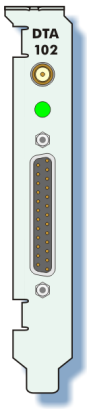
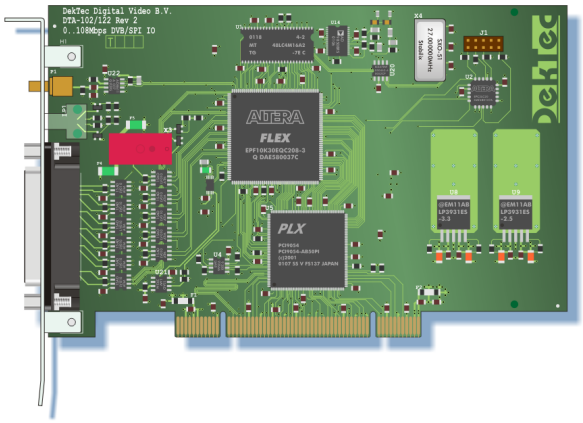
DVB/SPI Output Adapter for PCI Bus

- 0...108 Mbit/s, 8-MBytes Buffer
- Support for External Signal Adapters
- External Clock Input

FEATURES





- High-speed Transport-Stream output, compliant to DVB/SPI (Synchronous Parallel Interface) as defined in DVB document A010 rev 1 and EN50083
- Special support for external signal adapters to e.g. ECL or TTL
- Clock rate from 0 to 13.5 MHz with software-selectable clock source:
  - external clock input, or
  - on-board programmable clock generator
- Short-circuit detection with bi-colour LED indicator
- 8-Mbytes on-board buffer
- Hardware-assisted NULL packet insertion
- Packet sizes supported: 130, 188, 192 and 204 bytes; arbitrary size in raw mode
- Transmit modes for adding 16 bytes to 188-byte packets and for invalidating last 16 bytes of 204-byte packets
- DMA burst-mode transfers for optimal usage of PCI Bus
- Scatter/Gather DMA for efficient handling of fragmented host memory

APPLICATIONS

- LVDS output adapter for applications that generate an MPEG-2 Transport Stream: multiplexer, IP gateway, data generator, ...
- Transport-Stream generator with clock locked to external modulator
- With external adapter: test-stream generator with special physical interfaces, e.g. for set-top-box validation
- With external clock generator: test-stream generator for jitter-susceptibility measurements

KEY ATTRIBUTES

Parameter	Value
Physical Layer	DVB/SPI
DVB/SPI Connector	25-pin sub-D
External-Clock Connector	50 Ω SMA
DVB/SPI Clock	0...13.5 MHz
Transmit Bit Rate	0...108 Mbit/s
Clock-Generator Resolution	<0.1 Hz
Transmit Bit Rate Stability	±10 ppm
Packet Size in Bytes	130,188,192,204*
Target-Adapter Power Supply	5 V, 2 A

* Arbitrary packet size in raw mode

PCI-BUS CHARACTERISTICS

PCI rev 2.2, 32 bit, 33 MHz,
Master/Target, Universal 3.3V/5V

RELATED PRODUCTS

Type	Description
DTT-01/TTL	LVDS→**TTL** Converter for DTA-102
DTA-100	DVB/**ASI** Output Adapter for PCI Bus
DTA-122	DVB/SPI **Input** Adapter for PCI Bus
DTC-300	Transport-Stream Player **Software**
DTC-7X1	"Ray" **Stream Player**

Table of Contents

1. General Description 3
   1.1. Typical Application 3
      1.1.1. Transport-Clock Generation 3
      1.1.2. Target Adapter 3
   1.2. Software 3
   1.3. Block Diagram 4
      1.3.1. PCI-9054 Bus Interface 4
      1.3.2. Transmit FIFO 4
      1.3.3. Register Set 5
      1.3.4. Direct-Digital Synthesizer 5
      1.3.5. Short-Circuit Detection 5
      1.3.6. Target-Adapter Detection 5
   1.4. References 5
   1.5. Document Overview 6
2. External Interfaces 7
   2.1. Overview 7
   2.2. External-Clock Input 7
   2.3. LED Status Indicator 7
   2.4. DVB/SPI Output 8
      2.4.1. Standard DVB/SPI Mode 8
      2.4.2. Target-Adapter Mode 8
   2.5. Encoding of CODE Signal 9
   2.6. Target-Adapter Detection 9
3. Streaming Data 11
   3.1. DMA vs. Direct Writes 11
   3.2. Latencies 11
      3.2.1. PCI-Bus Latency 11
      3.2.2. Interrupt Latency 12
      3.2.3. Scheduling Latency 12
      3.2.4. Compensating Latencies 12
   3.3. Buffer Model 12
      3.3.1. DMA Buffer 13
      3.3.2. Transmit FIFO 14
   3.4. Synchronisation 14
      3.4.1. DMA-Driven 14
      3.4.2. Transmission-Driven 15
   3.5. Buffer Management 15
      3.5.1. Ping-Pong, DMA-Driven 15
      3.5.2. DMA-Driven, Start Up 16
      3.5.3. Ping-Pong, Transmission-Driven 16
      3.5.4. Transmission-Driven Start Up 17
4. Configuration Space 18
5. Target Address Space 20
   5.1. General Control (Offset = 00h) 22
      5.1.1. PE – Program Enable 22
      5.1.2. PRE – Protect Register Enable 22
      5.1.3. Reset – Software Reset 22
      5.1.4. AlteraRev – Firmware Revision 23
      5.1.5. TypeNum – Type Number 23
   5.2. Transmit Control (Offset = 04h) 23
      5.2.1. TxMode – Transmit Mode 23
      5.2.2. PckStuff – Packet Stuffing 25
      5.2.3. TxCtrl – Transmit Control 25
      5.2.4. TxDis – Disable LVDS Outputs 26
      5.2.5. PerIntEn – Periodic Interrupt Enable 26
      5.2.6. UflIntEn – Underflow Interrupt Enable 26
      5.2.7. SyncIntEn – Synchronisation-Error Interrupt Enable 26
      5.2.8. ShortIntEn – Short-Circuit Interrupt Enable 26
      5.2.9. ThrIntEn – Threshold-Crossed Interrupt Enable 26
      5.2.10. UseExtClk – Use External Clock 26
      5.2.11. EnaPwr – Enable Power 26
      5.2.12. ShortErr – Short-Circuit Error 26
      5.2.13. LedControl – Take over LED Control 26
      5.2.14. LedGreen – State of Green LED 26
      5.2.15. LedRed – State of Red LED 27
      5.2.16. ClrFifo – Clear Transmit FIFO 27
   5.3. Transmit Status (Offset = 08h) 27
      5.3.1. FifoFilled – Transmit-FIFO Filled 27
      5.3.2. SdramSize – SDRAM Size 28
      5.3.3. PerInt – Periodic Interrupt 28
      5.3.4. UflInt – Underflow Interrupt 28
      5.3.5. SyncInt – Synchronisation-Error Interrupt 28
      5.3.6. ShortInt – Short-Circuit Interrupt 28
      5.3.7. ThrInt – Threshold-Crossed Int 28
   5.4. Transmit Clock (Offset = 0Ch) 28
   5.5. Ext. Clock Count (Offset = 10h) 29
   5.6. FIFO Size (Offset = 14h) 29
   5.7. FIFO Load (Offset = 18h) 30
   5.8. Diagnostics (Offset = 1Ch) 30
   5.9. Loop-Back Data (Offset = 20h) 32
   5.10. Threshold Control (Offset = 24h) 32
      5.10.1. UpdDone – Update Done 33
      5.10.2. CmpA – Output Comparator A 33
      5.10.3. CmpB – Output Comparator B 33
      5.10.4. An/B – Update Threshold A or B 33
      5.10.5. ThreshA – Threshold A 33
      5.10.6. ThreshB – Threshold B 33
   5.11. FIFO Data (Offset = 40h…7Ch) 33
6. Vital Product Data 35
   6.1. Serial EEPROM Lay-Out 35
   6.2. VPD ID String 35
   6.3. VPD Read-Only Resources 35
   6.4. VPD Read-Write Resources 36
   6.5. Reading VPD Data 37
   6.6. Writing VPD Data 37
7. Transmit FIFO 39
   7.1. Coarse-Grain Model 39
   7.2. Fine-Grain Model 39
      7.2.1. Block Diagram 39
      7.2.2. PCI FIFO 40
      7.2.3. Burst FIFO 40
      7.2.4. Bulk FIFO 40
      7.2.5. Smoothing FIFO 40
   7.3. Real-Time Streaming 41
      7.3.1. Absolute Minimum Delay 41
      7.3.2. Practical Minimum Delay 41

1. General Description

The DTA-102 is a PCI adapter card for outputting a DVB/SPI-compliant Transport Stream. The transport rate of the channel is programmable to any value between 0 and 108 Mbit/s.

Note

- The DVB/SPI specification limits the maximum bit rate to 108 Mbit/s. However, the DTA-102 works well up to 150 Mbit/s.

1.1. Typical Application

The DTA-102 is typically deployed as DVB/SPI output card for MPEG-2 applications running on a PCI-based system. The low cost and high performance of generic computer platforms (e.g. Industrial PC[1]) can be leveraged to create cost-effective digital-video solutions.

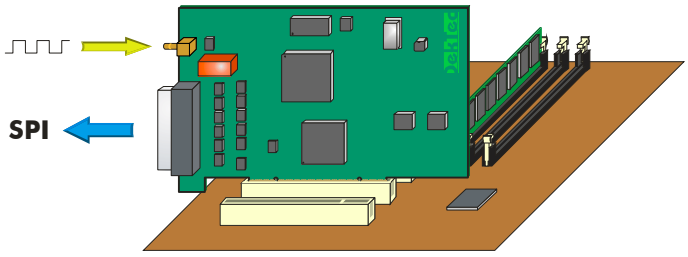


Figure 1. Typical application of the DTA-102 as DVB/SPI output stage in a PC-based digital-video application.

The DVB/SPI output signal makes the DTA-102 an excellent fit for direct connection to a modulator (cable/satellite/terrestrial), which is convenient for e.g. a Video-On-Demand server.

Furthermore, the DTA-102 supports a number of features that add versatility to the output adapter:

- Flexibility in generation of the transport clock (§1.1.1),
- Support for *Target Adapters* (§1.1.2),
- Short-circuit detection (§1.3.5).

1.1.1. Transport-Clock Generation

The DVB/SPI transport clock can originate from two sources, software selectable:

- On-board clock synthesizer;
- External clock input.

The on-board clock synthesizer enables stand-alone operation with a stable (10 ppm), high-resolution clock.

The external clock input allows a number of special configurations:

- The DTA-102 can be slaved to the clock output of a modulator.
- A high-quality external clock generator may be used to get a transport clock of exceptional accuracy and stability.
- Multiple DTA-102 boards may be synchronised to the same clock.
- A purposely distorted clock may be used, to test the jitter-susceptibility of the device under test.

1.1.2. Target Adapter

A *Target Adapter* is a "little box" outside the DTA-102 that converts the DVB/SPI signals from LVDS to another format, e.g. TTL. Optionally, synchronisation signals can be tailored to the target device, and a custom physical connector can be used.

Target adapters enable the construction of test generators that plug directly into the target device, e.g. a set-top box.

The DTA-102 supports the use of a Target Adapter in the following ways:

- A special mode reuses two DVB/SPI pins to supply power to the Target Adapter. No separate power supply is required!
- Identification of the target adapter. A simple scheme allows the identification of 35 different types of target adapters.
- Decoding of error signalling from target adapter to DTA-102.

1.2. Software

The DTA-102 hardware offering includes the following software:

- WDM device driver for Windows-2000 and Windows-XP;
- *DTAPI* library.

The WDM device driver implements "low-level" operations that require direct access to the DTA-102 hardware, such as initiation and coordination of DMA transfers, handling inter-

[1] Application of the DTA-100 is not limited to the PC: Any platform that supports the PCI bus can be used.

rupts and reading Vital Product Data (VPD, §6).

The DTAPI library is a thin layer of user-mode software that packages the driver functions into an easy to use API.

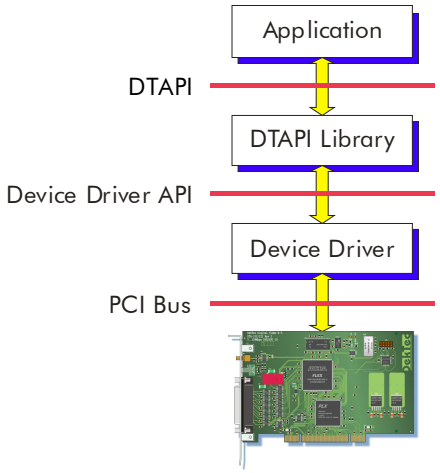


Figure 2. Software stack for DTA-102.

For customers that require a standard streaming solution, DEKTEC can offer a Transport-Stream Player application running on Windows-2000/XP. This program provides a convenient user interface for playing out Transport-Stream files. Features include:

- DVB PSI/SI decoding;
- Automatic transport rate computation;
- Continuous play with optional continuity-counter- and PCR- smoothing at stream wrap-around.

1.3. Block Diagram

Figure 3 shows a conceptual block diagram of the DTA-102. Transport-Stream data (and all other control data) enters the board via the *PCI-9054 Bus Interface*. The *Master-Control* block relays the packets to the *Transmit FIFO*. From there, LVDS buffers translate the signals to the required DVB/SPI levels.

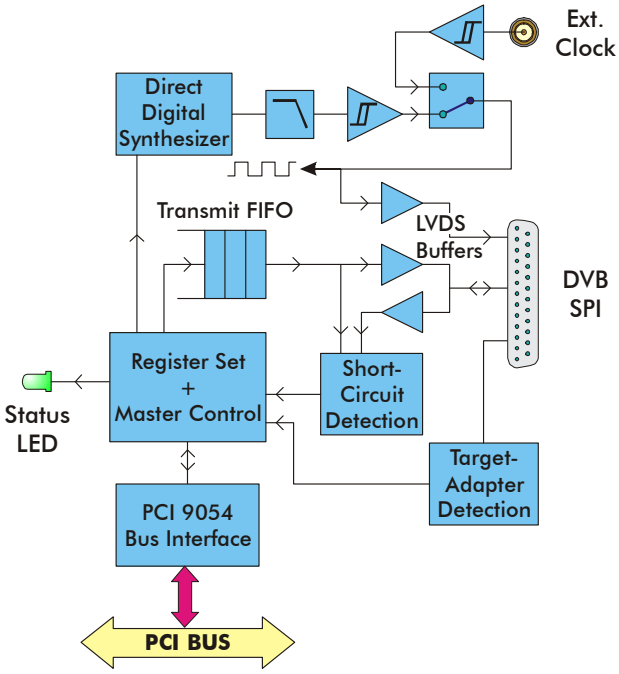


Figure 3. Conceptual block diagram of the DTA-102.

The other functional blocks in the diagram support the various special functions offered by the DTA-102. The major functional units are discussed below in separate subsections.

1.3.1. PCI-9054 Bus Interface

The DTA-102 uses the PCI-9054 IC made by PLX for interfacing to the PCI Bus. The PCI-9054 also implements the DMA functions required for high-speed streaming of transport packets, with minimal host interaction.

The PCI-9054 supports large host buffers with *scatter/gather* DMA mode. Such buffers may get fragmented through allocation / de-allocation of memory by OS-components. A scatter/gather list glues the buffer together without requiring software intervention.

