SyncTERM v1.3 Manual

Table of Contents

SyncTERM is a terminal program written specifically for connecting to Bulleten Board Systems (BBSs). Despite the name, SyncTERM is in no way Synchronet specific, it just happens to share a large portion of code with the rest of the Synchronet project, and live in the same git repository.

Getting Support

If you need help with SyncTERM, the best places are:

IRC

Connect to irc.synchro.net and find Deuce in #Synchronet. Ask your question, then idle. I can take hours to respond. Do not give up, this is the quickest way to get a response.

SourceForge

The official SyncTERM project page has a bug tracker and other features that will email me and provide tracking for issues that are filed.

E-Mail

I am usually fairly responsive to emails sent to me at shurd@sasktel.net. Please describe your issue as clearly as possible.

Dove-Net

> While I no longer read Dove-Net regularly, many other users can often help with support issues. Ask questions in the Hardware/Software Help sub. If your local BBS does not carry Dove-Net, you can telnet to vert.synchro.net and leave messages there.

Throughout this document, I will mention things which are not supported. These are things which I don't normally test, and are unlikely to work at any given time. If you ask for support on one of these issues, I may help out, or I may not. It doesn't bother me if you ask for help on these things, but if you continue to ask for help after I refuse, it will make it less likely I'll work on it in the future.

History

I started writing SyncTERM in June of 2004. There weren't any good BBS clients available for FreeBSD at the time, so I started writing one. Initially, it was RLogin only and no file transfer support existed or was planned. since RLogin supports auto-login with user ID and password on Synchronet systems, and RLogin is a much simpler protocol than telnet, no telnet support was planned. Digital Man (authour of Synchronet) added telnet and ZModem support a year later, and SyncTERM became a generally usable BBS client. New features continued to be added slowly over the years.

Getting SyncTERM

Releases of SyncTERM are available on the SourceForge project page. Nightly builds and source bundles are also available at the SyncTERM website for the more adventurous.

Compiling SyncTERM from Source

Windows users should not need to build SyncTERM from source. Windows specifically is not an easy build to do. The Windows builds are done on a FreeBSD system using MinGW, and this is the only supported build method currently. There is a Visual Studio project for building SyncTERM, but this is not supported.

For *nix systems (Linux, FreeBSD, macOS, and others), a GNU make based build system is used. There are a number of optional dependencies, and a large number of supported compile flags (many of which are shared with Synchronet).

The biggest optional dependency is SDL 2. SyncTERM can use SDL for both graphics and sound. X11 can also be used for graphics, and OSS, ALSA, or Portaudio can also provide sound. These use run-time linking, so at compile time, only the headers are needed. Static linking with SDL is also supported.

For SSH, a copy of Peter Gutmann's Cryptlib is provided along with a set of patches. This is still an optional dependency, so if Cryptlib doesn't build on your platform, you can still use SyncTERM's other connection options. Cryptlib must be statically linked if it is used.

Other optional dependencies (such as Portaudio) can not be statically linked, and are only

supported with static linking.

Once you have the desired dependencies installed, change to the syncterm directory in the source tree (ie: syncterm-YYYYMMDD/src/syncterm) and run the `make RELEASE=1` command. This will generate the binary in a subdirectory with the following name format: [compiler].[os].[arch].exe.[build]

[compiler]

is likely either gcc or clang depending on the system compiler.

[os]

is the OS name reported by uname.

[arch]

is the architecture reported by uname unless it is x86 compatible in which case it is forced to x86 for historical reasons.

[build]

is "release" for release builds and "debug" for debug builds.

SyncTERM can be installed with `make RELEASE=1 install`

Running SyncTERM

SyncTERM supports many command-line options to control behaviour. Options begin with a - followed by one or more other characters. The following options are supported (options are not case sensitive unless specifically noted):

-6

Specifying -6 forces SyncTERM to use IPv6 addresses when possible.

-4

Specifying -4 forces SyncTERM to use IPv4 addresses when possible.

-B/path/to/bbs/list.lst

Loads the user BBS list from the specified file instead of the default.

-C

Changes the default to no status line.

-E##

Specifies the escape delay in ANSI on Curses modes. The escape delay is how long SyncTERM will wait after an escape key is received from the user to see if it's a control sequence or a bare `Escape` press. The units are millisecods, and the default is 25.

-H

Use SSH mode when connecting.

-I[ACFXWS[WF]O[WF]]

Selects the output mode. Not all modes are available in all builds or on all platforms. Legal values are:

A

ANSI output mode. This mode outputs ANSI control sequences to stdout.

C

Curses output mode. For use in *nix terminals.

F

Curses with forced IBM character set. Limited usefulness.

G[WF]

Windows GDI output mode. Uses the Win32 API directly, and is the default for Windows. Additionally, a 'W' or 'F' can be specified to force windowed or full-screen mode respectively.

I

Curses in ASCII mode.

S[WF]

SDL window output mode. Uses the SDL library for output. Additionally, a 'W' or 'F' can be specified to force windowed or full-screen mode respectively.

W

Windows console mode. Windows only mode which uses the system console APIs for output.

X[WF]

X11 output mode. UNIX only mode which directly uses the X11 libraries for drawing. The default where supported. Additionally, a 'W' or 'F' can be specified to force windowed or full-screen mode respectively.

Refer to the Output Modes section for more details.

-L##

Specifies the number of lines on the "screen". Supported values are: 14, 21, 25 (default), 28, 43, 50, and 60. If an unsupported value is used, it will default use 50.

