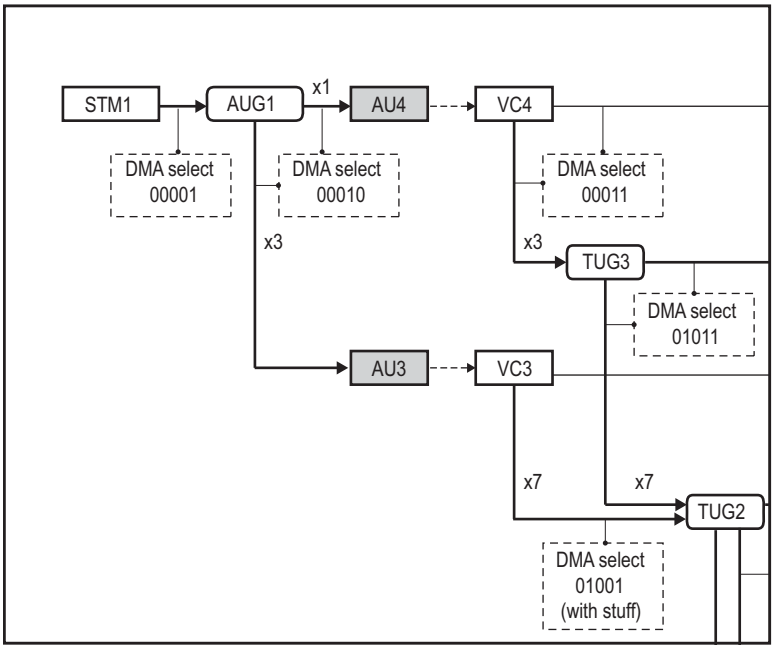
User's Guide Addendum

DE1

Configuration Package



Demultiplexer to E1
for use with specified EDT boards

Date: 2015 June 24
Rev.: 0006

Contact

Sky Blue Microsystems GmbH
Geisenhausenerstr. 18
81379 Munich, Germany
+49 89 780 2970, info@skyblue.de
www.skyblue.de

In Great Britain:
Zerif Technologies Ltd.
H5 Ash Tree Court
Nottingham NG8 6PY, England
+44 115 855 7883, info@zerif.co.uk
www.zerif.co.uk

Contact

Sky Blue Microsystems GmbH
Geisenhausenerstr. 18
81379 Munich, Germany
+49 89 780 2970, info@skyblue.de
www.skyblue.de

In Great Britain:
Zerif Technologies Ltd.
H5 Ash Tree Court
Nottingham NG8 6PY, England
+44 115 855 7883, info@zerif.co.uk
www.zerif.co.uk

Terms of Use Agreement

Definitions. This agreement, between Engineering Design Team, Inc. (“Seller”) and the user or distributor (“Buyer”), covers the use and distribution of the following items provided by Seller: a) the binary and all provided source code for any and all device drivers, software libraries, utilities, and example applications (collectively, “Software”); b) the binary and all provided source code for any and all configurable or programmable devices (collectively, “Firmware”); and c) the computer boards and all other physical components (collectively, “Hardware”). Software, Firmware, and Hardware are collectively referred to as “Products.” This agreement also covers Seller’s published Limited Warranty (“Warranty”) and all other published manuals and product information in physical, electronic, or any other form (“Documentation”).

License. Seller grants Buyer the right to use or distribute Seller’s Software and Firmware Products solely to enable Seller’s Hardware Products. Seller’s Software and Firmware must be used on the same computer as Seller’s Hardware. Seller’s Products and Documentation are furnished under, and may be used only in accordance with, the terms of this agreement. By using or distributing Seller’s Products and Documentation, Buyer agrees to the terms of this agreement, as well as any additional agreements (such as a nondisclosure agreement) between Buyer and Seller.

Export Restrictions. Buyer will not permit Seller’s Software, Firmware, or Hardware to be sent to, or used in, any other country except in compliance with applicable U.S. laws and regulations. For clarification or advice on such laws and regulations, Buyer should contact: **U.S. Department of Commerce, Export Division, Washington, D.C., U.S.A.**

Limitation of Rights. Seller grants Buyer a royalty-free right to modify, reproduce, and distribute executable files using the Seller’s Software and Firmware, provided that: a) the source code and executable files will be used only with Seller’s Hardware; b) Buyer agrees to indemnify, hold harmless, and defend Seller from and against any claims or lawsuits, including attorneys’ fees, that arise or result from the use or distribution of Buyer’s products containing Seller’s Products. Seller’s Hardware may not be copied or recreated in any form or by any means without Seller’s express written consent.

No Liability for Consequential Damages. In no event will Seller, its directors, officers, employees, or agents be liable to Buyer for any consequential, incidental, or indirect damages (including damages for business interruptions, loss of business profits or information, and the like) arising out of the use or inability to use the Products, even if Seller has been advised of the possibility of such damages. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to Buyer. Seller’s liability to Buyer for actual damages for any cause whatsoever, and regardless of the form of the action (whether in contract, product liability, tort including negligence, or otherwise), will be limited to fifty U.S. dollars ($50.00).

Limited Hardware Warranty. Seller warrants that the Hardware it manufactures and sells shall be free of defects in materials and workmanship for a period of 12 months from date of shipment to initial Buyer. This warranty does not apply to any product that is misused, abused, repaired, or otherwise modified by Buyer or others. Seller’s sole obligation for breach of this warranty shall be to repair or replace (F.O.B. Seller’s plant, Beaverton, Oregon, U.S.A.) any goods that are found to be non-conforming or defective as specified by Buyer within 30 days of discovery of any defect. Buyer shall bear all installation and transportation expenses, and all other incidental expenses and damages.

Limitation of Liability. *In no event shall Seller be liable for any type of special consequential, incidental, or penal damages, whether such damages arise from, or are a result of, breach of contract, warranty, tort (including negligence), strict liability, or otherwise.* All references to damages herein shall include, but not be limited to: loss of profit or revenue; loss of use of the goods or associated equipment; costs of substitute goods, equipment, or facilities; downtime costs; or claims for damages. Seller shall not be liable for any loss, claim, expense, or damage caused by, contributed to, or arising out of the acts or omissions of Buyer, whether negligent or otherwise.

No Other Warranties. Seller makes no other warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding Seller’s Products or Documentation. Seller does not warrant, guarantee, or make any representations regarding the use or the results of the use of the Products or Documentation or their correctness, accuracy, reliability, currentness, or otherwise. All risk related to the results and performance of the Products and Documentation is assumed by Buyer. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to Buyer.

Disclaimer. Seller’s Products and Documentation, including this document, are subject to change without notice. Documentation does not represent a commitment from Seller.

Contents

DE1 Configuration Package

Overview

DE1 is an EDT configuration package that demultiplexes STM1 inputs down to E1 signals. The package does this through sixteen STM1 demultiplexers (DXs), a crosspoint switch, and overhead information in the STM1 signals and their subsignals. (The structure and relationships of these signals and subsignals are described throughout the remainder of this user's guide.)

The DE1 package is designed to work with an EDT board pair. This board pair includes one main board (PCIe8 LX / FX) for DMA, plus one EDT mezzanine board (either the OCMP or the OC192) for I/O.

Related Resources

The table below shows EDT products that are compatible with the DE1 package, as well as other resources that may be necessary or helpful for your applications.