Whenever appropriate, this specification provides information on the way the PCI-9054's registers should be used on the DTA-102. To obtain more details on the operation of the PCI-9054, please refer to the *PCI 9054 Data Book*.

1.3.2. Transmit FIFO

The DTA-102 contains a large (8-MBytes) buffer for Transport-Stream data: the Transmit

FIFO. A standard SDRAM is used to implement this buffer economically.

The function of the Transmit FIFO is to enable the sustained generation of a low-jitter DVB/SPI stream. The Transmit FIFO compensates for latencies on the PCI Bus and for scheduling jitter of the software that generates the Transport Stream.

When the Transmit FIFO underflows, the DTA-102 supports a mode to automatically insert null packets. Alternatively, in **Raw** mode the data-valid signal is set to '0' upon Transmit-FIFO underflow.

1.3.3. Register Set

Next to the registers in the PCI-9054, the DTA-102 contains a number of dedicated registers in PCI Memory Space (refer to §5 for syntax and semantics). With these Registers, the application software can configure and operate the DVB/SPI-specific features of the DTA-102.

1.3.4. Direct-Digital Synthesizer

The internal Transport-Stream clock is generated by a so-called direct-digital synthesizer. This circuit internal clock generator can be programmed to any frequency (byte clock) between 0 and 13.5 MHz, with a resolution better then 0.1 Hz.

An electronic switch selects between the output of the internal clock generator and the external clock input. The output of this switch is used to clock all logic used for transmission of the DVB/SPI signal.

1.3.5. Short-Circuit Detection

The DTA-102 reads back the LVDS signals on the DVB/SPI connector and compares the logical level to the intended level. In case of a discrepancy on any of the signal lines, the DTA-102 asserts a "short-circuit" condition.

When one or more outputs short-circuit, the status LED on the PCI bracket, positioned next to the DVB/SPI connector, flashes in red. The short-circuit condition can also be observed programmatically through a status register.

1.3.6. Target-Adapter Detection

A target adapter is identified by a resistor to ground. The target-adapter detection circuit can be used to measure the value of the resistor, and thus the type of the target adapter.

When the target-adapter type has been determined, a "window" can be programmed around the resistor value. This way, the software can easily test whether the resistor value has changed to a value outside the window, indicating that the cable is disconnected, or that the target adapter signals an error.

1.4. References

- *Interfaces for CATV / SMATV Headends and Similar Professional Equipment, DVB DOCUMENT A010 rev.1, May 1997* – This is the original DVB document that specifies physical interfaces for the interconnection of signal processing devices for professional digital-television equipment. One of the interfaces described is DVB/SPI.
  DVB document A010 document has also been issued as *CENELEC EN50083-9*.
- *ISO/IEC 13818-1, Information technology – Generic coding of moving pictures and associated audio information: Systems, April 27th, 1995*, also known as "MPEG-2 Systems" – Specification of the structure of a MPEG-2 Transport Stream.
- *DTAPI: C++ API for DTA-series of Digital-Video PCI-Bus Cards, DEKTEC Digital Video B.V., 2001* – Specification of **DTAPI**: the C++ interface to access the DTA-102 functions at a higher level of abstraction than would be possible using direct device-driver calls.
- *PCI 9054 Data Book, PLX Technology, V2.1, January 2001* – Specification of the PCI 9054, the chip used on the DTA-102 to interface with the PCI bus. Use this document if you need to program the PCI-9054 directly, e.g. when writing a custom device driver.
  The latest version of this document is available on line at http://www.plxtech.com.

- *PCI Local Bus Specification, Revision 2.2, December 18, 1998* – Formal specification of the protocol, electrical, and mechanical features of the PCI bus.

1.5. Document Overview

This specification describes the details relevant for operating the DTA-102. The information herein is primarily intended for device driver writers and for software developers that have to access the DTA-102 directly from a real-time operating system.

The WDM device driver and DTAPI library encapsulate many programming details of the DTA-102. Users of DTAPI may find this document useful for providing background information, but do not need to master each and every detail.

- *Section 1* introduces the main features of the DTA-102.
- *Section 2* describes the physical interfaces of the DTA-102.
- *Section 3* provides a detailed description of synchronisation and buffer-management, in order to stream data efficiently and reliably.
- *Section 4* lists the PCI Configuration-Space registers supported by the DTA-102.
- *Section 5* describes the *operational registers* on the DTA-102. These registers can be used to control and monitor the streaming of digital-video data.
- *Section 6* defines the structure of *Vital Product Data (VPD)* as supported by the DTA-102 and other DEKTEC PCI cards.
- Finally, *Section 7* details the structure of the Transmit FIFO, for applications that require the ultimate in performance.

2. External Interfaces

2.1. Overview

Next to a DVB/SPI connector, the DTA-102 supports an external clock input and a LED status indicator, as shown in Figure 4 below.

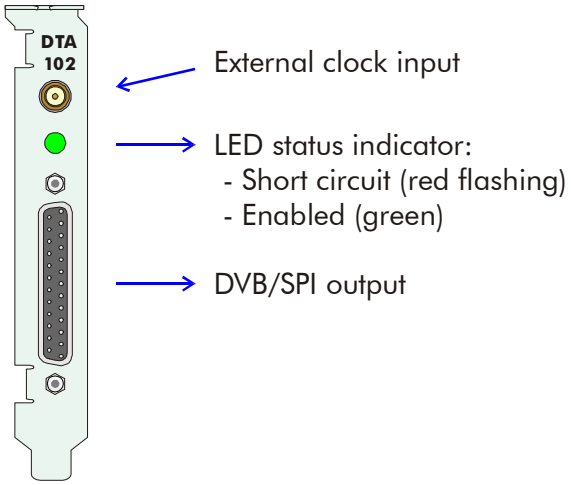


Figure 4. DTA-102 physical interfaces.

2.2. External-Clock Input

The DTA-102 supports an *external-clock input*, which enables usage of an external clock source to generate the byte clock for DVB/SPI transmission.

The characteristics of the external-clock input are listed in Table 1.

Table 1. External Clock Input – Characteristics

Parameter	Value
Physical Connector	SMA
Impedance	50Ω
Signal Level	>100mV top-top
Signal Shape	block, sine
Frequency	10kHz … 13.5 MHz

The external clock (instead of the internal clock generator) is used as DVB/SPI clock source if the `UseExtClk` field (§5.2.10) in the Transmit-Control register is set to '1'.

The frequency of the external clock can be measured in software using the External-Clock-Count register (§5.5). This measurement is independent from the value of the `UseExtClk` field: when the internal clock generator is used, the frequency of the external clock can still be measured.

Usage of an external clock generator may be advantageous for the following purposes:

- To slave the DVB/SPI signal to a high-precision clock generator or to a modulator;
- To generate a DVB/SPI test signal with a purposely low-quality clock signal, in order to test clock-jitter rejection by a device with a DVB/SPI Transport-Stream input.

2.3. LED Status Indicator

The LED on the PCI-bracket of the DTA-102 is a bi-colour (red/green) LED that indicates the status of the DVB/SPI output signal.

Table 2. LED Indicator – Meaning

LED Status	Meaning
LED off	LVDS buffers in tri-state
LED **green**	LVDS buffers enabled
LED **orange**	LVDS buffers enabled, power applied to target adapter
LED **red** flashing	Short-circuit error detected

Note

- Software may overrule the LED indication, and thus assign a different meaning to the LED Status patterns.

Just after power-up, the LED flashes a few times to indicate that the board is initialising. During this short period, the usual meaning of the LED indications does not apply: the LVDS buffers remain tri-stated. Instead, the LED pattern shows the board type and the firmware revision. An example start-up pattern is shown in Figure 5 below.

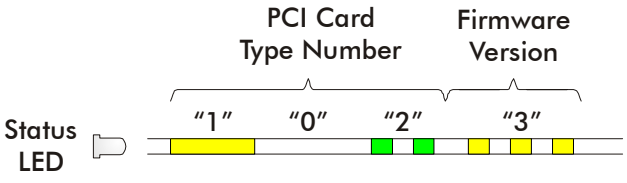


Figure 5. Power-up pattern for DTA-102 with (as example) firmware version 3.

2.4. DVB/SPI Output

The main interface of the DTA-102 is the DVB/SPI output on the 25-pin sub-D connector.

Two modes are supported:
- §2.4.1 Standard DVB/SPI Mode
- §2.4.2 Target-Adapter Mode

The mode is software-selectable with the *Enable-Power* field in the Transmit-Control register (§5.2.11).

2.4.1. Standard DVB/SPI Mode

In *Standard DVB/SPI Mode*, the Transport-Stream output of the DTA-102 conforms to the DVB/SPI specification (refer to §1.4 for references), with one small exception explained below the pin out of the 25-pin sub-D connector in Figure 6.

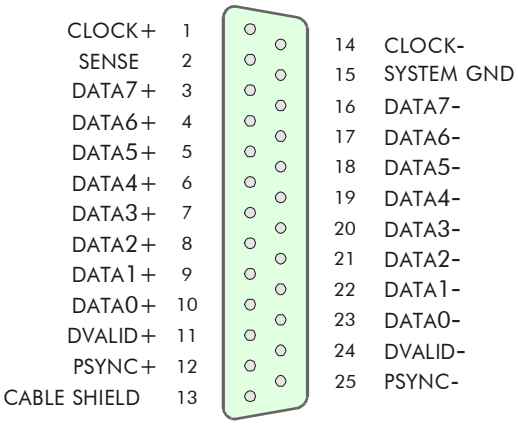


Figure 6. Pin assignment of 25-pin sub-D connector in *Standard DVB/SPI Mode*.

Notes
- The **CABLE-SHIELD** pin is connected to system ground on the DTA-102 board.
- In Figure 6, pin 1 is drawn in the upper-left corner. However, physically the pin is located at the bottom-right side (PCI bracket held in its normal position).

The usage of pin 2 on the DTA-102 is non-conformant. The DVB/SPI specification defines this pin as a **SYSTEM-GND** pin. The DTA-102 uses pin 2 as "**SENSE**" pin, to detect whether a target adapter is attached to the DTA-102. Target-adapter detection is discussed in §2.6.

2.4.2. Target-Adapter Mode

The other mode supported by the DTA-102 is *Target-Adapter Mode*. A target adapter is a device that converts DVB/SPI to a signal appropriate for the target. In this mode, the DTA-102 supplies power to the target adapter, so that no additional power supply is required.

Target-Adapter Mode is effected when the Enable-Power field in the Transmit-Control register (§5.2.11) is set to '1'.

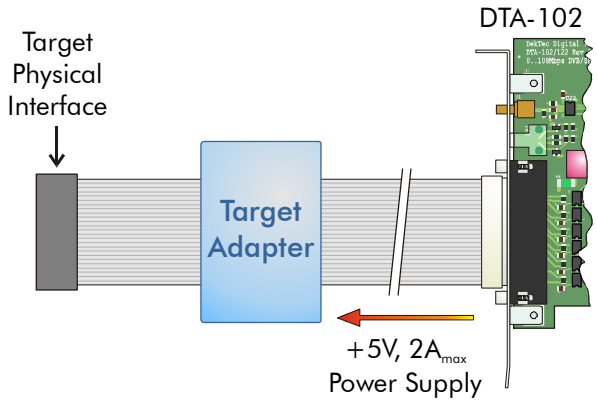


Figure 7. Target adapter connected to the DVB/SPI interface of the DTA-102.

The+5V power-supply is applied on two **POWER** pins, replacing the two pins that originally carried **PSYNC+/-**. Each power pin is fused and can carry a maximum current of 1A.

To keep all signal information available in Target-Adapter Mode, a special signal **CODE+/-** is introduced. **CODE** encodes **DVALID** and **PSYNC** into a single signal (refer to §2.5 for encoding of the **CODE** pin).

The altered pin out of the DVB/SPI connector in Target-Adapter Mode is shown in Figure 8.

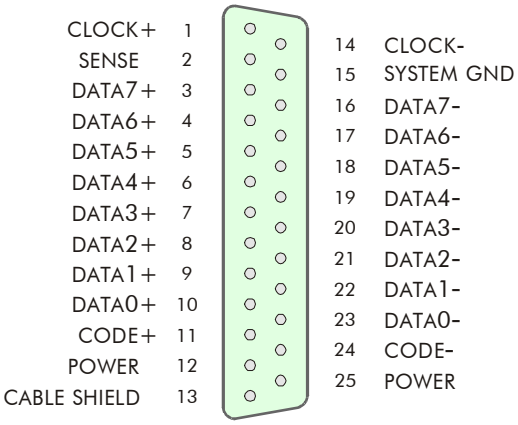


Figure 8. Pin assignment of 25-pin sub-D connector in *Target-Adapter Mode*.

2.5. Encoding of CODE Signal

In Target-Adapter Mode, the **CODE** signal is used to encode both **DVALID** and **PSYNC** on a single pin.

Note

- In **Raw** mode, the **CODE** signal directly outputs **DVALID**!
  A side-effect of this behaviour is that target adapters may get confused when the DTA-102 is operated in **Raw** mode: The target adapter interprets **DVALID** as if it were a **CODE** signal.
- Target adapters may still be used in **Raw** mode by (1) ignoring the **DVALID** and **PSYNC** output signals of the target adapter and (2) using **CODE** as data valid signal.
- Alternatively, a special version of the target adapter may be created that operates in **Raw** mode only. In this case, the target adapter should interpret the **CODE** line as data-valid signal.

The remainder of this section assumes that the DTA-102 is not operated in **Raw** mode.

The encoding of the signal on the **CODE** pin is listed in Table 3. The signal is synchronous to **CLOCK**. Commands are encoded in a simple serial format. While **CODE** is '0', nothing changes. A '1' is a "start-bit", followed by a 1-bit command code.

Table 3. Encoding of the **CODE** signal

Seq.	Mnem	Definition
0	**NO_CHANGE**	**PSYNC**: '0' **DVALID**: unchanged
10	**PULSE_PSYNC**	**PSYNC**: 1-clock cycle pulse **DVALID**: set to '1'.
11	**RESET_DVALID**	**PSYNC**: '0' **DVALID**: reset to '0'.

Figure 9 illustrates the relation between **CODE** and the derived signals **PSYNC** and **DVALID**.

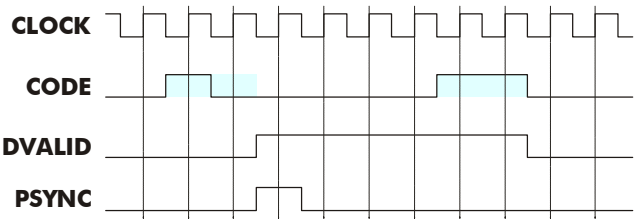


Figure 9. Timing diagram for encoding of **CODE** pin.

The two blue-shaded areas illustrate a **PULSE_PSYNC** and **RESET_DVALID** command respectively.

2.6. Target-Adapter Detection

A target adapter is identified by a resistor connected between **SYSTEM GND** and **SENSE** pin on the DVB/SPI connector.

The DTA-102 target-adapter detection circuitry has been designed to differentiate between 35 different resistor values. To achieve sufficient accuracy, the target-adapter resistor should be made from two 1% resistors in series. Refer to §5.10 for a description of target-adapter-detection circuitry, and how to measure the resistor value in software.

