
Table of Contents

Chapter 7: Terminal Definitions	7-2
Introduction	7-2
Purpose	7-2
Building a new Terminal Definition	7-2
TDF Records in H3000UTL 'SAVE' Format	7-3
TDF Records in Local Services Format	7-3
Terminal Options and Control Functions	7-5
Overview	7-5
H3000UTL Terminal Options and Control Functions: Examples and Field Definitions	7-5
H3000UTL Example:	7-6
Local Services Example: Screen 1	7-7
Local Services Example: Screen 2	7-7
Field Definitions	7-8
Terminal Option Fields	7-8
Terminal Control Fields	7-10
Status Line Fields	7-12
Control Sequence Action codes	7-13
Action Codes in Action: some examples	7-14
Coding for Extended Attributes	7-15
Standard Attribute Bit Values Used to Select Video Index Byte from the IBM4C Table	7-16
Bit Definitions for the Video Index Byte	7-17
Example from the HYNM TDF:	7-17
Example with HNETIX TDF, based on its screen version	7-18
Screen 1	7-18
Screen 2	7-18
Terminal Input Sequences	7-19
Overview	7-19
H3000UTL Terminal Input Function Names, Types, and Tokens	7-20
H3000 Example:	7-22
Local Services Terminal Input Sequence Tables	7-23
Screen 3	7-23
Screen 4	7-23
Explanation of Terminal Input Sequence Tables	7-24
Terminal Function Token Tables	7-27
Terminal Data Characters; Token Class 0	7-27
Local Editing Functions; Token Class 1	7-27
Terminal Control Functions; Token Class 2	7-28
Attention Indicator Functions; Token Class 3	7-28
Special <i>HYDRA 3000</i> Function Keys	7-29
ANSI/DEC VT100 Keyboard Mapping	7-30

Chapter 7: Terminal Definitions

Introduction

Purpose

This chapter provides detailed information concerning the structure and function of *HYDRA 3000* Terminal Definitions (**TDFs**). It is a supplement to the more general discussion of *HYDRA 3000* configuration presented in Chapters 4 and 5, and to the discussion of the mainframe configuration utility, **H3000UTL**, given in Chapter 6.

A **Terminal Definition** record, a part of *HYDRA 3000*'s File System, is used by the firmware to control the display of and interpret input from a particular model of terminal or corresponding PC emulation. **TDF** records are accessible through **Local Services** (via the menus or the **TDF** keyword; see Chapter 5). They are also a part of the mainframe disk file created by the **SAVE** command of the mainframe utility, **H3000UTL** (**TDF** record type; see Chapter 6).

The recommended method for creating or modifying Terminal Definitions is through the mainframe utility, **H3000UTL**. This program provides an easy means for translating ASCII control and key sequences into the somewhat cryptic internal storage format of the **TDF** as displayed under Local Services. There are situations, however, where Local Services provides a quicker method, or perhaps the only method at the moment, for creating or making changes to a **TDF**. This chapter should provide you with the information needed to build or modify a Terminal Definition by either the **H3000UTL** or the Local Services method.

Building a new Terminal Definition

The process of building a new **Terminal Definition** involves first gathering together documents relating to the terminal or terminal emulation in question. These documents should contain all necessary information regarding terminal characteristics, display control sequences, and keyboard sequences used to emulate 3270 keyboard functions. If information about keyboard sequences is not documented, it is possible to determine this using *HYDRA 3000*'s internal trace facility or an external monitoring device. You may discover during your research that the terminal or emulation you are considering has very limited special function keys, in which case you may need to think of creative ways to extend the standard to more fully emulate a 3270 terminal. In addition to terminal documentation, you should also have a chart of ASCII character codes at hand.

The next task is to find a good starting point, that is, an existing **Terminal Definition** to use as a template. Despite the great variety of ASCII terminals and emulations, the specifications for your terminal or emulation will probably closely match one of Hydra Systems's existing **TDFs**. Armed with the reference materials mentioned above, look at the display control sequences used to address the cursor, clear the screen, and set video attributes. In particular, examine the lead-in character or characters used in these control sequences.

You may also want to look at the character sequences generated by function keys such as the arrow keys, Home, Delete, Insert, F-keys, etc. Then examine existing *HYDRA 3000* TDFs for similarities, and select the TDF that most closely matches the characteristics of your terminal; use the H3000UTL versions of the TDFs in making your comparisons. You may need to refer to information below in order to identify and understand the various **Terminal Definition** fields.

The next step is to make a copy of the TDF you wish to use as a template. If you have chosen H3000UTL to build the new TDF, use your mainframe editor to duplicate the template TDF, either within the saved configuration or as a separate file, then rename the TDF as required. If you are building the new TDF under Local Services, first display the TDF to be used as a template, then over-type its record name and save the new record using PF6.

At this point you can begin to modify the existing field information according to the documentation for your terminal. The remainder of this chapter explains the structures, functions, and fields involved. Contact Hydra Systems Support if you need help in this process.

TDF Records in H3000UTL 'SAVE' Format

In the format produced by the H3000UTL 'SAVE' command, each TDF record consists of a series of lines of text. Each line contains information about one field. The first line gives the record name and type; succeeding lines assign values to TDF field variables. All lines except for the last are terminated by commas; the last ends with a semicolon. As with any record produced by the H3000UTL SAVE function, a TDF record can be modified using any standard mainframe editor, and can be reloaded via the H3000UTL ADD command.

Some TDF fields contain simple YES/NO values. Others contain single numeric values in hexadecimal. Still others contain text or sequences of hexadecimal ASCII character codes, or a combination of both. Finally, several fields contains references to external records. Field names are listed below, together with a brief explanation.

From the information contained in the TDF, the H3000UTL RESTORE or ADD function builds a compact memory image and sends it down the channel to the *HYDRA 3000* for storage in the file system. The memory image is based on field information contained in the Control File, which resides in *HYDRA 3000* memory, in the section reserved for microcode storage. We will refer to this Control File in following sections. Refer to Chapter 6 for further discussion of the SAVE file format and use of the utility.

TDF Records in Local Services Format

In its interactive manifestation (i.e., via Local Services), each TDF record consists of six screens, which are described briefly in the remaining paragraphs of the Introduction, and in detail in the sections that follow.

The first screen identifies the TDF and specifies general terminal attributes and options, such as default and alternate screen sizes, video attribute bytes (both

standard and extended), custom translation tables, cursor addressing offset, etc. This screen also contains pointers into the Terminal Control Sequence table (second TDF screen) for functions such as cursor addressing, screen or line clear, video attribute, default/alternate screen selection, terminal initialization and de-initialization, auxiliary printer activation and deactivation, and status reporting.

The second screen is the 256-byte Terminal Control Sequence table pointed to by various fields in the first screen. The sequences of this table are coded in hexadecimal ASCII, with special hex codes, in the range x'80' - x'FF', which specify an action to be performed (for example, the insertion of items such as standard lead-in sequences, the required row and column in the cursor addressing sequence, or the video attribute in the video attribute sequence). Each sequence is terminated by a null byte (which precludes the transmission of a null character to a terminal as part of a sequence).

The final four screens contain paired 256-byte Terminal Input Sequence tables for the translation of inbound ASCII sequences used for the various 3270 and display functions. All inbound characters are processed via these tables and are either passed unchanged, translated to a 3270 or display function token, or discarded. The translation algorithm can handle a complex sequence of character codes representing a single function. Each screen displays a 64-byte segment of the paired tables consisting of two lines: the top line from the Segmented Search Table (SST) contains the ASCII codes to scan; the bottom line from the Function Interpretation Table (FIT) contains the codes, offsets, and function tokens used by *HYDRA 3000* to accomplish the translation. The translation made by the Terminal Input Sequence tables is completely distinct from the ASCII-to-EBCDIC translation accomplished by an ATE table (described in Chapter 5).

As with all other *HYDRA 3000* configuration screens, successful manipulation of the TDF record via Local Services requires the use of certain PF/PA keys, cursor movement keys, and the SysRequest key; thus you must use a terminal already well-defined to the *HYDRA 3000*. A very common emulation is ANSI (closely related to the DEC VT100 terminal). A keyboard mapping for ANSI is included at the end of the chapter.

Refer to Chapter 5, "Configuration/Local Services Reference" and Chapter 4, "*HYDRA 3000* Conventions and Configuration", for general instructions on the use of Local Services functions.

The following sections of this chapter discuss the specific fields which comprise a Terminal Definition.