Compatible EDT products		
• OCMP Mezzanine Board	Datasheet & user's guide	www.edt.com/pciss_gs_ocm.html
• OC192 Mezzanine Board	Datasheet & user's guide	www.edt.com/pcigs_oc192.html
• PCI / PCIe Main Board (PCIe8 LX / FX only)	Datasheet & user's guide	www.edt.com/main_boards.html
Other Resources	*Detail*	*Web link*
• G.707 specification	From ITU	www.itu.int
• Mezzanine board documentation	Datasheets & user's guides	www.edt.com (find specific board)
• PCI / PCIe Main Board documentation	Datasheet & user's guide	www.edt.com/main_boards.html
• Installation packages	Software &firmware	www.edt.com/software.html
• Application Programming Interface (API)	HTML & PDF versions	www.edt.com/manuals.html

The API includes:

— Board initialization to load FPGA configuration files for both boards (main and mezzanine), and to return a handle for use with the `lib_sdh` API functions.
— DMA channel setup, with user function callback registration, to process demultiplexed E1 packets.
— Access to the STM1 demultiplexing register set via the `lib_sdh` handle.

Included Files

The DE1 Configuration Package ships with the following files.

`init_oc192_de1.c`	Initialization file that enables the OC192 to work with main board.
`oc192_de1.bit`	For OC192 – VHDL configuration file for the UI FPGA on main board.
`ocm_de1.bit`	For OCM – VHDL configuration file for the UI FPGA on main board.
`pcd_config/stmX_de1.cfg`	

Initialization script to be used with `initpcd` command. The *x* will be 1, 4, or 16 (respectively) for STM1, STM4, or STM16 input signals.

`stm1e1prbs15.c` Example application to check E1 prbs15 data.

`sdhgui.src` GUI for monitoring the demultiplexing process of the mezzanine board.

`lib_sdh.c` and `lib_sdh.h`

EDT source files for the `lib_sdh` API.

DE1 Crosspoint Switch

This section explains how the crosspoint switch works with your EDT board pair. Each of the sixteen STM1 demultiplexers (DXs) is accessed through its own set of indirect registers. For each DX, you can select any STM1 input available from the OCM or OC192 mezzanine board (see notes on Figure 1and Figure 2).

Write, into the register 0x00D086 STM1 Source Select, the number of the desired STM1 input.

NOTE Having two or more (or even all) DX inputs from the same STM1 is perfectly acceptable, and can be advantageous when data from multiple tributaries in one STM1 is required.

Figure 1. DE1 Firmware with OCM

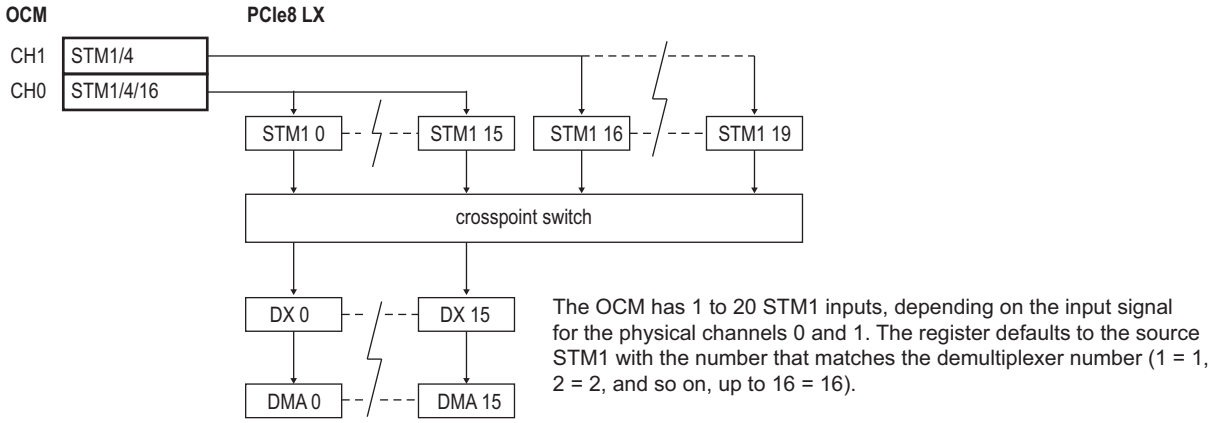
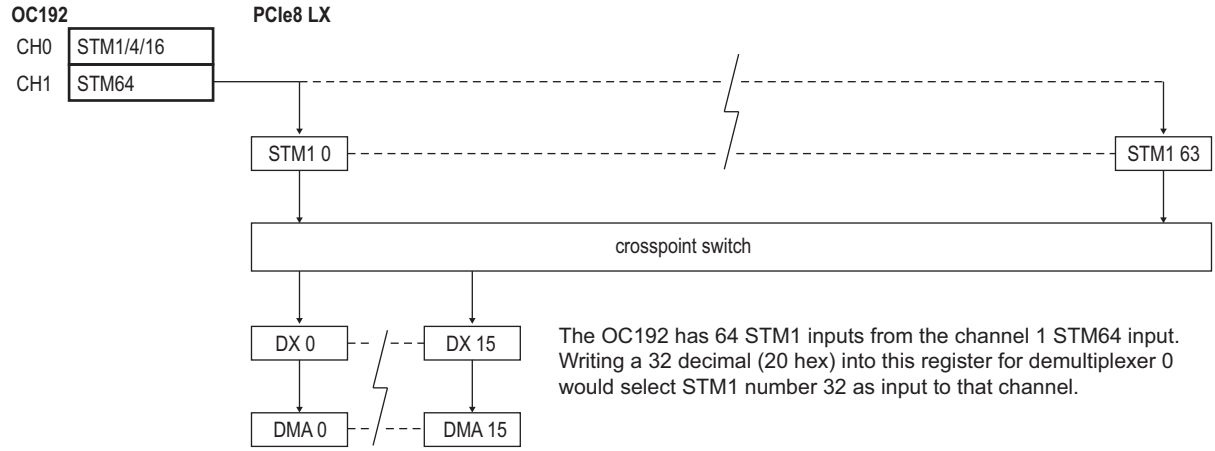


