usitt

*The Association of Design, Production, and Technology Professionals in the Performing Arts and Entertainment Industry*

# ASCII Text Representation for Lighting Console Data

**Version 3.0**
**Ident 3:0**
**March 1992**

Duplication Date
March 2004

**United States Institute for Theatre Technology, Inc.**

6443 Ridings Road · Syracuse, NY 13206-1111 USA
800-93USITT · 315-463-6463 · Fax: 315-463-6525
info@office.usitt.org · www.usitt.org

# ASCII Text Representation for Lighting Console Data

Version 3.0

Ident 3:0

March 1992

## Table of Contents

# 1 INTRODUCTION

## 1.1 Scope

This specification describes a manufacturer independent method for communicating the theatrical lighting system control data normally stored in lighting consoles. Manufacturer independence is obtained by encoding the data using ASCII character sequences, among other things. The principal objectives of this specification are discussed in Section 2.

This specification does **not** define the methods and mechanisms by which the data is communicated from one system to another. This specification intends that the data be communicated by any means deemed appropriate by the sending and receiving parties. It is assumed that floppy disks and network links will be among the communications methods used. The recording format used on the floppy disks and the communications protocols used by the network links are ignored by this specification.

## 1.2 System Definitions

The following paragraphs define the terms used to describe participants in lighting cue data transmissions. These terms are used throughout this specification. Several specialized terms are used in describing the contents and structure of a data stream. These are defined in Section 5.

Data Stream: An ordered collection of ASCII characters using the sequences defined in this specification is termed a data stream. When greater clarity is required, the phrase, "lighting console data stream," is used. Defining the data stream structure is the primary purpose of this specification.

Receiving System: The receiving system is the lighting console or a specialized lighting cue data manipulation program that is reading and interpreting a data stream. Because a data stream is encoded using ASCII characters, other programs may read data streams. However, such programs are not receiving systems.

Data Stream Source: The data stream source is the entity that is providing (or has provided) the data stream being processed by the receiving system. The data stream source may or may not treat its data as lighting console data. (See source system definition for examples.)

Source System: A source system is a data stream source that treats its data as lighting console data. For example, a lighting console is a source system but a text editor is not. A text editor is just a data stream source.

System Operator: The system operator is the person running the lighting console that is acting as a receiving system. If the receiving system is a lighting cues manipulation program, the system operator is the person who is running that program.

# 2 OBJECTIVES

## 2.1 Editing show data

The lighting console data stream shall be constructed in a way that allows manipulation of the data using a text editor, common word processor or spreadsheet program. To this end, the data stream shall be readable and mnemonic. It shall favor words and numbers instead of special characters.

## 2.2 Transferring show data between consoles

The data stream shall allow show data to be transferred from one lighting console to another, different console. Naturally, this capability must be limited to those areas in which the two consoles have similar features. The data stream definition shall provide a data transfer mechanism for the largest possible set of common lighting console features.

Transfers of recorded channel *looks* (*cues* or *memories*) shall be the minimal capability provided. When dimmer to channel electronic patching is available in both consoles, transfer of that information shall be provided. Split cross fades, linking of cues, and division of cues into multiple parts are some of the more sophisticated capabilities specified here.

Console manufacturers may extend the data stream contents in ways that are specifically useful to their systems. The extended data stream may transfer data between different console models from that manufacturer's product line. Data stream extensions shall be easily recognizable. Thus, receiving systems that do not accept such extensions can reliably ignore them.

## 2.3 Transfer medium not specified

This proposal **does not** address the storage medium or transfer mechanism for the lighting console data stream. Only the interpretation of the data stream is defined.

The defined data stream contents **do** assume that the data can be generated, and interpreted, as a sequential text stream. Random access into a disk file is not acceptable.

Lighting console data streams may be transferred on disk, via serial port, or even via parallel port. Disks may or may not be IBM compatible.

## 2.4 Non lighting console systems

Although lighting consoles are the primary generators and consumers of data streams, other specialized programs also can read and write such data. This specification anticipates two types of such programs:

1) Preprocessing programs -- These programs relieve a console from having to process a data stream internally. They read a data stream, translate it into the internal format used by the console, and transmit the reformatted data to the console. This allows a simpler internal console design.

2) Lighting design programs -- These programs provide lighting designers with sophisticated capabilities beyond those found in lighting consoles. The results produced by the design program can be transmitted to a console as a lighting console data stream.

## 2.5 One show at a time

A single data stream shall describe no more than one show. When a receiving system can hold multiple shows concurrently, each show shall be loaded via a separate data stream. This simplifies both the data stream definition and the receiving system implementation.

## 2.6 Integer numbers

This specification uses integer data values wherever possible. The use of floating point data values by a receiving system is always optional. This allows construction of receiving systems from simpler components.

# 3 SPECIFICATION FORMAT

Seven sections of this document describe the data stream processing requirements that form this protocol. These sections and the information they cover are as follows:

| Section | Contents |
|---|---|
| 4 | how to report alternate interpretation usages or exception behavior occurrences |
| 5 | the basic components of a data stream |
| 6 | how a data stream is organized and processed |
| 7 | the formats of data values used in more than one part of the data stream |
| 8 | definitions for the basic keywords |
| 9 | definitions for the primary keywords |
| 10 | definitions for the secondary keywords |

Note: Section 6.2 describes the differences between basic, primary and secondary keywords.

Throughout this specification alternate interpretations of required actions are noted in paragraphs beginning with: **"Alternate interpretation (*description*)."** These paragraphs are called "alternate interpretation paragraphs." These paragraphs describe actions a receiving system may take when some aspect of its construction prohibits the preferred interpretation previously described. For example:

> **"Alternate interpretation (levels above 100 not supported):** If a receiving system does not support level values above 100, it shall convert all level values between 101 and 999 that it encounters to 100."

Throughout this specification, exception conditions are noted in paragraphs beginning with: **"Exception behavior (*condition information*)."** These paragraphs are called "exception behavior paragraphs." These paragraphs describe incorrect conditions in the data stream. These paragraphs also describe the actions a receiving

system shall take when such conditions are encountered in the data stream. For example:

> **"Exception behavior (text too long):** Characters in excess of the receiving system's limit shall be ignored."

Unless otherwise noted, the actions described in exception behavior paragraphs are required for all receiving systems.

Throughout this specification, fine print notes provide examples of important situations, or discussions of the details and motivations behind a specified behavior. These fine print notes are headed by, "FPN:" For example:

> "FPN: The secret is in the frosting."

Data field processing descriptions (in Section 7) and keyword descriptions (in Sections 8 through 10) are organized in the same order:

1) Keyword (or field) prototype
2) Examples
3) Other reference information
4) Normal processing actions
5) Keyword data field definitions
6) Alternate interpretation paragraphs
7) Exception behavior paragraphs

The preferred interpretation for each keyword (or field) is always described first. The preferred interpretation shall be used unless some aspect of the receiving system's construction prohibits it. In those cases where a definition does not contain any alternate interpretation paragraphs, the preferred interpretation is the required (only permissible) interpretation.

Apparently, these rules prohibit optional keywords (i.e., keywords that are not effective on all receiving systems). However, some keywords are optional. Their optional nature is defined using alternate interpretation paragraphs.

# CONDITION REPORTING

When many exception conditions occur, the system operator shall (or should) be notified. Similarly, when some alternate interpretations are used, operator notification may be recommended. Note: Operator notification is never required when an alternate interpretation is used.

This section describes:

1. How to identify alternate interpretation and exception behavior paragraphs for which system operator reporting is defined,

2. How to identify when reporting of an exception behavior is required, and

3. How to perform such reporting.

FPN: If the terms alternate interpretation paragraph or exception behavior paragraph need clarification, please review Section 3 before proceeding with this section.

## 4.1 Identifying condition reporting situations

The situations that may be reported to the system operator are either alternate interpretation usages or exception behavior occurrences. These situations are identified by alternate interpretation or exception behavior paragraphs. When system operator reporting is defined these paragraph headers are enhanced slightly to read:

**Alternate interpretation (*description, STnnnn*):**
**Exception behavior (*condition information, STnnnn*):**

*S*, *T* and *nnnn* represent the condition severity, reporting type and condition number, respectively. They are described in the paragraphs that follow.

The letter *S* represents the condition severity. It will be replaced by one of I, W, or E, whose meanings are as follows:

| S | Meaning |
|---|---|
| I | Informational - a condition for which totally automated recovery is possible without adverse results. Still, operator notification of the event would be helpful. |

| S | Meaning |
|---|---|
| W | Warning - a condition for which automated recovery is possible. However, said recovery may result in undesirable actions. |
| E | Error - a condition for which no recovery exists. Typically, error exceptions result in termination of data stream processing. |

The letter *T* represents the condition reporting type. It will be replaced by one of R or O, whose meanings are as follows:

| T | Meaning |
|---|---|
| R | Required - this condition shall be reported whenever it is encountered. |
| O | Optional - reporting of this condition is optional, but recommended whenever possible. |

FPN: Reporting of alternate interpretation usages is always optional.

The *nnnn* is the condition number. Each condition identified in this specification is assigned a unique condition number. The number may be used as a shorthand way of reporting an occurrence of the condition to the system operator. See Section 4.2 for details about how conditions are reported to the system operator.

The *nnnn* condition number also can be used to locate the condition in Appendix B. The reported conditions table in Appendix B gives the condition, severity, reporting type, and suggested reporting text for every condition identified in this specification. The entries in Appendix B are sorted by ascending numerical condition number value.

## 4.2 Reporting conditions to the system operator

Three levels of condition reporting to the system operator are defined. All receiving systems shall provide at least Level 1 condition reporting. Levels 2 and 3 are improvements over Level 1. Level 3 is the most sophisticated condition reporting level. Receiving systems should use the highest condition reporting level that is consistent with their design goals, system hardware and cost constraints.

The three condition reporting levels are:

1. Reporting only required type conditions using condition number values.

2. Reporting both required and optional type conditions using condition number values.

3. Reporting both required and optional type conditions using text messages.

FPN: To reduce the costs associated with Level 1 condition reporting, all the required type conditions have condition number values between 0 and 99. Therefore, Level 1 condition reporting can be implemented using only a two decimal digit display.

Since the condition number values for optional type conditions are between 100 and 9999, Level 2 condition reporting requires a four decimal digit display. Level 3 condition reporting requires full text display capability.

Each condition shall be displayed at the time the condition is encountered. Depending on the condition reporting level, this may be a number or a text string.

When data stream processing terminates, Level 1 and Level 2 reporting system shall display the report for the last error severity condition encountered. If no error severity conditions have been encountered, the report for the last warning severity condition encountered shall be displayed. If no warning severity conditions have been encountered, 0000 (or 00) shall be displayed.

Level 3 reporting systems shall display each condition report on a separate line of a scrolling display composed of at least 5 lines. The line shall be formatted as follows:

1) the condition number
2) a dash, one letter representing the condition severity, a space
3) the condition text.

Suggested condition text is shown in Appendix B. Thus, the condition noted as ER0099 would be reported as:

0099-E Ident mismatch prohibits processing

It is recommended (but not required) that Level 3 reporting systems give the number of the record in which the reported condition occurs. The record number can be displayed any way the reporting system chooses. A suggested way is placing the record number enclosed in parentheses before the required reporting text. Thus, if record 52 contains an invalid hexadecimal value (condition WO0172), the condition report would read:

(00052) 0172-W Invalid hexadecimal value /h025G/

**FPN:** This example also shows how the offending field data is substituted for /field/ in the suggested message text. See Appendix B for a detailed discussion of substitution within suggested message text. See Section 5.4 for a definition of the term, "field."

# 5 DATA STREAM COMPONENTS

This section describes the basic components of a lighting console data stream. The simple (or lowest level) components are described first. They act as building blocks for the more complex components that are described toward the end of this section.

## 5.1 Data stream character set

A data stream shall consist only of:

the printable ASCII characters, 20-7E hex
tab, 09 hex
line feed, 0A hex
carriage return, 0D hex

The high bit of all characters generated by source systems shall be zero.

Not all of the characters in the data stream character set are required in construction of a valid data stream. Many special characters (such as the asterisk) are not required anywhere in a data stream.

Any character in the data stream character set but not assigned a usage in the data stream definition shall appear only in one of the following places:

1. In comments (see Section 5.5),
2. As data for a TEXT keyword (see Section 10.6), or
3. In manufacturer specific data (see Section 5.8).

**Exception behavior (character with high bit set, WO0701):** Any characters that have the high bit set shall be ignored. Although ignored, these characters shall count toward the size of the record. As such, they may cause the record to be oversized. (See exception behavior in Section 5.2.)

FPN: Ignoring characters that have the high bit set is for safety with some word processing programs. It **does not** release source systems from the obligation to make the high bit zero when creating data streams!

**Exception behavior (non-printing ASCII characters, WO0702):** The non-printing ASCII characters, 00 to 1F hex (excluding carriage return, line feed, and tab), shall be ignored. Although ignored, these characters shall count toward the size of the record. As such, they may cause the record to be oversized. (See exception behavior in Section 5.2.)

## 5.2 Records

A data stream shall consist of a series of records of text. Usually, a record is equivalent to a line of text. However, since a data stream can be transmitted over a network link, this specification will use the term record instead of line.

A record shall contain from 0 to 80 characters. The characters in a record shall be members of the character set described in Section 5.1.

A record shall be followed by a terminator. Valid record terminators are: carriage return (CR), line feed (LF), CR+LF, or LF+CR.

The record terminator **shall not** be counted as part of the limit on the number of characters sent or received in a record. So, record buffers must hold at least 82 characters.

For compatibility with text editors, it is recommended that source systems create data streams whose records are terminated by the CR+LF combination (a carriage return followed immediately by a line feed).

FPN: Although the record terminator combinations of CR+LF and LF+CR could be accommodated as delimiting a zero length record, doing so would incorrectly bias the record number values in Level 3 condition reports (see Section 4.2).

**Exception behavior (oversized record, ER0091):** Receipt of a record longer than 80 characters (excluding the record terminator) shall cause processing of the data stream to be aborted. The condition shall be reported as noted above.

## 5.3 Terminating a data stream

A data stream shall be terminated by a record containing the ENDDATA keyword (see Section 8.3).

**Exception behavior (data stream terminated without ENDDATA, EO0100):** If an end of data condition (such as an end-of-file marker) occurs before an ENDDATA record is encountered, data stream processing shall be terminated. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

## 5.4 Delimiters and fields

A data stream record shall be composed of one or more fields; except in zero length records (see Section 5.2) and comments records (see Section 5.5). The fields in a data stream record shall be separated by one or more delimiter characters. The first field in a record may or may not be preceded by one or more delimiter characters. The following table lists the character set from which delimiters shall be chosen:

| Symbol | Character name | ASCII code value |
|---|---|---|
| | tab | 09 hex |
| | space | 20 hex |
| , | comma | 2C hex |
| / | slash | 2F hex |
| ; | semi-colon | 3B hex |
| < | less than | 3C hex |
| = | equal sign | 3D hex |
| > | greater than | 3E hex |
| @ | at-sign | 40 hex |

FPN: The delimiter characters are listed in ascending ASCII code value sequence.

Source systems shall **not** generate tab characters. If a source system wishes to align portions of a data stream vertically, it must insert the appropriate number of space characters.

FPN: **There is no consistent interpretation of the number of spaces** per tab character. By eliminating tabs from source system data streams, we avoid the possible vertical alignment problems resulting from incorrect tab translation.

Receiving systems shall treat all delimiter characters equally. **Receiving systems shall attach no significance** to which delimiter character is used where. Anywhere a single delimiter character is permitted, receiving systems shall accept multiple different delimiter characters.

