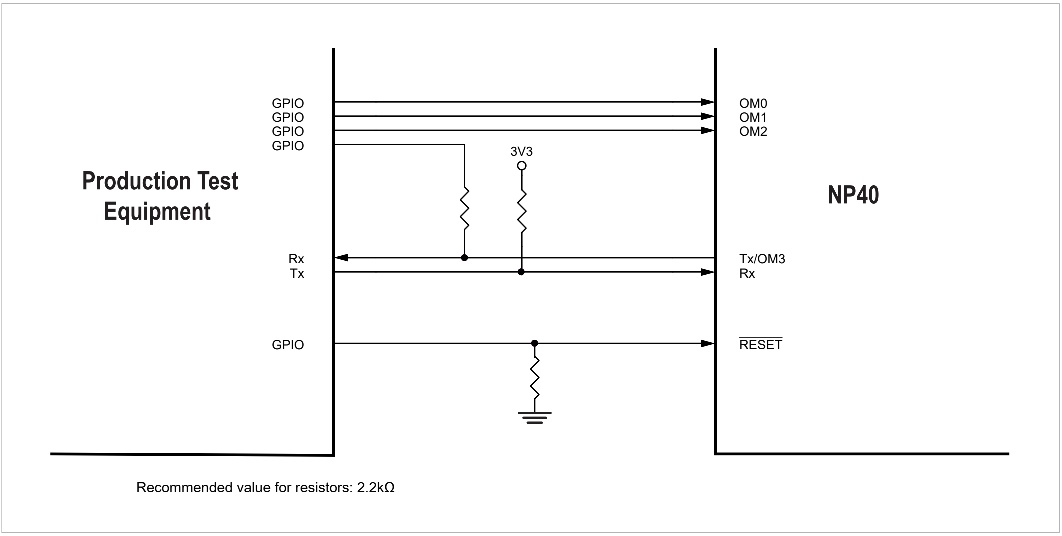

GPIO
GPIO
GPIO
GPIO
Production Test Equipment
Rx
Tx
GPIO
3V3
OM0
OM1
OM2
NP40
Tx/OM3
Rx
RESET
Recommended value for resistors: 2.2kΩ

Fig. 21

! It is very important that the hardware is prepared for access to this interface from the beginning, since it is the only way to download production parameters and firmware in an empty NP40. Some downloads are large, and it is recommended to prepare the interface for the highest supported baud rate (625kbps) to speed up the procedure. Also consider if multiple serial UART interfaces for parallel programming of multiple units shall be supported to speed up programming when in serial production.

7.4 C40 Production Parameters

Depending on what industrial network(s) the C40 shall be programmed to support, the production parameter set to be programmed differs. The production parameters binary data blob that should be programmed has the layout as described in the table below.

LE notation in the Field (Data type) column below means that the data representation is in little-endian.

BE notation in the Field (Data type) column below means that the data representation is in big-endian.

Address offset	Field (Data type)	Size (bytes)	Important!	Description
00000000h	Structure version (UINT8)	1	Always use this value!	Version of the content layout of this block of production parameters. Value: 01h = This layout
00000001h	CapabilityFlags (UINT8)	1		Bit 0: Configuration bit to enable possibility to support the RMII interface in the host interface. 0: RMII not allowed 1: RMII allowed Bit 1–7 are reserved and must all be set to 0.
00000002h	ModuleType (UINT16–LE)	2	Always use this value!	Module type identifier Value: 0403h = Anybus CompactCom 40
00000004h	GenericLockTypeID (UINT16-LE)	2		Generic lock type of NetworkType. The same hardware can in some cases be used for different networks. If this feature is to be used, the Generic Lock Type has to be set to 0000h. If the network will not be changed, set the Generic Lock Type to the corresponding industrial network value below: Values: 0000h = Unlocked 0005h = Locked to PROFIBUS 0020h = Locked to CANopen 0025h = Locked to DeviceNet 0087h = Locked to EtherCAT 0089h = Locked to PROFINET IRT (copper) 0090h = Locked to CC-Link 0093h = Locked to Modbus TCP 009Ah = Locked to BACNet/IP 009Bh = Locked to EtherNet/IP 009Dh = Locked to PROFINET IRT Fiber Optic 009Fh = Locked to POWERLINK
00000006h	ProviderID (UINT16-LE)	2	Always use this value!	Provider ID Value: 0001h = HMS Networks
00000008h	Serial number (UINT32-LE)	4		Product serial number. The serial number needs to be uniquely assigned for each produced C40 unit. **Important notes**: • Some industrial network organizations, for example ODVA, demand that the serial number must be uniquely assigned independent of internal network type, i.e. a DeviceNet node MUST NOT have the same serial number as an EtherNet/IP node. • You should ALWAYS change the Vendor ID from HMS default to your own assigned value per industrial network organization in your C40 application. This has to be done to avoid producing a C40 unit that violates an already assigned serial number from HMS.

Address offset	Field (Data type)	Size (bytes)	Important!	Description
0000000Ch	Reserved generic 2	4	Always use this value!	Reserved field for generic production parameters. Value: Set all bytes to 00h
00000010h	MAC address 1 (Array of UINT8)	6		MAC address of Ethernet interface 1. Needs to be uniquely assigned if the C40 is produced to be able to support EtherNet/IP, PROFINET IRT, Modbus TCP, or POWERLINK else all bytes should be set to 00h If EoE is used, this field has to hold a unique and correct MAC address for EtherCAT.
00000016h	Port 1 MAC address (Array of UINT8)	6		MAC address of Ethernet port 1. Needs to be uniquely assigned if the C40 is produced to be able to support PROFINET IRT else all bytes should be set to 00h
0000001Ch	Port 2 MAC address (Array of UINT8)	6		MAC address of Ethernet port 2. Needs to be uniquely assigned if the C40 is produced to be able to support PROFINET IRT else all bytes should be set to 00h
00000022h	Reserved fieldbus specific 1	94	Always use this value!	Reserved for fieldbus (industrial network) specific production parameters. Value: Set all bytes to 00h
00000080h	Target hardware descriptor	82		This descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with, see *Target Hardware Descriptor Production Data Field, p. 34* .
000000D2h	Reserved fieldbus specific 2	46	Always use this value!	Reserved for network specific production parameters. Value: Set all bytes to 0x00

7.4.1 Target Hardware Descriptor Production Data Field

The target hardware descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with. It has an address offset that is 00000080h.

Field (Data type)	Description/Value	
Number of elements (UINT16-LE)	Number of array elements following this entry. Value: Valid range: 1-20	
Descriptor array (Array of struct)	**Field (Data Type)**	**Description/Value**
	NetworkType(UINT16-LE)	Network type identifier
	HardwareFuncID(UINT16-LE)	Hardware functionality identifier

See the table below for values to use depending on what networks the produced C40 unit should be able to support.

The field holds what industrial network hardware the produced C40 unit is compatible with. It is an array and consists of a list of NetworkType and HardwareFunc identifiers. The table lists the current identifiers per network type and also some examples of how to populate the list entry depending on your C40 design capabilities.

! Depending on C40 silicon revision, the **HardwareFuncId** must be correctly assigned. If not, it may be impossible to update the firmware of the C40 chip and the chip may become bricked. See the table below for the correct **HardwareFuncId** value.

! Always check the production release information document of each firmware release for the correct production parameter settings.

Product Type	Network Type	HardwareFuncID
Generic 2-port Ethernet 100BASE-TX 10BASE-T platform Product types: BACnet/IP EtherCAT EtherNet/IP Modbus TCP POWERLINK PROFINET IRT (copper)	0300h	0002h
Generic 2-port Ethernet FO (Fiber optic) platform Product types: PROFINET IRT (fiber)	0303h	0001h
Generic RS-485 platform Product types: CC-Link PROFIBUS	0301h	0001h
Generic CAN platform Product types: DeviceNet CANopen	0302h	0001h

7.4.2 Production Data Examples

All examples below use data as defined in the table above. The tables in the examples only show values that differ from the default values given in the table above, and that have to be assigned to allow for correct production.

Example 1: PROFINET IRT only

Address offset	Field (Data type)	Size (bytes)	Value
00000004h	GenericLockTypeID (UINT16-LE)	2	0089h = Locked to PROFINET IRT (copper)
00000008h	Serial number (UINT32-LE)	4	Uniquely assigned and managed per produced C40 by the user Example: FFFFFFF0h
00000010h	MAC Address 1 (Array of UINT8)	6	Uniquely assigned and managed per produced C40 by the user. The three MAC IDs have to be consecutive, for example (xx:yy:zz:aa:bb:01), (xx:yy:zz:aa:bb:02), and (xx:yy:zz:aa:bb:03).
00000016h	Port 1 MAC Address (Array of UINT8)	6	
0000001Ch	Port 2 MAC Address (Array of UINT8)	6	
00000080h	Target hardware descriptor	82	This descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with, see table below.

Target hardware descriptor (address offset 00000080h):

Field (Data type)	Description/Value	
Number of elements (UINT16-LE)	0001h	
Descriptor array (Array of struct)	**Field (Data Type)**	**Description/Value**
	NetworkType(UINT16-LE)	0300h
	HardwareFuncID(UINT16-LE)	See table giving Target hardware descriptor values in *Target Hardware Descriptor Production Data Field, p. 34*.
Remaining 76 bytes	Set all bytes to 00h	

Example 2: EtherNet/IP only

Address offset	Field (Data type)	Size (bytes)	Value
00000004h	GenericLockTypeID (UINT16-LE)	2	009Bh = Locked to EtherNet/IP
00000008h	Serial number (UINT32-LE)	4	Uniquely assigned and managed per produced C40 by the user Example: FFFFFFF0h
00000010h	MAC Address 1 (Array of UINT8)	6	Uniquely assigned and managed per produced C40 by the user
00000016h	Port 1 MAC Address (Array of UINT8)	6	Not applicable. Set all bytes to 00h.
0000001Ch	Port 2 MAC Address (Array of UINT8)	6	
00000080h	Target hardware descriptor	82	This descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with, see table below.

