DTM-3224

OEM Quad ASI-to-IP Converter

Table of Contents

Firmware Revision History 4

1. Introduction 5
   1.1 General description 5
   1.2 ASI-to-TSoIP conversion 5
   1.3 Control 5
   1.4 DTM-3224 protocol handler 6
   1.5 Theory of operation 6
      1.5.1 Functional block diagram 7
   1.6 List of abbreviations 7
   1.7 References 8

2. Getting Started 9
   2.1 Introduction 9
   2.2 Example: Converting one ASI to TSoIP 9
      2.2.1 Test set-up 9
      2.2.2 Configuring the ASI to TSoIP conversion 9

3. Layout and Installation 10
   3.1 Physical layout 10
   3.2 Mechanical dimensions 11
   3.3 Order codes 11
   3.4 Hardware installation 11
      3.4.1 Mechanical installation 11
      3.4.2 ASI connectors 11
      3.4.3 Control connectors 12
      3.4.4 Ethernet connector 12
      3.4.5 Power connector 13
      3.4.6 ASI Stream status LEDs 13
      3.4.7 DTM-3224 power LED 14
      3.4.8 DTM-3224 Security status LED 14

4. Device Configuration and Monitoring 15
   4.1 Control interfaces 15
   4.2 Command protocol 15
      4.2.1 Command protocol on USB and RS-232 15
      4.2.2 Command protocol on I^2C 16
   4.3 Manageable items 17
   4.4 Categories 18
      4.4.1 Data types 18
      4.4.2 Device properties 18
      4.4.3 Overall configuration 19
      4.4.4 Network settings 20
      4.4.5 Firmware update 21
      4.4.6 IP transmit settings 23
      4.4.7 ASI input settings 24

4.5 Firmware upgrade 25
4.5.1 Firmware upgrade - Phases 25
4.5.2 Firmware upload – Example 25
4.5.3 Binary data to Ascii data function 26
4.6 Failsafe mode 27
4.7 Maximum input and output bitrate 27
5. Specifications 29
5.1 Network connection 29
5.2 DVB-ASI inputs 29
5.3 Transport-Stream output over IP 29
5.4 Serial control port 30
5.5 I^2C control port 30
5.6 Other specifications 31
Appendix A. Mechanical Dimensions 32
Appendix B. DTM-3224 Development Kit 33
Appendix C. Command-Line Tool - DtmCmd 35
Appendix D. Communication Example 37

Firmware Revision History

Version	Date	Changes
0.1	2016.03.18	Creation of document

1. Introduction

1.1 General description

The DTM-3224 is a compact OEM module to convert four ASI channels to Transport Streams over IP (TS-over-IP or TSoIP).

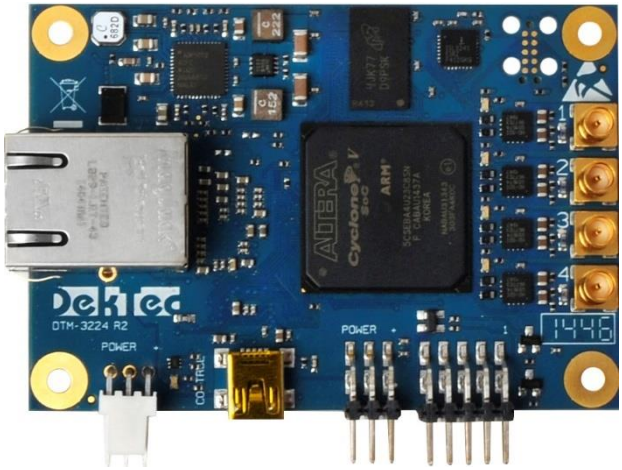


Figure 1. The PCB of the DTM-3224

The DTM-3224 offers the following features:

- DTM-3224 without accessories;
- DTM-3224-DEVKIT: DTM-3224 with power supply and cables.

1.2 ASI-to-TSoIP conversion

The DTM-3224 accepts ASI input. Key features include encapsulation of UDP or RTP according to SMPTE 2022-2, controlled scheduling of IP packets to prevent IP jitter (zero jitter playout), and adding forward error correction (FEC) according to SMPTE 2022-1.

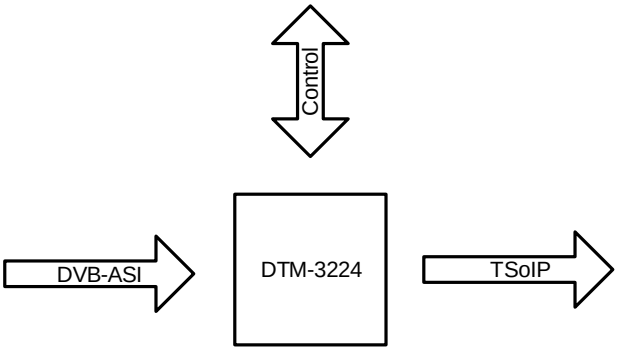


Figure 2. The DTM-3224

1.3 Control

The unit can be managed and controlled through one of the available control interfaces: USB[1], I²C or RS-232. Settings applied through one of the control interfaces can be stored in non-volatile memory

[1] The DTM-3224 USB interface emulates a UART/serial/RS-232 interface

(if setting ***Volatile storage*** is '0') or in volatile memory (if setting ***Volatile storage*** is '1') on the unit. Settings stored in non-volatile memory are automatically reloaded after a power cycle. It is not possible to configure the device via the Ethernet interface.

There are three ways to control the DTM-3224:

1. From a development PC using the serial RS-232 or USB control interface. This way of controlling can be used for pre-configuring the DTM-3224, or for experimenting with the DTM-3224's settings.
2. Using a controller on-board of the equipment that uses the DTM-3224 for I/O conversion. In this case I^2C is a plausible choice, but the other interfaces may also be used.
3. Stand-alone mode. The DTM-3224 is pre-configured and no dynamic control is applied.

Two tools are available:

1. ***Dtm3224Util*** – Windows GUI tool to view status and control settings of the DTM-3224. The tool can also be used to upload firmware. ***Dtm3224Util*** is convenient for initial configuration and experimentation with the DTM-3224.
2. ***DtmCmd*** – Command-line tool to send commands to the DTM-3224. Multiple commands can be combined in a script to apply a group of settings in one go. ***DtmCmd*** is useful for studying the low-level commands available for the DTM-3224. It is also useful to apply a pre-defined group of setting values from a script.

1.4 DTM-3224 protocol handler

An open source implementation of a protocol handler for DTM-32xx devices is available from www.dektec.com free of charge. It consists of two source files, `DtmHandler.c` and `DtmHandler.h`, which can be compiled and linked into your C or C++ application. Please refer to `DtmHandler.h` for information about how to use the protocol handler in your application.

Note:

- Command-line tool ***DtmCmd*** is an example of an application that uses the DTM handler. The source code of ***DtmCmd*** is also available on the DekTec website.

1.5 Theory of operation

Essentially, the DTM-3224 consists of two subsystems:

- A Stream Pipeline, which converts the ASI input channels to TS-over-IP packets;
- A processor subsystem that handles all internal and external control (USB, I^2C and RS232).

1.5.1 Functional block diagram

Figure 3 shows the functional block diagram of the DTM-3224.

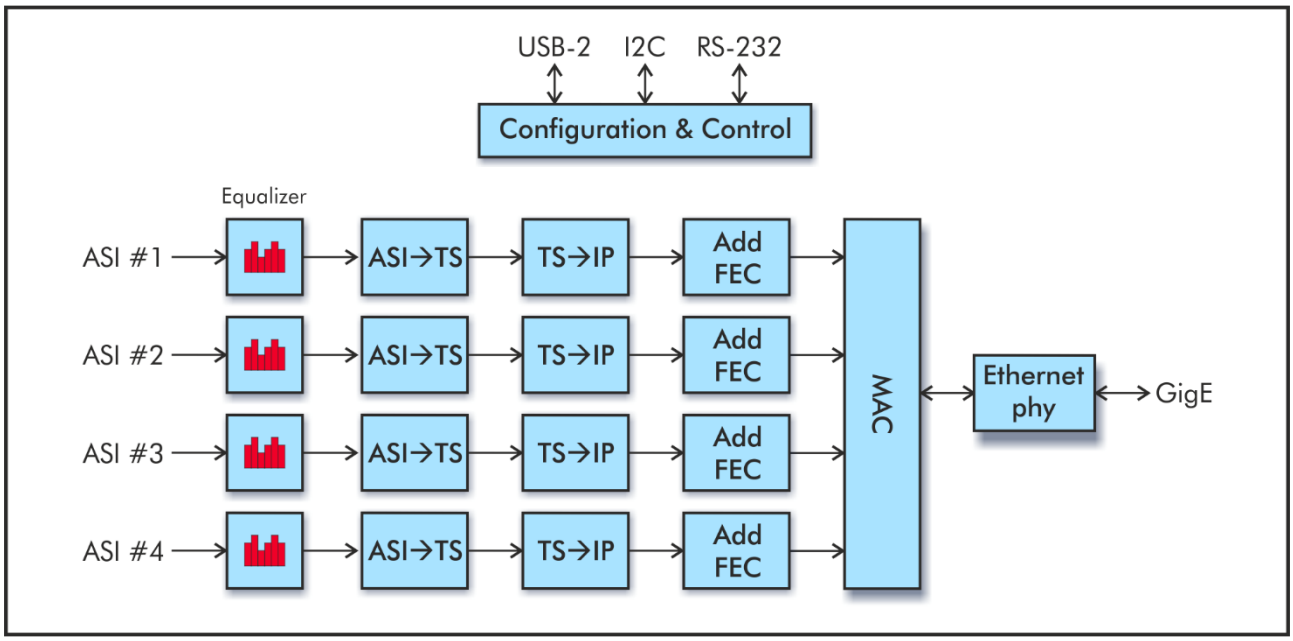


Figure 3: Functional block diagram of the DTM-3224

TS Receive: synchronizes to the stream and determines the packet size (188 or 204 bytes). IP Embedder: embeds the TS packets in IP packets. FEC Generator: will create row- and column FEC data if the DTM-3224 is configured to generate FEC packets. After multiplexing the TS packets the MAC and PHY transmits the IP packets through the Ethernet interface.

