ClimateTalk 2.0
Light Weight Protocol (CT-LWP) Specification

Document revision: 01
Release: June 12, 2013

Abstract

ClimateTalk is a universal language for innovative, cost-effective solutions that optimize performance, efficiency and home comfort. The ClimateTalk Open Standards define a set of messages and commands to enable interoperability, enhanced user interface, and machine to machine control independent of the physical layer connecting the devices.

This document defines the network, data link and physical layer requirements for CT-LWP which is a Light Weight Protocol that provides simplified structure for connecting peripheral devices in master-slave configuration and transmitting messages as defined by ClimateTalk Application Standards.

Updates

This specification may be updated at any time and may be superseded by a more recent version or amended to from time to time. Users should be certain they are using the current ClimateTalk version and the latest revision of the documents.

The released versions of all specifications are available at http://www.ClimateTalk.org

Version History

ClimateTalk Version	Document Revision	Release Date	Comments
V 1.0		2009-08-24	Initial Release
V 1.3		2011-11-02	Revised Formatting
V 2.0	00	2013-01-18	Version 2.0 Release – Increment version, no updates.
V 2.0	01	2013-06-12	No changes. Revision increment for package release.

Contributors

The following is a list of ClimateTalk Alliance member companies that were actively engaged in the development of this standard:

A.O. Smith Motor Company

A.O. Smith Water Products Company

Arzel Zoning Technology Inc.

Emerson Electric, Co.

Nidec Motor Company

Research Products Corp.

Rheem Manufacturing Company

Zonefirst

Table of Contents

List of Figures

List of Tables

1.0 Overview

1.1 ClimateTalk Model

ClimateTalk is a universal language for innovative, cost-effective solutions that optimize performance, efficiency and home comfort. The ClimateTalk Open Standards define a set of messages and commands to enable interoperability, enhanced user interface, and machine to machine control independent of the physical layer connecting the devices.

The messages and commands defined by ClimateTalk Information Model are the presentation and application layers as defined by the OSI Model[1]. ClimateTalk Applications are fully defined at Layer 7 of the OSI model by a combination of a Device Specific Application Profile, an Application Specification, and the Command Reference.

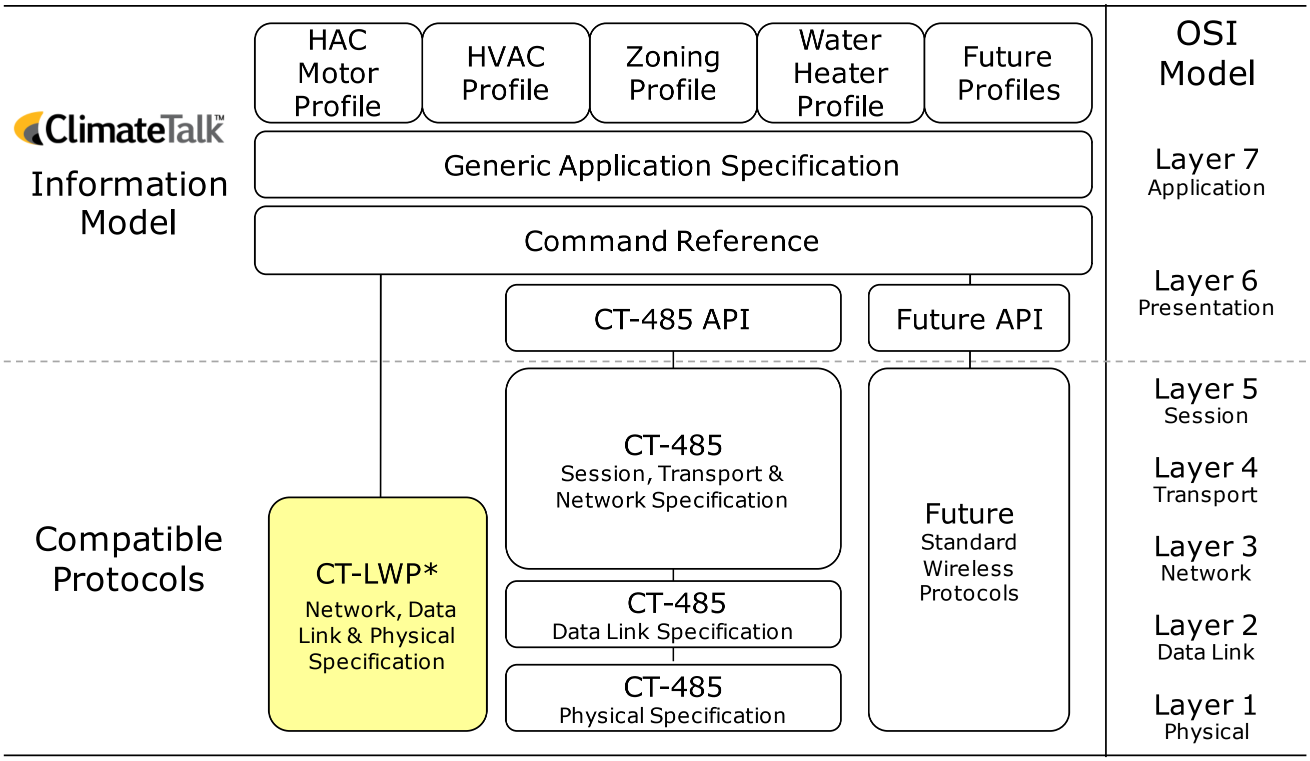
Below the application layer, ClimateTalk messages can be carried over any physical medium following the OSI model. CT-485 and CT-LWP are wired serial physical and network layers designed to support the formation of ClimateTalk networks and transport ClimateTalk messages, but other OSI based protocols –including wireless transports - can be used as well.