Target hardware descriptor (address offset 00000080h):

Field (Data type)	Description/Value	
Number of elements (UINT16-LE)	0001h	
Descriptor array (Array of struct)	**Field (Data Type)**	**Description/Value**
	NetworkType(UINT16-LE)	0300h
	HardwareFuncID(UINT16-LE)	See table giving Target hardware descriptor values in *Target Hardware Descriptor Production Data Field, p. 34*.
Remaining 76 bytes	Set all bytes to 00h	

Example 3: All Ethernet networks accepted

Address offset	Field (Data type)	Size (bytes)	Value
00000004h	GenericLockTypeID (UINT16-LE)	2	0000h = Unlocked
00000008h	Serial number (UINT32-LE)	4	Uniquely assigned and managed per produced C40 by the user Example: FFFFFFF0h
00000010h	MAC Address 1 (Array of UINT8)	6	Uniquely assigned and managed per produced C40 by the user. The three MAC IDs have to be consecutive, for example (xx:yy:zz:aa:bb:01), (xx:yy:zz:aa:bb:02), and (xx:yy:zz:aa:bb:03).
00000016h	Port 1 MAC Address (Array of UINT8)	6	
0000001Ch	Port 2 MAC Address (Array of UINT8)	6	
00000080h	Target hardware descriptor	82	This descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with, see table below.

Target hardware descriptor (address offset 00000080h):

Field (Data type)	Description/Value	
Number of elements (UINT16-LE)	0001h	
Descriptor array (Array of struct)	**Field (Data Type)**	**Description/Value**
	NetworkType(UINT16-LE)	0300h
	HardwareFuncID(UINT16-LE)	See table giving Target hardware descriptor values in *Target Hardware Descriptor Production Data Field, p. 34*.
Remaining 76 bytes	Set all bytes to 00h	

Example 4: PROFIBUS only

Address offset	Field (Data type)	Size (bytes)	Value
00000004h	GenericLockTypeID (UINT16-LE)	2	0005h = Locked to PROFIBUS
00000008h	Serial number (UINT32-LE)	4	Uniquely assigned and managed per produced C40 by the user Example: FFFFFFF0h
00000010h	MAC Address 1 (Array of UINT8)	6	Not applicable. Set all bytes to 00h.
00000016h	Port 1 MAC Address (Array of UINT8)	6	
0000001Ch	Port 2 MAC Address (Array of UINT8)	6	
00000080h	Target hardware descriptor	82	This descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with, see table below.

Target hardware descriptor (address offset 00000080h):

Field (Data type)	Description/Value	
Number of elements (UINT16-LE)	0001h	
Descriptor array (Array of struct)	**Field (Data Type)**	**Description/Value**
	NetworkType (UINT16-LE)	0301h
	HardwareFuncID (UINT16-LE)	See table giving Target hardware descriptor values in *Target Hardware Descriptor Production Data Field, p. 34*.
Remaining 76 bytes	Set all bytes to 00h	

Example 5: All generic RS485 fieldbuses accepted

Address offset	Field (Data type)	Size (bytes)	Value
00000004h	GenericLockTypeID (UINT16-LE)	2	0000h = Unlocked
00000008h	Serial number (UINT32-LE)	4	Uniquely assigned and managed per produced C40 by the user Example: FFFFFFF0h
00000010h	MAC Address 1 (Array of UINT8)	6	Not applicable. Set all bytes to 00h.
00000016h	Port 1 MAC Address (Array of UINT8)	6	
0000001Ch	Port 2 MAC Address (Array of UINT8)	6	
00000080h	Target hardware descriptor	82	This descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with, see table below.

Target hardware descriptor (address offset 00000080h):

Field (Data type)	Description/Value	
Number of elements (UINT16-LE)	0001h	
Descriptor array (Array of struct)	**Field (Data Type)**	**Description/Value**
	NetworkType(UINT16-LE)	0301h
	HardwareFuncID(UINT16-LE)	See table giving Target hardware descriptor values in *Target Hardware Descriptor Production Data Field, p. 34*.
Remaining 76 bytes	Set all bytes to 00h	

Example 6: All generic Ethernet and generic RS485 fieldbuses accepted

Address offset	Field (Data type)	Size (bytes)	Value
00000004h	GenericLockTypeID (UINT16-LE)	2	0000h = Unlocked
00000008h	Serial number (UINT32-LE)	4	Uniquely assigned and managed per produced C40 by the user Example: FFFFFFF0h
00000010h	MAC Address 1 (Array of UINT8)	6	Uniquely assigned and managed per produced C40 by the user. The three MAC IDs have to be consecutive, for example (xx:yy:zz:aa:bb:01), (xx:yy:zz:aa:bb:02), and (xx:yy:zz:aa:bb:03).
00000016h	Port 1 MAC Address (Array of UINT8)	6	
0000001Ch	Port 2 MAC Address (Array of UINT8)	6	
00000080h	Target hardware descriptor	82	This descriptor lists the possible NetworkType and the corresponding hardware versions that the product is compatible with, see table below.

Target hardware descriptor (address offset 00000080h):

Field (Data type)	Description/Value	
Number of elements (UINT16-LE)	0001h	
Descriptor array (Array of struct)	**Field (Data Type)**	**Description/Value**
	NetworkType(UINT16-LE)	0300h
	HardwareFuncID(UINT16-LE)	See table giving Target hardware descriptor values in *Target Hardware Descriptor Production Data Field, p. 34*.
	NetworkType(UINT16-LE)	0301h
	HardwareFuncID(UINT16-LE)	See table giving Target hardware descriptor values in *Target Hardware Descriptor Production Data Field, p. 34*.
Remaining 76 bytes	Set all bytes to 00h	

7.4.3 C40 Production Data Generator Sample Application

HMS provides a sample Python GUI application that can be used to construct/edit the production data **.bin** files that are required by the C40 production scripts, if production data is to be written (see *Automated Production Scripts, p. 53*). The sample application is located in the **./proddata/** directory of the licensing package and requires **Python 3.9** together with **EasyGUI** to be installed on a PC.

Installation links:

- Python 3.9: https://www.python.org/downloads/
- EasyGUI: http://easygui.sourceforge.net/

7.5 C40 Production Test Sequences

The production tests that need to be performed are divided into two parts: tests to be performed before and tests to be performed after the firmware is downloaded to the NP40.

The tests are divided because some tests are only available in the FPGA with the preprogrammed firmware from factory (FPGA image shipped with HMS specific boot code) and these can ONLY be performed before the final product firmware is downloaded.

The product firmware specific tests, which require the specific industrial network or fieldbus construction in the FPGA and its sub-components to execute properly, must of course be performed AFTER product firmware download.

! If the wrong firmware has been downloaded to a C40, for example EtherCAT firmware instead of the wanted EtherNet/IP firmware, it will not be possible to reprogram the C40.

7.5.1 Product Firmware Download and Preceding Tests

The embedded common test cases are located in the application **/testapps/ABCC40_ CommonTests/7013-RLDS_App-ABCC40_CommonTests_X_YY_ZZ.hiff**. This file is downloaded to the C40 in service mode as described in section “Common Test Sequences” below. The **_X_YY_ZZ** suffix can differ depending on the version of the C40 licensing package. Each test case has dedicated documentation that can be consulted to obtain extended information about the test case and input/output data.

Test case ID-Name (HEXID-Name)	**Description**	**Documentation (/testapps/ABCC40_CommonTests)**
#16-ReadRegister (0010-ReadRegister)	Generic test case that can perform a read of a register inside the C40 chip. Can be used by IO pin tests with preprogrammed factory C40 firmware.	SDD-7013-022 0010-ReadRegister TCEP Test case documentation template.pdf
#26-WriteRegister (001A-WriteRegister)	Generic test case that can perform a write to a register inside the C40 chip. Can be used by IO pin tests with preprogrammed factory C40 firmware.	SDD-7013-023 001A-WriteRegister TCEP Test case documentation template.pdf
#17-FormatFATFS (0011-FormatFATFS)	Test will perform a creation and format of the FATFS in the external C40 SPI flash and scale the partition size based on mounted SPI flash size.	SDD-7013-025 0011-FormatFATFS - TCEP Test case documentation.pdf
#20-VirtualAttributes (0014-VirtualAttributes)	Test allows the programming of the virtual attributes section in the external SPI flash. For more information about "virtual attributes" concept consult the Anybus CompactCom 40 Software Design Guide	SDD-7013-026 0014-VirtualAttributes - TCEP Test case documentation.pdf
#19-BlackWhiteList (0013-BlackWhiteList)	Test allows the programming of the black/white list segment in the external SPI flash. For more information about the "black/white list" concept consult the Anybus CompactCom 40 Software Design Guide.	SDD-7013-027 0013-BlackWhiteList Test Documentation.pdf
#27-DCDCSwPwm (001B-DCDCSwPwm)	Test used to enable the external PWM for the DCDC generated ISO_5V voltage net. Required for some industrial network tests performed on factory FPGA image.	SDD-7013-030 001B-DCDCSwPwm - TCEP Test case documentation.pdf

Common Test Sequences

The table below lists the generic test sequences that are performed on all C40 units independent of what final product firmware that will be downloaded. The tests in the suite require the factory pre-programmed FPGA image to be present to work and cannot be performed on an already produced unit.