1.6 List of abbreviations

ASI	Asynchronous Serial Interface. Shorthand for DVB-ASI.
auto-MDIX	Automatic medium-dependent interface crossover. Technique to automatically detect the type of network cable: straight-through or crossover.
DHCP	Dynamic Host Configuration Protocol. Network protocol to automatically assign an IP address to a network port from a server.
DVB	Digital Video Broadcasting
FEC	Forward Error Correction
IP	Internet Protocol
MAC	Media Access Controller
Mbps	Megabit per second
NA	Not Applicable
NC	Not Connected
PCR	Program Clock Reference
R/W	Read / Write

RO	Read Only
RTP	Real-time Transport Protocol
TSoIP	Transport Stream over IP
UDP	User Datagram Protocol
WO	Write Only

1.7 References

[1] SMPTE-2022-1, Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks

[2] SMPTE-2022-2, Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks

2. Getting Started

2.1 Introduction

This section provides a walkthrough for getting started with the DTM-3224. One example is described: converting one ASI to TSoIP.

The description below assumes that the DTM-3224 development kit (see Appendix B) is used to control the DTM-3224 over USB from a development PC. The GUI tool ***Dtm3224Util*** is used to apply settings and observe status.

2.2 Example: Converting one ASI to TSoIP

This set-up will receive a stream on the ASI interface and transmit the stream over IP.

2.2.1 Test set-up

For testing this configuration, an external ASI source and a TSoIP receiver are required.

This tutorial assumes that the network dynamically assigns IP addresses through DHCP, and that an ASI transmitter is connected to port 1.

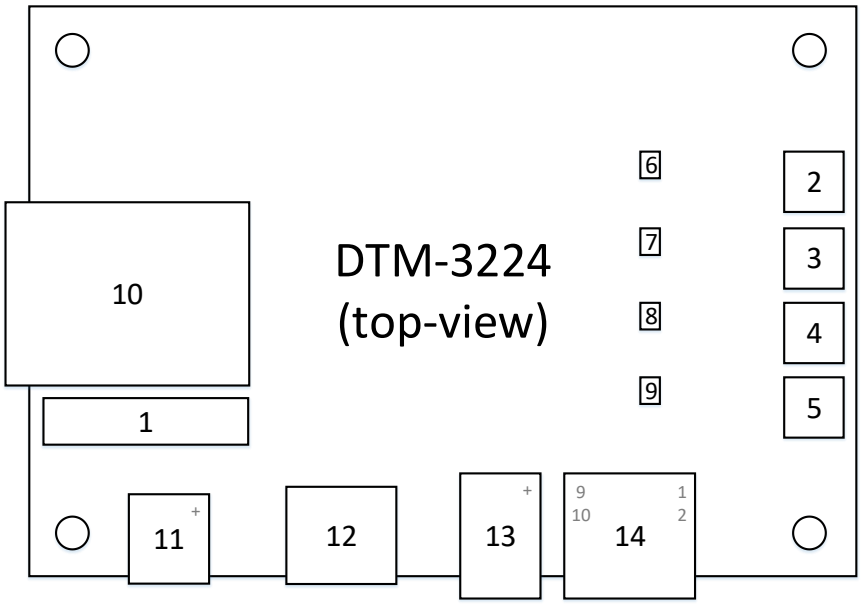
2.2.2 Configuring the ASI to TSoIP conversion

The settings below (sequential order) set up the DTM-3224 for one channel ASI to IP conversion. No FEC will be generated in this example.

	Category	Setting	Index	Value
1	0x83: ASI input	8: Enable	1	0 = Off
2	0x82: IP transmit	3: FEC Enable	1	0 = Off
3	0x82: IP transmit	6: IPv4 Address	1	224.1.1.1
4	0x82: IP transmit	8: UDP Port	1	5678
5	0x82: IP transmit	9: #TP per IP	1	3 = 3 Transport Packets per IP pckt
6	0x82: IP transmit	10: Protocol	1	0 = UDP
7	0x82: IP transmit	13: Time to Live	1	100
8	0x83: ASI input	1: Packet Size	1	0 = 188-byte MPEG-2 packets
9	0x83: ASI input	9: Polarity	1	0 = Automatic
10	0x83: ASI input	8: Enable	1	1 = On

3. Layout and Installation

3.1 Physical layout



Nr.	Function	Connector	Description
1	Identifier		Type and revision number
2	ASI	MCX 75Ω	ASI input channel 1
3	ASI	MCX 75Ω	ASI input channel 2
4	ASI	MCX 75Ω	ASI input channel 3
5	ASI	MCX 75Ω	ASI input channel 4
6	LED		ASI Steam status LED channel 1
7	LED		ASI Steam status LED channel 2
8	LED		ASI Steam status LED channel 3
9	LED		ASI Steam status LED channel 4
10	Ethernet	RJ-45	Ethernet port for TS-over-IP transmission
11	Power	3-pin header 2.54mm pitch	Power option 1
12	Control	Mini-USB	Serial-over-USB interface for board control
13	Power	6-pin header 2.54mm pitch	Power option 2
14	Control	10-pin header 2.54mm pitch	RS-232 and I^2C interface for board control

3.2 Mechanical dimensions

See Appendix A.

3.3 Order codes

Order Code	Picture	Description
DTM-3224		DTM-3224 without accessories
DTM-3224-DEVKIT	The DTM-3224 development kit contains the following items: • DTM-3224 placed on four plastic studs • DTM-3301 USB-to-RS232 interface board • 12V/0.7A power supply • USB cable type A to mini B • 4x MCX to BNC cable assemblies with a length of 130 mm	

3.4 Hardware installation

3.4.1 Mechanical installation

The unit can be mounted onto a support plate by means of four 3.0 mm bolts and appropriate spacers. Ensure that there is sufficient airflow to provide cooling of the board.

3.4.2 ASI connectors

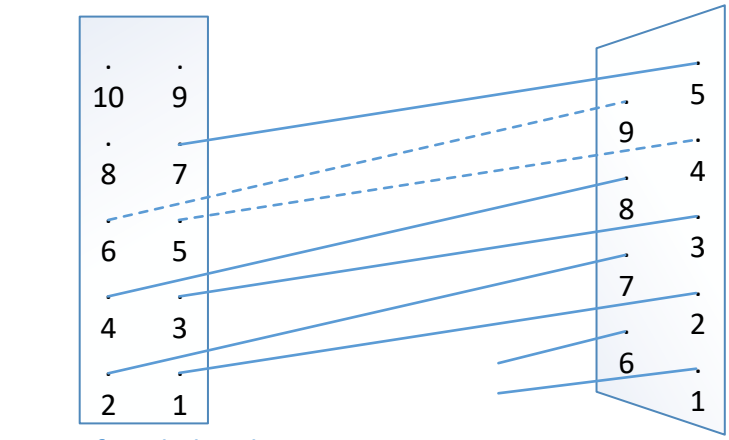
The ASI connectors (2-5) are MCX connectors with an impedance of 75 ohm.

3.4.3 Control connectors

The DTM-3224 has two control connectors. One connection is standard mini-USB type. The pinning of the other connector is shown in the table below. It's a double pin row 2.54 mm standard type male header for connecting the RS-232 or I²C control bus.

Pin	RS-232/I²C function
1	TX
2	CTS
3	RX
4	RTS
5	NC
6	NC
7	GND
8	NC
9	SDA
10	SCL

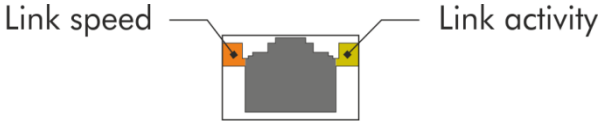
The pinning of this connector has been chosen in such a way that a 9-way flatcable with a press-fit sub-D flatcable connector can be connected directly to pin 1 – 7, see the figure below. The leads coming from pin 1 and 6 are not connected. (The dotted lines indicate connections with do not actually have a function for the DTM-3224, i.e. go to a NC pin.)