Terminal Options and Control Functions

Overview

We will start with a general discussion of what you should expect to find in the **Terminal Options and Control Functions** section of a TDF. This is the first section of the H3000UTL 'SAVE' version of a TDF, and the first two screens of the Local Services version, in which are defined general terminal characteristics, terminal control sequences, and status line output. After the general discussion, we present an example of a common TDF, in both H3000UTL and Local Services versions, followed by a detailed discussion of each field. Terminal keyboard input mapping will be discussed separately in subsequent sections, under the heading **Terminal Input Sequences**.

For Terminal Options and Control functions, the **H3000UTL** version directly assigns a single value or a string of characters and/or values to the field name. Single values can be Boolean (YES/NO) or one-byte hexadecimal data. Documentation fields (NAME and REVISION) and fields containing references to external records (XLT and PDF) are fixed-length fields of text surrounded by single quotes. Control sequences and associated fields take the form of one or more two-digit, hexadecimal byte values, or upper-case ASCII text enclosed in slashes, or as a combination of hexadecimal and ASCII. The ASCII Space character, lower-case characters, and any characters outside the normal ASCII character code range must always be represented as hexadecimal values.

In the **Local Services** version, fields related to terminal characteristics are presented in the first screen and are handled in much the same way as in the H3000UTL version; that is, they contain Boolean or hexadecimal values. Fixed length fields are allotted the correct amount of space on the screen; single quotes should not appear in these fields. In fields with fixed selections, such as Boolean values or external record references, you may either type the desired selection or toggle it with PF1 or PF2. Fields related to terminal control sequences are also presented in the first screen, but contain hexadecimal pointers; the actual control sequences are contained in a 256-byte table presented as a second screen, and the pointers specify the offsets of the beginning of the sequences. You can display the second screen by hitting PF10. Note that the first digit of the hexadecimal pointer gives the row, the second digit the column of the beginning of a particular sequence. A value of FF indicates that the field is not used. Each sequence in Screen 2 is terminated by a null byte (x'00').

H3000UTL Terminal Options and Control Functions: Examples and Field Definitions

The next several pages are devoted to presenting an example of an Extended Attribute TDF in both H3000UTL and Local Services formats, then defining the fields in detail.

H3000UTL Example:

```
HNETIX   TDF
         ATTR.TYPE=NO,
         BACKGROUND=YES,
         XMITNONDIS=NO,
         EXTATTR=YES,
         TN3270=NO,
         PAPER.TERM=NO,
         AUTO.SCROLL=YES,
         AUTO.WRAP=NO,
         AUTO.LINE=NO,
         AUTO.LF=NO,
         AUTO.CR=NO,
         ACP=YES,
         DATA.ENTRY.KBD=NO,
         NAME='HNET IBM EXT   ',
         REVISION=' 1.1 ',
         M2.NR=19,
         M3.NR=19,
         M4.NR=19,
         M5.NR=1B,
         M2.NC=50,
         M3.NC=50,
         M4.NC=50,
         M5.NC=84,
         NBS=08,
         NFS=00,
         NCU=00,
         NCD=00,
         ADD=20,
         ATVI=00,
         VST.0=0209040C0A0B0E0F,
         VST.1=8289848C82838E8F,
         VST.2=2717404727386770,
         VST.3=7271747C72737E3F,
         VST.4=0000000000000000,
         VST.5=0000000000000000,
         VST.6=0000000000000000,
         VST.7=0000000000000000,
         IBM4C=04040300040403000505070005050700,
         SIVI=00,
         XLT.REF='           ',
         PDF.REF='           ',
         M2.SS.S=1B041B451B03,
         M5.SS.S=1B051B451B03,
         M2.L2.S=1B5938,
         M5.L2.S=1B593B,
         CPF.S=1B598081,
         SGR.S=1B539B9785,
         CLRS.S=1B53021B451B031B48,
         CLSS.S=1B041B451B031B02,
         AXON.S=1B4C,
         AXOF.S=1B4D,
         KLCK.S=9227/L/6F636B20,
         KUNL.S=9227/R/65616479,
         ISET.S=9220/I/6E73657274,
         IRES.S=9220202020202020,
         SERP.S=92/8E/7270,
         RERP.S=92/8/202020,
         SUNO.S=92/-/202020202020,
         SSCP.S=92/-SSCP-/89,
         SPLU.S=92/-/20/PLU-/89,
         STST.S=92/-TEST/2020,
         SSND.S=92/4S/6E64,
         SRCV.S=92/4R/6376,
         SCNT.S=92/4C/6E74,
```

Local Services Example: Screen 1

HYDRA 3000 r2.5 Copyright (c) 1992-2000 Hydra Systems Inc All Rights Reserved

Port 1B Logical Terminal 00 -- Link 00 Host 05 Nau 1F

```
TDF KEY --> HNETIX NAME: HNET IBM EXT 1.1 XLT : PDF :

ATTR : NO SSO NR NC L0 L1 L2 L3 VST0: 0209040C 0A0B0E0F
CBGF : YES (24X80) : 26 19 50 FF FF 09 FF 8289848C 82838E8F
TNDD : NO (32X80) : FF 19 50 FF FF FF FF 27174047 27386770
ASCRL: YES (43X80) : FF 19 50 FF FF FF FF 7271747C 72737E3F
AWRAP: NO (27X132): 2D 1B 84 FF FF 0D FF 00000000 00000000
ALINE: NO 00000000 00000000
ACRNL: NO IBM4C: 04040300 04040300 05050700 05050700 00000000 00000000
ALFNL: NO PAPER: NO 00000000 00000000
ACP : YES
DKBRD: NO AXON: 34 CPF : 11 SIVI: 00 SSND: 8C SUNO: 68
EXTAT: YES AXOF: 37 ADD : 20 EEOL: FF KLCK: 3A SRCV: 92 SSCP: 71
3270 : NO SGR : 16 EEOS: FF KLNK: 42 SCNT: 98 SPLU: 7A
NBS : 08 CRS : FF ATVI: 00 CEOL: FF ISET: 4A STST: 83
NFS : 00 CLS : FF OPNS: FF CEOS: FF IRST: 53 RANS: FF SDRT: FF
NCU : 00 CUS : FF CLSS: 00 ERSS: FF SERP: 5C SASA: FF
NCD : 00 CDS : FF CLRS: 1C RERP: 62 SNSA: FF LPSS: FF

PF1 PF2 PF3 PF4 PF5 PF6 PF7 PF8 PF9 PF10 PF11 ENTER PA2
Bwd-F Fwd-F RECVR DEFLT DEL-R SAV-R BWD-R FWD-R Bwd-P Fwd-P View Re-Dsp Exit
```

Local Services Example: Screen 2

HYDRA 3000 r2.5 Copyright (c) 1992-2000 Hydra Systems Inc All Rights Reserved

Port 1B Logical Terminal 00 -- Link 00 Host 05 Nau 1F

```
TDF KEY --> HNETIX NAME: HNET IBM EXT 1.1 XLT : PDF :
00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F

00 1B 04 1B 45 1B 03 1B 02 00 1B 59 38 00 1B 59 3B 00
10 00 1B 59 80 81 00 1B 53 9B 97 85 00 1B 53 02 1B 10
20 45 1B 03 1B 48 00 1B 04 1B 45 1B 03 00 1B 05 1B 20
30 45 1B 03 00 1B 4C 00 1B 4D 00 92 27 4C 6F 63 6B 30
40 20 00 92 27 52 65 61 64 79 00 92 20 49 6E 73 65 40
50 72 74 00 92 20 20 20 20 20 20 00 92 38 45 72 50
60 70 00 92 38 20 20 20 00 92 2D 20 20 20 20 20 60
70 00 92 2D 53 53 43 50 2D 89 00 92 2D 20 50 4C 55 70
80 2D 89 00 92 2D 54 45 53 54 20 20 00 92 34 53 6E 80
90 64 00 92 34 52 63 76 00 92 34 43 6E 74 00 00 00 90
A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0
B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B0
C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C0
D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D0
E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E0
F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F0

00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F
```

Field Definitions

TDF KEY {TDF}

The **TDF KEY** field defines the name of the record.
Referred to in: **APD** record, **TDF** field; displayed at connection time in the terminal selection menu.

NAME {NAME}

NAME is a descriptive field. It is displayed next to the **TDF** record name when the **TDF** menu is presented at connection time, and in the **APD** record, **TDF** field, next to the selected **TDF**.

Terminal Option Fields

The following Terminal Option fields must be assigned a record name or a YES/NO or hexadecimal value, as appropriate.

XLT {XLT.REF}

EBCDIC-to-ASCII/ASCII-to-EBCDIC translation tables: This field contains the record name for the collection of eight **ETA** and **ATE** translation tables to be used with this terminal. The eight pairs of tables correspond to the eight alternate character sets of Extended Attributes. A blank record name indicates that the **XLT** record specified in the **APD** (Asynchronous Port Definition) record is to be used.