! Sample test scripts must be adapted to actual device design to avoid risk of hardware damage. Pin signal descriptions will be available upon signed license agreement.

Step / Item	**Description**	**Test Data**	**Pre-Constraints**
1. Enable power to DUT	1. Power supply shall be current-limited 2. Reset of C40 HW; after reset RLDS boot code is active in service mode.	NA	OM[0..3] pins = [1,1,1,1] (Service Mode)
2. Measure DUT current	Fail out complete test if current is to high	C40 user needs to define the limits of their construction	NA
3. Load TCEP test case file for C40 factory pre-programmed FPGA image.	HIFF file contains all TCEP test cases that can be executed on the C40 preprogrammed FPGA image.	1. RLDS load file: Test file from C40 production package: /testapps/ ABCC40_CommonTests/7013-RLDS_ App-ABCC40_CommonTests_X_YY_ ZZ.hiff 2. Protocol after RLDS load: TCEP (Test Case Execution Protocol)	NA
3.1 Execute C40 LED test for MS/NS	Test will verify that the MS/ NS HW LEDs (4 different LEDs) are working properly.	Test case: 001A-WriteRegister. • LED1A (MS Red): Should be red • LED1B (MS Green): Should be green • LED2A (NS Red): Should be red • LED2B (NS Green): Should be green Refer to sample script C40_Inc_LedTest_ CommonTests_Generic.txt.	MS/NS LEDs should be connected on the C40 product
3.2 Execute C40 LED test for Ethernet LINK/ACT	Test will verify that the LINK/ACT HW LEDs on C40 Ethernet HW are working properly.	Test case: 001A-WriteRegister. Verify ON/ OFF status for each LED. Refer to sample script C40_Inc_LedTest_CommonTests_ GenericEthernet.txt .	C40 needs to be an Ethernet HW design and LINK/ ACT needs to be connected. Also, see important note above table.
3.3 Execute C40 application connector test	This test should physically verify that the connector pins are connected as expected.	Test case: 001A-WriteRegister. Verify HIGH/LOW status for each connector pin. Refer to sample scripts C40_Inc_ AppConnTest_CommonTests *.txt	Tested pins depend on C40 design and what is to be verified. Also, see important note above table.

Step / Item	Description	Test Data	Pre-Constraints
3.4 Program default black/white list in C40 external SPI flash	Test will program the black/ white list segment in the SPI flash. The default data will create an empty black list (i. e. no network types are blocked by the application)	Test case: 0013-BlackWhiteList Test input data (empty black list, binary hex): 00h 00h Test output data (for success): Test case status code: 00000000h	NA
3.5 Program default virtual attributes in C40 external SPI flash.	Test with the default test data will program an empty list. If the user wants to have another list this is possible and the test case documentation should be consulted.	Test case: 0014-VirtualAttributes Test input data (binary hex): FFh 0000h 0000h 0000h Test output data (for success): Test case status code: 00000000h	NA
3.6 Format FATFS file system in C40 external SPI flash	Test will perform a creation and format of the FATFS in the external C40 SPI flash and scale the partition size based on mounted SPI flash size.	Test case: 0011-FormatFATFS Test input data (binary hex): NA Test output data (for success): Test case status code: 00000000h	NA
3.7 Execute product specific test sequences that needs factory FPGA image for C40 HW: 1. PROFIBUS, CC-Link 2. DeviceNet	Allows product specific test sequences to be executed on factory FPGA image.	1. PROFIBUS, CC-Link: see “Product Tests for: PROFIBUS, CC-Link” below. 2. DeviceNet: See “Product Tests for: DeviceNet” below.	NA
4. Trigger C40 HW reset or power-cycle	Reset of C40 HW; after reset RLDS boot code is active in service mode.	NA	OM[0..3] pins = [1,1,1,1] (Service Mode)
5. Program C40 Golden recovery image that contains C40 boot code to external SPI flash.	Will enable the C40 chip to automatically recover itself to factory mode in case of FW download failure.	1. RLDS load file: Firmware file from C40 production package: /firmware/7010-GI-RAMLoaderDS-M2S010-1VF400-PartTable0_2_02_B.hiff. Check the production release information for the correct RLDS file. 2. Protocol after RLDS load: FWLoaderDS 2.1. Program FSTM streams in HIFF file 2.2. Finalize FSTM streams 3. During sequences in (2) above the MS LED will flash red/green with a 250 ms interval whenever a flash writing command has been executed.	C40 must be out-of-box and NOT have any production data previously set.
6. Trigger C40 HW reset or power-cycle	Reset of C40 HW; after reset RLDS boot code is active in service mode.	NA	OM[0..3] pins = [1,1,1,1] (Service Mode)
7. Program C40 final product firmware and production data	Will reprogram the whole C40 chip, except the boot code, to the desired industrial network. It will also, on a successful completion of the FW download, write the user provided production data in the C40.	1. RLDS load file: Firmware file from C40 production package: /firmware/ABCC_40_XXX_XXXX_X_YY_BB.hiff 2. Protocol after RLDS load: FWLoaderDS 2.1. Program FSTM streams in HIFF file 2.2. Set production data with FLDS command 0x06 2.3. Finalize FSTM streams 3. During sequences in (2) above the MS LED will flash red/green with a 250 ms interval whenever a flash writing command has been executed.	C40 must be out-of-box and NOT have any production data previously set.
8. Trigger C40 HW reset or power-cycle	Reset of C40 HW; after reset RLDS boot code is active in service mode.	NA	OM[0..3] pins = [1,1,1,1] (Service Mode)
9. Perform product firmware tests; sequences differ depending on programmed firmware in (7)	See *Product Firmware Tests (after Product Firmware Download), p. 45*	NA	NA

Product Tests for: PROFIBUS, CC-Link

The tests below allow the physical pins of the network interface to be verified in a generic electrical manner.

Step/Item	**Description**	**Test Data**	**Pre-Constraints**
3.7. Enabled SW controlled DCDC PWM to create the ISO_5V voltage net.	To test the physical external fieldbus pins, the ISO_5V voltage net needs to be enabled. After this test is executed successfully the voltage net should be available.	Test case: 001B-DCDCSwPwm Test input data (binary hex): 01h Test output data (for success): Test case status code: 00000000h	NA
3.8 Test fieldbus pin		Test case: 001A-WriteRegister.	Step 3.7 needs to be executed successfully.
3.9. Disable SW controlled DCDC PWM to create the ISO_5V voltage net.	Disables the ISO_5V voltage net.	Test case: 001B-DCDCSwPwm Test input data (binary hex): 00h Test output data (for success): Test case status code: 00000000h	NA

Product Tests for: DeviceNet, CANopen

The tests below allow the physical pins of the network interface to be verified in a generic electrical manner.

Step/Item	**Description**	**Test Data**	**Pre-Constraints**
3.7. Enabled SW controlled DCDC PWM to create the ISO_5V voltage net.	To test the physical external fieldbus pins, the ISO_5V voltage net needs to be enabled. After this test is executed successfully the voltage net should be available.	Test case: 001B-DCDCSwPwm Test input data (binary hex): 01h Test output data (for success): Test case status code: 00000000h	NA
3.8 Test fieldbus pin		Test case: 001A-WriteRegister.	Step 3.7 needs to be executed successfully.
3.9. Disable SW controlled DCDC PWM to create the ISO_5V voltage net.	Disables the ISO_5V voltage net.	Test case: 001B-DCDCSwPwm Test input data (binary hex): 00h Test output data (for success): Test case status code: 00000000h	NA

7.5.2 Product Firmware Tests (after Product Firmware Download)

The embedded product firmware specific tests cases are located in the application **/testapps/ABCC40_ProductFirmwareTests/7013-RLDS_App-ABCC40_ProductFirmwareTests_X_YY_ZZ.hiff**. This file is downloaded to the C40 in service mode as described in sub chapters below per network. The **_X_YY_ZZ** suffix can differ depending on the version of the C40 licensing package. Each test case has dedicated documentation that can be consulted to get extended information about what the test case does and about input/output data.

Test Case ID-Name (HEXID-Name)	**Description**	**Documentation (/testapps/ABCC40_ProductFirmwareTests)**
#16-ReadRegister (0010-ReadRegister)	Generic test case that can perform a read of a register inside the C40 chip.	SDD-7013-022 0010-ReadRegister TCEP Test case documentation template.pdf
#26-WriteRegister (001A-WriteRegister)	Generic test case that can perform a write to a register inside the C40 chip.	SDD-7013-023 001A-WriteRegister TCEP Test case documentation template.pdf
#18-SRAMTest (0012-SRAMTest)	Test case that can perform an SRAM test if an SRAM is connected to the C40 (Ethernet hardware).	SDD-7013-024 0012-SRAM TCEP Test case documentation.pdf
#23-ECT_test (0017-ECT_test)	Network specific test for EtherCAT.	SDD-7013-019 0017-EtherCAT TCEP Test case documentation.pdf
#24-EPL_test (0018-EPL_test)	Network specific test for POWERLINK.	SDD-7013-018 0018-EPL - TCEP Test case documentation.pdf
#25-ETN_test (0019-ETN_test)	Network specific test for: EtherNet/IP, BACnet/IP, Modbus TCP and PROFINET IRT.	SDD-7013-021 0019-ETN - TCEP Test case documentation.pdf