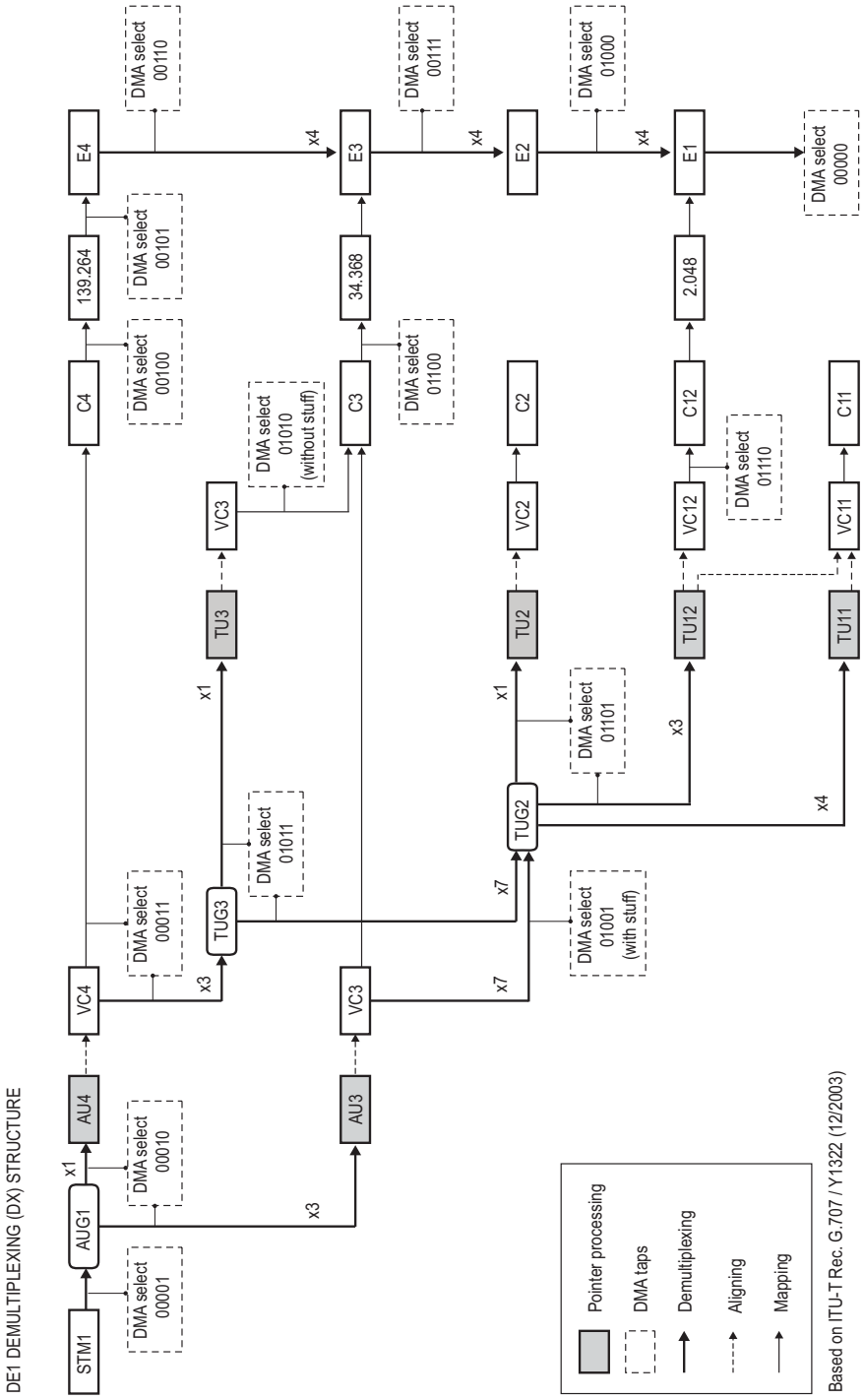
Figure 2. DE1 Firmware with OC192



STM1 Demultiplexing Structure

As described in the G.707 specification, each STM1 signal can be constructed of many types of lower-level subsignals – for example, C (container), VC (virtual container), and TUG (tributary user group) signals – all the way down to E1 signals. For details on the various multiplexing combinations, see Figure 3.

Figure 3. DE1 demultiplexing (DX) structure



Anomalies

In some cases, the overhead information needed for automatic demultiplexing is missing or incorrect. To handle these anomalies, the DE1 firmware provides path control bits to force the demultiplexing behavior required (see DE1 Behavior and Path Control on page 10).

E1 Packets

As part of the demultiplexing process, the DE1 firmware formats the E1 subsignals into packets. Each E1 packet contains a timestamp, status bits, an E1 descriptor, and E1 frame data.

The E1 data format is encapsulated in `struct edt_sdh_e1_buf_t`, defined in `lib_sdh.h` (included in the DE1 API in your EDT installation package).

The raw STM1 data can be included in the E1 packet stream, contained in special packets marked with a tag (see bit 15 in Table 2, below). This mode, useful for debugging, is enabled via bit 7 in 0x00D000 DMA Selection.

Below, Table 1 shows the general layout / data format of one of the packets of an E1 subsignal, while Table 2 shows how the bits are used for the timestamp, status bits, and E1 subsignal descriptor.

Table 1. E1 / VC12 Layout / Data Format

	bit 31			bit 0
word	byte 3	byte 2	byte 1	byte 0
0	timestamp (in seconds)			
1	timestamp (in fractions of seconds)			
2	frame status	E1 number	length	
3	channel 3	channel 2	channel 1	channel 0
4	channel 7	channel 6	channel 5	channel 4
5	channel 11	channel 10	channel 9	channel 8
6	channel 15	channel 14	channel 13	channel 12
7	channel 19	channel 18	channel 17	channel 16
8	channel 23	channel 22	channel 21	channel 20
9	channel 27	channel 26	channel 25	channel 24
10	channel 31	channel 30	channel 29	channel 28
	32–35 are extra placeholder bytes. 32–35 will contain VC12 data when capturing VC12.			
11	34–35 can contain specialized synchronous E1 signals which are outside the standard.			

Table 2. Frame Status, E1 Number, and Length Fields

Bit	Name	Description
15	[no name]	Set when the packet contains raw STM1 data. When this is true, the length is correct, but the timestamp and all other tag bits are 0.
14–13	[no name]	Reserved; always 0.
12	VC12_FRAME_1	Valid when this packet contains a VC12 (see register 0x00D4YY VC12 Control). Used in conjunction with bit 7 (VC12_FRAME_0) to indicate which VC12 frame: 00 = V5 frame 10 = N2 frame 01 = J2 frame 11 = K4 frame
11–9	E1_TAG	Describes the source of the E1. Of the eight possible bit patterns, only these four are valid: 100 = from E4 010 = asynchronous E1 contained in a VC12 011 = synchronous E1 contained in a VC12 000 = asynchronous E1 from TUG3 to VC3 to E3 path. In other words: Bit 11 = Set if E1 is from an E4. Bit 10 = Set if E1 is from a VC12. Bit 9 = Set for synchronous E1.
8	E1_FRAMED	Set when the asynchronous E1 (bit 9 = 0) signal frame alignment is valid; otherwise, always 0.
7	E1_ODDFRM	When capturing a framed E1, set when this packet is is the odd frame of a frame pair; cleared when it is even.
	VC12_FRAME_0	Valid when capturing VC12 data (see register 0x00D4YY VC12 Control). Used in conjunction with bit 12.
6	[no name]	Reserved; always 0.
5-0	E1_NUMBER	Demultiplex from E4: 5-4 = E3 number 3-2 = E2 number 1-0 = E1 number Demultiplex from TUG3 to VC3: 5-0 = (TUG3 number * 21) + (E2 number * 4) + E1 number Demultiplex from TUG3 to TUG2: 5-0 = (TUG3 number * 21) + (TUG2 number * 3) + VC12 number
15–0	LENGTH	16-bit unsigned integer containing the size of the E1 packet, in bytes.

For the complete STM1-to-E1 demultiplexing structure, see Figure 3 on page 7. For details on DE1 automatic and override behavior, see DE1 Behavior and Path Control on page 10.

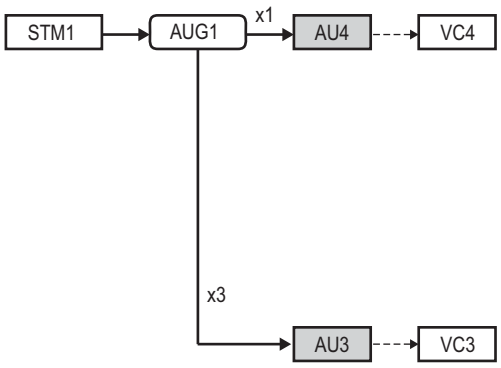
DE1 Behavior and Path Control

This section shows details from Figure 3 on page 7 to explain how the DE1 firmware deconstructs STM1 signals into their various subsignals.

NOTE As you use this section, note that several of the subsignals can be interpreted more than one way.