**FPN:** Based on these rules, all of the following are valid data stream records that produce the same results:

```
CUE 2.3                    CUE,;2.3
CUE=2.3                    CUE @/>=@ 2.3
```

The delimiter usage in the keyword prototypes and examples throughout this document represent the whimsy of the specification writer. Wherever a prototype or example contains a character from the delimiter character set, that character can be replaced by one or more other characters from the delimiter character set.

**FPN:** This receiving system delimiter definition simplifies data stream generation by non source systems, such as spreadsheet programs. In this case, the data stream record might read:

```
PATCH,1,1,101,100,1,102,100,2,103,80
```

**FPN:** This apparently liberal delimiter usage definition also intends encouragement of data streams that are natural for the nationality of individual system operators. Source systems shall adopt a delimiter usage scheme that will read naturally to their major customer nationality and use that scheme throughout all data streams.

For example, the following might be a natural PATCH delimiter usage in the United States:

```
PATCH 1 1<101@100, 1<102@100, 2<103@80
```

Whereas the same data stream record might more appropriately read as follows in Europe:

```
PATCH 1 1=101/100 1=102/100 2=103/80
```

Source systems should avoid creation of data streams that are difficult for people to read. Examples of data streams that should be avoided by source systems are:

```
PATCH,1,1,101,100,1,102,100,2,103,80
PATCH@1@1>101,100<1/102=100;2@103<80
```

Notes about the examples:

1. A receiving system shall process both of the above examples without errors and produce the same result as for all other PATCH examples in this section.

2. The first example record is acceptable from a general data stream source (such as a spreadsheet program).

3. However, the first example record is undesirable as the product of a source system (which is defined to be knowledgeable about the lighting console data that it is processing).

FPN: This delimiters definition has a noteworthy side effect. It requires that all fields always be present. For example, a receiving system cannot assume that omitted patch level values should be replaced with full. The receiving system cannot detect the omission of patch level values.

If optional fields are used, they must appear at the end of a record. This is the only place where the omission of an optional field can be detected by a receiving system.

All delimiter characters appearing in a record shall be counted toward the size of the record. As such, they may cause the record to be oversized. (See exception behavior in Section 5.2.)

## 5.5 Comments

The character "!" (ASCII exclamation point, 21 hex) shall terminate data stream processing for that record. Text following the exclamation point shall be ignored by a receiving system. Such text is assumed to be comments about the data stream.

The exclamation point may appear anywhere in a record. It may be the first character in the first field of the record. Then, receiving systems shall ignore the entire record. That is, receiving systems shall treat the entire record as comments text. Records that begin with an exclamation point are called comments records.

Reminder: the first field of a record may be proceeded by one or more delimiter characters. See Section 5.4.

All characters appearing after the exclamation point and the exclamation point itself shall be counted toward the size of the record. As such, they may cause the record to be oversized. (See exception behavior in Section 5.2.)

## 5.6 Keywords

The first field in each record shall be a keyword; except in zero length records (see Section 5.2) and comments records (see Section 5.5). Keywords shall contain only letters (A-Z and a-z), numbers (0-9), the special character "$" (ASCII dollar sign, 24 hex), and the special character "_" (ASCII underscore, 5F hex).

The first character in a keyword shall be either a letter or a dollar sign. When the first character of a keyword is a letter, the keyword is a standard keyword. All standard keywords shall be defined in this specification (in this or subsequent versions).

When the first character of a keyword is a dollar sign, the keyword is manufacturer-specific. All manufacturer-specific keywords shall begin will either "$" or "$$". The first character following the "$" or "$$" in a manufacturer-specific keyword shall be a letter. More information about manufacturer-specific keywords can be found in Section 5.8.

A keyword may contain as many characters as will fit in a record (see Section 5.2). However, the first 10 characters shall differentiate any keyword from any other keyword.

FPN: For example, the following two pairs of keywords are illegal because they are not unique in the appropriate number of characters:

| | |
|---|---|
| MANUFACTURER | $EFFECTS_OFF |
| MANUFACTURED | $EFFECTS_ON |

The following examples show how various keyword possibilities are unique within this definition. **FOLLOWON and FOLLOW are** different because the seventh letter is either an "O" or not present. EFFECTS and $EFFECTS are unique because the first character is either a "$" or an "E".

Receiving systems shall test just the first 10 characters of a keyword to identify it. All remaining characters in the keyword field up to the first delimiter character shall be ignored.

For standard keywords, source systems shall always generate keyword text as shown in this specification. For example, MANUFACTUR is acceptable as test keyword in a receiving system, but a source system shall generate MANUFACTURER.

Receiving systems shall ignore letter case when interpreting keywords.

FPN: For example, all of the following keywords shall be interpreted as being the same:

ENDDATA enddata Enddata EndData

Source systems shall generate all keywords in one of the following ways: all upper case, all lower case, or capitalized. A source system shall generate all keywords using the same letter casing scheme.

Source systems shall delimit the end of the keyword field with one or more spaces or a record terminator. If a source system includes delimiter characters before the keyword field, those characters shall be spaces. (See Section 5.2 for more information about record terminators. See Section 5.4 for more information about field delimiters.)

There are three types of keywords: basic, primary and secondary. Basic keywords can be processed independently from the data in any previous or subsequent record. Secondary keywords provide additional information related to the most recently processed primary keyword. Some types of exception behavior processing require knowledge of the keyword type.

More information about the various keyword types can be found in Sections 6.2 and 6.3. For information about how the three types of keywords relate to manufacturer-specific keywords see Section 5.8.

All characters in keyword text shall be counted toward the size of the record. As such, they may cause the record to be oversized. (See exception behavior in Section 5.2.)

**Exception behavior (Undefined standard keyword, WO0151):** Receipt of a standard keyword other than those defined in this specification shall cause the entire record containing the offending keyword to be ignored. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

## 5.7 Data

All fields after the keyword field shall be data fields. The keyword shall identify the type, number, format, and interpretation of the data fields. Individual data fields shall be separated from each other and from the keyword field by delimiters as described in Section 5.4.

Data fields shall contain only printable ASCII characters. The following characters shall **not** appear in any data field:

delimiter characters (see Section 5.4)
the comments separator (see Section 5.5)
() left & right parentheses (ASCII 28 & 29 hex)
{} left & right curly brackets (ASCII 7B and 7D hex)
the vertical bar (ASCII 7C hex)

All these characters either already have reserved uses in the data stream or are reserved for future expansion of the data stream definition.

FPN: Currently, all the data fields formats defined in this specification use only letters (A-Z and a-z), numbers (0-9), the special character "." (ASCII period, 2E hex), and the special character ":" (ASCII colon, 3A hex).

The common data field formats are described in Section 7. The data formats described there include; time values, cue number values, and intensity level values. The data formats described in Section 7 are referenced by the keyword descriptions in Sections 9 through 10.

When a keyword uses a common data field format, both the prototype and the descriptive text will show the section in which the common format is defined. The prototype will give the section number enclosed in curly brackets. For example:

CHAN *chan1*{7.3}*@level1*{7.5} *chan2@level2* . . .

This example also shows how a keyword prototype indicates that the keyword accepts repetition of one data field or sequence of data fields. The first listing of the data fields gives the common data field format section reference. The second listing of the fields shows just the field names. A relative position number is appended to each field name. Thus, *chan1* and *chan2* both represent *chan* (or channel) numbers. Finally, an ellipsis indicates that continued repetition is allowed.

Sometimes, one or more data fields at the end of a record may be omitted. Data fields that may be present or omitted from a record are called optional data fields. A keyword prototype indicates which data fields are optional by enclosing them in square brackets. For example:

DOWN *fade-time*{7.7} [, *delay-time*{7.7} ]

FPN: This example shows an instance where two data fields use the same common data format.

All characters in all data fields shall be counted toward the size of the record. As such, a single data field or excessive repetition of data field values can cause a record to be oversized. (See exception behavior in Section 5.2.)

## 5.8 Manufacturer specific data

Manufacturer specific keywords are the mechanism by which the definition of a data stream's contents can be extended to suit special features of individual lighting consoles. Similarly, the data stream extension may apply to a whole family of lighting consoles from a manufacturer.

Manufacturer specific keywords shall be defined as the individual manufacturer deems appropriate. These definitions should be published as part of a lighting console's documentation. Manufacturer specific keywords shall conform to the rules specified in Section 5.6. The data fields associated with manufacturer specific keywords shall conform to the rules specified in Section 5.7. Manufacturer-specific data fields may use the data field formats defined in Section 7 by referencing this specification.

A receiving system shall decide whether it can process a manufacturer-specific keyword based on the most recent MANUFACTURER (Section 8.5) and CONSOLE (Section 8.2) keyword records. The data from these two keywords shall select a list of manufacturer-specific keywords that may be encountered. This list is called the active manufacturer keyword table. Once selected, the active manufacturer keyword table shall remain in effect until other MANUFACTURER or CONSOLE keywords are encountered.

Sometimes, the MANUFACTURER or CONSOLE data will be unknown to the receiving system. Then, the receiving system has no active manufacturer keyword table. Also, the receiving system has no active manufacturer keyword table until at least one MANUFACTURER and one CONSOLE

record have been processed. When no manufacture keyword table is active, all records containing manufacturer-specific keywords are skipped. This is an exception behavior and is so noted at the end of this section.

FPN: A data stream may contain repeated instances of the MANUFACTURER and CONSOLE keywords. For example, a manufacturer-independent lighting design program might produce a data stream suitable for reading by several different consoles. Such a data stream might contain repeated usages of the MANUFACTURER and CONSOLE keywords in order to provide special data to each supported console type. For example:

```
MANUFACTURER alpha
CONSOLE      alpha
    $EFFECTS . . .
MANUFACTURER beta
CONSOLE      beta
    $EFFECTS . . .
```

When changing the active manufacturer keyword table within a data stream, the MANUFACTURER keyword can be omitted if the data it would provide is the same as that from the most recent MANUFACTURER keyword.

FPN: Based on the above rule, the following two data streams produce the same results:

```
MANUFACTURER alpha
CONSOLE      alpha
    $EFFECTS . . .
MANUFACTURER alpha
CONSOLE      beta
    $EFFECTS . . .
```

```
MANUFACTURER alpha
CONSOLE      alpha
    $EFFECTS . . .
CONSOLE      beta
    $EFFECTS . . .
```

Manufacturer specific keywords shall be differentiated into primary and not-primary based on the number of dollar signs present at the beginning of the keyword. Manufacturer specific keywords beginning with a single dollar sign are primary keywords. Manufacturer specific keywords beginning with two dollar signs are basic or secondary keywords.

FPN: The special identification of primary manufacturer-specific keywords permits standard secondary keywords to augment a primary manufacturer-specific keyword. For example:

```
CUE 2.3
  CHAN 2@25 . . .
$EFFECTS . . .
  CHAN 3@95 . . .
CUE 3.0
  CHAN . . .
```

The single dollar sign on the $EFFECTS keyword identifies it as a primary manufacturer-specific keyword. Even when a receiving system has no active manufacturer keyword table, it can recognize the $EFFECTS as a primary keyword. Because of this, the

receiving system will skip both the $EFFECTS and CHAN keywords. Without the special identification of $EFFECTS as a primary keyword, a receiving system would incorrectly include channel 3 at 95% as part of cue 2.3. This idea is discussed in greater detail in Section 6.3.

FPN: Similarly, a manufacturer-specific secondary keyword can augment a standard primary keyword. For example:

```
CUE 2.3
  CHAN 2@25 . . .
  UP . . .
  DOWN . . .
  $$FADER A
```

See Section 6.3 for more information about primary and secondary keywords.

FPN: Upon first inspection, the fact that both manufacturer-specific basic and secondary keywords begin with two dollar signs might seem to be a problem. How does a receiving system know which double dollar sign keywords to skip during the exception handling described in Section 6.3?

A review of the two applicable processing cases shows that there is no problem. The factor that defines the two applicable processing cases is whether the receiving system has an active manufacturer keyword table. Thus, the two applicable processing cases, and basic/secondary keyword handling are as follows:

1. Active manufacturer keyword table: Since the receiving system has an active manufacturer keyword table, that table can identify by name which manufacturer-specific keywords are basic, primary, and secondary. Manufacturer-specific keyword type identification is handled exactly the same as for standard keywords. The keyword text defines its type.

2. No active manufacturer keyword table: In this case, the receiving system also does not need to know the difference between basic and secondary keywords. Due to the lack of a keyword table, the receiving system is skipping all manufacturer-specific keywords. Only the distinction between primary and secondary keywords is required, in order to perform the special handling described in Section 6.3 for standard secondary keywords.

**Exception behavior (improper manufacturer-specific keyword, WO0153):** When any of the following conditions occur the entire record containing the offending manufacturer-specific keyword shall be ignored: 1) a manufacturer-specific keyword that is not in the active manufacturer keyword table is encountered, or 2) there is no active manufacturer keyword table. If the keyword is a primary manufacturer-specific keyword, subsequent secondary keyword records are also skipped as described in Section 6.3. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

# DATA STREAM ORGANIZATION AND PROCESSING

This section describes how the records in a data stream relate to each other, or how the records are organized. The intent is provision for the most flexible possible record organization within the bounds of supporting a large variety of lighting consoles.

Data streams are processed sequentially. A receiving system processes a data stream starting with the first record, continuing with the next record, and so on until the end of the data stream is reached. The data in any subsequent record may augment or overwrite the data found in any previous record.

Most primary keyword exception behaviors require that additional records be skipped. See Section 6.3. Under normal processing, however, no data stream record may cause processing to continue at any point other than the record that immediately follows it. Under no circumstances can a record cause data stream processing to continue at a previously processed record.

Processing a data stream shall not automatically clear any information in a receiving system's memory. The CLEAR keyword (Section 8.1) must be used to clear all or some of the data in the receiving system's memory. Any memory areas that are not cleared shall be changed only as keywords in the data stream cause them to be over written.

A single data stream shall describe no more than one show. When a receiving system can hold multiple shows concurrently, each show shall be loaded via a separate data stream. In such cases, the memory location for each show shall be provided either by a system operator command or by a manufacturer-specific keyword in the data stream.

## 6.1 Elementary data stream ordering considerations

A receiving system shall read and process the entire data stream before control is returned to the system operator. Only the system operator may interrupt this processing; via a CTRL-C program interrupt, for example. When the receiving system can detect such an operator interrupt, the condition shall be processed as described in the exception behavior paragraph at the end of this section.

Based on the contents of the data stream, cues, patch and other information shall be stored in the receiving system's memory. All consequences of a particular keyword ordering within the data stream shall happen before control is returned to the system operator.

FPN: For example, system operators will observe the same result from loading each of the following data streams:

```
CUE 2 . . .          PATCH . . .
PATCH . . .          CUE 2 . . .
CUE 3 . . .          CUE 3 . . .
```

In both cases, the PATCH keyword's effect will be visible immediately following completion of data stream processing. The example on the left **does not** make the console automatically repatch something after running cue 2.

Unless otherwise specified, the last occurrence of a given operation shall be the only one whose effect is observed by the system operator after data stream processing has completed.

FPN: For example, the following two data streams produce the same results:

```
SUB 1                SUB 1
CHAN 45@20           CHAN 22@100
. . .
SUB 1
CHAN 22@100
```

In both cases, submaster 1 is defined to operate channel 22, with the submaster at full raising channel 22 to full. In the example on the left, the first SUB 1 and CHAN records' actions are overwritten when the second SUB 1 record is encountered. If the intent is for SUB 1 to store both channel 22 at full and channel 45 at 20%, then one following data streams (or some equivalent) must be used:

```
SUB 1                SUB 1
CHAN 22@100          CHAN 22@100, 45@20
CHAN 45@20
```

**Exception behavior (operator interrupted data stream processing, WO0709):** If the operator interrupts data stream processing and then continues, the remainder of the data stream shall be ignored. The receiving system shall proceed as if an ENDDATA keyword (Section 8.3) has been processed. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

## 6.2 Keyword types (basic, primary, and secondary)

Keywords are categorized into three types -- basic, primary and secondary. **Basic keywords can be processed** independently from the data in any previous or subsequent record. They are discussed later in this section. Secondary keywords provide additional information related to the most recently processed primary keyword.

