LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
-LIGO-
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note LIGO-E0900036 01/24/09

Diagnostics Information for the Advanced LIGO Optical Timing Distribution System

Imre Bartos, and Daniel Sigg

This is an internal working note
of the LIGO Project.

California Institute of Technology
LIGO Project – MS 51-33
Pasadena CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, MIT NW22-295,
185 Albany St.,Cambridge, MA 02139 USA
Phone (617) 253 4824
Fax (617) 253 7014
E-mail: info@ligo.mit.edu

Columbia University
Columbia Astrophysics Laboratory
Pupin Hall - MS 5247
New York NY 10027
Phone (212) 854-8209
Fax (212) 854-8121
E-mail: geco.cu@gmail.com

WWW: http://www.ligo.caltech.edu

Diagnostics Information for the Advanced LIGO Optical Timing Distribution System

---

In this document we describe diagnostics information transfer, protocol and structure of the Advanced LIGO Optical Timing Distribution System (OTD). We first review how data is collected and transferred from the OTD to a remote computer. Next, we briefly discuss how data is processed. We discuss in detail the structure of the collected data, as it appears on a remote computer. Finally, we describe the status information that is being collected. In the Appendix, we present an example status information file.

0.1 Data Collection

Diagnostics information between the Master, FanOut and Slave modules of the OTD are transferred every second through optical fibers along with timing information (8 MHz clock signal and 1PPS). Data encoding convention can be found in, e.g., LIGO-T070218-00-D. Data is sent from each Master and FanOut module to a DMT machine using RS422 to ethernet converters. A DMT monitor, called *"TimingCollector",* collects 2308 bytes of data from each Master and FanOut module every second. It then uses the addresses of the modules to construct a map of the physical fiber connections between the modules; it arranges the information in a tree with the Master being the root node. Each of the 16 branches of a Master or FanOut can be either an unconnected port, another FanOut or a Slave module on the map. The monitor *TimingCollector* then transforms this data structure into an exact XML representation using the LIGO light-weight format. *TimingCollector* finally writes the XML data into a shared memory buffer (same organization as the frame distribution) which can be read by any number of readers.

0.2 Data Processing

In the near future, additional reader monitors will be designed to support EPICS, the data acquisition system and the alarm handler. Each reader monitor will get the XML data from the shared memory buffer at 1 Hz, and will convert it into a C++ tree representing the original status information and fiber connections. User data processing will be added depending on the requirements and control room needs.

1. Internal Data Structure

In this section we discuss the internal data structure of the OTD, i.e. the data structure sent from the Master, FanOut and Slave modules that is eventually collected and preprocessed in a DMT computer. The review of this data structure allows an insight to the operation of the OTD. We will

also build the discussion of the characteristics of preprocessed data to this structure, i.e. we will trace back the parameters of the output XML data to the internal parameters.

A Master or FanOut module sends the following data to the DMT computer:

- Status & Configuration Information: 16 dwords (double word – 32 bit),
- Downlink port records (delay): 2 dwords each, one record for each of the 16 downlink port,
- GPS status: 8 dwords,
- Padding: 8 dwords,
- Slave information: 32 dwords each, one for each of the 16 possible slave ports, and
- CRC: 1 dword.

This accounts for 2308 bytes per Master or FanOut module (one dword is 32 bits).
Each Slave module sends the following data to a Master or FanOut module, where it appears as Slave information:

- Status & Configuration Info: 8 dwords, and
- Slave specific information: 24 dwords.

The parameters of Slave specific information depend on the Slave type. The following Slave types are defined:

- FanOut
- DuoTone
- Timing Comparator
- XO locking

The total amount of information written to the output is the sum of data from each module connected to the system. As mentioned before, information collected from different modules is organized in a tree-like structure.

In the following, we provide the detailed structure of the internal data. We will use the main data groups defined above, since the internal data inside the OTD is structured similarly. Each data group will be discussed in a separate subsection.

1.1 Status & Configuration Information

Status and configuration information is stored in 16 dwords called w0, w1,..., and w15, each containing 32 bits of data. Below is a list of the information content of w0 – w15:

w0 -- board id/revision
w1 -- board serial
w2 -- software id
w3 -- software revision
w4 -- GPS time
w5 -- module address
w6 -- Status
w7 -- Configuration
w8 -- OCXO Controls
w9 -- OCXO Error
w10 -- Uplink 1PPS delay
w11 -- Ext 1PPS delay
w12 -- GPS 1PPS delay
w13 -- Fanout Up/LOS
w14 -- Fanout Miss/DErr
w15 -- CRC & GPS error

In the following we list in detail the information content w0 – w15. We will sometimes use VHDL (*VHSIC hardware description language*) notation and the variable names used in the FPGA code of the OTD.

```
-- board id/revision (example)
w0 <= X"070011B0";
```