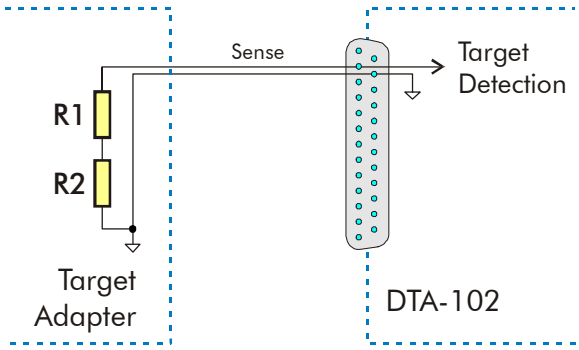


Figure 10. Two 1% resistors identify a target

adapter.

Table 4 provides the combination of standard resistors that best approximate the "ideal" resistor value for each target-adapter number.

Table 4. Target-Adapter – Resistor Values (Ω)

#	R1	R2	#	R1	R2
1	330	15	19	8200	150
2	560	39	20	8200	1000
3	820	47	21	10000	82
4	1000	150	22	10000	1000
5	1200	270	23	12000	150
6	1500	270	24	10000	3300
7	1800	330	25	12000	2700
8	2200	270	26	15000	1200
9	2700	120	27	18000	0
10	2700	560	28	18000	1800
11	3300	330	29	22000	220
12	3900	220	30	22000	2700
13	4200	390	31	27000	1000
14	4700	390	32	27000	4700
15	5600	56	33	33000	3300
16	5600	680	34	39000	2700
17	6800	100	35	47000	1800
18	6800	820			

Note

- Target-adapter numbers 1...15 (red-shaded cells) are reserved for use by DEKTEC. Target-adapter numbers 16...35 can be used for proprietary target adapters.

3. Streaming Data

The primary application of the DTA-102 is the streaming of MPEG-2 Transport Packets. The trickiest part of streaming data is achieving *real-time* operation, or at least – for streams without time stamps – achieving *efficient* operation. A well-balanced buffering scheme is required to compensate for hardware and software latencies.

This section first argues why DMA-based streaming offers superior performance compared to direct writing over the PCI Bus, followed by a description of hardware and software latencies that need to be overcome for real-time streaming. Then the buffer model adopted on the DTA-102 is described. Finally, synchronisation and buffer management are covered.

Using these techniques, §3.5 describes an approach for real-time operation of the DTA-102 in a relatively simple and robust way. This scheme has been implemented in the drivers that come with the DTA-102.

Note

- Of course, other approaches to real-time streaming with the DTA-102 may be feasible as well.
- For Transport-Streams without real-time content (e.g. a data generator), a number of simplifications can be applied[2].

3.1. DMA vs. Direct Writes

Transport-Stream data can be transferred to the DTA-102's Transmit FIFO in one of two ways:

1. The host[3] writes the data over the PCI Bus to the DTA-102.
2. The host first writes the data in a DMA Buffer in host memory. Then, the DMA controller on-board of the DTA-102 transfers the data from the DMA Buffer to the DTA-102.

In the first method, the host processor executes a long series of write instructions to the PCI Bus. Effectively, this slows down the processor speed to the PCI-Bus rate. Theoretically the write-to-PCI instructions could be interleaved with other instructions, but in practice, this is awkward.

The second method allows the processor to write to main memory, which is at least an order of a magnitude faster than writing to the PCI Bus. The DMA cycles from main memory to PCI Bus – used to transfer the data to the DTA-102 – are invisible to the host processor.

To sum up, the "direct-write" method is simple, but ties the processor to the PCI timing. The DMA method is more complex, yet much faster as a consequence of the vast difference in speed between writing to main memory and writing to the PCI Bus.

The remainder of this section elaborates techniques to transfer data using DMA.

3.2. Latencies

A continuous Transport Stream is essential for flawless operation in many MPEG-2 applications[4]. If not properly compensated, data-transfer latencies may lead to discontinuities.

This section describes the hardware- and software- latencies that need to be taken into account when streaming MPEG-2 packets with the DTA-102.

3.2.1. PCI-Bus Latency

The DTA-102 shares the PCI Bus with other bus masters that also compete for PCI cycles. The DTA-102 DMA Controller may have to wait a certain amount of time – the *PCI-Bus latency* – before it can acquire the bus and begin a DMA transfer.

---

[2] For data-only applications it is not required to avoid FIFO underflows at all costs, as null packets are automatically inserted. A simplified buffering scheme could be used. Still, DEKTEC recommends using the buffering scheme described later in this section, to avoid loss of performance.

[3] This is: the application program or driver running on the host.

[4] MPEG-2 applications involving video or audio will badly suffer from any uncorrected distortion of the Transport Stream. Even the stuffing of a single packet invalidates the entire transport steam (leading to PCR errors and potential T-STD problems).

Under normal conditions, the maximum duration of PCI latencies is in the order of a few microseconds. On a heavily loaded PCI Bus, latencies can be longer, but practical experience indicates that in all but pathological cases (see note) 2 ms can be safely taken as the absolute maximum PCI-Bus latency.

Note

- Cases are known[5] in which PCI latencies become unbounded. If real-time streaming is required, it is essential to check the host system fur such adversary conditions.

3.2.2. Interrupt Latency

Interrupt latency is the time between a hardware device raising an interrupt and software actually servicing the interrupt.

The DTA-102 hardware/software synchronisation methods (as described below in §3.4) rely on interrupts to signal certain hardware conditions to the software. The maximum interrupt latency has to be added to the total latency.

3.2.3. Scheduling Latency

The host CPU cannot dedicate all of its time to computing packets for the DTA-102: Other threads need processor cycles as well. *Scheduling latency* is the maximum time – in the worst-case scenario – the CPU still has to spend on other jobs, when new data needs to be processed for the DTA-102. Scheduling latency has to be added to the total latency too.

Scheduling latency is hard to grasp. It depends on many factors, like operating system, other software running on the host, relative priorities of threads and other hardware to be serviced.

3.2.4. Compensating Latencies

Adding all up, the total latency is the sum of PCI-Bus-, interrupt- and scheduling- latencies. The principal technique to compensate for these latencies is to transfer packets to the DTA-102 well in advance of actual transmission. The Transmit FIFO on-board of the DTA-102 functions as a temporary buffer, so that a smooth data stream can be provided at all times.

The size of the Transmit FIFO is programmable (§5.6). Two conflicting factors should be taken into account when choosing a value for the Transmit-FIFO size.

- *Latency tolerance*. The larger the Transmit FIFO, the more headroom exists for compensating latencies.
- *End-to-end delay*. In applications that process real-time input data[6], the Transmit FIFO acts as a delay buffer. A larger FIFO implies a longer end-to-end delay of the application. Note that, given a certain FIFO size, the delay is inversely proportional to the Transport-Stream bit rate.

DEKTEC cannot provide a cut-and-dried recipe for determining the optimal size of the Transmit FIFO. As for any hard real-time system, careful system analysis is required for the configuration at hand.

Good engineering practice demands that computed values are validated by experiment. A final safety factor (say 20%) will add to the robustness of the application.

3.3. Buffer Model

Transport-Stream data to be transmitted by the DTA-102 is buffered in a cascade of two buffers:

1. The *DMA Buffer*, located in host memory. The host processor writes data directly in this buffer.
2. The *Transmit FIFO*, located on the DTA-102. The DMA Controller on the DTA-102 transfers data bytes from the DMA Buffer, via the PCI Bus, to the Transmit FIFO.

The DMA Buffer should be divided in multiple (sub-)buffers, to avoid contention between host processor and DMA controller[7]. The host writes

[5] E.g. the host CPU writes continuously to a frame buffer on the PCI-Bus for a long period of time.

[6] This is: the application cannot process data ahead of time. In applications that can pre-fetch data, such as in a disk-reader application, end-to-end delay is less of an issue.

[7] It is possible to use a single DMA Buffer, but then CPU write operations to the buffer and DMA read opera-

to one DMA Buffer, while the DMA Controller reads from another DMA Buffer.

A scheme with two DMA Buffers is elaborated in §3.5 below. Of course, advanced buffering schemes with more than two DMA Buffers may also be used[8].

Figure 11 shows a snapshot of the double-buffer model in action. The buffer is split in DMA Buffer 1 and DMA Buffer 2, together forming one circular buffer. The blue-hatched area represents data that has been written to the buffer earlier.

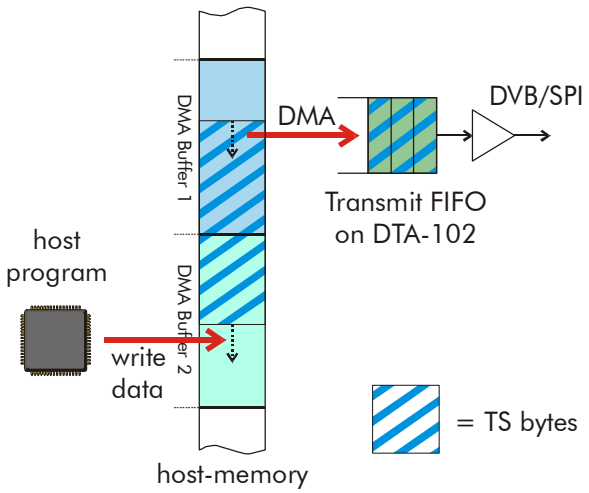


Figure 11. Double-buffer model. The two DMA Buffers are located in host memory. The Transmit FIFO is implemented in hardware on the DTA-102.

MPEG-2 data in DMA Buffer 1 is transferred to the Transmit FIFO by the DMA Controller on the DTA-102. At the same time, new packets are written in DMA Buffer 2 by the program running on the host.

When DMA Buffer 1 has been read empty, and DMA Buffer 2 has been filled completely, the function of both DMA Buffers is swapped.

3.3.1. DMA Buffer

A DMA Buffer is an array of bytes allocated in the application's address space. The start- and end- address of a DMA Buffer must be aligned on 4-byte boundaries. The byte at relative address 0 is the first byte that leaves the DTA-102, followed by the byte at address 1, etc.

Obviously, the DMA Buffer may not be virtual memory that is swapped out to disk. Either non-paged memory should be used, or the driver should ensure that the pages are locked into physical memory whenever the DTA-102's DMA Controller may access them.

A DMA Buffer maps to a contiguous address range in virtual-address[9] space. The DMA Buffer needs not be contiguous in physical-address space: Memory pages may be *scattered* over physical memory[10]. The PCI-9054 Scatter/Gather DMA mode can be used to transfer such a scattered DMA Buffer in one go, without requiring processor intervention to glue pages together.

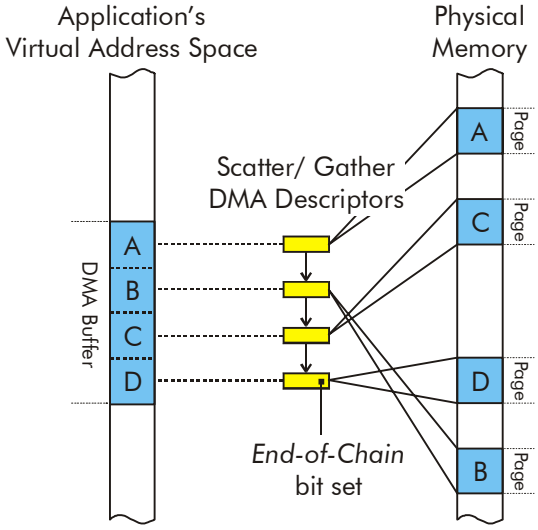


Figure 12. The DMA Buffer appears contiguous to the application, while pages are scattered over physical memory. Scatter/gather DMA allows transfer of all pages in the DMA Buffer in one sweep without processor intervention.

Scatter/Gather DMA uses a list of *Scatter/Gather DMA Descriptors* stored in (non-paged) host memory. This so-called *scat-*

tions from the buffer should be mutually exclusive in time.

[8] This specification does not provide further details on configurations with more than two DMA Buffers.

[9] It is not a strict requirement that the DMA Buffer is contiguous in virtual address space. Nonetheless, application programmers will find it very convenient.

[10] This means that the DMA Buffer may be allocated from a fragmented memory pool.

ter/gather list can be built at the same time as the DMA Buffer is allocated.

The last descriptor in the list shall have its *End-of-Chain* bit set. The corresponding interrupt can be enabled, so that the driver is alerted when the entire DMA Buffer has been transferred.

The descriptor syntax and the way to initialise and operate scatter/gather DMA are described in the PCI-9054 data book. Note that the DTA-102 uses demand-mode DMA. This implies that DMA channel 0 shall be used for performing the DMA transfers.

Note

- The scatter/gather mechanism incurs a little overhead per descriptor. Therefore, scatter/gather buffers should not be made too small, as this will lead to degraded performance. Buffers with the size of a memory page are fully acceptable.

3.3.2. Transmit FIFO

The contents of the DMA Buffer are transferred to the *Transmit FIFO* on-board of the DTA-102. The Transmit FIFO converts the write bursts to a smooth output stream: The hardware reads bytes at the output side of the Transmit FIFO reads bytes at a constant rate – the Transport-Stream rate – and converts the data bytes to DVB/SPI format.

For most applications, the Transmit FIFO can be considered a large conventional FIFO that can buffer 8-Mbytes of packet data. The maximum load of the Transmit FIFO can be artificially reduced through the FIFO-Size register (§5.5).

Notes

- The FIFO-Size register does not determine the 'real' FIFO size: The actual size of the FIFO is always 8 Mbytes (dependent on SDRAM size; §5.6). The FIFO-Size register is the high-water mark that controls the demand for DMA. Whenever the FIFO load becomes greater or equal than the value in the FIFO-Size register, the DTA-102 disables DMA until the FIFO load drops below FIFO Size again.
- It is hard to envisage an application that benefits from a reduced FIFO Size. Therefore, most applications can simply program FIFO Size to its maximum of 8 Mbytes.
- The hardware implementation of the Transmit FIFO is more complex than suggested by the model described above; Refer to §7 for particulars. The implementation details may be relevant for special applications, e.g. if one wants to achieve a low end-to-end delay at a low bit rate. Most regular applications need not bother about the finer points of the Transmit-FIFO implementation.

3.4. Synchronisation

The host processor generates Transport Packets, while the DTA-102 transmits them. Obviously, packet generation must be synchronised to packet transmission, or discontinuities in the outgoing Transport Stream will occur.

The DTA-102 has hardware support for two distinct synchronisation methods:

- DMA-driven; Refer to §3.4.1;
- Transmission-driven; Refer to §3.4.2.

3.4.1. DMA-Driven

With *DMA-Driven* synchronisation, the host software locks generation of new packets to the completion of the last DMA transfer. Refer to §3.5.1 for how this works in combination with buffer management.

DMA-driven synchronisation works reliably because the DTA-102 hardware implements *demand-mode* DMA: DMA transfers are requested on the PCI Bus only as long as the DTA-102's Transmit FIFO has empty buffer capacity left. When the FIFO becomes *filled*[11], the DMA process stalls. When FIFO space is available again, DMA resumes.

In other words: The Transmit FIFO cannot overflow in DMA-driven operation. The handshaking hardware prevents this from happening.

[11] "Filled" in the sense that the load of the Transmit FIFO has become greater or equal than the value programmed in the FIFO-Size register.

3.4.2. Transmission-Driven

An alternative method of synchronisation is *Transmission Driven*: the generation of new packets is locked directly to the transmission of (previously-computed) packets.

First, the Transmit FIFO is filled with an initial load of packets. Then, transmission is started and the host starts tracking the number of transmitted packets. When *x* packets have been transmitted, the host software generates precisely *x* new packets and transfers them to the Transmit FIFO. The host waits until another *x* packets have been sent, computes *x* new packets, etc.