FPN: For example, the CHAN keyword below provides channels level information for the CUE keyword that proceeds it.

```
CUE 2
    CHAN 1@50 10@100 5@90
```

CUE is a primary keyword. CHAN is a secondary keyword.

Most primary keyword exception behaviors require a special record skipping behavior. **This and other aspects of pri**mary and secondary keyword processing are discussed in Section 6.3.

Basic keywords stand alone. No other keyword data is required to augment a basic keyword's function. The table below shows all the basic keywords. The keywords are listed in alphabetical order. The number of the section that describes each keyword is shown in parentheses.

| | | | |
|---|---|---|---|
| CLEAR | (8.1) | MANUFACTURER | (8.5) |
| CONSOLE | (8.2) | PATCH | (8.6) |
| ENDDATA | (8.3) | SET | (8.7) |
| IDENT | (8.4) | | |

Manufacturer-specific basic keywords always begin with two dollar signs ($$manufacturer_specific).

FPN: Both manufacturer-specific basic and secondary keywords begin with two dollar signs. This is because data stream processing does not require generalized differentiation of these two manufacturer-specific keyword types. See Section 5.8 for details of why this differentiation is not necessary.

## 6.3 Primary and secondary keywords

Secondary keywords provide additional information related to the most recently processed primary keyword. **The table** below shows all the primary keywords and the secondary keywords that provide additional information for those primary keywords. **All keywords are listed in alphabetical** order. The number of the section that describes each keyword is shown in parentheses.

| Primary keyword | | Secondary keyword | |
|---|---|---|---|
| CUE | (9.1) | CHAN | (10.1) |
| | | DOWN | (10.2) |
| | | FOLLOWON | (10.3) |
| | | LINK | (10.4) |
| | | PART | (10.5) |
| | | TEXT | (10.6) |
| | | UP | (10.7) |
| GROUP | (9.2) | CHAN | (10.1) |
| | | PART | (10.5) |
| | | TEXT | (10.6) |
| SUB | (9.3) | CHAN | (10.1) |
| | | DOWN | (10.2) |
| | | TEXT | (10.6) |
| | | UP | (10.7) |

Secondary keywords shall always augment the most recent primary keyword encountered in the data stream. Encountering secondary keyword that cannot augment the most recent primary keyword (as defined in the table above) shall produce the exception behavior noted at the end of this section. All secondary keywords that augment a given primary keyword shall appear after that primary keyword and before the next primary keyword in the data stream.

FPN: In accordance with this rule, the example on the left is unacceptable but the example on the right is ok:

```
CUE . . .                    CUE . . .
    UP . . .                     UP . . .
    DOWN . . .                   DOWN . . .
    CHAN . . .                   CHAN . . .
SUB . . .                        LINK . . .
    LINK . . .               SUB . . .
    CHAN . . .                   CHAN . . .
```

For manufacturer-specific keywords, the number of dollar signs at the beginning of the keyword distinguishes between primary and secondary keywords. Primary manufacturer-specific keywords begin with a single dollar sign ($manufacturer_specific). Secondary manufacturer-specific keywords begin with two dollar signs ($$manufacturer_specific).

FPN: This is a distinction that can be made without regard for the actual text of the keyword.

FPN: Like manufacturer-specific secondary keywords, manufacturer-specific basic keywords always begin with two dollar signs ($$manufacturer_specific). See Section 5.8 for details of why these two keyword types need not be differentiated.

There are special condition handling requirements for primary and secondary keywords. The purpose of this special condition handling is treating a primary keyword and all the secondary keywords that augment it as a single informational unit.

Use of this specialized condition handling will be noted in each applicable alternate interpretation and exception behavior paragraphs. These paragraphs will contain a reference to this section.

The specialized primary and secondary keyword condition handling shall be performed when both of the following two conditions are met:

1. A recoverable error or applicable alternate interpretation occurrence is detected while processing the record containing a primary keyword, and

2. The recovery for that situation involves skipping the processing activities associated with the primary keyword.

When these two conditions occur, the receiving system shall skip both the record containing the primary keyword and subsequent records containing secondary keywords. The skipping of secondary keyword records shall continue until the next primary keyword record is encountered. Any basic keywords encountered while this skipping function is in progress shall be processed normally.

No error report shall be generated for any secondary keywords record skipped due to these rules. The condition report produced by the detection of the problem in the primary keyword record is sufficient.

**Exception behavior (Secondary keyword does not augment previous primary keyword, WO0152):** When a secondary keyword follows a primary keyword that it does not augment (as defined in the table above), the entire record containing the offending secondary keyword shall be ignored. This exception behavior also shall result from encountering a secondary keyword before any primary keywords have been processed. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

## 6.4 Load With Data Translation

Receiving systems shall provide two data stream processing schemes:

1. Normal Load -- Load the data stream without replacing cue numbers, group numbers, submaster numbers, etc. The bulk of this specification defines the operation of a normal load.

2. Load with Data Translation (a.k.a. LDT) -- Load the data stream with translation that compensates for differences in console operations. This section defines how load with data translation differs from a normal load. Except for the differences described in this section, normal load and LDT work identically.

The system operator shall specify whether a normal load or a load with data translation shall occur. The specifics of how the system operator does this may differ between receiving systems. The system operator shall specify which type of load shall occur as a part of initiating a data stream reading operation. This choice shall remain in effect until data stream processing is terminated.

FPN: LDT is intended to provide users with the ability to load lighting console data even when the source system and the receiving system are very different. The objective of LDT is reading the maximum amount of data into the receiving system. The price of this capability is consistent numbering of cues, groups, parts, and submasters. Once the data is loaded into the receiving system, the system operator probably will have to edit it significantly before it is used in a performance.

LDT shall function exactly as a normal load (described elsewhere in this specification) with the following exceptions:

1. Cue numbers and cue page numbers shall be replaced. See Section 6.4.1.

2. Group numbers and group page numbers shall be replaced. If the receiving system does not support group memory, group data shall be translated to cue data. See Section 6.4.2.

3. Submaster numbers and submaster page numbers shall be replaced. If the receiving system does not support submasters or employs a common cue and submaster memory, submaster data shall be translated to cue data. See Section 6.4.3

4. Part numbers shall be replaced. See Section 6.4.4.

5. When processing the CLEAR keyword (Section 8.1), the page data field shall be ignored.

Generally, any exception behavior whose condition number is between 41 and 79 inclusive that occurs during a normal load will not occur during a load with data translation.

FPN: The purpose of defining a range of condition numbers remedied by LDT processing is two fold. First, the range definition warns receiving system implementors of exception behavior paragraphs that may require special attention during LDT processing. Second, based on the range definition, a system operator can easily identify conditions that can be remedied using LDT.

Any receiving system shall give the system operator a method for selecting full LDT processing (as described above). In addition, a receiving system may give the system operator a method of selecting one or more subsets of the full LDT processing, or even some different translation processing. For example, a receiving system might allow its operator to select just cue and group number LDT processing.

FPN: Most of the exception conditions that LDT will remedy have warning condition severity. So, the system operator may also chose using the normally loaded data instead of the LDT processed data. The objective is giving the system operator the maximum number of options in LDT situations.

### 6.4.1 Load with Data Translation - Cue Processing

During LDT cue processing, the receiving system shall ignore cue number information in the data stream and generate cue numbers and pages for each CUE record encountered. See Sections 7.4, 7.6, and 9.1 for descriptions of how cue numbers and cue page numbers are normally processed.

FPN: Using LDT processing for cues allows exchange of lighting console data between systems that employ different cue numbering schemes. Examples of such differences include systems having different maximum cue numbers and systems having different numbers of cue memory pages. The generated cue numbering scheme used by LDT accommodates receiving systems that do not support fractional cue numbers. Only whole number cue numbers are generated by LDT. Therefore, LDT supports transferring cue data from systems that have fractional cue numbering to systems that do not.

All cue numbers and pages shall be replaced with generated cue numbers and pages during LDT processing. The cue numbers and pages generated during LDT processing shall comply with the following algorithm. The cue number 1.0

page 1 shall be generated for the first CUE record encountered. The cue number 2.0 page 1 shall be generated for the second CUE record encountered.

Cue numbers 3.0, 4.0, 5.0, etc. all in page 1 shall be generated for each succeeding CUE record encountered. When the memory in cue page 1 is exhausted, generated cue numbers shall continue with 1.0 in page 2. When the memory in cue page 2 is exhausted, generated cue numbers shall continue with 1.0 in page 3.

In the absence of any other considerations, LDT cue number generation ends when available cue memory is exhausted. However, the translation of groups or submasters to cues shall reduce the amount of memory available for cues. (See Sections 6.4.2 and 6.4.3 for descriptions of how groups or submasters are translated to cues.)

When group data or submaster data are being converted to cue data, generated cue numbers shall be outside that portion of cue memory reserved for converted group or submaster data. If no unreserved cue memory is available, the "insufficient cue memory" exception behavior paragraph (below) shall apply.

FPN: If converted group and submaster cue memory is reserved in the way suggested in Sections 6.4.2 and 6.4.3, then the largest cue number generated for a cue always will be less than the smallest cue number generated for a translated group or submaster. The simplicity of this test is a reason for following the suggested reservation policy. Care must be taken to prevent the system operator from defining reservations with unused holes above the simple cue memory limit.

As a part of LDT cue number generation, the receiving system shall propagate a generated cue number to all similar uses of that cue number. Consider the following example. Suppose a record "CUE 3.2" is encountered and the cue is renumbered to 8.0. Later, a "LINK 3.2" record is encountered. Like the original CUE record, the cue number in the LINK record must have its cue 3.2 renumbered to be cue 8.0.

This is also true when the LINK record precedes the CUE record in the data stream. Thus, during LDT cue number processing, a receiving system needs a list of all original and generated cue number pairs. After all records have been processed, the receiving system can apply that cue number pair list to all appropriate non-CUE keywords found in the data stream.

Note: The proceeding requirement is stated in terms of specific implementation. This most directly describes the

required behavior. However, this specific implementation is not required. Any implementation that produces the same net effect is acceptable.

**Exception behavior (insufficient cue memory, WR0021):** If the process of adding a cue under LDT processing exhausts the available cue memory, cue data shall be discarded by the receiving system. Note: This condition also occurs when the available cue memory excluding the memory reserved for translated groups or submasters is exhausted. The condition shall be reported as noted above.

### 6.4.2 Load with Data Translation - Group Processing

During LDT group processing, the receiving system shall process group information in one of the following ways:

1. If the receiving system supports group memory, then it shall ignore group number information in the data stream and generate group numbers and pages for each GROUP record encountered.

2. If the receiving system does not support group memory, then it shall convert group data to cue data using a portion of cue memory reserved specifically for that purpose.

This section describes both of these LDT processing methods. See Sections 7.4, 7.6, and 9.2 for descriptions of how group numbers and group page numbers are normally processed.

FPN: Using LDT processing for groups allows exchange of lighting console data between systems that handle groups differently. Examples of such differences include the presence or absence of dedicated group memory and systems having different numbers of group memory pages.

In receiving systems that support group memory, all group numbers and pages shall be replaced with generated group numbers and pages during LDT processing. The group numbers and pages generated during LDT processing shall comply with the following algorithm. The group number 1.0 page 1 shall be generated for the first GROUP record encountered. The group number 2.0 page 1 shall be generated for the second GROUP record encountered.

Group numbers 3.0, 4.0, 5.0, etc. all in page 1 shall be generated for each succeeding GROUP record encountered. When the group memory in group page 1 is exhausted, gen-

erated group numbers shall continue with 1.0 in page 2. When the memory in group page 2 is exhausted, generated group numbers shall continue with 1.0 in page 3. LDT group number generation ends when available group memory is exhausted.

In receiving systems that **do not** support group memory, group data shall be converted to cue data. The effect is equivalent to treating all GROUP records as if they are CUE records.

The receiving system shall reserve a portion of cue memory for the converted group data. Optionally, the receiving system may allow the system operator to specify what portion of cue memory is reserved for converted group data. A default reservation must be provided, in case the operator does not specify one.

The following hard coded or default cue memory reservation is recommended for group data. If the receiving system does not LDT convert submaster data to cue data, use of the submaster reservation scheme (found in Section 6.4.3) is recommended. Otherwise, the two reservation schemes described below are recommended for converted group data. Receiving systems should choose the scheme that best fits their design. If none of these schemes fit the receiving system's design, these recommendations need not be followed.

Receiving systems that do not access cue memory using data stream page numbers, should reserve cue numbers between 80% and 90% of the maximum cue number value for group data conversions. Receiving systems do access cue memory pages using data stream page numbers should reserve the second to the last cue memory page for group data conversions.

The lowest reserved whole number cue value shall be assigned to the first GROUP record encountered. Incremental whole number cue values shall be assigned to each succeeding GROUP record encountered. After the maximum reserved cue value has been used, processing proceeds as defined in the "insufficient group memory" exception behavior paragraph (below).

**Exception behavior (insufficient group memory, WR0022):** If the process of adding a group under LDT processing exhausts the available group or reserved cue memory, group data shall be discarded by the receiving system. The condition shall be reported as noted above.

### 6.4.3 Load with Data Translation - Submaster Processing

During LDT submaster processing, the receiving system shall process submaster information in one of the following ways:

1. If the receiving system supports submasters, then it shall ignore submaster number information in the data stream and generate submaster numbers and pages for each SUB record encountered.

2. If the receiving system does not support submasters or employs a common cue and submaster memory, then it shall convert submaster data to cue data using a portion of cue memory reserved specifically for that purpose.

This section describes both of these LDT processing methods. See Sections 9.3 and 7.6 for descriptions of how submaster numbers and submaster page numbers are normally processed.

FPN: Using LDT processing for submasters allows exchange of lighting console data between systems that handle submasters differently. Examples of such differences include the presence or absence of submasters and systems having different numbers of submasters.

In receiving systems that support separate submasters, all submaster numbers and pages shall be replaced with generated submaster numbers and pages during LDT processing. The submaster numbers and pages generated during LDT processing shall comply with the following algorithm. The submaster number 1 page 1 shall be generated for the first SUB record encountered. The submaster number 2 page 1 shall be generated for the second SUB record encountered.

Submaster numbers 3, 4, 5, etc. all in page 1 shall be generated for each succeeding SUB record encountered. When submaster page 1 is exhausted, generated submaster numbers shall continue with 1 in page 2. When submaster page 2 is exhausted, generated submaster numbers shall continue with 1 in page 3. LDT submaster number generation ends when available submasters have been used.

In receiving systems that **do not** support separate submasters, submaster data shall be converted to cue data. The effect is equivalent to treating all SUB records as if they are CUE records.

The receiving system shall reserve a portion of cue memory for the converted submaster data. Optionally, the receiv-

ing system may allow the system operator to specify what portion of cue memory is reserved for converted submaster data. A default reservation must be provided, in case the operator does not specify one.

The two hard coded or default cue memory reservation schemes described below are recommended for converted submaster data. Receiving systems should choose the scheme that best fits their design. If none of these schemes fit the receiving system's design, these recommendations need not be followed.

Receiving systems that do not access cue memory using data stream page numbers, should reserve cue numbers between 90% and 100% of the maximum cue number value for submaster data conversions. Receiving systems do access cue memory pages using data stream page numbers should reserve the last cue memory page for submaster data conversions.