Board ID encodes the DCC number of the Master / FanOut as well as the board revision, e.g. "0x070011B0" describes a revision B Master / FanOut with DCC number D070011.

```
-- board serial (currently not used)
w1 <= X"00000000";
-- software id (example)
w2 <= X"080284A0";
-- software revision (example)
w3 <= X"00000000";

-- GPS time
w4 <= GPS;
```

where GPS is a 32 bit UTC (Coordinated Universal Time) information.

```
-- address
w5 <= ADDR;
```

where ADDR is a 32 bit address of the board. Addresses are assigned automatically. A master node has always the address 0x00000000. Each of its branches gets assigned addresses 0x10000000, 0x11000000, 0x12000000, ... 0x1F000000 in order of the fanout ports. If a fanout is connected to port 5, its branches get assigned the addresses 0x25000000, 0x25100000, 0x25200000, ... 0x25F00000. The first nibble (4 bit) of each address represents the nesting level. Each following nibble encodes a port number until a final slave is reached.

```
-- Status
w6 (31..16) <= control voltage applied to the OCXO (VCXO_CTRL);
w6 (15..8)  <= status of the 8 DIP switches (SW);
w6 (7..6)   <= '0';
w6 (5)      <= loss of signal at the uplink (LOS);
w6 (4..1)   <= number of consecutive seconds with missing 1PPS signals up to 16 (ERRC);
w6 (0)      <= uplink connection is up and running (UP);
```

where VCXO_CTRL encodes the control voltage applied to the OCXO, SW is the state of the eight switches on the board. LOS is the state of the uplink connector, being one if there is no cable connected. ERRC shows for how many seconds there has been no 1PPS arriving through the uplink input, being reset-ed to zero every time an input arrives. It shows missing 1PPS signals up to 16, for more missing signals it remains at 16. UP shows that the uplink connection is up and running, i.e. an external device is connected to it (LOS = 0) and 1PPS signals come on time.

Here and thereafter, variable names in the brackets are the ones to denote the given variables in the FPGA code. For every one bit variable, the value is 1 if the description describes the actual state of the system. For example w(6) is 1 if the uplink connection exhibits loss of signal, while it is 0 if the signal is not lost. We will use these notations in the following as well.

```
-- Configuration
w7(31..16) <= unused
w7(15)     <= an OCXO is present (OCXO_PRESENT).
w7(14)     <= external 1 PPS present (EXT1PPS_ON)
w7(13)     <= a GPS module is present (GPS_ON)
w7(12)     <= the GPS module is locked (GPS_LOCKED)
w7(11)     <= the OCXO is locked (OCXO_LOCKED)
w7(10)     <= external 1PPS is used (OCXO_PRESENT and EXT1PPS_ON)
w7(9)      <= 1PPS from GPS is used (OCXO_PRESENT and GPS_ON and not
                                          EXT1PPS_ON)
w7(8)      <= 1PPS signal from uplink is used (not OCXO_PRESENT and UP)
w7(7)      <= uplink up and running (UP)
w7(6)      <= uplink LOS (LOS)
w7(5..2)   <= number of fan-out ports (0 represents 16 ports)
w7(1)      <= this is a fan-out module
w7(0)      <= this is a master module (OCXO_PRESENT)
```

where 'and' and 'no' are logical operators. OCXO_PRESENT is 1 if an OCXO is connected to the board, i.e. the board operates as a Master FanOut board. Accordingly, 0 shows that the board operates as a FanOut board. EXT1PPS_ON is 1 if 1PPS is arriving through the external 1PPS connector. GPS_ON is 1 if 1PPS is arriving through the GPS connector. GPS_LOCKED is 1 if the GPS board locks to the GPS signal received from the antenna. w7(10) is 1 if the board operates as a Master FanOut board and it receives the necessary information for proper operation through the external 1PPS connector. w7(9) is 1 if the board operates as a Master FanOut board and it receives the necessary information for proper operation through the GPS connector. W7(8) is 1 if the board operates as a FanOut board and it receives the necessary information for proper operation through the uplink input.

-- OCXO Controls
w8(31..16) <= unused;
w8(15..0) <= OCXO control voltage (OCXO_CTRL);

-- OCXO Error
w9 <= time difference between the internal 1PPS of the Master board obtained from the OCXO clock signal, and the external 1PPS, in units of 2^{-32} sec (OCXO_ERR);

-- Uplink 1PPS delay
w10 <= time difference between the uplink 1PPS input and the internal 1PPS. It is given in units of 2^{-32} s (UP1PPS);

-- Ext 1PPS delay
w11 <= time difference between the external 1PPS and the internal 1PPS. It is given in units of 2^{-32} s (EXT1PPS);

-- GPS 1PPS delay
w12 <= time difference between the GPS 1PPS and the internal 1PPS. It is given in units of 2^{-32} s (GPS1PPS);

-- Fanout Up/LOS
w13 <= FOUP & FOLOS;

where FOUP[0..15] shows if the 16 output channels are up and running (1 if an external device is connected and 1PPS is sent and received, and the external device is synchronized. FOUP[0] showing the state of output A1 and FOUP[15] showing the state of output D4). FOLOS[0..15] similarly shows if the 16 output channels are in the state of LOS (loss of signal, i.e. an external device is not connected).

-- Fanout Miss/DErr
w14 <= FOMISSING & FODLYERR;

where FOMISSING shows the missing status of external devices for the 16 output connectors (1 if the external device is not connected). An external device is considered missing from a connector if

there is no returned 1PPS signal from it to the Master FanOut or FanOut board. FODLYERR shows the delay error state for the 16 output connectors. One output connector exhibits delay error if the measured delay is not in the allowed window (1 ms).

```
-- CRC Error count
w15 <= X"00000" & GPS_ERROR & GPS_ERRC & CRCERR;
```

where GPS_ERROR is 1 if there is an error decoding the GPS signal AND CRCERR is 1. GPS_ERRC and CRCERR are error counters.