The section also explains how to use the automatic and override behavior of the DE1 firmware to get the results you need (see the registers 0x00D082-083 Force Path and 0x00D084 Disable Path).

From STM1 to VC4 or VC3



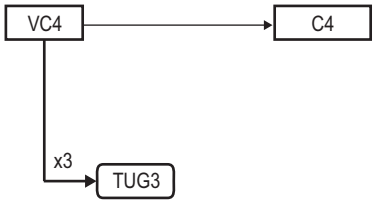
First, the DE1 firmware uses the contents of the STM1 overhead pointer bytes H1 and H2 to determine whether the original STM1 contains a VC4 or a VC3.

Each STM1 has three pointers – one for each of three possible VCs. If the second and third pointers are set to the concatenation value (0x3FF), then the first pointer is used to find the first data byte of a VC4. If the concatenation values are not present, each pointer references the first data byte of the respective VC3.

0x00D082-083 Force Path provides bit 0 to force a VC4 interpretation, and bit 1 to force three VC3s. If both bits are set, the VC4 overrides.

NOTE The STM1 pointers are unlikely to be incorrect, so this force path is unlikely to be needed. If you do use it, exercise caution: a VC3-to-TUG2-to-TU12-to-VC12 forced as a VC4 has been seen to generate some but not all E1s.

From VC4 to C4 or TUG3



The VC4 can be interpreted as a C4 (which carries data formats beyond the scope of G.707) or TUG3.

Each VC4 has certain associated path overhead (POH) bytes. One of these bytes, labeled C2, is the signal label. In the DE1 firmware, if C2 is set to 0x02, then further demultiplexing of the TUG3 structure is required; otherwise, a C4 is assumed.

NOTE The most likely error at this point would be to have the C2 byte set to unequipped (0x00).

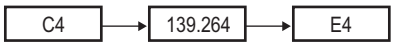
As above, the indirect register 0x00D082-083 Force Path provides bit 2 to force the VC4 to the C4 path, or bit 3 to force the VC4 to the TUG3 path. If the signal label is exactly opposite the desired path (e.g., C2 = 0x12 but TUG3 processing is required), the C4 path must be disabled in addition to forcing the TUG3. Bit 0 of register 0x00D084 Disable Path is provided to prevent the potential spurious data.

Table 3 is drawn from the G.707 specification.

Table 3. C2 byte hex codes

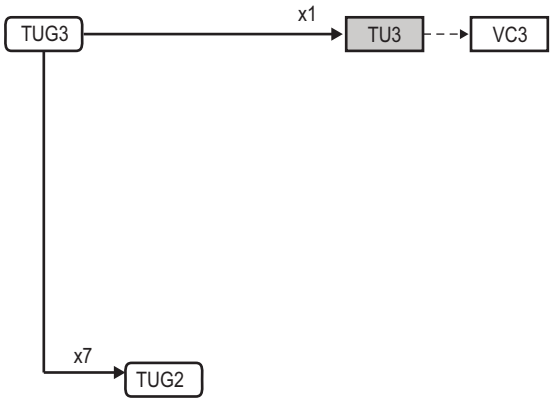
Code	Interpretation	Code	Interpretation
0x00	Unequipped or supervisory-unequipped	0x18	Mapping of HDLC/LAPS framed signals
0x01	Reserved	0x19	Reserved for proprietary use
0x02	TUG structure	0x20	Asynchronous mapping of ODUk (k=1,2) into VC-4-Xv (X=17,68)
0x03	Locked TU-n	0x1a	Mapping of 10 Gbit/s ethernet frames
0x04	Asynchronous mapping of thirty-four 368 kbit/s or forty-four 736 kbit/s into the C3	0x1b	GFP mapping
0x05	Experimental mapping	0x1c	Mapping of 10 Gbit/s fiber channel frames
0x12	Asynchronous mapping of 139 264 kbit/s into the C4	0xcf	Reserved
0x13	ATM mapping	0xd0–df	Reserved for proprietary use
0x14	MAN DQDB mapping	0xe1–fc	Reserved for national use
0x15	FDDI mapping	0xfe	Test signal, O.181 specific mapping
0x16	Mapping of HDLC/PPP framed signal	0xff	VC-AIS
0x17	Reserved for proprietary use		

From C4 to E1



Once the data is identified as C4, the DE1 firmware again uses the C2 label to determine further processing. The firmware processes only 139Mbit data (labeled 0x12). If the label is incorrect, you can force the C4 data to be interpreted as 139Mbit by setting bit 4 in 0x00D082-083 Force Path.

From TUG3 to VC3 or TUG2

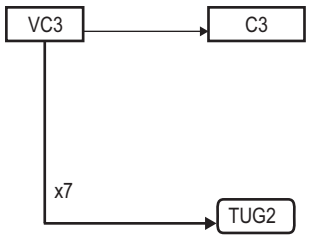


A TUG3 signal has three components, and each one can be demultiplexed independently to a VC3 or a TUG2.

There is no clear automatic method to determine which way a component can go, so the DE1 firmware forwards each component to both paths on the assumption that the next stage will be able to process only a correctly formatted data stream.

To allow the correct behavior to be forced for a known signal, 0x00D084 Disable Path is written with either a disable mask for the VC3 (34Mbit signal) bits 6-4, or TUG2 disable bits 3–1.

From VC3 to C3 or TUG2



Similarly, a VC3 signal (whether determined or forced to be a VC3) can be interpreted as either a C3 or a TUG2. In its POH, each VC3 has a signal label byte which can steer the path automatically (see Table 3). Again, if the C2 byte is not correct, the behavior can be overridden with bits 7-5 (force TUG2) or bits 10-8 (force C3) in 0x00D082-083 Force Path. Also, the disable bits will still function as well, so it is possible to force one direction and then disable that same direction to yield no data (see 0x00D084 Disable Path).

Examples

For each common force path below, use 0x00D082-083 Force Path and 0x00D084 Disable Path as shown.

Commonly used path: 0x00D082-083 Force Path / 0x00D084 Disable Path

64 E1s from VC4 (VC4 to C4 to 139Mb to E4)0x15 / 0x7e

63 E1s from VC4 (VC4 to TUG3 to TUG2)0xe9 / 0x71

63 E1s from three VC3s (VC3 to TUG2)0xe2 / 0x71

Two classes of register access are controlled by the most significant nibble: regular (0) and indirect (currently 1 for EDT indirect, and 2 for customer A indirect).

Basic Testing

The API calls in `lib_sdh.c` make it easy for you to: configure the mezzanine board interface; set up and run DMA datastream processing across all sixteen possible STM1 demultiplexing channels; obtain status information about the subsignal demultiplexing; and perform debugging by configuring each STM1 DMA channel to provide raw upstream data taps.

Example Code

This example code initializes the firmware and configures a single STM1 datastream for processing:

```
#include “edtinc.h”
#include “lib_sdh.h”
main()
{
    int i; int unit    = 0;
    int channel = 0;
    edt_sdh_t *sdh_p = NULL;
    sdh_p = edt_sdh_board_setup (unit, STM1_RATE);
    edt_sdh_board_set_clock_to_system_time (sdh_p);
    edt_sdh_start_stm1_to_e1_channel(sdh_p, channel, CALLBACK_PER_E1_FRAME, e1Callback,
    NULL);

}
```

For a complete example program, see `stm1e1prbs15.c` in your EDT installation package.

stm1e1prbs15

Running `stm1e1prbs15` with no arguments will try STM1, STM4, and STM16 configurations, in that order, until the program senses framelock. The following options are accepted.