An I²C controller can be connected to SDA and SCL on pin 9 and 10, with signal ground on pin 7.

3.4.4 Ethernet connector

The Ethernet connector is a standard shielded RJ45 jack with two status LEDs.



Link speed LED	
Orange	1000Mbps
Green	10 or 100Mbps (half duplex)
Off	No signal

The link-activity LED flashes whenever an Ethernet packet is received or transmitted.

A standard Cat5E (or higher) patch cable can be used to connect the DTM-3224 to a network. Either a straight-through or cross-over network cable can be used; the type of cable will be automatically recognized (auto-MDIX operation). The DTM-3224 will automatically select the link speed of the connected network (10/100/1000Mbps).

3.4.5 Power connector

The DTM-3224 must be powered from an external source with a voltage of 12V±10% DC. Power consumption is max. 5W. Two options are available for the power connection. Both connections are internally connected. Do not connect both connections to different power sources as that might cause short-circuit.

The pinning of the power connector 1 (11) is shown in the table below. The connector type is the Molex KK series 2.54 mm pitch.

Pin	Description
1	Positive power connection
2	Ground
3	Reset connection

The pinning of the power connector 2 (13) is shown in the table below. The connector type is the standard 2.54 mm pitch header.

Pin	Description
1 + 2	Positive power connection
3 + 4	Ground
5 + 6	Reset connection

The board can be reset by connecting either of the Reset connection pins to ground for at least 100ms.

3.4.6 ASI Stream status LEDs

These LEDs indicate the status of the ASI streams. The following colors are used for status indication:

LED status	Description
Off	Channel is disabled
Short green flashes	No carrier detected on ASI or parallel input
Long green flashes	Carrier detected but no data on ASI or parallel input
Green	Valid signal detected on ASI or parallel input
Red	Erroneous signal detected on ASI or parallel input

3.4.7 DTM-3224 power LED

This LED indicates the current power status of the DTM-3224. The following colors are used for status indication:

LED status	Description
Off	Board is not powered
Green	Board is powered

3.4.8 DTM-3224 Security status LED

These ASI LEDs indicates the current security status of the DTM-3224. The following colors are used for status indication:

LED status	Description
Normal behavior	Security Passed, normal behavior is described in 3.4.6
Short Yellow flashes	Security failed

4. Device Configuration and Monitoring

4.1 Control interfaces

The DTM-3224 can be configured and monitored using USB (UART-over-USB), RS-232 (DCE) and I^2C. It is not required to select between interfaces: The DTM-3224 will automatically use the interface on which it detects activity.

The control interfaces use a command and response protocol that is described in the next sections.

4.2 Command protocol

Commands and responses are wrapped into a frame structure that contains address, category, setting, read/write, index (optional) and data (optional). The DTM-3224 accepts uppercase and lowercase characters, but will always respond in uppercase.

4.2.1 Command protocol on USB and RS-232

The communication on the serial/UART-based interfaces uses ASCII representation of the hexadecimal representation to communicate the settings. E.g. 244 ➔ 0xF4 ➔ ASCII characters "F" and "4".

Field	Format	Description
Start	ASCII character STX (0x02)	ASCII "start of text" character
Address	2 hex digits	8-bit address. No effect for serial communication, except for CRC calculations.
Category	2 hex digits	Selects a "category" of settings
Setting	2 hex digits	Selects a setting within the selected category
Read/Write	ASCII character 'R' or 'W'	'R' for read and 'W' for write
Index	4 hex digits	(Only for selected categories) An index parameter to indicate the zero-based channel number
Data	n hex digits / n ASCII characters[2]	The data written or read. The data length is variable for each setting. In case of a write operation, the data is a (negative) acknowledgement
Checksum	2 hex digits	This is the least significant byte of the two's complement[3] sum of all characters in the message including Address, but excluding the STX and ETX characters and the checksum itself
End	ASCII character ETX (0x03)	

[2] The firmware update uses ASCII characters 128 to 255 for sending the firmware data, as this requires less communication overhead: 7 bits per symbol, instead of 4 bits.
[3] Two's complement: Invert all bits and add one.

Figure 4 below shows the structure of a command written through the serial interface. If the command is a read-command, the data may be omitted. If the category does not require an index, the index must be omitted.

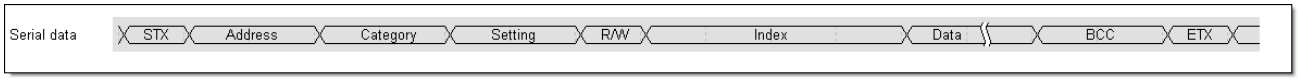


Figure 4. Command on one of the serial control interface

All commands successfully sent to the DTM-3224 are answered with a copy of the command including the data bytes.

When an incorrect checksum or a too-short message length is detected, the DTM-3224 will ignore the message and not return an answer. When protocol errors are detected, e.g. a combination of a valid category with an invalid setting, the R/W byte of the reply is replaced with the ASCII character 'E' and the data is removed from the message.

4.2.2 Command protocol on I²C

Field	Format	Description
Start	S	Standard I²C start condition
Address	I²C address byte	7-bit I²C address followed by the I²C R/W bit, which is set to 0 and 1 in the command- and response sequence respectively
Category	1 byte	Selects a "category" of settings
Setting	1 byte	Selects a setting within the selected category
Read/Write	1 byte	0x01 for read and 0x00 for write
Index	2 bytes	(Only for selected categories) An index parameter to indicate the zero-based channel number
Data	n bytes	The data written or read. The data length is variable for each setting. In case of a write operation, the actual data is returned as a (negative) acknowledgement
Checksum	1 byte	This is the least significant byte of the two's complement of the sum of the 7-bit I2C slave address and all data-bytes in the I2C message (excluding the checksum). The I2C R/W bit is not included, an incorrect value of this bit would cause the checksum to be not received at all.
End	P	Standard I²C stop condition. A repeated start condition can be used at all times to concatenate multiple I²C read / write actions

Figure 5 below shows the sequence to send a command over I²C to the DTM-3224. If the command is a read-command, the data may be omitted. If the category does not require an index, the index must be omitted.

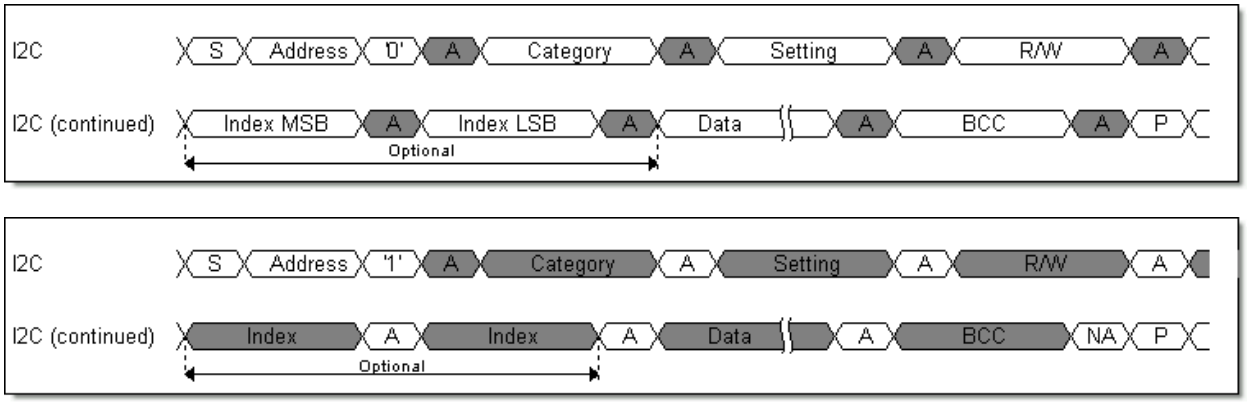


Figure 5. Command (upper sequence) and response (lower) sequence for I²C

When an incorrect checksum is detected, the DTM-3224 will not return an answer. When protocol errors are detected, e.g. a combination of a valid category with an invalid setting, the R/W byte of the reply is replaced by the ASCII character 'E' and the data is removed from the message.

When a master starts writing to a device while the previous command is still being executed, the device will ignore the data.

When a master starts reading from a device and there is no answer available (yet), the device will reply with the value 0x00. The value 0x00 will be returned until an answer is available and the master has initiated a new read transaction.

S and P are the standard I²C start and stop conditions.

4.3 Manageable items

The tables in this section provide lists of variables that can be configured and/or monitored using the USB, RS-232 or I²C interface. The 'Access' column indicates whether the variable is Read Only (RO), Write Only (WO), Read/Write (R/W), Read/Clear[4] (R/C) or not applicable (NA).

When, in the 'Access' column, the text "(V)" is written after the access type, the variable can be stored either in a volatile or non-volatile manner, depending on the 'Volatile Storage' setting (category 0x02, setting 0x03). The volatility of the variables without "(V)" cannot be controlled.