Status and configuration info is therefore constructed from the data described above as:

```
Status & Configuration Info <= w0 & w1 & w2 & w3  & w4   & w5   & w6   & w15 &
                               w7 & w8 & w9 & w10 & w11 & w12 & w13 & w14;
```

1.2 Downlink Port Records

Downlink port records contain delay and other related information for each of the 16 ports of a Master or FanOut module. For one port, the information is the following:

Delay Control (master/fanout):
DelayControl[31..14]: measured round-trip delay in units of 2^{-27} sec. The preset time advance is the same number in units of 2^{-28} sec. The actual number of cycles the outgoing 1PPS is advanced is given by first adding 2, and then dividing by 4 (discarding the fractional part).
DelayControl[13..12]: unused.
DelayControl[11..4]: counter for delay errors.
DelayControl[3]: missing delay measurement.
DelayControl[2]: measured delay error.
DelayControl[1]: fiber loss of signal.
DelayControl[0]: receiver is up and running.

Delay[31..0]: DELAY[31..0];

The output for port(i), called here PORT(i)(63..0), is constructed from the data described above as:

```
PORT(i)[63..0] <= DelayControl[31..0] & Delay[31..0];
```

1.3 GPS Status

GPS status contains information on the status of the GPS module, the universal time, and other information obtained by the GPS unit. GPS status information is transferred in the so called GPSStatus[1..8] unit, where, for simplicity, we use unsigned integer (uint32_t – C++ notation) representation instead of the usual bit representation. GPSStatus has the following components:

Latitude (double, "46.4551")
GPS latitude. Unit: degrees (-90 to +90).
= (double)((int)GPSStatus[0]) / 3.6E6;
Longitude (double, "-119.407")
GPS longitude. Unit: degrees (-180 to +180) .
= (double)((int)GPSStatus[1]) / 3.6E6;
Height (double, "163.92")
GPS height. Unit: m (-1000 to +18000).
= (double)((int)GPSStatus[2]) / 100.0;
Speed3D (double, "0.25")
GPS 3D speed. Unit m/s (0.0 to +514 m/s).
= (double)(GPSStatus[3] >> 16) / 100.0;
Speed2D (double, "0.01")
GPS 2D speed. Unit: m/s (0.0 to +514 m/s).
= (double)(GPSStatus[3] & 0xFFFF) / 100.0;
Heading (double, "153.4")
GPS heading. Unit: degree (0 to 360).
= (double)(GPSStatus[4] >> 16) / 10.0;
GPSDOP (double, "0.24")
GPS Dilution of Precision (0 to 99.9).
= (double)(GPSStatus[4] & 0xFFFF) / 100.0;
GPSSatellitesVisible (int, "10")
GPS number of visible satellites (0 to 99).
= (int)(GPSStatus[5] >> 24);
GPSSatellitesTracking (int, "8")
GPS number of tracked satellites (0 to 99).
= (int)((GPSStatus[5] >> 16) & 0xFF);
GPSReceiverStatus (int, "57345")
GPS receiver status word.
= (int)(GPSStatus[5] & 0xFFFF);
GPS receiver status word
The GPS message receiver decodes the "Hb" position/timing messages sent by the GPS module. The information is made available as a status string. Its format is 'aaaaoooohhhhVVvvhhddntssrrvvvvvv'.
aaaa = latitude in mas -324,000,000..324,000,000 (-90°..+90°)
oooo = longitude in mas -648,000,000..648,000,000 (-180°..+180°)
hhhh = GPS height in cm -100,000..+1,800,000 (-1000..+18,000m)
VV = 3D speed in cm/s 0..51400 (0.0 to 514 m/s)
vv = 2D speed in cm/s 0..51400 (0.0 to 514 m/s)

hh = 2D heading 0..3599 tenths of degrees (0.0 to 359.9°)
dd =current DOP 0..999 (0.0 to 99.9 DOP)
n = number of visible satellites 0..12
t = number of tracked satellites 0..12
ss = receiver status
Bits 15-13 (msb):
111 = 3D Fix
110 = 2D Fix
101 = Propagate Mode
100 = Position Hold
011 = Acquiring Satellites
010 = Bad Geometry
001 = Reserved
000 = Reserved
Bits 12-11: Reserved
Bit 10: Receiver in narrow-band tracking mode (M12M only)
Bit 9: Fast Acquisition Position
Bit 8: Filter Reset To Raw GPS Solution
Bit 7: Cold Start (no almanac, almanac out of date)
Bit 6: Differential Fix
Bit 5: Position Lock
Bit 4: Autosurvey Mode
Bit 3: Insufficient Visible Satellites
Bits 2-1: Antenna Sense 00 = OK
01 = Overcurrent
10 = Undercurrent
11 = No bias voltage
Bit 0: Code Location
0 = EXTERNAL
1 = INTERNAL
rr = reserved
vvvvvv = ID tag 6 characters (0x20 to 0x7e), serial number by default

```
GPSReceiverStatusHex (string, "0xE001")
    GPS receiver status word in hex format.
    = hexstring16(GPSReceiverStatus);
GPSFix (string, "3D Fix")
    GPS fix message.
    switch ((GPSReceiverStatus >> 13) & 0x7) {
    case 7: return "3D Fix";
        case 6: return "2D Fix";
        case 5: return "Propagate Mode";
        case 4: return "Position Hold";
        case 3: return "Acquiring Satellites";
        case 2: return "Bad Geometry";
        default: return "Reserved";
    }
```

```
GPSNarrowBand (bool, "0")
    1 if GPS is in narrow band mode.
    = GPSReceiverStatus & 0x0400;
GPSAntennaOK (bool, "1")
    1 if GPS antenna works properly.
    = (GPSReceiverStatus & 0x0006) == 0;
GPSSerial (string, "JX0225")
    GPS serial number.
     =    char buf[8];
       memset (buf, 0, sizeof (buf));
       buf[0] = (GPSStatus[6] >> 8) & 0xFF;
       buf[1] = (GPSStatus[6]) & 0xFF;
       buf[2] = (GPSStatus[7] >> 24) & 0xFF;
       buf[3] = (GPSStatus[7] >> 16) & 0xFF;
       buf[4] = (GPSStatus[7] >> 8) & 0xFF;
       buf[5] = (GPSStatus[7]) & 0xFF;
       return string (buf);
```

