DGT Electronic Chess Boards

Description of the communication protocol[1]

Author(s): Lucas van der Ploeg Signature: _____

Harry Roewen Signature: _____

Revision date: October 25th, 2019

Document approval: Signature: _____

Document release: Signature: _____

Product number: 72002 Document version: P1.0

CHANGE LOG			
Date (ddmmyyyy)	**Version**	**Description of change(s)**	**Author**
25-10-2019	1.0	Initial document, based on the various existing technical documents	HRO

REFERENCES	
[document ID]	**Title / Version (optional additional information)**

ABBREVIATIONS AND DEFINITIONS	
Abbr./Def.	**Description**
BCD	Binary Coded Decimal
DGT	Digital Game Technology
LSB	Least Significant Bit
MSB	Most Significant Bit
NA	Not applicable
TBD	To Be Discussed
TBS	To Be Specified

1.1.1 Index

2 Main functionality of the DGT Electronic Chess Board

The DGT e-Board is basically a sensor which senses the presence of the special chess set pieces on the squares of the board. The situation on the board is measured and can be communicated with an average maximum time delay of 200 milliseconds

Besides this detection function, the board communicates with an optional DGT clock (supported types see chapter 3), to give the data of the clock available to the general interface.

The board carries a 8 kB cyclic non-volatile memory, in which all position changes and clock information is stored during all power-up time. This data can be read and processed. (see chapter 19)

The board supports two methods of communication:

- For single-board situations a protocol for communication between one board and one computer is available.
- For situations with many boards a network communications protocol is available, where many boards can be connected in a bus structure. A separate communication protocol is available for this bus structure.

3 Communication interfaces

3.1 Board to computer

- Serial RS232, using 9600 baud, 8 bits, no parity, 1 start bit, 1 stop bit, no software or hardware handshaking
    - DGT e-Board serial
- USB2.0/connector type-B mini, using CDC (Communication Device Class)
    - DGT e-Board USB
- USB2.0/connector Type-C, using CDC
    - DGT SmartBoard
    - DGT e-Board USB-c
- Bluetooth, using RFCOMM (Radio Frequency Communication)
    - DGT Bluetooth e-Board

3.2 Board to clock

- I^2C, using multi master I^2C, both for communication with the DGT clocks as with a computer (see chapter 20).
    - DGT clocks with I^2C:
        - DGT3000
        - DGT-pi
- SBI, communication with old DGT clocks using a protocol based on the NEC SBI protocol (see chapter 21)
    - DGT clocks with SBI
        - DGT topmatch
        - DGT XL

4 Board Identification

4.1 version number

The boards have version numbers of which the first byte determines the board hardware, the second byte the software version.

- Revelation II has version number 0.60
- Serial e-Boards have version numbers up to 1.9
- USB e-Boards have version numbers up to 1.8 (no magic piece detection)
- Bluetooth e-Boards have version numbers up to 3.01
- Universal e-Boards have version number up to 4.x
- Centaurs have version number 5.x

4.2 serial number

Each board should have a unique 5-byte ASCII serial number.

- Serial e-Boards start at 0
- USB e-Boards start at 10000 (the serial boards have also passed the 10000)
- Bluetooth Boards start at 20000
- Universal e-Boards start at 40000
- Revelation and Centaur have a fixed serial number (not unique)

Bluetooth Boards and Universal e-Boards have a 10-byte ASCII serial number:

- Bluetooth Boards have 00000 added in front of the short serial number.
- Universal e-Boards have the complete production code as serial number.

4.3 Bus address

the bus address is derived from the last 5 numbers of the serial number and can be changed by a bus command in case of duplicate addresses.

In case the bus address has changed, this new address will be used.

4.4 Identification string

The boards also carry an identification which could look like:

```
"DGT Projects - This DGT board is produced by DGT Projects.\n
 DGT Projects is a registered trade mark.\n
 220798 ISP/bus/8KP/8KE/P6/Fritz5 Vs 1.00. Serial nr. 00137 1.0"
```

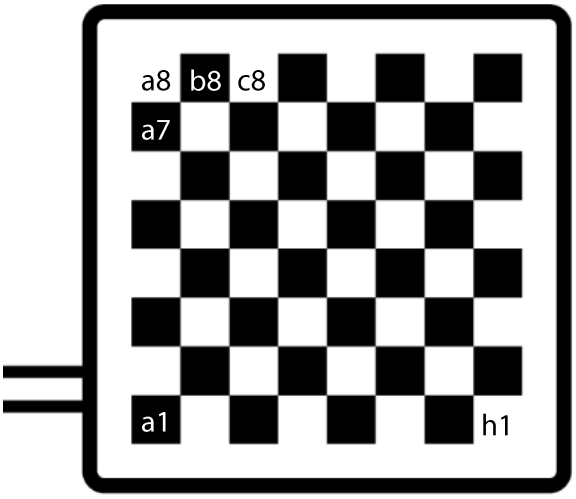
5 DGT field and piece definitions

5.1 DGT field definition

Board fields are numbered from 0 to 63, row by row, in normal reading sequence. When the connector is on the left hand, counting starts at the top left square. The board itself does not rotate the numbering, when black instead of white plays with the clock/connector on the left hand.

In non-rotated board use, the field numbering is as follows:

- Field a8 is numbered 0 (0x00)
- Field b8 is numbered 1 (0x01)
- Field c8 is numbered 2 (0x02)
- ..
- Field a7 is numbered 8 (0x08)
- ..
- Field h1 is numbered 63 (0x3f)



For digital DGT boards, field a1 is the black corner field closest to the connector (a1), which is number 56 (0x38).

The piece identification (numbering) can be found at the end of this document.

5.2 DGT piece definition

Codes for chess pieces:

- #define EMPTY 0x00; no piece
- #define WPAWN 0x01; white pawn
- #define WROOK 0x02; white rook
- #define WKNIGHT 0x03; white knight
- #define WBISHOP 0x04; white bishop
- #define WKING 0x05; white king
- #define WQUEEN 0x06; white queen
- #define BPAWN 0x07; black pawn
- #define BROOK 0x08; black rook
- #define BKNIGHT 0x09; black knight
- #define BBISHOP 0x0a; black bishop
- #define BKING 0x0b; black king
- #define BQUEEN 0x0c; black queen

Codes for special pieces:

- #define PIECE1 0x0d; Special piece: Draw
- #define PIECE2 0x0e; Special piece: White wins
- #define PIECE3 0x0f; Special piece: Black wins

6 Electronic board modes

The main function of the board is to transfer piece position- and, when available, clock information.

For this, seven board modes are available:

e-Board mode	Behavior
IDLE	When a 'single board' command is received, this will become the board mode. When the board was in one of the update states, this behavior is cancelled.
UPDATE_BOARD	At the moment that the board detects a removal, change or placement of a piece, it outputs a DGT_MSG_FIELD_UPDATE message.
UPDATE_SBI	e-Board with a DGT clock (SBI) As UPDATE_BOARD, where additional the clock data are send regularly (at least every second) using message DGT_MSG_SBI_CLOCK
UPDATE_SBI_NICE	As UPDATE_SBI, but clock data is only sent when it has changed.
UPDATE_I2C	e-Board with a DGT clock (I^2C) As UPDATE_BOARD, where additional the clock data are send regularly (at least every 2 seconds) using message DGT_MSG_I2C_CLOCK
UPDATE_I2C_NICE	As UPDATE_I2C, but clock data is only sent when it has changed
BUS	When a 'bus' command is received, this will become the board mode. When the board was in one of the update states, this behavior is cancelled.

Note 1: The regular 'single board' commands don't affect the current board mode. E.g. when in UPDATE-BOARD mode, it continues sending DGT_MSG_FIELD_UPDATE messages.

Note 2: Each bus command switches the board to bus mode. Each single board command switches the board to single board mode. When the board is not in the correct mode when a command arrives this first command is ignored by boards with version less than 4.x

Note 3: After power-up or reset, the board accepts both 'single board' and 'bus' commands. For boards with version less than 4.x, the 'bus' commands are ignored after the first 'single board' command.

7 Notes on the communication dynamics

The squares of the board are sampled one by one, where a full scan takes about 200-250 ms. Due to this scan time, it can be, that a piece that is moved from square e.g. A2 to A3 can be seen on both squares, in the same scan. On a next scan of course, the old position is not seen anymore.

When in UPDATE mode, this means, that the information on changes on the squares can come in opposite sequence: first the new occurrence is reported, then the clearing of the old position is sent.

When a piece X on C4 is taken by piece Y on B3, it can be that the following changes are reported:

- A on C4
- Empty on B3
- (and Empty on C4 is never seen)

Another extreme situation can occur e.g. when a queen on C1 takes a pawn on C4. The reported changes can be (in sequence):

- Empty on C4 (the pawn is taken by one hand)
- Queen on C2
- Queen on C3
- Empty on C1
- Empty on C2
- Queen on C4
- Empty on C3

For writing software it is important to take these dynamics into account.

Some effort needs to be made to reconstruct the actual moves of the pieces.

8 Communication protocol – general

8.1 7-bit definition

Since MSB (bit 8) is used for interpretation of the commands, the data and values in the message must be represented in a 7-bit notation.

8.1.1 2 byte value notation and functional definition

Notation of a 2 byte (decimal) value, used in the message format:

- Byte 1: value << 7
- Byte 2: value & 0x7f

Functional definition:

- For byte 1 (above):
  #define LLH_SEVEN(a) ((char) ((a & 0x3F80)>>7)) 0011 1111 1000 0000
- For byte 2 (above):
  #define LLL_SEVEN(a) ((char) (a & 0x007f)) 0000 0000 0111 1111

Note: The maximum value represented by 14-bits is 2^{14} (16.384)

8.1.2 Message- /data size

The size of a message or used data is represented in 14-bits, using 2 bytes.

[Example 1] When the size is “67” (decimal), which is “1000011” (binary) the size in the message is defined by:

- Byte 1: 0000 0000 (0, 0x00)
- Byte 2: 0100 0011 (67, 0x63)

[Example 2] When the size is “2345” (decimal), which is “100100101001” (binary) the size in the message is defined by:

- Byte 1: 0001 0010 (18, 0x12)
- Byte 2: 1010 1001 (169, 0xa9)

8.1.3 Bus-address

When using the bus protocol, where addresses are required, the addresses are represented in 14-bits, using 2 bytes. This is similar to the definitions and examples above.

9 Communication protocol – Single board

In single board mode the board has a small incoming commands buffer. Please pay attention on the command-load to prevent data overload or false replies.

9.1 single board commands without data (computer -> board)

message:

- Message is always 1 byte long
- Command code: bit 8 (0x80) and bit 6 (0x20) are not set
- Command code range:
  - 0x00-0x1f (not currently used)
  - 0x40-0x5f

message format:

- byte 1: command code

9.2 single board commands including data (computer -> board)

message:

- Message is at least 4 bytes, max 129 bytes long
- Command code: bit 8 (0x80) is not set, bit 6 (0x20) is set
- Command code range:
  - 0x20-0x3f
  - 0x60-0x7f
- The second and third byte determine the size. The size is max 127 (decimal)
- Size: number of data-bytes. This is excluding the header (byte 1-2), including the last byte of the data which is always zero (0x00)

message format:

- byte 1: command code
- byte 2: size << 7
- byte 3-(n+2): data
  Note: *n = size, byte n+2 is always zero (0x00)*

9.3 single board messages (board -> computer)

message:

- Messages from board to computer are always a reply on a message from the computer.
- The messages code from the board always start with the MSB set (>0x80).
- Message code range:
  - 0x80-0xff
  - Range is identical for both bus and single board messages. Identical message code with different message formats/contents.
- The second and third byte determine the size (max 2^{14} = 16.384). The size is including the header of the message (byte 1-3)
- The data bytes always have the MSB not set (<0x80).

message format:

- byte 1: message code
- byte 2: size << 7
- byte 3: size & 0x7f
- byte 4-n: data
  Note: *n = size*

10 Communication protocol - Bus

For most e-Boards, the bus mode has only a one-command incoming commands buffer. Please take this in account to prevent data overload or false replies.

Note: When using RS232 in bus mode, the RS232 input and RS232 output are connected to all boards. The RS232 output of the board is configured as a pull-up driver, with a pull-down resistor on the RS232 pull-up line. This must be used, so all boards receive all commands from the computer, and can all send data to the computer.

10.1 Checksum

The checksum used in the bus-protocol:

- Sum of all bytes in the message & 0x7f (7-bits notation)

10.2 bus commands without data (computer -> board)

message:

- Message is always 4 bytes
- Command code: bit 8 (0x80) is set, bit 6 (0x20) is not set.
- Command code range:
  - 0x80-0x9f
  - 0xc0-0xdf (not currently used)
- The second and third byte determine the bus address.
- Byte four is the checksum.

message format:

- byte 1: command code
- byte 2: bus address << 7
- byte 3: bus address & 0x7f
- byte 4: checksum

10.3 bus commands including data (computer -> board)

message:

- Message is at least 6 bytes max 16.384
- Command code always with the MSB set (>0x80).
- Command code range:
  - 0xa0-0xbf
  - 0xe0-0xff (only for debug)
- The second and third byte determine the size. The size is including the header: byte 1-5
- The fourth and fifth byte determine the bus address.
- The data bytes always have the MSB not set (<0x80).
- The last byte is the checksum.

message format:

- byte 1: command code
- byte 2: size << 7
- byte 3: size & 0x7f
- byte 4: bus address << 7
- byte 5: bus address & 0x7f
- byte 6-(n-1): data
- byte n: checksum
  Note: *n = size*

10.4 bus messages (board -> computer)

message:

- Messages from board to computer are always a reply on a message from the computer.
- The messages code from the board always start with the MSB set (>0x80)
- Message code range:
  - 0x80-0xff
  - Range is identical for both bus and single board messages. Identical message code with different message formats/contents.
- The second and third byte determine the size (max 2^{14} = 16.384). The size is including the header: byte 1-5
- The fourth and fifth byte determine the bus address.
- The data bytes always have the MSB not set (<0x80).
- The last byte is the checksum

message format:

- byte 1: message code
- byte 2: size << 7
- byte 3: size & 0x7f
- byte 4: bus address << 7
- byte 5: bus address & 0x7f
- byte 6-(n-1): data
- byte n: checksum
  Note: *n = size*

11 Single board use

11.1 General

All single board commands and messages mentioned in this chapter can be found in the chapters 13, 14 and 15.

Specific clock commands and messages can be found in the chapters 20 and 21.

11.2 Setup communication

For single board use only, the board must be connected to a computer. A clock can be connected to the board.

The types of communication are described in chapter 3.

Drivers for serial, USB, Bluetooth, etc. should be available at the computer. The board will be recognized by these drivers. After activating the communication device, the low-level communication is active.