Test Sequences for: EtherNet/IP, BACnet/IP, Modbus TCP and PROFINET IRT

Step/Item	Description	Test Data	Pre-Constraints
9. Load TCEP test case file for C40 product firmware.	HIFF file contains all TCEP test cases that can be executed on the different C40 product firmwares.	1. RLDS load file: Test file from C40 production package: /testapps/ABCC40_ ProductFirmwareTests/7013-RLDS_App-ABCC40_ ProductFirmwareTests_X_YY_ZZ.hiff 2. Protocol after RLDS load: TCEP (Test Case Execution Protocol)	NA
9.1 Execute external SRAM test	Test will perform a memory test on the external SRAM that is connected to the C40 and used by the product firmware.	Test case: 0012-SRAMTest Test input data (binary hex): 88000000h 00080000h 10h Test output data (for success): Test case status code: 00000000h	NA
9.2 Execute fieldbus test: Generic Ethernet	This test can be used to check the LINK status on each of the two Ethernet ports and can also be queried to check if any packets have been received on a specific MAC address per port.	Test case: 0019-ETN_test 1. Test step: Initialize test Input data (binary hex): 00h 90000000h 98000000h Test output data (for success): Test case status code: 00000000h 2. Test step: Detect link and frame on Port1 on embedded side a. Ask user to connect an Ethernet cable to Port1 and send a frame to address 00:30:11:00:00:01 Input data (binary hex): 02h 00000000h 00000000h Test output data (for success): Test case status code: 00000000hTranspData. bLinkStatus = 01h TranspData.lReceivedPackets = 00000001h 3. Test step: Detect link and frame on Port2 on embedded side a. Ask user to connect an Ethernet cable to Port2 and send a frame to address 00:30:11:00:00:01 Input data (binary hex): 03h 00000000h 00000000h Test output data (for success): Test case status code: 00000000hTranspData. bLinkStatus = 01h TranspData.lReceivedPackets = 00000001h 4. Test step: Send a frame not to DUT on Port1/ Port2 and observe the frame on the opposite port. a. Send a frame on each of the ports and verify that the same frame is received on the opposite port. 5. Test step: Stop test Input data (binary hex): FFh 00000000h 00000000h Test output data (for success): Test case status code: 00000000hTranspData. bLinkStatus = 00h TranspData.lReceivedPackets = 00000000h	1. Test client needs to be able to send a packet to a specific MAC address before the test is queried to check for the packet. 2. Ethernet cables needs to be connected to the Ethernet ports to observe link/ received packets. 3. See test case documentation.

Test Sequences for: EtherCAT

Step/Item	Description	Test Data	Pre-Constraints
9. Load TCEP test case file for C40 product firmware.	HIFF file contains all TCEP test cases that can be executed on the different C40 product firmwares.	1. RLDS load file: Test file from C40 production package: /testapps/ABCC40_ ProductFirmwareTests/7013-RLDS_App-ABCC40_ ProductFirmwareTests_X_YY_ZZ.hiff 2. Protocol after RLDS load: TCEP (Test Case Execution Protocol)	NA
9.1 Execute external SRAM test	The test will perform a memory test on the external SRAM that is connected to the C40 and used by product firmware.	Test case: 0012-SRAMTest Test input data (binary hex): 88000000h 00080000h 10h Test output data (for success): Test case status code: 00000000h	N/A
9.2 Execute fieldbus test: EtherCAT	This test can be used to verify that the EtherCAT interface is working properly. The test case enables EtherCAT. The test client should after this perform the sequences described in the test case documentation.	Test case: 0017-ECT_test Test input data: NA Test output data: Test case status code: 00000000h	See test case documentation

Test Sequences for: POWERLINK

Step/Item	Description	Test Data	Pre-Constraints
9. Load TCEP test case file for C40 product firmware.	HIFF file contains all TCEP test cases that can be executed on the different C40 product firmwares.	1. RLDS load file: Test file from C40 production package: /testapps/ABCC40_ ProductFirmwareTests/7013-RLDS_App-ABCC40_ ProductFirmwareTests_X_YY_ZZ.hiff 2. Protocol after RLDS load: TCEP (Test Case Execution Protocol)	NA
9.1 Execute external SRAM test	Test will perform a memory test on the external SRAM that is connected to the C40 and used by the product firmware.	Test case: 0012-SRAMTest Test input data (binary hex): 88000000h 00080000h 10h Test output data (for success): Test case status code: 00000000h	NA
9.2 Execute fieldbus test: POWERLINK	This test can be used to verify that the POWERLINK interface is working properly. The test case itself is used to detect a POWERLINK frame that the test client produces.	Test case: 0018-EPL_test Test input data: 00000000h (Timeout in ms) Test output data (success): Test case status code: 00000000h	See test case documentation

7.6 Developer GUI Interface

The production test sample application contains a Windows GUI interface that can be used to manually execute TCEP (Test Case Execution Protocol) test cases on the embedded C40 target. This is useful when you want to manually execute a specific test case to back trace a production failure or test some custom production data that should be deployed. The test sequences to perform are described in *C40 Production Test Sequences, p. 41* and with the help of this table you can manually iterate through the production steps.

7.6.1 Basic GUI Interface

The basic GUI interface is started by executing the standalone **C40 ProdTest Sample App.exe** application in the root directory of the C40 licensing production package:

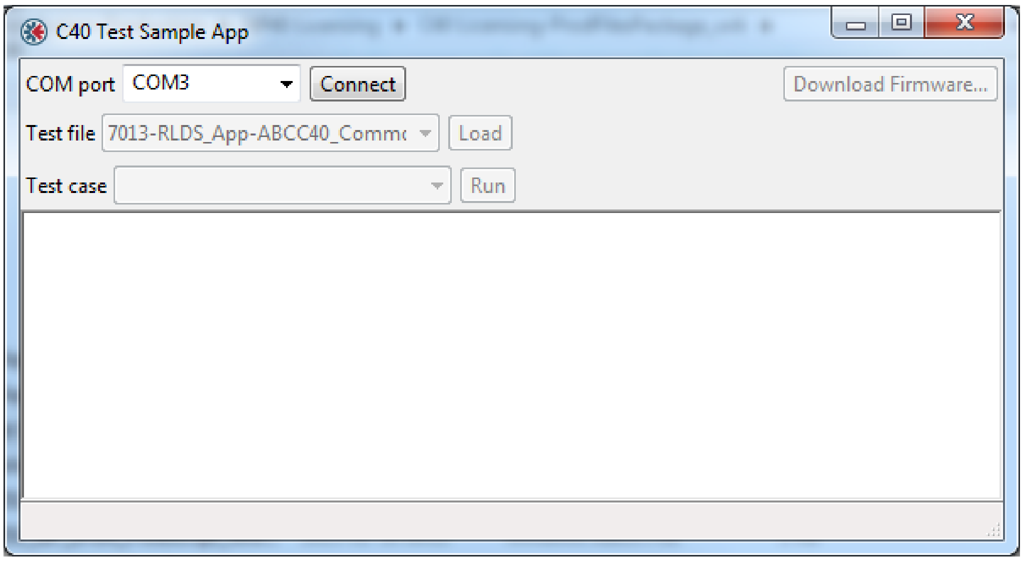


Fig. 22

1. Select the COM port the C40 is connected to.

   The RS232 interface of the PC must be connected to the C40 pins via an RS232 transceiver (see NP40 Pinning.pdf):

   – H_RX/ASM_RX

   – H_TX/ASM_TX/OM3

     OM3 has an internal weak pull-up inside C40. Alternatively an external pull-up can be used.

2. C40 should be put in: Service Mode

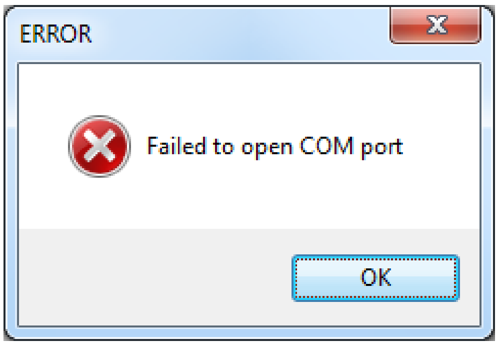
   – Set OM[0..3] inputs to [1,1,1,1]

   – Perform a hardware reset or power cycle of C40

3. Click **Connect**

4. If the C40 has not entered Service Mode or a COM port error exists the following message will appear:

   If this happens, check the COM port connected to the C40 for errors.


ERROR

Failed to open COM port

Fig. 23

5. If the connection was successful the drop-down box next to “Test file” will be enabled.

7.6.2 Loading Embedded Test Applications

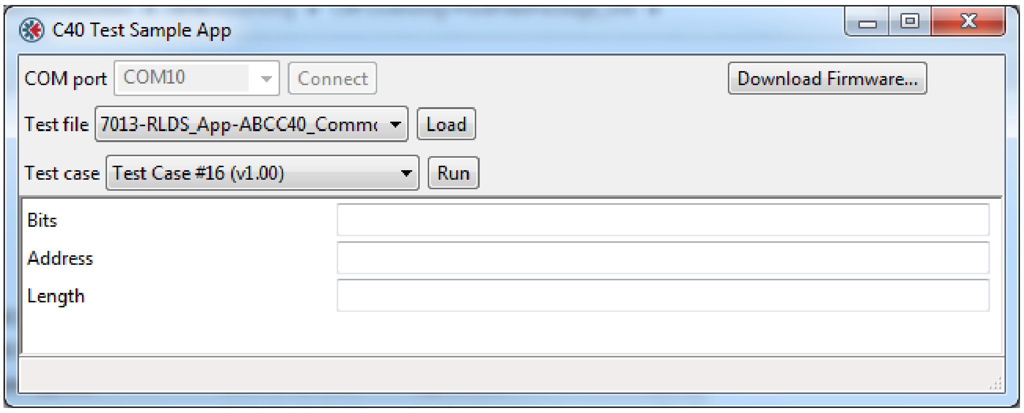
When the connection to the C40 is established, you can download an embedded test application to the C40 to execute test cases. In *C40 Production Test Sequences, p. 41* the available test applications and cases are described. Select one of the following files:

- 7013-RLDS_App-ABCC40_CommonTests_X_YY_XX.hiff

  This test application is used for common tests, performed **before** product firmware download, see *Product Firmware Download and Preceding Tests, p. 41*.