-pns_*

Only "supported" on macOS. Ignored.

-N/path/to/config.ini

Loads the configuration from the specified file instead of the default.

-Q

Quiet mode, doesn’t show popups by default.

-R

Use RLogin mode when connecting.

-T

Use Telnet mode when connecting. If an upper-case -T is the only argument passed to SyncTERM however, SyncTERM will output a terminfo entry on stdout then exit. See Installing Terminfo Entry for more details.

-S

Use "safe" mode. This mode attempts to restrict the ability of the user to modify the local drive contents. This has not been exhaustively audited, and should therefore not be trusted.

-v

This option is case sensitive and must the be only option passed to SyncTERM. Causes syncterm to output the version on stdout then exit.

After the options, a full URI, hostname, or dialing directory entry may be specified. Supported URI schemes are: rlogin://, ssh://, telnet://, raw://, shell://, ghost://.

If there is an entry matching the URI, hostname, or entry name, the settings will be loaded from the BBS list, then modified per the command-line arguments.

The User InterFaCe

Menus in SyncTERM use a common user interface library named UIFC. This library was originally developed for Synchronet.

The following is the general behaviour of UIFC menus.

Mouse controls

Right-click

Same as pressing ESC (ie: exit menu).

Left-click

Select an item in a menu.

If there is a blank line at the end of the menu, you can select it to insert a new item.

Menus have a standard set of mouse controls. If you click outside of a menu, that menu is usually closed, but in some cases, it may simply become inactive. At the top of each menu is a block which is used to close the menu. If there is help for the menu, there is also a ? button to bring up the help.

If there are more options than fit in the window, there is a scrollbar on the left side.

Left-Drag

Select and copy a region (the copy is made when the button is released).

Middle-click

Paste from PRIMARY selection or clipboard.

Keyboard Controls

`Return`

Select the currently highlighted option. If there is a blank line at the end of the menu, you can select it to insert a new item.

`Escape`

Exit the current menu.

`Backspace`

An alias for `Escape`.

`Ctrl` + `C`

An alias for `Escape`.

`Home`

Jump to the beginning of the menu

`↑`

Move to the previous item in the list

`Page Up`

Jump up in the menu by one screen.

`Page Down`

Jump down in the menu by one screen.

`End`

Jump to the end of the menu

`↓`

Move to the next item in the list.

`F1`

Help

`F2`

Edit

`F5`

Copy

`Ctrl` + `Insert`

An alias for `F5`

`Shift` + `Delete`
: Cut

`F6`
: Paste

`Shift` + `Insert`
: An alias for `F6`

`Insert`
: Inserts a new item.

`+`
: An alias for `Insert`

`Delete`
: Delete item at current location

`-`
: An alias for `Delete`

Any letter or number
: Jumps to the next item that has that character earliest in it's name.

`Alt` + `→`
: Snap the window size to the next smaller integer zoom level

`Alt` + `←`
: Snap the window size to the next larger smaller integer zoom level

`Ctrl` + `F`
: Find text in options

`Ctrl` + `G`
: Repeat last find

The Dialing Directory

This is the default startup screen if no BBS is specified on the command-line. At the top is the program name and version, a build date, the current output mode, and the current date and time.

With version numbers, trailing letters indicate pre-release versions. 'a' indicates an alpha build which will have known bugs and/or incomplete features. 'b' indicates a beta build which indicates there are expected to be no known bugs, but it has not received testing. "rcX" is a release candidate where X is a number. These indicate that after some period (usually one to two weeks) of no newly reported bugs, a release will be made.

The output mode is important to make note of when reporting issues, since many bugs only impact

one or two output modes. It's after the build date.

The bottom-most line contains help on the current menu, indicating what options are available in the most recent menu.

There are three areas the user can interact with in the dialing directory. The Directory menu, the SyncTERM Settings menu, and the comment line.

The comment line is directly above the help line at the bottom, and allows a per-BBS comment to be entered.

The Directory

This menu lists all the entries in the two dialing directory files. If you move the bar over one and press <Enter>, it will connect you to the highlighted system as configured in the entry.

In addition to the standard controls, this menu also has some extra keyboard shortcuts.

`Ctrl` + `D`
: Quick-connect to a URL

`Ctrl` + `E`
: Edit the selected entry (Alias for `F2`)

`Ctrl` + `S`
: Modify the sort order

`Alt` + `B`
: View the scrollback of the last session

`Tab`
: Move to the comment field for the current entry, or the settings menu if there is no current entry.

`Back Tab` (`Shift` + `Tab`)
: Move to SyncTERM Settings.

To add a new entry, go to the bottom of the list (by pressing end) and select the blank entry at the bottom. A window will pop up asking for the Name of the entry. This name must not already exist in the personal dialing directory.

Next you will be prompted for the protocol to use. Options include:

RLogin
: Uses the historic RFC1282 RLogin protocol without OOB data. This is an obsolete, unencrypted protocol that can allow auto-login, and is 8-bit clean (unlike telnet). It is very simple. Instead of the local username, the users password is sent.

RLogin Reversed

Some RLogin servers that support password auto-login have reversed the remote and local username fields. This allows connecting to these servers.

Telnet

Uses the historic and highly complex telnet protocol. This is an obsolete, unencrypted protocol and is not 8-bit clean and predates TCP/IP. It has been the source of many security vulnerabilities over the fifty years or so it has existed. Historically, it has been the most common way to connect to a remote system as a terminal, so is widely supported.

Raw

A raw 8-bit clean TCP connection. This is often what retro BBSs actually support when they say they support telnet.