Some commands are available to validate the board-communication:

- DGT_REQ_SERIALNR
- DGT_REQ_LONG_SERIALNR
- DGT_REQ_VERSION
- DGT_REQ_HARDWARE_VERSION
- DGT_REQ_TRADEMARK
- DGT_REQ_BUSADDRESS

Note: Since there are many different versions of the boards, identification of the board is essential to determine which functionality is available. See tables at chapter 22.

11.3 Setup of the e-Board

The board can be forced in the basic start situation using the command:

- DGT_REQ_RESET

Pushing messages from the board based on an event e.g. a move of a piece, can be done using one of the commands:

- Board only:
  - DGT_REQ_UPDATE_BOARD
- Board and clock:
  - DGT_REQ_UPDATE_SBI
  - DGT_REQ_UPDATE_SBI_NICE
  - DGT_REQ_UPDATE_I2C
  - DGT_REQ_UPDATE_I2C_NICE

Sending the command DGT_REQ_RESET will stop pushing of messages.

11.4 Setup of the clock

When a clock is connected to the e-Board, it can be put in the required playing-mode, times can be changed, etc.

DGT supports two types of clocks: connected to the e-Board via an I²C or a SBI communication interface. Commands are based on the related I²C and SBI commands.

11.4.1 Setup of the clock – I²C

Short overview of commands that can be used to setup the clock, using the command DGT_I2C_CLOCK_MESSAGE:

Command name	Purpose
I2C_CURRENTPROGRAM	Request for the current active program of the clock
I2C_PROGRAM	Program the clock (change settings)
I2C_CHANGESTATE	Select the required option of the clock

Pushing messages from the clock based on an event e.g. change of a period, can be done using one of the commands:

- DGT_REQ_UPDATE_I2C
- DGT_REQ_UPDATE_I2C_NICE
- ***Note:*** These commands also start pushing messages from het board.

Sending the command DGT_REQ_RESET will stop pushing of messages.

11.4.2 Setup of the clock – SBI

Short overview of commands that can be used to setup the clock, using the command DGT_SBI_CLOCK_MESSAGE:

Command name	Purpose
DGT_CMD_CLOCK_SETNRUN	Set the clock and start running

Pushing messages from the clock based on an event e.g. change of a period, can be done using one of the commands:

- DGT_REQ_UPDATE_SBI
- DGT_REQ_UPDATE_SBI_NICE
- ***Note:*** These commands also start pushing messages from het board.

Sending the command DGT_REQ_RESET will stop pushing of messages.

11.5 Read data from the e-Board

11.5.1 Stored games

The e-Boards have a logfile which contains the data of played games. This data can be retrieved from the board using the command:

- DGT_REQ_LOG_MOVES

11.5.2 During a game

When the board is forced in one of the update modes (see chapter 6 and chapter 11.3) messages of the board will be pushed. No request is required.

The current state of all pieces on the board can be requested, even when in one of the update modes, using the command:

- DGT_REQ_BOARD

Note: The command DGT_REQ_LOG_MOVES can be used also during a game. This can be used to restore the current game after communication loss or similar situations.

11.6 Read data from the clock

11.6.1 During a game - general

When the board is forced in one of the update modes for clock information (see chapter 6 and chapter 11.4) messages of the clock will be pushed. No request is required.

The current state of the clock can be requested, even when in one of the update modes, using the following related I^2C and SBI commands:

- DGT_REQ_SBI_CLOCK
- DGT_REQ_I2C_CLOCK

Note 1: The command DGT_REQ_LOG_MOVES can be used also during a game. This can be used to restore the current game after communication loss or similar situations.

Note 2: Some specific I^2C and SBI commands can be used during the game; see chapters 20 and 21. Because the clock pushes all data to the connected e-Board, the use of these specific commands is not required.

11.7 Special functions

11.7.1 Status of the board

Beside piece information and movements at the board, other status of the board can be requested.

At this moment only the battery status is available. Use the command:

- DGT_REQ_BATTERY_STATUS

11.7.2 Status and commands of the clock using I²C

Short overview of commands that can be used for status information and of special commands for the clock, using the command DGT_I2C_CLOCK_MESSAGE:

Command name	Purpose
I2C_DISPLAY	Display characters at the display of the clock
I2C_END_DISPLAY	Stop displaying characters send by the previous message
I2C_SETNRUN	Start the game/clock
I2C_TIMECORRECTION	Adjust the time at the clock; arbiter functionality
I2C_SETCENTRALCONTROL	The clock can be controlled by the computer; only possible before start of the game
I2C_RELEASECENTRALCONTROL	End the control of the clock by the computer

11.7.3 Status and commands of the clock using SBI

Short overview of commands that can be used for status information and of special commands for the clock, using the command DGT_SBI_CLOCK_MESSAGE:

Command name	Purpose
DGT_CMD_CLOCK_DISPLAY	Put (7-segment) characters at the clock
DGT_CMD_CLOCK_ICONS	Control the icons of the clock
DGT_CMD_CLOCK_END	Clear the characters at the clock
DGT_CMD_CLOCK_BUTTON	Request for the button pressed
DGT_CMD_CLOCK_VERSION	Request for the version of the clock
DGT_CMD_CLOCK_SETNRUN	Set the clock and start running
DGT_CMD_CLOCK_BEEP	Turn on the beep at the clock
DGT_CMD_CLOCK_ASCII	Put characters at the clock (DGT3000 only)

11.7.4 Modify the use of the LEDs

The LED's of the "Revelation-II" can be controlled using the command:

- DGT_SET_LEDS

11.7.5 Switch to bus-mode

When multiple boards will be used in a network, the board has to be in the so called bus-mode. The command DGT_TO_BUSMODE will put the board in this mode.

Note: When a board receives a bus-command in single board mode, the board will change to bus-mode automatically.

11.8 Communication shutdown

T.b.d.

12 Multiple board use

12.1 General

Multiple board use means use of a network. Since a network of DGT boards is bus-based, the communication uses so called bus-commands.

All bus commands and messages mentioned in this chapter can be found in the chapters 16, 16.24 and 18.

Specific clock commands and messages can be found in the chapters 20 and 21.

12.1.1 Master – slave communication

All commands are initiated by the computer (master). The boards are in slave mode and respond on the commands.

12.1.2 Response time messages - board

The response time at the boards normally varies up to 20 milliseconds. The computer should use an acceptable time-out on sent messages. An indication of such a time-out is 80 milliseconds.

12.1.3 Response time messages - clock

Reading data from the clock (if any) and make it available for communication causes a delay of up to one (1) second between the time actual at the clock and the received data.

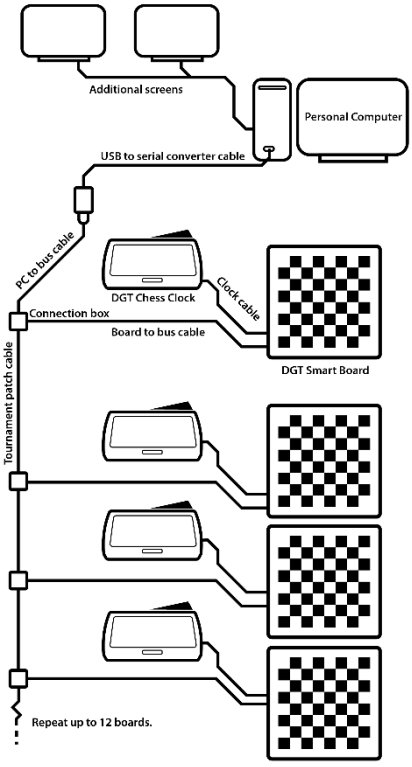
12.1.4 Checksum errors

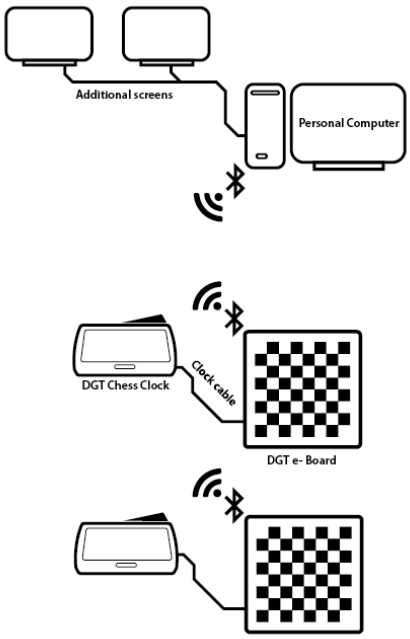
When a received checksum does not match, resend the command.

Note: Exception on the DGT_BUS_REQ_CHANGES. In that case use the command DGT_BUS_SREPEAT_CHANGES. See chapter 16.5.

12.2 Setup communication

For multiple board use, the boards can be connected wired and/or wireless. See pictures below.





The types of communication are described in chapter 3.

Drivers for serial, USB, Bluetooth, etc. should be available at the computer. The boards will be recognized by these drivers. After activating the communication device, the low-level communication is active.

12.2.1 Addressing of the boards

To be able to communicate with all connected boards, each board must have a unique address; the bus-address.

Find all connected boards

The first required action is to detect all boards in the network. For this the following commands are used:

- DGT_BUS_PING
- DGT_BUS_IGNORE_NEXT_BUS_PING
- DGT_BUS_RPING

Start with the following sequence:

1. Send DGT_BUS_PING with address zero (0); this is a broadcast message
2. Results in various valid replies. These boards are added to the list of connected boards
3. Send DGT_BUS_IGNORE_NEXT_BUS_PING to the detected boards from the list of connected boards
4. Repeat from the beginning (1.): repeat is done a few times until hardly any (or none) non-valid replies occur. This depends on the amount of boards connected

Note: When the bus-addresses of most of the connected boards are already known, the application can start with (3.) to avoid unnecessary sequences.

After this some invalid replies can remain. This might indicate some boards have the same bus-address. Therefore, the bus-address of these boards have to change.

Sequence to change the bus-address of boards:

1. Send DGT_BUS_RPING to a specific bus-address with address zero (0); this address has resulted in non-valid messages before.
2. Results in new bus-addresses of the addressed boards.
3. Send DGT_BUS_IGNORE_NEXT_BUS_PING to the detected boards from the list of connected boards
4. Send DGT_BUS_PING with address zero (0); this is a broadcast message
5. Results in various valid replies. These boards are added to the list of connected boards
6. Repeat sequence 3-5 and/or 1-5 until all boards are detected with a unique bus-address

Validate presence off all connected boards

In order to validate the presence of all connect boards, use the command DGT_BUS_PING with all bus-addresses in the list.

12.2.2 Identification of the boards

Since there are many different versions of the boards, identification of the board is essential to determine which functionality is available. See tables at chapter 23.

Some commands are available to identify the boards:

- DGT_BUS_REQ_SERIALNR
- DGT_BUS_REQ_VERSION
- DGT_BUS_REQ_HARDWARE_VERSION
- DGT_BUS_REQ_TRADEMARK

12.3 Setup the e-Board

12.3.1 Set board in start situation

The board can be forced in the basic start situation using the command:

- DGT_BUS_RESET

12.3.2 Configure piece detection of the board

The quality of the piece detection depends on the timing of the sensor in the board. This can be modified using the command:

- DGT_BUS_CONFIG

Note: This command is also used to determine the current configuration of the e-Board

12.4 Setup of the clock

When a clock is connected to the e-Board, it can be put in the required playing-mode, times can be changed, etc.

Pushing messages from the clock based on an event e.g. change of a period, cannot be done in bus mode.

DGT supports two types of clocks: with an I^2C or SBI communication interface.

12.4.1 Setup of the clock – I^2C

Short overview of specific I^2C commands that can be used to setup the clock, using the command DGT_BUS_I2C_CLOCK_MESSAGE. Overview of the commands, see chapter 11.4.1.

12.4.2 Setup of the clock – SBI

Short overview of specific SBI commands that can be used to setup the clock, using the command DGT_BUS_SBI_CLOCK_MESSAGE. Overview of the commands, see chapter 11.4.2.

12.5 Start of an event (tournament)

When all boards have pieces in the starting position send the command:

- DGT_BUS_SET_START_GAME

This results in a marker in the logfile at the boards, from which the match-data can be retrieved.

Also read all the clock-times from all boards, using one of the following commands, depending on the type of clock attached to the board (see chapter 3.2):

- DGT_BUS_REQ_SBI_CLOCK
- DGT_BUS_REQ_I2C_CLOCK

12.6 Read data from the e-Board

12.6.1 Stored games

In general, the bus mode is used at tournaments. Therefore, the full dataset of games stored in a board is not required for the tournament.
The data stored in the logfile can be requested using the command:

- DGT_BUS_REQ_FROM_START

Note: The marker to indicate the start of a game/event must be set. See previous paragraph.

12.6.2 During a game

The current state of all pieces on the board can be requested using the command:

- DGT_BUS_REQ_BOARD

All the changes at the board, including clock data (when available), since the previous request can be requested using the command:

- DGT_BUS_REQ_CHANGES

When there is no (valid) reply on DGT_BUS_REQ_CHANGES, the data can be requested again using the command:

- DGT_BUS_REPEAT_CHANGES

Note: The command DGT_BUS_REQ_FROM_START can be used also during a game. This can be used to restore the current game after communication loss or similar situations.

12.7 Read data from the clock

12.7.1 During a game – general

Note 1: The clock data is also available using the commands DGT_BUS_REQ_CHANGES or DGT_BUS_REPEAT_CHANGES. See chapter 16.

Note 2: The command DGT_BUS_REQ_FROM_START can be used also during a game. This can be used to restore the current game after communication loss or similar situations.

12.7.2 During a game – I²C

The current state of the clock can be requested using the command:

- DGT_BUS_REQ_I2C_CLOCK
- DGT_BUS_REQ_I2C_BUTTON
- DGT_BUS_REQ_I2C_ACK

When there is no (valid) reply on one of the above commands, the data can be requested again using one of the 'repeat'-commands:

- DGT_BUS_REPEAT_I2C_BUTTON
- DGT_BUS_REPEAT_I2C_ACK

Short overview of specific I^2C commands that can be used to request data from the clock, using the command DGT_BUS_I2C_CLOCK_MESSAGE. Overview of the commands, see chapter 20.

12.7.3 During a game – SBI

The current state of the clock can be requested using the command:

- DGT_BUS_REQ_SBI_CLOCK

Short overview of specific SBI commands that can be used to request data from the clock, using the command DGT_BUS_SBI_CLOCK_MESSAGE. Overview of the commands, see chapter 21.

12.8 Special functions

12.8.1 Status of the board

Beside piece information and movements at the board, other status of the board can be requested.

For the battery status use the command:

- DGT_BUS_REQ_BATTERY_STATUS

Various other information of the board is available (see chapter 18.13). Use command:

- DGT_BUS_REQ_STATS

12.8.2 Status and commands of the clock using I2C

Short overview of commands that can be used for status information and of special commands for the clock, using the command DGT_BUS_I2C_CLOCK_MESSAGE: see chapter 11.7.2.