The lowest reserved whole number cue value shall be assigned to the first SUB record encountered. Incremental whole number cue values shall be assigned to each succeeding SUB record encountered. After the maximum reserved cue value has been used, processing proceeds as defined in the "too many submasters" exception behavior paragraph (below).

**Exception behavior (too many submasters, WR0023):**
If the process of adding a submaster under LDT processing exhausts the available submasters or reserved cue memory, submaster data shall be discarded by the receiving system. The condition shall be reported as noted above.

### 6.4.4 Load with Data Translation - Part Processing

During LDT cue processing, the receiving system shall ignore part number information in the data stream and generate part numbers for each PART record encountered. See Section 10.5 for a description of how part numbers are normally processed.

FPN: Using LDT processing for parts allows exchange of lighting console data between systems that allow different numbers of parts per cue.

All part numbers shall be replaced with generated part numbers during LDT processing. The part numbers generated during LDT processing shall comply with the following algorithm. Part number 1 shall be generated for the first PART record encountered in a given record collection.

Part number 2 shall be generated for the second PART record encountered in a given record collection.

Part numbers 3, 4, 5, etc. shall be generated for each succeeding PART record encountered in a given record collection. LDT part number generation ends when all available parts have been used.

**Exception behavior (too many parts, WR0024):** If the process of adding a part under LDT processing exhausts available parts, the offending PART record shall be ignored. For any excess cue parts, the effect shall be similar to that found in receiving systems that do not support parts at all. (The operation of receiving systems that do not support parts is described in an alternate interpretation paragraph in Section 10.5.) The condition shall be reported as noted above. If the condition occurs more than once in a single record collection, a report shall be generated only for the first occurrence.

# COMMON VALUE FORMATS

This section defines the formats for commonly used data fields. The first sub-section defines an exception behavior that is used for many of the data field formats. The next sub-section defines a simple integer numeric data field format. Then, several lighting console specific data field formats are defined.

## 7.1 Data field exception handling

Most exception conditions encountered within data fields employ a common exception behavior. That behavior requires skipping the remainder of the record containing the offending field. If the offending field is in a record containing a primary keyword, then processing is skipped for that record and subsequent secondary keyword records as described in Section 6.3. The exception behavior paragraph below formally defines this behavior.

FPN: The choice to ignore the remainder of the record containing the offending field is based on some assumptions about the data fields in the record. The first assumption is that the offending field is above the allowed range. The second assumption is that future incarnations of the same data field also will be above the permitted range, because the data stream was created with a monotonically increasing value in that field. Of course, ignoring the remainder of the record also is easier to implement.

**Exception behavior (data field error):** When an exception behavior paragraph references this section, the following processing shall be performed. **Processing for the remainder** of the record that contains the offending field shall be skipped. **If the record containing the exception also** contains a primary keyword, that record and subsequent secondary keyword records shall be skipped as described in Section 6.3.

The situation shall be reported to the system operator based on the information in the original exception behavior paragraph (the one that references this section). The situation shall be reported if all of the following conditions are met: 1) The original exception behavior paragraph contains condition reporting information, and 2) The receiving system provides condition reporting of the type described in the original exception behavior paragraph.

## 7.2 Integer data field format

The integer data field format is simply an integer numeric value. Its ASCII representation and exception behavior definitions are provided in this section. This section is referenced for all integer data field values.

An integer data field shall contain only numeric characters, 0-9. The data field shall contain from 1 to 5 characters. The number value expressed in an integer data field shall be in decimal radix. The smallest value expressed in an integer data field shall be 0. The largest value shall be 65535. (Some specific usages of integer data fields may further restrict this value range.)

Receiving systems shall permit but not require leading zeros in integer data fields. Thus, 010 and 10 are the same integer data values. Unless otherwise specified, source systems shall never add leading zeros to integer data fields.

**Exception behavior (non-numeric character, WO0171):** If a non-numeric character (a character other than 0-9) is encountered in an integer data field, exception processing shall proceed as described in Section 7.1. **The exception** reporting information for that processing is shown above.

**Exception behavior (integer value out of range, WO0173):** If an integer value less than 0 or greater than 65535 is encountered, exception processing shall proceed as described in Section 7.1. The exception reporting information for that processing is shown above.

## 7.3 Channel data field format

A channel is a controllable lighting level unit. Generally, one or more dimmers are patched to a channel using one or more PATCH keywords (Section 8.6). Lighting levels can then be specified on a per channel basis using CHAN keywords (Section 10.1).

Channel data is formatted as an integer value whose range is more restricted than that of a general integer value. Channel data fields shall be processed as described in Section 7.2, with the range restrictions defined in the exception behavior paragraph below. Usually, the smallest permissible channel value is 1. But, for the PATCH keyword, a channel value of 0 is permissible.

**Exception behavior (channel out of range, WO0201):** When a channel value of 0 is encountered for any keyword other than PATCH, exception processing shall proceed as described in Section 7.1. When a channel value exceeding the receiving system's limit or the value set by the most recent SET CHANNELS record (which ever is smaller) is encountered, exception processing shall proceed as described in Section 7.1. The exception reporting information for that processing is shown above.

## 7.4 Cue or group number data field format

Cue numbers identify each cue in the data stream. Group numbers identify each group in the data stream. Both numbers use the same basic format. Receiving systems shall store cues and groups so that they are executed in ascending cue/group number order. **The order that cues and** groups appear in the data stream shall not affect the order in which they are stored or executed.

When the system operator selects load with data translation, cue and group numbers are ignored. See Section 6.4 for more information about load with data translation processing.

Cue or group number prototype format:

*whole*{7.2}[.*tenths*{7.2}] -- *whole*[.*tenths*]

Example cue and group numbers:

```
0   0.0    2   2.0  8.5  101  102.9   9999.9
```

A cue or group number data field shall be composed of two integer parts. The first part is the whole number. The second part is the tenths. **The whole number part shall** always be present. The tenths part is optional.

The whole number part shall be processed as an integer value (as described in Section 7.2). The smallest permissible whole number part value is 0. The largest permissible whole number part value is 9999.

When present, the tenths part shall be preceded by a decimal point (or ASCII period, 2E hex). The absence of a decimal point shall indicate that the tenths part of the cue number is not available. When a tenths part is required but none is present, ".0" shall be used. Therefore, cue number 1 is equivalent to cue number 1.0.

That portion of the tenths part following the decimal point shall be processed as an integer value (as described in Section 7.2). The smallest permissible tenths part value is 0. The largest permissible tenths part value is 9. Note: The tenths part is restricted to a single numeric digit.

Cue and group numbers 0 and 0.0 shall be considered invalid.

Source systems that support the tenths part cue or group numbers shall always include tenths in the data streams that they generate. Source systems that do not support the tenths part shall always generate whole number cue or group numbers (i.e., the tenths part shall be omitted).

**Alternate interpretation (tenths not allowed in cue or group numbers, WR0041):** Receiving systems that do not allow a tenths part in cue numbers or group numbers shall apply the "invalid cue or group number" exception behavior paragraph (below) to any such cue or group numbers encountered.

**Exception behavior (invalid cue or group number, WR0041):** If a data field that should contain a cue or group number contains 0, 0.0, or text that does not have the format defined above, exception processing shall proceed as described in Section 7.1. The exception reporting information for that processing is shown above.

**Exception behavior (cue or group number too large, WR0042):** If the whole number part of a cue or group number is greater than the receiving system permits (or greater than 9999, which ever is smaller), exception processing shall proceed as described in Section 7.1. The exception reporting information for that processing is shown above.

FPN: Both the "invalid cue or group number" and the "cue or group number too large" exception behaviors usually can be remedied using load with data translation. To do this, the system operator must reprocess the data stream using the load with data translation option. See Section 6.4 for details of load with data translation processing.

## 7.5 Level data field format

Level values generally indicate a dimmer or channel intensity setting in terms of a percentage of full. In this form, level values range from 0 to 100. Such level values can be applied to channels in CHAN records (Section 10.1), or to electronic patch setups in PATCH records (Section 8.6). There are, however, some additional needs that add complexity to level value specifications.

Sometimes it is useful to express levels over 100%. In complex electronic patching situations, the combination of the patch level and the channel level determines the actual dimmer level. So long as the result is less than 100%, one or the other of the patch and channel levels can be greater than 100%.

In more sophisticated lighting situations, level values can manipulate mechanical systems such as gel scrollers and moving luminaries. Here, the resolution of a 0 to 100 value is inadequate. For example, a change from 0 to 1 in a moving luminaries position level amounts to 3.6 degrees of movement (assuming a 360 degree movement capability in the luminaries).

To provide the greatest possible flexibility for expressing level information, two level formats are defined:

1) Whole-number percentage (www)
2) Hexadecimal value (Hnn or Hnnnn)

Each format is described in one of the sub-sections below. Because of the alternate interpretations provided, the only required format is a whole number percentage (Section 7.5.1) between 0 and 100.

The whole number percentage format is the most universally recognizable format, but it has the lowest resolution. The hexadecimal value format is exact, but not universally accepted. These facts should be considered when a source system chooses how level values are generated.

**Exception behavior (invalid level field, WO0221):** If a field that should contain level data is not formatted as described in any of the sub-sections below, exception processing shall proceed as described in Section 7.1. The exception reporting information for that processing is shown above.

### 7.5.1 Whole-number percentage level format

A level can be represented as a whole-number percentage between 0 and 999. The whole-number percentage shall be processed as described in Section 7.2. **All receiving** systems shall accept whole-number percentages between 0 and 100. Additionally, receiving systems may support whole-number percentages between 101 and 999.

Conversion of whole-number percentage values between 0 and 100 to 8-bit binary values shall be performed as specified in Table C-2.

**Alternate interpretation (levels above 100 not supported):** If a receiving system does not support level values above 100, it shall convert all level values between 101 and 999 that it encounters to 100.

**Exception behavior (level too large):** If a level value greater than the receiving system's maximum between 101 and 999 is encountered, it shall be converted to the maximum level.

### 7.5.2 Hexadecimal level format

The hexadecimal level format provides a means for the data stream to contain exact binary level values for a channel or dimmer. There is no conversion formula for changing hexadecimal data in to binary level data. However, any 8-bit binary level can be converted to percentage level values using Table C-1.

Hexadecimal level prototype format:

*Hxxxx* *hxxxx* *Hxx* *hxx*

Example hexadecimal level values:

H9ABC h9AbC H8F

Receiving systems shall accept both upper and lower case for all letters in hexadecimal values. The discussions and definitions below all use upper case. However, a data stream can contain either case.

A level can be represented as a hexadecimal value by preceding the hexadecimal numeric value with an H. This identifies the level value as being in hexadecimal format. The H shall be followed by 2 or 4 hexadecimal digits. Hexadecimal digits are numbers (0-9) and the first six letters of the alphabet (A-F or a-f).

Receiving systems shall accept leading zeros in a hexadecimal value. So, H009F and H9F are the same hexadecimal level value. The leading zeros shall be considered significant because they indicate the field size intended by the data stream source.

Source systems shall generate hexadecimal fields having a number of digits that indicate the size of the datum that produced the field data. If the datum is a byte, then the hexadecimal field shall contain 2 digits (e.g., H04). If the datum is a two-byte integer, then the hexadecimal field shall contain 4 digits (e.g., H0004).

**Alternate interpretation (hexadecimal levels not supported, WO0225):** If a receiving system that does not support hexadecimal level values encounters one, it shall treat the condition as an exception and process it as described in Section 7.1. The exception reporting information for that processing is shown above.

**Exception behavior (invalid hexadecimal field, WO0172):** If any of the following conditions are encountered in a hexadecimal field, exception processing shall proceed as described in Section 7.1; 1) a character other than 0-9, A-F, or a-f, or 2) more than 4 digits. The exception reporting information for that processing is shown above.

**Exception behavior (1 or 3 digits in field):** If a hexadecimal field containing 1 or 3 digits is encountered, leading zeros shall be added until the field has the necessary number of digits.

**Exception behavior (4 digits supplied for a one-byte datum):** If 4 digit hexadecimal field is encountered for a one-byte datum (two hexadecimal digits), the least significant two hexadecimal digits shall be discarded. Processing shall continue using the truncated data value.

**Exception behavior (hexadecimal value to large):** If a hexadecimal value is too large (after application of the exception behaviors in the previous paragraphs), the largest acceptable value shall be substituted.

## 7.6 Data page number field format

Many lighting consoles allow some of their data to be organized in pages, numbered collections of similar information. For example, a console might have two sets of submaster information called page 1 and page 2. To cover all possibilities, this specification defines pages for cues, groups, submasters, and dimmer-to-channel patches.

With the exception of the PATCH keyword (Section 8.6), page number usage is optional.

A page number datum is formatted as an integer value. It shall be processed as described in Section 7.2. A page number of 0 is out of range. The largest valid page number is a basic characteristic of the receiving system.

Page numbers of 2, 3, etc. may be supported for any data type. Different data types may have different numbers of pages supported. For example, a receiving system may have 2 pages of submaster data but only 1 page of cue, group, and patch data. So, the validity of a page number depends both on the receiving system and on what data type is being processed.

Receiving systems shall start the page numbering for each data type with page 1. Lack of any support for a given data type shall be dealt with before page numbers are processed.

Two exception behaviors are provided for those cases where multiple pages are not supported. The first applies to all cases except patch pages. The second applies only to patch pages.

**Exception behavior (page number out of range, WR0043):** When a page number of 0 or larger than is supported is encountered, exception processing shall proceed as described in Section 7.1. The exception reporting information for that processing is shown above.

FPN: The "page number out of range" exception behavior usually can be remedied using load with data translation. To do this, the system operator must reprocess the data stream using the load with data translation option. See Section 6.4 for more information about load with data translation processing.

**Exception behavior (patch page skipped, WO0231):** When a PATCH keyword page number of 0 or larger than the receiving system supports is encountered, the PATCH record containing the offending page number shall be skipped. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

## 7.7 Time data field format

Time values indicate the amount of time that should elapse before, during, or after execution of some operation.

Time prototype format:

[[*hours*{7.2}:]*minutes*{7.2}:]*seconds*{7.2}[.*tenths*{7.2}]

-- or --

[[*hours*:]*minutes*:]*seconds*[.*tenths*]

Example times:

| | |
|---|---|
| 30 | 30 seconds |
| 1:00 | 1 minute |
| 2:00:00 | 2 hours |
| 2:00:10 | 2 hours and 10 seconds |
| 0.5 | 5 tenths of a second |
| 8.2 | 8 and 2 tenths seconds |
| 3:00:13.5 | 3 hours 13 and one half seconds |

A time field shall consist of from one to three integer values separated by colons (ASCII 3A hex). These values shall be the hours part, the minutes part, and the seconds part, respectively. The hours and minutes parts are optional. The seconds part shall always be present. Optionally, the seconds part may be followed by a decimal point (or ASCII period, 2E hex) and a tenths part.

If the hours part is present, it shall be followed by a colon and a minutes part. If the minutes part is present, it shall be followed by colon, which shall separate it from the seconds part (whose presence is required). If the tenths part is present, it shall be proceeded by a decimal point.

The absence of any colons in a time field shall indicate that both the hours and the minutes parts are not present. The presence of only one colon in a time field shall indicate that the hours part is not present. The absence of a decimal point in a time field shall indicate that the tenths part is not present.

All four integers shall be processed as described in Section 7.2. The hours, minutes, and seconds parts shall range from 0 to 999. The tenths part shall range from 0 to 9. Note: The tenths part is restricted to a single numeric digit.

Receiving systems shall allow omission of leading zeros in the hours, minutes, and seconds parts. This means that 0:3 represents 3 seconds, not 30 seconds.