SSH

The Secure Shell v2 protocol. This is the modern replacement for both telnet and rlogin, and is widely supported. This is encrypted and performs user and server authentication as part of the protocol instead of inline. SyncTERM supports authenticating with both a password and a public key.

SSH (no auth)

The SSH protocol, but will not send a password or public key. Used for auto-login systems where the user name by itself is sufficient.

Modem

SyncTERM can directly control a modem for making outgoing calls.

Serial

Direct communication with a serial port.

3-wire

Serial, but with only transmit and receive. In this mode, there is no way to detect if the remote has hung up and there is no flow control, so bytes can easily be lost. This is primarily used for communicating with embedded hardware, and not BBSs.

Shell

Runs a shell in the terminal.

MBBS GHost

The MajorBBS 'GHost' protocol.

TelnetS

Telnet over TLS. All the drawbacks of the telnet protocol, but at least it’s encrypted.

Finally, you will be promted for the "address". This is the DNS addres, IP address, serial port, or command to connect to. If the connection will be made over the network, and the name is a valid hostname, it will be auto-filled in this field. To overwrite it, simply start typeing.

Once these three pieces of information are entered, the entry is created and you are returned to the Directory. To further modify the settings, you can press `F2` to enter the Edit Directory Entry menu.

Edit Directory Entry

In this menu, you can modify all the connection settings for an entry. The exact contents of this menu will vary a bit by connection type, but most of the options are the same or silimar.

Name

The name of the entry.

Phone Number (Modem only)

The phone number to dial.

Device Name (Serial and 3-wire only)

The device name to open.

Command (Shell only)

The command to run (usually a shell such as /bin/sh)

Address

IP address or host name

Connection Type

Protocol to use. See [protocols]. When you change the protocol, the port number value will be updated as well.

Flow Control (Modem, Serial, and 3-wire)

The type of flow control to use. RTC/CTS, XON/XOFF, Both, or None

TCP Port

The TCP port to connect to.

SSH Username (SSH no auth only)

The username to send for the SSH protocol. Some BBSs have everyone log in to SSH with the same username, then log into the BBS with their name. This allows setting the first username.

BBS Username (SSH no auth only)

The username to send when `Alt` + `L` is entered.

BBS Password (SSH no auth only)

The password to send on `Alt` + `L`.

Username

The user name to send. Used by SSH, RLogin, and GHost. For other protocols, send when `Alt` + `L` is pressed.

Password

The password to send.

GHost Program (GHost)

The program name to send to the remote.

System Password

An additional password that can be sent after the first `Alt` + `L` using successive `Alt` + `L`s.

Screen Mode

Selects the format of the window when connected. A mode specifies the number of columns, the number of rows, the aspect ratio, and the font size. Some modes such as the Commodore and Atari modes will also change the selected font. If the current font is a mode-specific one (such as C64 or Atari), Changing from that mode to a standard one will change the font to Codepage 437 English. See Current Screen Mode and Custom Screen Mode for additional information.

Hide Status Line

Indicates that the "status line" at the bottom of the window when connected should not be displayed. This allows for an extra line of text from the remote to be shown.

Download Path

The location to save downloaded files.

Upload Path

The location to start at when browsing for files to upload.

Log Configuration

This brings up a sub-menu to control a debug log. There are four options in this sub-menu:

Log Filename

If this is not blank, specifies the file to write the log data to. When this is blank, disable logging.

File Transfer Log Level

May be one of None, Alerts, Critical Errors, Errors, Warnings, Notices, Normal, or Debug.

Telnet Command Log Level

Chosen from the same list as above.

Append Log File

If set you Yes, the log file retains old information and will keep growing. If set to No, the log file is emptied for each new connection.

Comm Rate

For networked modes, specifies the sharacter pacing speed. For serial and modem types, specifies the speed to open the port at.

ANSI Music

There are three options in this sub-menu.

ESC [| only

With this setting, ANSI music is fully compliant with the standards (ECMA-48, ANSI, etc), but almost no software works with this.

BANSI Style

Supports both the SyncTERM (CSI |) and BananaCom (CSI N) ANSI music styles. Support is still very rare, but slightly more common than the first.

All ANSI Music Enabled

In addition to the previous two, also supports CSI M for ANSI music. This is by far the most common sequence used by software that supports ANSI music. Unfortunately, this prevents the ANSI Delete Line sequence from working correctly.

Address Family

Selected IP address family for network connections.

As per DNS

Uses the first address returned by getaddrinfo()

IPv4 only

Will only connect over an IPv4 address. If none is available, the connection will fail.

IPv6 only

Will only connect over an IPv6 address. If none is available, the connection will fail.

Font

Choses a font (and by implication, a codepage) for the connection. Custom fonts are also listed in this menu.

Hide Popups

Do not show status and progress popups.

RIP

Selects the version for Remote Imaging Protocol ("RIP"). RIP allows graphics and mouse usage, and was used by doors and BBSs starting in the early 90s. The RIP support in SyncTERM is not complete, and may not be compatible with other terminals. RIPv1 is the one most commonly used by old BBS software, and it requires that the Screen Mode be set to an EGA mode. RIPv3 is an updated version that is not backward compatible, but can be used in any mode.

Force LCF Mode

This setting will force the DEC terminal "Last Column Flag" mode to always be enabled. This mode is almost always used in modern terminal emulators, which are almost all VT-102 emulators at least. LCF controls the wrapping behaviour when the cursor is on the last column of a line. The specific rules used are complex and not implemented the same in all terminal emulators even today.