The number of transmitted packets can be tracked with the so-called *Periodic Interrupt*, which is generated precisely every 2^{21} cycles of the 27-MHz reference clock (this is: about once every 77.7 ms). The number of packets transmitted in this time interval is easily computed by multiplying $\frac{2^{21}}{27*10^6}$ with the packet rate.

Example

- Suppose 188-byte packets are transmitted at 40 Mbps, corresponding to a packet rate of 26596 Transport Packets per second. Multiplying by 77.7 ms yields the number of packets transmitted between two periodic interrupts: 2066.
  So, after each periodic interrupt the program may write 2066 new packets to the DTA-102.

The example above uses approximations. In practice, the synchronisation method will work correctly in the long run only if the *full fractional* part ("bits behind the comma") is taken into account. Otherwise, slight timing errors may accumulate and cause underflow or overflow in time.

Taking into account "all bits" involves using in the computations the full 32-bits value programmed in the Transmit-Clock (§5.4) register. As the periodic interrupt and the transmit clock are derived from the same master 27-MHz clock oscillator, the number of packets transmitted can be tracked exactly.

3.5. Buffer Management

This section discusses how to manage DMA Buffers such that synchronisation of packet generation by the host and packet transmission by the DTA-102 is achieved.

3.5.1. Ping-Pong, DMA-Driven

As explained in §3.3, efficient streaming of data to the DTA-102 requires at least two DMA Buffers. The host program computes new packets and writes them to one buffer. At the same time, the DMA Controller on the DTA-102 reads data from the other buffer. When both DMA is done and a new buffer with packet data has been filled, the DMA Buffers swap function. This process continues ad infinitum.

Note

- An advanced buffer-management scheme may use more than two DMA Buffers. However, for the majority of applications a double-buffering will suffice.

The use of two buffers that swap function after each cycle – also known as *Ping-Pong* buffering – is illustrated in Figure 13.

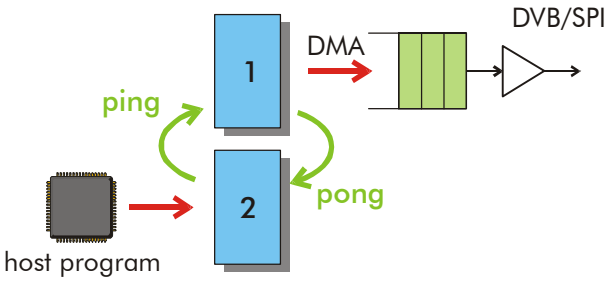


Figure 13. Ping-Pong buffering. The host writes packets in one buffer, while the DMA Controller reads packets from the other buffer. When both are finished, the "Ping-Pong" swap is executed.

DMA-Driven flow control in a double-buffering scheme is illustrated in the message-sequence chart shown in Figure 14.

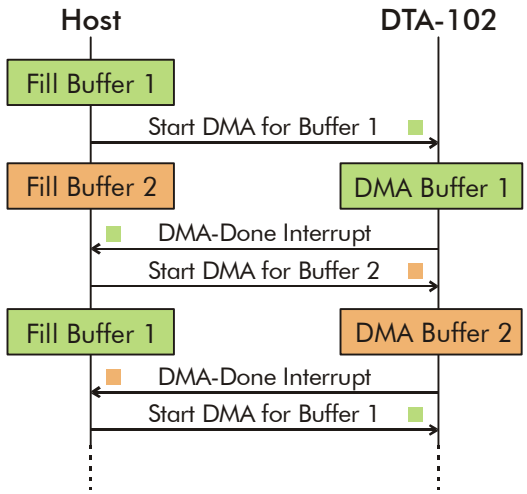


Figure 14. Ping-Pong buffer management using DMA-Driven flow control. After the host has filled a buffer and DMA on the other buffer is done, the buffers swap function.

The DMA-Done Interrupt is the handshake signal. It triggers the host to compute new packets and to initiate a new DMA transfer.

Note

- While waiting for the DMA-Done Interrupt, it is opportune for a device driver to sleep the process and give another process a chance to run.

It is instructive to ponder on the limiting factor in the Ping-Pong process: Host or DMA?

In the beginning, when the Transmit FIFO is not filled yet, the host will be the bottle neck. DMA transfers complete quickly, because DMA is only constrained by the PCI Bus.

After some time, when the Transmit FIFO is filled, DMA will become the limiting factor. The effective average rate of DMA will drop to the transmit rate, because of demand-mode DMA (Refer to §3.4.1). The host has to wait for the DMA-Done Interrupt before it can execute the next *Ping-Pong* swap.

3.5.2. DMA-Driven, Start Up

The robustness of the start-up phase of the Ping-Pong process can be improved by pre-loading the Transmit FIFO with data before starting the Ping-Pong process.

This can be accomplished with the Transmit-Control field (§5.2.3) in the Transmit-Control register. Preloading is enabled by setting Transmit Control to **Hold** and issuing DMA transfers until the Transmit FIFO is (almost) filled. Then, Transmit Control is set to **Send**. The DTA-102 starts packet transmission and the Ping-Pong process may begin.

The controlled start-up method prevents potential underflow of the Transmit-FIFO in the start-up phase[12].

3.5.3. Ping-Pong, Transmission-Driven

The Ping-Pong buffering scheme can also be used in combination with Transmission-Driven flow control. Figure 15 shows two Ping-Pong cycles using Transmission-Driven synchronisation, assuming that the process has already stabilised.

The most obvious difference with DMA-Driven flow control is that the host software now has to wait for both DMA done and the periodic interrupt before proceeding with the next Ping-Pong cycle[13].

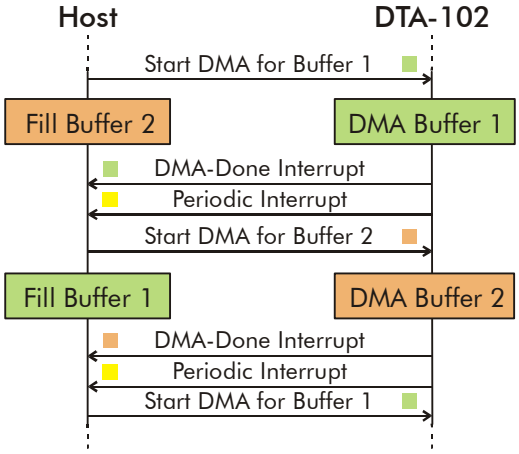


Figure 15. Ping-Pong buffer management using Transmission-Driven flow control. The buffers swap function when both the DMA-Done Interrupt and the Periodic Interrupt have occurred.

[12] If the process is started with Transmit Control set to **Send**, the Transmit FIFO may underflow early in the process (e.g. due to PCI-Bus latency), because no buffer load has been accumulated yet for compensation.

[13] Under normal circumstances, DMA is much faster than packet transmission. This means that the DMA interrupt will almost always occur long before the periodic interrupt arises. However, for robustness the software should also wait for the DMA interrupt.

A minor complication specific to Transmission-Driven flow control is that, in general, the number of packets transmitted between two Periodic Interrupts is not integer. The number of packets computed in cycle N must be rounded to the nearest integer. The length of the DMA transfer in cycle N+1 must be adapted to this size.

Note

- In Transmission-Driven flow control it makes no sense to artificially reduce the FIFO size. The FIFO-Size register should be programmed to its maximum value.

3.5.4. Transmission-Driven Start Up

The start-up procedure for Transmission-Driven flow control is somewhat more complicated then the procedure for DMA-Driven flow control.

In the start-up phase, the Transmit FIFO has to get an initial load. Figure 15 shows an example start-up procedure in which two DMA-Buffer loads are transferred to the Transmit FIFO.

Note

- This example uses initial loading in the "Ping-Pong way". A method with a single buffer could have been used too.

The start-up scenario proceeds through the following steps, assuming that the board and DMA Buffers have been properly initialised.

1. Host sets `TxCtrl` to **Hold**.
   This enables DMA, but no packets will be transmitted yet.
2. Host fills DMA Buffer 1 with packets, and starts DMA.
3. Host fills DMA Buffer 2, while the DTA-102 reads DMA Buffer 1 in parallel.
4. Host waits for DMA-Done interrupt for DMA Buffer 1, then starts DMA for DMA Buffer 2.

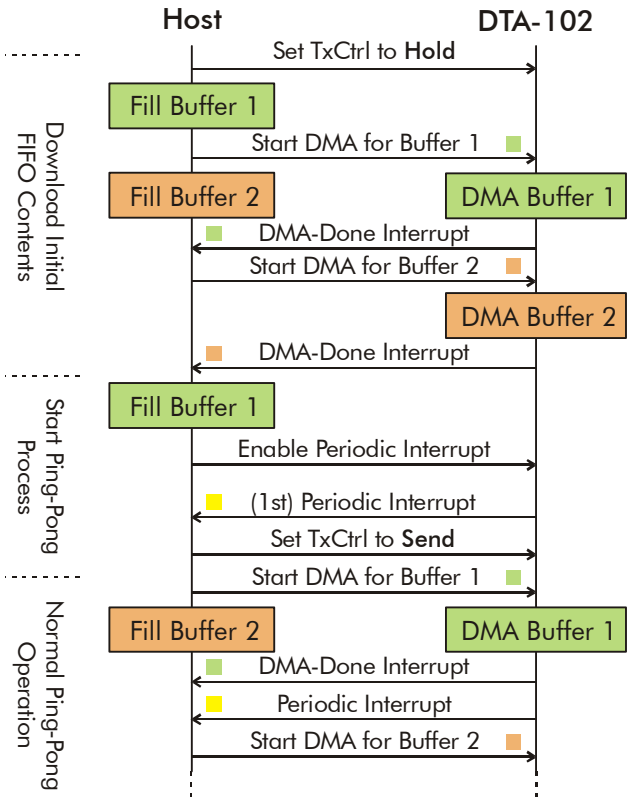


Figure 16. Start-up phase of Ping-Pong buffer management with Transmission-Driven flow control. In this scenario, the initial load of the Transmit FIFO is set to 2 times the size of the DMA Buffer.

5. Host waits for DMA-Done interrupt for DMA Buffer 2.
   The Transmit FIFO now has an initial load.
6. Host fills DMA Buffer 1 with packet data. This will be the first buffer that is transferred in "Normal Ping-Pong Operation".
7. Host enables Periodic Interrupts and waits for the occurrence of the first Periodic Interrupt.
8. Host sets `TxCtrl` to **Send**.
9. Normal Ping-Pong Operation is entered.

4. Configuration Space

The DTA-102 acts as a single logical PCI Bus device. It implements the configuration registers required for identifying the device, control PCI Bus functions, and provide PCI Bus status.

Table 5 displays the address map of registers defined in configuration space:

- Black fields indicate configuration registers supported by the DTA-102.
- Red-text cells represent registers supported by the PCI bridge chip, but not used for operating the DTA-102.
- Grey-text cells represent registers defined in the PCI Local Bus Revision 2.2 specification, but not supported on the DTA-102.

Table 5. Configuration Space – Address Map

Address Offset	Byte			
	3	2	1	0
00h	Device ID		Vendor ID	
04h	Status Register		Command Register	
08h	Class Code			Revision ID
0Ch	BIST	Header Type	Latency Timer	Cache Line Size
10h	PCI Base Address 0; used for memory-mapped configuration registers (PCI 9054)			
14h	PCI Base Address 1; not used			
18h	PCI Base Address 2; used for memory-mapped operational registers[14]			
1Ch	PCI Base Address 3; not used			
20h	PCI Base Address 4; not used			
24h	PCI Base Address 5; not used			
28h	*Card Bus CIS Pointer; not supported*			
2Ch	Subsystem ID		Subsystem Vendor ID	
30h	Expansion ROM Base Address Register; not used			
34h	*Reserved*			Next_Cap = 40h
38h	*Reserved*			
3Ch	Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line
40h	Power Management Capabilities; not used		Next_Cap = 48h	Capability ID = 01h
44h	Power Management Register; not used			
48h	Hot Swap; not used		Next_Cap = 4Ch	Capability ID = 06h
4Ch	VPD; used for manufacturing / service		Next_Cap = 00h	Capability ID = 03h
50h	VPD; used for manufacturing / service			

Table 6 shows a short description of the registers in configuration space.

[14] Operational registers are mapped in "Local Address Space 0" of the PCI 9054.

Table 6. Configuration Space – Register Overview				
Register	**Bits**	**RW***	**Value**	**Short Description**
Vendor ID	16	R	10B5h	Identifies PLX as manufacturer of the PCI interface chip.
Device ID	16	R	9054h	Identifies the PCI interface chip (PCI 9054).
Command Register	16	RW	-	Provides coarse control on the ability to generate and respond to PCI cycles.
Status Register	16	RWC	-	Status of PCI-Bus relevant events.
Revision ID	8	R	0	Revision number of your DTA-102.
Class Code	24	R	FF0000h	Generic function of the DTA-102.
Cache Line Size	8	R	16	System cache line size in units of 32-bit words.
Latency Timer	8	RW	-	Amount of time in PCI-Bus-clock units that the DTA-102 may retain ownership of the PCI Bus.
Header Type	8	R	0	Specifies layout of configuration addresses 10h through 3Fh and single / multiple functions.
BIST	8	R	0	PCI Built-In Self Test (BIST).
PCI Base Address 0	32	RW	-	Memory attributes and base memory address for memory accesses to PCI 9054 registers
PCI Base Address 2	32	RW	-	Memory attributes and base memory address for memory accesses to *Local Address Space 0*, which is used to access the DTA-102's operational registers (Refer to Table 7).
Subsystem Vendor ID	16	R	14B4h	Identifies the manufacturer of the DTA-102. Subsystem Vendor ID and Subsystem Device ID are leased from Philips BE.
Subsystem Device ID	16	R	D102h	Identifies the PCI card as a DTA-102.
Interrupt Line	8	RW	-	Interrupt line routing information.
Interrupt Pin	8	R	01h	Interrupt pin used by the DTA-102.
Minimum Grant	8	R	10h	Length of time (in 250-ns units) the DTA-102 would like to retain master ship of the PCI Bus.
Maximum Latency	8	R	1Ah	Frequency in which the DTA-102 would like to gain access to the PCI Bus.

5. Target Address Space

The DTA-102's operational registers are mapped in Local-Address Space 0 of the PCI 9054. The PCI Base address of these registers is specified in BAR2. All accesses to the operational registers shall be 32-bit transfers.