1.4 Padding

Padding is 8 dwords of information that is reserved for future use.

1.5 Slave Information

As described above, Slave information consists of 32 dwords. 8 status & configuration info dwords and 24 Slave specific dwords. The status & configuration info dwords, called s0-s7, are the following:

s0 -- board id/revision
s1 -- board serial
s2 -- software id
s3 -- software revision
s4 -- GPS time
s5 -- address
s6 -- Status
s7 – CRC error

Or in more details:

-- board id/revision *(example)*
s0 <= X"070071B0";

Board ID encodes the DCC number of the module as well as the board revision, e.g. "0x070071B0" describes a revision B Slave module with DCC number D070071.

-- board serial *(currently not used)*
s1 <= X"00000000";
-- software id *(example)*
s2 <= X"070568B0";
-- software revision *(example)*
s3 <= X"00000000";

-- GPS time
s4 <= GPS;
-- address
s5 <= ADDR (see description for w5 in section 1.1);

-- Status
s6 (31 downto 16) <= VCXO control value (VCXO_CTR);
s6 (15 downto 8) <= DIP switch settings (SW(8 downto 1));
s6 (7 downto 6) <= Daughter board switch settings (SW(10 downto 9));
s6 (5) <= uplink LOS (LOS);
s6 (4 downto 1) <= synchronization error counter for uplink (ERRC);
s6 (0) <= uplink up and running (UP);

s7 <= X"000000" & CRCERR;

There are 24 Slave specific dwords. The first 23 are called A, B .., W, while the 24th is reserved for a CRC. Slave information is constructed the following way:

```
Slave_Information <= s0 & s1 & s2 & s3 & s4 & s5 & s6 & s7 &
                     A & B & C & D & E & F & G & H &
                     I & J & K & L & M & N & O & P &
                     Q & R & S & T & U & V & W & CRC;
```

A..W, being slave specific, have different values for different slave types. As mentioned previously, the defined Slave types are:

- DuoTone
- FanOut
- Timing Comparator
- XO locking

For DuoTone Slaves, A..W are currently unused.

For FanOut Slaves, the total 32 dwords sent uplink are

```
FanOut Slave Information <= w0 & w1 &  w2 &  w3 &  w4 &  w5 &  w6 &  w15 &
                            w7 & w8 & w9 & w10 & w11 & w12 & w13 & w14;
```

where w0..w15 is the same as the ones defined for Master / FanOut modules in section 1.1. This means that A -> w7, B -> w8, etc, while w0..w6 = s0..s6. The Master/FanOut w15 fills in the Slave s7 slot. The definition of s7 in the Slave is

s7[11]: GPS error encountered when decoding GPS status information
(1 if there was an error)
s7[10..8]: GPS error counter
s7[7..0]: CRC error counter for the uplink receiver

For Comparator Slaves the dword usage is the following:

A[31..7] <= unused;
A[6..0] <= BNC_UP[6..0];
B[31..0] <= BNC0_DIFF[31..0];
C[31..0] <= BNC1_DIFF[31..0];
D[31..0] <= BNC2_DIFF[31..0];
E[31..0] <= BNC3_DIFF[31..0];
F[31..0] <= BNC4_DIFF[31..0];
G[31..0] <= BNC5_DIFF[31..0];
H[31..0] <= BNC6_DIFF[31..0];

Here BNC_UP[i] is 1 if an external 1PPS source is connected to the i^{th} port, while BNCi_DIFF[31..0] is the time difference between the Comparator Slave's internal 1PPS and the 1PPS received through the i^{th} port in units of 2^{-26} s.

Slave specific information for XOLocking Slaves are:

A: Preset frequency in Hz
B: Measured oscillator frequency
C: oscillator phase error in units of 2^-32 sec
D: oscillator control and status
D[17]: oscillator is locked
D[16]: oscillator is present
D[15..0]: oscillator control value (volts = cts * 5 / 32768)

1.6 CRC

The 32 bit CRC (cyclic redundancy check) value is calculated as it is done for the Ethernet standard.

2. Structure of the Preprocessed Output Data: Overview

Internal parameters of the OTD are designed to accurately group all necessary information that is to be communicated to the consumer, user and/or developer. The DMT computer interface of the OTD redistributes this information in order to make it easy to decipher, focusing on the requirements from the user's perspective. In this section we review the structure of the preprocessed data that appears on the output in every second in the form of an XML file.

Since the structure of the OTD (i.e. the architecture of the network of Master, FanOut and Slave modules) is not fixed, the structure of the data transferred to the DMT computer also needs to be dynamic. The present data structure implementation can also dynamically handle any structural changes, e.g. it adapts to changes without the need to restart the system.