Yellow is Yellow

By default, SyncTERM displays low-intensity yellow as brown. This originated in the IBM CGA monitors, and was carried forward to EGA and even most VGA modes. Some digital monitors that were CGA compatible did not have the brown hack. While the vast majority of software will assume that low-intensity yellow should be brown, this allows strict standard compliance.

SFTP Public Key

For SSH connections, SyncTERM can open another SSH channel and write the public key to .ssh/authorized_keys on the remote, which will enable authentication using the private key on at least OpenSSH and new versions of Synchronet. This option requires SFTP support from the remote side, and may cause connection stability issues if SFTP is not available or does not work correctly.

Modifying the Sort Order

You can change the order that entries appear in the Directory via the `Ctrl` + `S` key. This brings up a menu where you can add entries by either pressing Insert to add before the current item, or by pressing Return on the enpty line at the end. This will bring up a list of fields that can be sorted on.

Pressing enter on a field will toggle if it's reversed (highest to lowest) or not.

Viewing the Scrollback

When viewing the scrollback, the following keys are supported:

`↑`
: Move up one line

`J`
: Move up one line

`↓`
: Move down on line

`K`
: Move down one line

`Page Up`
: Move up one screen

`H`
: Move up one screen

`Page Down`
: Move down on screen

`L`
: Move down one screen

`Escape`
: Exit scrollback mode

SyncTERM Settings

The SyncTERM Settings menu has the following options:

Default Connection Settings
: Set the default values for a new directory entry. See Edit Directory Entry for details on these options.

Current Screen Mode
: Changes the current screen mode. For directory entries where the screen mode is "Current", will be used during the connection. This setting is not saved across program restarts. To change the startup screen mode, see SyncTERM Settings → Program Settings → Startup Screen Mode.

Font Management
: Allows setting up custom font files.

Program Settings
: Allows changing settings that are preserved across reboots. Refer to the Program Settings section for details.

File Locations
: Shows the paths to the various files and directories that SyncTERM will access.

Build Options
: Shows which optional components SyncTERM was built to support.

Current Screen Mode

This temporarily sets the screen mode. A screen mode defines the number of rows and columns in the window, which font size to use (8x8, 8x14, or 8x16), and the aspect ratio to scale pixels to. The majority of these modes are based on historical analog hardware modes, so most of them do not use square pixels. The main exceptions are LCD80x25, which is an 80x25 mode that uses square pixels and the 8x16 font, and VGA80x25, which is an 80x25 mode that uses square pixels, the 8x16 font, and performs the VGA column expansion to use 9 pixel wide cells. For further details, see the Text Modes section of the ciolib chapter.

Font Management

The Font Management menu allows you to add and remove fonts. Each font should have a unique name, and at least one file for 8x8, 8x14, or 8x16 fonts. The font format is the one used by "DOS fonts". You can insert and delete items using normal UIFC commands.

Program Settings

Confirm Program Exit

Asks if you are sure you want to quit when pressing ESC or right-clicking in the main SyncTERM screen.

Prompt To Save

If enabled, when SyncTERM is started with a URI that is not in a dialing directory, asks if you want to add the entry to the directory.

Startup Screen Mode

The screen mode that is used when SyncTERM starts.

Video Output Mode

The method of displaying SyncTERM output. The options will vary by OS, compile-time options, and installed libraries. See the -I option to SyncTERM for details on the various modes.

Scrollback Buffer Lines

The maximum number of lines to keep in the scrollback. Once this number is reached, the oldest lines are removed to make room for new lines.

Modem/Comm Device

The device name for the Modem device. For UNIX-like systems, this will be something like "/dev/ttyd0". For Windows, this will be "COM1" to "COM9". If it's COM10 or higher, it needs to be specified as "\\.\COM10".

Modem/Comm Rate

Specifies the speed to communicate with the modem at. If set to 0, the speed is not set by SyncTERM, and the default is used.

Modem Init String

The string to send to the modem when the device is first opened to prepare it to be used.

Modem Dial String

A string that is sent immediately before the phone number to cause the modem to dial.

List Path

The path to load the personal dialing directory from.

TERM For Shell

The value that the TERM variable is set to for shell type connections.

Scaling

Select to cycle through the values "Blocky", "Pointy", and "External". Blocky scales each pixel to a rectangle, and Pointy will use 45° angles. External will use hardware scaling if possible. The quality of External scaling varies wildly based on OS, device drivers, and output mode.

Custom Screen Mode

Allows defining the Rows, Columns, Font Size, and Aspect Ratio of the "Custom" screen mode.

Connected State

When you are connected to a system, there are a number of controls available that are not sent to the remote.

For Curses and ANSI modes, only two controls are avilable, and they are not available in other modes. This is because these two modes don't have access to keys that could not potentially be sent to the remote. To help avoid conflict with remote systems, the XON (`Ctrl` + `Q`) and XOFF (`Ctrl` + `S`) codes that are used for software flow control are used.

`Ctrl` + `Q`
: Disconnects from the current session.

`Ctrl` + `S`
: Brings up the Online Menu (see below)

For all other modes, the ALT key is used for SyncTERM commands, and the following combinations are supported:

`Shift` + `Insert`
: Pastes the current PRIMARY selection or clipboard contents to the remote.

`Alt` + `B`
: View scrollback (See "Viewing The Scrollback")

`Alt` + `C`
: Capture Control. Allows starting and stopping capturing the session to a file. Useful for stealing ANSIs and debugging emulation issues.