`-r n` Replace *n* with the number that corresponds to the configuration you need:

0 = use the existing configuration

1 = STM1

4 = STM4

16 = STM16

`-c n` Replace *n* with 0 through 15 to select one of the sixteen DMA channels / DXs.

`-l n` Replace *n* with the number of buffers you wish to process. Each buffer contains the E1 packets (32 x 1024) from the selected DMA channel / DX.

Output should look like this (where each Y shows no `prbs15` errors for the corresponding E1 number):

```
E1-0x6e: e3frm 1 invert 1 skp_ts16 0
prbs(YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY) time
1250120188.771120
```

GUI Program

The EDT installation package also includes a GUI program and its source code (`sdhgui`, in the `sdhgui.src` subdirectory) which displays the SDH path currently input to the DE1 interface, along with path information. To create and run this GUI program:

1. Change directory to `sdhgui.src.`
2. Run `make` to create the program in your EDT installation package.
3. Run `sdhgui` from your EDT installation package.

EDT Time Functions

Certain EDT FPGA configuration files incorporate time stamps associated with the acquired data. In general, the time can be set either from an external time source, or by software synchronization with the host system time. However, the files `combo_pdh_demuxin` and `combo3_pdh_demuxin` are designed to be synchronized with the system time.

The EDT time functions use Unix time, which counts the seconds from the start of January 1, 1970, with periodic adjustments to match the rotation of the earth (UTC). EDT time functions represent time as a 64-bit value in which the most significant 32 bits equal the least significant 32 bits of a 64-bit representation of Unix time. If you need to track the most significant 32 bits of Unix time (which increment once every 136 years), your application must implement this functionality.

The least significant 32 bits of EDT time represent fractions of a second. A single increment in this number is approximately 233 picoseconds. This level of accuracy is impractical and unnecessary for most applications, so the number of significant bits your application uses can vary as needed. The `combo_pdh_demuxin` and `combo3_pdh_demuxin` configuration files implement twenty bits, for a time stamp accuracy of $1/2^{20}$ of a second, or approximately 954 nanoseconds.

A 32-bit operating system representation of Unix time keeps the second counter in a single signed 32-bit integer. The 32-bit second counter in EDT time will be compatible with this representation of time until the year 2036, when the 32-bit Unix time will wrap around to 1904.

Adjustments

Because EDT time is kept on the board, it is based on a crystal oscillator, which is subject to initial accuracy errors as well as temperature and aging errors. To compensate for these errors, the time circuits include an adjustment counter. This counter adds or holds the least significant bit of the fractional second counter on each overflow.

For time based on the host system, the EDT software routines make adjustments gradually, as the application detects divergence from the system time (not automatically, as would occur if the time were hardware-based). Due to such adjustments, the least significant bit sometimes repeats or jumps by two, so the relative accuracy of the clock is, in the best case, plus or minus one least significant bit (about one µsecond for `combo_pdh_demuxin` and `combo3_pdh_demuxin`). The absolute accuracy of the agreement between the system clock and EDT time depends on the operating system and your system's response time. Linux can maintain 20- to 50-µsec agreement; Windows is more erratic.

EDT software implements these adjustments gradually to prevent large jumps in time between events, particularly negative jumps where the relative time is important. If the time is significantly wrong, such as at startup, your application can set it directly.

EDT Time Software Functions

EDT time software functions include setting the board time to system time as Unix time, retrieving the 64-bit time value, and adjusting for the errors between system time and EDT time. The clock on the EDT board can be adjusted to compensate for the drift between board time and system time, as well as adjusted to converge back to the desired system time without time values ever decreasing. Also, functions are provided to create a monitoring thread that periodically samples the error between EDT time and system time, and then adjusts the board time accordingly.

EDT time starts automatically as soon as the FPGA configuration file is loaded.

For complete function documentation, see the API link under DE1 Firmware with OC192 on page 6. For some of the most useful functions, see Table 4.

Table 4. EDT Time Software Functions - Abridged List

Purpose	Function
To set the time to current system time:	`edt_sstm_set_to_sys`
To retrieve the current time:	`edt_ss_timestamp`
To get the current error between EDT time and system time:	`edt_sstm_measure_drift`
To measure the drift between EDT time and system time:	`edt_sstm_sys_error`
To calculate the current error and revert to system time gradually:	`edt_sstm_iterate_adjust`
To create and start an adjustment thread:	`edt_sstm_launch_adjuster`

There is only one time clock per board, so it doesn't matter which DMA channel an application opens.

The example code below will set the board time and then launch an adjustment thread that samples every five minutes.

```
edt_p = edt_open(EDT_INTERFACE, unit);
edt_sstm_set_to_sys(edt_p);
adjuster = edt_sstm_launch_adjuster(edt_p,
       300, // check every 5 minutes
       20, / # of iterations of adjustment as error gets smaller
       10, // each iteration should take 10 secs.
       200,  // maximum 200 microsecond error allowed
       20,  // try to get within 20 microseconds
       0    // loop indefinitely
       );
// for this example just go to sleep
while (1)
       edt_msleep(300000);
```

The example program provided, `edt_ss_time.c`, implements the above code. To run it, enter:

```
edt_ss_time -T -L 300 20 200
```

The example program also exercises the other EDT time functions.

Accessing the Registers

The DE1 registers are accessed indirectly through a 32-bit control word which accesses 24 bits of register address space.

NOTE Indirect register access involves EDT board register accesses which are not atomic, so indirect registers are not thread-safe; each one should be accessed by a single thread at a time.

In the OCM or OC192 user's guide (under Related Resources on page 5), see the Extended Indirect Register Address Registers (0x60, 0x61, 0x62) and Extended Indirect Register Data Register (0x63).

Use the special indirect register functions (from the `lib_sdh` library) to access individual registers:

```
va1=edt_indirect_reg_read(edt_p,addr)
```

or

```
edt_indirect_reg_write(edt.p,addr,val)
```

Two classes of register access are controlled by the most significant nibble: direct (0) and indirect (currently 1 for EDT indirect, and 2 for a reserved custom indirect interface).

Within regular access, there are various types of register access controlled by the third most significant byte (*TT* below): DMA registers (0), UI FPGA registers (1), per-channel DMA registers (3), and others.

The register access control word (32 bits) is defined in Table 5, below.

Table 5. Register Access Control Word (32 bits)

Address:		0xIS *TT* AAAA, defined as follows: 31–28 (I) = direct / indirect access code 27–24 (S) = size of register in bytes 23–20 (TT) = register type code 19–0 (AAAA) = 16-bit register offset (or address)
Bit	**Purpose**	**Description**
31-28	Direct / indirect access control nibble	Byte 3, nibble 1 contains an indirect access code, defined below. For access control values 0 and 1, data is written to multibyte registers starting at the base address + 0, + 1, + 2, etc. 0 = Direct access. 1 = EDT indirect access; indirect access register base is 0x60 unless otherwise specified by: `edt_set_indirect_register_base(edt_p, base_address)` If `base_address` is set to 0, then the default 0x60 value is used instead; otherwise its value is the value of M_EXTRegBase. 2 = A reserved custom indirect register access method.
27-24	Register size	Byte 3, nibble 0 contains the size of the register in bytes.
23-20	Register type code	If indirect access nibble is 0: Byte 2 selects the type (access method) of the register.
19-0	16-bit register offset (or address)	If indirect access nibble is 0: Bytes 0 and 1 are the type-dependent register address.