[4] Write any value to clear/reset the counter value

4.4 Categories

Manageable Items – Categories			
Nr	Settings Category	Description	Index required
0x01	**Device**	Device Properties	No
0x02	**Configuration**	Overall configuration	No
0x03	**Network**	Network settings	No
0x82	**IP transmit**	IP transmit settings	Yes
0x83	**ASI Input**	ASI input settings	Yes
0x86	**Firmware update**	Firmware update (type 3)	Yes

4.4.1 Data types

Manageable Items – Data types			
Type	Description	USB / RS-232	I²C
Uint8	8-bit unsigned integer	2 hex digits	1 byte
Uint16	16-bit unsigned integer	4 hex digits	2 bytes
Uint32	32-bit unsigned integer	8 hex digits	4 bytes
Uint64	64-bit unsigned integer	16 hex digits	8 bytes
IPv4 addr	IPv4 address	8 hex digits	4 bytes
IPv6 addr	IPv6 address	32 hex digits	16 bytes
String	String of ASCII characters	Variable	String

All data types are sent with the most-significant byte first.

4.4.2 Device properties

Manageable Items - Category 0x01 – Device properties				
Nr	Variable	Description	Access	Type
0x01	**FPGA version**	Version number of the FPGA code on-board of the DTM-3224	RO	Uint8
0x03	**Firmware version**	Firmware version: the major version is encoded in the tens, the minor version in the units, e.g. '10' indicates v1.0	RO	Uint32
0x04	**Serial number**	Unique serial number for this device, e.g. 3224.000.010	RO	Uint32
0x05	**Type**	Device type number, e.g. 3224	RO	Uint32
0x06	**Hardware revision**	Hardware revision number, e.g. 302 = 3.2	RO	Uint32
0x08	**Production date**	Production date of this board Bit 31..8: Year Bit 7..0: Month	RO	Uint32

Manageable Items - Category 0x01 – Device properties				
Nr	Variable	Description	Access	Type
0x09	**I²C address**	7-bit I²C address Default: 0x60 Range: 0x10 – 0x77	R/W	Uint8
0x0A	**RS-232 serial control port baud rate**	Serial baud rate Valid values 9600, 115200 and 256000. Changes to the baud rate are automatically persisted in flash memory. Default: 115200	R/W	Uint32
0x0B	**Subtype**	Device subtype, e.g. 0=none, 1=A, …	RO	Uint8

4.4.3 Overall configuration

The configuration settings are used to switch the mode and application of the DTM-3224. When setting 0x02 'Application' is changed the DTM-3224 is rebooted automatically.

Manageable Items - Category 0x02 – Overall configuration				
Nr	Variable	Description	Access	Type
0x02	**Application**	0 = Failsafe application 1 = Normal operation	R/W	Uint8
0x03	**Volatile settings**	0 = Settings are persisted in flash memory 1 = Settings are volatile (not persisted in flash memory)	R/W	Uint8
0x04	**Persist all settings**	Store all current settings in flash memory. Data: Don't care	WO	Uint8
0x05	**Number of writes to flash memory**	Number of times the settings have been written to flash memory	RO	Uint32

When the 'Volatile storage' variable is set to 1, any changes made to settings with access type R/W (V), will be lost after reboot.

4.4.4 Network settings

The network settings are used to configure the IP address, subnet mask and gateway for the network connection. The MAC address can be read from a read-only variable.

Manageable Items - Category 0x03 – Network settings				
Nr	Variable	Description	Access	Type
0x01	**IPv4 address**	IPv4 Address	R/W	IPv4 addr
0x02	**IPv4 Subnet mask**	IPv4 Subnet mask	R/W	IPv4 addr
0x03	**IPv4 Gateway**	IPv4 Gateway	R/W	IPv4 addr
0x04	**DHCP enabled**	0 = Use static IP address 1 = Use DHCP	R/W	Uint8
0x05	**Reboot**	1 = Reboot	WO	Uint8
0x06	**MAC-address**	MAC address	RO	Int48
0x07	**IP Version**	0 = IPv4 only 1 = IPv6 only 2 = IPv4 + IPv6	R/W	Uint8
0x08	**IPv6 address**	IPv6 address	R/W	IPv6 addr
0x09	**IPv6 prefix**	Length (in bits) of the routing prefix/CIDR netmask of the IPv6 address	R/W	Uint8
0x0A	**IPv6 gateway**	IPv6 gateway	R/W	IPv6 addr
0x0B	**Port enable**	0 = Network is disabled 1 = Network is enabled, settings changed during Hold are now applied 2 = Hold settings execution on both network port and connected ASI channels.[5]	R/W	Uint8
0x0C	**Channel settings status**	0 = All settings are valid 1 = The total bitrate configured using items 0x83 0x0A exceeds device specs.	R	Uint8

When the 'DHCP enabled' behavior variable is read, it indicates whether a static IP address or DHCP is used. The 'DHCP enable' setting can be used to turn on and turn off the DHCP client:

- When 1 is written to 'DHCP enabled', the IP address is set to 0.0.0.0 and the DHCP client will start looking for a DHCP server to acquire an IP address.
- When 0 is written to 'DHCP enabled', the DTM-3224 retrieves the most recently manually entered IP-address (fixed IP) and uses this address as its fixed IP-address.

- When a fixed IP-address has never been entered before, writing 0 to 'DHCP enabled' will cause the DTM-3224 to use a hardcoded factory default address (192.168.144.120 / 255.255.255.0) as its fixed IP address. Naturally this default IP address, selected by writing 0 to 'DHCP enabled', can be overridden by entering an IP address manually.

[5] The 'Hold settings' option in the ASI input and network categories indicate that the current channel or all channels should hold their current settings and state. Written settings are flushed to the actual device only when the Hold setting is removed from both the ASI channel and the network settings.
Note that 'hold settings' on a channel also affects channel's IP Transmit settings: they will also be held.

4.4.5 Firmware update

To upgrade the DTM-3224, the new firmware has to be uploaded in "file parts". The firmware-update settings are used to upload the firmware file and control the programming process.

Manageable items – Category 0x86 – Firmware upgrade type 3				
Nr	Variable	Description	Access	Type
0x01	**Application index**	Index of application to upgrade, set before issuing "Start upgrade" command. Note that the Failsafe (0) application cannot be upgraded through this interface. Must be "1" for the DTM-3224.	R/W	Uint16
0x02	**Command**	Write to run command 0 = Start upgrade 1 = Flush image 2 = Start verification 3 = Activate upgrade	WO	Uint8
0x03	**Status**	0 = Idle 1 = Write in progress 2 = Write done 3 = Verification in progress 4 = Verification done 5 = Activation in progress 6 = Activation done 255 = Error (see Error reason)	RO	Uint8
0x04	**Progress**	Status: 0-254: 0 – 100, percentage completed of current phase. 255: 0 (error)	RO	Uint8
0x05	**Error reason**	0 = No error 1 = Failed to open flash device for writing 2 = Failed to detect information about NAND layout 3 = Upgrade image is too big 4 = Not enough free space. Too many bad blocks? 5 = Failed to get bad block information 6 = Failed to update bad block information 7 = Failed to seek back to written data 8 = Failed to read back written data 9 = Image has incorrect header 10 = Image header CRC incorrect 11 = Incorrect offset in header 12 = File CRC incorrect	RO	Uint16
0x06	**Error information**	Not defined for the DTM-3224	RO	Uint32

Manageable items – Category 0x86 – Firmware upgrade type 3				
Nr	Variable	Description	Access	Type
0x07	**Transmit**	Transmission of firmware image, continuity counter in index starting with 1. Note: Never send more than 'Buffer available'.	WO	1..1024 bytes
0x08	**Buffer available**	Number of bytes the device is ready to accept at this time.	RO	Uint32

This category requires an index to be sent with each command. For the "File part" setting, index should be the file part number. For the other settings, the index value is ignored.

Each part may contain at most 1024 data-bytes.

For USB/RS-232 a different encoding is used for update data, as explained below.

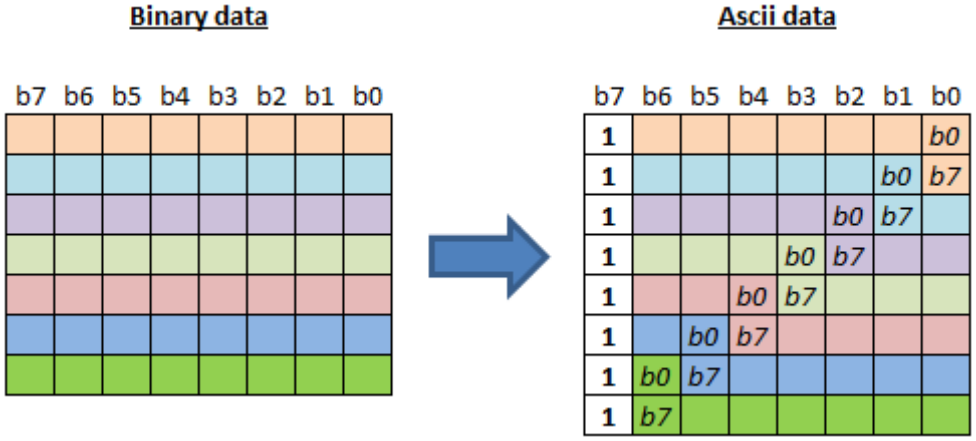


Figure 6. ASCII encoding for firmware update data

For each 7 bits of data one 8-bit ASCII character is send, where the MSB of the ASCII character is set to 1 (known as the extended ASCII range). The translation is illustrated in Figure 6.