12.8.3 Status and commands of the clock using SBI

Short overview of commands that can be used for status information and of special commands for the clock, using the command DGT_BUS_SBI_CLOCK_MESSAGE: see chapter 11.7.3.

12.8.4 Switch to single board mode

When the board will be used in single mode afterwards, the board has to be in the so called idle-mode. The command DGT_BUS_END_BUSMODE will put the board in this mode.

Note: When a board receives a single board command in bus-mode, the board will change to single board mode automatically.

13 Single board command codes from computer to board without data

13.1 General

For all messages without data, the format below is used.

Message format:

- byte 1: Command code (see chapter 9.1)

13.2 DGT_REQ_RESET (0x40)

Purpose: Puts the board in IDLE mode, cancelling any UPDATE mode.
Response: none

13.3 DGT_REQ_SBI_CLOCK (0x41)

Purpose: Request for clock-data. Used for clocks with SBI communication.
Response: Results in a DGT_MSG_SBI_CLOCK message

13.4 DGT_REQ_BOARD (0x42)

Purpose: Request for all pieces at the chessboard.
Response: Results in a DGT_MSG_BOARD_DUMP message

13.5 DGT_REQ_UPDATE_SBI (0x43)

Purpose: Request for all pieces at the chessboard and clock information, every fixed period (x seconds) Used for clocks with SBI communication.
Response: Results in DGT_MSG_FIELD_UPDATE messages and DGT_MSG_SBI_CLOCK messages as long as the board is in UPDATE_SBI mode

13.6 DGT_REQ_UPDATE_BOARD (0x44)

Purpose: Request for all pieces at the chessboard every fixed period (x seconds
Response: results in DGT_MSG_FIELD_UPDATE messages as long as the board is in UPDATE_BOARD mode

13.7 DGT_REQ_SERIALNR (0x45)

Purpose: Request for the serial number of the board
Response: Results in a DGT_MSG_SERIALNR message

13.8 DGT_REQ_BUSADDRESS (0x46)

Purpose: Request for the bus address of the board
Response: Results in a DGT_MSG_BUSADDRESS message

13.9 DGT_REQ_TRADEMARK (0x47)

Purpose: Request for the identification of the board
Response: Results in a DGT_MSG_TRADEMARK message

13.10 DGT_REQ_HARDWARE_VERSION (0x48)

Purpose: Request for the hardware version of the board
Response: Results in a DGT_MSG_HARDWARE_VERSION message

13.11 DGT_REQ_LOG_MOVES (0x49)

Purpose: Request for all moves stored in the board, not asked before.
Response: Results in a DGT_MSG_LOG_MOVES message

13.12 DGT_TO_BUSMODE (0x4a)

Purpose: Change the board mode to BUS
Response: None

This is an addition on the other single-board commands. This command is recognized in single-board mode. The bus commands are immediately recognized hereafter.

Note: When the board is in single-board mode, and eventually a bus mode command is found, this command is not processed at boards with revision lower than 4.x, but the board switches to bus mode. The next (bus) command is processed regularly.

13.13 DGT_REQ_UPDATE_SBI_NICE (0x4b)

Purpose: Request for all pieces at the chessboard and clock information, only after a change of the clock has occurred Used for clocks with SBI communication.
Response: results in DGT_MSG_FIELD_UPDATE messages and DGT_MSG_SBI_CLOCK messages, the latter only at time changes, as long as the board is in UPDATE_SBI_NICE mode

13.14 DGT_REQ_BATTERY_STATUS (0x4c)

Purpose: Requests the battery status of the board.
Response: Results in DGT_MSG_BATTERY_STATUS
Only valid when the board is equipped with a battery. Otherwise no response.

13.15 DGT_REQ_VERSION (0x4d)

Purpose: Request for the firmware version of the board
Response: Results in a DGT_MSG_VERSION message

13.16 DGT_REQ_LONG_SERIALNR (0x55)

Purpose: Request for the long serial number of the board.
Response: Results in a DGT_LONG_SERIALNR message

13.17 DGT_REQ_I2C_CLOCK (0x56)

Purpose:	Request for clock-data. Used for clocks with I²C communication.
Response:	Results in a DGT_MSG_I2C_CLOCK message

13.18 DGT_REQ_UPDATE_I2C (0x57)

Purpose:	Request for all pieces at the chessboard and clock information, every fixed period (x seconds) Used for clocks with I²C communication.
Response:	Results in DGT_MSG_FIELD_UPDATE messages and DGT_MSG_I2C_CLOCK messages as long as the board is in UPDATE_I2C mode

13.19 DGT_REQ_UPDATE_I2C_NICE (0x58)

Purpose:	Request for all pieces at the chessboard and clock information, only after a change of the clock has occurred. Used for clocks with I²C communication.
Response:	results in DGT_MSG_FIELD_UPDATE messages and DGT_MSG_I2C_CLOCK messages, the latter only at time changes, as long as the board is in UPDATE_I2C_NICE mode

14 Single board command codes from computer to board containing data

14.1 DGT_SBI_CLOCK_MESSAGE (0x2b)

Purpose:	This message contains a command for the clock. Used for clocks with SBI communication. There are clock commands for showing text, displaying icons, setting beep, clearing display, and for setting clock times. All these clock commands are wrapped within the DGT_SBI_CLOCK_MESSAGE command.
Response:	The board responds with a DGT_MSG_SBI_CLOCK message if the board is in UPDATE or UPDATE_SBI_NICE mode. If the board is in IDLE or UPDATE_BOARD mode, then no response.

Constants:

- #define DGT_CMD_CLOCK_START_MESSAGE 0x03
- #define DGT_CMD_CLOCK_END_MESSAGE 0x00

Message format:

- byte 1-2: header (see chapter 9.2)
- byte 3: DGT_CMD_CLOCK_START_MESSAGE
- byte 4: one of the clock command id's
- byte 5-(n+1): the content (can be empty)
- byte n+2: DGT_CMD_CLOCK_END_MESSAGE

for SBI clock commands see chapter 21.

14.1.1 USB2.0; type-B mini or Type-C/Serial

You cannot send multiple clock commands directly after each other. After each command, you should wait for the response/acknowledge before sending the next one. The response/acknowledge is usually returned within one or two seconds.

14.1.2 BT2.10

Clock commands are processed faster and more reliable, but note:

- When a clock command is sent, it is possible that an empty DGT_MSG_SBI_CLOCK (see chapter 21.3.1) message is returned (full of zeroes). This is to be ignored.
- Within 1000ms after sending a clock command (and receiving its ack), subsequent clock update requests may return this ack instead of the clock update. This is because the clock is polled only 2-3 times a second.

14.2 DGT_I2C_CLOCK_MESSAGE (0x2c)

Purpose:	This message contains a command for the clock. There are clock commands for showing text, displaying icons, setting beep, clearing display, and for setting clock times. All these clock commands are wrapped within the DGT_I2C_CLOCK_MESSAGE command.
Response:	DGT_MSG_I2C_CLOCK

Message format:

- byte 1-2: header (see chapter 9.2)
- byte 3-(n+1): I^2C packet
- byte n+2: end of message (= 0x00)

for I^2C packet description see chapter 20.

14.3 DGT_SET_LEDS (0x60)

Purpose:	Only for the Revelation II to switch a LED pattern.
Response:	None

Message format:

- byte 1-2: header (see chapter 9.2)
- byte 3: the pattern to display (0 - off, 1 - on, 2 - auto flags)
  - Firmware <=3.21: only 0x01 - other values ignored
- byte 4: the start field
- byte 5: the end field
- byte 6: end of message (= 0x00)

Start- and end-field have the range 0-63 where 0 is field a8 and 63 is field h1. This is compliant with the DGT field coding of the board. Other values are ignored. See chapter 5.1.

If "Off" mode (byte 3 = 0x00) and byte 4 = 0x40 (byte5 doesn't matter) all lights are cleared

By default, the light dims on a field if a piece change is detected. This behavior can be switched off (see below).

By default, the board and led signaling is reversed when a reversed board is detected (black pieces in correct order on two bottom rows). This behavior can be switched off (see below).

byte3 = 0x02 and byte 4-5:

- 0x01 0x00 : Auto OFF functions LEDS disabled.
- 0x01 0x01 : Auto OFF functions LEDS enabled (=default).
- 0x02 0x00 : Auto Reverse board function disabled.
- 0x02 0x01 : Auto Reverse board function enabled (=default).

15 Single board messages from board to computer

15.1 General

The board responds in general only to a request. No message will be sent initiated by the board.

Only after the board is set in one of the UPDATE-modes (see chapter 6), the board will send every move at the board. Clock data will be sent on clock-change or predefined time-interval.

Note: Bit 8 of the byte indicates a message from the board to the computer (reply).

15.2 DGT_MSG_BOARD_DUMP (0x86)

Reply on:	DGT_REQ_BOARD
Response:	Information of all board fields. Board fields are numbered from 0 to 63, the code of the pieces is in the data. For the definitions, see chapter 5.

message format:

- byte 1-3: header (see chapter 9.3)
- byte 4-67: Pieces on position 0-63

15.3 DGT_MSG_HARDWARE_VERSION (0x96)

Reply on:	DGT_REQ_HARDWARE_VERSION
Response:	Message which contains the hardware version of the board.

message format:

- byte 1-3: header (see chapter 9.3)
- byte 4: main version number; 7 bits hexadecimal value
- byte 5: sub-version number; 7 bits hexadecimal value

15.4 DGT_MSG_SBI_CLOCK (0x8d)

Reply on:	DGT_REQ_SBI_CLOCK
Response:	Clock-data, according SBI command structure.
Reply on:	DGT_REQ_UPDATE_SBI
Response:	Clock-data, according SBI command structure. This message will be sent on a fixed time-base as long as the board is in UPDATE_SBI mode
Reply on:	DGT_REQ_UPDATE_SBI_NICE Clock-data, according SBI command structure. This message will be sent on a change of the clock only, as long as the board is in UPDATE_SBI_NICE mode

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4-10: SBI-message

For SBI clock commands see chapter 21.

15.5 DGT_MSG_FIELD_UPDATE (0x8e)

Reply on:	DGT_REQ_UPDATE BOARD
Response:	Message from the board when one of the fields changes; field number where the change occurred, current piece at that fieldnumber (including empty field). As long as the board is in UPDATE_BOARD mode
Reply on:	DGT_REQ_UPDATE_SBI
Response:	As above, as long as the board is in UPDATE_SBI mode
Reply on:	DGT_REQ_UPDATE_SBI_NICE
Response:	As above, as long as the board is in UPDATE_SBI_NICE mode
Reply on:	DGT_REQ_UPDATE_I2C
Response:	As above, as long as the board is in UPDATE_I2C mode
Reply on:	DGT_REQ_UPDATE_I2C_NICE
Response:	As above, as long as the board is in UPDATE_I2C_NICE mode

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4: field number (0-63) which changed the piece code
- byte 5: piece code including EMPTY, where a non-empty field became empty

15.6 DGT_MSG_LOG_MOVES (0x8f)

Reply on:	DGT_REQ_LOG_MOVES
Response:	Message contains the contents of the storage at the board (field changes, clock changes), starting with the oldest data, until the last written changes.

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4-end: field change storage stream.

Note: Explanation and content of the log-messages: see chapter 19.

15.7 DGT_MSG_BUSADDRESS (0x90)

Reply on: DGT_REQ_BUSADDRESS
Response: Message which contains the bus-address of the board.

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4-5: Bus-address in 2 bytes of 7 bits hexadecimal value

The value of the 14-bit bus-address is the hexadecimal representation of the (decimal coded) serial number. See chapter 8.1.3

15.8 DGT_MSG_SERIALNR (0x91)

Reply on: DGT_REQ_SERIALNR
Response: The 5-digit serial number of the board; represented in a character string

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4-8: serial number string, last byte is LSByte

15.9 DGT_MSG_TRADEMARK (0x92)

Reply on: DGT_REQ_TRADEMARK
Response: Trademark of the board, represented in an ASCII-character string format. Character string length may be in the range of 0 to 256

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4-end: ASCII trademark message, codes 0 to 0x3f

15.10 DGT_MSG_VERSION (0x93)

Reply on: DGT_REQ_VERSION
Response: Message which contains the firmware version of the board.

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4: main version number; 7 bits hexadecimal value
- byte 5: sub-version number; 7 bits hexadecimal value

15.11 DGT_MSG_BATTERY_STATUS (0xa0)

Reply on: DGT_REQ_BATTERY_STATUS
Response: Usage and status of the battery. See message format below. Response only when the board contains a battery.

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4: Current battery capacity left in %
- byte 5: Running/Charging time left (hours) (0x7f is N/A)
- byte 6: Running/Charging time left (minutes) (0x7f is N/A)
- byte 7: On time (hours)
- byte 8: On time (minutes)
- byte 9: Standby time (days)
- byte 10: Standby time (hours)
- byte 11: Standby time (minutes)
- byte 12: Status bits:
  - bit 0: Charge status
  - bit 1: Discharge status
  - bit 2: N.a.
  - bit 3: N.a.
  - bit 4: N.a.
  - bit 5: N.a.
  - bit 6: N.a.
  - bit 7: N.a.

Note: When the status bits indicate charging, byte 5 and 6 indicate charging time left.
When the status bits indicate discharging, byte 5 and 6 indicate running time left.

15.12 DGT_MSG_LONG_SERIALNR (0xa2)

Reply on: DSG_REQ_LONG_SERIAL
Response: The 10-digit serial number of the board; represented in a character string

Message format:

- byte 1-3: header (see chapter 9.3)
- byte 4-13: long serial number string, last byte is LSByte

15.13 DGT_MSG_I2C_CLOCK (0xa3)

Reply on:	DGT_REQ_I2C_CLOCK
Response:	Clock-data, according I²C command structure.
Reply on:	DGT_REQ_UPDATE_I2C
Response:	Clock-data, according I²C command structure. This message will be sent on a fixed time-base as long as the board is in UPDATE_I2C mode
Reply on:	DGT_REQ_UPDATE_I2C_NICE
Response:	Clock-data, according I²C command structure. This message will be sent on a fixed time-base as long as the board is in UPDATE_I2C_NICE mode

Message format:

- byte 1: DGT_MSG_I2C
- byte 1-3: header (see chapter 9.3)
- byte 4-n: I²C packet

for I²C packet description see chapter 20.

16 bus command codes from computer to board without data

16.1 General

For all messages without data, the format below is used.