Table 7. Operational Registers – Memory Map

Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
00h	0 0 0 0 0 0 0 0	General Control		
04h	0 0 0 0 0 0 0 0 0	Transmit Control		
08h	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Transmit Status		
0Ch	Transmit Clock			
10h	External-Clock Count			
14h	0 0 0 0 0 0 0 0	FIFO Size		
18h	0 0 0 0 0 0 0 0	FIFO Load		
1Ch	Diagnostics			
20h	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Loop-Back Data		
24h	0 0 0 0 0 0 0 0	Threshold Control		
28h … 3Ch	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			
40h … 7Ch	FIFO Data			

Table 8. Operational Registers – Register Overview

Register	Field	Bit Pos	#	RWC*	Short Description
General Control	PE	0	1	RW	Serial EEPROM Program Enable
	PRE	1	1	RW	Serial EEPROM Protect Register Enable
	Reset	2	1	W	Reset DTA-102 circuitry
	reserved	7...3	5	R	Not used
	AlteraRev	15...8	8	R	Firmware Revision
	TypeNum	23...16	8	R	Type Number: 102 for DTA-102
Transmit Control	TxMode	1...0	2	RW	Transmit Mode: 188/192/...
	reserved	2	1	R	Not used
	TxModeExt	3	1	RW	Extension bit to TxMode[1..0]
	PckStuff	4	1	RW	Null Packet Stuffing On/Off
	TxCtrl	6...5	2	RW	Transmit Control: Idle/Hold/Send
	TxDis	7	1	RW	Disable (tri-state) LVDS outputs
	PerIntEn	8	1	RW	Periodic-Interrupt Enable
	UflIntEn	9	1	RW	Transmit-FIFO-Underflow Interrupt Enable
	SyncIntEn	10	1	RW	Synchronisation-Error Interrupt Enable
	ShortIntEn	11	1	RW	Short-Circuit-Detected-Interrupt Enable

Table 8. Operational Registers – Register Overview					
Register	**Field**	**Bit Pos**	**#**	**RWC***	**Short Description**
	ThrIntEn	12	1	RW	Threshold-Crossed Interrupt Enable
	reserved	15...13	3	R	Not used
	UseExtClk	16	1	RW	Use External Clock
	EnaPwr	17	1	RW	Enable Power to Target Adapter
	ShortErr	18	1	R	Short-Circuit Error detected on LVDS outputs
	LedControl	19	1	RW	Take over LED Control
	LedGreen	20	1	RW	State of Green LED (if LedControl is set)
	LedRed	21	1	RW	State of Red LED (if LedControl is set)
	ClrFifo	22	1	W	Clear Transmit FIFO
Transmit Status	FifoFilled	0	1	R	Transmit-FIFO-Filled flag
	reserved	3...1	3	R	Not used
	SdramSize	7...4	4	R	SDRAM Size = Maximum FIFO size
	PerInt	8	1	RC	Periodic Interrupt
	UflInt	9	1	RC	Transmit-FIFO-Underflow Interrupt
	SyncInt	10	1	RC	Synchronisation-Error Interrupt
	ShortInt	11	1	RC	Short-Circuit-Detected Interrupt
	ThrInt	12	1	RC	Threshold-Crossed Interrupt
Transmit Clock	TxClock	31...0	32	RW	Internal Clock Generator
Ext. Clock Count	ExtClkCnt	31...0	32	RW	External-Clock Counter
FIFO Size	FifoSize	23...0	24**	RW	Size of Transmit FIFO in #bytes
FIFO Load	FifoLoad	23...0	24**	R	Current Load of Transmit FIFO in #bytes
Diagnostics	SfData	7...0	8	R	Data at output of Smoothing FIFO
	LoopBack	8	1	RW	Loop-Back mode
	SfDtValid	9	1	R	Smoothing-FIFO Data Valid
	BfFull	10	1	R	Burst-FIFO Full flag
	DmaReq	11	1	R	DMA Request
	BfCanBurst	12	1	R	Burst-FIFO Can-Burst
	SfCanBurst	13	1	R	Smoothing-FIFO Can-Burst
	SfLoad	23...14	10	R	Smoothing-FIFO Load in #bytes
	BfLoad	31...24	8	R	Burst-FIFO Load in #32-bit words
Loop-Back Data	SfDataNxt	7...0	8	R***	Data at output of Smoothing FIFO

Table 8. Operational Registers – Register Overview					
Register	Field	Bit Pos	#	RWC*	Short Description
Threshold Control	UpdDone	0	1	RC	Update of threshold done
	CmpB	1	1	R	Output of comparator B
	CmpA	2	1	R	Output of comparator A
	An/B	3	1	RW	Update threshold A ('0') or threshold B ('1')
	reserved	7...4	4	R	Not used
	ThreshA	15...8	8	RW	Threshold A
	ThreshB	23...15	8	RW	Threshold B
FIFO Data	FifoData	31...0	4x8	RW	Transport-Stream data: 4 bytes at a time

* R=Readable, W=Writeable, C=Clearable (clear when a '1' is written to bit position).
** Number of bits depends on SdramSize. Shown size (24-bits) is valid for 8-Mbyte SDRAM.
*** Reading from the Loop-Back Data register causes next byte to be read from smoothing FIFO.

5.1. General Control (Offset = 00h)

Table 9. General-Control Register – Format		
Bit	Mnem	Description
0	PE	Serial EEPROM Program Enable
1	PRE	Serial EEPROM Protect Register Enable
2	Reset	Soft Reset
7...3	*reserved*	
15...8	AlteraRev	Firmware Revision
23...16	TypeNum	Value=102
Bits 31...24 of this register are tied to '0'.		

The General-Control register has a common structure for all DTA-1xx PCI adapter cards. The register contains control bits that are independent of the specific functions of the board.

5.1.1. PE – Program Enable

The Program-Enable field directly controls the PE signal of the serial EEPROM. The EEPROM can only be programmed if this bit is set to '1'. In normal operation, PE should remain '0'.

5.1.2. PRE – Protect Register Enable

The Protect-Register-Enable field directly controls the PRE signal of the serial EEPROM. It enables the write-protection mechanism in the EEPROM. In normal operation, this field should remain '0'.

Warning

- Issuing a write-protection command to the serial EEPROM is an irreversible operation. Incautious use of the PRE bit may destroy the Vital-Product Data read/write capability!

5.1.3. Reset – Software Reset

Writing a '1' to the Reset bit issues a "soft" reset to the DTA-102. The following fields and logic circuitry are affected:

- TxCtrl in the Transmit-Control register is reset to **Idle**.
- Interrupt-Status flags in the Transmit-Status register are cleared, except PerInt.
- The entire contents of the Transmit FIFO is cleared, including data in the Burst-, Bulk- and Smoothing FIFO (§7).
- The FIFO-Load register is reset to zero.
- A number of internal state machines are reset.

Other fields in the operational registers are not affected, notably:

- TxMode, TxClock and FifoSize.
- Loop-Back Mode in the Diagnostics register.
- Interrupt-Enable bits in the Transmit-Control register: interrupts that were enabled remain enabled.
- PE and PRE in the General-Control register.

This behaviour is by design, so that the data-processing pipeline in the DTA-102 can be reset without compromising other processes running on the PCI card.

Warning

- If the DTA-102 is reset while packets are being transmitted – including during Null-Packet Stuffing – then the packet that is currently being transmitted may be truncated.

The Reset bit is write-only. The write operation triggers the reset action: it is not required to reset the bit to '0' again. The next time a '1' is written to the Reset bit, the board will be reset again.

5.1.4. AlteraRev – Firmware Revision

This read-only field identifies the current revision level of the firmware programmed into the Altera FPGA on-board of the DTA-102.

Note

- The Firmware Revision level is independent of the DTA-102 *board* revision (which can be read from VPD).

5.1.5. TypeNum – Type Number

The Type-Number field identifies the board in a straightforward way. For the DTA-102, the field's value is fixed to **102**.

Next to this field, the board's type number is also encoded in the Vital Product Data (VPD), which is the primary source of descriptive data. The purpose of the Type-Number field is to provide a convenient way for device drivers to distinguish between different kinds of DTA-1xx boards at start-up.

5.2. Transmit Control (Offset = 04h)

The Transmit-Control register contains a number of fields that allow the device driver to control transmission-specific functions of the DTA-102.

Table 10. Transmit-Control Register – Format

Bit	Mnem	Description
1...0	`TxMode`	Transmit Mode
2	*`reserved`*	
3	`TxModeExt`	Extension to `TxMode`
4	`PckStuff`	Null-Packet Stuffing
6...5	`TxCtrl`	Transmit Control
7	`TxDis`	Disable LVDS Outputs
8	`PerIntEn`	Periodic-Interrupt Enable
9	`UflIntEn`	Transmit-FIFO-Underflow-Interrupt Enable
10	`SyncIntEn`	Synchronisation-Error-Interrupt Enable
11	`ShortIntEn`	Short-Circuit-Detected-Interrupt Enable
12	`ThrIntEn`	Threshold-Crossed Interrupt Enable
15...13	*`reserved`*	
16	`UseExtClk`	Use External Clock
17	`EnaPwr`	Enable Power
18	`ShortErr`	Short-Circuit Error
19	`LedControl`	Take over LED Control
20	`LedGreen`	State of Green LED
21	`LedRed`	State of Red LED
22	`ClrFifo`	Clear Transmit FIFO
Bits 31..23 of this register are tied to '0'.		

5.2.1. TxMode – Transmit Mode

Transmit Mode is a 3-bit (composite) field in the Control / Status register, consisting of two fields: `TxMode[1..0]` as the two LSBs (bit 1, 0) and `TxModeExt` as the MSB (bit 2).

Transmit Mode controls the size of transmitted packets and the automatic insertion / removal of extra bytes at the end of transport packets, as shown in the following table.

Table 11. Transmit Mode – Values		
Value	Mode	Definition
000	**188**	188-byte packets in FIFO, 188-byte packets transmitted.
001	**204**	204-byte packets in FIFO, 204-byte packets transmitted.
010	**Add16**	188-byte packets in FIFO, 204-byte packets transmitted; 16 extra bytes appended to each packet.
011	**Raw**	No notion of packets. Data bytes are transmitted "as is". Null-packet stuffing is not supported. No **PSYNC**.
100	**192**	192-byte packets in FIFO, 192-byte packets transmitted.
101*	**130**	No synchronisation; **PSYNC** is generated every 130 bytes.
110**	**Min16**	204-byte packets in FIFO, 188 bytes transmitted with DVALID asserted and 16 bytes with DVALID de-asserted

* Firmware revision ≥ 7
** Firmware revision ≥ 8

The default Transmit Mode is **188**. The DTA-102 assumes that the Transmit FIFO contains transport packets of 188 bytes. Packets are sent unmodified as plain 188-byte packets. The transport rate of the output stream (as defined in the MPEG-2 Systems specification) is equal to the transmit clock.

In mode **204**, the DTA-102 expects that transport packets in the Transmit FIFO consist of 204 bytes: an MPEG-2 compliant packet of 188 bytes, followed by 16 "extra" bytes. The software has full control over the contents of these 16 bytes[15].

Operation in Transmit Mode **Min16** is the same as in mode **204**, except that **DVALID** is de-asserted for the last 16 bytes of the 204-byte packets.

[15] For example, a fast processor (fed with a smart algorithm), may fill the extra bytes on the fly with a RS(204,188) forward-error-correction code, or any other packet attachment.

In mode **192**, transport packets are expected to consist of 192 bytes: 188 bytes for the MPEG-2 compliant packet and 4 "extra" bytes.

In mode **Add16**, the DTA-102 expects 188-byte packets (like in mode **188**), but now appends 16 null bytes to each packet. These bytes may act as a placeholder for external equipment that inserts e.g. a FEC code.

In Transmit Modes **192**, **204**, **Add16** and **Min16**, the transmit-clock rate (Tx-Rate) is greater than the Transport-Stream rate (TS-Rate). Additional clock cycles are required for outputting the extra bytes. These clock cycles are included in the transmit clock but excluded from the (MPEG-2 defined) Transport-Stream rate.

The relation between Tx-Rate and TS-Rate can be expressed in a formula. Refer to Table 17 in §5.4.

Table 12. Relation between TxRate and TSRate	
Transmit Mode	Relation
188	$TxRate = TSRate$
192	$TxRate = \frac{192}{188} * TSRate$
204, **Add16, Min16**	$TxRate = \frac{204}{188} * TSRate$

Note

- In transmit modes **188**, **192**, **204**, **Add16** and **Min16**, the DTA-102 expects that *correctly formatted* transport packets of the specified length are written to the Transmit FIFO. If this assumption fails, the DTA-102 will drop data, until packet synchronisation is achieved again.
- The DTA-102 does not implement "correctly formatted" in a very sophisticated way: The only requirement is that sync bytes appear at the right position…

In **Raw** mode, the DTA-102 has no notion of transport packets. Bytes in the Transmit FIFO are output without any processing. The packet sync signal (**PSYNC**) is not generated. Whenever the FIFO underflows, **DVALID** is set to low.

Furthermore packet stuffing is not possible (by definition, because in **Raw** mode the DTA-102 knows nothing about packets).

Raw mode can be used to simulate special Transport-Stream faults, e.g. an occasional packet with 187 or 189 bytes for testing synchronisation behaviour. However, as **PSYNC** is not generated, this option may have limited utility.

Note

- In **Raw** mode, the DTA-102 will *not* drop bytes from the Transmit FIFO: The notion of packets does not exist; therefore packets cannot be formatted incorrectly.

5.2.2. PckStuff – Packet Stuffing

The Packet Stuffing bit controls the behaviour of the output stream when no packet data is available in the Transmit FIFO.

- If Packet Stuffing is '0' (Off), nothing is sent: the output stream is not stuffed with null packets.
- If Packet Stuffing is '1' (On), the output stream is stuffed with null packets. The size of inserted null packets is matched to Transmit Mode.

In Transmit Mode **Raw**, Packet Stuffing is not supported: Both packet size and packet boundaries are unknown. Consequently, setting Packet Stuffing to '1' in **Raw** mode has no effect. When the Transmit FIFO underflows in **Raw** mode, the **DVALID** output signal is set to low and transmission of valid data stalls until data is written to the Transmit FIFO again.

The DTA-102 powers up with `PckStuff` set to '0', this is without packet stuffing.

5.2.3. TxCtrl – Transmit Control

The Transmit-Control field controls the packet-transmission process. After a power-up condition, Transmit Control is initialised to **Idle**.

Table 13. Transmit Control – Values

Value	Mnem	Definition
00	**Idle**	No DMA requests, no packet transmission.
01	**Hold**	Request DMA until Transmit FIFO is filled, no packet transmission.
10	**–**	Reserved.
11	**Send**	Read and transmit packets.

Whenever Transmit Control is set to **Idle**, the transmission process becomes inactive. A pending DMA transfer, if any, is suspended. No new DMA transfers are initiated. The DTA-102 stuffs the output stream with null packets if transmit mode is **188**, **204**, **Add16** or **Min16**, and Packet Stuffing is '1'. Otherwise, the DVB/SPI output stream is 'empty' (VALID line set to '0').

In **Hold** mode, packet transmission is stalled in the same way as in **Idle** mode (null-packet stuffing or empty output). However, unlike in **Idle** mode, DMA is enabled. If a DMA transfer is set up in the PCI 9054 – and the corresponding Scatter/Gather descriptor(s) has been created – DMA transfers will be initiated so that packet data is written to the Transmit FIFO. DMA continues until the DMA transfer terminates or the FIFO load reaches the high-level watermark specified in the FIFO-Size register.

When Transmit Control is set to **Send**, actual transmission of transport packets begins. New DMA data is requested through the PCI 9054, but only when sufficient space is available in the Transmit FIFO.

For a clean start-up of continuous packet transmission, it is required to sequence control operations in the following order:

- Allocate the DMA Buffer(s) in host memory and create the corresponding scatter/gather descriptor list[16];
- Write **Idle** to Transmit Control;
- Set-up DMA-transfer in PCI 9054;
- Write **Hold** to Transmit Control;
- Wait until DMA transfer is completed. Initi-

[16] Only required if DMA Buffer is scattered over physical memory.

ate additional DMA transfers in the PCI 9054 until the Transmit FIFO contains the desired initial load;
- Write **Send** to Transmit Control.

For data-only applications, the transition through **Hold** might be omitted. In that case, the DTA-102 may insert null packets due to Transmit-FIFO underflow, but this is irrelevant for data-only applications.