This encoding is only applied to the data-part of the "File part" message. (category 0x86, setting 0x07). The rest of the messages are encoded as before.

An example of a firmware update sequence and an example routine for the above translation can be found in paragraph 4.5.

4.4.6 IP transmit settings

The IP transmit settings category requires an index to be sent with each command.

Manageable Items - Category 0x82 – IP transmit				
Nr	Variable	Description	Access	Type
0x03	**FEC enable**	0 = Disable generation of FEC packets 1 = Enable generation of FEC packets	R/W (V)	Uint8
0x04	**FEC #columns**[6]	1..20; #FEC columns generated	R/W (V)	Int16
0x05	**FEC #rows**[6]	4..20; #FEC rows generated	R/W (V)	Int16
0x06	**IPv4 address**	Destination IPv4 address for the TSoIP packets Note: If an IPv6 address is configured, this field will read as "0.0.0.0".	R/W (V)	IPv4 addr
0x07	**IP address status**	0 = Destination IP address has been resolved to a MAC address 1 = Failed to resolve destination IP address	RO	Uint8
0x08	**UDP port**	0..65536; Destination UDP port number	R/W (V)	Uint16
0x09	**#TP per IP**	1..7; Number of transport packets to be generated per TSoIP packet	R/W (V)	Uint8
0x0A	**Protocol**	Protocol used for the TSoIP output: 0 = UDP 1 = RTP	R/W (V)	Uint8
0x0B	**Index**	Channel index	RO	Int32
0x0C	**Input type**	Identifies the type of input interface: Always 1 (= ASI input) for DTM-3224	RO	Uint8
0x0D	**Time to live**	0..255; The generated IP packets will use this TTL (= Time To Live)	R/W (V)	Uint8
0x0E	**IPv6 address**	Destination IPv6 address for the TSoIP packets	R/W (V)	IPv6 addr
0x0F	**FEC dimension**	0 = 1D 1 = 2D (default)	R/W (V)	Uint8

[6] FEC #columns × FEC #rows ≤ 400

4.4.7 ASI input settings

The ASI input settings category requires an index to be sent with each command.

Manageable Items - Category 0x83 – ASI input settings				
Nr	Variable	Description	Access	Type
0x01	**Packet size**	0 = Generate IP packets with 188-byte transport packets 1 = Generate IP packets with 204-byte transport packets	R/W (V)	Uint8
0x02	**Physical port**	Physical port number of ASI input	RO	Uint8
0x03	**Status**	0 = Valid ASI signal detected 1 = No carrier/signal detected 2 = Failed to lock to ASI input	RO	Uint8
0x04	**TS rate**	Transport stream rate of the received ASI stream in bps for 188 byte packets	RO	Int32
0x05	**Valid count**	Number of received bytes. Write any value to reset the counter.	R/C	UInt64
0x06	**Violation count**	Number of ASI code violation detected. Write any value to reset the counter.	R/C	UInt64
0x08	**Enable**	0 = Disable 1 = Enable, settings changed during Hold will now be applied 2 = Hold settings for this port[7]	R/W (V)	Uint8
0x09	**Polarity**	0 = Automatic 1 = Normal 2 = Inverted	R/W (V)	Uint8
0x0A	**Maximum bitrate**	Maximum bitrate in mbps Packets exceeding this bitrate will be dropped.	R/W (V)	UInt8
0x0B	**Current state**	Current state of the ASI channel. A channel may be Inactive even though it is set Enabled if the total bitrate settings exceed the DTM-3224's specifications. 0 = Inactive 1 = Active	RO	Uint8
0x0C	**Detected polarity**	0 = Unknown 1 = Normal 2 = Inverted	RO	Uint8
0x0D	**Detected packet size**	0 = Unknown 1 = 188-byte per packet detected 2 = 204-byte per packet detected	RO	Uint8
0x0E	**Bitrate overflow count**	Number of packets dropped because they exceeded the configured maximum bitrate. Write any value to reset the counter.	R/C	UInt64

4.5 Firmware upgrade

4.5.1 Firmware upgrade - Phases

Updating the firmware of the device consists of four phases:

1. Uploading the file. The file is uploaded in "parts", one part at a time. This involves the following steps:
   a. Abort previous file upload to get the DTM-3224 into an idle state.
   b. Send the 'Start upgrade' command.
   e. Upload file parts: Check the index number of the last uploaded part and send the next part.
2. The actual flashing of the device firmware, as follows:
   a. Send the 'Flush image' command. The uploaded file is read from RAM and programmed in the flash memory of the DTM-3224.
   b. (Optional) Read programming progress as a percentage.
   c. Check for update errors (category 0x86, setting 0x03 status: 255=error)

   Warning: Do not power off the device while flash programming is in progress.
3. When the firmware flashing is complete, the programmed firmware may be verified.
4. When verification of the firmware image is complete, the DTM-3224 should be rebooted to make the upgrade effective. The DTM-3224 is rebooted automatically after completing a firmware update.

After the device has been updated and rebooted, the controller may check the status of the device/firmware using a combination of the following settings:

- Check the application. Category "***configuration settings***", setting "***application***" indicates whether the device is in normal operation mode or in ***failsafe*** mode. If the device was in normal operation mode and reboots in failsafe mode, the current application firmware is corrupt. See section 4.5.3 for more information about failsafe mode.
- Check the package version. Category "***device settings***", setting "***package version***" shows the version number of the current package.

4.5.2 Firmware upload – Example

If a file consisting of 1000 bytes must be sent in packets of 150 data-bytes, there are going to be 7 parts. The first six parts are 150 bytes long and the last part consists of 100 bytes. The table below

[7] The 'Hold settings' option in the ASI input and network categories indicate that the current channel or all channels should hold their current settings and state. Written settings are flushed to the actual device only when the Hold setting is removed from both the ASI channel and the network settings.
Note that 'hold settings' on a channel also affects channel's IP Transmit settings: they will also be held.

shows the content of the communication messages. Each line represents a message and the lines are shown in chronological order.

	Category (hex)	Setting (hex)	Read/write (I^2C / RS-XXX)	Index (hex)	Data	Description
Application idx	86	01	00 / 'W'	0000	1 (0x01)	
Reply	86	01	00 / 'W'	0000	1 (0x01)	
Command	86	02	00 / 'W'	0000	0 (0x00)	
Reply	86	02	00 / 'W'	0000	0 (0x00)	
File part	86	07	00 / 'W'	0001	File bytes: 0 - 149	
Reply	86	07	00 / 'W'	0001	-	
File part	86	07	00 / 'W'	0002	File bytes: 150 - 299	
Reply	86	07	00 / 'W'	0002	-	
File part	86	07	00 / 'W'	0003	File bytes: 300 - 449	
Reply	86	07	00 / 'W'	0003	-	
File part	86	07	00 / 'W'	0004	File bytes: 450 - 599	
Reply	86	07	00 / 'W'	0004	-	
File part	86	07	00 / 'W'	0005	File bytes: 600 - 749	
Reply	86	07	00 / 'W'	0005	-	
File part	86	07	00 / 'W'	0006	File bytes: 750 - 899	
Reply	86	07	00 / 'W'	0006	-	
File part	86	07	00 / 'W'	0007	File bytes: 900 - 999	
Reply	86	07	00 / 'W'	0007	-	
Flush image	86	02	00 / 'W'	0000	1 (0x01	
Reply	86	02	00 / 'W'	0000	1 (0x01	
Status	86	03	01 / 'R'	0000	–	Write in
Reply	86	03	01 / 'R'	0000	1 (0x01)	progress
Status	86	03	01 / 'R'	0000	–	Write done
Reply	86	03	01 / 'R'	0000	2 (0x02)	
Start verification	86	02	00 / 'W'	0000	2 (0x02)	
Reply	86	02	00 / 'W'	0000	2 (0x02)	
Status	86	03	01 / 'R'	0000	–	Verification
Reply	86	03	01 / 'R'	0000	4 (0x04)	done
Activate upgr.	86	02	00 / 'W'	0000	3 (0x03)	
Reply	86	02	00 / 'W'	0000	3 (0x03)	
Status	86	03	01 / 'R'	0000	–	Activation
Reply	86	03	01 / 'R'	0000	4 (0x04)	done
Reboot	03	05	00 / 'W'	0000	1 (0x01)	
Reply	03	05	00 / 'W'	0000	1 (0x01)	

4.5.3 Binary data to Ascii data function

To convert the program data (File bytes) to ASCII the following (C++14) function could be used.

```
void BinToAsc128(
    const std::vector<unsigned char>& BinData,
    std::vector<unsigned char>& Asc128Data) {
    // keep track of position, because encoding 8 bits in 7
    unsigned int BitPos(0);
    unsigned int Remainder(0);
    for (auto InputElement : BinData)
    {
```

```
        // add input element to remainder
        Remainder |= InputElement << BitPos++;
        // output is in the range 128..255
        Asc128Data.push_back((Remainder & 0x7F) | 0x80);
        // remove written data
        Remainder >>= 7;
        // if remainder has 7 bits, push back those
        if (BitPos == 7)
        {
            // output is in the range 128..255
            Asc128Data.push_back((Remainder & 0x7F) | 0x80);
            // remove written data
            Remainder >>= 7;
            // reset bit position
            BitPos = 0;
        }
    }
    // check for final element:
    // if there is one: push_back
    if (BitPos > 0)
        // output is in the range 128..255
        Asc128Data.push_back((Remainder & 0x7F) | 0x80);
}
```

4.6 Failsafe mode

The DTM-3224 supports a special "failsafe" mode to enable recovery from an erroneous configuration. In failsafe mode the unit has no ASItoIP functionality: the user can only configure the IP address and load new firmware.