**Alternate interpretation (large times not supported):** If a receiving system encounters a time that is larger than it supports, that time shall be converted to the largest time supported by the receiving system.

**Exception behavior (invalid time field, WO0241):** If a field that should contain time data is not formatted as described above, exception processing shall proceed as described in Section 7.1. The exception reporting information for that processing is shown above.

# 8 BASIC KEYWORDS

This section describes each basic keyword that may be present in a lighting console data stream. Basic keywords stand alone. No other keyword data is required to augment a basic keyword's function.

A data stream source may send all or none of the keywords described in this section. If sent, the keywords may be present in any order within a data stream. Most keywords may be sent repeatedly. For convenient reference, the basic keywords are described in alphabetical order.

## 8.1 CLEAR

Format prototype:

CLEAR *clear-item* [*page*{7.6}]

Examples:

```
CLEAR ALL
CLEAR CUES
CLEAR SUBMASTERS 2
CLEAR $EFFECTS
```

The CLEAR keyword instructs the receiving system to remove all previously stored information in all or a specified subsection of its memory. If a data stream contains no CLEAR keywords, then no memory is cleared. Information in the data stream is added to or overwrites information already present in the receiving system.

FPN: Use of a CLEAR keyword in the middle of a data stream, may have the unintended effect of clearing the effects of previously processed keywords.

The optional *page* datum shall be processed as described in Section 7.6. When present, the *page* datum shall limit the clearing to a single page. When *page* is omitted, all pages shall be cleared.

FPN: For example:

| | |
|---|---|
| CLEAR SUBMASTERS | clears all pages of submaster data |
| CLEAR SUBMASTERS 2 | clears only the second page of submaster data |

During load with data translation processing, the *page* datum shall be ignored. See Section 6.4 for details of load with data translation processing.

*Clear-item* shall be a text string that identifies the memory subsection to be cleared. Valid ***clear-item*** strings are:

ALL
CUES
GROUPS
PATCH
SUBMASTERS
$*manufacture-specific-item*

The *clear-item* field shall be formatted, generated and tested according to the equivalent rules for keywords, described in Section 5.6.

CLEAR ALL clears previously stored programmable information in all memory subsections. If a *page* datum is present, it shall be used wherever applicable and shall inhibit the clearing operation wherever it is not. The effect is identical with that caused by a data stream containing a CLEAR record for every currently valid *clear-item*.

FPN: The effect of CLEAR ALL can change depending on the most recently encountered MANUFACTURER (Section 8.5) and CONSOLE (Section 8.2) keywords.

CLEAR CUES deletes cue information.

CLEAR GROUPS deletes group information.

CLEAR PATCH clears dimmer-to-channel patch and level information. Dimmers are unpatched.

CLEAR SUBMASTERS clears submaster information.

CLEAR $*manufacturer-specific-item* clears some manufacturer specific subsection of the receiving system's memory. The *manufacturer-specific-item* shall be formatted, generated and tested according to the equivalent rules for manufacturer-specific keywords, described in Section 5.8. Among other things, this means that the interpretation of *manufacturer-specific-item* depends on the most recently processed MANUFACTURER and CONSOLE keywords.

FPN: During load with data translation (LDT) processing, group or submaster data may be converted into cue data. The handling of a CLEAR GROUPS or a CLEAR SUBMASTERS record during LDT has no standard definition. This omission will be fixed in the next version. Currently, the recommended practice is clearing the cue number range to which the group or submaster data is being converted. LDT is discussed in Section 6.4.

**Alternate interpretation (memory subsection not present):** If a receiving system encounters a *clear-item* naming a memory subsection that it does not have, the record containing the CLEAR keyword shall be ignored.

## 8.2 CONSOLE

Format prototype:

CONSOLE *console-identifier*

Examples:

```
CONSOLE ACCSPRO
CONSOLE PALT2
CONSOLE EXP
```

Combined with the data from the most recent MANUFACTURER (Section 8.5) keyword, *console-identifier* uniquely identifies the interpretation that shall be applied to subsequent manufacturer-specific keywords. **See Section 5.8** for details of manufacturer-specific keyword processing.

*Console-identifier* shall be text string of from 1 to 10 characters. *Console-identifier* shall be composed only of letters (A-Z or a-z) and numbers (0-9). Receiving systems shall ignore letter case when testing *console-identifier* strings. **Manufacturers should publish the *console-identifier* strings** used by their systems.

If a source system generates manufacturer-specific keyword records, then it shall generate both a MANUFACTURER record and a CONSOLE record before the first manufacturer-specific record. **If a source system does not generate** any manufacturer-specific keyword records, then the MANUFACTURER and CONSOLE keywords may be omitted.

**Alternate interpretation (manufacturer-specific data ignored): A receiving system may ignore all MANUFACTURER** and CONSOLE keywords in a data stream. If that is done, then the receiving system is in a perpetual state of having no manufacturer keyword table active (a state defined in Section 5.8). In this state, all manufacturer-specific keywords are skipped.

**Exception behavior (CONSOLE before MANUFACTURER, WO0154):** When a CONSOLE keyword is encountered before the first MANUFACTURER keyword, the record containing the CONSOLE keyword shall be skipped. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

## 8.3 ENDDATA

Format prototype:

ENDDATA

The ENDDATA keyword marks the end of a data stream. A receiving system shall ignore all data received after an ENDDATA keyword until the system operator initiates another data stream reading operation. Source systems shall always terminate a data stream with an ENDDATA keyword.

FPN: A receiving system must be prepared for a data stream to be terminated by an end-of-file marker. However, this is an exception condition. See Section 5.3.

FPN: The ENDDATA keyword facilitates transmission of data streams via mechanisms that do not provide for end-of-file markers. A network is an example of such a data transmission mechanism.

The ENDDATA keyword has no data fields.

## 8.4 IDENT

Format prototype:

IDENT *major*{7.2}:*minor*{7.2}

Examples:

IDENT 1:263
IDENT 3:0

The IDENT keyword identifies the version of this specification used to create the data stream. The IDENT keyword data field shall be composed of two integer parts separated by a colon (ASCII 3A hex). Both parts shall be processed as integer values (as described in Section 7.2).

The first part is called *major* ident. **The *major* ident** changes only when older data streams cannot be processed by newer receiving systems. The second part is called *minor* ident. The *minor* ident changes every time this specification is enhanced.

When the *minor* ident changes but the *major* ident remains the same, the new specification is a superset of the old specification. This is called an upward compatible change. **Upward compatible changes mean that new receiving** systems can process older data streams. Making upward

compatible changes is a goal of the USITT for this specification.

The ident for this version of the specification is shown on the title page. For approved versions, the ident and the version (also shown on the title page) shall contain the same values (with a different separator). For draft versions, the title page shall show the ident that will be used when the draft is approved. Draft versions will never have idents assigned.

Source systems shall always begin a data stream with an IDENT record. This does not guarantee that all data streams will have an IDENT record. Data stream sources that are not source systems may omit the record. However, its inclusion is recommended for data streams that will be saved for long periods of time.

Receiving systems shall compare the *major* and *minor* ident values to those for the specification that defines their operation. **Processing shall proceed when the following** two conditions are met:

1) Both *major* idents match, and
2) **The *minor* ident from the IDENT record is less than or** equal to the minor ident that defines the receiving system's operation.

**Exception behavior (ident mismatch, ER0099):** If either of the *major* or *minor* ident tests described above fail, processing shall be terminated. **This means that the** data stream contains newer format information that the receiving system cannot process. **The condition shall** be reported as noted above.

## 8.5 MANUFACTURER

Format prototype:

MANUFACTURER *manufacturer-identifier*

Examples:

MANUFACTURER STRAND
MANUFACTURER ETC
MANUFACTURER COLORTRAN

Combined with the data from a subsequent CONSOLE (Section 8.2) keyword, *manufacture-name* uniquely identifies the interpretation that shall be applied to subsequent man-

ufacturer-specific keywords. See Section 5.8 for details of manufacturer-specific keyword processing.

*Manufacturer-identifier* shall be text string of from 1 to 10 characters. *Manufacturer-identifier* shall be composed only of letters (A-Z or a-z) and numbers (0-9). Receiving systems shall ignore letter case when testing *manufacturer-identifier* strings. Manufacturers should publish their *manufacturer-identifier* string.

If a source system generates manufacturer-specific keyword records, then it shall generate both a MANUFACTURER record and a CONSOLE record before the first manufacturer-specific record. If a source system does not generate any manufacturer-specific keyword records, then the MANUFACTURER and CONSOLE keywords may be omitted.

**Alternate interpretation (manufacturer-specific data ignored):** A receiving system may ignore all MANUFACTURER and CONSOLE keywords in a data stream. If that is done, then the receiving system is in a perpetual state of having no manufacturer keyword table active (a state defined in Section 5.8). In this state, all manufacturer-specific keywords are skipped.

## 8.6 PATCH

Format prototype:

```
PATCH page{7.6} chan1{7.3}<dim1{7.2}@level1{7.5} ...
PATCH page chan1<dim1@level1 chan2<dim2@level2 . . .
```

Examples:

```
PATCH 1 1<1@100 2<2@100 3<3@100 4<4@100
PATCH 1 1<53@33 1<78@90 1<82@25 1<101@100
```

The PATCH keyword shall specify dimmer-to-channel patch information.

The *page* datum shall be processed as described in Section 7.6. The *page* datum specifies the patch memory page where the dimmer-to-channel patch data is to be stored. Source systems that do not have pages of patch memory shall generate a 1 for the *page* datum. Note: This is the only case where the *page* datum is required, not optional.

A repeated sequence of *chan, dim* and *level* fields shall provide the dimmer-to-channel patch data.

The *chan* datum is a channel number. It shall be processed as described in Section 7.3.

The *dim* datum is an integer dimmer number. It shall be processed as described in Section 7.2. The *dim* datum shall range from 1 to the receiving system's maximum dimmer number or the value set by the last SET DIMMERS record (whichever is smaller).

The *level* datum is a level value. It shall be processed as described in Section 7.5. Source systems that do not have a proportional patch shall generate 100 for the *level* datum. An alternate interpretation paragraph is provided below for receiving systems that do not have a proportional patch memory.

A single sequence entry, *chan*<*dim*@*level*, shall patch dimmer *dim* into channel *chan* at level *level*. This sequence may be repeated as many times as will fit in a record. See Section 5.2 for the definition of a record's size.

FPN: For example:

```
PATCH 1 1<101@50 2<201@100 1<102@75 2<202@75
```

shall patch:

```
dimmer 101 to channel 1 at 50%,
dimmer 102 to channel 1 at 75%,
dimmer 201 to channel 2 at 100%,
dimmer 202 to channel 2 at 75%.
```

Note: There are no restrictions on the order in which *chan*<*dim*@*level* sequences appear.

Receiving systems shall allow multiple dimmers to be patched into a single channel (as shown by many of the PATCH examples). There shall be no restrictions on the number of dimmers that can be patched into a single channel.

The ability to patch a single dimmer into multiple channels shall be the default interpretation. However, an alternate interpretation paragraph is provided below for receiving systems that do not allow a single dimmer to be patched into more than one channel.

A dimmer may be patched to channel zero; 0<*dim*@*level*. This shall cause the dimmer to be patched to no channel. So, 0<50@0 will unpatch dimmer 50. The *level* datum must be present although it is irrelevant. (See the end of Section 5.4 for an explanation of this requirement.) Source systems shall generate 0 for the *level* datum when the *chan* datum is 0. Receiving systems shall ignore the *level* datum when the *chan* datum is 0.

**Alternate interpretation, (no softpatch):** Receiving systems that do not support dimmer-to-channel softpatch shall ignore the PATCH commands.

**Alternate interpretation (no proportional patch):** Receiving systems that do not support proportional patch shall ignore all *level* data. In effect, the level is always 100.

**Alternate interpretation (dimmer patched to too many channels):** **A receiving system can overwrite previously** stored dimmer-to-channel patch data when all the following conditions occur; 1) the new *dim* datum names a dimmer that is already patched, and 2) the named dimmer is patched to too many channels.

The definition of, "patched to too many channels," is a receiving system characteristic. **Some receiving systems** allow only a single channel per dimmer. Other receiving systems may allow multiple channels per dimmer. If multiple channels are allowed per dimmer, then the oldest patch data shall be over written.

**Exception behavior (dimmer out of range, WO0203):** When a *dim* datum of 0 or in excess of the receiving system's limit or the value set by the most recent SET DIMMERS record (which ever is smaller) is encountered, the remainder of the record containing the errant *dim* datum shall be ignored. **If reporting is provided for optional** type conditions, the situation shall be reported as noted above.

## 8.7 SET

Format prototype:

```
SET set-item value{7.2}
SET PATCH DEFAULT
```

Examples:

```
SET CHANNELS 300
SET DIMMERS 1000
SET $BUBBLES 50000
```

The SET keyword instructs the receiving system to set a major system parameter or piece of system state.

*Set-item* shall be a text string that identifies what is being set. **Where applicable, *value* shall be an integer,** processed as described in Section 7.2. *Value* shall

provide the value to which the major system parameter named by *set-item* is set. Valid *set-item* strings are:

CHANNELS
DIMMERS
PATCH
$*manufacture-specific-item*

The *set-item* field shall be formatted, generated and tested according to the equivalent rules for keywords, described in Section 5.6.

SET CHANNELS *value* sets the maximum legal channel number for all subsequent CHAN (Section 10.1) and PATCH (Section 8.6) keywords. *Value* shall range from 1 to the receiving systems maximum channel number.

SET DIMMERS *value* sets the maximum legal dimmer number for all subsequent PATCH (Section 8.6) keywords. *Value* shall range from 1 to the receiving systems maximum dimmer number.

SET PATCH DEFAULT sets the dimmer-to-channel patch to the receiving system's default. The three fields in this record must appear exactly as shown here (except for letter casing).

FPN: Usually, a receiving system's default patch maps dimmer 1 to channel 1 at full, dimmer 2 to channel 2, etc.

SET $*manufacturer-specific-item* sets some manufacturer specific major system parameter. The *manufacturer-specific-item* shall be formatted, generated and tested according to the equivalent rules for manufacturer-specific keywords, described in Section 5.8. Among other things, this means that the interpretation of *manufacturer-specific-item* depends on the most recently processed MANUFACTURER and CONSOLE keywords.

**Exception behavior (value field out of range, WO0302):** If a SET record has a *value* field that is out of range (either 0 or too large), the receiving system's maximum value shall be used. If reporting is provided for optional type conditions, the situation shall be reported as noted above.

## 8.8 $$manufacturer_specific

Keywords beginning with two dollar signs are manufacturer-specific basic keywords (or manufacturer-specific secondary keywords). Manufacturer-specific keywords allow extension of the data stream format for a manufacturer's special features. They are discussed in Section 5.8.

**Alternate interpretation (manufacturer-specific basic keywords not supported):** Receiving systems may ignore all manufacturer-specific keywords that begin with two dollar signs, which includes all basic manufacturer-specific keywords. However, manufacturer-specific keywords beginning with a single dollar sign shall initiate the primary-secondary record skipping behavior described in Section 6.3.

# 9 PRIMARY KEYWORDS

This section describes each primary keyword that may be present in a lighting console data stream. A primary keyword record heads a collection of secondary keyword records. The complete record collection (primary plus all secondaries) describes one cue, one group, or one submaster. The collection is terminated by another primary keyword record or the end of the data stream (Section 5.3).

Section 6.2 gives an overview of primary and secondary keywords. Section 6.3 discusses the special processing requirements associated with primary and secondary keywords. Each section below includes a list of the secondary keywords can be in a record collection headed by that primary keyword. The presence of other secondary keywords in a record collection shall initiate the exception behavior described in Section 6.3.