1.2 Scope

This document defines the network, data link and physical layer requirements for a Light Weight Protocol (CT-LWP), which corresponds to OSI Layers 1- 3. CT-LWP provides simplified structure for connecting peripheral devices in master-slave configuration and transmitting messages as defined by ClimateTalk Application Standards. See Figure 1: OSI Layers for a CT-LWP Implementation for a diagram of relevant standards.

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=20269

Figure 1: OSI Layers for a CT-LWP Implementation



**This Document*

The ClimateTalk Open Standards package shown in Figure 1: OSI Layers for a CT-LWP Implementation prescribes the mandatory requirements to ensure proper network formation of interoperable devices. Each device must comply with the mandatory requirements defined in this document as well as all other ClimateTalk standards applicable to the device functionality.

Membership in the ClimateTalk Alliance as well as successful completion of mandatory conformance testing is required for listing a product as a ClimateTalk Certified Device.

2.0 Normative References

A good understanding of the most recent version of the following documents is required to apply the contents of this specification correctly.

ClimateTalk HAC Motor Application Specification

ClimateTalk Generic Application Specification

ClimateTalk Command Reference

ClimateTalk CT-485 Application Protocol Interface

ClimateTalk CT-485 Networking Specification

ClimateTalk CT-485 Data Link Specification

ClimateTalk CT-485 Physical Specification

3.0 Terminology

3.1 Definitions

Master	Network node in charge of all communication functions. There can only be one master node per CT-LWP network.
Slave	Network nodes that only communicate upon initiation from a Master node. There can be up to 254 slave nodes per CT-LWP network.

3.2 Acronyms

ACK	Acknowledge
AWG	American Wire Gauge
bps	Bits per second
CRC	Cyclic Redundancy Check
CT-LWP	ClimateTalk Lightweight Protocol
LSB	Least significant bit
ms	Milliseconds
MSB	Most significant bit
MTU	Maximum transmission unit
ACK	Acknowledge
NAK	Negative acknowledge
Q(N)	Transistor
R	Resistor
Rx	Receive
Tx	Transmit
Vcc	Supply voltage

3.3 Word Usage

The conventions used in this document are modelled after the definitions of the *2009 IEEE Standards Style Manual*. The *IEEE Standards Style Manual* can be downloaded from https://development.standards.ieee.org/myproject/Public/mytools/draft/styleman.pdf.

can Equivalent to *is able to* or *is capable of*.

may Equivalent to *is permitted to* or *is allowed to*. The use of *may* means that something is optional and does not imply a requirement.

must Used to describe situations where no other course of action is possible.

shall Equivalent to *is required to*. Use of the word *shall* means that the specification shall be implemented exactly as described in order to ensure correct operation and interoperability with other devices.

should Equivalent to *is recommended that*. This is used in situations where there are several possible options, but one option is preferable to the others.

4.0 Physical Layer

4.1 Interface

CT-LWP physical layer has been developed to allow flexibility in serial interfaces using 4 wires. The interface requires the following voltage levels to operate as shown below.

