ClimateTalk 2.0
CT-485 Data Link 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 data link requirements for CT-485. Corresponding to OSI Layer 2, the Data Link specification defines CRCs, bit level error correction, physical node addressing, subnet addressing, and data transmission rules.

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This specification may be updated at any time and may be superseded by a more recent version or amended to from time to time.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 0.9		2008-11-07	Pre-Release
V 1.0		2009-08-24	Initial Release
V 1.1		2011-06-23	Errata Package
V 1.3		2011-11-02	Additional Errata Updates, 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 Water Products Company

ecobee inc.

EDC

Emerson Electric, Co.

EWC Controls, Inc.

Microchip Technologies, Inc.

Nogginhaus, LLC.

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 an open standard that defines 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 (CIM) 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, the Generic Application Specification and the Command Reference.

ClimateTalk messages can be carried over any physical medium following the OSI model. The ClimateTalk Presentation Layer defines how messages are executed over the various physical mediums in use.

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.

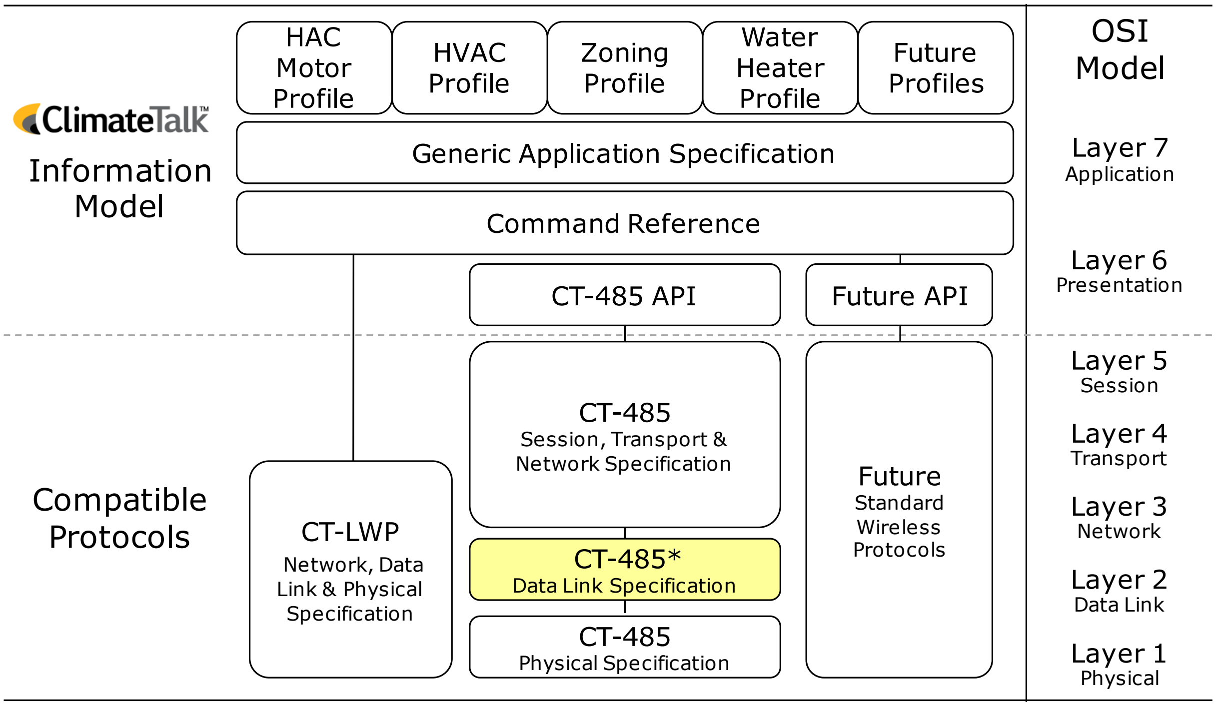
1.2 Scope

CT-485 is a Physical, Data Link, and Networking set of specifications that define one of the physical media over which ClimateTalk messages are sent. CT-485 is a variant of EIA/TIA-485[2] standards with provisions against incorrect wiring and grounding requirements that meet the needs of residential systems.

This document defines the Data Link Layer requirements for CT-485. Corresponding to OSI Layer 2, the Data Link specification defines CRCs, bit level error correction, physical node addressing, subnet addressing, and data transmission rules. See Figure 1 - OSI Layers for a CT-485 Implementation for a diagram of relevant standards.