A data stream source may send all or none of the keywords described in this section. If sent, the keywords may be present in any order within a data stream. However, the primary to secondary keyword relationships discussed above must be maintained. Keywords may be sent repeatedly. For convenient reference, the primary keywords are described in alphabetical order.

## 9.1 CUE

Format prototype:

CUE *cue-number*{7.4} [*page*{7.6}]

Examples:

```
CUE 2
CUE 3.0
CUE 1.0  2
```

Applicable secondary keywords:

| | | | |
|---|---|---|---|
| CHAN | (10.1) | PART | (10.5) |
| DOWN | (10.2) | TEXT | (10.6) |
| FOLLOWON | (10.3) | UP | (10.7) |
| LINK | (10.4) | | |

The CUE keyword shall head a record collection describing one cue. The CUE keyword record identifies the cue. Subsequent secondary keyword records shall define the

channel levels, timing, parts, and linkages to other cues for the cue named in the CUE keyword record.

The *cue-number* datum is processed as described in Section 7.4. The *page* datum is processed as described in Section 7.6. Taken together, *cue-number* and *page* identify the cue being described by the subsequent secondary keyword records.

Source systems shall generate cue information in ascending cue number and page number order. This will produce better load with data translation processing, in the event that the data stream is read in that way. See Section 6.4 for details of load with data translation.

Receiving systems shall store cues so that they are executed in ascending cue number order. The order that cues appear in the data stream shall not affect the order in which they are stored or executed. A repeated occurrence of a given *cue-number* and *page* in the data stream shall over write that cue. This behavior is defined in Section 6.1.

If a load with data translation is in progress, both *cue-number* and *page* shall be replaced with generated values. See Section 6.4 for details of load with data translation.

All applicable secondary keywords may be present or absent. CHAN keyword records shall specify the levels of **all** channels that are above zero in the cue. If necessary, the receiving system shall convert these to "moving fade" representation (each cue containing only the differences from its predecessor). Similarly, source systems shall convert "moving fade" channel levels to the format required in a data stream.

FPN: A CUE record collection that contains no CHAN records represents a blackout cue. So, based on the overwrite behavior defined above and in Section 6.1, the following sequence produces a cue 5 that is a blackout cue with a down fade time of 2 seconds:

```
CUE 5
CHAN 10@50, 15@100, 22@42
CUE . . .
. . .
CUE 5
DOWN 2
CUE . . .
. . .
```

FPN: Questions have been raised about the following situation:

```
CUE 32
$$EFFECT 2
CUE 38
. . .
```

What does this do when the $$EFFECT record can be processed? What does it do when $$EFFECT is not in the active manufacturer keyword table? These questions will be addressed in the next version of the standard. In the interim, creation of a blackout cue number 32 is recommended, unless receiving system implementation details suggest otherwise.

If the CUE record collection contains neither a DOWN nor an UP keyword, then cross fading shall be controlled manually. If only one of DOWN or UP is present, then both down and up fade times shall be set to the data that record contains. If both DOWN and UP are present, a split fade shall be setup.

Source systems that do not have timed faders shall omit both DOWN and UP keywords from the data streams that they produce. For a non-split fade, source systems may use just a DOWN or just an UP record. Both records are required only for split fades.

If the CUE record collection contains a LINK keyword, then the data in that record shall name the next cue executed. If no LINK keyword is present, then the next higher numbered cue shall be the next cue executed.

If the CUE record collection contains a FOLLOWON keyword, then the data in that record shall set the interval from initiation of the current cue until initiation of the next cue. The current cue is the one described by the current CUE record collection. When the FOLLOWON keyword is present, initiation of the next cue shall occur automatically (after the specified interval). If the FOLLOWON keyword is absent, the next cue shall be initiated by some manual action.

**Alternate interpretation (no manual fades):** Receiving systems that do not provide manual fades shall substitute a system operator supplied default fade time when neither a DOWN nor an UP keyword appears in a CUE record collection.

**Alternate interpretation (no timed fades):** Receiving systems that do not have timed faders shall ignore all DOWN and UP keyword records.

**Alternate interpretation (split fades not supported):** Receiving systems that do not support split fades shall

use the fade time from the UP record and ignore the fade time from the DOWN record.

**Alternate interpretation (no LINK support):** Receiving systems that do not support LINK altered cue execution sequences shall ignore any LINK records that they encounter.

**Alternate interpretation (no FOLLOWON support):** Receiving systems that do not support timed automatic initiation of cues shall ignore any FOLLOWON records that they encounter.

**Exception behavior (insufficient cue memory, WR0021):** If the process of inserting, adding, or modifying a cue exhausts cue memory, cue data shall be discarded by the receiving system. The condition shall be reported as noted above.

Note: Additional alternate interpretation and exception behavior paragraphs appear in the sections describing each applicable secondary keyword.

## 9.2 GROUP

Format prototype:

GROUP *group-number*{7.4} [*page*{7.6}]

Examples:

```
GROUP 2
GROUP 4.3
GROUP 5.0  2
```

Applicable secondary keywords:

```
CHAN (10.1)              TEXT (10.6)
PART (10.5)
```

The GROUP keyword shall head a record collection describing one group. A group is levels memory that is neither a cue nor a submaster. The exact interpretation of a group varies from one receiving system to the next. The GROUP keyword record identifies the group. Subsequent secondary keyword records shall define the channel levels and parts for the group named in the GROUP keyword record. Groups cannot have fader timings or linkages to other groups.

The *group-number* datum is processed as described in Section 7.4. The *page* datum is processed as described in

Section 7.6. Taken together, *group-number* and *page* identify the group being described by the subsequent secondary keyword records.

Source systems shall generate group information in ascending group number and page order. This will produce better load with data translation processing, in the event that the data stream is read in that way. See Section 6.4 for details of load with data translation.

A repeated occurrence of a given *group-number* and *page* in the data stream shall over write that group. If a load with data translation is in progress, both *group-number* and *page* shall be replaced with generated values. See Section 6.4 for details of load with data translation.

All applicable secondary keywords may be present or absent. CHAN keyword records shall specify the levels of all channels that are above zero in the group.

FPN: A group record collection that contains no CHAN records represents a blackout group.

**Exception behavior (GROUPs not supported, WR0051):** Receiving systems that do not support groups shall discard all group data. **The condition shall be reported as noted** above.

FPN: The "GROUPs not supported" exception behavior can be remedied using load with data translation. To do this, the system operator must reprocess the data stream using the load with data translation option. See Section 6.4 for more information about load with data translation processing.

**Exception behavior (insufficient group memory, WR0022):** If the process of inserting, adding, or modifying a group exhausts the available group memory, group data shall be discarded by the receiving system. **The condition shall be** reported as noted above.

Note: Additional alternate interpretation and exception behavior paragraphs appear in the sections describing each applicable secondary keyword.

## 9.3 SUB

Format prototype:

SUB *sub-number*{7.2} [*page*{7.6}]

Examples:

```
SUB 22
SUB 1  2
```

Applicable secondary keywords:

| | |
|---|---|
| CHAN (10.1) | TEXT (10.6) |
| DOWN (10.2) | UP (10.7) |

The SUB keyword shall head a record collection describing one submaster. The SUB keyword record identifies the submaster. **Subsequent secondary keyword records shall** define the channel levels and fader timing for the submaster named in the SUB keyword record. Submasters cannot have parts or linkages to other submasters.

The *sub-number* datum is processed as described in Section 7.2. The *sub-number* datum shall range from 1 to a receiving system specific maximum. The *page* datum is processed as described in Section 7.6. Taken together, *sub-number* and *page* identify the submaster being described by the subsequent secondary keyword records.

Source systems shall generate submaster information in ascending submaster number and page number order. **This** will produce better load with data translation processing, in the event that the data stream is read in that way. See Section 6.4 for details of load with data translation.

A repeated occurrence of a given *sub-number* and *page* in the data stream shall over write that submaster. This behavior is defined in Section 6.1.

If a load with data translation is in progress, both *sub-number* and *page* shall be replaced with generated values. See Section 6.4 for details of load with data translation.

All applicable secondary keywords may be present or absent. CHAN keyword records shall specify the levels of **all** channels that are above zero in the submaster.

FPN: A SUB record collection that contains no CHAN records represents a blackout submaster. So, based on the overwrite behavior defined above and in Section 6.1, the following

sequence produces a submaster 7 that is a blackout with a down fade time of 2 seconds:

```
SUB 7
CHAN 1@100, 2@75, 3@90
SUB . . .
. . .
SUB 7
DOWN 2
SUB . . .
. . .
```

If a DOWN record is present, it shall set down fade time for the submaster. If an UP record is present, it shall set the up fade time for the submaster. When either a DOWN or an UP record is missing, that fade time shall be controlled manually. If both the DOWN and UP records are absent, all submaster fading shall be controlled manually.

**Alternate interpretation (common cue and submaster memory):** Some receiving systems do not make a distinction between cue memories and submaster memories. For example, cue 100 and submaster 100 are the same levels. Such systems shall convert submaster data to cue data, so long as submaster numbers do not conflict with cue numbers. In effect, the SUB keyword record shall become a CUE record (see Section 9.1). If a conflict between cue numbers and submaster numbers is encountered, the "cues and submasters not supported" exception behavior paragraph (below) shall apply.

**Alternate interpretation (timed submaster fades not supported):** Receiving system that do not support timed fades or split timed fades on submasters shall ignore any DOWN or UP records that are present in a SUB record collection, as appropriate.

**Exception behavior (SUBmasters not supported, WR0052):** Receiving systems that do not support submasters shall discard all submaster data. The condition shall be reported as noted above.

**Exception behavior (submaster number out of range, WR0044):** If a *sub-number* datum exceeds the range supported by the receiving system, the data for that submaster shall be discarded. The condition shall be reported as noted above.

**Exception behavior (cues and submasters not supported, ER0071):** When a receiving system that uses common cue and submaster memories encounters a conflict between cue numbers and submaster numbers, data stream processing

shall be terminated. The condition shall be reported as noted above.

FPN: The "SUBmasters not supported," "submaster number out of range," and "cues and submasters not supported" exception behaviors usually can be remedied using load with data translation. To do this, the system operator must reprocess the data stream using the load with data translation option. See Section 6.4 for more information about load with data translation processing.

Note: Additional alternate interpretation and exception behavior paragraphs appear in the sections describing each applicable secondary keyword.

## 9.4 $manufacturer_specific

Keywords beginning with one dollar sign are manufacturer-specific primary keywords. Manufacturer-specific keywords allow extension of the data stream format for a manufacturer's special features. They are discussed in Section 5.8.

Receiving systems cannot totally ignore primary manufacturer-specific keywords. At a minimum, a receiving system that is otherwise ignoring manufacturer-specific keywords shall initiate the primary-secondary record skipping behavior described in Section 6.3 when it encounters a primary manufacturer-specific keyword (beginning with a single dollar sign).

# 10 SECONDARY KEYWORDS

This section describes each secondary keyword that may be present in a lighting console data stream. Secondary keywords provide additional information related to the most recently processed primary keyword. See Sections 6.2 and 6.3 for more information about primary and secondary keywords.

Each section below includes a list of the primary keywords for which the defined secondary keyword can provide additional data. The presence of a secondary keyword in a record collection headed by a primary keyword other than those listed shall initiate the exception behavior included in Section 6.3. Section 9 defines the term, "record collection."

A data stream source may send all or none of the keywords described in this section. If sent, the keywords may be present in any order within a data stream. However, the primary to secondary keyword relationships discussed above must be maintained. Keywords may be sent repeatedly. For convenient reference, the secondary keywords are described in alphabetical order.

## 10.1 CHAN

Format prototype:

CHAN *chan1*{7.3}@*level1*{7.5} *chan2*@*level2* . . .

Examples:

```
CHAN 1@100 2@75 4@50 5@25 6@H0F 7@22
CHAN 5@90 100@2 3@60
```

Primary keywords augmented:

| | | | |
|---|---|---|---|
| CUE | (9.1) | SUB | (9.3) |
| GROUP | (9.2) | | |

A CHAN keyword record shall specify channel level data for the CUE, GROUP, or SUB record collection in which it appears.

A repeated sequence of *chan* and *level* fields shall provide the channel level data. The *chan* datum is a channel number. It shall be processed as described in Section 7.3. The *level* datum is a level value. It shall be processed

as described in Section 7.5. A single entry, *chan@level*, shall set channel *chan* to level *level*.

FPN: For example:
CHAN 13@100 14@56 12@75

shall set:
channel 12 to 75%,
channel 13 to 100%,
channel 14 to 56%.

Note: There are no restrictions on the order in which *chan@level* sequences appear.

Channels may be omitted from the list. If no CHAN record in an entire record collection gives a value for a channel, that channel shall be set to zero.

**Exception behavior (channel set to multiple levels):** If one channel is set to multiple levels in a single record collection, the last level value encountered shall be used. Note: This exception behavior may be initiated by receiving systems that do not support parts. The PART secondary keyword is discussed in Section 10.5.

## 10.2 DOWN

Format prototype:

DOWN *fade-time*{7.7} [, *delay-time*{7.7} ]

Examples:

DOWN 5:00
DOWN 5
DOWN 3:33, 2.6

Primary keywords augmented:

CUE (9.1) SUB (9.3)

A DOWN keyword record shall specify the down fade-time for the CUE or SUB record collection in which it appears. Optionally, a DOWN record can specify the down fade delay time for that same record collection. The delay time is the interval that passes from the time that cue execution is initiated until the time that the down fade commences.

Both *fade-time* and *delay-time* shall be processed as described in Section 7.7. If the DOWN record is terminated before a *delay-time* field is encountered, then processing

shall proceed as if a *delay-time* of zero had been present. Record termination is discussed in Section 5.2.

**Alternate interpretation (no timed fades):** Receiving systems that do not have timed faders shall ignore all DOWN keyword records. Note: This alternate interpretation may be applied differently for CUE and SUB record collections. That is, cues may have timed fades while submasters do not.

**Alternate interpretation (no delay support):** Receiving systems that do not support delaying timed fade initiation shall ignore all *delay-time* information encountered. Note: This alternate interpretation may be applied differently for CUE and SUB record collections. That is, submasters may use *delay-time* while cues do not.

**Exception behavior (multiple DOWN records in a collection):** If a CUE or SUB record collection contains more than one DOWN record, only the last DOWN record shall set the fade timing information. Note: This exception behavior may be initiated by receiving systems that do not support parts. The PART secondary keyword is discussed in Section 10.5.

## 10.3 FOLLOWON

Format prototype:

FOLLOWON *delay-time*{7.7}

Examples:

```
FOLLOWON 1.5
FOLLOWON 1:30
```

Primary keywords augmented:

CUE (9.1)

A FOLLOWON keyword record shall specify the interval from initiation of the current cue until automatic initiation of the next cue. The current cue is the one described by the CUE record collection in which the FOLLOWON record appears. If a CUE record collection contains no FOLLOWON keyword, then initiation of the next cue shall require some manual action.

The *delay-time* datum shall be processed as described in Section 7.7. The *delay-time* datum shall specify the

interval from when current cue execution is initiated until automatic initiation of the next cue occurs.

Note: The FOLLOWON *delay-time* shall be unaffected by the delay times found in DOWN (Section 10.2) or UP (Section 10.7) records.

**Alternate interpretation (no FOLLOWON support):** Receiving systems that do not support timed automatic initiation of cues shall ignore any FOLLOWON records that they encounter.

FPN: Some receiving systems do not support a FOLLOWON *delay-time* that is less than the largest fade time (down or up) for the whole cue. This case is not officially covered in the Version 3.0 standard. The recommended practice is increasing the *delay-time* to the smallest value that is acceptable to the receiving system. This recommendation will be proposed as new alternate interpretation paragraph in the next voting draft of the standard.