PDF {PDF.REF}

Printer Definition: This field contains a reference to the Printer Definition (PDF) record that will control multiplexed printing for this terminal.

ATTR {ATTR.TYPE}

A YES in this field indicates the terminal is an **attribute-type terminal**, which will accept and process standard attribute bytes preceding each field on the screen. A NO in this field means that special, terminal-dependent sequences must be sent to the terminal (via the Set Graphics Rendition (SGR) function) to accomplish field highlighting, etc.

CBGF {BACKGROUND}

Change in Back-Ground Forced: This field indicates that the ASCII space character must be explicitly rewritten whenever the field attribute governing that position of the screen changes, since background characteristics may have changed (e.g., with reverse video or variable background color).

TNDD {XMITNONDIS}

Transmit Non-Display Data: Set **TNDD** to YES to enable display of non-display fields (used almost exclusively for CICS IND\$FILE transfers). Set this option to NO for normal port usage.

ASCRL {AUTO.SCROLL}

Auto-Scroll: If the terminal scrolls on receipt of a character in the last screen position, set this field to YES, so the HYDRA will not write to the last screen position; otherwise set the field to NO.

AWRAP {AUTO.WRAP}

Auto-Wrap: If, after receipt of a character in the last screen column, the terminal wraps to the next line only if the following character is a printable character, set this field to YES; otherwise set the field to NO.

ALINE {AUTO.LINE}

Auto-Line: If, after receipt of a character in the last screen column, the terminal always repositions the cursor to column 1 of the next line, set this field to YES; otherwise set the field to NO.

ACRNL {AUTO.CR}

Auto-New-Line-After-Carriage-Return: Not currently used. Set to NO.

ALFNL {AUTO.LF}

Auto-New-Line-After-Line-Feed: Not currently used. Set to NO.

ACP {ACP}

Asynchronous Communication Protocol: Indicates whether or not the proprietary HydraNet Asynchronous Communications Protocol is active. This protocol must be used for a HydraNet server connection and should not be used otherwise.

DKBRD {DATA.ENTRY.KBD}

Data-Entry Keyboard: If the terminal supports a data-entry keyboard and you wish to use this feature, set this field to YES; otherwise set the field to NO. The data-entry keyboard will only be active in numeric fields.

EXTAT {EXTATTR}

Extended Attribute Support Active: If YES, indicates that Extended Attribute support is possible. In order for Extended Attributes to be fully active, the LU/CUU and port must also be set up to support this feature.

3270 {TN3270}

TelNet 3270 Terminal: If YES, indicates the terminal should be handled as a TelNet 3270 device.

NBS {NBS}

Non-destructive Backspace: This field holds the ASCII code for a non-destructive backspace, used to optimize single-position cursor moves.

NFS {NFS}

Non-destructive Forward Space: This field holds the ASCII code for a non-destructive forward space, used to optimize single-position cursor moves.

NCU {NCU}

Non-destructive Cursor Up: This field holds the ASCII code for a non-destructive cursor up, used to optimize single-position cursor moves.

NCD {NCD}

Non-destructive Cursor Down: This field holds the ASCII code for a non-destructive cursor down, used to optimize single-position cursor moves.

NR column {M2.NR, M3.NR, M4.NR, M5.NR}

Number of Rows: Sets the number of rows for the four standard screen sizes (in hex).

NC column {M2.NC, M3.NC, M4.NC, M5.NC}

Number of Columns: Sets the number of columns for the four standard screen sizes (in hex).

PAPER

Paper Terminal: If YES, specifies support for a paper-printing terminal, such as the TI Silent 700; provides compatibility with HydraII applications.

Terminal Control Fields

In the Local Services version, these fields point to sequences that control the terminal display, for example, sequences to select different screen sizes, to position the cursor, and to set video attributes.

SSO {M2.SS, M3.SS, M4.SS, M5.SS}

Screen Size Option: selects the terminal's screen size for Model 2-5 emulation.

L0 - L3 columns {M2.L0-M2.L3, M3.L0-M3.L3, etc.}

Lead-in ("leader") Sequences 0 – 3: These sixteen fields point to standard character code sequences, used to avoid repetition and thus save space in the sequence table. There are pointers for each screen size. They are referred to in other sequences as **90 – 93**. The correct sequence is chosen to match the current screen size, in accordance with the port setup, and whether the mainframe application has selected the default or alternate screen.

VST0 {VST.0 through VST.7}

Video Attribute Bytes: These 8 rows of 8 bytes each are the actual video attribute bytes to embed in the video attribute sequence sent to the terminal. These rows and bytes are pointed to either by the **IBM4C** byte corresponding to the appropriate non-extended attribute, or by the extended attribute in the data stream in accordance with the Action Codes contained in the **SGR** sequence (see the Action Codes section below).

IBM4C {IBM4C}

Standard Attribute Table: These 16 bytes correspond in order to the 16 standard 3270 attributes and are used in conjunction with the sequence pointed to by the **SGR** field to set the terminal's video mode (e.g., intensity, color, underline, etc.).

AXON/AXOFF {AXON/AXOFF}

Auxiliary Port on/off: These two fields point to sequences that activate/deactivate the terminal's auxiliary port for multiplexed printing.

LPSS {LPSS.S}

Local/Shared Screen Print Sequence: This field points to the sequence that causes the terminal to dump the screen to the attached printer.

CPF {CPF.S}

Cursor Positioning Function: This field points to the full cursor addressing sequence. Special codes are used to indicate where and in what format to insert the actual row and column. The **ADD** field gives the offset used in conjunction with this cursor-addressing scheme.

ADD {ADD}

Cursor addressing offset: This field contains the hexadecimal position offset used in the terminal's cursor positioning function. The default home position is row 0/column 0; the offset is used for both row and column positioning. Common offsets are x'01' (DEC VT100/200 series) and x'20' (first printable ASCII character code; IBM31xx, DEC VT52)

SGR {SGR.S}

Set Graphics Rendition: This field specifies the sequence used to set the terminal screen's video mode (e.g., normal/reverse video, high/low intensity, underline, color, etc.) for a given attribute sent from the host. A set of special action codes in the **SGF** sequence indicate which byte from the **VST0** table should be inserted at that point. (See below for a list of those action codes and for an explanation of internal workings of the **SGR** function.)

CRS/CLS/CUS/CDS {CRS.S/CLS.S/CUS.S/CDS.S}

Cursor Right/Left/Up/Down Sequence: These fields specify the sequences used to move the cursor the required number of positions in the required (single) direction from the current position. Note that an action code in each sequence indicates where and in what format to insert the number of positions to move. (See below for a list of those action codes.)

EEOL/EEOS {EEOL.S/EEOS.S}

Erase End-of-Line/Screen: These fields specify sequences used to erase the screen from the current cursor position to the end of the current line or to the end of the screen.

CEOL/CEOS {CEOL.S/CEOS.S}

Clear End-of-Line/Screen: These fields specify sequences used to erase the screen from the cursor position to the end of the line/screen. In addition, *HYDRA 3000* clears the corresponding locations in the image buffer.

ERSS {ERSS.S}

Erase Screen: This field specifies the sequence used to erase the screen.

CLRS {CLRS.S}

Clear Screen: The sequence specified by this field clears the terminal's screen. In addition, the *HYDRA 3000* clears the image buffer associated with this terminal.

OPNS/CLSS {OPNS.S/CLSS.S}

Opening/Closing Sequence: These fields specify special (optional) sequences that may be used to initialize/restore terminal features at connect/disconnect time.

Status Line Fields

The remaining fields define various optional terminal status messages. They should be implemented if the terminal (or PC emulation) supports an additional line for status reports. In the H3000UTL SAVE version, these lines directly assign ASCII terminal control sequences and messages to each message variable. In the Local Services version, the fields contain hexadecimal pointers to sequences in the Terminal Control Sequence table.

KLCK/KUNL {KLCK.S/KUNL.S}

Keyboard Lock/Unlock: These fields point to the sequences that will be sent to the terminal's status line when the keyboard lock bit is set by an inbound attention key or cleared by the application.

ISET/IRST {ISET.S/IRST.S}

Set/Reset Insert Mode: These fields specify the sequences to indicate terminal's insert-mode status.

SERP/RERP {SERP.S/RERP.S}

Set/Reset ERP (Error)

SUNO {SUNO.S}

Unowned state: This field points to a message that will be displayed whenever the terminal is in Unowned state.

SSCP {SSCP.S}

SSCP state: This field points to a message that will be displayed whenever the terminal is in SSCP state.

SPLU {SPLU.S}

PLU state: This field points to a message that will be displayed whenever the terminal is in PLU state.

STST {STST.S}

TEST state: This field points to a message that will be displayed whenever the terminal is in Test state.