- 7013-RLDS_App-ABCC40_ProductFirmwareTests_X_YY_ZZ.hiff

  This test application is used for tests performed **after** product firmware download, *Product Firmware Tests (after Product Firmware Download), p. 45*


C40 Test Sample App

COM port COM10

Test file 7013-RLDS_App-ABCC40_Comm

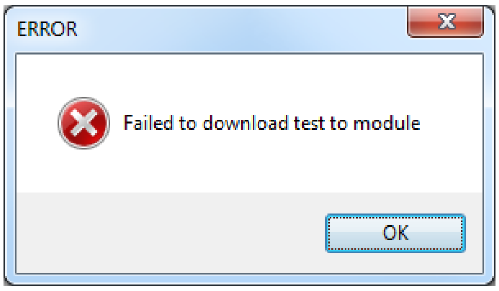
Test case Test Case #16 (v1.00)

Fig. 24

1. Select one of the presented test applications an click **Load**.

2. The test application will take 5 - 7 seconds to load. The GUI uses serial baud rate 57600 bps. When ready, you should see the view shown above, with the “Test case” drop down box enabled.

   If an error occurs, the following message will appear and you should check the connection to the C40 for errors.


ERROR

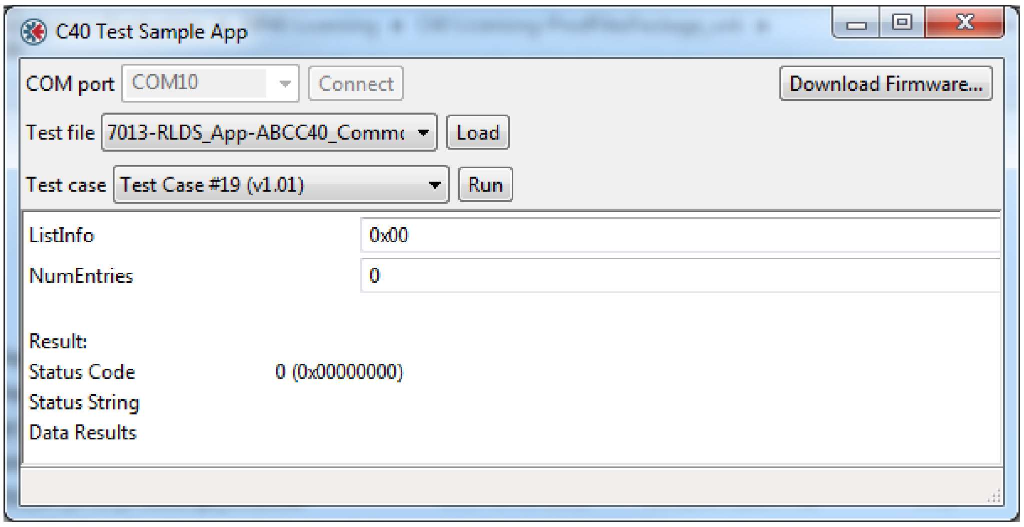
Failed to download test to module

Fig. 25

3. Each embedded test case reports the metadata of its input and output parameters. This information is described for each test case in *Product Firmware Download and Preceding Tests, p. 41* and *Product Firmware Tests (after Product Firmware Download), p. 45*.

4. Select which test case to execute in the “Test case” drop-down box and enter input parameter data (if available) in the fields that appear when the test case is selected.

5. Click **Run** to execute the test case. When finished, the output result will be presented below the input parameters in the application window. An example of the result of a successful response is shown in the picture below.



C40 Test Sample App

COM port: COM10 | Connect | Download Firmware...

Test file: 7013-RLDS_App-ABCC40_Comm | Load

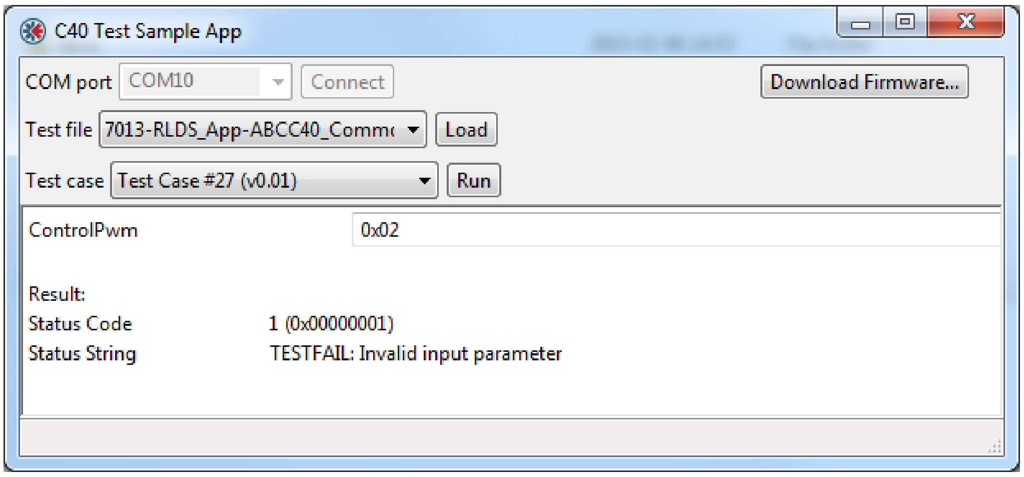
Test case: Test Case #19 (v1.01) | Run

ListInfo	0x00
NumEntries	0

```
Result:
Status Code          0 (0x00000000)
Status String
Data Results
```

Fig. 26

The following pictures show responses to failed tests.



C40 Test Sample App

COM port: COM10 | Connect | Download Firmware...

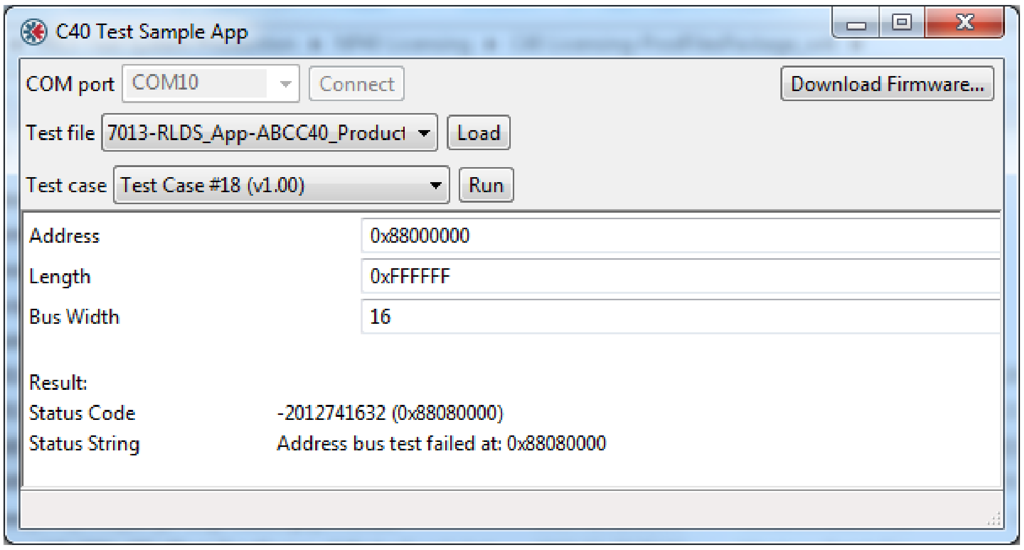
Test file: 7013-RLDS_App-ABCC40_Comm | Load

Test case: Test Case #27 (v0.01) | Run

ControlPwm	0x02

```
Result:
Status Code          1 (0x00000001)
Status String        TESTFAIL: Invalid input parameter
```

Fig. 27



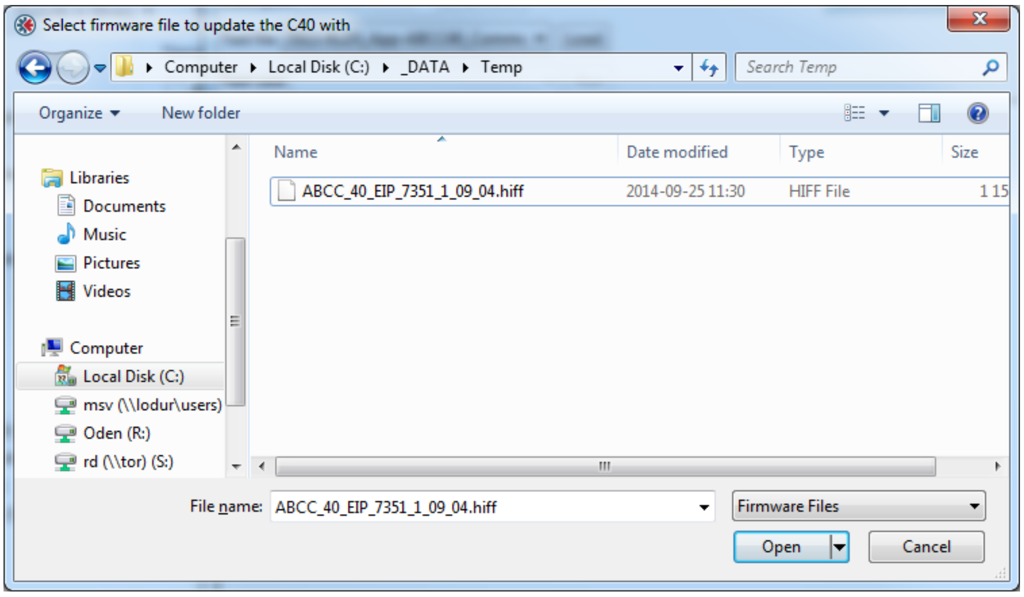
```
Address         0x88000000
Length          0xFFFFFF
Bus Width       16

Result:
Status Code     -2012741632 (0x88080000)
Status String   Address bus test failed at: 0x88080000
```

Fig. 28

7.6.3 Downloading Product Firmware

If the user wants to download a product firmware .HIFF file delivered from HMS to the C40 this can be done by clicking the **Download Firmware** button and select the .HIFF file to download. It does not matter if production data has been written or not for this feature to work.