## 10.4 LINK

Format prototype:

LINK *cue-number*{7.4}

Examples:

LINK 2
LINK 8.3

Primary keywords augmented:

CUE (9.1)

A LINK keyword record shall name next cue executed following the current cue. The current cue is the one described by the CUE record collection in which the LINK keyword appears. If a CUE record collection contains no LINK keyword, then the next higher numbered cue shall be the next cue executed.

The *cue-number* datum shall be processed as described in Section 7.4. It shall name the next cue to be executed. If a load with data translation is in progress, the replacement of the *cue-number* value may be necessary. See Section 6.4 for details of load with data translation.

**Alternate interpretation (no LINK support):** Receiving systems that do not support LINK altered cue execution

sequences shall ignore any LINK records that they encounter.

**FPN:** Some receiving systems support only automatically initiated cue linkages. For such receiving systems, any cue record collection that contains a LINK record also must contain a **FOLLOWON** record. Version 3.0 of the standard does not address this receiving system requirement. The recommended practice is simulation of the missing FOLLOWON record with the largest *delay-time* value that the receiving system supports. This recommendation will be proposed as new alternate interpretation paragraph in the next voting draft of the standard.

## 10.5 PART

Format prototype:

PART *part-number*{7.2}

Examples:

PART 1
PART 8

Primary keywords augmented:

CUE (9.1) GROUP (9.2)

A PART keyword record shall separate a CUE or GROUP record collection into multiple parts. If a PART record appears anywhere in a record collection, there shall be a PART record immediately following the CUE or GROUP record. Currently, there is no valid interpretation for data appearing between the CUE or GROUP record and the first PART record.

The data between any two PART records or between a PART record and the end of the record collection is called a "cue part." Receiving systems shall accommodate at least 3 cue parts in a record collection.

The *part-number* datum shall be processed as described in Section 7.2. **It shall specify a numeric value that is the** name for the cue part that follows.

Source system shall generate *part-number* values between 1 and 99. Source systems shall generate parts in ascending part number order. **This will produce better load with** data translation processing, in the event that the data stream is read in that way. See Section 6.4 for details of load with data translation.

Receiving systems shall store parts in increasing numeric value order. The order that parts appear in the data stream shall not affect the order in which they are stored. Receiving systems shall accept *part-number* values between 1 and a system specific maximum. The maximum *part-number* value accepted shall be greater than or equal to maximum number of parts allowed in a single cue.

A repeated occurrence of a given *part-number* in a record collection shall over write that part. If a load with data translation is in progress, the *part-number* shall be replaced with a generated value. See Section 6.4 for details of load with data translation.

The presence of PART records in a record collection produces several important effects that must be noted. Each cue part can contain DOWN (Section 10.2) and UP (Section 10.7) records. A given channel can be assigned different levels in CHAN (Section 10.1) records in each cue part. This is a proper situation for receiving systems that support parts. When parts are not supported, however, the presence of this extra data will initiate exception behaviors for each keyword involved.

**Alternate interpretation (PARTs not supported):** Receiving systems that do not support division of cues or groups in to parts shall ignore any PART records that they encounter. As noted above, usage of this alternate interpretation probably will result in exception behavior conditions being encountered in some CHAN, DOWN, and UP keyword processing. The exception behavior for each of these cases appears in Sections 10.1, 10.2 and 10.7, respectively.

FPN: Some receiving systems do not allow a channel to appear in multiple parts. This restriction is not sanctioned in Version 3.0 of the standard. However, an alternate interpretation paragraph covering this case is being considered for the next voting draft. For now, the recommended practice in such cases is ignoring all duplicate channel levels except the one in either the lowest or the highest numbered part.

**Exception behavior (part number out of range, WR0045):** If a *part-number* of 0 or in excess of the receiving system's maximum value is encountered, the offending PART record shall be skipped. The condition shall be reported as noted above.

FPN: The "part number out of range" exception behavior usually can be remedied using load with data translation. To do this, the system operator must reprocess the data stream using the load with data translation option. See Section 6.4 for more information about load with data translation processing.

**Exception behavior (PART not first after CUE or GROUP):** If a PART record is encountered when PART was not the first keyword after a CUE or GROUP record collection header, that PART record and all other PART records in that record collection shall be ignored.

**Exception behavior (too many parts, WR0024):** Receiving systems that encounter more PART records in one record collection than they support shall ignore all cue parts in excess of their supported limit. For the excess cue parts, the effect shall be similar to that found in receiving systems that do not support parts at all. If reporting is provided for optional type conditions, the situation shall be reported as noted above. If the condition occurs more than once in a single record collection, a report shall be generated only for the first occurrence.

**Exception behavior (insufficient cue memory, WR0021):** If the process of inserting, adding, or modifying a cue part exhausts cue memory, all remaining cue data shall be discarded by the receiving system. **The condition shall be** reported as noted above.

**Exception behavior (insufficient group memory, WR0022):** If the process of inserting, adding, or modifying a group exhausts group memory, all remaining group data shall be discarded by the receiving system. The condition shall be reported as noted above.

## 10.6 TEXT

Format prototype:

TEXT *multi-field-text-datum*

Examples:

```
TEXT Blackout cue at the end of ACT I
TEXT House interior group
TEXT Drummer reds submaster
```

Primary keywords augmented:

| | | | |
|---|---|---|---|
| CUE | (9.1) | SUB | (9.3) |
| GROUP | (9.2) | | |

A TEXT keyword record shall provide text commentary about the CUE, GROUP, or SUB record collection in which it appears. **The receiving system shall store this commentary** in association with the cue, group, or submaster. The

commentary shall be displayed to the system operator in some appropriate form.

The *multi-field-text-datum* begins with the first non-delimiter character after the keyword. It continues until a comments character or record terminator is encountered. Delimiters are discussed in Section 5.4. The comments character and comments in general are discussed in Section 5.5. Record termination is discussed in Section 5.2.

**Alternate interpretation (commentary text not supported):** Receiving systems that do not support alphanumeric commentary for cues, groups, or submasters shall ignore TEXT keyword records, as appropriate. Note: Commentary text may be supported in some cases but not others. For example, submasters may support commentary text but not cues or groups.

**Exception behavior (commentary text too long):** Characters in excess of the receiving system's limit shall be ignored. Note: The record size and field delimiting rules stated in Sections 5.2 and 5.4 limit commentary text to 75 characters.

## 10.7 UP

Format prototype:

```
UP fade-time{7.7} [, delay-time{7.7} ]
```

Examples:

```
UP 3:00
UP 3
UP 4:45, 1.2
```

Primary keywords augmented:

CUE (9.1) SUB (9.3)

A UP keyword record shall specify the up fade-time for the CUE or SUB record collection in which it appears. Optionally, a UP record can specify the up fade delay time for that same record collection. The delay time is the interval that passes from the time that cue execution is initiated until the time that the up fade commences.

Both *fade-time* and *delay-time* shall be processed as described in Section 7.7. If the UP record is terminated before a *delay-time* field is encountered, then processing

shall proceed as if a *delay-time* of zero had been present. Record termination is discussed in Section 5.2.

**Alternate interpretation (no timed fades):** Receiving systems that do not have timed faders shall ignore all UP keyword records. Note: This alternate interpretation may be applied differently for CUE and SUB record collections. That is, cues may have timed fades while submasters do not.

**Alternate interpretation (no delay support):** Receiving systems that do not support delaying timed fade initiation shall ignore and *delay-time* information encountered. Note: This alternate interpretation may be applied differently for CUE and SUB record collections. That is, submasters may use *delay-time* while cues do not.

**Exception behavior (multiple UP records in a collection):** If a CUE or SUB record collection contains more than one UP record, only the last UP record shall set the fade timing information. Note: This exception behavior may be initiated by receiving systems that do not support parts. The PART secondary keyword is discussed in Section 10.5.

## 10.8 $$manufacturer_specific

Keywords beginning with two dollar signs are manufacturer-specific secondary keywords (or manufacturer-specific basic keywords). Manufacturer-specific keywords allow extension of the data stream format for a manufacturer's special features. They are discussed in Section 5.8.

**Alternate interpretation (manufacturer-specific basic keywords not supported): Receiving systems may ignore** all manufacturer-specific keywords that begin with two dollar signs, which includes all secondary manufacturer-specific keywords. However, manufacturer-specific keywords beginning with a single dollar sign shall initiate the primary-secondary record skipping behavior described in Section 6.3.

# A KEYWORD PROCESSING STATE TABLE

The table below shows the relationships between basic, primary, and secondary keywords in a state table format. This is another way of presenting the ideas found in Sections 5.8, 6.2, and 6.3. Should there be any discrepancies between the representation here and the non-appendix text in the specification, the text Sections 5.8, 6.2, and 6.3 shall be considered correct.

Each state represents a situation that can occur during the processing of a data stream. For example, the BDS (beginning of data stream) state represents conditions before any data stream records have been processed. Or, the CUE-COL (cue record collection) state represents conditions after a CUE keyword is processed, but before any other primary keyword is processed.

For each state, every keyword is listed. The actions performed as a result of encountering that keyword are described. Then, the state that processing will enter after that keyword has been processed is named. Thus, for any existing state, the state table shows what happens when any next keyword is processed.

The four states beginning with NEW- require additional explanation. These NEW- states simplify the state table representation of the transition from one primary keyword record collection to the next. Combined with the entries that reference them, these NEW- states provide a way for the state table to say both, "finish processing the previous record collection," and, "begin processing the new record collection." Specifically, the NEW- states make the second statement.

For example, the state NEW-GROUP is entered after previous record collection processing is finished as a GROUP keyword record is encountered. Like the other NEW- states, the NEW-GROUP state lists no keywords. This is because the NEW-GROUP state only covers the action of processing the GROUP keyword that has just been encountered.

| ASCII Text Representation for Lighting Console Data Keyword Processing State Table | | |
|---|---|---|
| Keyword | Action | New State |
| State: **BDS** *(beginning of data stream)* | | |
| CLEAR<br>CONSOLE<br>IDENT<br>MANUFACTURER<br>PATCH<br>SET<br>$$m-specific | Process basic keyword as specified in Section 8 | NO-PRIMARY |
| ENDDATA<br>(end-of-file) | Finish data stream processing | Exit state table |
| CUE | | NEW-CUE |
| GROUP | | NEW-GROUP |
| SUB | | NEW-SUBMASTER |
| $m-specific | | NEW-MPRIMARY |
| CHAN<br>DOWN<br>FOLLOWON<br>LINK<br>PART<br>TEXT<br>UP<br>$$m-specific | Exception Behavior<br>. . . WO0152 . . .<br>See Section 6.3 | NO-PRIMARY |

| ASCII Text Representation for Lighting Console Data<br>Keyword Processing State Table | | |
|---|---|---|
| Keyword | Action | New State |
| State: **NO-PRIMARY** *(no primary keyword encountered, yet)* | | |
| CLEAR<br>CONSOLE<br>IDENT<br>MANUFACTURER<br>PATCH<br>SET<br>$$m-specific | Process basic keyword as specified in Section 8 | NO-PRIMARY |
| ENDDATA<br>(end-of-file) | Finish data stream processing | Exit state table |
| CUE | | NEW-CUE |
| GROUP | | NEW-GROUP |
| SUB | | NEW-SUBMASTER |
| $m-specific | | NEW-MPRIMARY |
| CHAN<br>DOWN<br>FOLLOWON<br>LINK<br>PART<br>TEXT<br>UP<br>$$m-specific | Exception Behavior<br>. . . WO0152 . . .<br>See Section 6.3 | NO-PRIMARY |

| ASCII Text Representation for Lighting Console Data<br>Keyword Processing State Table | | |
|---|---|---|
| Keyword | Action | New State |
| State: **NEW-CUE** | | |
| | Start new cue processing as described in Section 9.1 | CUE-COL |
| State: **CUE-COL** *(CUE record collection)* | | |
| CLEAR<br>CONSOLE<br>IDENT<br>MANUFACTURER<br>PATCH<br>SET<br>$$m-specific | Process basic keyword as specified in Section 8 | CUE-COL |
| ENDDATA<br>(end-of-file) | Finish cue and data stream processing | Exit state table |
| CUE | Finish cue processing | NEW-CUE |
| GROUP | Finish cue processing | NEW-GROUP |
| SUB | Finish cue processing | NEW-SUBMASTER |
| $m-specific | Finish cue processing | NEW-MPRIMARY |
| CHAN<br>DOWN<br>FOLLOWON<br>LINK<br>PART<br>TEXT<br>UP<br>$$m-specific | Process secondary keyword as specified in Section 10 | CUE-COL |

| ASCII Text Representation for Lighting Console Data Keyword Processing State Table | | |
|---|---|---|
| Keyword | Action | New State |
| State: **NEW-GROUP** | | |
| | Start new group processing as described in Section 9.2 | GROUP-COL |
| State: **GROUP-COL** *(GROUP record collection)* | | |
| CLEAR<br>CONSOLE<br>IDENT<br>MANUFACTURER<br>PATCH<br>SET<br>$$m-specific | Process basic keyword as specified in Section 8 | GROUP-COL |
| ENDDATA<br>(end-of-file) | Finish group and data stream processing | Exit state table |
| CUE | Finish group processing | NEW-CUE |
| GROUP | Finish group processing | NEW-GROUP |
| SUB | Finish group processing | NEW-SUBMASTER |
| $m-specific | Finish group processing | NEW-MPRIMARY |
| CHAN<br>PART<br>TEXT<br>$$m-specific | Process secondary keyword as specified in Section 10 | GROUP-COL |
| DOWN<br>FOLLOWON<br>LINK<br>UP | Exception Behavior<br>. . . WO0152 . . .<br>See Section 6.3 | GROUP-COL |

| ASCII Text Representation for Lighting Console Data<br>Keyword Processing State Table | | |
|---|---|---|
| Keyword | Action | New State |
| State: **NEW-SUBMASTER** | | |
| | Start new submaster processing as described in Section 9.3 | SUBMASTER-COL |
| State: **SUBMASTER-COL** *(SUB record collection)* | | |
| CLEAR<br>CONSOLE<br>IDENT<br>MANUFACTURER<br>PATCH<br>SET<br>$$m-specific | Process basic keyword as specified in Section 8 | SUBMASTER-COL |
| ENDDATA<br>(end-of-file) | Finish submaster and data stream processing | Exit state table |
| CUE | Finish submaster processing | NEW-CUE |
| GROUP | Finish submaster processing | NEW-GROUP |
| SUB | Finish submaster processing | NEW-SUBMASTER |
| $m-specific | Finish submaster processing | NEW-MPRIMARY |
| CHAN<br>DOWN<br>TEXT<br>UP<br>$$m-specific | Process secondary keyword as specified in Section 10 | SUBMASTER-COL |
| FOLLOWON<br>LINK<br>PART | Exception Behavior<br>. . . WO0152 . . .<br>See Section 6.3 | SUBMASTER-COL |

| ASCII Text Representation for Lighting Console Data Keyword Processing State Table | | |
|---|---|---|
| Keyword | Action | New State |
| State: **NEW-MPRIMARY** | | |
| | Start new manufacturer specific record collection processing | MPRIMARY-COL |
| State: **MPRIMARY-COL** *(manufacturer specific record collection)* | | |
| CLEAR<br>CONSOLE<br>IDENT<br>MANUFACTURER<br>PATCH<br>SET<br>$$m-specific | Process basic keyword as specified in Section 8 | MPRIMARY-COL |
| ENDDATA<br>(end-of-file) | Finish manufacturer specific and data stream processing | Exit state table |
| CUE | Finish manufacturer specific processing | NEW-CUE |
| GROUP | Finish manufacturer specific processing | NEW-GROUP |
| SUB | Finish manufacturer specific processing | NEW-SUBMASTER |
| $m-specific | Finish manufacturer specific processing | NEW-MPRIMARY |
| CHAN<br>DOWN<br>FOLLOWON<br>LINK<br>PART<br>TEXT<br>UP<br>$$m-specific | Based on manufacturer keyword table:<br>Process secondary keyword as specified in Section 10<br>-- OR --<br>Exception Behavior<br>. . . WO0152 . . .<br>See Section 6.3 | MPRIMARY-COL |

# B REPORTED CONDITIONS TABLE

The table below lists reporting information for exception and alternate interpretation conditions. The conditions are ordered based ascending numerical value of condition numbers. The severity and reporting type are shown for each condition.

