Network Working Group   Mark Crispin
Request for Comments 734   SU-AI
NIC 41953   7 October 1977

SUPDUP Protocol

INTRODUCTION

This document describes the SUPDUP protocol, a highly efficient display
telnet protocol. It originally started as a private protocol between the
ITS systems at MIT to allow a user at any one of these systems to use one
of the others as a display. At the current writing, SUPDUP user programs
also exist for Data Disc and Datamedia displays at SU-AI and for
Datamedias at SRI-KL. The author is not aware of any SUPDUP servers other
than at the four MIT ITS sites.

The advantage of the SUPDUP protocol over an individual terminal's
protocol is that SUPDUP defines a "virtual" or "software" display terminal
that implements relevant cursor motion operations.   The protocol is not
built on any particular display terminal but rather on the set of
functions common to all display terminals; hence it is completely device-
independent. In addition, the protocol also provides for terminals which
cannot handle certain operations, such as line or character insert/delete.
In fact, it is more than this.   It provides for terminals which are
missing any set of features, all the way down to model 33 Teletypes.

The advantage over the TELNET protocol is that SUPDUP takes advantage of
the full capabilities of display terminals, although it also has the
ability to run printing terminals.

It is to be noted that SUPDUP operates independently from TELNET; it is
not an option to the TELNET protocol.   In addition, certain assumptions
are made about the server and the user programs and their capabilities.
Specifically, it is assumed that the operating system on a server host
provides all the display-oriented features of ITS. However, a server may
elect not to do certain display operations available in SUPDUP; the SUPDUP
protocol is far-reaching enough so that the protocol allows terminals to
be handled as well as that host can handle terminals in general.   Of
course, if a host does not support display terminals in any special way,
there is no point in bothering to implement a SUPDUP server since TELNET
will work just as well.

A more complete description of the display facilities of SUPDUP and ITS
can be found by FTP'ing the online file .INFO.;ITS TTY from ARPAnet host
MIT-AI (host 206 octal, 134. decimal). For more information, the mailing
address for SUPDUP is "(BUG SUPDUP) at MIT-AI". If your mail system won't
allow you to use parentheses, use Bug-SUPDUP@MIT-AI.

\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

BACKGROUND

The SUPDUP protocol originated as the internal protocol used between parts
of ITS, and between ITS and "intelligent" terminals. Over the network, a
user host acts like an intelligent terminal programmed for ITS.

The way terminal output works in ITS is as follows: The user program
tells the system to do various operations, such as printing characters,
clearing the screen, moving the cursor, etc. These operations are formed
into 8-bit characters (using the %TD codes described below) and stored
into a buffer.   At interrupt level, as the terminal demands output,
characters are removed from the buffer and translated into terminal
dependent codes. At this time padding and cursor motion optimization are
also done.

In some cases, the interrupt side does not run on the same machine as the
user program. SUPDUP terminals have their "interrupt side" running in the
user host. When SUPDUP is run between two ITS's, the SUPDUP user and
server programs and the network simply move characters from the buffer in
the server machine to the buffer in the user machine. The interrupt side
then runs on the user machine just as if the characters had been generated
locally.

Due to the highly interactive characteristics of both the SUPDUP protocol
and the ITS system, all transactions are strictly character at a time and
all echoing is remote.   In addition, all padding and cursor control
optimization must be done by the user.

Because this is also the internals of ITS, the right to change it any time
if necessary to provide new features is reserved by MIT. In particular,
the initial negotiation is probably going to be changed to transmit
additional variables, and additional %TD codes may be added at any time.
User programs should ignore those they don't know about.