Data communicated to the DMT computer is received and preprocessed by a DMT code. Data is accessible by the user in the following structure (see Figure 1).

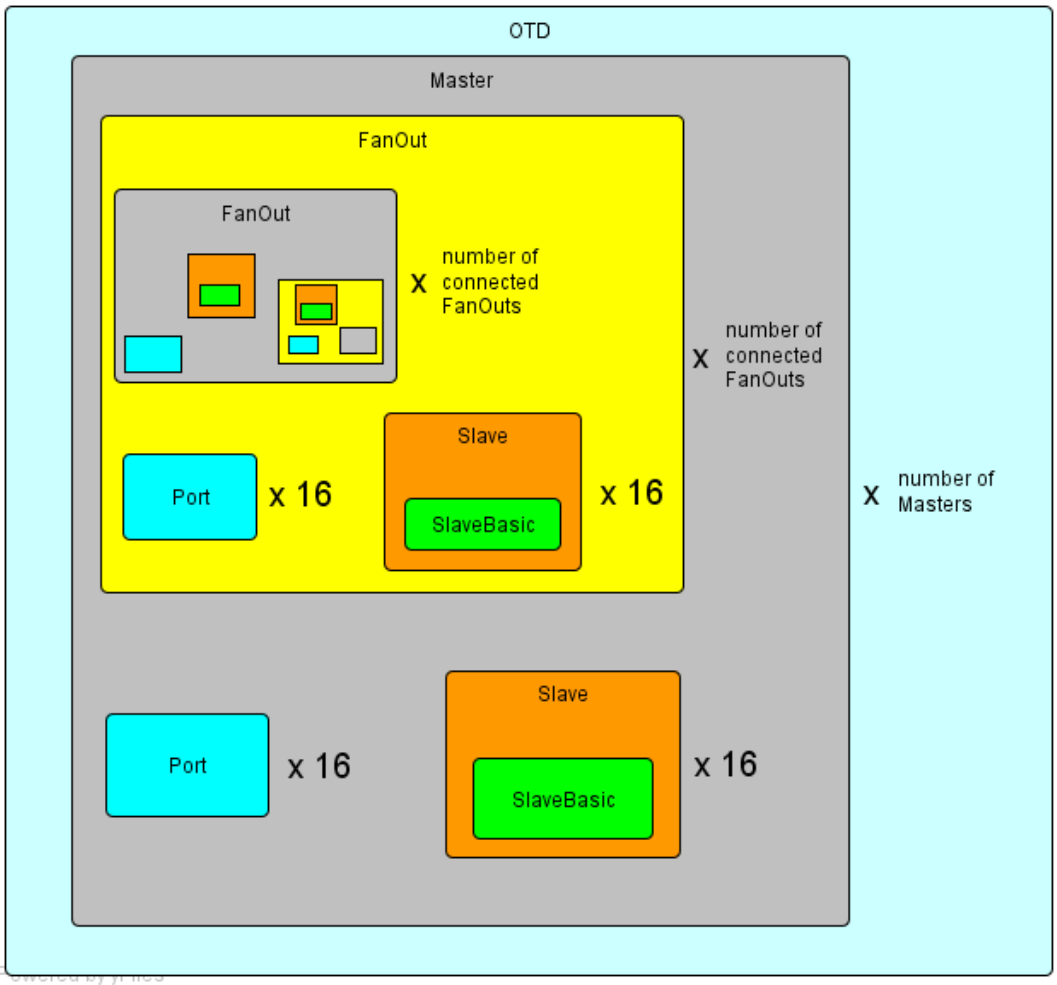

Figure 1 Diagram showing the data structure of the DMT code output.

The data structure, similarly to the structure of the OTD, is based on a hierarchical, treelike structure. Data from an OTD module (i.e. a Master, FanOut or Slave) is represented as a data unit, while other modules connected to the OTD module are represented by data subunits. As shown on Figure 1, the main data unit represents the OTD itself. This has data subunits representing each Master. Data units representing a Master contain subunits for each FanOut and Slave connected to them, as well as subunits for every port (containing synchronization information). While there are 16 Slave and Port subunits independently of the number of connected modules, the number of FanOut subunits is equal to the number of connected modules. Note that the data units for Masters and FanOuts are identical, i.e. they contain identical parameters; the difference is only in the values assigned to these parameters. This identity mirrors the identity of Master and FanOut modules, the only difference being the presence of an OCXO and consequently the usage of the modules.

Data units represent modules and parts of the OTD system, as follows:

OTD - represents the OTD timing system, containing the data units of each Master present.
Master / FanOut - represents a Master or a FanOut module (it also contains information on which module it represents). Besides containing the information on the module, it also contains the data modules of each FanOut or Slave connected to the given Master/FanOut, as well as a data unit for every connected port (containing synchronization information).
Slave - represents a Slave module. Contains information that is specific to Slave type, i.e. FanOut[1], Slave, DuoTone and XO locking. It also contains a data subunit SlaveBasic.
SlaveBasic - contains the information of a Slave module that is independent of the Slave type, i.e. basic information.

In the following we describe the preprocessed output data of the OTD in detail, as they are grouped into data units discussed above. Each data unit will be described separately, as their combination depends on the specific architecture of the OTD.

3. Structure of the Preprocessed Output Data: Detailed Structure

3.1. Master / FanOut

Master[i] / FanOut[i]
Header. Master for Master modules and FanOut for FanOut modules. "i" is the serial number of the module; e.g. FanOut [4] corresponds to the 4th FanOut for the given Master.
Module (string, "LVEA1")
Name of Master / FanOut module.
// GPS time & address----------
GPS (GPS, "917381733.0")
GPS time.