`Alt` + `D`
: Begins a download from the remote system.

`Alt` + `E`
: Brings up the Dialing Directory

`Alt` + `F`
: Allows selecting a different font. In some output modes, the selected font will change text that is already on the screen. In most modes however, only newly displayed text will be in the new font.

`Alt` + `H`
: Hangup and return to the main menu.

`Alt` + `L`
: Send auto-login information. For protocols that allow auto- login, only the system password is sent. For all others, the username is sent followed by a carriage return, then the password followed by a CR, then the system password followed by the CR. If any of these are not configured for the current entry, neither them, nor the CR are sent for that item.

`Alt` + `M`

Changes the currently supported "ANSI" Music prefix.

`Alt` + `O`

Toggles remote mouse support. With the remote capturing mouse events, it can be difficult to select text to copy. `Alt` + `O` allows taking the mouse away from the remote.

`Alt` + `U`

Upload a file to the remote.

`Alt` + `X`

Disconnect and exit SyncTERM. Does not return to the main menu.

`Alt` + `Z`

Brings up the Online Menu (see below)

`Alt` + `]`

Selects the next fastest character pacing speed. If the fastest speed is currently selected, disables character pacing. If pacing is currently disabled, selects the slowest pacing speed.

`Alt` + `[`

Selects the next slowest character pacing speed. If the slowest speed is currently selected, disables character pacing. If pacing is currently disabled, selects the fastest pacing speed.

Online Menu

Allows menu-based selection of some of the above options, as well as some less-common operations that don't have a keyboard shortcut.

Scrollback

Same as `Alt` + `B`

Disconnect

Same as `Alt` + `H`

Send Login

Same as `Alt` + `L`

Upload

Same as `Alt` + `U`

Download

Same as `Alt` + `D`

Change Output Rate

Allows selecting a specific character pacing to use.

Change Log Level

Temporarily changes the file transfer log level

Capture Control

Same as `Alt` + `C`

ANSI Music Control

Same as `Alt` + `M`

Font Setup

Same as `Alt` + `F`

Toggle Doorway Mode

Turns on or off Doorway mode without the host specifying it. Can be used to recover from broken remote software.

Toggle Remote Mouse

Same as `Alt` + `O`

Toggle Operation Overkill][Mode

Turns on or off OO][mode. Can be used to recover from broken remote software.

Exit

Same as `Alt` + `X`

Edit Dialing Directory

Same as `Alt` + `E`

Techical Details

The following sections delve deeper into technical details, and should not be required for normal use.

Persistant State

SyncTERM will preserve a very small amount of state when exited normally and restore it when the program is restarted. At present, this state is limited to the scaling factor applied to the window. The scaling factor is a floating-point value that indicates the largest value which both the width and height can be multiplied by and still fit inside the current window size. If you manually resize the window only making it wider for example, the additional width will not be saved as the height will control the size the next time SyncTERM is started.

This state is only saved under two specific circumstances. Either the current text mode at exit is the same as the configured Startup Screen Mode, or the Startup Screen Mode is "Current" and the current text mode is "80x25". In all other cases (such as after the Current Screen Mode is changed), it will not be saved. Also, it’s only saved when the program exits normally. In the case of a crash, the setting will not be updated.

Installing Terminfo Entry

When using *nix software through SyncTERM either locally via the shell protocol, or remotely via SSH or as a door on a BBS, it helps immensely if the remote has a terminfo entry for SyncTERM installed. To install the terminfo entry, follow the following steps:

1. Write the terminfo entry to a file
   `syncterm -T > syncterm.terminfo`
2. Compile the entry and install it
   `tic -sx syncterm.terminfo`
   By default, it is only installed for the current user.

Once the terminfo entries are installed, you can use them by setting the `TERM` environment variable to `syncterm`. In Bourne shells, this is usually accomplished with the command `export TERM=syncterm`.

MBBS GHost

"GHost" in SyncTERM refers to the "Galacticomm Host Program" (called Ghost) that was included in Major BBS and Worldgroup (MBBS/WG) that allowed a Sysop to connect another (DOS-based) PC to the BBS by use of a null modem cable. This was a way for a MBBS/WG Sysop to offer DOS doors, something that wasn't normally possible.

The functions of the Ghost software itself are beyond the scope of SyncTERM, and you should consult the MBBS/WG Ghost documentation for operation details. However, broadly speaking, it worked like this:

1) MBBS/WG would send a signal down the null modem cable to alert the DOS PC (running Ghost) that it wanted to run a door. 2) Using a simple protocol, MBBS/WG would transmit information required to run the door (username, time remaining, whether ANSI-BBS graphics were supported, etc) to Ghost. 3) Ghost would then launch the door in DOS, using a batch file to call Ghost back once the door exited to wait for the next request.

While few people are connecting DOS-based PC's to anything by null modem cables anymore, the Ghost protocol (as offered in SyncTERM) is still useful because it's a way to run DOS doors inside a virtual machine and expose them outside of that virtual machine. The idea being that the VM would configure a serial port as some kind of network passthrough, so when SyncTERM connects, it's passed through to the VM and then Ghost.

One use case for this is to offer DOS doors in environments where it would normally be difficult or impossible. For example, a UNIX user could run SyncTERM on a remote system in curses mode, where it would then connect to a VM and launch a DOS door via Ghost. This would all be presented to the end UNIX user in a seamless way, so all they would see is the door startup.