PIN	Master Side Voltage Levels	Slave Side Voltage Levels	ON State	OFF State
1	Vcc: 5 to 15 Volts	Vcc: 5 to 15 Volts	-	-
2	Master Rx*: Pull up to Vcc	Slave Tx: Open Collector Output	<1V	Vcc
3	Master Tx*: Nominal 12 Volts (Range: 9 to 15 volts)	Slave Rx: Nominal 12 Volts (Range: 9 to 15 volts)	>9V	<1 V
4	COMMON	COMMON	-	-

*Master should be able to source 5mA at 12 V nominal

Table 1 - CT-LWP Voltage Level Definition

The following example interface circuit is provided for clarification of On and Off states as defined in Table 1 - CT-LWP Voltage Level Definition. In this particular implementation, Q1 and Q2 provide optical isolation between the master and the slave devices. For applications not requiring isolation, Q1 and Q2 can be changed to regular transistors. Filtering and other conditioning circuitry is not shown.

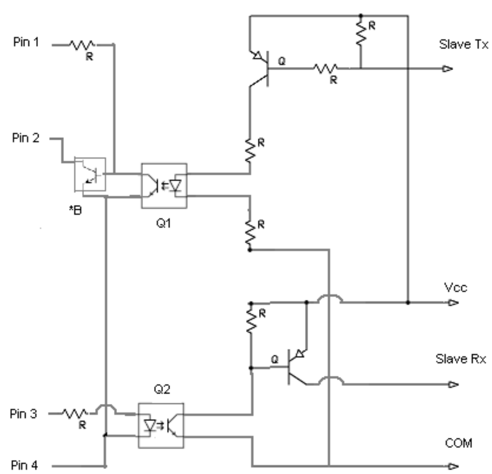


Figure 2 - Example of Interface Circuit for a Slave Device

In Figure 2 - Example of Interface Circuit for a Slave Device, *B is optional level shifting and, or, inversion logic circuitry which may be required for optically isolated devices.

4.2 Data Transmission

All numeric fields of the packet are stored most significant bit (MSB) first. In each byte, the least significant bit (LSB) is the first bit transmitted on to the wire, similar to RS-232 data transfers.

	MSB							LSB	
Micro Code	7	6	5	4	3	2	1	0	BUS →

Figure 3 - Data Transmission Bit Order

4.2.1 Network Transmission Wire

The minimum wire size for the CT-LWP physical layer is 22AWG.

4.2.2 Transmission Format

Each byte transmitted shall contain one (1) start bit, eight (8) data bits, no parity, and one (1) stop bit. Data rate for transmission is 4800 bps. The Start Bit is the ON state as defined in Section 4.1.

4.2.3 Packet Size

Information is sent as blocks of data referred to generically as a network data frame. A network data frame (i.e. – packet) includes checksum, destination address, and payload data. The largest packet that can be sent is defined as the Maximum Transmission Unit (MTU). The MTU for CT-LWP is 16 bytes.

4.2.4 Baud Rate and Byte Timing Calculations

Networking over any serial communications bus requires that the time to transmit data be taken into account when developing a timing specification for that network. To make intelligent decisions about timing specifications in the system it is necessary to determine the time it will take to transmit a packet at its maximum length. The following section will outline a simple way to do this calculation.

Baud Rate is defined as the number of bits per second that can be transmitted onto a bus. The Bit Time (the time to transmit on bit of data) can therefore be defined mathematically as the following formula.

Bit Time = 1 / Baud Rate (in Seconds)

Knowing the time it takes to transmit 1 bit of data, it is simple to determine the time it takes to transmit one byte of information. The Byte Time (the time to transmit one byte) is the cumulative time it takes to transmit all of the bits that make up that byte of data. The number of bits required to transmit one byte of data can be found by adding the number of Data Bits, Start Bits and Stop Bits. Below is the formula to calculate Byte Frame Time.

Byte Frame Time = (1 / Baud Rate) * Number of Bits per byte frame

Example:

For our example calculation, we will use the default settings for a CT-LWP physical layer as outlined in Section 4.2.2.

Byte Frame Time = (1 / 4800) * (8 + 1 + 1)

Byte Frame Time = (1 / 4800) * 10

Byte Frame Time = 2.08 millisecond

Since the MTU is defined as 16 bytes, the nominal time for a maximum size packet on the network shall be:

MTU Frame Time = MTU*Byte Frame Time

MTU Frame Time = 16*2.08

MTU Frame Time = 33.28 millisecond

5.0 Data Link Layer

The data link layer handles the physical and logical connections to the packet's destination, using a network interface. The main responsibilities of the data link layer is to provide an interface to the physical layer that can translate the application data into a physical layer transmission and translate physical layer receptions into application data.

