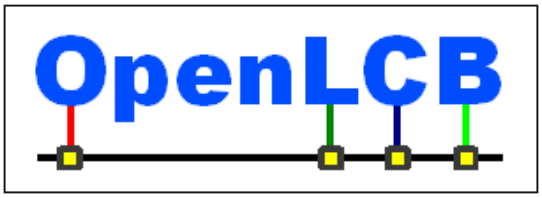

OpenLCB Standard	
Time Broadcast Protocol	
Apr 7, 2013	Preliminary

1 Introduction (Informative)

A layout control bus can do a number of useful things with fast-time information:

- Connect a number of clock displays to keep them synchronized.
- Provide time displays on plug-in devices, e.g. throttles.
- Provide cueing for time-based occurrences, such as lights turning on and off at specific modeled times.

Generally, existing fast clock systems have one unit that produces time information, here called a clock generator, and one or more units that consume it. Existing fast clock systems typically only report minutes, not seconds or finer time divisions. Some existing fast clock systems track a day/date, in addition to time.

Fast clocks run at various rates, and can be controlled by the user either at the clock generator or from other locations. Some fast clock systems broadcast run/stop and rate information, which can also be useful when interpolating within a fast-minute.

OpenLCB broadcasts time information by producing Event IDs. Specific Event IDs correspond to specific times with the day, for example “08:10”, so that consumers can be taught to react to time-of-day. The year and date are handled separately for those installations that choose to use it.

2 Intended Use (Informative)

The primary use of this information is to display it on clock faces around the layout.

Since remote control of the fast clock is desired, a protocol for setting using produced and consumed events is defined. This makes it possible for throttles and other user-interface nodes to have a general fast-clock-control capability built in.

In addition, simple nodes can use specific EventIDs to trigger their actions at specific times. For example, lights in buildings in a model town can be sequenced to come on at various times by configuring consumers in a node to react to time events by changing output lines.

3 References and Context (Normative)

This specification is in the context of the following OpenLCB-CAN Standards:

- OpenLCB Event Transport Standard, which defines messages for transporting Event IDs and identifying producers and consumers.

- The OpenLCB Event Identifiers Standard, which defines the format and content of Event IDs including the class of Well-Known Event IDs and Automatically-Routed Event IDs.
- OpenLCB Unique Identifiers Standard, which defines the allocation of OpenLCB 48-bit unique identifiers

For more information on format and presentation, see:

- OpenLCB Common Information Technical Note

4 Message Formats (Normative)

This Standard defines a number of Event IDs.

The well-known event ID “Delivers Clock Protocol” is defined as 0x01.01.00.00.00.00.05.01.

The upper six bytes of the event IDs defined in the following subsections must be one of the following:

- 01.01.00.00.01.00 – referred to as “Default Fast Clock”
- 01.01.00.00.01.01 – referred to as “Default Real-time Clock”
- 01.01.00.00.01.02
- 01.01.00.00.01.03
- a valid unique ID under the control of the manufacturer of the clock generator node
- a valid Unique ID under the control of the person or organization configuring the clock generator node

The upper six bytes are referred to as “Specific Upper Part” in the subsections below.

4.1 Set/Report Time Event ID

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Specific Upper Part						Hours 0-23	Minutes 0-59

The upper nibble of byte 6 is only 0 or 1, which can be used to distinguish this format.

4.2 Set/Report Date Event ID

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Specific Upper Part						0x40+Month 0x41-0x4C	Day 1-31

The upper nibble of byte 6 is 4, which can be used to distinguish this format.

4.3 Set/Report Year Event ID

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Specific Upper Part						0x3000+Year 0x3000-0x3FFF	

The upper nibble of byte 6 is 3, which can be used to distinguish this format.

The lower twelve bits are the year, 0AD to 4095AD.

4.4 Set/Report Rate Event ID

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Specific Upper Part						0x5000+Rate	

The upper nibble of byte 6 is 5, which can be used to distinguish this format.

Rate is a 12 bit signed fixed point rrrrrrrrrr.rr, -511.75, -511.00, …, -1.00, …, -.025, 0.0, 0.25, 0.50, ..., 511.75

4.5 Stop/Start Clock

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Specific Upper Part						Stop 0x6001 Start 0x6002	

The upper nibble of byte 6 is 6, which can be used to distinguish this format.

5 States (Normative)

Each clock has an independent current time, independent running/stopped state and an independent rate.

When the clock is in stopped state, its internal time is not changing.

When the clock is in running state, its internal time is advancing (rate X) as fast as normal time.

6 Interactions (Normative)

The well-known event ID “Delivers Clock Protocol” shall be produced by every node when it first starts to operate as a clock generator. When an enquiry about “Delivers Clock Protocol” is received, the node has to return “valid” if the clock is operating, and “invalid” if it is not.

A Set/Report Rate event for which there are one or more consumers shall be produced every real 60 seconds while the clock is running.

The Set/Report Date event is produced when the date changes. The Set/Report Date event is produced when the year changes.

Clock producer nodes shall produce a Set/Report Date event if they receive one but do not update their internal date. Clock producer nodes shall produce a Set/Report Year event if they receive one but do not update their internal date.

A Set/Report Time event is produced every time the current time changes, e.g. every fast minute.

When a Set/Report Time event is received at a clock-producer node, it may, but is not required to, set the time in the clock producer node. If the time is not set, the current time event is produced immediately after.

If a Set/Report Rate event is received, the clock-producer's rate is set to the rate embedded in the event. The run/stop bits are ignored. If the clock does not support the requested rate, it moves to the closest non-zero supported rate, and produces a Set/Report Rate event containing the current rate. Rate can be set while the clock is running or stopped. The clock producer node may, but is not required to, produce a Set/Report Time immediately after the rate event is produced to ensure that all agree on the exact time.

6.1 Event Identification and Reporting

When a clock producer node receives an Identify Events message, the node shall reply with, in order:

- An IdentifyProducedEvent message for the well-known event ID “Delivers Clock Protocol”
- A ProducerRangeIdentified that covers the entire set of time event IDs
- A ConsumedRangeIdentified that covers the entire set of time event IDs
- An IdentifyProducedEvent message for the current start or stop state, showing valid & active
- An IdentifyProducedEvent message for the current rate, showing valid & active
- An IdentifyProducedEvent message for the current year showing valid & active
- An IdentifyProducedEvent message for the current date, showing valid & active
- An IdentifyProducedEvent message for the current time, showing valid & active

When a clock producer node receives an IdentifyProducers message that covers any of the events it handles (Set/Report Time, Set/Report Date, Set/Report Year, Set/Report Rate, Stop/Start Clock) it will reply with a ProducerIdentified message showing valid. If the queried event is the current state (same time, same date, same year, same rate & start/stop status, or same start/stop status respectively), the reply will be marked active. Otherwise, it will be marked inactive.

When a clock producer node receives an IdentifyConsumers message that covers any of the events it handles (Set/Report Time, Set/Report Date, Set/Report Year, Set/Report Rate, Stop/Start Clock) it will reply with a ConsumerIdentified message showing valid. If the queried event is the current state (same time, same date, same year, same rate & start/stop status, or same start/stop status respectively), the reply will be marked active. Otherwise, it will be marked inactive.

Table of Contents