Message format:

- byte 1-4: header and checksum (see chapter 10.2)

16.2 DGT_BUS_REQ_SBI_CLOCK (0x81)

Purpose:	Request for clock-data of the addressed board. Used for clocks with SBI communication.
Response:	Results in a DGT_MSG_BUS_SBI_CLOCK message

16.3 DGT_BUS_REQ_BOARD (0x82)

Purpose:	Request for all pieces at the addressed board.
Response:	Results in a DGT_MSG_BUS_BOARD_DUMP message

16.4 DGT_BUS_REQ_CHANGES (0x83)

Purpose:	Request for all stored information changes at the addressed board from the moment of the last DGT_BUS_REQ_CHANGES.
Response:	Results in a DGT_MSG_BUS_UPDATE_ODD message from the board

Note: In case this data does not arrive properly, the data can be asked again with the DGT_BUS_REPEAT_CHANGES command.

16.5 DGT_BUS_REPEAT_CHANGES (0x84)

Purpose:	This command causes the addressed board to send last sent packet of changes again.
Response:	Results in a DGT_MSG_BUS_UPDATE_ODD message from the board.

16.6 DGT_BUS_SET_START_GAME (0x85)

Purpose:	This command sets a tag in the internal board changes buffer (addressed board). After this the positions of the pieces and clock data (when available) are all logged at the board. The logged data can be retrieved using the command DGT_BUS_REQ_FROM_START
Response:	DGT_MSG_BUS_START_GAME_WRITTEN message. ***Note:*** Response about 70 msec. after receipt of DGT_BUS_SET_START_GAME

16.7 DGT_BUS_REQ_FROM_START (0x86)

Purpose: This command causes the addressed board to send a message, containing all update information. The data starts from the tag set by the command DGT_BUS_SET_START_GAME until the last registered changes, already sent before.

Note 1: The new added data has to be requested by the DGT_BUS_REQ_CHANGES command.

Note 2: All piece positions are written in the log file.

Response: DGT_MSG_BUS_FROM_START

16.8 DGT_BUS_PING (0x87)

Purpose: This message is used to identify the connected boards or to check the availability of an addressed board.

Response: DGT_MSG_BUS_PING

Note: when the DGT_BUS_PING command is sent with board address zero (0) all connected boards will reply with a DGT_MSG_BUS_PING message, randomly spread over a 1100 msec. interval. This to avoid collision. For reliable identification of all connected boards, this process should be repeated sometimes with checking of checksums!

16.9 DGT_BUS_END_BUSMODE (0x88)

Purpose: Change the board mode to IDLE (single-board mode)

Response: None

Note 1: Any single-board command arriving during bus mode will switch a board to single-board mode and will be processed.

Note 2: When sent with address zero (0) the command is processed on all connected boards.

16.10 DGT_BUS_RESET (0x89)

Purpose: This command forces a power-up reset procedure.

Response: None. The board mode changes, see chapter 6

Note 1: This procedure takes some seconds.

Note 2: When sent with address zero (0) the command is processed on all connected boards

16.11 DGT_BUS_IGNORE_NEXT_BUS_PING (0x8a)

Purpose: This command is used in the process of detecting connected boards on a bus. After this command to the addressed board, the first following DGT_BUS_PING with address zero (0) is ignored.
This command is used to suppress response of already detected boards, and decreases the chance of bus collisions.
Response: DGT_MSG_BUS_PING

16.12 DGT_BUS_REQ_VERSION (0x8b)

Purpose: Request for the firmware version of the addressed board
Response: Results in a DGT_MSG_BUS_VERSION message

16.13 DGT_BUS_REQ_ALL_D (0x8d)

Purpose: Request for all field-data and the clock-data of the addressed board. Used for clocks with SBI communication.
Is identical to the command sequence:

- DGT_BUS_REQ_CHANGES
- DGT_BUS_SEND_SBI_CLOCK
- DGT_BUS_REQ_BOARD

Response: Results in:

- DGT_MSG_BUS_UPDATE_ODD
- DGT_MSG_BUS_SBI_CLOCK
- DGT_MSG_BUS_BOARD_DUMP

16.14 DGT_BUS_REQ_SERIALNR (0x91)

Purpose: Request for the serial number of the addressed board
Response: Results in a DGT_MSG_BUS_SERIALNR message

16.15 DGT_BUS_RPING (0x92)

Purpose: When multiple boards with the same bus-address are detected, this command can be used to change the bus-address of these boards.
When send to a specific address (so not zero (0)) all the boards with this address will respond with a message containing a new random address within a random time of 1100ms.
Response: DGT_MSG_BUS_PING

Note: The new bus-address will be stored at the board and remains the bus-address until it will be changed again using this command.

16.16 DGT_BUS_REQ_I2C_CLOCK (0x93)

Purpose: Request for clock-data of the addressed board. Used for clocks with I^2C communication.
Response: Results in a DGT_MSG_BUS_I2C message
Note: Only one (1) clock-data message in the buffer of the board

16.17 DGT_BUS_REQ_I2C_BUTTON (0x94)

Purpose: Request for the next I^2C clock-button message from the buffer of the addressed board. Used for clocks with I^2C communication.
Response: Results in a DGT_MSG_BUS_I2C message
Also confirms the previous button message.
Note: Maximum of eight (8) clock-button messages in the buffer of the board

16.18 DGT_BUS_REQ_I2C_ACK (0x95)

Purpose: Request for the next I^2C message from the buffer of the addressed board. Used for clocks with I^2C communication.
Response: Results in a DGT_MSG_BUS_I2C message
Also confirms the previous I^2C message.
Note: Only one (1) I^2C message in the buffer of the board

16.19 DGT_BUS_REQ_STATS (0x96)

Purpose: This command asks for some statistics from the addressed board.
Response: Results in a DGT_MSG_BUS_STATS message

16.20 DGT_BUS_REQ_TRADEMARK (0x97)

Purpose: Request for the identification of the addressed board
Response: Results in a DGT_MSG_BUS_TRADEMARK message

16.21 DGT_BUS_REQ_BATTERY (0x98)

Purpose: Requests the battery status of the addressed board.
Response: Results in DGT_MSG_BUS_BATTERY_STATUS
Only valid when the board is equipped with a battery

16.22 DGT_BUS_REPEAT_I2C_BUTTON (0x99)

Purpose: Repeated request for the oldest unconfirmed I^2C clock-button message from the buffer of the addressed board. Used for clocks with I^2C communication.
Response: Results in a DGT_MSG_BUS_I2C message

16.23 DGT_BUS_REPEAT_I2C_ACK (0x9a)

Purpose: Repeated request for the latest unconfirmed I^2C message, other than button or clock, from the buffer of the addressed board. Used for clocks with I^2C communication.
Response: Results in a DGT_MSG_BUS_I2C message

16.24 DGT_BUS_REQ_HARDWARE_VERSION (0x9b)

Purpose:	Request for the hardware version of the addressed board
Response:	Results in a DGT_MSG_BUS_HARDWARE_VERSION message

17 bus command codes from computer to board containing data

17.1 General

Message description and message format: see chapter 10.3

17.2 DGT_BUS_CONFIG (0xa0)

Purpose: Read current setting or configure (= write) the scan speed of the addressed board (only for the DGT e-Board with version 4.x or higher).
Response: DGT_MSG_BUS_CONFIG

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: 0=read current setting, 1=write (configure)
- byte 7: quick scan interval in 2^x/16 sec. Default is 2 (4/16 sec)
- byte 8: complete scan interval in 2^x quick scans. Default is 4 (16 scans)
- byte 9: Checksum

Note: Complete scan is used to detect all 64 chess-fields.
Quick scan only detects the change of existing pieces at the board.

17.3 DGT_BUS_SBI_CLOCK_MESSAGE (0xa1)

Purpose: This message contains a SBI command for the clock attached at the addressed board. Used for clocks with SBI communication.
There are clock commands for showing text, displaying icons, setting beep, clearing display, and for setting clock times. All these clock commands are wrapped within the DGT_BUS_SBI_CLOCK_MESSAGE command.
Response: DGT_MSG_BUS_SBI_CLOCK

Constants:

- #define DGT_CMD_CLOCK_START_MESSAGE 0x03
- #define DGT_CMD_CLOCK_END_MESSAGE 0x00

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: DGT_CMD_CLOCK_START_MESSAGE
- byte 7: one of the 7 clock command id's
- byte 8-(n-2): the content (can be empty)
- byte n-1: DGT_CMD_CLOCK_END_MESSAGE
- byte n: Checksum

For SBI clock commands see chapter 21.

17.3.1 USB2.0; Type-B mini or Type-C/Serial

You cannot send multiple clock commands directly after each other. After each command, you should wait for the response/acknowledge before sending the next one. The response/acknowledge is usually returned within one or two seconds.

17.3.2 BT2.10

Clock commands are processed faster and more reliable, but note:

- When a clock command is sent, it is possible that an empty DGT_MSG_SBI_CLOCK (see chapter 21.3.1) message is returned (full of zeroes). This is to be ignored.
- Within 1000ms after sending a clock command (and receiving its ack), subsequent clock update requests may return this ack instead of the clock update. This is because the clock is polled only 2-3 times a second.

17.4 DGT_BUS_I2C_CLOCK_MESSAGE (0xa2)

Purpose:	This message contains a command for the I²C clock attached at the addressed board. There are clock commands for showing text, displaying icons, setting beep, clearing display, and for setting clock times. All these clock commands are wrapped within the DGT_BUS_I2C_CLOCK_MESSAGE command.
Response:	DGT_MSG_BUS_I2C

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 4-(n-2): I²C packet
- byte n-1: end of message (= 0x00)
- byte n: Checksum

for I²C packet description see chapter 20.

18 bus message codes from board to computer

18.1 General

In BUS-mode the boards respond only to a request. No message will be sent initiated by the board.

18.2 DGT_MSG_BUS_BOARD_DUMP (0x83)

Reply on: DGT_BUS_REQ_BOARD
Response: Information of all board fields.
Board fields are numbered from 0 to 63, the code of the pieces is in the data.
For the definitions, see chapter 5.

message format:

- byte 1-5: header (see chapter 10.3)
- byte 6-69: Pieces on position 0-63
- byte 70: Checksum

18.3 DGT_MSG_BUS_SBI_CLOCK (0x84)

Reply on: DGT_BUS_REQ_SBI_CLOCK
Response: Clock-data, according SBI command structure.

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6-12: SBI-message
- byte 13: Checksum

For SBI clock commands see chapter 21.

18.4 DGT_MSG_BUS_UPDATE_ODD (0x85)

Reply on: DGT_BUS_REQ_CHANGES
Response: Message contains the contents of the storage, starting with the oldest data, until the last written changes.

Reply on: DGT_BUS_REPEAT_CHANGES
Response: Message contains the contents of the last sent packet of changes again.

Reply on: DGT_BUS_REQ_ALL_D
Response: Message contains the contents of the storage, starting with the oldest data, until the last written changes

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6-(n-1): field change storage stream.
- byte n: Checksum

Note 1: The data area contains a variable amount of change information, formatted as described in chapter 19. This message is used for the odd numbered messages.

Note 2: In the original design of the protocol, a distinction between ODD and EVEN messages was made to be in control on the order of the messages. In practice this didn't seem to be necessary, so the EVEN messages are not used anymore.

18.5 DGT_MSG_BUS_FROM_START (0x86)

Reply on: DGT_BUS_REQ_FROM_START
Response: Message contains the contents of the storage, starting from the marked point (start of game), until the last written changes

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6-(n-1): field change storage stream.
- byte n: Checksum

Note: The data area contains a variable amount of change information, formatted as described in chapter 19.

18.6 DGT_MSG_BUS_PING (0x87)

Reply on: DGT_BUS_PING
Response: Message with an empty data area; acknowledge of the addressed board
Note: Send this message only when the previous message was not DGT_BUS_IGNORE_NEXT_BUS_PING

Reply on: DGT_BUS_IGNORE_NEXT_BUS_PING
Response: Message with an empty data area; acknowledge of the addressed board

Reply on: DGT_BUS_RPING
Response: Message with an empty data area, but with a new random bus-address. Message will be sent within a random time of 1100ms.
Note: This new bus-address is the new bus-address of the board and will only change after another DGT_BUS_RPING command.

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: Checksum

18.7 DGT_MSG_BUS_START_GAME_WRITTEN (0x88)

Reply on: DGT_BUS_SET_START_GAME
Response: Message with an empty data area; acknowledge of the addressed board
Note: The tag in the internal board changes buffer is set at the actual buffer position (start of game)

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: Checksum

18.8 DGT_MSG_BUS_VERSION (0x89)

Reply on: DGT_BUS_VERSION
Response: Message which contains the firmware version of the board.

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: main version number; 7 bits hexadecimal value
- byte 7: sub-version number; 7 bits hexadecimal value
- byte 8: Checksum

Note: This was a 6 byte message only up to firmware revision 1.2

18.9 DGT_MSG_BUS_UPDATE_EVEN (0x8b)

Identical to DGT_MSG_BUS_UPDATE_ODD, see chapter 18.4

This message is reserved for future use.

18.10 DGT_MSG_BUS_HARDWARE_VERSION (0x8c)

Reply on: DGT_BUS_HARDWARE VERSION
Response: Message which contains the hardware version of the board.

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: main version number; 7 bits hexadecimal value
- byte 7: sub-version number; 7 bits hexadecimal value
- byte 8: Checksum

18.11 DGT_MSG_BUS_SERIALNR (0xb0)

Reply on: DGT_BUS_REQ_SERIALNR
Response: The 10-digit serial number of the board; represented in a character string

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6-15: long serial number string, last byte is LSB
- byte 16: Checksum

18.12 DGT_MSG_BUS_I2C (0xb1)

Reply on: DGT_BUS_REQ_I2C_CLOCK
Response: Clock-data, according I^2C command structure.
Note: Only the most recent clock-update is available at the board.

Reply on: DGT_BUS_REQ_I2C_BUTTON
Response: The next I^2C clock-button message from the buffer of the addressed board, according I^2C command structure.
Note 1: Is also acknowledge of the previous sent I^2C command from the board.
Note 2: Up to the last eight (8) clock-button messages are available at the board.

Reply on: DGT_BUS_REQ_I2C_ACK
Response: The next I^2C message from the buffer of the addressed board, according I^2C command structure.
Note 1: Is also acknowledge of the previous sent I^2C command from the board.
Note 2: Only one (1) I^2C message at a time at the board.

Reply on: DGT_BUS_REPEAT_I2C_BUTTON
Response: The oldest unconfirmed I^2C clock-button message from the I^2C-data buffer of the addressed board, according I^2C command structure.

Reply on: DGT_BUS_REPEAT_I2C_ACK
Response: The latest unconfirmed I^2C message, other than button or clock, from the I^2C-data buffer of the addressed board, according I^2C command structure.

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6-(n-1): I^2C packet
- byte n: Checksum

for I^2C packet description see chapter 20.

18.13 DGT_MSG_BUS_STATS (0xb2)

Reply on: DGT_BUS_REQ_STATS
Response: List of counters for debugging and informational purpose.