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=20269
[2] http://standardsdocuments.tiaonline.org/tia-tsb-89-a.htm

Figure 1 - OSI Layers for a CT-485 Implementation



**This Document*

The ClimateTalk Open Standards package shown in Figure 1 - OSI Layers for a CT-485 Implementation prescribes the 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 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 Generic Application Specification

ClimateTalk Command Reference

ClimateTalk CT-485 Application Protocol Interface

ClimateTalk CT-485 Networking Specification

ClimateTalk CT-485 Physical Specification

3.0 Terminology

3.1 Definitions

Network Coordinator	Device responsible for addressing the entire network and determining to which subnet new devices are assigned
Off-Board Bus Interface (OBBI)	Interface converter between devices on a communicating ClimateTalk HVAC network and devices that use 24 VAC relays for control
Subnet Coordinator	Device responsible for an individual subnet and for performing periodic status queries of each device on the subnet. The Subnet Coordinator can also be the Network Coordinator
Subordinate	Device in a network managed by a Network Coordinator

3.2 Acronyms

CT	ClimateTalk
IEEE	Institute of Electrical and Electronics Engineers
LSB	Least Significant Bit
MAC	Media Access Control
MSB	Most Significant Bit
OBBI	Off-Board Bus Interface
R2R	Request to Receive
UART	Universal Asynchronous Receiver Transmitter
VAC	Volts Alternating Current
VDC	Volts Direct Current

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 Communication Models

Applications designed to ClimateTalk, CT-485 standards can operate in two types of communication models. The communications bus can either communicate to devices in a synchronous (Coordinator/Subordinate) relationship or in a low traffic asynchronous mode. Profiles define which model is used for the associated network.

4.1 Synchronous

In this type of communication, the Coordinator Node initiates communications with another node known as the Subordinate node. The Coordinator starts all communication sequences. The Subordinate node never initiates communications with the Coordinator.

The Coordinator Node is any supervisory node controlling or monitoring the Subordinate nodes. The Coordinator Node shall control communication with the Subordinates using a defined transport mechanism as defined by the *CT-485 Network Specification* [B2] using this Data Link protocol.

If a device is configured as a Subordinate, the Coordinator controls the communications of this device. The Subordinate may or may not care what other devices are on the network unless the Subordinate wants to communicate with other devices. If that is the case, all third-party communications are through the Coordinator.

4.2 Broadcast Communications

The transmission to all devices is supported, but no response is mandated. Each profile determines the broadcast capabilities in addition to the rules for determining if one or more devices respond.

5.0 Data Transmission

5.1 Bit and Byte Order

5.1.1 Bit Order

All numeric fields of the packet are stored most significant bit (MSB) first. In each byte, the least significant bit (LSB) shall be the first bit transmitted on the wire. Not every microcontroller has a UART that follows this hardware rule, so this needs to be verified per the manufacturer.

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

5.1.2 Byte Order

For transmitting multiple bytes of data such as a 16-bit word, the Low Byte shall be transmitted first followed by the higher bytes as seen in the following example. Notice the LSB of the Low Byte is sent first.

	MSB LSB	MSB LSB	
Micro Code	HIGH BYTE	LOW BYTE	UART →

5.1.3 CRC Byte Order

The Checksum value is the last two bytes from the sum of byte zero added with and including Bytes N-2. The calculation of this value is discussed more in Section 7.0. In this case, the normal multiple-byte data transfer is different as seen below. The CRC Low Byte shall be transmitted first and then the CRC High Byte.

	BYTE N MSB LSB	BYTE N -1 MSB LSB	
Micro Code	CRC HIGH BYTE	CRC LOW BYTE	UART →

5.2 Frame Format

The CT-485 Message Frame can be divided into four basic segments:

1. Message Header
2. Packet Header
3. Packet Payload
4. Message Footer

The Message Frame also can be more granular by dividing it into 11 individual elements.

Table 1 – Message Structure

Element	Segment	Size (in bytes)	Starting Offset in Hex
Destination Address	Message Header	1	0x00
Source Address		1	0x01
Subnet		1	0x02
Send Method		1	0x03
Send Parameters		2	0x04
Source Node Type		1	0x06
Message Type	Packet Header	1	0x07
Packet Number		1	0x08
Packet Length		1	0x09
Packet Payload	Packet Payload	0-240	0x0A
Message Checksum	Message Footer	2	0x0A + Packet Length (directly after packet payload)

Within the CT-485 frame, the Data Link Layer is responsible for generating the Checksum field in the Message Footer, highlighted in blue in Table 1 above. The other elements are covered in more detail in the *CT-485 Network Specification* [B2] *and CT-485 Application Protocol Interface Specification* [B3].

5.2.1 Message Checksum

The checksum is a two-byte Fletcher Checksum calculated from the contents of the entire CT-485 Message Frame. This provides a significant level of data integrity. The Message Footer shall reside at the expected end of the Message Frame based on the payload length to align the checksum at the expected location for correct processing. This minimizes processing corrupted data streams.

See section 7.0 - Checksum for information on the calculation of the checksum.

6.0 Addressing

6.1 CT-485 MAC Address

Each CT-485 device must store an eight-byte hardware MAC Address that provides a unique identifier for each device.

A MAC Address of all zeros is not allowed. At least one bit must be set in the eight-byte field.

The Coordinator is responsible for tracking the MAC Addresses of each Subordinate and actively tracking any changes to a Subordinate's MAC Address. A change in MAC Address indicates that a device has been replaced.