Failsafe mode is entered in the following cases:

- The active image has been corrupted and booting to it failed.
- The user selected the failsafe mode through configuration option 'application'

The normal operation mode can be selected again using configuration option 'application'.

4.7 Maximum input and output bitrate

Due to technical limitations the maximum bitrate for the IP output of the DTM-3224 is 750mbps. This includes overhead from IP packets, overhead from FEC and idle time on the Ethernet line. The DTM--3224 verifies the configuration to prevent unexpected packet drops due to configuration errors and reports via item 0x0C in category 0x03 if the configured max bitrate is valid.
The DTM-3224 computes the required output bandwidth for each ASI port separately. The following parameters are relevant:

- Maximum bitrate (0x83, 0x0A)
- IP version
- Protocol (UDP / RTP)
- #TP per IP packet
- TP size
- FEC parameters (1D/2D, #columns, #rows)

Computation for each port is as follows:

$$IP\ packets/s = \left\lceil \frac{max\ bitrate\ in\ mbps * 125000}{\#TP\ per\ IP * TP\ size} \right\rceil$$

$$IP\ packet\ size = \#TP\ per\ IP * TP\ size + 50$$

If using IPv6 (as opposed to IPv4), add 20 to packet size.

If using RTP (as opposed to UDP), add 12 to packet size.

$$Required\ channel\ bandwidth = IP\ packet\ size * IP\ packets/s$$

Additional bandwidth is required for FEC (only if enabled):

$$FEC\ computations/s = \left\lceil \frac{IP\ packets/s}{\#columns * \#rows} \right\rceil$$

If 1D FEC is enabled:

$$FEC\ packets\ = (FEC\ computations/s)\ *\ \#columns$$

If 2D FEC is enabled:

$$FEC\ packets\ = (FEC\ computations/s)\ * (\#columns + \#rows)$$

$$Required\ channel\ bandwidth += FEC\ packets * (IP\ packet\ size + 16)$$

Examples:

150Mbps input, IPv4, UDP, 7 TP per IP, 188 bytes, no FEC

⇨ Requires **158Mbps** channel bandwidth

5Mbps input, IPv4, RTP, 7 TP per IP, 188 bytes, 2d FEC; 5 rows, 5 columns

⇨ Requires **7.5Mbps** channel bandwidth

*The total channel bitrate should be lower than **750Mbps***

5. Specifications

5.1 Network connection

	Min	Typ	Max	Unit / Remarks
Network Port				
Standard		IEEE 802.3a		
Data rate		100/1000 auto detect		Mbps
Connector		RJ-45 with LEDs		
Control				
Ethernet encapsulation		Ethernet II		
IP support		IPv4, IPv6		
IP-address assignment		DHCP, link local or static		
Multicast support		IGMP v3, MLD v2		
Network management		Not supported		

5.2 DVB-ASI inputs

	Min	Typ	Max	Unit / Remarks
Standard				
DVB-ASI		EN50083-9		
Ports				
Number of channels		4		
Connectors		75-Ω MCX		Input
Return loss	15	17		dB
Error-free cable length	100			m
Transport-stream bitrate	0.1		214	Mbps
Packet size		188/204		bytes

5.3 Transport-Stream output over IP

	Min	Typ	Max	Unit / Remarks
General				
TSoIP encapsulation		UDP or SMPTE 2022-2		
Input-to-IP delay			1	ms
UDP				
TS-packet size		188 or 204		bytes
TS packets / IP packet	1		7	No long UDP packets
SMPTE 2022-2				

	Min	Typ	Max	Unit / Remarks
Transport protocol		RTP		
FEC		SMPTE 2022-1		
Packet size		188 or 204		bytes
TS packets / IP packet	1		7	
FEC Size: L	1		20	
FEC Size: D	4		20	
FEC Size: LxD	4		400	

5.4 Serial control port

	Min	Typ	Max	Unit / Remarks
Interface port				
Connector		10-way pin header, 2.54 mm pitch		Pin 1..7 compatible with standard RS-232 (DCE) sub D connector
Signals		TX/RX/RTS/CTS		
Serial Format				
Interface standard		RS-232		
Format		8 bit, one stop bit, no parity		
Handshaking		hardware flow control		
Speed		9600, 115200 or 256000 115200 by default		Baud, configurable through command protocol

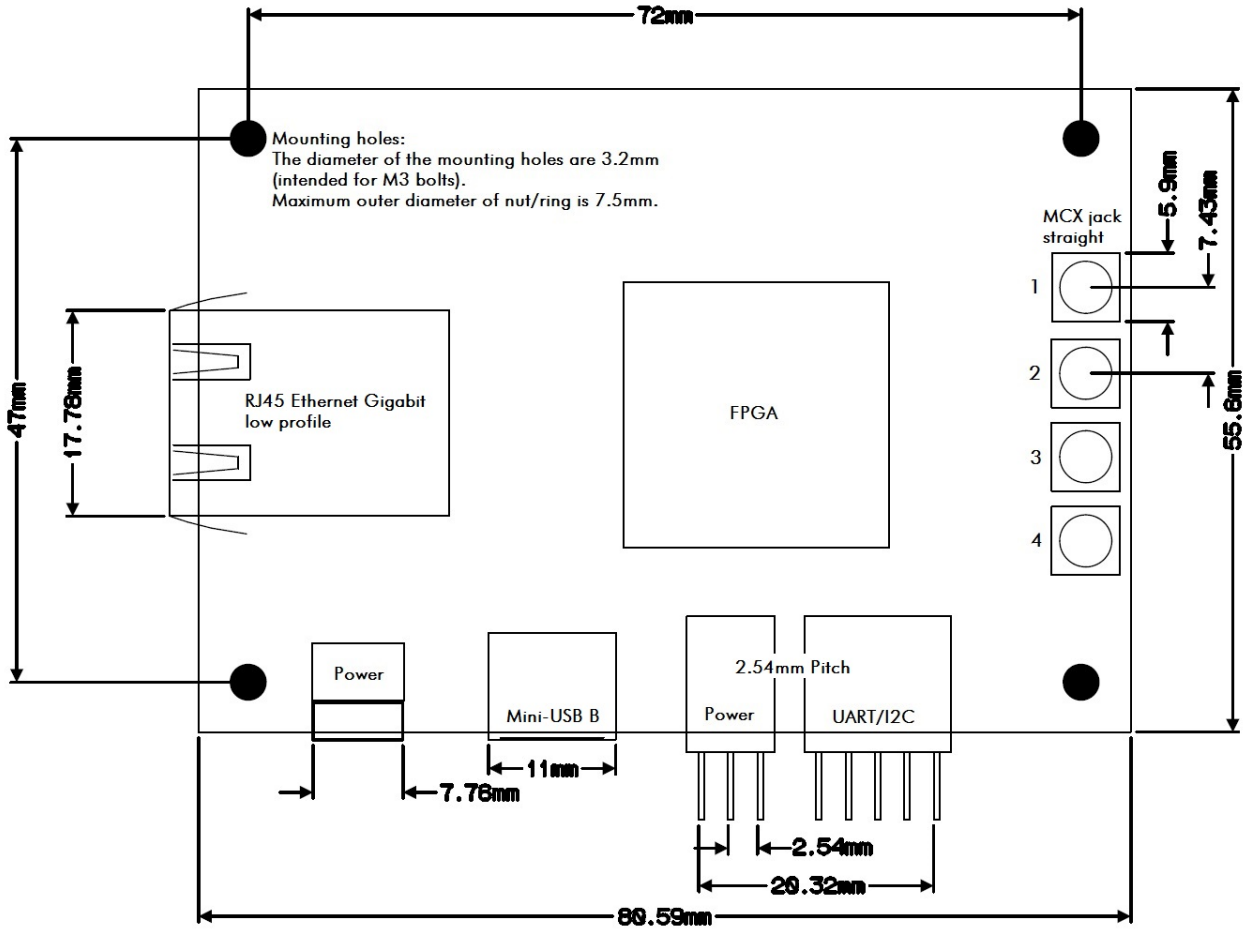
5.5 I²C control port

	Min	Typ	Max	Unit / Remarks
Interface port				
Connector		10-way pin header, 2.54 mm pitch		SDA/SCL Signals available on pin 9-10 of serial port control connector
Signals		SDA/SCL		
I²C				
Interface voltage		3.3		V
Speed			400	kbit/s
Device address	0x10	0x60	0x77	

5.6 Other specifications

	Min	Typ	Max	Unit / Remarks
Power				
Power supply voltage	11	12	13	V
Power consumption		4	5	W
Environmental				
Hazardous substances		RoHS compliant		
Flammability		UL-94 HB		
Operational Temperature	0		≥+45	°C
Mechanical				
Mounting hole diameter		3.2		mm (4 holes)
Dimensions W x H x D		81x56x17		mm (max)
Weight		39		g

Appendix A. Mechanical Dimensions



Warning: While mounting the DTM-3224, care should be taken not to damage components that are close to the mounting holes, both on the top and bottom side of the board.