The following conventions are used in this document: function keys (ie,
keys which represent a "function" rather than a "graphic character") are
in upper case in square brackets. Prefix keys (ie, keys which generate no
character but rather are held down while typing another character to
modify that character) are in upper case in angle brackets.   Hence
"<CONTROL><META>[LINE FEED]" refers to the character generated when both
the CONTROL and META keys are held down while a LINE FEED is typed. Case
should be noted; <CONTROL>A refers to a different character from
<CONTROL>a. Finally, all numbers which do not explicitly specify a base
(ie, octal or decimal) should be read as octal unless the number is
immediately followed by a period, in which case it is decimal.
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

INITIALIZATION

The SUPDUP server listens on socket 137 octal. ICP proceeds in the normal
way for establishing 8-bit connections.   After the ICP is completed, the
user side sends several parameters to the server side in the form of
36.-bit words. Each word is sent through the 8-bit connection as six
6-bit bytes, most-significant first. Each byte is in the low-order 6 bits
of a character. The first word is the negative of the number of variables
to follow in the high order 18. bits (the low-order 18. bits are ignored),
followed by the values of the TCTYP, TTYOPT, TCMXV, TCMXH, and TTYROL
terminal descriptor variables (these are the names they are known by at
ITS sites). These variables are 36.-bit binary numbers and define the
terminal characteristics for the virtual terminal at the REMOTE host.

The count is for future compatability. If more variables need to be sent
in the future, the server should assume "reasonable" default values if the
user does not specify them. PDP-10 fans will recognize the format of the
count (ie, -count,,0) as being an AOBJN pointer. At the present writing
there are five variables hence this word should be -5,,0.

The TCTYP variable defines the terminal type. It MUST be 7 (%TNSFW). Any
other value is a violation of protocol.

The TTYOPT variable specifies what capabilities or options the user's
terminal has. A bit being true implies that the terminal has this option.
This variable also includes user options which the user may wish to alter
at his or her own descretion; these options are included since they may be
specified along with the terminal capabilities in the initial negotiation.
See below for the relevant TTYOPT bits.

The TCMXV variable specifies the screen height in number of lines.

The TCMXH variable specifies the line width in number of characters. This
value is one less than the screen width (ITS indicates line overflow by
outputting an exclamation point at the end of the display line before
moving to the next line). Note: the terminal must not do an automatic
CRLF when a character is printed in the rightmost column.   If this is
unavoidable, the user SUPDUP must decrement the width it sends by one.

Note: Setting either the TCMXV or TCMXH dimension greater than 128. will
work, but will have some problems as coordinates are sometimes represented
in only 7 bits.   The main problems occur in the SUPDUP protocol when
sending the cursor position after an output reset and in ITS user programs
using the display position codes ^PH and ^PV.

The TTYROL variable specifies the "glitch count" when scrolling. This is
the number of lines to scroll up when scrolling is required. If zero, the
terminal is not capable of scrolling. 1 is the usual value, but some
terminals glitch up by more than one line when they scroll.

Following the transmission of the terminal options by the user, the server
should respond with an ASCII greeting message, terminated with a %TDNOP
code (%TD codes are described below). All transmissions from the server
after the %TDNOP are either printing characters or virtual terminal
display codes.
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

```
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
```

The user and the server now both communicate using the intelligent
terminal protocol (described below) from the user and %TD codes from the
server. The user has two commands in addition to these; they are escaped
by sending 300 (octal). If following the escape is a 301 (octal), the
server should attempt to log off the remote job (generally this is sent
immediately before the user disconnects, so this logout procedure should
be done regardless of the continuing integrity of the connection). If the
character following the escape is a 302 (octal), all ASCII characters
following up to a null (000 octal) are interpreted as "console location"
which the server can handle as it pleases. No carriage return or line
feed should be in the console location text. Normally this is saved away
to be displayed by the "who" command when other users ask where this user
is located.
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

TTYOPT FUNCTION BITS

The relevant TTYOPT bits for SUPDUP usage follow. The values are given in
octal, with the left and right 18-bit halves separated by ",," as in the
usual PDP-10 convention.

Bit name	Value	Meaning

%TOALT   200000,,0   characters 175 and 176 are converted to
                                altmode (033) on input.

%TOERS   40000,,0   this terminal is capable of selectively
                                erasing its screen. That is, it supports
                                the %TDEOL, the %TDDLF, and (optionally)
                                the %TDEOF operations.   For terminals
                                which can   only do   single-character
                                erasing, see %TOOVR.