5.2.4. TxDis – Disable LVDS Outputs

Writing a '1' to this bit disables the LVDS Output buffers. Each LVDS pair is tri-stated.

The DTA-102 powers up with `TxDis` set to '1', this is with the output buffers *disabled*.

5.2.5. PerIntEn – Periodic Interrupt Enable

Writing a '1' to this bit enables[17] the *Periodic Interrupt* (§5.3.3). Refer to Table 14 for a description of the interrupt status flags.

5.2.6. UflIntEn – Underflow Interrupt Enable

Writing a '1' to this bit enables[17] the Transmit-FIFO-*Underflow Interrupt* (§5.3.4). Refer to Table 14 for a description of the interrupt status flags.

5.2.7. SyncIntEn – Synchronisation-Error Interrupt Enable

Writing a '1' to this bit enables[17] the *Synchronisation-Error Interrupt* (§5.3.5). Refer to Table 14 for a description of the interrupt status flags.

5.2.8. ShortIntEn – Short-Circuit Interrupt Enable

Writing a '1' to this bit enables[17] the *Short-Circuit-Detected Interrupt* (§5.3.6). Refer to Table 14 for a description of the interrupt status flags.

[17] To actually enable the Periodic Interrupt on the PCI Bus, the Local-Interrupt-Input-Enable bit in the PCI9054's Interrupt Control/Status register must also be set to '1'.

5.2.9. ThrIntEn – Threshold-Crossed Interrupt Enable

Writing a '1' to this bit enables[17] the *Threshold-Crossed Interrupt* (§5.3.7). Refer to Table 14 for a description of the interrupt status flags.

5.2.10. UseExtClk – Use External Clock

Writing a '0' to this field selects the *internal* clock generator as clock source for the outgoing Transport Stream.

Writing a '1' to this field selects the *external* clock input as clock source.

5.2.11. EnaPwr – Enable Power

Writing a '1' to this field closes the relay so that +5V power is applied to the **PSYNC** output. At the same time the **DVALID** output changes to **CODE**, which encodes both **DVALID** and **PSYNC** in one signal. Refer to §2.5 for a description.

5.2.12. ShortErr – Short-Circuit Error

This flag indicates whether the hardware currently detects a short-circuit condition on the LVDS outputs. Contrary to `ShortInt` in the Transmit-Status register, `ShortErr` is not latched and cannot be cleared by writing a '1'.

When `ShortErr` is '1', the LED on the PCI bracket of the DTA-102 will be flashing in red.

5.2.13. LedControl – Take over LED Control

When this field is '0', the state of the bi-colour LED indicator on the PCI bracket is determined by the hardware, as described in §2.3.

When this field is '1', the hardware is disconnected from the LED indicator. Instead, the LED is controlled directly by fields `LedGreen` and `LedRed`.

This bit is reset to '0' upon a hardware- or software reset.

5.2.14. LedGreen – State of Green LED

If `LedControl` is '1', this field controls the **green** colour of the bi-colour LED next to the connector on the PCI bracket.

5.2.15. LedRed – State of Red LED

If `LedControl` is '1', this field controls the **red** colour of the bi-colour LED next to the connector on the PCI bracket.

5.2.16. ClrFifo – Clear Transmit FIFO

Writing a '1' to this field[18] clears the contents of the Transmit FIFO and resets the FIFO-Load register to zero. Furthermore, `TxCtrl` in the Transmit-Control register is reset to **Idle**.

In all Transmit Modes except **Raw**, the Transmit FIFO can be cleared with inducing a truncated packet. The packet framing structure (distance between sync bytes) is not broken.

If the Transmit FIFO is transmitting data while the clear operation occurs, then the contents of the last packet sent may be corrupted. Nevertheless, the packet's length is preserved.

On the other hand, if no data is being transmitted (Transmit Control is **Idle** and/or Transmit FIFO empty), no corrupted data will be sent at all. Therefore, DEKTEC recommends the following procedure to clear the Transmit FIFO:

- Set Transmit Control to **Idle**.
- Wait a few packet periods so that the DTA-102 pipeline is flushed.
- Write '1' to `ClrFifo`.

5.3. Transmit Status (Offset = 08h)

The Transmit-Status register contains a number of fields that allow the device driver to read status information from the DTA-102.

Table 14. Transmit-Status Register – Format

Bit	Mnem	Description
0	`FifoFilled`	Transmit-FIFO Filled
3...1	*`reserved`*	
7...4	`SdramSize`	SDRAM Size
8	`PerInt`	Periodic Interrupt
9	`UflInt`	Transmit-FIFO Underflow Interrupt
10	`SyncInt`	Synchronisation Error
11	`ShortInt`	Short-Circuit Detected
12	`ThrInt`	Threshold-Crossed
Bits 31...13 of this register are tied to '0'.		

The interrupt-status flags (bit 8...12) in this register share common behaviour:

- An interrupt-status flag is set when the corresponding condition occurs. The flag remains set until it is explicitly cleared.
- Writing a '1' to the flag clears the interrupt-status flag, and also clears the PCI interrupt[19] (unless another interrupt condition is pending).
- The interrupt-status flag only leads to an interrupt if the corresponding interrupt-enable bit in register `TxControl` is set, and interrupts in the PCI 9054 have been enabled.
- The operation of the interrupt-status bits is independent from the state of the interrupt-enable bit: If the interrupt-enable bit is '0', the interrupt-status flag still latches the corresponding condition.

5.3.1. FifoFilled – Transmit-FIFO Filled

When set to '1', this read-only bit indicates that the DTA-102's Transmit FIFO is filled up to the level specified in the FIFO-Size register. The FIFO-Filled flag may be used in a controlled initialisation procedure to signal to the device driver that the Transmit FIFO is completely filled.

Notes

- The buffer-management examples provided in this specification do not use the FIFO-

[18] The `ClrFifo` bit is write-only. The write operation triggers the clear action, in a similar way to the Software-Reset bit (§5.1.3).

[19] No write action to a PCI-9054 register is required.

Filled flag.

- In DMA-driven operation (§3.4.1), the DTA-102 attempts to keep the Transmit FIFO filled by requesting DMA data whenever empty space is available in the Transmit FIFO. The FIFO-Filled flag will occasionally be set to '1'. This does not imply an error.

5.3.2. SdramSize – SDRAM Size

SDRAM Size is a static read-only field that indicates the size of the SDRAM on-board of the DTA-102. The SDRAM size determines the maximum size of the Transmit FIFO.

Table 15. SDRAM Size – Values

Value	Size	Comment
0000	8 MB	Minimum supported size.
0001	16 MB	May be supported in future revisions of the DTA-102.
0010	32 MB	May be supported in future revisions of the DTA-102.
other	*reserved*	

5.3.3. PerInt – Periodic Interrupt

When set to '1', this bit indicates that the *Periodic Interrupt* is pending. The Periodic Interrupt is generated automatically every 2^{21} clock cycles of the on-board 27 MHz reference clock. This corresponds to approximately once every 77.7 ms, or 12.87 times per second.

5.3.4. UflInt – Underflow Interrupt

When set to '1', this bit indicates that an underflow condition has occurred for the Transmit FIFO: The host could not supply enough Transport-Stream data to sustain a continuous output stream.

5.3.5. SyncInt – Synchronisation-Error Interrupt

When set to '1', this bit indicates that a synchronisation error has occurred in the packet-processing logic on the DTA-102. The distance between two MPEG-2 sync bytes (0x47) in the Transmit FIFO appeared to be not 188 (Packet Mode **188** or **Add16**), 192 (Packet Mode **192**) or 204 (Packet Mode **204** or **Min16**.)

The DTA-102 only checks for synchronisation errors if Transmit Control is **Send**. When a synchronisation error occurs, the DTA-102 will try to resynchronise to the data from the Transmit FIFO. In this process, a number of bytes will be skipped (not transmitted), until resynchronisation has been established again.

In Packet Mode **Raw**, no notion of packet size exists and therefore synchronisation-error checking is disabled.

5.3.6. ShortInt – Short-Circuit Interrupt

When set to '1', this bit indicates that a short-circuit condition has been detected on one of the LVDS outputs.

Contrary to `ShortErr` in the Transmit-Control register, `ShortErr` is latched and can only be cleared by writing a '1' to the bit position.

5.3.7. ThrInt – Threshold-Crossed Int.

When set to '1', this bit indicates that one of the threshold comparators has changed value. Thresholds are used in the target-adapter detection process, refer to §5.10.

5.4. Transmit Clock (Offset = 0Ch)

The Transmit-Clock register defines the frequency of the internal clock generator. If `UseExtClk` is '0', the internal clock generator determines the clock rate of the transmitted DVB/SPI signal.

Table 16. Transmit-Clock Register – Format

Bit	Mnem	Definition
31...0	`TxClock`	32 bits of the phase-increment value used to generate the transmit clock.

The relation between the Transmit-Clock register and the resulting transmit-clock frequency is defined as follows:

$$R_{Tx}(MHz) = \frac{27 * TxClock}{2^{32}}$$

Note
- The clock signal on the DVB/SPI output of the DTA-102 is a byte clock.

To compute the value of the Transmit-Clock register for a given desired transmit-clock frequency, the reverse formula shall be used:

$$\texttt{TxClock} = \frac{2^{32} * R_{Tx}(MHz)}{27}$$

The maximum transmit-clock frequency that can be handled properly by the DTA-102 is 13.5 MHz. This is equivalent to a `TxClock` value of 2,147,483,648 (80000000h). The behaviour of the board for frequencies above 13.5 MHz is unspecified.

Note
- The DTA-102 allows programming of bit rates higher than 13.5 MHz: No interrupt or other error-trap mechanism is invoked.

The transmit-clock frequency is not necessarily equal to the Transport-Stream bit rate (as defined in the MPEG-2 Systems specification) divided by 8. For 188-byte packets the two rates are equivalent, but for 204-byte packets, the Transport-Stream bit rate is 188/204 * Transmit-Clock bit rate. Table 17 below summarises the relationship between the different kinds of rates.

Table 17. Relation between Transmit-Clock rate and Transport-Stream rate.

Mode	TxRate	TSRate
188	$\text{Tx Rate} = \frac{\text{TS Rate}}{8}$	$\text{TS Rate} = \frac{\text{Tx Rate}}{8}$
192	$\text{Tx Rate} = \frac{192}{188} * \frac{\text{TS Rate}}{8}$	$\text{TS Rate} = \frac{1504}{192} * \text{Tx Rate}$
204 Add16 Min16	$\text{Tx Rate} = \frac{204}{188} * \frac{\text{TS Rate}}{8}$	$\text{TS Rate} = \frac{1504}{204} * \text{Tx Rate}$

5.5. Ext. Clock Count (Offset = 10h)

The External-Clock Count register can be used to read a sample from a free-running counter that is incremented at every positive edge of the external clock input.

Table 18. External-Clock Count – Format

Bit	Mnem	Definition
31...0	`ExtClkCnt`	Counter running at external clock.

The purpose of the External-Clock-Count register is to enable software measurement of the frequency of the signal applied to the external clock input.

Notes
- The external clock is a *byte* clock. The bit clock is a factor 8 higher;
- The external-clock counter operates directly on the external-clock input. It is irrelevant (for the External-Clock Count register) whether the internal clock generator is used.

5.6. FIFO Size (Offset = 14h)

The FIFO-Size register defines the maximum load of the DTA-102's Transmit FIFO when data is transferred using DMA. When the Transmit-FIFO load reaches the value in FIFO Size, the DTA-102 inhibits DMA requests.

The maximum load is expressed in number of bytes. The number of significant bits in the FIFO-Size register depends on the type of the SDRAM on-board of the DTA-102. Table 19 below is valid for an SDRAM size of 8 Mbytes.

Table 19. FIFO-Size Register – Format[20]

Bit	Mnem	Definition
23...0	`FifoSize`	Maximum FIFO load. May not be 0.
Bits 31...24 of this register are tied to '0'.		

The FIFO-Size register can be programmed to any value between 0 and 2^{24}-1 bytes[20], but not all values make sense:

- The absolute minimum meaningful value for FIFO Size is the number of bytes in a transport packet (188 or 204). Setting FIFO Size to a lower value may block demand-mode DMA entirely.
  The minimum *practical* value for FIFO Size

[20] Assuming an SDRAM-size of 8 MB. If the SDRAM is larger, more significant bits are included.

is about 500 bytes (refer to §7.3).

- The maximum meaningful FIFO-Size value is 2^{23} = 8.388.592 bytes[20]. Values between $2^{23}+1$ and 2^{24} are invalid: These values can be programmed in the FIFO-Size register without warning, but they would render FIFO-overflow protection inoperative.

The register name "FIFO Size" is somewhat misleading: the absolute size of the FIFO is always 8 Mbytes[20, 21]. In fact, the FIFO-Size register specifies a *virtual* FIFO size that is used for flow control of DMA transfers: When the FIFO load becomes greater or equal than FIFO Size, the FIFO-Filled flag is asserted and DMA transfers are temporarily disabled. DMA is re-enabled when the FIFO load drops below FIFO Size. In other words, DMA is *demanded* whenever the FIFO load is less than the FIFO size.

When the DTA-102 is operated under DMA, the Transmit-FIFO load can become slightly higher than FIFO Size. This is due to the fact that, when demand for DMA stops, a few words may be in the pipeline and still need to be stored in the Transmit FIFO. Even so it is safe to program the FIFO-Size register to precisely 2^{23} (= 8 Mbytes[20]), because the Transmit FIFO can accommodate for a few hundred additional bytes.

Note

- When data is written into the Transmit FIFO *directly* through the FIFO-Data register (without using DMA), the FIFO-Size register has no effect.

5.7. FIFO Load (Offset = 18h)

The FIFO-Load register contains the current load of the DTA-102's Transmit FIFO, expressed in number of bytes. Table 20 below is valid for an SDRAM size of 8 Mbytes.

Table 20. FIFO-Load Register – Format[20]		
Bit	**Mnem**	**Definition**
23...0	`FifoLoad`	Current FIFO load.
Bits 31...24 of this register are tied to '0'.		

While the DTA-102 is streaming data, the value read from this register is volatile. The value may change with every transmitted byte and with every DMA transfer.

Note

- The actual number of bytes buffered on the PCI card may be slightly higher than FIFO Load due to words residing in the PCI-9054 FIFO (not accounted for in FIFO Load) and/or in pipeline registers.

The absolute maximum value of FIFO Load on a DTA-102 is the SDRAM size plus 256 bytes. When FIFO Load has reached this value, the hardware blocks write operations to the Transmit FIFO. Write attempts to the FIFO-Data register will seem to succeed, but the data disappears and FIFO Load is not incremented.

The use of the FIFO-Load register in flow-control algorithms is optional. It can be used for enhancing robustness by checking at specific moments in time whether the FIFO Load is contained within a certain expected range.

5.8. Diagnostics (Offset = 1Ch)

The Diagnostics register contains a number of special fields that can be used for validation and testing of the DTA-102. In normal operation, this register should not be touched. It is recommended to clear the Diagnostics register to all zeros in the device-driver's initialisation routine[22].

[21] Plus a few hundred bytes (refer to §7).

[22] Issuing a soft reset through the General-Control register will also clear the Diagnostics register.