The Ghost protocol consists of a single line starting with 'MBBS:', terminated with \r\n, and contains five parameters:

MBBS: PROGRAM PROTOCOL 'USER' TIME GR

You don't need to worry about sending this since SyncTERM will format it for you based on the

SyncTERM configuration options. But it is helpful to understand how various SyncTERM options will translate to the Ghost protocol parameters:

PROGRAM: The name of the DOS door/software to ask the Ghost side to run. Configured in SyncTERM in the 'GHost Program' field of a directory entry, or after the final slash in a ghost:// style URL. For example: ghost://user@203.0.113.64/program

PROTOCOL: Always set to 2. Not configurable in SyncTERM.

USER: Username of the person connecting. Configured in SyncTERM in the 'username' field of a directory entry, or before the '@' in a ghost:// style URL. For example: ghost://user@203.0.113.64/program

TIME: Amount of time the user has remaining. Always set to 999. Not configurable in SyncTERM.

GR: Set to GR (for "GRaphics", meaning ANSI-BBS support) or NG (for "No Graphics"). Always set to GR. Not configurable in SyncTERM.

CTerm Manual

CTerm terminal characteristics

End of line behaviour (wrapping):

The cursor is moved to the first character of the next line as soon as a character is written to the last column of the current line, not on the next character. A tab will wrap to the next line only if the current cursor position is the last character on the line. This behavior is often surprising to people who are used to VT emulators which implement the LCF as documented in [STD-070], who expect the cursor to "stick" in the last column until the next character is received.

There are two settable flags that will impact the default behaviour.

`CSI ? 7 l` will disable wrapping at the end of line completely, and any characters written to the last column will not move the cursor at all, overwriting the existing charater. Default behaviour can be restored with `CSI ? 7 h`.

If the `CSI = 4` h sequence is received, CTerm will enable LCF mode as documented in [STD-070], and `CSI = 4 l` will restore default behaviour. `CSI = 5 h` will set LCF mode and disable `CSI = 4 l`, as well as cause LCF to remain enabled across an `ESC c` (RIS).

Specifically, the LCF will be set when displaying a printable character advances the cursor to the right margin, and cleared by any of the following being received: `CSI ? 6 h`, `CSI ? 6 l`, `CSI ? 7 l`, `CSI @`, `CSI A`, `CSI B`, `CSI a CSI j`, `CSI H`, `CSI f`, `CSI I`, `CSI Y`, `CSI J`, `CSI K`, `CSI P CSI X`, `CSI r`, `ESC E`, `ESC M`, `CR`, `LF`, `BS`, `TAB` Any normal printable character when the cursor is at the right margin (of the screen or scrollable area).

Control characters

`0x00` NUL (`NUL`)

In doorway mode, indicates that the next character is a literal character. The IBM CP437 character will be displayed. This allows ESC and other control characters to be placed on the screen.

SOURCE: [BANSI]

`0x07` Bell (`BEL`)

Beep

`0x08` Backspace (`BS`)

Non-destructive backspace. Moves cursor position to the previous column unless the current column is the first, in which case no operation is performed.

SOURCE: [ECMA-48]

`0x09` Horizontal Tab (`HT`)

Moves to the next horizontal tab stop. Does not overwrite any characters in between. If there are no tab stops left in the line, moves to the first position of the next line. If the starting position is on the last line, will perform a scroll, filling the new line at bottom with the current attribute.

SOURCE: [ECMA-48]

`0x0A` Line Feed (`LF`)

Move cursor position to same column of the next row. If current row is the last row, scrolls the screen up and fills the new row with the current attribute.

SOURCE: [ECMA-48]

`0x0D` Carriage Return (`CR`)

Move cursor position to column 1 of the current line

SOURCE: [ECMA-48]

`0x1B` Escape (`ESC`)

Introduces a control code. The `ESC` and the next byte together form the control code. If the control code is not valid, the `ESC` is ignored.

SOURCE: [ECMA-48]

Control Codes

Control codes are in the following format:
`ESC {'0'` to `'~'}` Legal combinations which are not handled are silently dropped.

`ESC E` Next Line (`NEL`)

Moves to the line home position of the next line. (Same as `CR LF`)

SOURCE: [ECMA-48]

`ESC H` Set Tab (`HTS`)

Sets a tab stop at the current column

SOURCE: [ECMA-48]

`ESC M` Reverse Line Feed (`RI`)

Move up one line

SOURCE: [ECMA-48]

`ESC P`Device Control String (`DCS`)

Begins a string consisting of the characters 0x08 - 0x0d and 0x20-0x7e, terminated by a String Terminator (`ST`)

SOURCE: [ECMA-48]

Supported `DCS` string values

`CTerm:Font:p1:<b64>` CTerm Loadable Font (`CTLF`)

Indicates the string is a loadable font. (CTerm 1.213) + `p1` is a font slot number, which must be higher than the last default defined font (See `CSI sp D` for list of predefined fonts). `<b64>` is the base64 encoded font data. Font size is deduced from the size of the data. This replaces the now deprecated `CSI = Ps1 ; Ps2 {`

`[ p1 [ ; p2 ] ] q` Sixel Sequence

Defaults: `p1` = 0 `p2` = 0 Indicates the string is a sixel sequence.

`p1` selects the vertical height of a single pixel. This may be overridden by the raster attributes command, and is deprecated. Supported values

Table 1. Supported Values of `p1`

Value	Vertical Size
0,1,5,6	2 pixels
2	5 pixels
3,4	3 pixels
7,8,9	1 pixel

`p2` indicates if unset sixels should be set to the current background colour. If p2 is 1, positions specified as 0 remain at their current colour.