SSND {SSND.S}

SEND state (in bracket) : This field points to a message that will be displayed whenever the terminal is in Send state.

SRCV {SRCV.S}

Receive state (in bracket) : This field points to a message that will be displayed whenever the terminal is in Receive state.

SCNT {SCNT.S}

Contention state (not in bracket) : This field points to a message that will be displayed whenever the terminal is in Contention state.

Control Sequence Action codes

Terminal Control Sequences, in addition to containing strings of ASCII character codes, may also contain special Action Codes. These internal codes have the high bit set, i.e., they are greater than or equal to x'80', and are therefore outside the range of standard ASCII codes. Action Codes are placed in a Terminal Control Sequence at a point where data needs to be inserted, for example, a variable byte, such as a field attribute or cursor row or column, or a standard, fixed, multi-character sequence. Action Codes are processed and acted upon as the Terminal Control Sequence is processed for transmission to the terminal display. Some Action Codes specify the type of value to insert, such as a binary row or column number; others specify manipulations of data associated with the control sequences, for example, how to extract the appropriate attribute byte; still others indicate insertion of a standard lead-in sequence, to avoid unnecessary repetitions in the Terminal Control Sequence table.

Action Codes and their meanings are given below. Codes associated with Extended Attributes are marked by ** and will be discussed separately.

80 -- Insert the row as a binary number.

81 -- Insert the column as a binary number.

82 -- Insert the row number as one to two ASCII decimal digits.

83 -- Insert the column number as one to three ASCII decimal digits.

85 -- Insert Extracted Value **

86 -- Insert a constant in binary format (e.g., video attribute byte for the Set Graphic Rendition function).

87 -- Insert a constant as one to three ASCII digits (e.g., the number of positions to move the cursor).

88 -- Insert a two-byte constant.

89 -- Insert the current Session Number.

8E -- Extract Immediate Value **

8F -- Select video attribute table VST.x (constant 0-7) **

90 - 93 -- Insert a "leader" sequence, pointed to by L0 - L3 from Screen 1. Note that there are separate "leader" fields for each screen size.

NOTE: Leader data is sent as is, with no command interpretation.

- 94** - Set to extract by “OR” (default) **
- 95** - Set to extract by “XOR” **
- 96** - Set to extract by “AND” **
- 97** - Extract VST byte based on “COLOR” **
- 98** - Extract VST byte based on “TYPE” **
- 99** - Extract VST byte based on “CHAR.SET” **
- 9A** - Select VST segment based on “COLOR” **
- 9B** - Select VST segment based on “TYPE” **
- 9C** - Select VST segment based on “CHAR.SET” **
- 9D** - Select VST.0 segment **
- 9E** - Select VST.4 segment **
- 9F** - Select VST.7 segment **

Action Codes in Action: some examples

In the **HNETIX TDF** example above, given in both its H3000UTL and Local Services forms, the Cursor Positioning Function sequence (**CPF.S** or **CPF**) is given as:

1B598081

The lead-in bytes, **1B59**, identify this to the HydraNet emulator as cursor positioning. The last two bytes, **8081**, indicate the the binary row and column values should be inserted at this point, after adding to each the offset contained in the **ADD** field, in this case hex **20**. The sequence, **1B592020**, puts the cursor at the home position (row 1, column 1), for example.

A second example from HNETIX is the status line message indicating PLU state for a session (**SPLU.S** or **SPLU**): **922D20504C552D89**. This sequence calls for insertion of leader sequence 2: **1B5938**, which positions the cursor in row 25 (x'38' minus x'20' plus 1); the cursor column is given in the next byte, **2D** (column 14). The next 5 bytes, **20504C552D**, are ASCII codes for “ PLU-”, and the last byte, **89**, indicates insertion of the session number.

Coding for Extended Attributes

In order to make the ASCII display screen look as much as possible like a true IBM 3270 display, *HYDRA 3000* sends a video attribute command string, based on the 3270 attribute, to the ASCII display device prior to sending the actual character data. The Set Graphic Rendition (**SGR**) function draws on information in several fields of the selected **Terminal Definition** to compose this command string.

For both standard and Extended Attributes, the **SGR** function uses the **SGR** field of a **TDF** as the skeleton, i.e., the basic components, of the final string sent to the terminal. The **SGR** skeleton must contain ASCII codes that inform the display device that this is a video attribute command; a termination character may be part of this. The **SGR** further contains internal Action Codes that determine how to select the appropriate video attribute byte, and where to insert the byte in the final string.

Other fields associated with the **SGR** function are the **VST.0** through **VST.7** fields, from which the final attribute byte is selected, the **IBM4C** field, used to select the attribute byte in the case of standard (non-Extended) attributes, and the **EXTAT** field, which enables Extended Attribute processing for the **TDF**.

But the **EXTAT** field of a **TDF** does not by itself dictate use of Extended Attributes. Extended Attribute support must also be enabled in the port's Logical Device (**LDV**) setup and

The video attribute command string composed by the **SGR** function is based on information from three sources:

- 1) the Terminal Definition (**TDF**) selected for this display device;
- 2) the assigned Logical Device (**LDV**) record; and
- 3) the 3270 attribute specified by the mainframe application for a given field or character, from which we extract a Video Index Byte (**VIB**).

The exact nature of the Video Index Byte depends on several factors. We will construct the VIB based on the extended attribute byte if:

- 1) **EXTAT** is set to **YES** in the selected **TDF**, and
- 2) the active **LDV** record has **EXTENDED ATTRIBUTES** set to **FORCED** or **DYNAMIC**, and
- 3) there is a non-zero extended attribute byte from the application.

If any one of the above conditions is not met, the standard field attribute will act as an index to a byte of the **IBM4C** field, and that byte will be the VIB.

In either case, the **SGR** function then uses the VIB to extract a byte from the **TDF**'s VST table, and sends that to the display device, according to the **SGR** string encoded in the selected **TDF**.

Outline

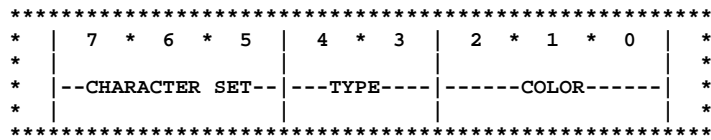
- I. VIB (Video Index Byte) origin
 - A. Standard 3270 (VIB from field att): Application uses S3270; or TDF not E3270-capable; or port not E3270-capable; or E3270 att byte is zero.
 - B. Extended 3270 (VIB from ext att): App uses E3270 and TDF is E3270-capable and port is E3270-capable and E3270 att byte is non-zero
- II. VIB use in SGR
 - A. S3270 field att: Use VIB as index into IBM4C; use that to determine row and column of VST table and extract byte
 - B. E3270 att: Use VIB directly to determine row/column of VST table and extract byte
- III. VST byte selection (see table below for details)
 - A. Row
 - 1. Color
 - 2. Type
 - 3. Char. set
 - B. Column
 - 1. Color
 - 2. Type
 - 3. Char. Set

Standard Attribute Bit Values Used to Select Video Index Byte from the IBM4C Table

EQU X'08'	PROTECTED FIELD INDICATED IF SET
EQU X'04'	NUMERIC FIELD INDICATED IF SET
EQU X'02'	HIGH INTENSITY FIELD INDICATED IF SET
EQU X'01'	LIGHT PEN DETECTABLE FIELD INDICATED IF SET

NOTE: X'03' = NON-DISPLAY FIELD INDICATED

Bit Definitions for the Video Index Byte



TYPE (Middle 2 bits)

- * EQU X'00' USE NON-EXTENDED ATTRIBUTE ABILITY
- * EQU X'08' BLINK (X'F1' DATA STREAM VALUE)
- * EQU X'10' REVERSE VIDEO (X'F2' DATA STREAM)
- * EQU X'18' UNDERSCORE (X'F4' DATA STREAM)

COLOR (Right 3 bits)

- * EQU X'00' X'F0' DISPLAY-BASE COLOR 3287-BLACK
- * EQU X'01' X'F1' BLUE
- * EQU X'02' X'F2' RED
- * EQU X'03' X'F3' PINK
- * EQU X'04' X'F4' GREEN
- * EQU X'05' X'F5' TURQUOISE
- * EQU X'06' X'F6' YELLOW
- * EQU X'07' X'F7' WHITE 3287 BLACK

Example from the HYNM TDF:

```

VST.0=02020C0202020C02
VST.1=09090F0909090F09
VST.2=0000000000000000
VST.3=0000000000000000
VST.4=0000000000000000
VST.5=0000000000000000
VST.6=0000000000000000
VST.7=0000000000000000
    
```

```

IBM4C=000102030405060708090A0B0C0D0E0F
    
```

```

SGR.S=1B539B9785
    
```

In this example, **1B53** is the lead-in sequence required by HydraNet/PC. The next byte, **9B**, selects the **VST.x** table based on the **TYPE** bits (4 and 3) of the Video Index Byte, which is taken either from the appropriate byte of **IBM4C** (for standard 3270 attributes) or the extended attribute byte from the secondary (extended) image buffer. The **TYPE** bits in **IBM4C** in this example are b'00 for the first 8 bytes and b'01 for the last eight bytes. B'00 selects VST.0; b'01 selects VST.1. Next, **97** extracts the byte of VST.x indicated by the **COLOR** bits (2, 1, and 0) of the Video Index Byte. In this example, for standard attributes, the **COLOR** bits of the bytes of **IBM4C** point in succession to the bytes of the selected VST.x line. Finally, **85** is an order to transmit the assembled string.