A suggested message text is also shown for each condition. This message text is suggested for usage by receiving systems that provide Level 3 condition reporting (see Section 4). Many suggested text lines contain either /keyword/ or /field/. These notations signify text that should be replaced by information from the data stream record that caused the condition.

Consider the following example. Suppose record 88 contains a cue number of 8.A. This is incorrect because only numbers can appear after the decimal point (condition WO0041). The suggested text for this condition reads:

Invalid cue or group number /field/

A receiving system that uses Level 3 condition reporting might display the following message (based on the suggested message text):

(0088) 0041-W Invalid cue or group number /8.A/

Condition numbers are allocated to the various condition severity and reporting type combinations according to the following table:

| | Required | Optional |
|---|---|---|
| Informational | 0000 | 1000-9999 |
| Warning | 0001-0069 | 0150-0999 |
| Error | 0070-0099 | 0100-0149 |

Condition numbers 0041 through 0079 are reserved for situations that usually can be remedied by using load with data translation. See Section 6.4 for details of load with data translation processing.

Receiving systems that detect error conditions not otherwise listed in the table below may report them using error condition number 0111. Receiving systems that detect warning conditions not otherwise listed in the table below may report them using warning condition number 0911. These two non-specific condition numbers permit reporting of non-standard errors or warnings that a receiving system

might encounter. One example of such situations is errors or warnings that occur during the processing of manufacturer-specific keywords.

FPN: The two non-specific conditions numbers described above were added during the voing review for Version 3.0. They are placeholders for a more general reporting mechanism for manufacturer-specific conditions. The more general mechanism will be introduced in the next voting draft of the standard. The more general mechanism will reduce the range of condition numbers allocated to optional informational conditions from 1000-9999 to 1000-4999. The conditions numbers taken away from optional informational conditions will be allocated to manufacturer-specific uses. If the planned changes are adopted, the condition numbers allocation table will read as follows:

| | Required | Optional | Manufacturer Specific |
|---|---|---|---|
| Informational | 0000 | 1000-4999 | 5000-5999 |
| Warning | 0001-0069 | 0150-0999 | 6000-7999 |
| Error | 0070-0099 | 0100-0149 | 8000-9999 |

Under this scheme, the use of manufacturer-specific condition numbers requires Level 2 condition reporting. See Section 4.2 for more information about Level 2 condition reporting.

| Cond. No. | S | T | Suggested Message Text |
|---|---|---|---|
| 0000 | I | O | Data stream successfully processed |
| | | | |
| 0021 | W | R | Cue memory exhausted |
| 0022 | W | R | Group memory exhausted |
| 0023 | W | R | Too many submasters |
| 0024 | W | R | Too many parts |
| | | | |
| 0041 | W | R | Invalid cue or group number /field/ |
| 0042 | W | R | Cue or group number /field/ too large |
| 0043 | W | R | Page /field/ out of range |
| 0044 | W | R | Submaster number /field/ out of range |
| 0045 | W | R | Part number /field/ out of range |
| | | | |
| 0051 | W | R | Groups not supported |
| 0052 | W | R | Submasters not supported |

| Cond. No. | S | T | Suggested Message Text |
|---|---|---|---|
| 0071 | E | R | Cues and submasters not supported |
| | | | |
| 0091 | E | R | Encountered record longer than 80 characters |
| 0099 | E | R | Ident mismatch prohibits processing |
| 0100 | E | O | Data stream terminated without ENDDATA |
| 0111 | E | O | Unidentified error condition encountered |
| | | | |
| 0151 | W | O | Undefined standard keyword /keyword/ |
| 0152 | W | O | Wrong secondary keyword /keyword/ |
| 0153 | W | O | Improper manufacturer keyword /keyword/ |
| 0154 | W | O | CONSOLE found before first MANUFACTURER |
| 0171 | W | O | Invalid integer /field/ |
| 0172 | W | O | Invalid hexadecimal value /field/ |
| 0173 | W | O | Integer value /field/ out of range |
| | | | |
| 0201 | W | O | Channel /field/ out of range |
| 0203 | W | O | Dimmer /field/ out of range |
| 0221 | W | O | Invalid level /field/ |
| 0225 | W | O | Hexadecimal levels not supported |
| 0231 | W | O | Patch page /field/ skipped |
| 0241 | W | O | Invalid time /field/ |
| | | | |
| 0302 | W | O | SET value /field/ out of range |
| | | | |
| 0701 | W | O | Ignored character with high-bit set |
| 0702 | W | O | Ignored non-printing character |
| 0709 | W | O | Data stream processing interrupted |
| 0911 | W | O | Unidentified warning condition encountered |

# C DMX TO/FROM LEVEL PERCENTAGE CONVERSIONS

The following tables show the conversions between percentage levels DMX data values. For each table, a mathematical equation provides an algorithmic method for obtaining the values shown.

Both algorithms use the Round(x) function. This is the standard mathematical rounding of the value x to an integer. Some examples are:

```
Round(1.49) = 1
Round(1.50) = 2
Round(1.51) = 2

Round(2.49) = 2
Round(2.50) = 3
Round(2.51) = 3
```

The tables are based on the work of Mr. Jon Farley performed at Entertainment Technology, Inc. The original work can be found in the Entertainment Technology, Inc. Application Note, AN-002; dated September 27, 1990. Direct written requests to:

```
Entertainment Technology, Inc.
1771 N.W. Pettygrove
Portland, OR 97209
```

C language support for these tables is available on the USITT CallBoard. The directory where the supporting download files are stored is:

```
>udd>USITT>common>standards>light_cues
```

Two download files are available. DMXLEVEL.H is a C language include file that defines two conversion arrays, one for each of the two tables in this appendix. Array C_DMX_to_Level converts a DMX level to a percentage level. It is used in the following way:

```
Percentage = C_DMX_to_Level[ DMX_Level ] ;
```

Array C_Level_to_DMX converts a percentage level to a DMX level. It is used in the following way:

```
DMX_Level = C_Level_to_DMX[ Percentage ] ;
```

DMXLTEST.C is a C program that validates these two arrays.

The table below shows the conversion from DMX data values (0 to FFh) to level percentages (0 to 100%). The table is based on the formula:

Percentage = Round( DMX_level * (100/255) )

Asterisks mark those conversions that also appear in the Percentage Level to DMX Data Value Conversion table.

Table C-1 -- DMX Data Value to Percentage Level Conversion

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 * | 2C | 17 | 58 | 35 | 84 | 52 | B0 | 69 * | DC | 86 |
| 1 | 0 | 2D | 18 | 59 | 35 * | 85 | 52 * | B1 | 69 | DD | 87 |
| 2 | 1 | 2E | 18 * | 5A | 35 | 86 | 53 | B2 | 70 | DE | 87 * |
| 3 | 1 * | 2F | 18 | 5B | 36 | 87 | 53 * | B3 | 70 * | DF | 87 |
| 4 | 2 | | | 5C | 36 * | | | B4 | 71 | | |
| 5 | 2 * | 30 | 19 * | 5D | 36 | 88 | 53 | B5 | 71 * | E0 | 88 * |
| 6 | 2 | 31 | 19 | 5E | 37 * | 89 | 54 | B6 | 71 | E1 | 88 |
| 7 | 3 | 32 | 20 | 5F | 37 | 8A | 54 * | B7 | 72 | E2 | 89 |
| | | 33 | 20 * | | | 8B | 55 | | | E3 | 89 * |
| 8 | 3 * | 34 | 20 | 60 | 38 | 8C | 55 * | B8 | 72 * | E4 | 89 |
| 9 | 4 | 35 | 21 | 61 | 38 * | 8D | 55 | B9 | 73 | E5 | 90 |
| A | 4 * | 36 | 21 * | 62 | 38 | 8E | 56 | BA | 73 * | E6 | 90 * |
| B | 4 | 37 | 22 | 63 | 39 * | 8F | 56 * | BB | 73 | E7 | 91 |
| C | 5 | | | 64 | 39 | | | BC | 74 | | |
| D | 5 * | 38 | 22 * | 65 | 40 | 90 | 56 | BD | 74 * | E8 | 91 * |
| E | 5 | 39 | 22 | 66 | 40 * | 91 | 57 * | BE | 75 | E9 | 91 |
| F | 6 * | 3A | 23 | 67 | 40 | 92 | 57 | BF | 75 * | EA | 92 |
| | | 3B | 23 * | | | 93 | 58 | | | EB | 92 * |
| 10 | 6 | 3C | 24 | 68 | 41 | 94 | 58 * | C0 | 75 | EC | 93 |
| 11 | 7 | 3D | 24 * | 69 | 41 * | 95 | 58 | C1 | 76 | ED | 93 * |
| 12 | 7 * | 3E | 24 | 6A | 42 | 96 | 59 * | C2 | 76 * | EE | 93 |
| 13 | 7 | 3F | 25 | 6B | 42 * | 97 | 59 | C3 | 76 | EF | 94 |
| 14 | 8 * | | | 6C | 42 | | | C4 | 77 * | | |
| 15 | 8 | 40 | 25 * | 6D | 43 | 98 | 60 | C5 | 77 | F0 | 94 * |
| 16 | 9 | 41 | 25 | 6E | 43 * | 99 | 60 * | C6 | 78 | F1 | 95 |
| 17 | 9 * | 42 | 26 * | 6F | 44 | 9A | 60 | C7 | 78 * | F2 | 95 * |
| | | 43 | 26 | | | 9B | 61 | | | F3 | 95 |
| 18 | 9 | 44 | 27 | 70 | 44 * | 9C | 61 * | C8 | 78 | F4 | 96 |
| 19 | 10 | 45 | 27 * | 71 | 44 | 9D | 62 | C9 | 79 * | F5 | 96 * |
| 1A | 10 * | 46 | 27 | 72 | 45 | 9E | 62 * | CA | 79 | F6 | 96 |
| 1B | 11 | 47 | 28 * | 73 | 45 * | 9F | 62 | CB | 80 | F7 | 97 * |
| 1C | 11 * | | | 74 | 45 | | | CC | 80 * | | |
| 1D | 11 | 48 | 28 | 75 | 46 * | A0 | 63 | CD | 80 | F8 | 97 |
| 1E | 12 | 49 | 29 | 76 | 46 | A1 | 63 * | CE | 81 | F9 | 98 |
| 1F | 12 * | 4A | 29 * | 77 | 47 | A2 | 64 | CF | 81 * | FA | 98 * |
| | | 4B | 29 | | | A3 | 64 * | | | FB | 98 |
| 20 | 13 | 4C | 30 | 78 | 47 * | A4 | 64 | D0 | 82 | FC | 99 * |
| 21 | 13 * | 4D | 30 * | 79 | 47 | A5 | 65 | D1 | 82 * | FD | 99 |
| 22 | 13 | 4E | 31 | 7A | 48 * | A6 | 65 * | D2 | 82 | FE | 100 |
| 23 | 14 | 4F | 31 * | 7B | 48 | A7 | 65 | D3 | 83 | FF | 100 * |
| 24 | 14 * | | | 7C | 49 | | | D4 | 83 * | | |
| 25 | 15 | 50 | 31 | 7D | 49 * | A8 | 66 * | D5 | 84 | | |
| 26 | 15 * | 51 | 32 | 7E | 49 | A9 | 66 | D6 | 84 * | | |
| 27 | 15 | 52 | 32 * | 7F | 50 | AA | 67 | D7 | 84 | | |
| | | 53 | 33 | | | AB | 67 * | | | | |
| 28 | 16 | 54 | 33 * | 80 | 50 * | AC | 67 | D8 | 85 | | |
| 29 | 16 * | 55 | 33 | 81 | 51 | AD | 68 * | D9 | 85 * | | |
| 2A | 16 | 56 | 34 | 82 | 51 * | AE | 68 | DA | 85 | | |
| 2B | 17 * | 57 | 34 * | 83 | 51 | AF | 69 | DB | 86 * | | |

The table below shows the conversion from level percentages (0 to 100%) to DMX data values (0 to FFh). The table is based on the formula:

DMX_level = Round( Percentage * (255/100) )

Table C-2 -- Percentage Level to DMX Data Value Conversion

| | | | |
|---|---|---|---|
| 0 | 0 | 51 | 82 |
| 1 | 3 | 52 | 85 |
| 2 | 5 | 53 | 87 |
| 3 | 8 | 54 | 8A |
| 4 | A | 55 | 8C |
| 5 | D | 56 | 8F |
| 6 | F | 57 | 91 |
| 7 | 12 | 58 | 94 |
| 8 | 14 | 59 | 96 |
| 9 | 17 | 60 | 99 |
| 10 | 1A | 61 | 9C |
| 11 | 1C | 62 | 9E |
| 12 | 1F | 63 | A1 |
| 13 | 21 | 64 | A3 |
| 14 | 24 | 65 | A6 |
| 15 | 26 | 66 | A8 |
| 16 | 29 | 67 | AB |
| 17 | 2B | 68 | AD |
| 18 | 2E | 69 | B0 |
| 19 | 30 | 70 | B3 |
| 20 | 33 | 71 | B5 |
| 21 | 36 | 72 | B8 |
| 22 | 38 | 73 | BA |
| 23 | 3B | 74 | BD |
| 24 | 3D | 75 | BF |
| 25 | 40 | 76 | C2 |
| 26 | 42 | 77 | C4 |
| 27 | 45 | 78 | C7 |
| 28 | 47 | 79 | C9 |
| 29 | 4A | 80 | CC |
| 30 | 4D | 81 | CF |
| 31 | 4F | 82 | D1 |
| 32 | 52 | 83 | D4 |
| 33 | 54 | 84 | D6 |
| 34 | 57 | 85 | D9 |
| 35 | 59 | 86 | DB |
| 36 | 5C | 87 | DE |
| 37 | 5E | 88 | E0 |
| 38 | 61 | 89 | E3 |
| 39 | 63 | 90 | E6 |
| 40 | 66 | 91 | E8 |
| 41 | 69 | 92 | EB |
| 42 | 6B | 93 | ED |
| 43 | 6E | 94 | F0 |
| 44 | 70 | 95 | F2 |
| 45 | 73 | 96 | F5 |
| 46 | 75 | 97 | F7 |
| 47 | 78 | 98 | FA |
| 48 | 7A | 99 | FC |
| 49 | 7D | 100 | FF |
| 50 | 80 | | |

# D REVISION HISTORY

The following table gives a brief historical account of previous and current revisions of the ASCII Text Representation for Lighting Console Data USITT specification.

| Version Ident | Date | Specification Writers | Comments |
|---|---|---|---|
| 1.0 | 10 May 89 | B. Rodriguez | Initial proposal |
| 2.0 | 02 Sep 89 | B. Rodriguez & A. Ekvall | Based on 28 committee proposals |
| 2.1 3:0 | 30 Oct 91 | R. Weber | Based on 68 committee proposals |
| 2.5 3:0 | 03 Jan 92 | R. Weber | Incorporate review comments from LDI '91 * voting draft * |
| 3.0 3:0 | 29 Mar 92 | R. Weber | Approved Standard (first implementations beginning) |