Below are example macro definitions to access standard indirect registers.

```
/* EDT indirect registers */
#define INDREG_1BYTE0x11000000
#define INDREG_2BYTE0x12000000
#define INDREG_4BYTE0x14000000
#define INDREG_TEST00x11000000
#define INDREG_TEST10x11000001
#define INDREG_TEST20x11000002
#define INDREG_TEST30x11000003
#define INDREG_TEST40x12000004
#define INDREG_TEST50x14000006
#define INDREG_TEST60x1800000a
```

Registers

These registers are accessed indirectly using the standard registers 0x60–63.

In the OCM or OC192 user's guide (see Related Resources on page 5), see the Extended Indirect Register Address Registers (0x60, 0x61, 0x62) and Extended Indirect Register Data Register (0x63).

- For AU3, TU3, TUG3, TU12, VC3, and VC12: Valid indexes are 0 to 2; an index of 3 is undefined.
- For AU4 and VC4: Valid index is 0; indexes 1 to 3 are undefined.
- For TUG2: Valid indexes are 0 to 6; an index of 7 is undefined.
- For E1, E2, E3, E4, and STM1: All indexes are valid.

Each STM1 demultiplexer (DX) contains the registers below. To select the DX you want to program, write its number in place of the "D" in the register address.

0x00D000 DMA Selection

Access / Notes: 8-bit read-write. D defines which DX.

Bit	Name	Description
7	[no name]	Enables raw STM1 data packets in demultiplexed E1 packet stream. See Tables 1 and 2 on page 8.
6-5	[no name]	Reserved.
4-0	[no name]	*DMA select: 00000–00111* 00000 = tagged E1 00001 = STM1 data 00010 = AUG1 data 00011 = VC4 data 00100 = C4 data 00101 = M139 data 00110 = E4 data 00111 = E3 data *DMA select: 01000–01110* 01000 = E2 data 01001 = VC3 data with stuff 01010 = VC3 data without stuff 01011 = TUG3 data 01100 = C3 data 01101 = TUG2 data 01110 = VC12 data

0x00D001 DMA Subselection

Access / Notes: 8-bit read-write. D defines which DX.

First, use 0x00D000 DMA Selection to select your data type; then, if needed, find the corresponding data source in this register (below).

Bit	Name	Description
7	[no name]	Enable unframed data.
6-0	[no name]	Below, select the appropriate addressing scheme; set unused bits to 0.

Data source	*Bits*	*Data source*	*Bits*
E3	E3 = 1-0	TUG3	TUG 3 = 1-0
E2	E2 = 3-0	C3 via AU3	VC3 = 1-0 (set bit 2 to 0)
VC3 with stuff	VC3 = 1-0	C3 via AU4	VC3 = 1-0 (set bit 2 to 1)
VC3 via TU3	VC3 = 1-0	TUG2	TUG3 = 4-3; TUG2 = 2-0
VC3 via AU3	VC3 = 1-0	VC12	TUG3 = 6-5; TUG2 = 4-2; VC12 = 1-0

0x00D080 STM1 Status

Access / Notes: 8-bit read-only. D defines which DX.

Bit	Name	Description
7-5	[no name]	Reserved.
4-3	[no name]	Frame count.
2-0	[no name]	Framing status: 000 = unframed 001 = A1 011 = potential framing 010 = A2 100 = framed

0x00D082-083 Force Path

Access / Notes: 16-bit read-write. D defines which DX.

Bit	Name	Description
15-11	[no name]	Reserved.
10-8	[no name]	Set to override the signal label and force a selected C3 signal as a 34 Mbit signal.
7-5	[no name]	Set to override the signal label and force a selected VC3 signal as a TUG2 signal.
4	[no name]	Set to override the signal label and force a selected C4 signal as a 139 Mbit signal.
3	[no name]	Set to override the signal label and force a selected VC4 signal as a TUG3 signal.
2	[no name]	Set to override the signal label and force a selected VC4 signal as a C4 signal.
1	[no name]	Set to override the concatenation pointer state and force a selected STM1 signal as three VC3 signals.
0	[no name]	Set to override the concatenation pointer state and force a selected STM1 signal as a VC4 signal.

0x00D084 Disable Path

Access / Notes: 8-bit read-write. D defines which DX.

Bit	Name	Description
7	[no name]	0
6-4	[no name]	Set to disable selected 34 Mbit signal.
3-1	[no name]	Set to disable selected TUG2 signal.
0	[no name]	Set to disable 139 Mbit signal.

0x00D086 STM1 Source Select

Access / Notes: 8-bit read-only. D defines which DX. See DE1 Crosspoint Switch on page 6.

Bit	Name	Description
7-0	[no name]	Select the STM1 you wish to input to this DX, depending on the mezzanine board you are using. The default is that the “D” in the register address equals the DX number. **OC192:** Physical channel 1 (fixed) is always a STM64 signal. Valid values are 0-63. **OCM:** For each physical channel, the allowable signals and values are shown below.

CH 0 STM1	CH 0 STM4	CH 0 STM16
0	0	0
	1	1
	2	2
	3	3
		...
		15

CH1 STM1	CH1 STM4
16	16
	17
	18
	19

0x00D1YY AU4/3 Current Pointer Low

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
00 = AU4 or AU3 number 1
08 = AU3 number 2
10 = AU3 number 3

Pointer is a 10-bit value.

Bit	Name	Description
7-0	[no name]	Pointer low bits (7 to 0).

0x00D1YY AU4/3 Current Pointer High

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
01 = AU4 or AU3 number 1
09 = AU3 number 2
11 = AU3 number 3

Bit	Name	Description
7-4	[no name]	0000
3-2	[no name]	SS value.
1-0	[no name]	Pointer high bits (9 and 8).

0x00D1YY AU4/3 Next Pointer State

Access / Notes: 8-bit read-only. D defines the which DX and YY defines bits as:
02 = AU4 or AU3 number 1
0A = AU3 number 2
12 = AU3 number 3

State bits are saved when H1 is received to process with the next H2 and update the current pointer (see 0x00D1YY AU4/3 Current Pointer High).

Bit	Name	Description
7-4	[no name]	0000
6-5	[no name]	XOR of current pointer bits 9-8 with the potential next pointer (used to determine pointer increment / decrement operation).
4	NDF	New data flag.
3-2	SS	Next SS bits.
1-0	[no name]	Next pointer high.

0x00D103 AU4/3 Pointer Status

Access / Notes: 8-bit read-only. D defines which DX.

Bit	Name	Description
7-6	[no name]	Reserved
5-3	[no name]	Set when the respective pointer matches G.707 rules for validity (reset indicates loss of pointer).
2-0	[no name]	Set when the pointer is a concatenation pointer (this should always be 000 for VC3 or 110 for VC4).

0x00D1YY Current VC4/3 B3

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
80 = VC4 or VC3 number 1
88 = VC3 number 2
90 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Value of current B3 byte.

0x00D1YY Last VC4/3 B3

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
81 = VC4 or VC3 number 1
89 = VC3 number 2
91 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Value of current B3 byte.

0x00D1YY Low VC4/3 B3 Error Count

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
7 = 1
82 = VC4 or VC3 number 1
8A = VC3 number 2
92 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Low 8 bits of B3 error count.

0x00D1YY High VC4/3 B3 Error Count

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
83 = VC4 or VC3 number 1
8B = VC3 number 2
93 = VC3 number 3

Bit	Name	Description
7-0	[no name]	High 8 bits of B3 error count.