5.1 Message Timing

Minimum spacing between two consecutive request messages originating from the master shall be 150 ms. For any request, the addressed slave is expected to start its response within 50 ms from the end of the Request Message on the bus.

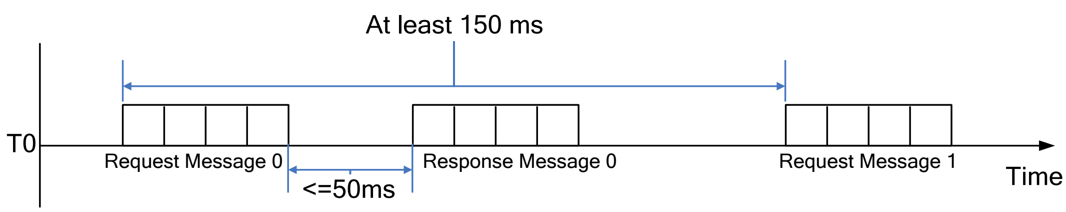


Figure 4 - Acceptable Delimiting Timing and Maximum Packet Duration

5.2 Checksum

The checksum will be calculated using the Fletcher's Checksum method.

5.2.1 Seed

All devices on a network shall share the same initial checksum seed of 0x00 for the least and most significant Bytes (bytes 1 & 2).

5.2.2 Validation

Checking a message frame for a valid Fletcher Checksum is calculated as follows:

1. Initialize check byte 1 to 0x00.
2. Initialize check byte 2 to 0x00.
3. Starting with the first byte in the message frame:
    a. Update check byte 1 to hold the remainder of: (new byte from message + check byte 1) / 255.
    b. Update check byte 2 to hold the remainder of: (check byte + check byte 1) / 255.
4. Continue the logic in step 2 until every byte in the Message Frame, including the checksum, is processed.
5. If at the end of the stream check byte 1 and check byte 2 are equal to zero then the message checksum is valid, otherwise it is invalid.

Calculating the Fletcher Checksum to append to a message frame can be calculated as follows:

1. Initialize the check byte 1 to 0x00.
2. Initialize check byte 2 to a value of 0x00.
3. Starting with the first byte in the message frame:
    a. Update check byte 1 to hold the remainder of: (new byte from message + check byte 1) / 255.
    b. Update check byte 2 to hold the remainder of: (check byte + check byte 1) / 255.
4. Continue the logic in step 2 until every byte the Message Frame, excluding the checksum, is processed.
5. After processing the entire message frame…
    a. The first byte of the appended checksum is: 255 – (remainder of ((check 1 + check 2) / 255))
    b. The second byte of the appended checksum is: 255 + (remainder of ((check 1 + check 2) / 255))

The following section provides example source code in the C programming language that can be used to validate a message frame for a valid Fletcher Checksum.

```
//=========================================================================================================
void CT_FletcherChecksumCalcInit(void)
//=========================================================================================================//
// Entry : CT_iSum1, CT_iSum2 exist globally in the project somewhere.
//
// Exit : CT_iSum1, CT_iSum2 are the initial values before beginning to accumulate a checksum.
//
// Summary : Initializes the Fletcher Checksum check bytes. This is needed before beginning to calculate
//           any new checksum.
//
//=========================================================================================================
{
       CT_iSum1 = 0;  // New Fletcher Seed.
       CT_iSum2 = 0;
}
//=========================================================================================================
```

Figure 5 - Method to Initialize the Fletcher Checksum Calculation

```
//=================================================================================================
void CT_FletcherChecksumCalcAcc(unsigned char ReceiveData)
//=================================================================================================
//
// Entry : CT_iSum1, CT_iSum2 are the current values for the accumulated checksum.
//
// Exit : CT_iSum1, CT_iSum2 are the update values for the checksum after the entry byte is accumulated.
//
// Summary : This routine will be called for each character that is added to the string.
//
//=================================================================================================
{
        CT_iSum1 = (CT_iSum1 + ReceiveData) % 255;
        CT_iSum2 = (CT_iSum2 + CT_iSum1) % 255;
}
//=================================================================================================
```