1 Note that FanOut boards are also Slave boards from this perspective; therefore a FanOut board will have two subunits, one for its FanOut purposes and one for its Slave purposes (i.e. locking to another FanOut or a Master).

GPSUTC (ISO-8601, "2009-01-30 20:15:19")
GPS time in ISO-8601 format.
Address (int, "0")
Master / FanOut address. Addresses are assigned automatically. A master node has always the address 0x00000000. Each of its branches gets assigned addresses 0x10000000, 0x11000000, 0x12000000, ... 0x1F000000 in order of the fanout ports. If a fanout is connected to port 5, its branches get assigned the addresses 0x25000000, 0x25100000, 0x25200000, ... 0x25F00000. The first nibble of each address represents the nesting level. Each following nibble encodes a port number until a final slave is reached.
AddressNtuple (int" Dim="8, "0 0 0 0 0 0 0 0")
Master / FanOut address in 4-tuple format.
// Board & software----------
Board (int, "117445040")
Master / FanOut board ID. Board ID, in its Hex format, encodes the DCC number of the Master / FanOut as well as the board revision. "0x070011B0" describes a revision B Master / FanOut with DCC number D070011.
BoardHex (string, "0x070011B0")
Master / FanOut board ID in hex format.
= hexstring(Board);
Serial (int, "0")
Master / FanOut board serial number. Currently not used.
Program (int, "134382752")
Master / FanOut software ID.
ProgramHex (string, "0x080284A0")
Master / FanOut software ID in hex format.
= hexstring(Program);
Revision (int, "0")
Master / FanOut software revision number.
// Configuration ----------
Configuration (int, "64579")
Master / FanOut configuration parameters.
Configuration[31..16]: unused
Configuration[15]: an OCXO is present (OCXO_PRESENT).
Configuration[14]: external 1 PPS present (EXT1PPS_ON)
Configuration[13]: a GPS module is present (GPS_ON)
Configuration[12]: the GPS module is locked (GPS_LOCKED)
Configuration[11]: the OCXO is locked (OCXO_LOCKED)
Configuration[10]: external 1PPS is used (OCXO_PRESENT and EXT1PPS_ON)
Configuration[9]: 1PPS from GPS is used (OCXO_PRESENT and GPS_ON and not EXT1PPS_ON)
Configuration[8]: 1PPS signal from uplink is used (not OCXO_PRESENT and UP)
Configuration[7]: uplink up and running (UP)
Configuration[6]: uplink LOS (LOS)
Configuration[5..2]: number of fan-out ports (0 represents 16 ports)
Configuration[1]: this is a fan-out module
Configuration[0]: this is a master module (OCXO_PRESENT)

ConfigurationHex (string, "0x0000FC43")
Master / FanOut configuration in hex format.
= hexstring(mConf);
IsMaster (bool, "1")
is this a Master board?
= OCXO_PRESENT;
HasFanout (bool, "1")
are there any FanOut ports connected?
= Configuration[1];
Ports (int, "16")
number of FanOut ports.
= (int)(Configuration[5..2]) if HasFanout, otherwise 0.
HasExtPPS (bool, "1")
is external 1PPS present?
= EXT1PPS_ON;
HasOCXO (bool, "1")
is OCXO present?
= OCXO_PRESENT;
OCXOLocked (bool, "1")
is OCXO locked?
= OCXO_LOCKED;
HasGPS (bool, "1")
does the module have GPS board present?
=GPS_ON;
GPSLocked (bool, "1")
is GPS locked?
= GPS_LOCKED;
UseExtPPS (bool, "1")
is the module using external 1PPS to lock its internal clock?
= OCXO_PRESENT and EXT1PPS_ON;
UseGPSPPS (bool, "0")
is the module using its GPS 1PPS to lock its internal clock?
= OCXO_PRESENT and GPS_ON and not EXT1PPS_ON;
UseUplinkPPS (bool, "0")
is the module using Uplink 1PPS ti lock its internal clock?
= not OCXO_PRESENT and UP;

// Status ----------

Status (int, "2028339006")
Master / FanOut status information.
Status[31..16] = VCXO_CTRL;
Status[15..8] = SW;
Status[7..6] = '00';
Status[5] = LOS;
Status[4..1] = ERRC;
Status[0] = UP;
StatusHex (string, "0x78E5FF3E")