Table 21. Diagnostics Register – Format		
Bit	Mnem	Definition
7...0	SfData	Data at output of Smoothing FIFO
8	LoopBack	Loop-Back mode
9	SfDtValid	Smoothing-FIFO Data Valid
10	BfFull	Burst-FIFO Full flag
11	DmaReq	DMA Request
12	BfCanBurst	Burst-FIFO Can-Burst
13	SfCanBurst	Smoothing-FIFO Can-Burst
23...14	SfLoad	Smoothing-FIFO Load
31...24	BfLoad	Burst-FIFO Load

5.8.1. SfData – Smoothing-FIFO Data

Smoothing-FIFO Data is a read-only 8-bit field that represents the output data of the Smoothing-FIFO (§7.2.5). In normal operation, the value may change with every transmitted byte.

The Smoothing-FIFO-Data register is not suited for a data-path or memory test, because reading this register will only take a snapshot of the value at the FIFO's output. The data byte is not popped from the Smoothing FIFO.

Note

- The Loop-Back-Data register can be used for simultaneous reading and popping of data bytes from the Smoothing FIFO.

5.8.2. LoopBack – Loop-Back Mode

Writing a '1' to this bit disconnects the DVB/SPI output circuitry from the Smoothing FIFO. This enables diagnostic software to read back bytes at the end of the FIFO pipeline through the Loop-Back Data register.

In normal operation, Loop-Back Mode should be set to '0'. Loop-Back Mode can be set to '1' for a manufacturing-test set-up in which data-path and memory integrity must be tested.

Loop-Back Mode field is not cleared to '0' by a software reset.

5.8.3. SfDtValid – Smoothing-FIFO Data Valid

Diagnostic read-only status flag that indicates whether the Smoothing FIFO has valid data at its output. This status flag has no purpose in normal operation.

5.8.4. BfFull – Burst-FIFO Full

Diagnostic read-only bit that is set if the Burst FIFO (§7.2.2) is full. The DTA-102 will only assert BfFull in exceptional circumstances. This status flag has no purpose in normal operation.

5.8.5. DmaReq – DMA Request

Diagnostic read-only bit that is set if the DTA-102's internal play-out logic is requesting a DMA transfer. The request is relayed to the PCI bus only if DMA is also enabled in the PCI-9054. This status flag has no purpose in normal operation.

5.8.6. BfCanBurst – Burst FIFO Can Burst

Diagnostic read-only bit that is set if the Burst FIFO (§7.2.2) can produce a burst of 16 bytes to write in the SDRAM (for efficiency reasons, the DTA-102 writes bursts of 16 bytes to the SDRAM). This status flag has no purpose in normal operation.

5.8.7. SfCanBurst

Diagnostic read-only bit that is set if the Smoothing FIFO (§7.2.5) can produce a burst of data. This status flag has no purpose in normal operation.

5.8.8. SfLoad – Smoothing-FIFO Load

Smoothing-FIFO Load is a read-only 10-bit field that indicates the number of 16-bit words in the Smoothing FIFO (§7.2.5). This status field has no purpose in normal operation.

5.8.9. BfLoad – Burst-Fifo Load

Burst-FIFO Load is a read-only 8-bit field that indicates the number of 32-bit words in the Burst FIFO (§7.2.2). This status field has no purpose in normal operation.

5.9. Loop-Back Data (Offset = 20h)

The Loop-Back-Data register is an extension to the Diagnostics register. SfDataNxt[7..0] in the Loop-Back-Data register holds the same data as SfData[7...0] in the Diagnostics register. However, reading the Loop-Back-Data register (SfDataNxt) will also cause a data byte to be popped from the Smoothing FIFO.

Table 22. Loop-Back Data Register – Format		
Bit	**Mnem**	**Definition**
7...0	SfDataNxt	Read data at output of Smoothing FIFO *and issue a read pulse*
Bits 31...8 of this register are tied to '0'.		

The purpose of this register is to enable diagnostics data-path-integrity and memory-test software.

Note

- Loop-Back Mode must be '1' for meaningful use of Loop-Back Data.

5.10. Threshold Control (Offset = 24h)

The Threshold-Control register contains a number of control- and status- fields to manage *target-adapter detection*.

Table 23. Threshold-Control Register – Format		
Bit	**Mnem**	**Definition**
0	UpdDone	Update of threshold done
1	CmpB	Output of comparator B
2	CmpA	Output of comparator A
3	An/B	Update A ('0') or B ('1')
7...4	*reserved*	Not used
15...8	ThreshA	Threshold A
23...15	ThreshB	Threshold B
Bits 31...24 of this register are tied to '0'.		

Each target adapter is identified by a specific resistor located in the target adapter, as described in §2.6. The DTA-102 hardware, with the help of software, measures the value of the external resistor, so that the target-adapter type can be determined automatically.

Three special cases can be detected:

- If the resistor is temporarily shorted, the target adapter signals an error condition to the DTA-102. Target adapters may optionally implement this feature with an NPN transistor parallel to the resistor.
- If the measured resistor value is close to 0Ω right from the start, then the DTA-102 is connected to a standard DVB/SPI sink.
- If the measured resistor value is ∞ (infinite), nothing is connected to the DTA-102.

A schematic diagram of the target-adapter detection circuitry, and the fields in the Threshold-Control register, are shown in Figure 17.

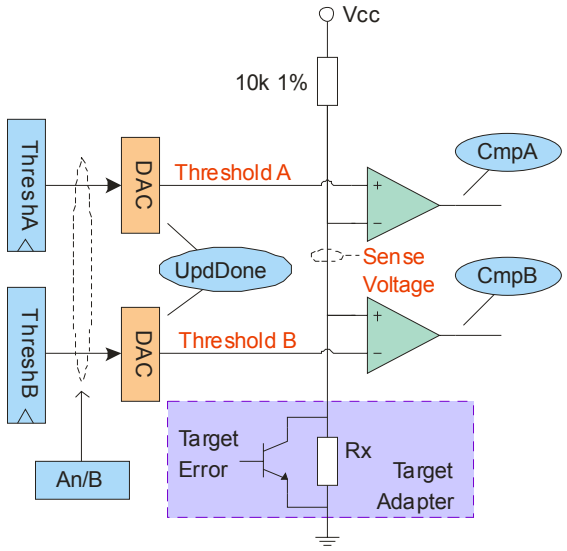


Figure 17. Target-adapter detection. Text on light-blue background refers to fields in the Threshold-Control register.

The target-adapter resistor is converted to the *sense voltage* using a resistor divider with a 10k 1% resistor to +5 Volts. The sense voltage is compared to two *thresholds*, which are set by two D/A Converters: Threshold A and Threshold B. The result of the comparison can be read through status flags CmpA and CmpB. The voltage can be determined by executing a binary search with one of the DACs and observing the comparator's output.

Once the resistor value is known, the state of the target adapter can be monitored by setting a *window* around the voltage induced over the target-adapter resistor. One DAC is set to the lower end of the window, the other to the higher end. By observing the comparator outputs, the software can be alerted to two events:

- The high end of the window is crossed. This event indicates that the cable to the target adapter is disconnected.
- The low end of the window is crossed. This event indicates that the target adapter is signalling an error.

5.10.1. UpdDone – Update Done

This status flag indicates that the value in one of the DACs (selected by the `An/B` field) has been updated. Waiting for this status flag is necessary because – after writing to the Threshold Register – it takes some time to transmit the value from the Threshold Register to the DAC.

After Update-Done has become '1', it is a requirement for the driver to wait another 10 µs before reading the comparator output. The DAC and comparator need this period to settle to a stable value.

5.10.2. CmpA – Output Comparator A

This status flag holds the result of comparing the output of D/A Converter A and the sense voltage. If `CmpA` is '1', the sense voltage is lower than the DAC voltage.

5.10.3. CmpB – Output Comparator B

This status flag holds the result of comparing the output of D/A Converter B and the sense voltage. If `CmpB` is '1', the sense voltage is higher than the DAC voltage.

5.10.4. An/B – Update Threshold A or B

The `An/B` field determines whether D/A Converter A (`An/B` is '0') or D/A Converter B (`An/b` is '1') is updated.

Due to hardware limitations, the two D/A Converters cannot be updated in one go. To write both values, follow these steps:

1. Write the value for DAC A in field `ThreshA` while, in the same write operation, `An/B` is '0'.
2. Wait until field `UpdDone` is '1'.
3. Write the value for DAC B in field `ThreshB` while, in the same write operation, `An/B` is '1'.
4. Wait until field `UpdDone` is '1'.
5. Wait 10 µs to let the DACs and comparators settle.

5.10.5. ThreshA – Threshold A

Value for the D/A Converter that generates Threshold A.

The value in this field is transferred to D/A Converter A when a value is written to the Threshold-Control register with `An/B` is '0'.

5.10.6. ThreshB – Threshold B

Value for the D/A Converter that generates Threshold B.

The value in this field is transferred to D/A Converter B when a value is written to the Threshold-Control register with `An/B` is '1'.

5.11. FIFO Data (Offset = 40h...7Ch)

The FIFO-Data register is the input register of the Transmit FIFO. It's the main register for streaming data through the DTA-102.

Data written to the FIFO-Data register directly enters the Transmit FIFO, four bytes at a time. Byte 0 (bit 7…0) will be the first transmitted byte, byte 3 (bit 3…24) the last one.

Table 24. FIFO-Data Register – Format

Bit	Mnem	Definition
7...0	`FifoData`	First transmitted byte
15...8	`FifoData`	Second byte
23...16	`FifoData`	Third byte
31...24	`FifoData`	Last transmitted byte

The suggested way to operate the DTA-102 is to program (in the PCI-9054) DMA transfers

with the FIFO-Data register as constant destination address[23].

Instead of using DMA, an application program may also choose to write directly to the FIFO-Data register. In certain circumstances this may be a simple, convenient way to write data, e.g. for diagnostic purposes. However, performance will suffer, mainly because the host processor is required to write each individual word. The recommended way to operate the DTA-102 is using DMA transfers.

Notes

- The FIFO-Data register appears at 16 consecutive word addresses in the Operational-Registers Memory Map. This enables write bursts on the PCI bus, so that applications that choose not to use DMA can still achieve reasonable performance.
- When the DTA-102 is operated under DMA, a mechanism is in effect to protect the Transmit FIFO against overflow (§5.5, §7). When data is written directly to the FIFO, no such mechanism exists. Direct write operations always succeed, but when the Transmit FIFO contains SDRAM size + 256 bytes, new data just vanishes without warning.

[23] Local Address in the PCI-9054's gather/scatter DMA descriptor should be set to 00000040h and Local Addressing Mode to '1' (hold Local Address bus constant).

6. Vital Product Data

Vital Product Data (VPD) is information stored in a PCI device to uniquely identify the hardware and, potentially, software elements of the device. *PCI Local Bus Specification Rev2.2* defines both a standard storage structure and access method for VPD.

The DTA-102 uses VPD to store the serial number, revision level, etc. The sections below list all supported fields. The VPD is stored in the serial EEPROM on-board of the DTA-102. The VPD can be accessed through the VPD-function support built in the PCI-9054.

The DEKTEC DTA-series of PCI cards share the same layout of the serial EEPROM, so that the VPD data can be accessed in a uniform way for each board.

6.1. Serial EEPROM Lay-Out

Figure 18 below shows the memory map of the serial EEPROM and the positioning of VPD elements within the EEPROM memory. Note that addresses are *byte* addresses, whereas the PCI-9054 specification sometimes uses word (16-bit) or long word (32-bit) addresses.

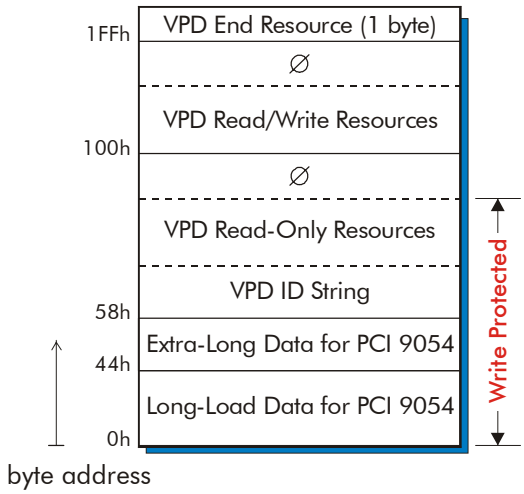


Figure 18. Memory Map of the Serial EEPROM.

The 88 bytes at address 0h…57h are loaded into the PCI-9054 upon power-on reset, to define the initial value of the register set. This part of the EEPROM data is not related to VPD.

The other 424 bytes in the serial EEPROM, starting at address 58h, are dedicated to VPD. First, the VPD Identification String and the VPD Read-Only Resources are stored sequentially, followed by zero-byte stuffing. The VPD Read/Write Resources are located at address 100h, also followed by zero-byte stuffing. The EEPROM is "closed" by a VPD End resource at address 1FFh.

The VPD Read-Only and Read/Write Resources each consist of a two-character key (e.g. "PN" for Part Number) and a length-prefixed string defining the actual resource.

The first part of the serial EEPROM (register-data for PCI-9054, VPD ID String and VPD Read-Only Resources) is programmed in the factory and write-protected to avoid accidental modification. The VPD Read/Write Resources are not write-protected and may be modified by the end user, e.g. for storing the customer's system-asset identifier. DEKTEC utilises VPD Read/Write Resources to store software licenses.

6.2. VPD ID String

The VPD ID-String Resource contains the name of the board in ASCII characters. The content of this resource is fixed for all incarnations of the DTA-102.

Table 25. VPD ID String – Syntax

Addr[24]	Value	Item
58h	82h	ID-String tag.
59h	22h	Length LSB.
5Ah	00h	Length MSB.
5Bh	"DTA-102 DVB/SPI Output 0..108 Mbps"	ID-String data[25].

6.3. VPD Read-Only Resources

The VPD Read-Only Resources describe permanent hardware characteristics of the PCI card. This VPD section is stored in the EEPROM just behind the VPD ID-String section.

[24] Byte address in serial EEPROM.
[25] No trailing zero ('\0') character!

The following VPD read-only resources are supported on the DTA-102:

- PN = Part Number
  The PN Resource is fixed to "DTA-102".
- EC = Engineering Change Level
  The EC Resource identifies the hardware revision of the board, e.g. "Rev 2".
- MN = Manufacture ID
  The MN Resource is a 2-digit code[26] identifying the manufacturer of the board.
- CL = Customer ID
  The CL Resource is a 6-digit code[26] identifying the (initial) customer of the board.
- SN = Serial Number
  The SN Resource holds a unique serial number. For the DTA-102, this number begins with "4102", followed by a sequence number in 6 or more digits.
- GC = Guard Code
  The GC Resource is a coded string used in DEKTEC license management.
- PD = Production Date
  The PD Resource keeps the production date of this DTA-102 instance, e.g. "2001.07".
- XT = Crystal Accuracy
  The XT Resource lists the accuracy of the 27-MHz crystal oscillator as a string, e.g. "100ppm".

Table 26 below shows an example of the contents of the VPD Read-Only Resources section. DEKTEC reserves the right to append private descriptors in the reserved part of the read-only area.

Table 26. VPD Read-Only Resources – Syntax

Addr	Value	Item
07Dh	90h	VPD-R tag.
07Eh	7Eh	Length LSB.
07Fh	00h	Length MSB.
080h	"PN"	VPD keyword.
082h	7	Field length.
083h	"DTA-102"	Part number.

[26] DEKTEC internal code.