%TOMVB   10000,,0   this terminal is capable of backspacing
                                (ie, moving the cursor backwards).

%TOSAI   4000,,0   this   terminal   has   the Stanford/ITS
                                extended ASCII graphics character set.

%TOOVR   1000,,0   this terminal is capable of overprinting;
                                if two characters are displayed in the
                                same position, they will both be visible,
                                rather than one replacing the other.

                                Lack of this capability but the capability
                                to backspace (see %TOMVB) implies that the
                                terminal can do single character erasing
                                by overstriking with a space. This allows
                                terminals without the %TOERS capability to
                                have display-style "rubout processing", as
                                this capability depends upon either %TOERS
                                or [%TOMVB and not %TOOVR].

%TOMVU   400,,0   this terminal is capable of moving the
                                cursor upwards.

%TOLWR   20,,0   this terminal's keyboard is capable of
                                generating lowercase characters; this bit
                                is mostly provided for programs which want
                                to know this information.

%TOFCI   10,,0   this terminal's keyboard is capable of
                                generating CONTROL and META characters as
                                described below.

%TOLID   2,,0   this terminal is capable of doing line
                                insert/delete operations, ie, it supports
                                %TDILP and %TDDLP.

%TOCID   1,,0   this   terminal   is   capable   of doing
                                character insert/delete operations, ie, it
                                supports %TDICP and %TDDCP.
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

```
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000
```

TTYOPT FUNCTION BITS (continued)

Bit name   Value   Meaning

%TPCBS   0,,40   this terminal is using the "intelligent
                                    terminal protocol".
                                     THIS BIT MUST BE ON.

%TPORS   0,,10   the server should process output resets
                                    instead of ignoring them.
                                     IT IS HIGHLY RECOMMENDED THAT THIS BIT BE
                                    ON; OTHERWISE THERE MAY BE LARGE DELAYS IN
                                    ABORTING OUTPUT.

The following bits are user option bits. They may be set or not set at
the user's discretion. The bits that are labelled "normally on" are those
that are normally set on when a terminal is initialized (ie, by typing
[CALL] on a local terminal).

Bit name   Value   Meaning

%TOCLC   100000,,0   convert lower-case input to upper case.
                                    Many terminals have a "shift lock" key
                                    which makes this option useless.
                                     NORMALLY OFF.

%TOSA1   2000,,0   characters 001-037 should be displayed
                                    using the Stanford/ITS extended ASCII
                                    graphics character set instead of uparrow
                                    followed by 100+character.
                                     NORMALLY OFF.

%TOMOR   200,,0   the   system   should   provide "**MORE**"
                                    processing when the cursor reaches the
                                    bottom line of the screen.   **MORE**
                                    processing is described in ITS TTY.
                                     NORMALLY ON.

%TOROL   100,,0   the terminal should scroll when attempting
                                    output below the bottom line of the screen
                                    instead of wrapping around to the top.
                                     NORMALLY OFF.
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000

INPUT -- THE INTELLIGENT TERMINAL PROTOCOL

Note: only the parts of the intelligent terminal protocol relevant to
SUPDUP are discussed here. For more information, read ITS TTY.

CHARACTER SETS

There are two character sets available for use with SUPDUP; the 7-bit
character set of standard ASCII, and the 12-bit character set of extended
ASCII. Extended ASCII has 5 high order or "bucky" bits on input and has
graphics for octal 000-037 and 177 (see the section entitled "Stanford/ITS
character set" for more details). The two character sets are identical on
output since the protocol specifies that the host should never send the
standard ASCII formatting characters (ie, TAB, LF, VT, FF, CR) as
formatting characters; the characters whose octal values are the same as
these formatting characters are never output unless the user job has these
characters enabled (setting %TOSAI and %TOSA1 generally does this).