Master / FanOut status information in hex format.
= hexstring(mStatus);
Up (bool, "0")
1 if the module is Up and Running.
= UP;
LOS (bool, "0")
1 if the module experiences loss of signal (LOS) in its Uplink port?
= LOS;
ErrorCount (int, "15")
Error counter. Counts the number of consecutive errors every second. Resets if there is no error.
= ERRC;
DIP[1..10] (bool, "1 1 1 1 1 1 1 1 0 0")
DIP switch status.DIP settings (1 through 8) for board; 9 and 10 for daughter board.
VCXOControl (double, "2.36122")
VCXO control voltage. Unit: V.
= VCXO_CTRL * 2.5 / 32768.;
OCXOControl (double, "0.257568")
OCXO control voltage. Unit: V.
= OCXO_CTRL * 10. / 32768. - 10.0;
OCXOErorr (double, "0")
OCXO error compared to reference 1PPS. Unit: us.
= OCXO_ERR / 4.294967296E3;
ExtPPSDelay (double, "0")
external 1PPS delay compared to internal clock. Unit: us.
= EXT1PPS / 4.294967296E3;
UplinkDelay (double, "-1.90735")
Uplink 1PPS delay compared to internal clock. Unit: us
= UP1PPS / 4.294967296E3;
GPSDelay (double, "461407")
GPS 1PPS delay compared to internal clock. Unit: us.
= GPS1PPS / 4.294967296E3;
// FanOut status ----------
FanoutUp[1..16] (bool, "0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0")
1 for FanOuts Up and Running?
= FOUP;
FanoutLOS[1..16] (bool, "1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1")
1 for FanOuts that have loss of signal (LOS)?
= FOLOS;
FanoutMissingDelay[1..16] (bool, "1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1")
1 for missing FanOuts.
= FOMISSING;
FanoutDelayError[1..16] (bool, "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0")
1 for FanOuts that have delay error (delay is unphysically large: > 1 ms).
= FODLYERR;
W15 (int, "0")

Currently not used;

Port[1..16] (data subunit)

Port data subunit. See section Port.

GPSStatus[1..8] (int, "167238453 -429866030 16392 1638401 100532248 168353793 19032 808596021")

GPS status message.

// the following GPS information (until "//end") is written to the output ONLY if the module is a Master

Latitude (double, "46.4551")

GPS latitude. Unit: degrees (-90 to +90).

= (double)((int)GPSStatus[0]) / 3.6E6;

Longitude (double, "-119.407")

GPS longitude. Unit: degrees (-180 to +180) .

= (double)((int)GPSStatus[1]) / 3.6E6;

Height (double, "163.92")

GPS height. Unit: m (-1000 to +18000).

= (double)((int)GPSStatus[2]) / 100.0;

Speed3D (double, "0.25")

GPS 3D speed. Unit m/s (0.0 to +514 m/s).

= (double)(GPSStatus[3] >> 16) / 100.0;

Speed2D (double, "0.01")

GPS 2D speed. Unit: m/s (0.0 to +514 m/s).

= (double)(GPSStatus[3] & 0xFFFF) / 100.0;

Heading (double, "153.4")

GPS heading. Unit: degree (0 to 360).

= (double)(GPSStatus[4] >> 16) / 10.0;

GPSDOP (double, "0.24")

GPS Dilution of Precision (0 to 99.9).

= (double)(GPSStatus[4] & 0xFFFF) / 100.0;

GPSSatellitesVisible (int, "10")

GPS number of visible satellites (0 to 99).

= (int)(GPSStatus[5] >> 24);

GPSSatellitesTracking (int, "8")

GPS number of tracked satellites (0 to 99).

= (int)((GPSStatus[5] >> 16) & 0xFF);

GPSReceiverStatus (int, "57345")

GPS receiver status word.

= (int)(GPSStatus[5] & 0xFFFF);

GPSReceiverStatusHex (string, "0xE001")

GPS receiver status word in hex format.

= hexstring16(GPSReceiverStatus);

GPSFix (string, "3D Fix")

GPS fix message.

GPSNarrowBand (bool, "0")

1 if GPS is in narrow band mode.

= GPSReceiverStatus & 0x0400;

GPSAntennaOK (bool, "1")
1 if GPS antenna works properly.
= (GPSReceiverStatus & 0x0006) == 0;
GPSSerial (string, "JX0225")
GPS serial number.
//end----------
Extended[1..8] (int, "0 0 0 0 0 0 0 0")
Reserved for future use, currently not used.
CRC (int, "295080708")
Master / FanOut CRC value;
// List of Slave modules----------
Slave[1..16] (data subunit)
Slave data subunit. See section Slave.
// FanOut modules connected to this Master / FanOut module ----------
FanOut[1..16] (data subunit)
FanOut data subunit. See section Master / FanOut.

3.2 Port

Port[i] (header)
Header. "i" is the serial number of the port; e.g. Port[4] corresponds to the 4th port for the given Master / FanOut.
//FanOut port status ----------
Up (bool, "0")
1 if uplink connection is up and running.
= UP;
LOS (bool, "1")
1 if the uplink connection has loss of signal (LOS).
= LOS;
MissingDelay (bool, "1")
1 if port is missing.
= (DelayControl & 0x08) != 0;
DelayError (bool, "0")
1 if delay is unphysically large (> 1 ms).
= (DelayControl & 0x04) != 0;
ErrorCount (int, "70")
Error counter. Counts the number of consecutive errors every second. Resets if there is no error.
= ERRC;
Delay (double, "0")
1PPS delay time on this port. Unit: us.
= (double)DelayMeasured / 4.294967296E3;
Advance (double, "0")
Delay time measured on the port. Unit: us.
= (double)(DelayControl >> 14) / 268.435456E6;

UsedAdvance (double, "0")
Actual delay time of the arrival of the 1PPS to the external device, calculated from the measured delay(Advance, see above). This is the time that will be used to offset the downlink 1PPS to synchronize modules. Unit: us.
= (double)(((DelayControl >> 14) + 2) >> 2) / 67.108864E6;
End
Trailer.