Example with HNETIX TDF, based on its screen version

Screen 1

HYDRA 3000 r2.5 Copyright (c) 1992-2000 Hydra Systems Inc All Rights Reserved
 Port 1B Logical Terminal 00 -- Link 00 Host 05 Nau 1F

```
TDF KEY --> HNETIX NAME: HNET IBM EXT 1.1 XLT : PDF :

ATTR : NO SSO NR NC L0 L1 L2 L3 VST0: 0209040C 0A030E0F
CBGF : YES (24X80) : 23 19 50 FF FF 09 FF 8289848C 82838E8F
TNDD : NO (32X80) : FF 19 50 FF FF FF FF 27174047 27386770
ASCRL: YES (43X80) : FF 19 50 FF FF FF FF 7271747C 72737E3F
AWRAP: NO (27X132): 2A 1B 84 FF FF 0D FF 00000000 00000000
ALINE: NO 00000000 00000000
ACRNL: NO IBM4C: 04040200 04040200 01010700 01010700 00000000 00000000
ALFNL: NO 00000000 00000000
ACP : YES
DKBRD: NO AXON: 31 CPF : 11 SIVI: 00 SSND: 89 SUNO: 65
EXTAT: YES AXOF: 34 ADD : 20 EEOL: FF KLCK: 37 SRCV: 8F SSCP: 6E
SGR : 16 EEOS: FF KLNK: 3F SCNT: 95 SPLU: 77
NBS : 08 CRS : FF ATVI: 00 CEOL: FF ISET: 47 STST: 80
NFS : 00 CLS : FF OPNS: FF CEOS: FF IRST: 50 RANS: FF SDRT: FF
NCU : 00 CUS : FF CLSS: 00 ERSS: FF SERP: 59 SASA: FF
NCD : 00 CDS : FF CLRS: 1C RERP: 5F SNSA: FF

PF1 PF2 PF3 PF4 PF5 PF6 PF7 PF8 PF9 PF10 ENTER PA2
Bwd-F Fwd-F RECVR DEFLT DEL-R SAV-R BWD-R FWD-R Bwd-P Fwd-P Re-Dsp Exit
```

Screen 2

HYDRA 3000 r2.5 Copyright (c) 1992-2000 Hydra Systems Inc All Rights Reserved
 Port 1B Logical Terminal 00 -- Link 00 Host 05 Nau 1F

```
TDF KEY --> HNETIX NAME: HNET IBM EXT 1.1 XLT : PDF :
00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F

00 1B 04 1B 45 1B 03 1B 02 00 1B 59 38 00 1B 59 3B 00
10 00 1B 59 80 81 00 1B 53 9B 97 85 00 1B 45 1B 03 10
20 1B 48 00 1B 04 1B 45 1B 03 00 1B 05 1B 45 1B 03 20
30 00 1B 4C 00 1B 4D 00 92 27 4C 6F 63 6B 20 00 92 30
40 27 52 65 61 64 79 00 92 20 49 6E 73 65 72 74 00 40
50 92 20 20 20 20 20 20 20 20 00 92 38 45 72 70 00 92 50
60 38 20 20 20 00 92 2D 20 20 20 20 20 20 00 92 2D 60
70 53 53 43 50 2D 89 00 92 2D 20 50 4C 55 2D 89 00 70
80 92 2D 54 45 53 54 20 20 00 92 34 53 6E 64 00 92 80
90 34 52 63 76 00 92 34 43 6E 74 00 00 00 00 00 00 90
A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0
B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B0
C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C0
D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D0
E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E0
F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F0

00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F
```

Terminal Input Sequences

Overview

The last section of the H3000UTL version of a Terminal Definition deals with keyboard mapping and the interpretation of input sequences generated by the terminal keyboard or analogous input device (e.g., barcode or magnetic card reader, or automated laboratory equipment). In the Local Services version of a **TDF**, the last four screens perform this function (screens 3-6).

In the H3000UTL version, this section of the **TDF** begins with “**KEY.SEQ=(**” and ends with “**);**”. In between are a series of assignments of keyboard and 3270 functions to input sequences of the form “**(function,sequence),**”, where the functions include PF-keys, PA-keys, Enter, Clear, Attention, SysRequest, EraseInput, EraseEOF, arrow keys, Home, NewLine, Tab, BackTab, Insert, Delete, etc., plus special *HYDRA 3000* functions for switching sessions and entering Test Mode (Local Services access). A complete list of functions follows this section.

In the Local Services version, the presentation closely matches the way the sequences are actually stored in *HYDRA 3000* memory, as a pair of tables. The Terminal Input Sequence Tables consist of a 256-byte Segmented Search Table (**SST**) with segments separated by null bytes, coupled, byte for byte, with a second 256-byte Function Interpretation Table (**FIT**), which gives an interpretation of each byte of the **SST**. These two tables are presented to the user in four screens of four paired rows per screen. Each row contains 16 bytes. The top line of each pair is searched for a match with the inbound (keyed) character code. If a match is found, the code below the matching byte contains bit-mapped information on how to interpret this inbound character, and what to do with the next one. While it is possible to manipulate these tables via Local Services, anything but the simplest change may involve massive rearrangement of the tables, which is naturally quite error-prone. Hydra Systems strongly recommends using the H3000UTL version to make changes to keyboard mapping. But for those who do not have a choice, we offer a discussion of the Local Services method below under the heading “Explanation of Terminal Input Sequence Tables”.

H3000UTL Terminal Input Function Names, Types, and Tokens

The following list contains the names, types, and token values for all supported keyboard input functions, from the Terminal Definition Keyboard Sequence section of the H3000UTL Control File.

```
KEY.SQ=(PKTH,512((PF1,01F1),
                 (PF2,01F2),
                 (PF3,01F3),
                 (PF4,01F4),
                 (PF5,01F5),
                 (PF6,01F6),
                 (PF7,01F7),
                 (PF8,01F8),
                 (PF9,01F9),
                 (PF10,017A),
                 (PF11,017B),
                 (PF12,017C),
                 (PF13,01C1),
                 (PF14,01C2),
                 (PF15,01C3),
                 (PF16,01C4),
                 (PF17,01C5),
                 (PF18,01C6),
                 (PF19,01C7),
                 (PF20,01C8),
                 (PF21,01C9),
                 (PF22,014A),
                 (PF23,014B),
                 (PF24,014C),
/ EXTENDED PF KEYS (PF25-29)
                 (PF25,01D1),
                 (PF26,01D2),
                 (PF27,01D3),
                 (PF28,01D4),
                 (PF29,01D5),
                 (PF30,01D6),
                 (PF31,01D7),
                 (PF32,01D8),
                 (PF33,01D9),
                 (PF34,015A),
                 (PF35,015B),
                 (PF36,015C),
                 (PF37,015D),
                 (PF38,015E),
                 (PF39,015F),
                 (PF40,01E2),
                 (PF41,01E3),
                 (PF42,01E4),
                 (PF43,01E5),
                 (PF44,01E6),
                 (PF45,01E7),
                 (PF46,01E8),
                 (PF47,01E9),
                 (PF48,016A),
                 (PF49,016F),
```

```

/ OTHER 3270 AID AND KEYBOARD FUNCTIONS
    (PAL,016C)
    (PA2,016E),
    (PA3,016B),
    (CLEAR,016D),
    (ENTER,017D),
    (SYSRQ,01F0),
(NSNA.TSTRQ,01E0), / NON-SNA TEST REQUEST
    (INSERT,020E),
    (INSCHAR,0209), / INSERT SINGLE BLANK
    (DELCHAR,020A),
(LOCAL.PRINT,020B),
    (ERASE.IPT,020C),
    (ERASE.EOF,020D),

/ CURSOR MOVEMENT
    (HOME,0208),
    (TAB,0205),
    (BTAB,0206),
    (NL,0207),
    (UP,0203),
    (DOWN,0204),
    (LEFT,0202),
    (RIGHT,0201),

    (FM,0025), / FIELD MARK
    (DUP,0210),
(CURSOR.SEL,0211),
    (LP.REPORT,0212), / LIGHT PEN POSITION REPORT
    (CP.REPORT,0213), / CURSOR POSITION REPORT

(RESTORE.KBD,0300),
    (CLEAR.KBD,0301),
    (DISCON.KBD,0302),
    (ATTENTION,0303), / GENERATE INBOUND SIGNAL

    (TSTREQ,0100), / HYDRA 3000 TEST MODE
    (SLT0,0108), / SELECT SESSION 0
    (SLT1,0109), / SELECT SESSION 1
    (SLT2,010A), / SELECT SESSION 2
    (SLT3,010B), / SELECT SESSION 3
    (SLTN,0110), / SELECT NEXT SESSION
    (SLTP,0111), / SELECT PRECEDING SESSION
    (SLTL,0112), / SELECT PRIOR SESSION

    (DRST,0113), / DATA ROUTING SESSION TOGGLE
    (DRSB,0114), / DATA ROUTING SESSION BINARY
    (DRSD,0115), / DATA ROUTING SESSION DISCON
/ SPECIAL FUNCTIONS FOR IND$FILE TRANSFERS WHERE THE MAINFRAME
/ APPLICATION SPECIFIES THE ENTIRE SCREEN AS NON-DISPLAY
    (SET.TNDD,0304), / DISPLAY NON-DISPLAY FIELDS
    (RESET.TNDD,0305), / NORMAL NON-DISPLAY
/ NUMERIC KEYBOARD FUNCTIONS
    (ALPHA.SI,0218),
    (ALPHA.SO,0219),
    (NUMERIC.SI,021A),
    (NUMERIC.SO,021B),

    (REFRESH,0101),
    (IGNORE,021F))