Appendix B. DTM-3224 Development Kit

B.1 DTM-3224 development kit – Contents

The DTM-3224 development kit contains the following items:

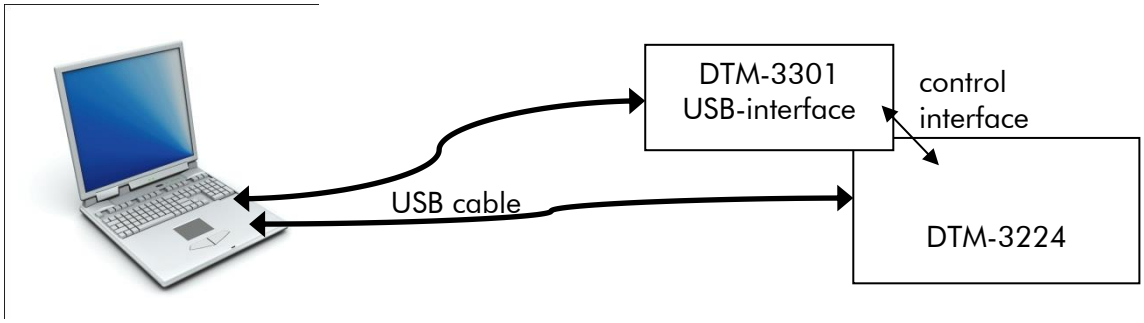
- DTM-3224 placed on four plastic studs
- DTM-3301 USB-to-RS232 interface board
- 12V/0.7A power supply
- USB cable type A to mini B
- MCX to BNC cable assembly (length = 130mm)

The development kit can be ordered from DekTec using type number DTM-3224-DEVKIT.

B.2 Using the DTM-3224 development kit

B.2.1 Hardware installation

The DTM-3301 USB-to-serial interface board has to be plugged on the Control pin-header of the DTM-3224. The pin-1 indication of the DTM-3301 should match the pin-1 indication of the DTM-3224. DTM-3301 is used to easily test the RS232 interface. Another option is to use a direct USB connection (both will be configured as a Virtual COM port).



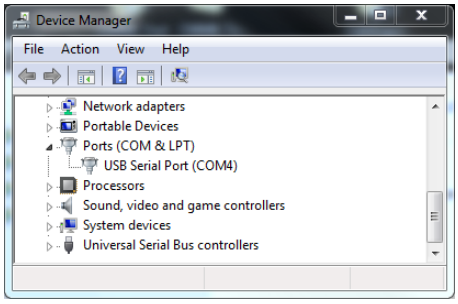
Connect the DTM-3224 to the power-supply using the power connector (see §3.2.7). The DTM-3224 will boot which will take a few seconds. During this time the LEDs on the DTM-3224 are green. Wait until the DTM-3224 input LED's start flashing green.

B.2.3 USB driver

Depending on your Windows version, it may or it may not be required to install a USB serial driver. On Windows 7 the driver is usually part of the Windows installation.

Connect the DTM-3224 or DTM-3301 to a USB port on the PC with the USB cable included in the development kit. For this step, it is not strictly necessary to power the DTM-3224 board, because the DTM-3301 is bus powered. After a while the USB connection to the interface board will become visible on the PC as a COM port in the Device Manager, section Ports (COM&LPT). The COM port number is displayed behind the USB Serial port entry, see the screen shot below.

If no USB COM port appears in the Device-Manager window, you can download and install the FTDI USB driver from http://www.ftdichip.com/Drivers/VCP.htm. After this step, you may also have to unplug and reconnect the USB cable.

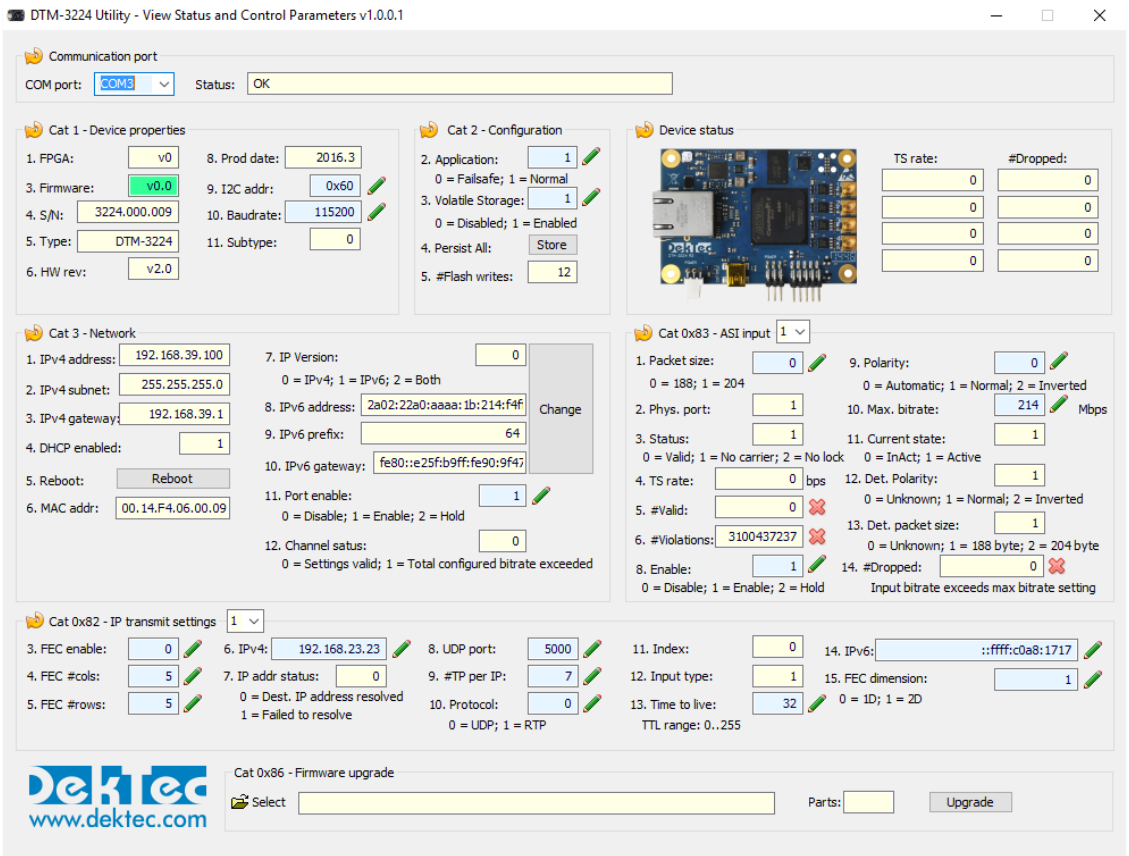


In this case the COM port number is COM4.

B.2.4 Debugger

Dtm3224Util is a GUI tool to view status, control settings and upload firmware to the DTM-3224. The ***Dtm3224Util*** debugger tool can be found on the DekTec USB flash drive. It can also be downloaded from the DekTec website. ***Dtm3224Util*** is an executable that can be run from any directory on your PC.

When started, the debugger enumerates serial ports and lets the user select the serial port to which the DTM-3224 is connected. When a valid serial port is selected, all registers are read from the DTM-3224 and shown in the GUI. Blue fields can be edited and written to the DTM-3224 by clicking the pen symbol to the right of the edit fields. Yellow fields are read only; they are read when the refresh arrow is clicked.



Appendix C. Command-Line Tool - DtmCmd

C.1 General description of DtmCmd

DtmCmd is a cross-platform (Windows & Linux) command-line tool for simple control of DTM-32xx devices. The user can read and write device settings, e.g. the command "`DtmCmd -r 1 5`" reads and prints the value of setting 5 in settings category 1. The most advanced capability of ***DtmCmd*** is upgrading the firmware of a DTM-32xx device.