Input differs dramatically between the 7-bit and 12-bit character sets.
In the 7-bit character set, all characters input whose value is 037 octal
or less are assumed to be (ASCII) control characters. In the 12-bit
character set, there are 5 "bucky" bits which may be attached to the
character. The two most important of these are CONTROL and META, which
form a 9-bit character set. TOP is used to distinguish between printing
graphics in the extended character set and ASCII controls. The other two
are reserved and should be ignored. Since both 7-bit and 12-bit terminals
are commonly in use, 0001, 0301, and 0341 are considered to be <CONTROL>A
on input by most programs, while 4001 is considered to be downwards arrow.

MAPPING BETWEEN CHARACTER SETS

Many programs and hosts do not process 12-bit input. In this case, 12-bit
input is folded down to 7-bit as follows: TOP and META are discarded.   If
CONTROL is on, then if the 7-bit part of the character specifies a lower
case alphabetic it is converted to upper case; then if the 7-bit part is
between 077 and 137 the 100 bit is complemented or if the 7-bit part is
040 the 040 bit is subtracted (that's right, <CONTROL>? is converted to
[RUBOUT] and <CONTROL>[SPACE] is converted to [NULL]).   In any case the
CONTROL bit is discarded, and the remainder is treated as a 7-bit ASCII
character. It should be noted that in this case downwards arrow is read
by the program as standard ASCII <CONTROL>A.

Servers which expect 12-bit input and are told to use the 7-bit character
set should do appropriate unfolding from the 7-bit character set to
12-bit. It is up to the individual server to decide upon the unfolding
scheme. On ITS, user programs that use the 12-bit character set generally
have an alternative method for 7-bit; this often takes the form of prefix
characters indicating that the next character should be "controllified" or
"metized", etc.
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

INPUT -- THE INTELLIGENT TERMINAL PROTOCOL (continued)

BUCKY BITS

Under normal circumstances, characters input from the keyboard are sent to
the foreign host as is. There are two exceptions; the first occurs when
an octal 034 character is to be sent; it must be quoted by being sent
twice, because 034 is used as an escape character for protocol commands.
The second exception occurs when %TOFCI is set and a character with
non-zero bucky bits is to be sent. In this case, the character, which is
in the 12-bit form:

Name   Value   Description

%TXTOP   4000   This character has the [TOP] key depressed.

%TXSFL   2000   Reserved, must be zero.

%TXSFT   1000   Reserved, must be zero.

%TXMTA   400   This character has the [META] key depressed.

%TXCTL   200   This character has the [CONTROL] key depressed.

%TXASC   177   The ASCII portion of the character

is sent as three bytes. The first byte is always 034 octal (that is why
034 must be quoted). The next byte contains the "bucky bits", ie, the
%TXTOP through %TXCTL bits, shifted over 7 bits (ie, %TXTOP becomes 20)
with the 100 bit on. The third byte contains the %TXASC part of the
character. Hence the character <CONTROL><META>[LINE FEED] is sent as 034
103 012.

OUTPUT RESETS

The intelligent terminal protocol also is involved when a network
interrupt (INR/INS) is received by the user program. The user program
should increment a count of received network interrupts when this happens.
It should not do any output, and if possible abort any output in progress,
if this count is greater than zero (NOTE: the program MUST allow for the
count to go less than zero).

Since the server no longer knows where the cursor is, it suspends all
output until the user informs it of the cursor position. This also gives
the server an idea of how much was thrown out in case it has to have some
of the aborted output displayed at a later time. The user program does
this when it receives a %TDORS from the server. When this happens it
should decrement the "number of received network interrupts" count
described in the previous paragraph and then send 034 followed by 020, the
vertical position, and the horizontal position of where the cursor
currently is located on the user's screen.
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

OUTPUT -- DISPLAY PROTOCOL (%TD CODES)

Display output is somewhat simpler. Codes less than 200 octal are
printing characters and are displayed on the terminal (see the section
describing the "Stanford/ITS character set"). Codes greater than or equal
to 200 (octal) are known as "%TD codes", so called since their names begin
with %TD. The %TD codes that are relevant to SUPDUP operation are listed
here. Any other code received should be ignored, although a bug report
might be sent to the server's maintainers. Note that the normal ASCII
formatting characters (011 - 015) do NOT have their formatting sense under
SUPDUP and should not occur at all unless the Stanford/ITS extended ASCII
character set is in use (ie, %TOSAI is set in the TTYOPT word).