```

PF1-PF24	INSERT	NL	UP	SLT0-3
PAL-PA3	INSCHAR	ERASE.IPT	DOWN	SLTN
ENTER	DELCHAR	ERASE.EOF	RIGHT	SLTP
CLEAR	HOME	DUP	LEFT	SLTL
SYSRQ	TAB	CURSOR.SEL		DRST
ATN	BTAB	LIGHT.PEN		DRSB
REFRESH	RESTORE.KBD			DRSD
TESTREQ	IGNORE			

H3000 Example:

```
KEY.SQ=( (PF1,1B61),
          (PF2,1B62),
          (PF3,1B63),
          (PF4,1B64),
          (PF5,1B65),
          (PF6,1B66),
          (PF7,1B67),
          (PF8,1B68),
          (PF9,1B69),
          (PF10,1B6A),
          (PF11,1B21),
          (PF11,1B6B),
          (PF12,1B22),
          (PF12,1B6C),
          (PF13,1B31),
          (PF14,1B32),
          (PF15,1B33),
          (PF16,1B34),
          (PF17,1B35),
          (PF18,1B36),
          (PF19,1B37),
          (PF20,1B38),
          (PF21,1B39),
          (PF22,1B3A),
          (PF23,1B23),
          (PF23,1B3B),
          (PF24,1B24),
          (PF24,1B3C),
          (SLT0,1BA5),
          (SLT1,1BA6),
          (SLT2,1BA7),
          (SLT3,1BA8),
          (SLTN,1BA9),
          (SLTP,1BAA),
          (SLTL,1BAB),
          (DRST,1B7C),
          (DRSB,1B7D),
          (DRSD,1B7E),
          (PA1,1B1B),
          (PA2,1B56),
          (PA3,1B57),
          (CLEAR,1B4A),
          (CLEAR,1BAD),
          (SYSRQ,1B45),
          (INSERT,1B50),
          (DELCHAR,1B7F),
          (ERASE.IPT,1B05),
          (ERASE.EOF,1B49),
          (HOME,1B5A),
          (BTAB,1B47),
          (NL,1B51),
          (UP,1B41),
          (DOWN,1B42),
          (LEFT,1B44),
          (RIGHT,1B43));
```

Local Services Terminal Input Sequence TablesScreen 3

HYDRA 3000 r2.5 Copyright (c) 1992-99 Hydra Systems Inc All Rights Reserved
 Port 1B Logical Terminal 00 -- Link 00 Host 05 Nau 1F

TDF KEY --> HNETIX NAME: HNET IBM EXT 1.1 XLT : PDF :

00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F

00 10 1B FF FF 00 05 1B FF 21 FF 22 FF 23 FF 24 FF 00
 DB 04 00 00 00 DC F0 6C F0 7B F0 7C F0 4B F0 4C

10 31 FF 32 FF 33 FF 34 FF 35 FF 36 FF 37 FF 38 FF 10
 F0 C1 F0 C2 F0 C3 F0 C4 F0 C5 F0 C6 F0 C7 F0 C8

20 39 FF 3A FF 3B FF 3C FF 41 42 43 44 45 FF 47 49 20
 F0 C9 F0 4A F0 4B F0 4C D3 D4 D1 D2 F0 F0 D6 DD

30 4A FF 50 51 56 FF 57 FF 5A 61 FF 62 FF 63 FF 64 30
 F0 6D DE D7 F0 6E F0 6B D8 F0 F1 F0 F2 F0 F3 F0

PF1 PF2 PF3 PF4 PF5 PF6 PF7 PF8 PF9 PF10 ENTER PA2
 Bwd-F Fwd-F RECVR DEFLT DEL-R SAV-R BWD-R FWD-R Bwd-P Fwd-P Re-Dsp Exit

Screen 4

HYDRA 3000 r2.5 Copyright (c) 1992-99 Hydra Systems Inc All Rights Reserved
 Port 1B Logical Terminal 00 -- Link 00 Host 05 Nau 1F

TDF KEY --> HNETIX NAME: HNET IBM EXT 1.1 XLT : PDF :

00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F

40 FF 65 FF 66 FF 67 FF 68 FF 69 FF 6A FF 6B FF 6C 40
 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 7A F0 7B F0

50 FF 7C 7D 7E 7F A5 FF A6 FF A7 FF A8 FF A9 AA AB 50
 7C E8 E9 EA DA E0 10 E0 11 E0 12 E0 13 ED EE EF

60 AD FF 00 00 00 00 00 00 00 00 00 00 00 00 00 60
 F0 6D 00 00 00 00 00 00 00 00 00 00 00 00 00

70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 70
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

PF1 PF2 PF3 PF4 PF5 PF6 PF7 PF8 PF9 PF10 ENTER PA2
 Bwd-F Fwd-F RECVR DEFLT DEL-R SAV-R BWD-R FWD-R Bwd-P Fwd-P Re-Dsp Exit

Note: The last two Terminal Input screens are omitted here since they contain only nulls in this example. They are always available if needed for a more complex keyboard mapping.

Explanation of Terminal Input Sequence Tables

As mentioned above, the Terminal Input Sequence Tables are presented as the last four screens of the TDF, in paired rows, where the top row (Segmented Search Table, or **SST**) is searched for a match with the incoming character, and the bottom row (Function Interpretation Table, or **FIT**) provides an interpretation for each matched byte. Segments are separated by a null (x'00') byte in both **SST** and **FIT**; while x'FF' serves as a place holder or continuation byte.

The rows are labelled **00** through **F0**, the columns **00** through **0F**. The absolute position of any byte is found by combining its row and column; e.g., row **60**, column **0A** yield an absolute position of **6A**. The relative position, or offset, of one byte from another is found by subtracting absolute positions, e.g., the offset from **2C** to **73** is **47** (**73** minus **2C**).

For each incoming character, the top row (**SST**) of the appropriate segment is scanned until a match is found or the null byte segment terminator is encountered. If no match is found or a partial sequence is encountered, the characters are passed on unchanged, and the process restarted for the next input character. If a match is found, the corresponding byte in the bottom row (**FIT**) is examined to decide how to proceed. Processing of subsequent characters of the same sequence continues in similar fashion in following segments of the **SST** until either a function is determined and the sequence terminates, or until no further match is found, in which case processing is aborted.

When a match is found, the bit-mapped **FIT** byte will indicate whether or not a function token is at hand, if so, what class of function. The **FIT** byte will also indicate whether or not the sequence terminates with this match. A second, and possibly third, **FIT** byte may be required to specify the function token and/or the displacement to another **SST** segment; in this case, the bytes of the **SST** corresponding to these extra **FIT** bytes must be x'FF.

The left four bits of the first **FIT** byte are mapped as follows.