0x00D1YY VC4/3 Signal Label

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
84 = VC4 or VC3 number 1
8C = VC3 number 2
94 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Value of VC4/3 signal label.

0x00D1YY Current VC4/3 Multiframe (MF) Indicator

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
85 = VC4 or VC3 number 1
8D = VC3 number 2
95 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Value of current multiframe indicator.

0x00D200 E4 Frame Status

Access / Notes: 8-bit read-only. D defines which DX.

Bit	Name	Description
7-5	[no name]	National bits
4	AI	Alarm indicator
3	[no name]	Caught
2-1	[no name]	Frame count
0	[no name]	Framed

0x00D210 E4 Frame Disable

Access / Notes: 8-bit read-write. D defines which DX.

Bit	Name	Description
7-2	[no name]	Unused.
1	[no name]	Set to disable passing E4 data to E3 framer; otherwise, an unused E4 can accidentally frame to spurious data and send that data to the associated E3 framer.
0	[no name]	Set to disable E4 framing logic when acquiring unframed E4 data; otherwise, coincidental framing patterns will cause bit shifts as the framer tries to align the data.

0x00D2YY E3 Frame Status

Access / Notes: 8-bit read-only .D defines which DX and YY defines bits as:
7 = 1
6-5 = E3 number
4-0 = 00000

Bit	Name	Description
7-5	[no name]	National bits.
4	AI	Alarm indicator.
3	[no name]	Caught.
2-1	[no name]	Frame count.
0	[no name]	Framed.

0x00D2YY E3 Frame Disable

Access / Notes: 8-bit read-write. D defines which DX and YY defines bits as:
7 = 1
6-5 = E3 number
4-0 = 10000

Bit	Name	Description
7-2	[no name]	Unused.
1	[no name]	Set to disable passing E3 data to E2 framer; otherwise, an unused E3 can accidentally frame to spurious data and send that data to the associated E2 framer.
0	[no name]	Set to disable E3 framing logic when acquiring unframed E3 data; otherwise, coincidental framing patterns will cause bit shifts as the framer tries to align the data.

0x00D3YY E2 Frame Status

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:
7 = 0
6-5 = E3 number
4 = 0
3-2 = E2 number
1-0 = 00

Bit	Name	Description
7-5	[no name]	National bits.
4	AI	Alarm indicator.

3	[no name]	Caught.
2-1	[no name]	Frame count.
0	[no name]	Framed.

0x00D3YY E2 Frame Disable

Access / Notes: 8-bit read-write. D defines which DX and YY defines bits as:

7 = 0
6-5 = E3 number
4 = 1
3-2 = E2 number
1-0 = 00

Bit	Name	Description
7-2	[no name]	Unused.
1	[no name]	Set to disable passing E2 data to E1 framer; otherwise, an unused E2 can accidentally frame to spurious data and send that data to the associated E1 framer.
0	[no name]	Set to disable E2 framing logic when acquiring unframed E2 data; otherwise, coincidental framing patterns will cause bit shifts as the framer tries to align the data.

0x00D3YY E1 Frame Status

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

If using E4/3 path...	*If using TUG2 path...*
7 = 1	7 = 1
6-5 = E3 number	6-5 = TUG3 or VC3 number (only 0-2 are valid)
4 = 0	4-2 = TUG2 number (only 0-6 are valid)
3-2 = E2 number	1-0 = VC12 number (only 0-2 are valid
1-0 = E1 number	

Bit	Name	Description
7-5	[no name]	National bits.
4	AI	Alarm indicator.
3	[no name]	Caught.
2-1	[no name]	Frame count.
0	[no name]	Framed.

0x00D4YY E1 Frame Disable

Access / Notes: 8-bit read-write. D defines which DX and YY defines bits as:

If using E4/3 path...	*If using TUG2 path...*
7 = 0	7 = 0
6-5 = E3 number	6-5 = TUG3 or VC3 number (only 0-2 are valid)
4 = 0	4-2 = TUG2 number (only 0-6 are valid)
3-2 = E2 number	1-0 = VC12 number (only 0-2 are valid
1-0 = E1 number	

Bit	Name	Description
7-2	[no name]	Unused.

1	[no name]	Set to disable passing E1 data to DMA; otherwise, an unused E1 can accidentally frame to spurious data and send that data to DMA, wasting system resources.
0	[no name]	Set to disable E1 framing logic when acquiring unframed E1 data; otherwise, coincidental framing patterns will cause bit shifts as the framer tries to align the data.

0x00D4YY VC12 Control

Access / Notes: 8-bit read-write. D defines which DX and YY defines bits as:

[VC12 is always a TUG2 path...]
7 = 1
6-5 = TUG3 or VC3 number (only 0-2 are valid)
4-2 = TUG2 number (only 0-6 are valid)
1-0 = VC12 number (only 0-2 are valid

Bit	Name	Description
7-4	[no name]	Unused.
3	[no name]	Read only; signal type in VC12 V5 path byte indicates synchronous E1.
2–1	[no name]	Forced mode select. If bit 0 is set... 00 = Asynchronous E1 01 = Synchronous E1 10 = Undefined 11 = VC12
0	[no name]	Set to force manual select mode.

0x00D5YY Current TU3 Pointer Low (H2)

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

00 = TU3 number 1
08 = TU3 number 2
10 = TU3 number 3

Bit	Name	Description
7-0	[no name]	Pointer low bits.

0x00D5YY Current TU3 Pointer High (H1)

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

01 = TU3 number 1
09 = TU3 number 2
11 = TU3 number 3

Bit	Name	Description
7-4	[no name]	Reserved.
3-2	[no name]	SS value.
1-0	[no name]	Pointer high (two bits, 9-8).

0x00D5YY TU3 Next State

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

02 = TU3 number 1
0A = TU3 number 2
12 = TU3 number 3

State bits are saved when H1 is received to process with the next H2 and update 0x00D4YY VC12 Control and 0x00D5YY Current TU3 Pointer High (H1).

Bit	Name	Description
7-4	AIS	Alarm indicator signal
6-5	[no name]	XOR of current pointer bits 8-9 with the potential next pointer (used to determine pointer increment / decrement operation).
4	NDF	New data flag.
3-2	SS	Next SS bits.
1-0	[no name]	Next pointer high.

0x00D5YY TU3 Pointer Status

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

03 = TU3 number 1
0B = TU3 number 2
13 = TU3 number 3

State bits are saved when H1 is received to process with the next H2 and update 0x00D4YY VC12 Control and 0x00D5YY Current TU3 Pointer High (H1).

Bit	Name	Description
7-3	[no name]	Reserved.
2-0	[no name]	Pointer valid.

0x00D5YY VC3 (via TU3) B3

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

80 = VC3 number 1
88 = VC3 number 2
90 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Value of current B3 byte.

0x00D5YY Last VC3 (via TU3) B3

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

81 = VC3 number 1
89 = VC3 number 2
91 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Value of last B3 byte.

0x00D5YY Low VC3 (via TU3) B3 Error Count

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

82 = VC3 number 1
8A = VC3 number 2
92 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Low 8 bits of B3 error count.

0x00D5YY High VC3 (via TU3) B3 Error Count

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

83 = VC3 number 1
8B = VC3 number 2
93 = VC3 number 3

Bit	Name	Description
7-0	[no name]	High 8 bits of B3 error count.

0x00D5YY VC3 (via TU3) Signal Label

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

84 = VC3 number 1
8C = VC3 number 2
94 = VC3 number 3

Bit	Name	Description
7-0	[no name]	Value of VC3 signal label.