! If no production data has been written, any **.hiff** firmware file will be accepted by the C40. Use this feature with care.



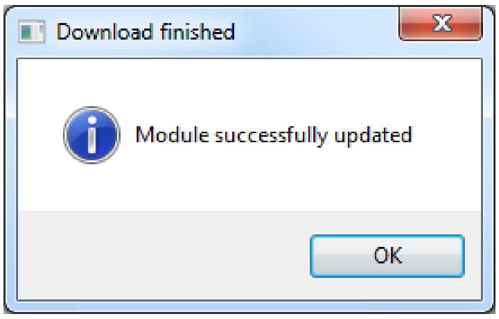
Name	Date modified	Type	Size
ABCC_40_EIP_7351_1_09_04.hiff	2014-09-25 11:30	HIFF File	1 15

File name: ABCC_40_EIP_7351_1_09_04.hiff — Firmware Files

Fig. 29

1. Select firmware and click “Open” to start the download process.
2. The serial COMM speed is 57600 bps. The download will typically be ready in about ~5 minutes.
3. During the firmware download process the MS LED connected to the C40 will flash red/ green in an interval of 250ms whenever a programming command has been executed.

4. At a successful download, the following window will appear:



Module successfully updated

Fig. 30

7.7 Automated Production Scripts

The production test sample application contains a script engine that can be used to automate some of the production test sequences described in *C40 Production Test Sequences, p. 41*. Example scripts are also provided per industrial network type, where the absolutely minimum required sequences are present to perform a production of the C40 chip.

The source code of the sample application is available in the C40 licensing production package and it is possible to extend the scripting language for specific needs.

7.7.1 Execution of a Production Script

To execute a production script, pass arguments to the sample application “C40 ProdTest Sample App.exe” in the root directory of the C40 licensing production package. It is recommended to perform the manual steps described in *Basic GUI Interface, p. 48*, to verify the serial connection.

1. Before a production script is executed the C40 must be put in **Service Mode**:
   - Set OM[0..3] inputs to [1,1,1,1]
   - Perform a hardware reset or power cycle of C40

2. Execute the script

   a. An example of command line syntax when executing a production script.

      `C40 ProdTest Sample App.exe` **`A1-COM A2-SCRIPT [A3-SPEED] [A4-QUIET]`**

Id no.	Argument	Description	Example/Comment
A1	COM	Local serial COM port to use. Syntax: COMx where x gives the number of the port.	Connection to COM port 10: COM10.
A2	SCRIPT	The path where to find the script to execute. It can be a full path or a relative part.	Full path: C:/scripts/C40_ABCC_40_XXX.txt Relative path: ./scripts/C40_ABCC_40_XXX.txt
A3	SPEED	Default: 57600 (bps) Optional. Gives the serial speed in bps for loading test applications and for firmware download.	Optional. For maximum speed for firmware download, a serial baud rate of 625000 bps is recommended, if the PC COM port is capable (for example a USB high speed serial dongle).
A4	QUIET	If the string "-quiet" is present, no success/error response GUI message boxes will be shown and the exit code of the application has to be examined to determine the result.	Optional

      There are different example scripts available. Each include the minimum required sequences, depending on module to produce. Make a copy of the correct file from the scripts folder and customize it with the correct firmware file name etc.

   b. Command line example to execute a production script on PC **COM port 10** with baud rate **625000:**

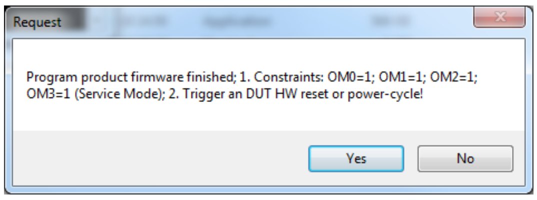
```
C40 ProdTest Sample App.exe COM10 ./scripts/C40_ABCC_40_XXX.
txt 625000
```

   c. Command line example to execute a production script on PC **COM port 10** with baud rate **625000** in **quiet** mode

```
C40 ProdTest Sample App.exe COM10 ./scripts/C40_ABCC_40_XXX.
txt 625000 -quiet
```

3. During script execution a production script can ask for user input via the GUI message boxes. Whenever a script has the **REQUEST** macro inside this will occur.

   Note that regardless of whether the script is executed in quiet mode or not this input will be required.


Request

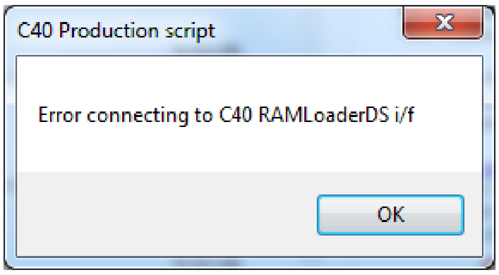
Program product firmware finished; 1. Constraints: OM0=1; OM1=1; OM2=1; OM3=1 (Service Mode); 2. Trigger an DUT HW reset or power-cycle!

Fig. 31

Script Execution Responses

If a production script is not executed in quiet mode, it will finish with displaying one of the following message boxes:

- The script was unable to connect to the C40 via the PC serial COM port:



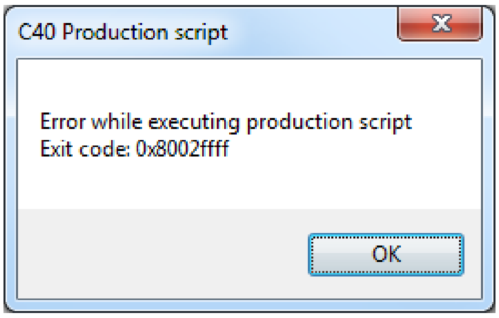
C40 Production script

Error connecting to C40 RAMLoaderDS i/f

Fig. 32

If this message appears check the connection for errors.

- The script failed when executing a command. The exit code specifies where the script failed:



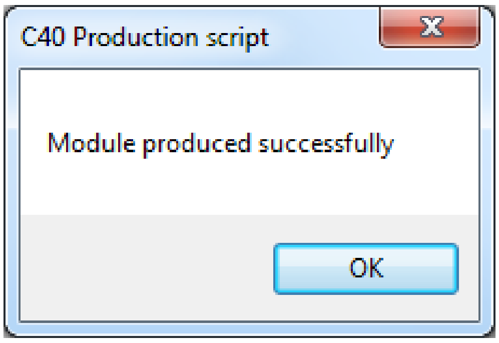
C40 Production script

Error while executing production script
Exit code: 0x8002ffff

Fig. 33

In this example, the embedded test application failed to load due to a script execution error. See *Exit Code Definition, p. 55*, for exit code definitions.

- If the script succeeded the following message box will appear:



C40 Production script

Module produced successfully

Fig. 34

In quiet mode no message boxes are displayed. In that case, examine the exit code of the application instead.

7.7.2 Exit Code Definition

The exit code that is returned by the script application and shown in the error message box when not in quiet mode has the following syntax:

Exit Code	Description
00000000h	Production script executed successfully
8ZZZXXXXh	Production script not executed successfully. • ZZZ: Internal error event code, see table below for more information. • XXXX: Embedded test case running when error occurred. Test cases are run with the TEST 0xXXXX macro. If FFFFh is returned, the error is not linked to a specific test case.

Error event code (ZZZ)	Description	Error Location in Script
0	Success	NA
1	Failed to load script file	Check argument **A2-SCRIPT** described in section *Execution of a Production Script, p. 53*.
2	Failed to load an embedded test HIFF file inside script.	Inside executed script check **FILE** load macros; see *Scripting Language, p. 57*
3	Internal error in test protocol	Internal error in test protocol; not a common error and linked to the protocol itself. Contact HMS for further analyze.
4	TestID not found in loaded embedded HIFF file.	In one of the loaded HIFF files with **FILE** macro, a test that was loaded with **TEST 0xXXXX** macro, was not found. The specific TestID can be found in the XXXX portion of the returned exit code.
5	Input variable enumeration name provided to TestID not found.	A test case executed with **TEST 0xXXXX** macro was not able to find the input variable provided by test script. Example line in script: **TEST 0xXXXX** TestVar=1 In the example above variable TestVar was not found among the input parameters. The specific TestID can be found in the XXXX portion of the returned exit code.
6	Syntax error	Returned if script data does not follow defined syntax; see section *Scripting Language, p. 57*. for scripting language. The specific TestID can be found in the XXXX portion of the returned exit code.
7	Test failure	An executed test case with macro **TEST 0xXXXX** failed at the embedded target side. The specific TestID can be found in the XXXX portion of the returned exit code.
8	Firmware download failed	A firmware download command performed by **FIRMWARE** macro inside the script failed.
9	Failed to create temporary test script file	Unable to create temporary script file in the current working directory.

7.7.3 Scripting Language

This section describes the scripting language and its syntax. The available macros are presented in the table below.

General notes:

- All macros and parameters are separated by whitespaces.
- Each new macro should be placed on a new line.
- If a semicolon, “;”, is prepended on a line in a script that line will be treated as a comment.