<u>Bit 7:</u> Token?	<u>Bit 6:</u> Terminated?	<u>Bit 5/Bit 4</u> (if Bit 7 = 1) Token Class
0 = no token 1 = token	0 = not terminated 1 = terminated	00 = Terminal Data Character 01 = Local Editing Function 10 = Terminal Control Function 11 = Attention Indicator Function

Bit 7 = 1: a token is present

When Bit 7 indicates a token is present, Bits 5/4 define the token class, as illustrated above. A brief explanation of each class follows; full lists are contained in the Function Token Tables below.

Attention Indicator Functions include all PF- and PA-keys, and Enter, Clear, Sys Request, Null Aid, and non-SNA Test Request.

Local Editing Functions are, for example, cursor control keys, Tab, BackTab, Newline, Home, and EraseInput.

Terminal Control Functions are exemplified by Reset Keyboard, SNA Attention, Refresh Screen, and *HYDRA 3000* session selection keys.

Terminal Data Characters provide the capability of translating an inbound sequence to an EBCDIC character code or special graphics code.

When a token is indicated (Bit 7 = 1), the token value is presented either in the remaining four bits (Bits 3-0) of the current **FIT** byte or in the following **FIT** byte, based on the following: if the token value is in the range x'01' – x'0F', it will fit in the remaining four bits; otherwise it must be placed in the next **FIT** byte, in which case the corresponding **SST** byte must be x'FF', and bits 3-0 of the first **FIT** byte must be zeros. The x'FF' is ignored in the match scanning process; see the examples below. Note that a token value of x'00' must always be presented as a separate byte.

Even though a token has been specified, there may be further, “trailing” characters to process in this sequence (Bit 6 = 0). In that case, the **SST** byte following the token must be x'FF', and its **FIT** byte must contain an absolute offset to the **SST** segment in which the next byte of this sequence is located. Eventually, Bit 6 must be set to 1 to indicate that processing of this sequence is complete.

Examples:

Example 1:

SST: ...43...
FIT: ...D1...

ASCII byte found: x'43'. The **FIT** byte, **D1**, breaks down into the binary bit sequence **1101 0001**, which indicates that: a token byte is present and included (1000 0001), with token value = **01**, the sequence is terminated (0100 0000), and the token class = **01** (0001 0000). This must be the sequence for a Right Arrow (see the Terminal Control Sequence Table below).

Example 2:

SST: ...33 FF...
FIT: ...F0 F3...

ASCII byte found: x'33'. The **FIT** byte, **F0**, or binary **1111 0000**, indicates that a token byte is present, but in the following **FIT** byte (since the low-order nibble is 0000), the token value = **F3**, the sequence is terminated, and the token class = **11**. This is the sequence for a PF3.

Example 3: (including termination)

SST: ...37 FF FF... ..7E FF
FIT: ...B0 F1 75... ..40 00

ASCII byte found: x'37'. Its **FIT** byte, **B0**, or binary **1011 0000**, indicates that a token byte is present in the following **FIT** byte (token value = **F1**), of token class = **11**, that the sequence is not yet terminated (Bit 6=0), with absolute offset to the next segment of x'75'. At offset x'75', the final byte of the sequence is found (ASCII byte x'7E'); its **FIT** byte, **40**, or binary **0100 0000**, indicates the sequence is terminated. This is the sequence for PF1.

Bit 7 = 0: no token is present

If Bit 6 and Bit 7 are both 0 in the **FIT** byte (non-terminated, no token), then a displacement to the beginning of another table segment is indicated, either as a relative offset from the current position in the table, or as an absolute offset from the beginning of the table. The possibilities are as follows:

If the relative displacement is less than x'40', but not x'00', then Bits 0-5 give the relative displacement, **d**, from this byte (x'00' < **d** < x'40').

If the relative displacement is greater than or equal to x'40', the next **FIT** byte gives the absolute displacement, **a**, from the beginning of the table to the beginning of the segment (x'3F' < **a** < x'100').

A few examples may help to clarify the translation process. In the Terminal Input Sequence Table (Screen 3, page 7-23), the first byte of the **SST** is x'10', the ASCII code for Ctrl-P, and the second byte is x'1B', the ASCII code for Escape. If the incoming character does not match either of these, the next two bytes scanned in the table are ignored (xFF'), and the fourth byte encountered is a null byte (x'00'), which terminates the segment and the search. Thus any incoming character that is not a Ctrl-P or an Escape (or is not immediately preceded by an Escape) passes through the TDF untouched. Note that this is independent of the ASCII-to-EBCDIC translation process.

Suppose that the hexadecimal sequence, **1B 33** (Escape 3), is received from the keyboard. When the first segment of the **SST** is scanned, the Escape character is found at offset x'01', and the x'04' byte (binary 0000 0100) under the x'1B' is deciphered. Since Bit 6 = 0, this is not the sequence terminator; since Bit 7 = 0, we have no token, and a relative displacement of four bytes is indicated to absolute location x'05'. When the next character of the sequence is received, the **SST** table is scanned starting with the fifth byte. A match for the '3', x'33', is found at position x'14'. The byte below that, x'F0' (binary '1111 0000'), tells us that this input byte terminates the sequence, and that a token value of class 'Attention Indicator' is to be found in the next byte under the x'FF' byte. The token value, x'C3'', is that of PF15 according to the table of Attention Indicator Tokens on page 7-28.

In the following example of a 3-character sequence, refer to the ANSI TDF, at the end of the chapter. We'll look at what happens with the sequence 1B 5B 41 (Escape [A). The 1B, found in byte 02, indicates a relative offset of 05, to position 07, the beginning of the next segment of the **SST**. A match for 5B will be found in this **SST** segment at x'2F'. The byte below that, x'38', indicates that a relative displacement of x'38'. Added to the current position, x'2F', this takes us to position x'67', the beginning of the last segment of the table; the search for the last byte of the sequence, 41, begins here. A match is found immediately. The x'D3' below the x'41' (binary '1101 0011'), tells us (and *HYDRA 3000*) that this is a terminating byte containing a token, to be interpreted as a 'Local Editing' class of token with a value of x'03'. According to the table of Terminal Function Tokens, the up-arrow key must have been depressed.

Function Token Tables**Terminal Data Characters; Token Class 0**

Data character	Token value	Data character	Token value
Field Mark (FM)	25	EBCDIC character	x'dd
Specific EBCDIC Characters:			
ECENT (Cent Sign)	4A	E2 (2)	F2
EDOT (Period)	4B	E3 (3)	F3
EPLUS (Plus Sign)	4E	E4 (4)	F4
ESTAR (Asterisk)	5C	E5 (5)	F5
EMINUS (Minus Sign)	60	E6 (6)	F6
ESLASH (Forward Slash)	61	E7 (7)	F7
E0 (0)	F0	E8 (8)	F8
E1 (1)	F1	E9 (9)	F9
Token Class = x'00'			

Local Editing Functions; Token Class 1

Editing function	Token value	Editing function	Token value
Cursor Right	01	Toggle Insert Mode	0E
Cursor Left	02	Duplicate (DUP)	10
Cursor Up	03	Cursor Select	11
Cursor Down	04	Light Pen Report	12
Forward Tab	05	Cursor Pos. Report	13
Backward Tab	06	Alpha Shift In	18
New-Line	07	Alpha Shift Out	19
Home Cursor	08	Numeric Shift In	1A
Insert Character	09	Numeric Shift Out	1B
Delete Character	0A	Toggle Echo	1C
Print Screen	0B	Set Echo	1D
Erase Input	0C	Clear Echo	1E
Erase EOF	0D	Ignore	1F
Token Class = x'01'			

Terminal Control Functions; Token Class 2

Terminal Control Function	Token value	Terminal Control Function	Token value
Reset 3270 Keyboard	00	Data Routing Binary Mode	09
Clear 3270 Kb	01	Data Routing Disconnect	0A
Disconnect 3270	02	Select Next Session	0D
SNA Attention	03	Select Previous Session	0E
Set TNDD	04	Select Prior Session	0F
Reset TNDD	05	Select Session 0	10
HYDRA 3000 Local services	06	Select Session 1	11
Refresh Screen	07	Select Session 2	12
Data Routing Toggle	08	Select Session 3	13

Token Class = x'10'

Attention Indicator Functions; Token Class 3

Attention Indicator Function	Token value	Attention Indicator Function	Token value
PA1	6C	PF26	D2
PA2	6E	PF27	D3
PA3	6B	PF28	D4
PF1	F1	PF29	D5
PF2	F2	PF30	D6
PF3	F3	PF31	D7
PF4	F4	PF32	D8
PF5	F5	PF33	D9
PF6	F6	PF34	5A
PF7	F7	PF35	5B
PF8	F8	PF36	5C
PF9	F9	PF37	5D
PF10	7A	PF38	5E
PF11	7B	PF39	5F
PF12	7C	PF40	E2
PF13	C1	PF41	E3
PF14	C2	PF42	E4
PF15	C3	PF43	E5
PF16	C4	PF44	E6
PF17	C5	PF45	E7
PF18	C6	PF46	E8
PF19	C7	PF47	E9
PF20	C8	PF48	6A
PF21	C9	PF49	6F
PF22	4A	Null Aid	60
PF23	4B	Clear	6D
PF24	4C	Enter	7D
PF25	D1	Sys Req	F0
		Non-SNA Test-Request	E0

Token Class = x'11'

Special HYDRA 3000 Function Keys

This section documents some special *HYDRA 3000* features that are outside of normal 3270 capabilities and require consideration when building a Terminal Definition.

