DEVELOPMENT STANDARD

QIC-122
Revision B
06 Feb 91

DATA COMPRESSION FORMAT FOR
1/4-INCH DATA CARTRIDGE TAPE DRIVES

USE OF THIS DEVELOPMENT STANDARD IS OPTIONAL. IT MAY BE IMPLEMENTED ON DRIVES SUPPORTING ONE OR MORE QIC DEVELOPMENT STANDARDS FOR RECORDING FORMAT, AS LISTED IN QIC-123. QIC MAY ADOPT OTHER OPTIONAL DEVELOPMENT STANDARDS ASSOCIATED WITH THE ABOVE-MENTIONED QIC DEVELOPMENT STANDARDS FOR RECORDING FORMAT.

Quarter-Inch Cartridge Drive Standards, Inc.

311 East Carrillo Street
Santa Barbara, California 93101
Telephone (805) 963-3853
Fax (805) 962-1541
www.qic.org

(See important notices on the following page)

Important Notices

This document is a development standard adopted by Quarter-Inch Cartridge Drive Standards, Inc. (QIC). This document may be revised several times during the development cycle. It is intended solely as a guide for companies interested in developing products which can be compatible with other products developed using this document. QIC makes no representation or warranty regarding this document, and any company using this document shall do so at its sole risk, including specifically the risks that a product developed will not be compatible with any other product or that any particular performance will not be achieved. QIC shall not be liable for any exemplary, incidental, proximate or consequential damages or expenses arising from the use of this document. This development standard defines only one approach to the product. Other approaches may be available in the industry.

This development standard is an authorized and approved publication of QIC. The underlying information and materials contained herein are the exclusive property of QIC but may be referred to and utilized by the general public for any legitimate purpose, particularly in the design and development of quarter-inch tape cartridge drive subsystems. This development standard may be copied in whole or in part *provided* that no revisions, alterations or changes of any kind are made to the materials contained herein. Only QIC has the right and authority to revise or change the material contained in this development standard, and any revisions by any party other than QIC are totally unauthorized and specifically prohibited.

Compliance with this development standard may require use of one or more features covered by proprietary rights (such as features which are the subject of a patent, patent application, copyright, mask work right or trade secret right). By publication of this development standard, no position is taken by QIC with respect to the validity or infringement of any patent or other proprietary right, whether owned by a Member or Associate of QIC, or otherwise. QIC hereby expressly disclaims any liability for infringement of intellectual property rights of others by virtue of the use of this development standard. QIC has not and does not investigate any notices or allegations of infringement prompted by publication of any QIC development standard, nor does QIC undertake a duty to advise users or potential users of QIC development standards of such notices or allegations. QIC hereby expressly advises all users or potential users of this development standard to investigate and analyze any potential infringement situation, seek the advice of intellectual property counsel, and, if indicated, obtain a license under any applicable intellectual property right or take the necessary steps to avoid infringement of any intellectual property right. QIC expressly disclaims any intent to promote infringement of any intellectual property right by virtue of the evolution, adoption, or publication of any QIC development standard.

PROPRIETARY RIGHTS.STATEMENT

The reader's attention is called to the possibility that compliance with this standard may require use of one or more inventions covered by patents either issued and/or pending.

Table of Contents

1.1 Scope

This document describes only an encoding method for compressed binary data. An apparatus for performing data compression which adheres to the format described herein is not presented. The method described herein is intended to be referenced by other format standards which incorporate optional data compression.

1.2 Purpose

The purpose of this document is to specify how data compression shall be implemented when optionally specified within QIC format documents. Strict adherence to the encoding format specified herein is necessary to insure data interchange between QIC tape systems that support data compression.

1.3 Conformance

A compressed data stream conforms to this standard if it strictly adheres to the encoding method described in Section 5.0 below.

2.0 References

None

3.0 Definitions

History Buffer. The compression format assumes that the compression and decompression mechanisms retain the last 2048 bytes of the uncompressed data stream in a memory. This 2048 bytes of the most recently processed uncompressed data stream is referred to as the history buffer herein.

Raw Data. Refers to data in the compressed data stream that was not found in any compressed data string by the compression apparatus. Raw data is differentiated from compressed string data in the output data stream by the Compression Flag field.

4.0 Data Reliability

The data compression encoding format described herein is simply another way of representing the information present in any byte-oriented data stream. The use of data compression does not alter data reliability. A minimum level of data reliability in the underlying tape data storage system is presumed by this document. The application of data compression information encoding has no effect on the data reliability of the underlying data storage system.

5.0 Data Format

5.1 Overview

This document describes a data compression encoding format and does not describe an algorithm for compressing or decompressing data streams. The data compression encoding format is designed t..) support an adaptive string compression algorithm that can find redundant strings in an uncompressed data stream and replace them with shorter tokens in the compressed output data stream. The tokens contain enough information about the string to allow for its reconstruction during decompression.