Macro [P1 P2 ..]	**Description**	**Parameters**	**Pre-constraints**
FILE **P1**	Loads an embedded test case HIFF file to target.	P1: Full or relative path to test case file Full path example: c:/c40_test/scripts/c40_test_script.txt Relative path example: ./scripts/c40_test_script.txt	NA
TEST **P1**	Prepares a test case with identifier **P1** to be executed.	P1: Test case identifier Test case identifier in format 0x**XXXX** where XXXX is the same id as described in "Product Firmware Download and Preceding Tests" on page 41 and "Product Firmware Tests (after Product Firmware Download)" on page 45.	**FILE** macro that has loaded an embedded test file containing **P1** TestID.

SET **P1=X**	Sets an input parameter of the test case to value X. X can be in the format as described by each test case documentation.	P1: Input parameter to test case
		Input parameter **P1** will be assigned value **X**.
		Example (TestVar = UINT8):
		SET TestVar=10
			TEST macro needs to be executed for a test case that has the input parameter named P1.
			Note: If SET_HEX has been used for corresponding test case SET will not work.

SET_HEX **P1 P2.. Px**	Builds the test case input data directly from **Px** inputs	Px: Hexadecimal number 8 bit Px will be interpreted as an 8 bit hexadecimal input and packed into the test case input parameter data array. Example: Value 0x1234 (UINT16): 34 12	**TEST** macro needs to be executed for a test case. Note: If SET has been used for corresponding test case SET_HEX will not work.
CHECK	Executes a test case that was loaded by the **TEST** macro and verifies that a success response is returned. If a failure was detected the script will exit and return corresponding exit code.	NA	1. **TEST** macro needs to be executed for a test case. 2. If input parameters are required **SET** or **SET_HEX** macros needs to be provided.
REQUEST **P1 P2**	Displays a GUI message box to the user of the script with **P1** as success method and **P2** as displayed test message.	P1: Message box user success input OK : Ok/Cancel message box where OK is success selection CANCEL : Ok/Cancel message box where Cancel is success selection. YES : Yes/No message box where Yes is success selection. NO : Yes/No message box where No is success selection. P2: Displayed text message Example: This is a test message	NA

Macro [P1 P2 ..]	Description	Parameters	Pre-constraints
FIRMWARE **"P1"** **[P2]**	Performs a firmware download of the HIFF firmware file provided in **P1**. Optional attribute **P2** can be provided and should point to the production data file to be written to C40 when firmware download is finished. Note: During firmware download the MS LED (if) connected to the C40 will flash red/green in a 250 ms interval when an internal programming command has been executed.	**P1: Full or relative path to HIFF firmware file** Note: Remember that the quotation marks "" MUST be around this parameter string! Full path example: "c:/c40_test/firmware/ABCC_40_EIP_7351_1_09_04.hiff" Relative path example: "./firmware/ABCC_40_EIP_7351_1_09_04.hiff" **[P2: Full or relative path to production data file]** Full path example: c:/c40_test/proddata/proddata.bin Relative path example: ./proddata/proddata.bin	If production data is provided to **P2** the target C40 unit **MUST be out-of-box** and never have any production data written to it. Otherwise it can become bricked and need special commissioning by HMS.
INCLUDE **P1**	Include another file containing script sequences into **this** script file. Note: The Sample application creates a temporary script file in the current working directory. The temporary file contains a flat set of instructions from top level script file and all included scripts.	**P1: Full or relative path to a file containing a script sequence** Full path example: c:/c40_test/otherScript.txt Relative path example: ./scripts/otherScript.txt The path is relative to the current working directory, i.e. from where the application is launched.	N/A

Example of Macro Usage in a Script

In the example below, a HIFF test file is loaded and some test cases are executed on the embedded target. After this a firmware file is downloaded together with production data. The test cases and the input parameter data used can be interpreted using the script syntax table in:

- *Scripting Language, p. 57*
- *Product Firmware Download and Preceding Tests, p. 41*
- *Product Firmware Tests (after Product Firmware Download), p. 45*

Command lines shall not be broken in code. Any line breaks in FILE and FIRMWARE commands, added for layout reasons in the examples below, shall be removed prior to running the scripts.

Example 1:

```
; 3. Load TCEP test case file for C40 factory pre-programmed FPGA
; image.
FILE ./testapps/ABCC40_CommonTests/
7013-RLDS_App-ABCC40_CommonTests_1_00_01.hiff
; 3.4 Program default black/white list in C40 external SPI flash
TEST 0x0013
   SET_HEX 00 00
   CHECK
; 3.5 Program default virtual attributes in C40 external SPI flash.
TEST 0x0014
   SET_HEX FF 0000 0000 0000
   CHECK
; 7. Program C40 final product firmware and production data
FIRMWARE "./firmware/ABCC_40_EIP_7351_1_09_04.hiff" ./
proddata/proddata.bin
```

Example 2:

```
; 3. Load TCEP test case file for C40 factory pre-programmed FPGA
; image.
FILE ./testapps/ABCC40_CommonTests/
7013-RLDS_App-ABCC40_CommonTests_1_00_01.hiff
; 3.4 Program default black/white list in C40 external SPI flash
TEST 0x0013
   SET NumEntries=0
   SET ListInfo=0
   CHECK
; 3.5 Program default virtual attributes in C40 external SPI flash.
TEST 0x0014
   SET Command=255
   SET ActualSize=0
   SET Offset=0
   SET FragmentSize=0
...CHECK
; 7. Program C40 final product firmware and production data
FIRMWARE "./firmware/ABCC_40_EIP_7351_1_09_04.hiff" ./
proddata/proddata.bin
```

7.7.4 Example Production Scripts

Example production scripts are provided in the C40 licensing production package that can be used to quickly get up to speed when developing prototypes for early development. Based on what industrial network(s) the C40 should be able to support, different scripts are available.

A sample production data file is also provided per script that has most of the correct configurations set. However some of the parameters must be changed per produced C40 unit such as MAC addresses and serial numbers. Consult the information available in*C40 Production Parameters, p. 32*, to properly create a correct production data file.

The example scripts and production data files can be found in the **./proddata/** and **./scripts/** folders of the C40 Licensing package. Remember that each production data file shall be edited with the provided **ProductionParametersFileGenerator.pyw** python script application to customize the production data, see *C40 Production Parameters, p. 32*.

! Remember that the production data can ONLY be written once to a C40 unit, so great care needs to be taken when generating and downloading it.

Note that not all of the sequences described in *C40 Production Test Sequences, p. 41* are performed by the scripts as some tests require test client interaction which is not possible to achieve with the scripts. Such tests should be added when integration is made with the intended final production platform.

The available examples are listed in the table below. Each script has its corresponding **.bat** file in the same folder, that demonstrates how to execute the script.

Differences of the "Production Type" per C40 Target Industrial Network:

- Full production: Generic xxx
  - Full production with tests, firmware download and programming of production data.
  - Production data is set to accept all network firmware on the current platform.
- Full production: Locked xxx
  - Full production with tests, firmware download and programming of production data.
  - Production data is set to lock C40 against the corresponding target industrial network

- Re-test and firmware download
  - Tests that can be executed on the final firmware are executed and a firmware download is performed
- Firmware download only
  - Target industrial network firmware is only downloaded

C40 target industrial network **Production type**	**Production script** **(Location: ./scripts/)**	**Production data file** **(Location: ./proddata/)**
PROFINET IRT (copper) Full production: Generic Ethernet	C40_ABCC_40_PIR_7258_ProdData_GenericEthernet.txt	C40_ABCC_40_ProdData_GenericEthernet.bin
PROFINET IRT (copper) Full production: Locked to PROFINET	C40_ABCC_40_PIR_7258_ProdData_Locked.txt	C40_ABCC_40_PIR_7258_ProdData_Locked.bin
PROFINET IRT (copper) Re-test and firmware download	C40_ABCC_40_PIR_7258_ReTest.txt	NA
PROFINET IRT (copper) Firmware download only	C40_ABCC_40_PIR_7258_FwUpdate.txt	NA
PROFINET IRT (fiber optic) Full production: Generic Ethernet FO	C40_ABCC_40_PIR_FO_7360_GenericEthernetFO.txt	C40_ABCC_40_ProdData_GenericEthernetFO.bin
PROFINET IRT (fiber optic) Full production: Locked to PROFINET FO	C40_ABCC_40_PIR_FO_7360_ProdData_Locked.txt	C40_ABCC_40_PIR_FO_7360_ProdData_Locked.bin
PROFINET IRT (fiber optic) Re-test and firmware download	C40_ABCC_40_PIR_FO_7360_ReTest.txt	NA
PROFINET IRT (fiber optic) Firmware download only	C40_ABCC_40_PIR_FO_7360_FwUpdate.txt	NA
BACnet/IP Full production: Generic Ethernet	C40_ABCC_40_BIP_7343_ProdData_GenericEthernet.txt	C40_ABCC_40_ProdData_GenericEthernet.bin
BACnet/IP Full production: Locked to BACnet/IP	C40_ABCC_40_BIP_7343_ProdData_Locked.txt	C40_ABCC_40_BIP_7343_ProdData_Locked.bin
BACnet/IP Re-test and firmware download	C40_ABCC_40_BIP_7343_ReTest.txt	NA
BACnet/IP Firmware download only	C40_ABCC_40_BIP_7343_FwUpdate.txt	NA
EtherNet/IP Full production: Generic Ethernet	C40_ABCC_40_EIP_7351_ProdData_GenericEthernet.txt	C40_ABCC_40_ProdData_GenericEthernet.bin
EtherNet/IP Full production: Locked to EtherNet/IP	C40_ABCC_40_EIP_7351_ProdData_Locked.txt	C40_ABCC_40_EIP_7351_ProdData_Locked.bin
EtherNet/IP Re-test and firmware download	C40_ABCC_40_EIP_7351_ReTest.txt	NA
EtherNet/IP Firmware download only	C40_ABCC_40_EIP_7351_FwUpdate.txt	NA
Modbus/TCP Full production: Generic Ethernet	C40_ABCC_40_EIT_7352_ProdData_GenericEthernet.txt	C40_ABCC_40_ProdData_GenericEthernet.bin
Modbus/TCP Full production: Locked to Modbus/TCP	C40_ABCC_40_EIT_7352_Locked.txt	C40_ABCC_40_EIT_7352_ProdData_Locked.bin
Modbus/TCP Re-test and firmware download	C40_ABCC_40_EIT_7352_ReTest.txt	NA
Modbus/TCP Firmware download only	C40_ABCC_40_EIT_7352_FwUpdate.txt	NA
POWERLINK Full production: Generic Ethernet	C40_ABCC_40_EPL_7273_ProdData_GenericEthernet.txt	C40_ABCC_40_ProdData_GenericEthernet.bin
POWERLINK Full production: Locked to POWERLINK	C40_ABCC_40_EPL_7273_ProdData_Locked.txt	C40_ABCC_40_EPL_7273_ProdData_Locked.bin
POWERLINK Re-test and firmware download	C40_ABCC_40_EPL_7273_ReTest.txt	NA