Set/Reset TNDD

TNDD (Toggle Non-Display to Display) exists to support certain flavors of IND\$FILE file transfer, which send screens of data with the Non-Display bit on. *HYDRA 3000* would normally suppress the display of such data, and thus stop the file transfer process. By coding Set and Reset TNDD sequences in the appropriate TDF, then sending the Set TNDD sequence just before the transfer begins, and the Reset TNDD sequence just after the transfer completes, these IND\$FILE transfers will work without completely defeating the purpose of Non-Display fields.

Select 3270 Session 0-3

This set of sequences directly selects each of the four sessions. If there is no active host session for the selected *HYDRA 3000* session, the Unowned screen will be displayed; if the selected session is configured as a printer, no session switch is performed.

Select Next 3270 Session

This sequence selects the next session, in numerical order, or wraps to session 0 if appropriate; a printer session will be bypassed.

Select Previous 3270 Session

This sequence selects the previous session, in numerical order, or wraps to session 3 if appropriate; a printer session will be bypassed.

Select Prior 3270 Session

This sequence selects the session selected prior to the current session.

Data Routing Session Toggle

The Session Toggle key sequence defined in the TDF switches the user between the current 3270 session and the active async-to-async connection, if one has been established, or between the current 3270 session and the Route Selection menu. If you select a new 3270 connection, your current 3270 connection will be terminated; similarly, if you request a new Data Routing connection, your current connection will be terminated.

Data Routing Session Toggle, Binary Mode

If you need to transmit binary (non-text) data over an async-to-async Data Routing link, you may wish to define a key to initiate Binary Mode. To escape from Binary Mode, type three minus signs (“---”) and pause for at least one second.

Data Routing Session Disconnect

The Session Disconnect key defined in the TDF disconnects the currently selected 3270 or Data Routing connection and presents you with the Route Selection menu. It leaves other session connections intact.

ANSI/DEC VT100 Keyboard Mapping

The ANSI TDF may be used with a terminal or emulator that has basic compatibility with a DEC VT100 terminal; that is, the terminal or emulator should respond correctly to VT100 clear screen, cursor addressing and video attribute sequences. In addition, the character sequences generated by the arrow keys must be compatible, i.e., 1B 5B 41, 1B 5B 42, 1B 5B 43, and 1B 5B 44, for cursor up, down, right, and left, respectively.

The 3270/display functions supported by the ANSI TDF are achieved for the most part through multiple key sequences, as illustrated below.

3270/Display function	Key(s) to be used	3270/Display function	Key(s) to be used
PF1	ESC 1	PF21	ESC a
PF2	ESC 2	PF22	ESC s
PF3	ESC 3	PF23	ESC d
PF4	ESC 4	PF24	ESC f
PF5	ESC 5	PA1	ESC z
PF6	ESC 6	PA2	ESC x
PF7	ESC 7	PA3	ESC c
PF8	ESC 8	SysRequest	ESC v
PF9	ESC 9	Back Tab	<Ctrl> B
PF11	ESC q	Clear	<Ctrl> C
PF14	ESC r	Delete Char.	<Ctrl> D
PF15	ESC t	Erase EOF	<Ctrl> F
PF16	ESC y	Enter	Return
PF17	ESC u	Return	<Ctrl> N
PF18	ESC I	Reset	<Ctrl> R
PF19	ESC o	Home	<Ctrl> V
PF20	ESC p	TEST MODE	<Ctrl> Y
Disconnect	<Ctl> @		

Chapter 7: Terminal Definitions

HYDRA 3000 r2.5 Copyright (c) 1992-99 Hydra Systems Inc All Rights Reserved
Port 1B Logical Terminal 00 -- Link 00 Host 05 Nau 1F

```
TDF KEY --> VT100      NAME: REAL DEC VT-100      1.0      XLT :

ATTR : NO                SSO NR NC L0 L1 L2 L3          VST0: 30303130 30303130
CBGF : NO      (24X80) : 3E 18 50 FF 00 FF FF          30303130 30303130
TNDD : NO      (32X80) : FF 19 50 FF FF FF FF          00000000 00000000
ASCRL: YES     (43X80) : FF 19 50 FF FF FF FF          00000000 00000000
AWRAP: YES     (27X132): 43 18 84 FF 03 FF FF          00000000 00000000
ALINE: NO
ACRNL: NO      IBM4C: 00010203 04050607 08090A0B 0C0D0E0F 00000000 00000000
ALFNL: NO
ACP : NO
DKBRD: NO      AXON: 36 CPF : 16                      SIVI: 00      SSND: FF      SUNO: 6E
EXTAT: NO      AXOF: 3A ADD : 01      EEOL: 22      KLCK: 48      SRCV: FF      SSCP: FF
3270 : NO      SGR : 1C      EEOS: 25      KNLK: 53      SCNT: FF      SPLU: FF
NBS : 00      CRS : 06 ATVI: 00      CEOL: 28      ISET: 5E      STST: 72
NFS : 00      CLS : 0A OPNS: 32      CEOS: 2B      IRST: 62      RANS: FF      SDRT: FF
NCU : 00      CUS : 0E CLSS: FF      ERSS: FF      SERP: 66      SASA: FF
NCD : 00      CDS : 12      CLRS: 2E      RERP: 6A      SNSA: FF

PF1 PF2 PF3 PF4 PF5 PF6 PF7 PF8 PF9 PF10      ENTER PA2
Bwd-F Fwd-F RECVR DEFLT DEL-R SAV-R BWD-R FWD-R Bwd-P Fwd-P      Re-Dsp Exit
```

```
TDF KEY --> VT100      NAME: REAL DEC VT-100      1.0      XLT:
00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F

00 1B 5B 00 1B 5B 00 91 87 43 00 91 87 44 00 91 87 00
10 41 00 91 87 42 00 91 82 3B 83 48 00 91 9B 97 85 10
20 6D 00 91 4B 00 91 4A 00 91 4B 00 91 4A 00 91 32 20
30 4A 00 1B 29 30 00 91 35 69 00 91 34 69 00 91 3F 30
40 33 6C 00 91 3F 33 68 00 91 31 71 91 31 3B 38 30 40
50 48 2A 00 91 30 71 91 31 3B 38 30 48 20 00 91 32 50
60 71 00 91 30 71 00 91 33 71 00 91 30 71 00 91 30 60
70 71 00 91 34 71 00 00 00 00 00 00 00 00 00 00 70
80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80
90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90
A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0
B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B0
C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C0
D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D0
E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E0
F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F0
```

```
TDF KEY --> VT100      NAME: REAL DEC VT-100      1.0      XLT :
00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F

00 08 0A 1B FF FF 7F 00 1B FF FF 30 FF 31 FF 32 FF 00
D6 D7 05 00 00 DA 00 3A 00 00 F0 7A F0 F1 F0 F2

10 33 FF 34 FF 35 FF 36 FF 37 FF 38 FF 39 FF 43 FF 10
F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 D0 11

20 44 FF 45 FF 46 FF 4F FF FF 51 FF 52 FF 57 FF 5B 20
D0 10 F0 6B C0 25 38 00 00 F0 6C F0 F0 F0 6E 38

30 FF FF 63 FF 64 FF 65 FF 66 FF 71 FF 72 FF 77 FF 30
00 00 D0 11 D0 10 F0 6B C0 25 F0 6C F0 F0 F0 6E

40 00 08 FF 2D FF 30 FF 31 FF 32 FF 33 FF 34 FF 35 40
00 F0 4C F0 C9 F0 C8 F0 7B F0 7C F0 C1 F0 C2 F0

50 FF 36 FF 37 FF 38 FF 39 FF 3D FF 60 FF 00 50 FF 50
C3 F0 C4 F0 C5 F0 C6 F0 C7 F0 4A F0 4B 00 F0 F1

60 51 FF 52 FF 53 FF 00 41 42 43 44 00 00 00 00 00 60
F0 F2 F0 F3 F0 F4 00 D3 D4 D1 D2 00 00 00 00 00

70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 70
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```