5.2 Limitations

The data compression encoding format provides a mechanism for compressing byteoriented data only. Data strea-ns which do not contain data organized on byte boundaries will generally result in poorer compression ratios.

5.3 Algorithms

The format requires that a history buffer of 2048 bytes be maintained during compression and decompression. A compression algorithm must find strings in the incoming uncompressed data stream which are present in the history buffer. Once located, the compression format defines how a token of shorter length may be inserted into the data stream in place of the redundant string. If no string match is found the format supports a token type which allows the algorithm to pass through the raw data unaffected. The token for a compressed string contains an offset into the history buffer and a length field.

The decompression algorithm merely keeps a history buffer and processes the tokens in the incoming compressed data stream as they are encountered. The processing involves the simple step of looking up the appropriate string in the history buffer according to its offset and length.

5.4 Compressed Data Encoding Format

In order to insure that the two complementary operations of compression and decompression are in fact inverse operations, an unambiguous format for compressed data streams must be defined and adhered to by systems that purport to be compatible according to this standard.

An extremely precise method borrowed from formal language theory is employed below to fully describe the universe of legal compressed data stream encodings. The implementor is cautioned however, that definition of what is legal does not in any way speak to the issue of performance. There are many ways in which a legal encoding of a non-compressed data stream may be derived while maintaining strict conformance with the encoding format given. The format only allows for, but does not enforce, the achievement of high levels of compression. It is the job of the implementor to insure that acceptable levels of compression and data throughput are achieved while maintaining strict adherence to the encoding format given.

The compressed data stream encoding format is described below using a BNF-like metalanguage to provide an unambiguous description. The following metasymbols are used:

Symbol	Description
:=	The non-terminal on the left side of the ":=" can be replaced by the expression on the right side.
<.name>	A non-terminal expression
[]	The expression inside the []
|	Is read as the disjunction, "or" may occur zero or more times.
( )	The text inside the "( )" is a comment provided for clarity.
0,1	The terminal binary digits 1 or 0.

Encoding Format Description

<Compressed_Stream> := [<Compressed_String>] <End_Marker>

<Compressed_String> := 0 <Raw_Byte> | 1 <Compressed_Bytes>

<Raw-Byte> := <b><b><b><b><b><b><b><b> (8-bit byte)

<Compressed_Bytes> := <Offset> <Length>

<Offset> := 1 <b><b><b><b><b><b><b> (7-bit offset) |
0 <b><b><b><b><b><b><b><b><b><b><b> (11-bit)

<End_Marker> := 110000000 (offset = 0)

<b> := 0 | 1

<Length> :=	00					(= 2 bytes) |
	01					(= 3 bytes) |
	10					(= 4 bytes) |
	11	00				(= 5 bytes) |
	11	01				(= 6 bytes) |
	11	10				(= 7 bytes) |
	11	11	0000			(= 8 bytes) |
	11	11	0001			(= 9 bytes) |
	11	11	0010			(= 10 bytes) |
	11	11	0011			(= 11 bytes) |
	11	11	0100			(= 12 bytes) |
	11	11	0101			(= 13 bytes) |
	11	11	0110			(= 14 bytes) |
	11	11	0111			(= 15 bytes) |
	11	11	1000			(= 16 bytes) |
	11	11	1001			(= 17 bytes) |
	11	11	1010			(= 18 bytes) |
	11	11	1011			(= 19 bytes) |
	11	11	1100			(= 20 bytes) |
	11	11	1101			(= 21 bytes) |
	11	11	1110			(= 22 bytes) |
	11	11	1111	0000		(= 23 bytes) |
	11	11	1111	0001		(= 24 bytes) |
	11	11	1111	0010		(= 25 bytes) |
	…					
	11	11	1111	1110		(= 37 bytes) |
	11	11.	1111	1111	0000	(= 38 bytes) |
	11	11	1111	1111	0001	(= 39 bytes) |
	etc.					

Note that the encoded output bits are stored from most significant bit to the least significant bit in the output byte stream. To decompress a string, the following algorithm is used:

```
for ( i = 0; i < length; i++, hptr++ )
     history [ hptr ] = history [ (hptr-offset) & 2047];
```

where hptr is the current history buffer pointer.

Appendix A - Example Compression Encoding

```
Input byte stream          Output bit stream        Comment

   +--->-  A               0 01000001               Raw byte ASCII A
   |       B               0 01000010               Raw byte ASCII B
   | +-->  A               0 01000001               Raw byte ASCII A
   | +-<-  A               1 1 0000001 1100         String of length 5
   |       A                                        at offset 1
   |       A
   |       A
   |       A
   |       C               0 01000011               Raw byte ASCII C
   +--<--  A               1 1 0001001 01           String of length 3
   +-->--  B                                        at offset 9
   |       A
   +--<--  B               1 1 0000010 10           String of length 4
           A                                        at offset 2
           B
           A
                           1 1 0000000              End o f data
```

```
   Output byte stream (hex):
      20 90 88 38 1C 21 E2 5C 15 80
```