Figure 2 – CT-485 MAC Address Structure

8 bytes

Most Significant Bit

1 Byte	1 Byte	1 Byte	5 Bytes
0x00	Manufacturer ID (D15:D8)	Manufacturer ID (D7:D0)	Unique Unit ID Per Manufacturer

Least Significant Bit

Table 2 – Fields within CT-485 MAC Address

Field	Length	Description
		Most significant
Reserved	8 bits	Reserved = 0x00
Manufacturer ID	16 bits	Unique Identifier assigned by the ClimateTalk Alliance to each Manufacturer
Unique ID	40 bits	Unique Unit ID Per Manufacturer
		Least significant

6.2 Node Address Assignments

The first two bytes transmitted from any device correspond to the node address assigned to the device by the Coordinator upon initialization. The first byte is the destination address and the second byte is the sender's address.

Certain addresses are predefined. Address 0x00 is a broadcast address that is recognized by all devices. The Network Coordinator always has an address of 0xFF.

6.3 Subnets

The protocol is capable of supporting up to 255 subnets. Each subnet is considered an isolated system for which a corresponding Subnet Coordinator is responsible. As in addressing, there are predefined subnets that correspond to specific system functionality.

Table 3 – Subnet Addresses

Function	Subnet	Description
Broadcast	0	Utilized to communicate to all devices
Maintenance (Single HVAC)	1	
Subnet Coordinators	2 – 254	Used to handle multiple systems and associated Subnet Coordinators.

For example, it is possible to broadcast to all devices on all subnets on node address zero and subnet zero, or to all devices on a specific subnet by using node address zero and the specific subnet address.

The main Network Coordinator is responsible for transmitting R2Rs to their associated subsystems and each Subnet Coordinator to ensure communications and control is maintained.

6.4 Node Type Field

The usage of Node Types is one way of receiving data from other devices on the network without having to initiate communications on the network. Subsystems can be in a listen mode to monitor the bus for certain Node Types to get information from that subsystem.

Node Types are listed in Annex C of the *ClimateTalk Command Reference*.

7.0 Checksum

The checksum is calculated using the Fletcher's Checksum method.

7.1 Seed

All devices on a network shall share the same initial checksum seed of 0xAA for the Least Significant Byte (Byte 1) and 0x00 for the Most Significant Byte (Byte 2). This helps fix an issue with error transmission integrity in the Fletcher's algorithm.

7.2 Example Checksum Algorithms

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

The procedure in Figure 3 is used to initialize the checksum and accumulate variables before starting the checksum calculations when calculating the checksum for a particular frame transmission.

Figure 3 – Method to Initialize the Fletcher Checksum Calculation

```
//==================================================================
=======================//
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 = 0xAA;		// New Fletcher Seed.
	CT_iSum2 = 0;
}
//==================================================================
=======================//
```

7.2.1 Validating integrity of a received packet

After initializing the accumulation of variables using the procedure in Figure 3, the procedure in Figure 4 is performed on every byte of a received packet. Once it has been executed over all bytes of a received packet, the procedure in Figure 5 is used to validate the packet.

Figure 4 – Method for On-the-Fly Checksum Calculations for validation

```
//==========================================================================
=======================//
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 5 – Method for Validating the Calculated Checksum of a Message Frame

```
//==========================================================================
=======================//
unsigned char CT_FletcherChecksumCheck(void)
//==========================================================================
=======================//
//
// Entry : CT_iSum1, CT_iSum2 are the check byte values after processing all the bytes in a
// received 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 before beginning 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 = 0xAA;        // New Fletcher Seed.
                return TRUE;            // CRC is valid. Note: iSum values are already == 0.
        }
        else
        {
                CT_iSum1 = 0xAA;        // New Fletcher Seed.
                CT_iSum2 = 0;
                return FALSE;           // CRC is invalid
        }
}
//============================================================//
```

7.2.2 Calculating the checksum to append for packet transmission

To calculate the checksum for a packet to be transmitted, after initializing the accumulation of variables using the procedure in Figure 3, the procedure in Figure 4 is to be used on every byte of the packet to be transmitted except the checksum bytes, which currently are unknown. Once this process is complete, a final step described by Figure 6 is performed on the result to provide the two checksum bytes that then are appended to the packet.

Figure 6 – Final step in calculating checksum to append to a packet to transmit

```
//===========================================================================
========================//
void CT_FletcherChecksumFinalStep(CT_iSum1, CT_iSum2)
//===========================================================================
========================//
//
// Entry: CT_iSum1, CT_iSum2 are the result of the checksum accumulation calculation on all bytes
// of the packet excluding the checksum.
//
// Exit: CT_grc1, CT_grc2 are the checksum bytes to be placed as the last two bytes of the packet
// in that order.
//
// Summary: This routine is called after the accumulation routine has run over all bytes.
//
//===========================================================================
========================//
{
        CT_grc1 = 255 - (((CT_iSum1 + CT_iSum2))% 255);
        CT_grc2 = 255 - (((CT_iSum1 + CT_grc1))% 255);
}
//===========================================================================
========================//
```

8.0 Annex A – Bibliography

"TIA-485 (Revision A), Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems" *Telecommunications Industry Association*, 1998.

Zimmermann, Hubert (April 1980). "OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection". *IEEE Transactions on Communications*