C40 target industrial network **Production type**	**Production script** **(Location: ./scripts/)**	**Production data file** **(Location: ./proddata/)**
POWERLINK Firmware download only	C40_ABCC_40_EPL_7273_FwUpdate.txt	NA
PROFIBUS DP-V1 Full production: Generic RS-485	C40_ABCC_40_DPV1_7332_ProdData_GenericRS485.txt	C40_ABCC_40_ProdData_GenericRS485.bin
PROFIBUS DP-V1 Full production: Locked to PROFIBUS	C40_ABCC_40_DPV1_7332_ProdData_Locked.txt	C40_ABCC_40_DPV1_7332_ProdData_Locked.bin
PROFIBUS DP-V1 Re-test and firmware download	C40_ABCC_40_DPV1_7332_ReTest.txt	NA
PROFIBUS DP-V1 Firmware download only	C40_ABCC_40_DPV1_7332_FwUpdate.txt	NA
DeviceNet Full production: Generic CAN	C40_ABCC_40_DEV_7129_ProdData_GenericCAN.txt	C40_ABCC_40_ProdData_GenericCAN.bin
DeviceNet Full production: Locked to DeviceNet	C40_ABCC_40_DEV_7129_ProdData_Locked.txt	C40_ABCC_40_DEV_7129_ProdData_Locked.bin
DeviceNet Re-test and firmware download	C40_ABCC_40_DEV_7129_ReTest.txt	NA
DeviceNet Firmware download only	C40_ABCC_40_DEV_7129_FwUpdate.txt	NA
CC-Link Full production: Generic RS485	C40_ABCC_40_CCL_7239_ProdData_GenericRS485.txt	C40_ABCC_40_ProdData_GenericRS485.bin
CC-Link Full production: Locked to CC-Link	C40_ABCC_40_CCL_7239_ProdData_Locked.txt	C40_ABCC_40_CCL_7239_ProdData_Locked.bin
CC-Link Re-test and firmware download	C40_ABCC_40_CCL_7239_ReTest.txt	NA
CC-Link Firmware download only	C40_ABCC_40_CCL_7239_FwUpdate.txt	NA

7.8 DLL API

A DLL API is available for the Windows platform. It provides all functionality that the sample application uses. This interface is intended to be used when the user wants to smoothly integrate the C40 production process into a custom production system. The source code is available, which makes it possible to port the API to the platform of choice.

HMS provides small example Visual Studio projects that import and use the DLL API to realize the production sequences described in *C40 Production Test Sequences, p. 41*.

7.8.1 DLL API Usage Examples

Two usage examples are provided that use the DLL API.

- The first example is the source code of the actual sample application, used for the Development GUI interface (*Developer GUI Interface, p. 48*) and production script engine (*Automated Production Scripts, p. 53*).

  See the Visual studio project in the **./source** directory of the C40 Licensing package.

- The second example is a straight forward Windows console application that imports the C40 DLL API and performs production sequences for C40: EtherNet/IP product.
  - See Visual studio project in the **./dllapi/C40_DllApiExample** directory of the C40 Licensing package.
  - The example sequences are based on the test sequences from *C40 Production Test Sequences, p. 41*, which are the same that the production script **C40_ABCC_40_EIP_7351_ProdData_Locked.txt** from *Example Production Scripts, p. 59* performs.
  - Additionally, different from the production script, this example will use the Generic Ethernet product firmware test case to verify that the two Ethernet ports are working properly.
  - Production data will be written to C40 if main.c define **C40_SET_PRODUCTION_DATA** is set to 1. Remember that production data can ONLY be written once and if used the **sProdData.xxx** assignments further down in main.c should be assigned properly as defined in *C40 Production Parameters, p. 32*.

The easiest way to get up to speed when using the DLL API is to examine the source code of example #2 above, where the DLL API is used straight off in main() function.

7.8.2 API Functions

A short description of each API function is provided below. All functions are defined in the file **.\dllapi\C40_DllApiExample\C40_DllApiExample\imp_api.h** in the C40 licensing package.

List of Functions

- C40_ConnectModule
- C40_DisconnectModule
- C40_PrepareTests
- C40_ResetModule
- C40_GetNumberOfTests
- C40_GetTestInfo
- C40_RunTest
- C40_GetTestResults
- C40_DownloadFirmware

C40_ConnectModule

Direction	Parameter
IN	comPort (PCSTR)
	baudRate (DWORD)
OUT	moduleId (C40_ModulePtr*)
RETURNS	BOOL

This function sets up a connection to a module via a serial line. The RAMloaderDS must start to communicate at 57.6 kb/s, but the baud rate can be increased, if the application can support it.

It is possible to connect to multiple modules at the same time, if parallel test runs are wanted (one module can only run one test at a time).

The result of this function is a moduleId that is used for all other functions.

See the files in the **./dllapi/C40_DllApiExample** directory of the C40 Licensing package for more information.

C40_DisconnectModule

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
OUT	-
RETURNS	-

This function shuts down the connection to the module.

C40_PrepareTests

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
	testHiffFile (PCSTR)
OUT	-
RETURNS	BOOL

This function loads and downloads a test file to the indicated module.

Each connected module must have a test file of its own loaded.

ⓘ *The module must be in RAMloaderDS mode before calling this function.*

C40_ResetModule

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
OUT	-
RETURNS	BOOL

This function returns the module to RAMLoaderDS mode in preparation for new tests or for firmware download.

The function should be called before C40_DisconnectModule() if C40_PrepareTests() was successfully called or the module will not be ready to accept new test files.

C40_GetNumberOfTests

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
OUT	-
RETURNS	DWORD

This function returns the number of test cases.

ⓘ *C40_PrepareTests() must have been called prior to this function.*

C40_TestInfo

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
	testIndex (DWORD)
OUT	info (C40_TestInfo*)
RETURNS	BOOL

This function provides information about a test from a specified module.

The information contains

- the unique test id
- the test version
- the parameter description for input and output data for the test

The information can be used for display purposes by the end user, and also to verify the in/out data format when matched to a test database.

C40_PrepareTests() must have been called prior to this function.

C40_RunTest

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
	testIndex (DWORD)
	params (LPCVOID)
	paramslen (DWORD)
OUT	-
RETURNS	BOOL

This function starts the indicated test. Only one test at a time can be run on a module, but it is possible to run a test on each connected module at the same time. The same test can be run multiple times, as long as the results are fetched for each test when it is finished.

The input parameters (params) are a binary pile of data. To format the test parameters properly, the input parameter description from C40_GetTestInfo() can be used. Most tests doesn't have any parameters though.

The information can be used for display purposes by the end user, and also to verify the in/out data format when matched to a test database.

C40_PrepareTests() must have been called prior to this function.

C40_GetTestResults

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
OUT	returnParams (PVOID)
	returnDataSize (DWORD)
RETURNS	C40_GetTestResultStatus

This function is used to poll for the test results. It is not possible to either start a new test or reset the module before the results have been returned (successful or not). Timeouts are handled internally, but could result in the module being in a bad state (i.e. requiring manual restart).

The output parameters (returnParams) are just like the parameters in RunTest(). To know how to interpret the data, it is necessary to use the output parameter description from C40_GetTestInfo ()

The return value is the current status of the test:

- the test is still in progress
- the test has completed and was successful

- the test has completed, but failed (test results will contain more information on what went wrong)
- an error has occurred preventing the test from completing (i.e. no test results will be available)

C40_DownloadFirmware

Direction	Parameter
IN	moduleId (C40_ModulePtr*)
	firmwareHiffFile (PCSTR)
	someProductionData (LPCVOID)
	ProductionDataLength (DWORD)
OUT	-
RETURNS	BOOL

If firmware is to be downloaded to the module, no tests should be prepared to be run in the module. If C40_PrepareTests() have been run C40_ResetModule() must be called prior to firmware download.

The update process should not be aborted in any way, since the module is being updated in parallel with the downloading.

Production data can be written during firmware download if desired (only once per produced unit). The user provides an array of bytes in the parameter someProductionData and the length of the array in ProductionDataLength. If no production data shall be written, these parameters should be set to NULL (0).

C40_GetErrorCode

Direction	Parameter
IN	-
OUT	-
RETURNS	C40_Errors

If another function in the API has failed, this function can be called to get the actual error code for the failure.