Table 26. VPD Read-Only Resources – Syntax

Addr	Value	Item
08Ah	"EC"	VPD keyword.
08Ch	5	Field length.
08Dh	"Rev 1"	Engineering-Change level.
092h	"MN"	VPD keyword.
094h	2	Field length.
095h	"01"	Manufacture ID.
097h	"SN"	VPD keyword.
099h	10	Field length.
09Ah	"4102000007"	Serial number.
0A4h	"CL"	VPD keyword.
0A6h	6	Field length.
0A7h	"300000"	Customer ID
0ADh	"GC"	VPD keyword.
0AFh	10	Field length.
0B0h	"/06(TP0*=0"	Guard Code
0BAh	"PD"	VPD keyword.
0BCh	7	Field length.
0BDh	"2001.06"	Production Date.
0C4h	"XT"	VPD keyword.
0C6h	2	Field length.
0C7h	"5ppm@25C;15 ppm"	Crystal Accuracy.
0D5h	"RV"	VPD keyword.
0D7h	28h	Field length.
0D8h	ABh	Checksum.
0D9h	39 x 00h	Reserved.
100h		Read-Write section

The length of the VPD Read-Only Resources section is tuned such that the VPD Read/Write Resources section starts at byte address 100h.

6.4. VPD Read-Write Resources

The VPD read/write section can hold 255 data bytes that can be updated dynamically from software. Potential usage includes diverse applications such as software keying, system-asset identification and storage of fault codes for inspection by service personnel.

Every byte in the serial EEPROM can be rewritten about 10^6 times. Therefore, the VPD Read/Write Resource cannot be used for data that is updated a lot, e.g. every second. It is recommended to use the Read/Write section only for data that has a near-static nature.

The following standard tags are defined in *PCI Local Bus Specification Rev2.2*.

- **Vx = Vendor Specific**
  This is a DEKTEC-specific item, e.g. a software license. The second character (x) of the keyword can be 0 through 9 and A through Z.
- **Yx = System Specific**
  This is a system-specific item. The second character (x) of the keyword can be 0 through 9 and B through Z.
- **YA = Asset Tag Identifier**
  The resource contains the system-asset identifier provided by the system owner.
- **RW = Remaining Read/Write Area**
  This descriptor is used to identify the unused portion of the read/write space.

The data bytes are stored in the serial EEPROM at address 100h up to 1FEh inclusive. The byte at address 1FFh is used to store the VPD-End tag. Table 27 below shows an example of the syntax of the VPD-Read/Write-Resources section.

Table 27. VPD Read/Write Resources – Syntax

Addr	Value	Item
100h	91h	VPD-W tag.
101h	FCh	Length LSB.
102h	00h	Length MSB.
103h	"V3"	VPD keyword.
105h	16	Field length.
106h	"3rS=;2kl`MD(#ac"	License string.
116h	"YA"	VPD keyword.
118h	25	Field length.
119h	"DVB/SPI Test Generator 14"	System-asset identifier.

Table 27. VPD Read/Write Resources – Syntax

Addr	Value	Item
132h	"RW"	VPD keyword.
134h	202	Field length.
135h	202 x 00h	Reserved.
1FFh	78h	VPD End tag.

6.5. Reading VPD Data

VPD Resources can be read 4 bytes at a time with the procedure described below. The hardware does not support any form of parsing VPD data, this is the job of the device driver.

1. Ensure that the read address is 32-bit aligned: the two least-significant address bits shall be zero.
2. Write the address to the 16-bit PCI-9054 register **PVPDAD** at PCI offset 4Eh in PCI-*Configuration* Space.
   Set the **VPD-Address** field to the read-address field (last two bits 0) and, in the same operation, set the **F**-flag field to '0', signalling a read operation.
3. Poll the **F**-flag in a loop until it becomes '1'. This indicates that the VPD read data is actually available.
4. Read the 32-bit PCI-9054 register **VPDDATA** at PCI offset 50h to obtain the requested 4 VPD data bytes.
5. Repeat steps 1..4 for all VPD words to be read.

6.6. Writing VPD Data

The VPD Read/Write Resources section can be written 4 bytes at a time with the procedure described below.

1. Ensure that the write address is 32-bit aligned: the two least-significant bits shall be zero.
2. Enable programming of the serial EEPROM by writing a '1' to PE in the General Control register (§5.1.1).

3. Change the *Serial EEPROM Write-Protected Address Boundary* register in the PCI 9054 (register PROT_AREA at PCI-offset 0Eh[27]) to a value less or equal than the write address divided by four.
   The division by four is required because PROT_AREA contains a 7-bit field that points to a 32-bit "long-word" address.
4. Write the desired data (32-bits!) to PCI-9054 register **VPDDATA.**
5. Write the destination address to the 16-bit PCI-9054 register **PVPDAD** at PCI offset 4Eh in PCI-*Configuration* Space.
   Set the **VPD-Address** field to the write address (last two bits 0) and, in the same operation, set the **F**-flag to '1', which signals a write operation.
6. Poll the **F**-flag until it changes to '0' to ensure that the write operation has completed.
7. Repeat steps 1..4 for all VPD words to be written.
8. For safety, change **PROT_AREA** back to 7Fh, and:
9. Disable programming of the serial EEPROM by writing a '0' to **PE** in the General Control register.

Note

- It is the responsibility of the device driver to maintain integrity of the VPD Resources.
  For example, if a VPD Resource to be rewritten does not start at a 32-bit boundary, then the host should first read the original 32-bit VPD word, AND/OR-in the new data, and write the resulting 32-bit word back.

[27] In PCI-memory space.

7. Transmit FIFO

Transport-Stream data enters the DTA-102 burst by burst. The Transmit FIFO temporarily buffers the data bytes, so that a smooth DVB/SPI stream can be generated. This section provides a number of details about the implementation of the Transmit FIFO.

This is an advanced section that only needs to be studied by developers that seek to achieve the maximum from the DTA-102.

7.1. Coarse-Grain Model

For most applications, the Transmit FIFO on the DTA-102 can be regarded as a conventional First-In First-Out buffer with a programmable maximum load, see Figure 19. This approximation of the Transmit FIFO is entirely adequate as long as the average FIFO Load remains well above a kilobyte.

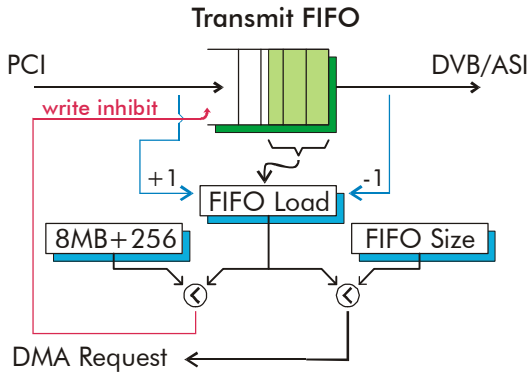


Figure 19. Coarse-grain model of the Transmit FIFO on the DTA-102.

The FIFO-Load register maintains the number of bytes loaded in the FIFO. Every time a data byte is written into the Transmit FIFO, the load is incremented. Whenever a byte is read for transmission in the DVB/SPI output stream, the load is decremented.

FIFO-overflow protection is implemented by continuously comparing the FIFO Load to the FIFO-Size register. As long as FIFO Load is less than FIFO Size, DMA transfers are requested. Otherwise, the Transmit FIFO is considered "full" and DMA is inhibited until FIFO Load drops below FIFO Size again.

The FIFO Load is also compared to the SDRAM Size + 256 bytes. If the FIFO Load is greater or equal than this load, new writes to the Transmit FIFO are inhibited, and FIFO Load is not incremented any more.

7.2. Fine-Grain Model

Some special applications may require a modest average FIFO load (say less than a kilobyte), e.g. when the transport rate is relatively low and/or the end-to-end delay has to be as low as possible. For these applications, knowledge of the intricacies of the Transmit FIFO may be useful for improving the performance of the system incorporating the DTA-102. The sections below provide a second-order model of the Transmit FIFO.

7.2.1. Block Diagram

Figure 20 shows a fine-grain block diagram of the Transmit FIFO.

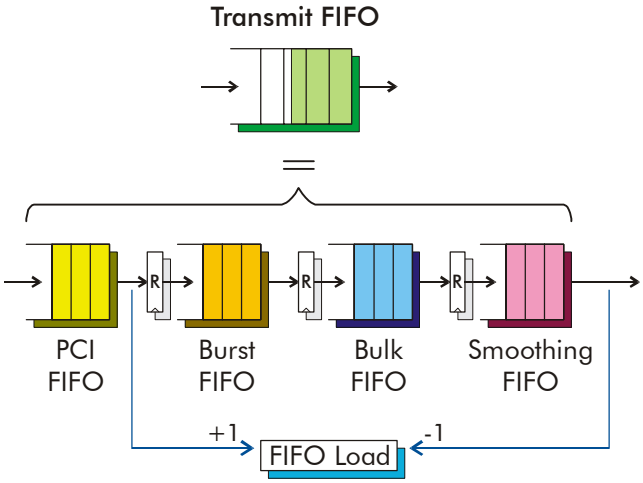


Figure 20. The Transmit FIFO can be modelled as a cascade of four FIFOs.

The Transmit FIFO can be considered as the concatenation of four FIFOs and a number of pipeline registers.

- First in line is the *PCI FIFO*, a small FIFO that buffers DMA bursts coming in from the PCI-Bus.
- The *Burst FIFO* is an extension of the PCI FIFO, to further enhance efficiency of DMA bursts on the PCI-Bus.
- The *Bulk FIFO* provides the major part of the buffer capacity, using an SDRAM chip.

- The *Smoothing FIFO* converts bursts from the SDRAM to a low-jitter DVB/SPI stream.

Why are four FIFOs required to implement a single Transmit FIFO? The core reason is the use of an SDRAM chip for the Bulk FIFO. The SDRAM does not support parallel reads and writes. Furthermore, the SDRAM can be operated efficiently only when read- and write operations are bundled in bursts. So, both at the input and the output-side of the SDRAM extra FIFOs are required to temporarily buffer data.

7.2.2. PCI FIFO

The *PCI FIFO* is a 32-word FIFO (128 bytes) embedded in the PCI-9054 bridge chip. This FIFO ensures that the DTA-102 can accept a 32-bit word at each PCI clock cycle, so that DMA transfers on the PCI Bus can occur in bursts.

Note

- For efficient utilisation of the PCI bus it is essential that DMA transfers occur in bursts. Every interruption of a burst incurs a penalty of many PCI cycles and a (potentially huge) degradation in net transfer speed.

The PCI FIFO cannot overflow: data words received in the PCI FIFO are always written to the Burst FIFO within a few clock cycles.

Note

- While passing data from PCI- to Burst FIFO, the DTA-102 does not check for overflow of the Burst FIFO.

7.2.3. Burst FIFO

The *Burst FIFO* extends the PCI FIFO with 256 words (1024 bytes), further increasing the length of DMA-transfer bursts. The Burst FIFO is implemented in the FPGA (Altera) on-board of the DTA-102.

The number of bytes loaded in the Burst FIFO controls whether DMA transfers are requested. DMA requests start when the Burst-FIFO load is less than 64 bytes[28]. The DTA-102 continues to request DMA until the Burst FIFO is half full (512 bytes). If this high-water mark is reached, the DTA-102 stops DMA until the load drops below 64 bytes again. At that moment DMA is re-enabled, unless the Transmit FIFO is full.

The DMA-Request signal can be observed through the DMA-Request field (`DmaReq`) in the Diagnostics register. The number of 32-bit words (not bytes) in the Burst FIFO can be observed in the Burst-FIFO-Load field (`BfLoad`) in the same register.

7.2.4. Bulk FIFO

The *Bulk FIFO* is the real meat of the Transmit FIFO. It provides the buffer capacity for compensating large PCI-, interrupt- and software scheduling latencies. The Bulk FIFO is implemented with an 8-Mbyte 16-bit wide SDRAM.

The SDRAM alternates read bursts, write bursts and refresh cycles. The hardware supports a maximum length of read- or write- bursts of 64 cycles (128 bytes).

Data is written to the SDRAM when (1) the Burst FIFO contains data, and (2) the maximum write-burst length has not been reached.

Data is read from the SDRAM as long as (1) the SDRAM contains data, (2) the Smoothing FIFO is not full and (3) the maximum read-burst length has not been reached.

7.2.5. Smoothing FIFO

The *Smoothing FIFO* is a 512-byte FIFO that smoothens data from the Bulk FIFO to a jitter-free DVB/SPI stream. This FIFO is required because the Bulk FIFO produces – by the nature of SDRAMs – bursts at its output.

The DTA-102 writes data from Bulk FIFO to Smoothing FIFO as long as if (1) the Bulk FIFO can supply data and (2) the Smoothing FIFO is not full.

If not in **Raw** mode, the DTA-102 reads data from the Smoothing FIFO when (1) the DVB/SPI output stage demands data to transmit another packet and (2) the Smoothing FIFO contains at least one packet. An Transmit-FIFO-Underflow condition occurs when it's time to send a packet, but the Smoothing FIFO contains less than one packet. In this case and when Null-Packet Stuffing is enabled (§5.2.2),

[28] And the load of the entire Transmit FIFO is less than FIFO Size.

the DTA-102 output stage will insert null packets.

In **Raw** mode, the DTA-102 does not check for the presence of at least one packet in the Smoothing FIFO. Whenever the DVB/SPI output stage requires another data byte (given the current transmit rate) and the Smoothing FIFO is not empty, a data byte is inserted in the output stream. If the Smoothing FIFO is empty (Transmit-FIFO Underflow), nothing is sent and the output bit rate will become lower than the rate specified in the Transmit-Clock register.

7.3. Real-Time Streaming

This section considers the question: In a real-time streaming application, given the structure of the Transmit FIFO described above, what's the minimum propagation delay incurred by the DTA-102?

The analysis assumes:

- Transmit Mode **188** (188-byte packets);
- A requirement for real-time operation in which null-packet stuffing is not acceptable, e.g. because it breaks MPEG-2 time stamping. This means that Transmit-FIFO Underflow may not occur.
- The output delay is the time passing between a transport packet being available in host memory and the packet appearing in the DVB/SPI output stream[29].
- To make the analysis more concrete, a transport rate of 40Mbps is assumed. The time to transmit one packet is approximately 38μs.

7.3.1. Absolute Minimum Delay

The absolute minimum output delay of the DTA-102 has three components.

- The time between starting a DMA transfer and the first data byte arriving in the Smoothing FIFO. If the PCI bus is not used for other purposes, this latency will be below 10μs.
- The Smoothing FIFO shall contain at least one packet (188 bytes) at all times, or an Transmit-FIFO Underflow may occur. The corresponding delay is 38μs.
- The propagation delay of the DVB/SPI output stage. This delay is below 1 μs and can safely be ignored.

So, the absolute minimum output delay with which the DTA-102 can be operated is about 50μs@40Mbps.

7.3.2. Practical Minimum Delay

In practice, the theoretical minimum delay cannot be met reliably. The main problem, which has nothing to do with the DTA-102, is the initiation of DMA transfers by the host. This issue has been discussed extensively in §3.2.

With respect to the Transmit FIFO the following remark can be made. For consistent, reliable operation the Smoothing FIFO should contain significantly more data than the absolute minimum of 1 packet. DEKTEC recommends trying to keep the Smoothing FIFO full, this is about 500 bytes. The corresponding minimum delay is 100μs.

In most applications, 100μs will be very small compared to the delay required for a reliable buffer-management and synchronisation model. So, in all but some very special applications, the fine-grain model of the Transmit FIFO is irrelevant.

[29] To compute the application's *end-to-end* delay, a similar analysis must be made for the input- and processing stage of the application.