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6-10: Counter-0
- : :
- : :
- Byte 76-80: Counter 14
- byte 81: Checksum

Counter format (5 bytes):

- byte 1: bits 0-3 = counter bits 28-31
- byte 2: bits 0-6 = counter bits 21-27
- byte 3: bits 0-6 = counter bits 14-20
- byte 4: bits 0-6 = counter bits 7-13
- byte 5: bits 0-6 = counter bits 0-6

Counter	e-Board / Revelation
0	power up count
1	total uptime (seconds)
2	new game count
3	start game count
4	clock updates count
5	field updates count
6	debug messages count
7	clear page count
8	quick scan count
9	complete scan count
10	redundant scan count
11	inconsistent scan count
12	current uptime (seconds)
13	USB voltage
14	jack voltage

18.14 DGT_MSG_BUS_CONFIG (0xb3)

Reply on: DGT_BUS_CONFIG
Response: When byte 6 of DGT_BUS_CONFIG is "read" then reply with the current settings of the board.
When byte 6 of DGT_BUS_CONFIG is "write" then reply with the new settings of the board.
Note 1: Only for the universal e-Board
Note 2: When "write", then the settings of the board will be changed accordingly.

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: quick scan interval in 2^x/16 sec. Default is 2 (4/16 sec)
- byte 7: complete scan interval in 2^x quick scans. Default is 4 (16 scans)
- byte 8: Checksum

18.15 DGT_MSG_BUS_TRADEMARK (0xb4)

Reply on: DGT_BUS_REQ_TRADEMARK
Response: Trademark of the board, represented in an ASCII-character string format. Character string length may be in the range of 0 to 256

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 5-(n-1): ASCII trademark message, codes 0 to 0x3f
- byte n: Checksum

18.16 DGT_MSG_BUS_BATTERY_STATUS (0xb5)

Reply on: DGT_BUS_REQ_BATTERY
Response: Status of the battery. See message format below

Message format:

- byte 1-5: header (see chapter 10.3)
- byte 6: bit 0-2:
  - 0 = below 5% 3.60v
  - 1 = below 20%3.68v
  - 2 = below 40%3.71v
  - 3 = below 60%3.76v
  - 4 = below 80%3.93V
  - 5 = below 100% (or full)

  bit 3-4:
  - 0 = discharging
  - 1 = charging
  - 2 = charging finished
- byte 7: Checksum

19 Logfile of the e-Boards

19.1 General

The internal log of the e-Board is a cyclic buffer with a maximum length of about 8Kb.

The new written data always overwrites the oldest data.

In this buffer, sequentially messages of the board and clock connected to the board (if any) are written. The messages are of various length, from 1 to 4 bytes, specific for every message. Various events generate a message that is written in the storage, in the sequence as the events occur.

19.2 Read the logfile

There are two ways to read the logfile of the board; in single board mode or in bus mode.

19.2.1 Single board mode

In single board mode the whole logfile can be requested using the command:

- DGT_REQ_LOG_MOVES. (see chapter 13.11)

A LOG_DOWNLOADED message is written in the logfile, indicating the start of new log messages.

19.2.2 Bus mode

In bus mode the logfile can be requested using the commands:

- DGT_BUS_REQ_FROM_START (see chapter 0)
- DGT_BUS_REQ_CHANGES (see chapter 16.4)
- DGT_BUS_REPEAT_CHANGES (see chapter 16.5)

The response on these commands will either be the logfile messages since the last request or all the messages from the start of the game.

To indicate the start of a game in the logfile, use the command:

- DGT_BUS_SET_START_GAME (see chapter 16.6)

As a result of this command, the message LOG_START_TAG (see chapter 19.3.17) is written in the logfile.

19.3 Logfile messages

Because of the framing of the bus protocol the log message only uses the 7 LSB of each byte. The data bytes of each message are always below 0x5a. This means that the message codes from 0x59 are unique and can be used to find for example the start of a game.

In the electronic boards of DGT timing information is added, if more than 1/16 of a second has passed since the last message a delay message is written before the new message. The 4th bit in a field update message is now used to identify board changes that could have been done earlier than the delay messages suggest.

19.3.1 LOG_NOP2 (0x00)

No function. Used for alignment of the messages at the physical memory.

Message code range:

- 0x00

Message format:

- Byte 1: 0x00.

19.3.2 LOG_SUBSEC_DELAY (0x10)

Delay since previous message; used when less than 1 second.

Message code range:

- 0x01-0x0f:

Message format:

- Byte 1: The 4 LSB determine the delay in 1/16th of a second.

Examples:

- 0x01: 1/16 second delay.
- 0x0f: 15/16 second delay.

19.3.3 LOG_SEC_DELAY (0x10)

Delay since previous message; used when more than 1 second but less than 1 minute.

Message code range:

- 0x10-0x1f

Message format:

- Byte 1: The 4 LSB determine the delay in 1/16th of a second.
- Byte 2: The delay in seconds.

Examples:

- 0x10 0x01: 0/16 + 1 second delay.
- 0x1f 0x3b: 15/16 + 59 seconds delay.

19.3.4 LOG_MIN_DELAY (0x20)

Delay since previous message; used when more than 1 minute but less than 1 hour.

Message code range:

- 0x20-0x2f

Message format:

- Byte 1: The 4 LSB determine the delay in 1/16th of a second.
- Byte 2: The delay in seconds.
- Byte 3: The delay in minutes.

Examples:

- 0x20 0x00 0x01: 0/16 + 0 seconds + 1 minute delay.
- 0x2f 0x3b 0x3b: 15/16 + 59 seconds + 59 minutes delay.

19.3.5 LOG_HOUR_DELAY (0x30)

Delay of 1 hour since the previous message. In contrary to the smaller delays this delay is immediately written when the hour has passed. When more hours pass, multiple LOG_HOUR_DELAY messages are written.

Message code range:

- 0x30

Message format:

- Byte 1: 0x30.

19.3.6 LOG_FIELD_UPDATE (0x40)

The value of a field has changed. Since the last quick scan.

Message code range:

- 0x40-0x4f

Message format:

- Byte 1: The 4 LSB determine the new piece code.
- Byte 2: The field number 0-63.

Examples:

- 0x40 0x00: now empty (0x00) on field a8 (0x00)
- 0x4c 0x3f: now a black Queen (0x0c) on field h1 (0x3f)

Note: Piece code and field number definitions, see chapter 5

19.3.7 LOG_FIELD_UPDATE_AS (0x50)

The value of a field has changed. Since the last complete scan.

Message code range:

- 0x50-0x5f

Message format:

- Byte 1: The 4 LSB determine the new piece code.
- Byte 2: The field number 0-63.

Example:

- 0x54 0x07: now a white Bishop (0x04) on field h8 (0x07)

19.3.8 LOG_CLOCK_UPDATE_R (0x60)

Time update on the right side of the clock, this message is logged when the right side of the lever is pressed. Left is now up.

Message code range:

- 0x60-0x69

Message format:

- Byte 1: The 4 LSB determine the hours 0-9.
- Byte 2: The minutes, BCD encoded 00-59.
- Byte 3: The seconds, BCD encoded 00-59.

Example:

- 0x61 0x23 0x45: right clock paused at 1:23:45.

Note: BDC encoded means: 4 bits of a byte represent a value between 0 and 9

19.3.9 LOG_POWERUP (0x6a)

The e-Board is turned on.

Message code range:

- 0x6a

Message format:

- Byte 1: 0x6a.

Note: After startup the e-Board writes the log-messages LOG_NOP2, LOG_NOP2, LOG_NOP2, LOG_POWERUP, LOG_EMPTY_BOARD and then start scanning all fields.

19.3.10 LOG_EOF (0x6b)

End of the data. Send after the last byte of the log.

Message code range:

- 0x6b

Message format:

- Byte 1: 0x6b.

19.3.11 LOG_FOURROWS (0x6c)

Rows 1,2,7 and 8 are filled with pieces, but not in the starting position. Rows 3,4,5 and 6 are empty. To be tolerant on erroneous placement and i.e. to be able to play the "Random Chess" as proposed by Bobby Fischer.

Message code range:

- 0x6c

Message format:

- Byte 1: 0x6c.

19.3.12 LOG_EMPTY_BOARD (0x6d)

All fields are empty.

Message code range:

- 0x6d

Message format:

- Byte 1: 0x6d.

19.3.13 LOG_DOWNLOADED (0x6e)

The log file is downloaded.

Message code range:

- 0x6e

Message format:

- Byte 1: 0x6e.

19.3.14 LOG_START_POS (0x6f)

All the pieces are in the starting position, with all the white pieces at row 1 and 2 and all the black pieces at row 7 and 8. Rows 3 to 6 are empty.

Message code range:

- 0x6f

Message format:

- Byte 1: 0x6f.

19.3.15 LOG_CLOCK_UPDATE_L (0x70)

Time update on the left side of the clock, this message is logged when the left side of the lever is pressed. Right is now up.

Message code range:

- 0x70-0x79

Message format:

- Byte 0: The 4 LSB determine the hours 0-9.
- Byte 1: The minutes, BCD encoded 00-59.
- Byte 2: The seconds, BCD encoded 00-59.

Example:

- 0x71 0x23 0x45: left clock paused at 1:23:45.

19.3.16 LOG_START_POS_ROT (0x7a)

All the pieces are in the reversed starting position; all the black pieces at row 1 and 2 and all the white pieces at row 7 and 8. Rows 3 to 6 are empty.

Message code range:

- 0x7a

Message format:

- Byte 1: 0x7a.

19.3.17 LOG_START_TAG (0x7b)

This tag can be written on request of the computer to indicate the start of a game. This is only used in bus-mode.

Message code range:

- 0x7b

Message format:

- Byte 1: 0x7b.

19.3.18 LOG_DEBUG (0x7c)

An undesirable event has happened.

Message code range:

- 0x7c

Message format:

- Byte 1: 0x7c.
- Byte 2: Debug code (see below).

Debug code	Warning
0x01	flash memory is empty
0x02	RX buffer is full
0x03	command not handled in 1 second
0x04	command not handled; message size incorrect
0x05	command not handled; message incomplete
0x06	command not handled; CRC error
0x07	command not handled; last byte not 0 (zero)
0x08	2 milliseconds of noise
0x09	I^2C send failed
0x0a	I^2C receive - CRC failed
0x0b	I^2C receive - length failed
0x0c	I^2C receive - buffer full
0x10	power low
0x11	power down
0x12	power up
0x13	bootloader reset
0x14	voltage monitor2 reset
0x15	voltage monitor1 reset
0x16	voltage low reset
0x17	power on reset
0x18	reset pin reset

Note: Warnings, can happen with noise or disconnecting during communication

Debug code	Error
0x20	watchdog reset
0x21	undefined instruction
0x22	flash memory is full
0x23	supervisor instruction
0x24	unknown software reset
0x25	calibration failed

Note: Errors, should not happen, please report to DGT

Debug code	Information
0x40	Info; button startup
0x41	Info; power-plug startup
0x42	Info; calibration successful
0x43	Info; aborted button startup
0x44	Info; aborted power-plug startup

19.3.19 LOG_NEW_CLOCK_STATE (0x7d)

A clock has been connected, disconnected or the clock changed state.

Message code range:

- 0x7d

Message format:

- Byte 1: 0x7d.
- Byte 2: bit 0-3: Clock state (see below)
  bit 4: Lever position; 0=left side up, 1=right side up.
  bit 5: Type of clock (communication); 0=SBI, 1=I²C.

Code	Clock state
1	Off / disconnected
2	Option select
3	Wait for first start
4	Running
5	Paused
6	Blocked on flag
7	Time correction is being made

Note: Log-messages for states 3 to 7 are always followed by log-messages LOG_CLOCK_UPDATE_L and LOG_CLOCK_UPDATE_R.

19.3.20 LOG_NEW_BUS_ADDRESS (0x7e)

The bus address has changed by the computer. The new 14-bit address is written in three bytes (= values below 0x5a; this keeps the logfile-messages unique)

Message code range:

- 0x7e

Message format:

- Byte 1: 0x7e.
- Byte 2: bit 0-1: bits 12-13 of the new bus address.
- Byte 3: bit 0-5: bits 6-11 of the new bus address.
- Byte 4: bit 0-5: bits 0-5 of the new bus address.

19.3.21 LOG_EMPTY (0xff)

Value of an empty byte before data is written.

Message code range:

- 0xff

Message format:

- Byte 1: 0xff.

20 Clock commands – I²C

20.1 General

Communication between an e-Board and some DGT clocks is mainly based on I²C communication (see chapter 3.2). Communication of a computer with the clock is based on communication with the e-Board, in which the I²C-commands and -messages are integrated.

In this chapter the clock states and clock functions are described.

Only the functions used for communication with a computer via the DGT electronic boards are described.

20.2 I²C message format

Message format:

- Byte 1: Receiver device address
- Byte 2: Sender device address
- Byte 3: Total message length n (n range 5 - 255)
- Byte 4: Command code (1 - 255)
- Byte 5-(n-1): Command Payload; only when any payload used
- Byte n: Checksum: CRC-8 PEC (8-bit Packet Error Checking);
  $C(x) = x^8 + x^2 + x + 1$, polynomial. Calculated over byte 1 to byte n-1
- Byte n+1 Empty bytes are allowed here for devices that don't support PEC
- Byte n+2 Empty bytes are allowed here for devices that don't support PEC

20.3 Clock states

ID	Clock state	Description
1	OFF	Clock in lowest power consumption mode. No display, no action. Only responding to pressing ON/OFF button or a specially addressed PING message on I2C
2	OPTION SELECT	Clock is in a state where it is controlled by the buttons.
3	WAIT FOR FIRST START	Ready for starting the clock, after selection of an option
4	RUNNING	Time counting is running
5	PAUSED	Clock paused during a game
6	BLOCKED ON FLAG	Clock blocked while one or more flags are raised.
7	TIME CORRECTION	Clock is in a state where it is controlled by the buttons.
16	Manual setting state	Period 1 Method
17-26	Manual setting state	Period 1 Time (left and right separate)
27-32	Manual setting state	Period 1 Additional time (left and right separate)
33-35	Manual setting state	Period 1 Number of moves
36	Manual setting state	Period 2 Method
37-41	Manual setting state	Period 2 Time (left and right equal)
42-44	Manual setting state	Period 2 Additional time (left and right equal)
45-47	Manual setting state	Period 2 Number of moves

ID	Clock state	Description
48	Manual setting state	Period 3 Method
49-53	Manual setting state	Period 3 Time (left and right equal)
54-56	Manual setting state	Period 3 Additional time (left and right equal)
57-59	Manual setting state	Period 3 Number of moves
60	Manual setting state	Period 4 Method
61-65	Manual setting state	Period 4 Time (left and right equal)
66-68	Manual setting state	Period 4 Additional time (left and right equal)
69-71	Manual setting state	Period 4 Number of moves
72	Manual setting state	Period 5 Method
73-77	Manual setting state	Period 5 Time (left and right equal)
78-80	Manual setting state	Period 5 Additional time (left and right equal)
81-83	Manual setting state	Period 5 Number of moves
84	Manual setting state	Sound on/off
85	Manual setting state	Freeze on/off

The clock states are used in the commands I2C_TIME (see chapter 20.7.3) and I2C_BUTTON (see chapter 20.7.4)

20.4 Clock modes

ID	Clock mode	Description
	MANUAL PROGRAM	Clock is in a state where it is controlled by the buttons.
	CENTRAL CONTROL	Flag set by communication. If CENTRAL CONTROL is set, and the clock is connected to a board, some behavior is changed - By communication, the timing behavior of the clock can be manipulated, especially starting, halting, time correction and switching off and reset is possible by communication. - When the clock cable is unplugged, the Central Control flag is cleared.
	Computer/internet use	The computer/internet use is selected: counting of the clock is controlled by SetnRun commands (and/or Display commands)

Note: Computer/internet use for DGT 3000: Option 25.