3.3 Slave

Slave[i] (header)
Header. "i" is the serial number of the Slave; e.g. Slave[4] corresponds to the Slave connected the the 4th port of this Master / FanOut.
CRCOK (bool, "0")
1 if Slave CRC is OK.
= (CRC == 0);
// Basic Slave parameters ----------
SlaveBasic (data subunit)
Basic Slave data subunit. See section SlaveBasic.
Extended[1..24] (int, "32 -64 -64 -64 -64 -64 128 -64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 108775799")
Slave specific data in one integer array. Slave specific information is also listed in the XML file in parts (see the following parameters).
//Slave specific information – depending on the type of the Slave module, one of the following data
//will be written to the output:
IF the Slave is FanOut board:
Type (string, "Fanout")
= "Fanout";
IF the Slave is DuoTone:
Type (string, "DuoTone")
= "DuoTone";
IF the Slave is Comparator:
Type (string, "Comparator")
= "Comparator";
HasExtPPS[1..8] (bool, "0 0 0 0 0 1 0");
1PPS presence on the 8 inputs of the comparator module
if ((i >= 0) && (i < 7)) {return mExtended[0] & (1 << i); }
else {return 0.0;}
ExtPPSDelay[1..8] (double, "0.0298023");
delay times for each external 1PPS.Unit: us. Each output is preceded by the text "ExtPPSDelay[i]", where i is the serial number of the input connector the delay corresponds to, therefore every element of this parameter array is written in separate lines. Below, we show the numerical values of the parameter array.
if ((i >= 0) && (i < 7)) {return (double)(int32_t)mExtended[i+1] / 4.294967296E3;}
else {return 0.0;}
IF the Slave is XOLocking:

Type (string, "XOLocking")
= "XOLocking";
HasOCXO (bool, "0");
1 if XO locking is present.
= mExtended[3] & 0x10000;
OCXOLocked (bool);
XO locked? = mExtended[3] & 0x20000;
OCXOError (double);
XO locking error. Unit: us.
= (double)mOCXOError / 4.294967296E3;
OCXOControl (double);
XO locking control voltage. Unit: V.
= (double)(mExtended[3] & 0x0FFFF) * 5. / 32768.;
SetFrequency (double);
set frequency of XO locking. Unit: Hz.
= (double)mExtended[0];
OCXOFrequency (double);
measured XO locking frequency. Unit: Hz.
= (double)mExtended[1];

IF the slave is none of the above:

Type (string, "Unknown")
output SlaveType = "Unknown";

3.4 SlaveBasic

GPS (GPS, "904189277.0")
GPS time.
GPSUTC (ISO-8601, "1980-01-06 00:00:00")
GPS time in ISO-8601 format.
Address (int, "609222656")
Address of the Slave module. Addresses are assigned automatically. A master node has always the address 0x00000000. Each of its branches gets assigned addresses 0x10000000, 0x11000000, 0x12000000, ... 0x1F000000 in order of the fanout ports. If a fanout is connected to port 5, its branches get assigned the addresses 0x25000000, 0x25100000, 0x25200000, ... 0x25F00000. The first nibble of each address represents the nesting level. Each following nibble encodes a port number until a final slave is reached.
AddressNtuple[1..8] (int, "2 4 5 0 0 0 0 0")
Address of the Slave module in 4-tuple format.
Board (int, "117469616")
Master / FanOut board ID. Board ID, in its Hex format, encodes the DCC number of the module as well as the board revision, e.g. "0x070071B0" describes a revision B Slave with DCC number D070071.
BoardHex (string, "0x070071B0")
Master / FanOut board ID in hex format.
= hexstring(mBoardId);

Serial (int, "0")
Slave board serial number (currently not used).
Program (int, "117794992")
This encodes the daughter board used by the Slave. It also determines the FPGA code which needs to be loaded onto the Slave. For example, "0x080665A0" describes a revision A XO locking slave daughter board with DCC number D080665. Other, currently supported Slaves are D080335 (DuoTone), D070568 (Comparator) and D070011 (Fanout, when seen through the uplink interface).
ProgramHex (string, "0x070568B0")
Slave module software ID in hex format.
= hexstring(Program);
Revision (int, "0")
Slave module software revision number.
Status (int, "2073788801")
Slave status information.
Status[31..16] = VCXO control value (VCXO_CTR);
Status[15..8] = DIP switch settings (SW(8 downto 1));
Status[7..6] = Daughter board switch settings (SW(10 downto 9));
Status[5] = uplink LOS (LOS);
Status[4..1] = synchronization error counter for uplink (ERRC);
Status[0] = uplink up and running (UP);
StatusHex (string, "0x78E5FF3E")
Slave module status information in hex format.
= hexstring(Status);
LOS (bool, "0")
1 if the module experiences loss of signal (LOS) in its Uplink port.
= LOS;
Up (bool, "1")
1 if the module is Up and Running.
= UP
ErrorCount (int, "0")
Error counter. Counts the number of consecutive errors every second. Resets if there is no error.
= ERRC;
DIP[1..10] (bool, "1 0 0 0 0 0 0 1 0 1")
DIP switch status.DIP settings (1 through 8) for board; 9 and 10 for daughter board.
VCXOControl (double, "2.41417")
VCXO control voltage. Unit: V.
= VCXO_CTRL * 2.5 / 32768.;
Configuration (int, "10")
Slave module configuration parameters.
Configuration[31..16]: VCXO control value.
Configuration[15..8]: DIP switch settings.
Configuration [7..6]: Daughter board switch settings.
Configuration[3..1]: synchronization error counter for uplink.
Configuration[0]: local clock is in synchronization.

ConfigurationHex (string, "0x0000000A")
Slave configuration parameters in hex string format.
= hexstring(Configuration);
CRCErrorCount (int, "10")
CRC error counter.