For cursor positioning operations, the top left corner is (0,0), ie,
vertical position 0, horizontal position 0.

%TD code	Value	Meaning
%TDMOV	200	General cursor position code. Followed by four bytes; the first two are the "old" vertical and horizontal positions and may be ignored. The next two are the new vertical and horizontal positions. The cursor should be moved to this position.
		On printing consoles (non %TOMVU), the old vertical position may differ from the true vertical position; this can occur when scrolling. In this case, the user program should set its idea of the old vertical position to what the %TDMOV says and then proceed. Hence a %TDMOV with an old vpos of 20. and a new vpos of 22. should always move the "cursor" down two lines. This is used to prevent the vertical position from becoming infinite.
%TDMV1	201	An internal cursor motion code which should not be seen; but if it is, it has two argument bytes after it and should be treated the same as %TDMV0.
%TDEOF	202	Erase to end of screen. This is an optional function since many terminals do not support this. If the terminal does not support this function, it should be treated the same as %TDEOL.
		%TDEOF does an erase to end of line, then erases all lines lower on the screen than the cursor. The cursor does not move.
%TDEOL	203	Erase to end of line. This erases the character position the cursor is at and all positions to the right on the same line. The cursor does not move.

\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

```
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
```

OUTPUT -- DISPLAY PROTOCOL (%TD CODES) (continued)

%TD code	Value	Meaning
%TDDLF	204	Clear the character position the cursor is on. The cursor does not move.
%TDCRL	207	If the cursor is not on the bottom line of the screen, move cursor to the beginning of the next line and clear that line. If the cursor is at the bottom line, scroll up.
%TDNOP	210	No-op; should be ignored.
%TDORS	214	Output reset. This code serves as a data mark for aborting output much as IAC DM does in the ordinary TELNET protocol.
%TDQOT	215	Quotes the following character. This is used when sending 8-bit codes which are not %TD codes, for instance when loading programs into an intelligent terminal. The following character should be passed through intact to the terminal.
%TDFS	216	Non-destructive forward space. The cursor moves right one position; this code will not be sent at the end of a line.
%TDMV0	217	General cursor position code. Followed by two bytes; the new vertical and horizontal positions.
%TDCLR	220	Erase the screen. Home the cursor to the top left hand corner of the screen.
%TDBEL	221	Generate an audio tone, bell, whatever.
%TDILP	223	Insert blank lines at the cursor; followed by a byte containing a count of the number of blank lines to insert. The cursor is unmoved. The line the cursor is on and all lines below it move down; lines moved off the bottom of the screen are lost.
%TDDLP	224	Delete lines at the cursor; followed by a count. The cursor is unmoved. The first line deleted is the one the cursor is on. Lines below those deleted move up. Newly-created lines at the bottom of the screen are blank.

\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

```
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000
```

OUTPUT -- DISPLAY PROTOCOL (%TD CODES) (continued)

%TD code   Value   Meaning

%TDICP   225   Insert blank character positions at the
                                cursor; followed by a count. The cursor
                                is unmoved.   The character the cursor is
                                on and all characters to the right on the
                                current line move to the right; characters
                                moved off the end of the line are lost.

%TDDCP   226   Delete characters at the cursor; followed
                                by a count. The cursor is unmoved.   The
                                first character deleted is the one the
                                cursor is on. Newly-created characters at
                                the end of the line are blank.

%TDBOW   227   Display black characters on white screen.
                                HIGHLY OPTIONAL.

%TDRST   230   Reset %TDBOW and such any future options.

```
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000
```

STANFORD/ITS CHARACTER SET

This section describes the extended ASCII character set.   It originated
with the character set developed at SAIL but was modified for 1968 ASCII.