Any additional parameters are ignored.

The rest of the string is made up of sixel data characters and sixel control functions. Sixel data characters are in the range of `?` (0x3f) to `~` (0x7e). Each sixel data character represents six vertical pixels. The data is extracted by subtracting 0x3f from the ASCII value of the character. The least significant bit is the topmost pixel.

Sixel Control Functions

`! Pn X` **Graphics Repeat Introducer**

The character X is repeated Pn times.

`" p1 ; p2 [ ; p3 [ ; p4 ] ]` **Raster Attributes**

p1 indicates the vertical size in pixels of each sixel. p2 indicates the horizontal size in pixels. p3 and p4 define the height and width (in sixels) respectively of a block to fill with the background colour. This block may not extend past the current bottom of the screen. If any pixel data characters proceed this command, it is ignored.

`# p1` **Colour Select**

Selects the current foreground colour from the sixel palette.

`# p1 ; p2 ; p3 ; p4 ; p5` **Palette map**

Defines sixel palette entry p1 and sets it as the current foreground colour. p2 specifies the colour space to define the colour in, the only supported value is 2. p3, p4, and p5 specify the red, green, and blue content as a percentage (0-100).

`$` **Graphics Carriage Return**

Returns the active position to the left border of the same sixel row. Generally, one pass per colour is used. In passes after the first one, sixels with a value of zero are not overwritten with the background colour.

`-` **Graphics New Line**

Moves the active position to the left border of the next sixel row.

SOURCE: [VT330/340]

`$ q pt` **Request Status String (`DECRQSS`)**

`pt` is the intermediate and/or final characters of a control function to query the status of. The terminal will send a response in the format

```
DCS p1 $ r pt ST
```

`p1` is 1 if the terminal supports querying the control function and 0 if it does not.

`pt` is the characters in the control function except the `CSI` characters.

Table 2. Currently supported values of pt:

pt	Request SGR parameters
r	Request top and bottom margins
s	Request left and right margins
t	Request height in lines
$|	Request width in columns
*|	Request height in lines

SOURCE: [VT420]

p1 [; p2 [; p3] ! z Define Macro (DECDMAC)

Defaults: p2 = 0 p3 = 0

Sets a macro to be replayed using CSI Pn * z

p1 is the macro number to set, and must be between 0 and 63 inclusive.

If p2 is zero, the macro numbered p1 will be deleted before the new macro is set. If p2 is one, all macros are deleted before the new macro is set. If the macro is zero length, only the delete action is stored, you can't store a zero-length macro.

If p3 is zero, the macro is defined using ASCII characters (0x20 - 0x7e and 0xa0 - 0xff only) if p3 is one, the macro is defined using hex pairs.

When the macro is defined using hex pairs, a repeat sequence may be included in the format of ! Pn ; D..D ; Pn specifies the number of repeats (default of one instance)+ D..D is the sequence of pairs to send Pn times. The terminating ; may be left out if the sequence to be repeated ends at the end of the string.

SOURCE: [VT420]

ESC X Start Of String (SOS)

As the above strings, but may contain any characters except a Start Of String sequence or a String Terminator sequence. The string is currently ignored.

SOURCE: [ECMA-48]

ESC \ String Terminator (ST)

Ends a string.

SOURCE: [ECMA-48]

ESC] Operating System Command (OSC)

Begins a string consisting of the characters 0x08 - 0x0d and 0x20-0x7e, terminated by a String Terminator (ST)

+ .Supported OSC values 4;(pX;pY)⋯:: Specifies one or more palette redefinitions.

pX is the palette index, and pY is the colour definition
Color format: rgb:R/G/B:::: Where R, G, and B are a sequence of one to four hex digits representing the value of the red, green, and blue channels respectively.

+ SOURCE: [XTerm]

104 [; Ps ⋯]

Resets palette entry to default. If the entire string is "104" (ie: no Ps present), resets all colours. Otherwise, only each index separated by a semicolon is reset.

SOURCE: [XTerm]

ESC ^ Privacy Message (PM)

Begins a string consisting of the characters 0x08 - 0x0d and 0x20-0x7e, terminated by a String Terminator (ST) The string is currently ignored.

SOURCE: [ECMA-48]

ESC _ Application Program Command (APC)

Begins a string consisting of the characters 0x08 - 0x0d and 0x20-0x7e, terminated by a String Terminator (ST)

SOURCE: [ECMA-48]

SyncTERM implements the following APC commands:

SyncTERM:C;S Ps1 Ps2 Store file (CTSFI)

Where Ps1 is a filename and Ps2 is the base64 encoded contents of the file. The named file is stored in the cache directory for the current connection.

SyncTERM:C;L [; Ps] List Files (CTLFI)

Defaults: Ps = *
Ps is the glob(3) pattern to use matching files. SyncTERM responds with an APC string with lines separated by newlines. The first line is always SyncTERM:C;L\n and for each matching file, a line in the form <Filename> TAB <MD5 sum> LF is sent (ie: "coolfont.fnt\t595f44fec1e92a71d3e9e77456ba80d1\n")

SyncTERM:C;SetFont; Pn ; Ps Set Font (CTSF)

Where Pn is a font slot number (max 255) and Ps is a filename in the cache. This sets font slot Pn to use the specified font file.

SyncTERM:C;DrawPPM Ps⋯ Ps1 Draw a PPM from Cache (CTDPFC)

Draws a PPM from the cache directory on the screen. Ps1 is the filename and is required. Arguments for Ps are optional. The following options can be included (separated by semi-colons):

SX=#

Sets the left X position in the specified image to copy from. Default = 0.