0x00DYYY Current TU12 Pointer Low

Access / Notes: 8-bit read-only. D defines which DX and YYY defines bits as:

11-9 = 011
8-7 = TUG3 or VC3 number
6 = 0
5-3 = TUG2 number
2-1 = VC12 number
0 = 0

Bit	Name	Description
7-0	[no name]	Pointer low.

0x00DYYY Current TU12 Pointer High and Status

Access / Notes: 8-bit read-only. D defines which DX and YYY defines bits as:

11-9 = 011
8-7 = TUG3 or VC3 number
6 = 0
5-3 = TUG2 number
2-1 = VC12 number
0 = 1

Bit	Name	Description
7	[no name]	Valid.
6-5	[no name]	Valid count.
4-2	[no name]	Reserved.
1-0	[no name]	Pointer high.

0x00D8YR STM1 VC4/3 POH

Access / Notes: 8-bit read-only. D defines which DX, R defines POH byte, and Y defines bits as:

3-2 = 0
1-0 = VC4/3

Bit	Name	Description
7-0	[no name]	POH. Below are the contents of each available POH byte in this register

Byte	*Label*	*Description*
0	J1	VC-n path trace.
1	B3	Path BIP-8.
2	C2	Path signal label.
3	G1	Path status.
4	F2	Path user channel.
5	H4	TU multiframe indicator.
6	F3	Path user channel.
7	K3	Automatic protection switching (APS).
8	N1	Network operator - tandem connection monitor (TCM).

0x00D8YR VC3 (via TU3) POH

Access / Notes: 8-bit read-only. D defines which DX, R defines POH byte, and Y defines bits as:

3-2 = 0
1-0 = VC3

R is a POH row between 0 and 8.

Bit	Name	Description
7-0	[no name]	POH. Below are the contents of each available POH byte in this register.

Byte	*Label*	*Description*
0	J1	VC-n path trace.
1	B3	Path BIP-8.
2	C2	Path signal label.
3	G1	Path status.
4	F2	Path user channel.
5	H4	TU multiframe indicator.
6	F3	Path user channel.
7	K3	Automatic protection switching (APS).
8	N1	Network operator - tandem connection monitor (TCM).

0x00DYYY VC12 POH

Access / Notes: 8-bit read-only. D defines which DX and YY defines bits as:

11-9 = 101
8-7 = TUG3 or VC3
6-4 = TUG2
3-2 = VC12
1-0 = POH byte

Bit	Name	Description
7-0	[no name]	POH. Below are the contents of each available POH byte in this register.

Byte	*Label*	*Description*
0	V5	TU multiframe indicator.
1	J2	Access path identifier.
2	N2	Network operator - tandem connection monitor (TCM).
3	K4	Automatic protection switching (APS).

0xFF0080 Capture Time

Access / Notes: EDT_TIME_SNAPSHOT / 64-bit read-only.

Reads EDT Time as of the last time the latch time bit was set in 0xFF008F Time Control.

Bit	Name	Description
63-32	SEC_TIME	Elapsed time in seconds.
31-12	FRAC_TIME	Elapsed time in fractions of a second. Each increment is $1/2^{20}$ second.
11-0	[no name]	Always 0; the minimum time resolution is therefore $1/2^{20}$ second.

0xFF0088 Set Time

Access / Notes: EDT_TIME_SET / 32-bit read-write.

Bit	Name	Description
61-0	SET_TIME	Write either a 32-bit start time for the second counter, or a 24-bit adjustment time. The value is written into the correct register by setting the appropriate bit in 0xFF008F Time Control.

0xFF008C Time Adjust

Access / Notes: EDT_TIME_ADJUST / 24-bit read-only.

Bit	Name	Description
23-0	ADJ_TIME	Read the current adjustment time. Adjustment time is explained in EDT Time Functions on page 13; for details, see API link under Related Resources on page 5. Requires that bit 4 be set in 0xFF008F Time Control.

0xFF008F Time Control

Access / Notes: EDT_TIME_CTL / 8-bit read-write.

Bit	Name	Description
7-4	[no name]	Reserved.
6	FREEZE_TIME	Set to disable increment of time.
5	ADJUST_PLUS	Set to increment fractional seconds by 2 when the adjustment timer expires. Clear to skip an increment when the adjustment timer expires. Requires that bit 4 be set.
4	ADJUST_EN	Set to enable time adjustment; clear if no adjustment is required.
3	[no name]	Reserved.
2	SET_ADJUST	Set to transfer the contents of 0xFF0088 Set Time into 0xFF008C Time Adjust.
1	CAP_TIME	Set to capture a 64-bit time value into 0xFF0080 Capture Time.
0	SET_SEC	Set to transfer the contents of 0xFF0088 Set Time into the seconds counter.

Revision Log

Below is a history of modifications to this guide.

Date	Rev	By	Page(s)	Detail
20150624	0006	PH,SB	6,10-12	Repaired diagrams.
20141209	0005	PH,MM	6	Under “DE1 Crosspoint Switch” section, changed cross-reference from ~~0x00D1YY AU4/3 Current Pointer Low~~ to 0x00D086 STM1 Source Select.
20130715	04b	PH,MM	7,9	- Resized Fig. 2 (demultiplexing structure) to fit with heading on page 7. - In Table 2 (frame status, E1 number, and length fields), under bits 5–0, for demultiplexing from E4: corrected “6–5 = E3 number” to “5–4 = E3 number.”
20120730	04a	PH,DL	8	Under Anomalies heading, first sentence, added the word “automatic” and deleted second clause, as follows: “In some cases, the overhead information needed for **automatic** demultiplexing is missing or incorrect~~; in other cases, a correct signal for one path can mimic a correct signal for another and so distort the data~~.”
20120730	04a	PH,MM, DL	28	In register 0x00DYYY VC12 POH, made one correction as shown: 11 – ~~9~~ **0** = 101
20120730	04a	PH,MM, DL	25	In register 0x00D5YY TU3 Pointer Status, made two corrections as shown: ~~03~~ **0B** = TU3 number 2 ~~12~~ **13** = TU3 number 3
20120730	04a	PH	All	Repaginated entire guide with continuous arabic numerals from title page to end.
20110419	04	PH,MM	4-5	In Table 2, changed title to “Frame Status, E1 Number, and Length Fields” and added a cell describing the length field (bits 15-0).
20110111	04	PH,DL	4-5	Under “E1 Packets” section, added paragraph about raw STM1. In Table 2, revised descriptions of bit 12 and bits 7-0.
20101122	03.11	PH,MM	4	In Table 1, reordered Word 2 as: frame status, E1 number, length.
20101013	03.10	PH,MM	3	Deleted final “x4” (under E1) from demultiplexing diagram.
20101013	03.10	PH,MM	17	On both 0x00D2YY registers, corrected titles from “E4” to “E3.”
20101004	03	PH,DL	2	Updated “DE1 Crosspoint Switch” diagram information for OCM and OC192.
20100604	02	PH,DL	4	Added VC12 information to tables.
20100415	01	PH	All	Updated formats.
20091100	00	PH,DL	All	Created this new guide.

Contact

Sky Blue Microsystems GmbH
Geisenhausenerstr. 18
81379 Munich, Germany
+49 89 780 2970, info@skyblue.de
www.skyblue.de

In Great Britain:
Zerif Technologies Ltd.
H5 Ash Tree Court
Nottingham NG8 6PY, England
+44 115 855 7883, info@zerif.co.uk
www.zerif.co.uk