This character set only applies to terminals with the %TOSAI and %TOFCI
bits set in its TTYOPT word. For non-%TOSAI terminals, the standard ASCII
printing characters are the only available output characters.   For
non-%TOFCI terminals, the standard ASCII characters are the only available
input characters.

PRINTING CHARACTERS

The first table describes the printing characters. For output, the 7-bit
code is sent (terminal operations are performed by %TD codes). For input,
the characters with values 000-037 and 177 must have the %TXTOP bit on to
indicate the graphic is intended rather than a function or ASCII control.

Value   Character

 4000   centered dot
 4001   downward arrow
 4002   alpha
 4003   beta
 4004   logical AND
 4005   logical NOT
 4006   epsilon
 4007   pi
 4010   lambda
 4011   gamma
 4012   delta
 4013   uparrow
 4014   plus-minus
 4015   circle-plus
 4016   infinity
 4017   partial delta
 4020   proper subset (left horseshoe)
 4021   proper superset (right horseshoe)
 4022   intersection (up horseshoe)
 4023   union (downward horseshoe)
 4024   universal quantifer
 4025   existential quantifier
 4026   circle-X
 4027   double arrow
 4030   left arrow
 4031   right arrow
 4032   not-equal
 4033   lozenge (diamond)
 4034   less-than-or-equal
 4035   greater-than-or-equal
 4036   equivalence
 4037   logical OR
 0040   first standard ASCII character (space)
  ..   . . .
 0176   last standard ASCII character (tilde)
 4177   integral
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000

STANFORD/ITS CHARACTER SET (continued)

FUNCTION KEYS AND SPECIAL CHARACTERS

In addition, the following special characters exist for input only. These
characters are function keys rather than printing characters; however,
some of these characters have some format effect or graphic which they
echo as; the host, not the SUPDUP program, handles any such mappings.

Value   Character   Usual echo   Usual Function

 0000   [NULL]
 0010   [BACK SPACE]   text formatting
 0011   [TAB]   text formatting
 0012   [LINE FEED]   text formatting
 0013   [VT]   text formatting
 0014   [FORM]   text formatting
 0015   [RETURN]   text formatting
 0032   [CALL]   uparrow-Z   escape to system
 0033   [ALTMODE]   lozenge or $ special activation
 0037   [BACK NEXT]   uparrow-underscore   monitor command prefix
 0177   [RUBOUT]   character delete

 4101   [ESCAPE]   local terminal command
 4102   [BREAK]   local subsystem escape
 4103   [CLEAR]
 4110   [HELP]   requests a help message

BUCKY BITS

For all input characters, the following "bucky bits" may be added to the
character. Their interpretation depends entirely upon the host. <TOP> is
not listed here, as it has been considered part of the character in the
previous tables.

<CONTROL> is different from ASCII CTRL, however, many programs may request
the operating system to map such characters to the ASCII forms (with the
<TOP> bit off). In this case <META> is ignored.

Value   Key

 2000   Reserved
 1000   Reserved
 0400   <META>
 0200   <CONTROL>
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000

ACKNOWLEDGEMENTS

Richard M. Stallman (RMS@MIT-AI) and David A. Moon (Moon@MIT-MC) of the
MIT-AI and MIT-MC systems staff wrote the source documentation and the
wonderful ITS terminal support that made this protocol possible. It must
be emphasized that this is a functional protocol which has been in
operation for some years now.

In addition, Moon, Stallman, and Michael McMahon (MMcM@SRI-KL) provided
many helpful comments and corrections to this document.

For further reference, the sources for the known currently existing SUPDUP
user programs are available online as:

[MIT-AI] SYSENG;SUPDUP >   for the ITS monitor,
[SU-AI] SUPDUP.MID[NET,MRC]   for the SAIL monitor,
[SRI-KL] <MMcM>SD.FAI   for the TOPS-20 monitor.

The source for the known currently existing SUPDUP server program is:

[MIT-AI] SYSENG;TELSER >   for the ITS monitor.

These programs are written in the MIDAS and FAIL dialects of PDP-10
assembly language.

\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000