To use the command-line tool under Windows, open a DOS box in a directory containing the ***DtmCmd*** executable. Each time ***DtmCmd*** is run, a single command specified with the command-line arguments is executed on the DTM-3224. See "`DtmCmd -?`" for help on the available commands.

You can specify the interface type (serial/I²C), interface settings and DTM address on the command line. The configuration settings are stored in file **`DtmCmd.ini`**. Every time ***DtmCmd*** starts, it first reads **`DtmCmd.ini`**, so that you don't need to specify the configuration settings every run of ***DtmCmd***.

C.2 Reading a setting from the DTM-3224

The following command reads device property ***Type*** (category 1, setting 5):

```
DtmCmd -interface Serial -serial COM3 -baudrate 9600 -addr 0x40
       -r 1 5
```

The following shortcut is equivalent once the configuration settings are available in **`DtmCmd.ini`**:

```
DtmCmd -r 1 5
```

The parameters used in this command have the following meanings:

- `-interface Serial` → Set the interface type to a serial COM port. The I²C interface is also supported by ***DtmCmd***.
- `-serial COM3` → Set the serial COM port identifier to COM port 3.
- `-baudrate 9600` → Set the baud rate to 9600bd.
- `-addr 0x40` → Set the address of the DTM-3224 to 0x40.
- `-r` → Set the command type to read.
- `1 5` → Specify command category 1 and setting 5.

This command results into the following output:

```
DtmCmd - DTM-32xx Command Utility v1.0.1 (c) 2013 DekTec Digital
Video B.V.

- Category          : 0x01 (Device properties)
- Setting           : 0x05 (Type)
- Index             : 0x00
- Interface         : Serial
- DTM address       : 0x40
- Serial path       : COM3
- Serial baud       : 9600

- Data read         : 3224
```

C.3 Writing to the DTM-3224

To demonstrate the writing of a setting, we write 0x1234 to the setting ***Number of parts*** (setting 7) of category ***Update firmware*** (category 0x80). This category requires an index to be sent to the DTM-3224. The index is added with the `-i` argument. The command below assumes that configuration settings are available in **`DtmCmd.ini`**:

```
DtmCmd -i 0 -w 0x80 7 0x1234
```

The parameters used in this command have the following meaning:

- `-i 0` → Set the index to 0.
- `-w` → Set the command type to write.
- `0x80 7 0x1234` → Specify command category 0x80, setting 7 and data 0x1234.

This command results into the following output:

```
DtmCmd - DTM-32xx Command Utility v1.0.1 (c) 2013 DekTec Digital
Video B.V.

- Category          : 0x80 (Firmware update type 1 (DTM-3224))
- Setting           : 0x07 (Number of parts)
- Index             : 0x00
- Interface         : Serial
- DTM address       : 0x40
- Serial path       : COM3
- Serial baud       : 9600

- Data written      : 4660
```

Appendix D. Communication Example

In the examples below, grey areas in the timing diagrams are sent by the DTM-3224, while white areas are sent by the master. The address of the DTM-3224 in these examples is ***0x12***.

Write command on RS-XXX serial interface

Figure 7 shows the write command of the FEC-enable at the receive channel settings (category 0x81, setting 4, index 0). All values are displayed as ASCII characters.

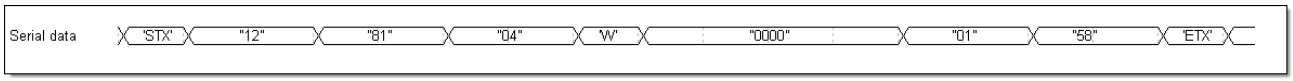


Figure 7: Write FEC enable in the receive channel settings

The command consists of the following parts:

- Start character 'STX'
- Two hexadecimal address characters ("12")
- Two hexadecimal category characters ("81")
- Two hexadecimal setting characters ("04")
- A write character 'W'
- Four hexadecimal index characters ("0000")
- Two hexadecimal characters ("01")
- Two hexadecimal checksum characters ("58", See Table 1)
- Stop character 'ETX'

Table 1: Checksum computation

Characters	ASCII value
STX	0x02
1	0x31
2	0x32
8	0x38
1	0x31
0	0x30
4	0x34
W	0x57
0	0x30
0	0x30
0	0x30
0	0x30
0	0x30
1	0x31
5	0x35
8	0x38
ETX	0x03
Sum:	0x2A8
Checksum:	0x58

Serial read command

Figure 8 shows the read command of the device type number (category 1, setting 5, no index). An index is not required for this category and the returned data consists of 4 bytes (int32). All values are displayed as ASCII characters.

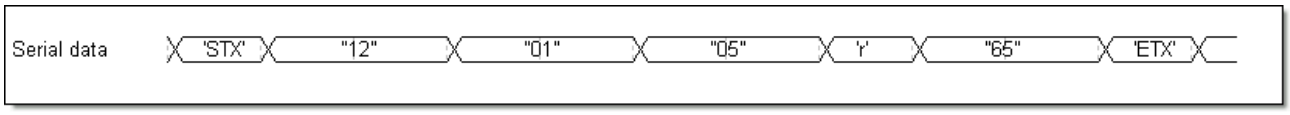


Figure 8: Read-command for the device type setting

The command consists of the following parts:

- Start character 'STX'
- Two hexadecimal address characters ("12")
- Two hexadecimal category characters ("01")
- Two hexadecimal setting characters ("05")
- A read character 'R'
- Two hexadecimal checksum characters ("85")
- End character 'ETX'

Figure 9 shows the two possible replies from the command in Figure 8. The replies are similar to the commands with the exception of the data-characters or the read character. On a successful command, the reply-data is set to the corresponding data (0x00000C80). When the received command cannot be executed, the read character is set to the ASCII character 'E'. In both cases the checksum is also updated.

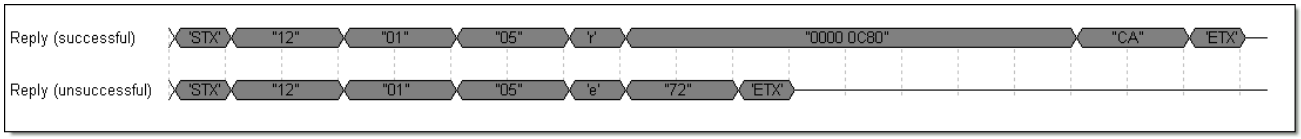


Figure 9: Reply after a device type read-command

I²C read command

Figure 10 shows the communication sequence used to issue a read subnet mask command (category 3, setting 2, no index). An index is not required for this category and the returned data consists of an IP-address.

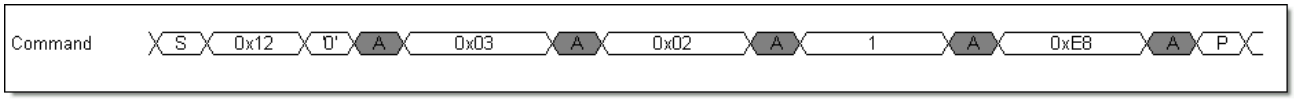


Figure 10: Send subnet-mask read-command

The command consists of the following bytes:

- Address and I²C write-bit (0x12 and '0')
- Category byte (0x03)
- Setting byte (0x02)
- Read byte (0x01)
- Checksum (0xE8, see Table 2). The checksum is computed with the address and without the I²C write-bit.

Figure 11 is the reply-sequence that may be executed after the read-command of Figure 10. After addressing this device, the bytes from the command are repeated followed with the 4-byte IP address.

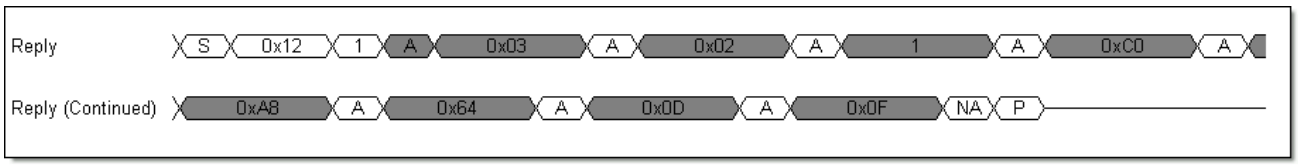


Figure 11: Read subnet-mask reply

The reply consists of the following bytes:

- Address and I²C write-bit (0x12 and '1')
- Category byte (0x03)
- Setting byte (0x02)
- Read byte (0x01)
- Four IP-address bytes (decimal 192.168.100.13 or hexadecimal C0.A8.64.0D)
- Checksum (0x0F, see Table 2)

Table 2: Checksum computation

	Command	Reply
Address	0x12	0x12
Category	0x03	0x03
Setting	0x02	0x02
R/W	'1'	'1'
Data byte 3	-	0xC0
Data byte 2	-	0xA8
Data byte 1	-	0x64
Data byte 0	-	0x0D
Sum:	0x18	0x1F1
Checksum:	0xE8	0x0F