20.5 I²C addresses

The following addresses are used in the DGT products:

Address	Device
0x00	General Call / broadcast
0x10	DGT3000
0x12	Local wireless socket in DGT3000
0x14	Electronic Chess Board
0x16	Audio output device
0x20	DGT Pi
0x50	Address to turn on the DGT3000 while it is off

20.6 I²C commands (computer via board to clock)

20.6.1 I2C_DEBUG (0x03)

Purpose: Sent to clock, to enable/disable certain debug features
Response: I2C_ACKNOWLEDGE

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: Debug command (see below)
- Byte 6: Checksum

Debug command	Description
0xd0	No reduction of automatic generated messages (Default power up)
0xd1	No messages when clock time did not change. Do send messages when buttons/leveler/events
0xd2	No messages when clock time did not change. No messages when buttons/levelers/events occur
0xd3	Never messages related to clock time Never messages related to buttons/levelers/events
0xe0	Disable all power-saving functions
0xd3	Enable all power-saving functions (Default power up)

20.6.2 I2C_DISPLAY (0x06)

Purpose: This command allows to overrun the clock display with content given in this message. The central 14-segment digits can be filled with ASCII characters, some icons can be controlled, and a beep signal can be generated.
Response: I2C_ACKNOWLEDGE indicating that the display is occupied
During the time that the display contains content from a I2C_DISPLAY command, no other I2C_DISPLAY commands can be accepted, and such command will result in a negative I2C_ACKNOWLEDGE command to the sender of it.
At the end of the duration timer (see message format below), when the display is restored, an I2C_ACKNOWLEDGE is sent, indicating that the display is available again.

Note: If a I2C_DISPLAY message is sent with a duration of 0 (so only buzzer can be activated), the I2C_ACKNOWLEDGE will respond that the display is free and is available. In case that a buzzer signal was initiated, the signal will continue according to the value of beep length. A new I2C_DISPLAY command will overwrite the buzzer signal.

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5-15: ASCII-string. See chapter 27 for the available characters and options.
- Byte 16: Duration
  - 0x00: no display change of ASCII string or icons. Effectively, by setting duration to 0, it is possible to generate only a beep, using beep length >0.
  - 0x01-0xa0: Duration of the message in 1/16 seconds. (max 10 seconds)
  - 0xff: infinite display; until I2C_ENDDISPLAY command, or I2C_SETNRUN and button pressed.
- Byte 17: Beep length
  - 0x00: no beep, or end running beep
  - 0x01-0x30: start beep with length in 1/16 seconds (max 3 seconds)
- Byte 18: Use icons
  - Bit 1: overwrite icons
    - 0: icons info (byte 19,20) are not processed
    - 1: overwrite icons according to bytes 19 and 20
  - Bit 0: display icons
    - 0: al icons are kept as they are
    - 1: clear all icons except low-bat, connected and central-control.
- Byte 19: Left side icons
  - Bit 5: dot between hours and minutes (left only)
  - Bit 4: dot between minutes and seconds
  - Bit 3: colon between hours and minutes
  - Bit 2: black king
  - Bit 1: white king
  - Bit 0: flag
  - For all icons: 1 = on, 0 = off
- Byte 20: Right side icons
  - Bit 4: dot between minutes and seconds
  - Bit 3: colon between hours and minutes
  - Bit 2: black king
  - Bit 1: white king
  - Bit 0: flag
  - For all icons: 1 = on, 0 = off
- Byte 21: Checksum

20.6.3 I2C_END_DISPLAY (0x07)

Purpose: The display is restored to the content as generated by the clock functions. A running beep sound due to a beep message is stopped.
Response: I2C_ACKNOWLEDGE after processing the message (general call, address 0)

Message format:

- Byte 1-5: I²C header and Checksum (see chapter 20.2)

20.6.4 I2C_CURRENTPROGRAM (0x08)

Purpose: Request for one of the available programs of the clock or for the current active program of the clock.
Response: I2C_PROGRAM (see chapter 0)

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: option
  - 1-30: Return I2C_PROGRAM message of the requested option (program)
  - 128: Return I2C_PROGRAM message of the currently selected option (active program)
- Byte 6: Checksum

20.6.5 I2C_PROGRAM (0x09)

Purpose: Programming of the clock. Only for the free programmable option(s) (program).
Response: I2C_ACKNOWLEDGE after processing the message

Note When this message was not sent to the clock, this message is the acknowledge on the command I2C_CURRENTPROGRAM (see chapter 20.6.4)

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: option/program number
  - Bit 7: Freeze
    - 0 = freeze off
    - 1 = freeze on
  - Bit 6: Sound
    - 0 = sound off
    - 1 = sound on
  - Bit 0-5: Option/program number (see specification of clock)
- Byte 6: Period 1 method (see table below)

- Byte 7: Period 1 left player hours
  - Bit 0-3: hours (0-9)
- Byte 8: Period 1 left player minutes
  - Bit 4-6: tens of minutes (0-5)
  - Bit 0-3: minutes units (0-9)
- Byte 9: Period 1 left player seconds
  - Bit 4-6: tens of seconds (0-5)
  - Bit 0-3: seconds units (0-9)
- Byte 10-12: Period 1 right player; as bytes 7-9
- Byte 13: Period 1 left player add minutes
  - Bit 0-3: minutes units (0-9)
- Byte 14: Period 1 left player add seconds
  - Bit 4-6: tens of seconds (0-5)
  - Bit 0-3: seconds units (0-9)
- Byte 15-16: Period 1 right player; as bytes 13-14
- Byte 17: Period 1 moves hundreds
  - Bit 0-3: hundreds moves (0-9)
- Byte 18: Period 1 moves
  - Bit 4-7: tens of moves (0-9)
  - Bit 0-3: moves units (0-9)
- Byte 19-26 Period 2; as bytes 6-18
- Byte 27-34 Period 3; as bytes 6-18
- Byte 35-42 Period 4; as bytes 6-18
- Byte 43-50 Period 5; as bytes 6-18
- Byte 51: Checksum

Code	Method
0	Time
1	Fischer
2	Delay
3	US delay
4	Byo-yomi
5	Canadian byo-yomi
6	Up-count
7	End (no further period defined)

20.6.6 I2C_SETNRUN (0x0a)

Purpose: The specified times are put on the display, and the clock starts counting according to the count values: up or down.
The buzzer signals are generated according to the user setting and default beeping moments. The corresponding flag is raised when a time runs to zero.

Note 1: If applicable: an eventually displayed message at the clock (see chapter 20.6.2) will be removed.

Note 2: Only valid when the clock is in option (program) "Computer/Internet use"

Response: I2C_ACKNOWLEDGE after processing the message

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: Period 1 left player hours
  - Bit 0-3: hours (0-9)
- Byte 6: Period 1 left player minutes
  - Bit 4-6: tens of minutes (0-5)
  - Bit 0-3: minutes units (0-9)
- Byte 7: Period 1 left player seconds
  - Bit 4-6: tens of seconds (0-5)
  - Bit 0-3: seconds units (0-9)
- Byte 8: Period 1 right player hours
  - Bit 0-3: hours (0-9)
- Byte 9: Period 1 right player minutes
  - Bit 4-6: tens of minutes (0-5)
  - Bit 0-3: minutes units (0-9)
- Byte 10: Period 1 right player seconds
  - Bit 4-6: tens of seconds (0-5)
  - Bit 0-3: seconds units (0-9)
- Byte 11: Count up or down
  - Bit 2-3: right player
    - 0 = no counting
    - 1 = down count
    - 2 = up count
  - Bit 0-1: left player
    - 0 = no counting
    - 1 = down count
    - 2 = up count
- Byte 12: Checksum

20.6.7 I2C_CHANGESTATE (0x0b)

Purpose: Command to start, pause, reset or switch off the clock. Also used to select one of the clock options.
Note: Only valid when the clock mode is CENTRAL CONTROL (see chapter 20.4)
Response: I2C_ACKNOWLEDGE after processing the message

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: State
  - Bit 5-7: Option select
    - 0 = clock switches to OFF state independent of its current state.
    - 1 = select the option specified in bit 0-4
    - 2 = Clock starts running. Only accepted in clock state WAIT FOR FIRST START or PAUSE
    - 3 = clock pauses. Only accepted in clock state RUN or BLOCKED ON FLAG.
    - 4 = forces a reset of the clock. The manually set programs will remain.
  - Bit 0-4: Option number (1-30)
    - ***Note:*** Only valid when option select = 1
- Byte 6: Checksum

20.6.8 I2C_SENDHELLO (0x0c)

Purpose: Message to verify the presence of a device. Can also be sent by the clock.
Response: I2C_SENDHELLO to the source address

Message format:

- Byte 1-5: I²C header and Checksum (see chapter 20.2)

20.6.9 I2C_PING (0x0d)

Purpose: Message to verify the presence of the clock
Note: When using destination address 0x50 (DGT3000 address + 0x40 offset) the clock will switch on when it was off.
Response: none

Message format:

- Byte 1-5: I²C header and Checksum (see chapter 20.2)

20.6.10 I2C_TIMECORRECTION (0x0e)

Purpose: Command to change the time, moves and period by the computer.

Note: Only valid when the clock mode is CENTRAL CONTROL (see chapter 20.4). The clock state must be PAUSED or WAIT FOR FIRST START also (see chapter 20.3).

Response: I2C_ACKNOWLEDGE after processing the message

Message format:

- Byte 1-4: I^2C header (see chapter 20.2)
- Byte 5: Left player hours
  - Bit 0-3: hours (0-9)
- Byte 6: Left player minutes
  - Bit 4-6: tens of minutes (0-5)
  - Bit 0-3: minutes units (0-9)
- Byte 7: Left player seconds
  - Bit 4-6: tens of seconds (0-5)
  - Bit 0-3: seconds units (0-9)
- Byte 8: Right player hours
  - Bit 0-3: hours (0-9)
- Byte 9: Right player minutes
  - Bit 4-6: tens of minutes (0-5)
  - Bit 0-3: minutes units (0-9)
- Byte 10: Right player seconds
  - Bit 4-6: tens of seconds (0-5)
  - Bit 0-3: seconds units (0-9)
- Byte 11: White position and hundreds of moves
  - Bit 4: white position
    - 0 = right
    - 1 = left
  - Bit 0-3: white moves (0-9); hundreds of moves
    - The black player move count is derived from the below white player move count, combining lever position and black/white icons.
- Byte 12: White moves
  - Bit 4-7: tens of moves (0-9)
  - Bit 0-3: moves units (0-9)
- Byte 13: Period
  - Bit 4-6: left player period (1-5)
  - Bit 0-2: right player period (1-5)
  - ***Remark:*** In case of Fischer with move counter transients, the value of the move counter has priority. That value forces the period number, the period value is not used.
- Byte 14: Checksum

20.6.11 I2C_SETCENTRALCONTROL (0x0f)

Purpose: Message to put the clock in CENTRAL CONTROL mode
Response: I2C_ACKNOWLEDGE after processing the message

Message format:

- Byte 1-5: I²C header and Checksum (see chapter 20.2)

20.6.12 I2C_RELEASECENTRALCONTROL (0x10)

Purpose: Message to release the clock from CENTRAL CONTROL mode
Response: I2C_ACKNOWLEDGE after processing the message

Message format:

- Byte 1-5: I²C header and Checksum (see chapter 20.2)

20.6.13 I2C_SERIAL (0x20)

Purpose: The serial command is used to transfer serial data between any device and the e-Board.

Note 1: The content of the serial data is one of the commands as described in the single board commands (chapters 13 and 14) and the bus commands (chapters 16 and 17).

Note 2: The I²C packet size (from device to board) has a maximum of128 bytes. So, a maximum of 123 bytes can be transferred in one serial packet. When the device can't handle any more serial packets it should use the hardware handshake of the I²C-bus. The other device should retry 3 times and not send any other serial commands in the meantime.

Note 3: This command is used e.g. when using nonstandard devices e..g. to create a wireless network.

Response: I2C_ACKNOWLEDGE after processing the message or just the replied message (I2C_SERIAL)

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: First byte of serial packet
- :
- Byte n+4: Last byte of serial packet
- Byte n+5: Checksum

Note: n= size of the serial packet

20.7 I²C messages (clock to computer via board)

20.7.1 I2C_ACKNOWLEDGE (0x01)

Reply on: Any I²C command
Response: Sent by clock after successful processing of a command, or after not successful processing of a command.

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: Acknowledged command
- Byte 6: Acknowledge result
  - Bit 0-2: Ack result (see below):
  - Bit 3: Central control
    - 0 = Cleared
    - 1 = Active
  - Bit 4-6: Ack type (see below):
- Byte 7: State value of the clock during processing of the command (see 20.3)
- Byte 8: Checksum

Ack result	Description
0	OK
1	Clock not in correct state for executing command
2	Reserved
3	Clock busy with executing other command
4	Inconsistent data in message
5	Clock already in requested state.

Ack type	Description
0	Acknowledgement on non-I2C_DISPLAY messages
1	Acknowledge that display is free
2	Acknowledge that display is occupied

20.7.2 I2C_HELLO (0x02)

Purpose: Broadcast command from clock to board
Response: The receiver does not send an acknowledge message.

Message format:

- Byte 1-4: I²C header (see chapter 20.2)
- Byte 5: Software version of the clock:
  - Bit 4-7: Main version
  - Bit 0-3: Sub version
- Byte 6: Hardware version of the clock
- Byte 7: Checksum

20.7.3 I2C_TIME (0x04)

Purpose: Send the clock information (see message content below) as a broadcast message (receiver address zero (0)).
The clock information is sent:

- Every 1s when clock is in running mode
- Every 2s (as a heartbeat) when clock is other mode than running mode
- On lever change, when the clock is in running mode - except when in option (program) 'computer/internet use'.
- When a transient to the next period occurs
- The moment a player's flag becomes active (period flag or final flag)
- The moment the clock is switched between running, pause, wait for start.