Figure 6 - Method for On-the-Fly Checksum Calculations

```
//=================================================================================================
unsigned char CT_FletcherChecksumCheck(void)
//=================================================================================================
//
// Entry : CT_iSum1, CT_iSum2 are the check byte values after processing all the bytes in a message
//              including the appended checksum.
//
// Exit : Return value will be True for a valid Checksum or False for an invalid Checksum.
//          CT_iSum1, CT_iSum2 are reset to their initial seed values to begin next calculation.
//
// Summary : This routine is called after all the bytes from a message frame have been run through
//              the Fletcher Checksum calculations. If all goes well the check bytes should be zero,
//              or else the checksum is not valid or calculated incorrectly.
//
//=================================================================================================
{
        if ( (CT_iSum1 == 0) && (CT_iSum2 == 0) )
        {
                CT_iSum1 = 0;  // New Fletcher Seed.
                return TRUE;            // CRC is valid. Note: iSum values are already == 0.
        }
        else
        {
                CT_iSum1 = 0;  // New Fletcher Seed.
                CT_iSum2 = 0;
                return FALSE;           // CRC is invalid
        }
}
//=================================================================================================
```

Figure 7 - Method for Validating the Calculated Checksum of a Message Frame

6.0 Network Layer

In the outlined network structure, the Network Layer is responsible for processing data packets to and from the network at the protocol level. The network layer's main responsibilities are to determine if a corresponding ACK is received for each packet that is sent (within the timeout window) and to send a NAK as defined in the following subsection to any invalid packet (to the correct address) received.

6.1 Packet Structure

CT-LWP is designed for one master with the option for one or multiple slaves. As such, messages are sent via pre-defined slave addresses as defined in Section 6.2. The packet format consists of a ClimateTalk packet pre-pended with a slave address and appended with a Fletcher's Checksum. Since the MTU for a CT-LWP message is restricted to 16 bytes, the data payload should always be less than or equal to 10 bytes.

BYTE	0	1	2	3	4 to (N-2)	(N-1) to N
Description	Slave Address	Message Type	Packet Number	Payload Length	Data Payload	Checksum
Length	8 bits	8 bits	8 bits	8 bits	1 to 10 bytes	2 bytes
Value	(0-255)*	(1-255)	(0-255)*	(1-10)	(1-N)	(0-65535)

Table 2 - ClimateTalk CT-LWP Packet Byte-By-Byte Breakdown

* Note that Slave Address (Byte 0) value of is reserved for broadcast messages only. Also, in this implementation of CT-LWP, the Packet Number (Byte 2) for all slave devices must be 0.

6.2 Slave Address Assignments

CT-LWP protocol allows pre-defined addresses of 254 different slave units. Address 0 is reserved for broadcast messages. The following addresses are reserved for corresponding slave types:

Node Type	Address
Broadcast (All Devices)	0
Reserved	1-9
1st or only Blower Motor	**10**
Any Additional Blower Motors	*11-19*
1st or only Inducer Fan Motor	**20**
Any Additional Inducer Fan Motors	*21-29*
1st or only Outdoor Fan Motor	**30**
Any Additional Outdoor Fan Motors	*31-39*
Reserved	40 -59
1st or only Stepper Gas Valve Motor	**60**
Any Additional Stepper Gas Valve Motors	*61-69*
Reserved	62-255

Table 3 - CT-LWP Addresses

NOTE: 0 is reserved for broadcasting messages to all the units at the same time. However, only certain commands that do not require replies can be broadcasted.

6.3 ACK and NAK Usage

CT-LWP uses multiple ACKs and NAKs from the slave to meet the application requirements defined in the ClimateTalk Motor Profile. The following table summarizes the proper response from the slave device under each circumstance.

ACK/ NAK	Applicable Layer	Hex Value	Description
ACK1	Application	0x06	Message Received with Correct CRC and valid command - Only sent for control commands from master
ACK2	Application	0x0A	Message Received with Correct CRC - Undesired Parameter Sent
ACK3	Application	0x0D	Message Received with Correct CRC - Minimum Parameters Not Complete
NAK1	Network	0x15	Message Received with Incorrect CRC
NAK2	Application	0x1B	Message Received - Invalid message for that specific application implementation

Table 4 - ACK and NAK Messages and Data Payload Values

Sending and processing NAK1 packets will be done at the network layer and will be transparent to higher layers. All other ACK and NAK packets are processed at the application level and apply to the slave devices defined in the ClimateTalk Motor Profile.

If a packet is sent and a NAK1 is received, then the packet is assumed to have not been sent successfully on the network. If a NAK1 is received five consecutive times for the same message type, then an error message is generated by the master device. The error message is cleared once communication is re-established.