Response: The receiver does not send an acknowledge message.

Message format:

- Byte 1-4: I^2C header (see chapter 20.2)
- Byte 5: Indicator valid time:
  - 0 = No valid time; bytes 6 to 17 have no meaning
  - 1 = Valid time available in bytes 6 to 17
- Byte 6: Left player method and hours:
  - Bit 4-7: Method (see below)
  - Bit 0-3: Hours (0-9)
- Byte 7: Left player minutes
  - Bit 4-6: Tens of minutes (0-5)
  - Bit 0-3: Minute units (0-9)
- Byte 8: Left player seconds
  - Bit 4-6: Tens of seconds (0-5)
  - Bit 0-3: seconds units (0-9)
- Byte 9: Left player period number and sub-seconds
  - Bit 4-6: Period number (1-5)
  - Bit 0-3: Sub-seconds (0-15) in 1/16 seconds
- Byte 10: Left player color and hundreds of moves
  - Bit 4: Color
    - 0 = Black
    - 1 = white
  - Bit 0-3: Hundreds moves (0-9)
- Byte 11: Left player moves
  - Bit 4-7: Tens of moves (0-9)
  - Bit 0-3: Moves units (0-9)
- Byte 12: Right player method and hours
  - Bit 4-7: Method (see below)
  - Bit 0-3: Hours (0-9)
- Byte 13: Right player minutes

- Bit 4-6: Tens of minutes (0-5)
- Bit 0-3: Minute units (0-9)

- Byte 14: Right player seconds
  - Bit 4-6: Tens of seconds (0-5)
  - Bit 0-3: Seconds units (0-9)
- Byte 15: Right player period number and sub-seconds
  - Bit 4-6: Period number (1-5)
  - Bit 0-3: Sub-seconds (0-15) in 1/16 seconds
- Byte 16: Right player color and hundreds of moves
  - Bit 4: Color
    - 0 = Black
    - 1 = white
  - Bit 0-3: Hundreds moves (0-9)
- Byte 17: Right player moves
  - Bit 4-7: Tens of moves (0-9)
  - Bit 0-3: Moves units (0-9)
- Byte 18: Option
  - Bit 0-4: Selected option (1-30); See manual of the clock.
- Byte 19: State id
  - Bit 0-7: Current state (1-85); see Clock states (chapter 20.3).
- Byte 20: Status flags
  - Bit 7: Buzzer
    - 0 = buzzer off
    - 1 = buzzer on
  - Bit 6: Freeze
    - 0 = no freeze
    - 1 = one of the clock reaches 0
  - Bit 4-5: Right flag
    - 0 = no
    - 1 = visible
    - 2 = final (blinking)
  - Bit 2-3: Left flag
    - 0 = no
    - 1 = visible
    - 2 = final (blinking)
  - Bit 1: Low battery
    - 0 = battery is ok
    - 1 = battery voltage is low
  - Bit 0: Lever position
    - 0 = left side up
    - 1 = right side up
- Byte 21: Time message changed
  - Bit 0: Time changed
    - 0 = a time change initiated the sending of the message
    - 1 = no changes found since 2 seconds after the previous sent message. The time message then acts as a 2 seconds heartbeat, allowing instant receive of time information by a slave, also when the clock is i.e. paused.

- Byte 22: Software version
  - Bit 4-7: Main version
  - Bit 0-3: Sub version
    - 1= overload error fixed
    - 2 = added lowercase font
- Byte 23: Software build number
  - Bit 0-6: Sequential number of the Software version
- Byte 24: Checksum

Code	Method
0	Time
1	Fischer
2	Delay
3	US delay
4	Byo-yomi
5	Canadian byo-yomi
6	Up-count
7	End (no further period defined)
8	Set and run

20.7.4 I2C_BUTTON (0x05)

Purpose: Send the state of the buttons, low battery and lever at every change. (see message content below) as a broadcast message (receiver address 0).
Also the current state of the clock is sent
The new state of the buttons, LowBat and Lever is sent at every change. Also a STATEID variable is sent. Always sent with destination address 0 (General Call)

Response:

Message format:

- Byte 1-4: I^2C header (see chapter 20.2)
- Byte 5: New button state
  - Bit 7: Low battery
    - 0 = battery is ok
    - 1 = battery voltage is low
  - Bit 6: Lever
    - 0 = left side up
    - 1 = right side up
  - Bit 5: On/off button
    - 0 = released
    - 1 = pressed
  - Bit 4: Forward button (as On/off button)
  - Bit 3: Plus button (as On/off button)
  - Bit 2: Play/pause button (as On/off button)
  - Bit 1: Minus button (as On/off button)

  - Bit 0: Back button (as On/off button)
- Byte 6: Old button state
  - Bit 0-7: As New button state
- Byte 7: State id
  - Bit 0-7: Current state (1-85); see Clock states (chapter 20.3)
- Byte 8: Variable value, meaning depends on byte 5:
  - When a button is pressed the old value of the variable currently edited is given
  - When a button is released the new value of the variable currently edited is given
- Byte 9: Elapsed time
  - Duration of buttons being pressed in 100 milliseconds; maximum of 25,5 seconds (255)
- Byte 10: Checksum

20.7.5 I2C_SENDHELLO (0x0c)

Purpose: Message to verify the presence of another I²C device
Response: The receiver will send an I2C_SENDHELLO message in return

Message format:

- Byte 1-5: I²C header and Checksum (see chapter 20.2)

20.7.6 I2C_PING (0x0d)

Purpose: Message to verify the presence of another I²C device
Response: The receiver will send an acknowledge command

Message format:

- Byte 1-5: I²C header and Checksum (see chapter 20.2)

21 Clock commands – SBI

21.1 General

Communication between an e-Board and some (old) DGT clocks is based on SBI communication (see chapter 3.2). Communication of a computer with the clock is based on communication with the e-Board, in which the SBI-commands and -messages are integrated.

In this chapter the clock states and clock functions are described.

Only the functions used for communication with a computer via the DGT electronic boards are described.

21.2 SBI messages (computer to clock via board)

21.2.1 DGT_SBI_CLOCK_MESSAGE (0x2b)

This is the general message from a computer to the clock (via board)

Constants:

- #define DGT_CMD_CLOCK_START_MESSAGE 0x03
- #define DGT_CMD_CLOCK_END_MESSAGE 0x00

Message format:

- byte 1: DGT_SBI_CLOCK_MESSAGE
- byte 2: Size: number of bytes to follow
- byte 3: DGT_CMD_CLOCK_START_MESSAGE
- byte 4: Clock command; see definitions in the next paragraphs
- byte 5-(n+1): Data
- byte n+2: DGT_CMD_CLOCK_END_MESSAGE

  Note: n = size

21.2.2 DGT_CMD_CLOCK_DISPLAY (0x01)

Purpose:	This command can control the segments of six 7-segment characters, two dots, two semicolons and the two '1' symbols. These are arranged as follows: "1A:BC 1D:EF" and thus capable of showing clock times or move text of up to 6 chars (the dots and '1' excluded). "1A:BC 1D:EF" : the A..F are six characters with 7 segments, the '1' is a single segment and the ':' has both a dot and semicolon segment.
Response:	Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: 'C' location segments. See table below
- byte 6: 'B' location segments.
- byte 7: 'A' location segments.

- byte 8: 'F' location segments.
- byte 9: 'E' location segments.
- byte 10: 'D' location segments.
- byte 11: icons. See table below
- byte 12: Beep (= 0x03) or no beep (= 0x01)
- byte 13: DGT_CMD_CLOCK_END_MESSAGE

Bit	Location segment	Icon
0	Top segment	Right dot
1	Right top	Right semicolon
2	Right bottom	Right '1'
3	Bottom	Left dot
4	Left Bottom	Left semicolon
5	Left top	Left '1'
6	Center segment	n.a.
7	n.a.	n.a.

21.2.3 DGT_CMD_CLOCK_ICONS (0x02)

Purpose: Used to control the clock icons like flags etc.
Note: For the DGT XL only.
Response: Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: Icons left. See table Icon - symbol-1 below
- byte 6: Icons right. See table Icon - symbol-1 below
- byte 7: Icons left. See table Icon - symbol-2 below
- byte 8: Icons right. See table Icon - symbol-2 below
- byte 9: Special (clr / condat). See table icon – symbol-3 below
- byte 10-12: Reserved (= 0x00)
- byte 13: DGT_CMD_CLOCK_END_MESSAGE

Bit	Icon – symbol-1	Icon – symbol-2	Icon – symbol-3
0	TIME	Period 1	Function: clear all icons
1	FISCH	Period 2	Sound
2	DELAY	Period 3	Black-white
3	HGLASS	Period 4	White-black
4	UPCNT	Period 5	Battery
5	BYO	Flag	Sound
6	END	n.a.	Display pool selec 0 = icons clear after time change 1 = icons stay visible
7	n.a.	n.a.	n.a.

21.2.4 DGT_CMD_CLOCK_END (0x03)

Purpose: This command clears the message and brings the clock back to the normal display (showing clock times).
Response: Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: DGT_CMD_CLOCK_END_MESSAGE

21.2.5 DGT_CMD_CLOCK_BUTTON (0x08)

Purpose: Requests the current button pressed (if any).
Response: Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: DGT_CMD_CLOCK_END_MESSAGE

21.2.6 DGT_CMD_CLOCK_VERSION (0x09)

Purpose: This command requests the clock version.
Response: Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: DGT_CMD_CLOCK_END_MESSAGE

21.2.7 DGT_CMD_CLOCK_SETNRUN (0x0a)

Purpose: This command controls the clock times and counting direction, when the clock is in option (program) 'computer/internet use'. A clock can be paused or counting down.
Note 1: Counting up isn't supported at all versions (Version 1.14 and lower).
Note 2: Option/internet use for DGT XL: Option 23
Response: Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: left hours (|0x10 if left counts up)
- byte 6: left minutes
- byte 7: left seconds
- byte 8: right hours (|0x10 if right counts up)
- byte 9: right minutes

- byte 10: right seconds
- byte 11: Control function. See table below.
- byte 12: DGT_CMD_CLOCK_END_MESSAGE

Bit	Control function
0	Left counts down
1	Right counts down
2	Pause clock
3	Toggle player at lever change
4	n.a.
5	n.a.
6	n.a.
7	n.a.

21.2.8 DGT_CMD_CLOCK_BEEP (0x0b)

Purpose: This clock command turns the beep on, for a specified time
Response: Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

This clock command turns the beep on, for a specified time

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: The time in multiplies of 64 milliseconds. (E.g. value=16, beep = 16*64=1024 ms)
- byte 6: DGT_CMD_CLOCK_END_MESSAGE

21.2.9 DGT_CMD_CLOCK_ASCII (0x0c)

Purpose: This clock commands sends a ASCII message to the clock that can be displayed only by the DGT3000.
Response: Acknowledged by DGT_MSG_SBI_CLOCK (see chapter 21.3.1)

Message format:

- byte 1-4: SBI Header (see chapter 21.2.1)
- byte 5: Character 1
- byte 6: Character 2
- byte 7: Character 3
- byte 8: Character 4
- byte 9: Character 5
- byte 10: Character 6
- byte 11: Character 7
- byte 12: Character 8
- byte 13: Beep value:
  - 0: No beep
  - 1-15: 62.5 milliseconds + (value/16) seconds
- byte 14: DGT_CMD_CLOCK_END_MESSAGE

Note: Characters, see DGT3000 ASCII table, chapter 27.

21.3 SBI messages (clock via board to computer)

21.3.1 DGT_MSG_SBI_CLOCK (0x8d)

There are two possible distinct DGT_MSG_SBI_CLOCK messages:

1. Message with clock data
2. Message with ack

The total size is always 10 bytes and the first byte is always DGT_MSG_SBI_CLOCK (=0x8d). If the (5th byte & 0x0f) equals 0x0a, or if the (8th byte & 0x0f) equals 0x0a, then the message is a Clock Ack message. Otherwise it is a Clock Data message (see next chapters).

21.3.2 DGT_MSG_SBI_CLOCK - Clock Data

Purpose: Send the actual clock data
Response: --

Message format:

- byte 1: DGT_MSG_SBI_CLOCK
- byte 2: LLH_SEVEN(message length) (=0 fixed)
- byte 3: LLL_SEVEN(message length) (=10 fixed)
- byte 4: Bit 0-3: Right player - hours (0-9, BCD coded)
  Bit 4: Right player – flag situation 1
    - 1 = Flag fallen for right player, and clock blocked to zero,
    - 0 = not the above situation

  Bit 5: Right player – time indicator
    - 1 = Time per move indicator on (i.e. Bronstein, Fischer)
    - 0 = Time per move indicator off for right player

  Bit6: Right player – flag situation 2
    - 1 = Right players flag fallen and indicated on display, clock possibly still running (e.g. activation of next time period)
    - 0 = not the above situation
- byte 5: Minutes (0-59, BCD coded)
- byte 6: Seconds (0-59, BCD coded)
- byte 7-9: Left player; similar to byte 4-6
- byte 10: Clock status:
  Bit 0: Clock running state
    - 1 = Clock running
    - 0 = Clock stopped by Start/Stop

  Bit 1: Right player leveler position
    - 1 = leveler position high on right player
    - 0 = leveler position high on left player

  Bit 2: Battery low
    - 1 = Battery low indication on display
    - 0 = no battery low indication on display

  Bit 3: Left player's turn

- 1 = Left player's turn
- 0 = not left player's turn

Bit 4: Right player's turn
- 1 = Right player's turn
- 0 = not right player's turn

Bit 5: No clock, reading invalid
- 1 = No clock connected; reading invalid
- 0 = clock connected, reading valid

Bit 6: not used (read as 0)
Bit 7: Always 0

Note: LLH_SEVEN and LLL_SEVEN, see chapter 8.1

21.3.3 DGT_MSG_SBI_CLOCK - Clock Ack

Purpose: Acknowledge of the command sent to the clock, including requested data.
Response: --

Message format:

- byte 1: DGT_MSG_SBI_CLOCK
- byte 2: LLH_SEVEN(message length) (=0 fixed)
- byte 3: LLL_SEVEN(message length) (=10 fixed)
- byte 4-10: used to construct a 4-byte ack message. See below.

4-byte ack message:

- ack0 = ((byte 5) & 0x7f) | ((byte 7 << 3) & 0x80)
- ack1 = ((byte 6) & 0x7f) | ((byte 7 << 2) & 0x80)
- ack2 = ((byte 8) & 0x7f) | ((byte 4 << 3) & 0x80)
- ack3 = ((byte 9) & 0x7f) | ((byte 4 << 2) & 0x80)

So ack0 is constructed from bits 0-6 from byte5, and bit 4 from byte7 (which goes to ack0's 7th bit).

ack0

Value	Description
0x40	Error. E.g. sending a CMD_CLOCK_SETNRUN while the clock is not in option 'computer/internet use'
0x10	Normal ack message

ack1

If the 8th bit is set, then it is an auto-generated ack, otherwise it is a response to a command

Value	Description
0x01	Display ack
0x08	Buttons ack, but no button information is returned though
0x09	Version ack. Version in ack2: ack2>>4 is main version, ack2&0x0f is sub version
0x0a	SETNRUN ack
0x0b	Beep ack
0x0c	Display ASCII ack
0x81	Ready

Value	Description
0x88	Button pressed, result in ack3 – button pressed (see below)
0x8a	When option (program) is 'computer/internet use', ack3 contains the clock mode
0x90	Not in option (program) 'computer/internet use'

Ack3

Value	Button pressed
0x31	BACK / PREVIOUS
0x32	PLUS
0x33	RUN
0x34	MINUS
0x35	OK / ADJUST

22 Summary of single board commands and messages

22.1 Single board command codes from computer to board without data

			Supported by				
			Serial USB Type-B mini	BT	Rev-II	USB Type-C	Centaur
Command code	Command name	Response	SW V1.x	SW V3.x	SW V0.6	SW V4.x	SW V5.x
0x40	DGT_REQ_RESET	None	Yes			Yes	
0x41	DGT_REQ_SBI_CLOCK	DGT_MSG_SBI_CLOCK	Yes	Yes	Yes	Yes	
0x42	DGT_REQ_BOARD	DGT_MSG_BOARD_DUMP	Yes	Yes	Yes	Yes	
0x43	DGT_REQ_UPDATE_SBI	DGT_MSG_FIELD_UPDATE DGT_MSG_SBI_CLOCK	Yes	Yes	Yes	Yes	
0x44	DGT_REQ_UPDATE_BOARD	DGT_MSG_FIELD_UPDATE	Yes	Yes	Yes	Yes	
0x45	DGT_REQ_SERIALNR	DGT_MSG_SERIALNR	Yes	Yes	Yes	Yes	
0x46	DGT_REQ_BUSADDRESS	DGT_MSG_BUSSADDRESS	Yes	Yes	Yes	Yes	
0x47	DGT_REQ_TRADEMARK	DGT_MSG_TRADEMARK	Yes	Yes	Yes	Yes	
0x48	DGT_REQ_HARDWARE_VERSION	DGT_MSG_HARDWARE_VERSION	No	No	No	ToDo	Yes
0x49	DGT_REQ_LOG_MOVES	DGT_MSG_LOG_MOVES	Yes	Yes	Yes	Yes	
0x4a	DGT_TO_BUSMODE	None	Yes	Yes		Yes	
0x4b	DGT_REQ_UPDATE_SBI_NICE	DGT_MSG_FIELD_UPDATE DGT_MSG_SBI_CLOCK	Yes	Yes		Yes	
0x4c	DGT_REQ_BATTERY_STATUS	DGT_MSG_BATTERY_STATUS	No	Yes		No	
0x4d	DGT_REQ_VERSION	DGT_MSG_VERSION	Yes	Yes		Yes	
0x4e	DGT internal use						
0x55	DGT_REQ_LONG_SERIALNR	DGT_MSG_LONG_SERIALNR	No	Yes		Yes	
0x56	DGT_REQ_I2C_CLOCK	DGT_MSG_I2C_CLOCK	No	No	No	Yes	
0x57	DGT_REQ_UPDATE_I2C	DGT_MSG_FIELD_UPDATE	No	No	No	Yes	

		DGT_MSG_I2C_CLOCK					
0x58	DGT_REQ_UPDATE_I2C_NICE	DGT_MSG_FIELD_UPDATE DGT_MSG_I2C_CLOCK	No	No	No	Yes	

22.2 Single board command codes from computer to board with data

			Supported by				
			Serial USB Type-B mini	BT	Rev-II	Type-C	Centaur
Command code	**Command name**	**Response**	**SW V1.x**	**SW V3.x**	**SW V0.6**	**SW V4.x**	**SW V5.x**
0x2b	DGT_SBI_CLOCK_MESSAGE	DGT_MSG_SBI_CLOCK or None	Yes	Yes	Yes	Yes	
0x2c	DGT_I2C_CLOCK_MESSAGE	DGT_MSG_I2C_CLOCK	Don't	Don't	Don't	Yes	
0x60	DGT_SET_LEDS	None	Don't	Don't	Yes	No	
0x61	DGT internal use						

22.3 Single board message codes from board to computer

Message code	Command name
0x86	DGT_MSG_BOARD_DUMP
0x8d	DGT_MSG_SBI_CLOCK
0x8e	DGT_MSG_FIELD_UPDATE
0x8f	DGT_MSG_LOG_MOVES
0x90	DGT_MSG_BUSADDRESS
0x91	DGT_MSG_SERIALNR
0x92	DGT_MSG_TRADEMARK
0x93	DGT_MSG_VERSION
0x96	DGT_MSG_HARDWARE_VERSION
0xa0	DGT_MSG_BATTERY_STATUS
0xa2	DGT_MSG_LONG_SERIALNR
0xa3	DGT_MSG_I2C_CLOCK

23 Summary of bus commands and messages

23.1 Bus command codes from computer to board without data

			Supported by				
			Serial USB Type-B mini	BT	Rev-II	Type-C	Centaur
Command code	Command name	Response	SW V1.x	SW V3.x	SW V0.6	SW V4.x	SW V5.x
0x81	DGT_BUS_REQ_SBI_CLOCK	DGT_MSG_BUS_SBI_CLOCK	Yes	Yes	Yes	Yes	
0x82	DGT_BUS_REQ_BOARD	DGT_MSG_BUS_BOARD_DUMP	Yes	Yes	Yes	Yes	
0x83	DGT_BUS_REQ_CHANGES	DGT_MSG_BUS_UPDATE_ODD	Yes	Yes	Yes	Yes	
0x84	DGT_BUS_REPEAT_CHANGES	DGT_MSG_BUS_UPDATE_ODD	Yes	Yes	Yes	Yes	
0x85	DGT_BUS_SET_START_GAME	DGT_MSG_BUS_START_GAME_WRITTEN	Yes	Yes	Yes	Yes	
0x86	DGT_BUS_REQ_FROM_START	DGT_MSG_BUS_FROM_START	Yes	Yes	Yes	Yes	
0x87	DGT_BUS_PING	DGT_MSG_BUS_PING	Yes	Yes	Yes	Yes	
0x88	DGT_BUS_END_BUSMODE	None	Yes	Yes	Yes	Yes	
0x89	DGT_BUS_RESET	None	Yes	Yes	Yes	Yes	
0x8a	DGT_BUS_IGNORE_NEXT_BUS_PING	DGT_MSG_BUS_PING	Yes	Yes	Yes	Yes	
0x8b	DGT_BUS_REQ_VERSION	DGT_MSG_BUS_VERSION	Yes	Yes	Yes	Yes	
0x8d	DGT_BUS_REQ_ALL_D	DGT_MSG_BUS_UPDATE_ODD DGT_MSG_BUS_SBI_CLOCK DGT_MSG_BUS_BOARD_DUMP	Yes	Yes	Yes	Yes	
0x91	DGT_BUS_REQ_SERIALNR	DGT_MSG_BUS_SERIALNR	No	No	No	Yes	
0x92	DGT_BUS_RPING	DGT_MSG_BUS_PING	No	No	No	Yes	
0x93	DGT_BUS_REQ_I2C_CLOCK	DGT_MSG_BUS_I2C (clock)	No	No	No	Yes	
0x94	DGT_BUS_REQ_I2C_BUTTON	DGT_MSG_BUS_I2C (button)	No	No	No	Yes	
0x95	DGT_BUS_REQ_I2C_ACK	DGT_MSG_BUS_I2C (ack)	No	No	No	Yes	
0x96	DGT_BUS_REQ_STATS	DGT_MSG_BUS_STATS	No	No	No	Yes	
0x97	DGT_BUS_REQ_TRADEMARK	DGT_MSG_BUS_TRADEMARK	No	No	No	Yes	
0x98	DGT_BUS_REQ_BATTERY	DGT_MSG_BUS_BATTERY_STATUS	No	No	No	No	Yes

			Supported by				
			Serial USB Type-B mini	BT	Rev-II	Type-C	Centaur
Command code	Command name	Response	SW V1.x	SW V3.x	SW V0.6	SW V4.x	SW V5.x
0x99	DGT_BUS_REPEAT_I2C_BUTTON	DGT_MSG_BUS_I2C (button)	No	No	No	To do	Yes
0x9a	DGT_BUS_REPEAT_I2C_ACK	DGT_MSG_BUS_I2C (ack)	No	No	No	To do	Yes
0x9b	DGT_BUS_REQ_HARDWARE_VERSION	DGT_MSG_BUS_HARDWARE_VERSION	No	No	No	To do	Yes

23.2 Bus command codes from computer to board containing data

			Supported by				
			Serial USB Type-B mini	BT	Rev-II	Type-C	Centaur
Command code	Command name	Response	SW V1.x	SW V3.x	SW V0.6	SW V4.x	SW V5.x
0xa0	DGT_BUS_CONFIG	DGT_MSG_BUS_CONFIG	No	No	No	Yes	
0xa1	DGT_BUS_SBI_CLOCK_MESSAGE	DGT_MSG_BUS_SBI_CLOCK	No	No	No	Yes	
0xa2	DGT_BUS_I2C_CLOCK_MESSAGE	DGT_MSG_BUS_I2C	No	No	No	Yes	
0xb0	DGT internal use						
0xb1	DGT internal use						
0xb2	DGT internal use						
0xb3	DGT internal use						

23.3 Bus message codes from board to computer

Message code	Command name
0x83	DGT_MSG_BUS_BOARD_DUMP
0x84	DGT_MSG_BUS_SBI_CLOCK
0x85	DGT_MSG_BUS_UPDATE_ODD
0x86	DGT_MSG_BUS_FROM_START
0x87	DGT_MSG_BUS_PING
0x88	DGT_MSG_BUS_START_GAME_WRITTEN
0x89	DGT_MSG_BUS_VERSION
~~0x8b~~	~~DGT_MSG_BUS_UPDATE_EVEN~~ (not used)
0x8c	DGT_MSG_BUS_HARDWARE_VERSION
0xb0	DGT_MSG_BUS_SERIALNR
0xb1	DGT_MSG_BUS_I2C
0xb2	DGT_MSG_BUS_STATS
0xb3	DGT_MSG_BUS_CONFIG
0xb4	DGT_MSG_BUS_TRADEMARK
0xb5	DGT_MSG_BUS_BATTERY_STATUS

24 Summary of the logfile message codes

Code	Description / name
0x00	LOG_NOP2
0x01 - 0xf	LOG_SUBSEC_DELAY
0x10 - 0x1f	LOG_SEC_DELAY
0x20 - 0x2f	LOG_MIN_DELAY
0x30	LOG_HOUR_DELAY
0x40 - 0x4f	LOG_FIELD_UPDATE
0x50 - 0x5f	LOG_FIELD_UPDATE_AS
0x60 - 0x69	LOG_CLOCK_UPDATE_R
0x6a	LOG_POWERUP
0x6b	LOG_EOF
0x6c	LOG_FOURROWS
0x6d	LOG_EMPTY_BOARD
0x6e	LOG_DOWNLOADED
0x6f	LOG_START_POS
0x70 - 0x79	LOG_CLOCK_UPDATE_L
0x7a - 0x79	LOG_START_POS_ROT
0x7b	LOG_START_TAG
0x7c	LOG_DEBUG
0x7d	LOG_NEW_CLOCK_STATE
0x7e	LOG_NEW_BUS_ADDRESS
0xff	LOG_EMPTY

25 Summary of the I²C messages

Command code	Description / name
0x01	I2C_ACKNOWLEDGE
0x02	I2C_HELLO
0x03	I2C_DEBUG
0x04	I2C_TIME
0x05	I2C_BUTTON
0x06	I2C_DISPLAY
0x07	I2C_END_DISPLAY
0x08	I2C_CURRENTPROGRAM
0x09	I2C_PROGRAM
0x0a	I2C_SETNRUN
0x0b	I2C_CHANGESTATE
0x0c	I2C_SENDHELLO
0x0d	I2C_PING
0x0e	I2C_TIMECORRECTION
0x0f	I2C_SETCENTRALCONTROL
0x10	I2C_RELEASECENTRALCONTROL
0x11	I2C_TRIGGERBOOTLOADER
0x20	I2C_SERIAL

26 Summary of the SBI messages

Command code	Description / name
0x01	DGT_CMD_CLOCK_DISPLAY
0x02	DGT_CMD_CLOCK_ICONS
0x03	DGT_CMD_CLOCK_END
0x08	DGT_CMD_CLOCK_BUTTON
0x09	DGT_CMD_CLOCK_VERSION
0x0a	DGT_CMD_CLOCK_SETNRUN
0x0b	DGT_CMD_CLOCK_BEEP
0x0c	DGT_CMD_CLOCK_ASCII
0x2b	DGT_SBI_CLOCK_MESSAGE
0x8d	DGT_MSG_SBI_CLOCK

27 DGT3000 ASCII table (14-segment display)

hex	char	DGT3000	hex	char	DGT3000	hex	char	DGT3000
0x20	SPACE		0x40	@		0x60	`	
0x21	!		0x41	A		0x61	a	
0x22	“		0x42	B		0x62	b	
0x23	#		0x43	C		0x63	c	
0x24	$		0x44	D		0x64	d	
0x25	%		0x45	E		0x65	e	
0x26	&		0x46	F		0x66	f	
0x27	‘		0x47	G		0x67	g	
0x28	(		0x48	H		0x68	h	
0x29	)		0x49	I		0x69	i	
0x2a	*		0x4a	J		0x6a	j	
0x2b	+		0x4b	K		0x6b	k	
0x2c	,		0x4c	L		0x6c	l	
0x2d	-		0x4d	M		0x6d	m	
0x2e	.		0x4e	N		0x6e	n	
0x2f	/		0x4f	O		0x6f	o	
0x30	0		0x50	P		0x70	p	
0x31	1		0x51	Q		0x71	q	
0x32	2		0x52	R		0x72	r	
0x33	3		0x53	S		0x73	s	
0x34	4		0x54	T		0x74	t	
0x35	5		0x55	U		0x75	u	
0x36	6		0x56	V		0x76	v	
0x37	7		0x57	W		0x77	w	
0x38	8		0x58	X		0x78	x	
0x39	9		0x59	Y		0x79	y	
0x3a	:		0x5a	Z		0x7a	z	
0x3b	;		0x5b	[		0x7b	{	
0x3c	<		0x5c	\		0x7c	|	
0x3d	=		0x5d	]		0x7d	}	
0x3e	>		0x5e	^		0x7e	~	
0x3f	?		0x5f	_		0x7f	DEL	

Note: Character 0x20 (SPACE): all segments are dimmed