`SY=#`

Sets the top Y position in the specified image to copy from. Default = 0.

`SW=#`

Sets the width of the portion of the image to copy. Default = Image width - `SX`

`SH=#`

Sets the height of the portion of the image to copy. Default = Image height - `SH`

`DX=#`

Sets the X position on the screen to draw the image at. Default = 0.

`DY=#`

Sets the Y position on the screen to draw the image at. Default = 0.

`MX=#`

Sets the X position in the mask to start applying from. Default = 0.

`MY=#`

Sets the Y position in the mask to start applying from. Default = 0.

`MW=#`

Sets the overall width of the mask (not the width to apply). If `MFILE` is not specified, and a mask is (ie: using `MASK=`), this is required. If `MFILE` is specified, the width is read from the file.

`MH=#`

Sets the overall height of the mask (not the height to apply). If MFILE is not specified, and a mask is (ie: using MASK=), this is required. If MFILE is specified, the width is read from the file.

`MFILE=<filename>`

Specifies a filename in the cache directory of a PBM file specifying a mask of which pixels to copy. Any pixel set to black (ie: 1) in the PBM will be drawn from the source image. Pixels set to white (ie: 0) will be left untouched.

`MASK=<maskbits>`

Specifies a base64-encoded bitmap, each set bit will be drawn from the source image, cleared bits will not be drawn. Requires MW= and MH= to be specified.

`MBUF`

Uses the loaded mask buffer.

The PPM file may be raw (preferred) or text. SyncTERM does not support more than 255 values per colour channel and assumes it is correctly using the BT.709 gamma transfer.

`SyncTERM:C;LoadPPM Ps… Ps0` Load a PPM to Buffer (`CTLPTB`)

Loads a PPM to a buffer. Ps0 is the filename

`B=#`

Selects the buffer (0 or 1 only) to paste from.

`SyncTERM:C;LoadPBM Ps··· Ps0` **Load a PBM to Buffer (`CTLPBTB`)**

Loads a PBM to a buffer. Ps0 is the filename

`SyncTERM:P;Copy Ps···` **Copy Screen into Buffer (`CTCSIB`)**

Copies a portion of the screen into an internal buffer for use with the Paste function. Defaults to copying the entire screen.

`B=#`

Selects the buffer (0 or 1 only) to copy to.

`X=#`

Sets the left X position on the screen to start copying at. Default = 0.

`Y=#`

Sets the top Y position on the screen to start copying at. Default = 0.

`W=#`

Sets the width to copy. Default = Screen width - X.

`H=#`

Sets the height to copy. Default = Screen height - X.

`SyncTERM:P,Paste Ps···` **Paste Buffer to Screen (`CTPBTS`)**

Pastes from the copied buffer. Supports the same options as the Cache DrawPPM command except for the filename, and adds the B= option.

`B=#`

Selects the buffer (0 or 1 only) to paste from.

`ESC c` **Reset to Initial State (`RIS`)**

Resets all the terminal settings, clears the screen, and homes the cursor.

SOURCE: [ECMA-48]

Control Sequences:

Control sequences start with the Control Sequence Introducer which is `ESC [`. `CSI` will be used to express this from now on.

Control sequences are in the following format:
`CSI {'0' (ZERO) to '?'}{SPACE to '/'}{'@' to '~'}`
There may be multiple characters from the {'0' (ZERO) to '?'} and {SPACE to '/'} before the terminating {'@' to '~'} character.

Legal combinations not handled are silently dropped. Illegal combinations are displayed.

Sequence Parameters

Parameters are expressed by the {'0' (ZERO) to '?'} character set.

Sequences which use parameters use decimal parameters separated by a ';'. The use of a ':' from the set is reserved.

If the parameter string begins with '<', '=', '>', or '?' then this is a non-standard extension to the ANSI spec.

Table 3. Sequence Paramters

`Pn`	Indicates a single numeric parameter
`Pn1 ; Pn2`	Two numeric parameters
`Pn…`	Any number of numeric parameters
`Ps`	Single selective parameter
`Ps1 ; Ps1`	Two selective parameters
`Ps…`	Any numer of selective parameters

If a default is defined, the parameter is optional

`CSI Pn @` Insert Character(s) (`ICH`)

Defaults: `Pn` = 1
Moves text from the current position to the right edge Pn characters to the right, with rightmost characters going off-screen and the resulting hole being filled with the current attribute.

SOURCE: [ECMA-48]

`CSI Pn SP @` Scroll Left (`SL`)

Defaults: `Pn` = 1
Shifts the contents of the screen left `Pn` columns(s) with leftmost columns going off-screen and the resulting hole being filled with the current attribute.

SOURCE: [ECMA-48]

`CSI Pn A` Cursor Up (`CUU`)

Defaults: `Pn` = 1
Moves the cursor position up `Pn` lines from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Pn SP A` Scroll Right (`SR`)

Defaults: `Pn` = 1
Shifts the contents of the screen right `Pn` columns(s) with rightmost columns going off-screen and the resulting hole being filled with the current attribute.

SOURCE: [ECMA-48]

`CSI Pn B` Cursor Down (`CUD`)

Defaults: `Pn` = 1
Moves the cursor position down `Pn` lines from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Pn C` Cursor Right (`CUF`)

Defaults: `Pn` = 1 Moves the cursor position right `Pn` columns from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Pn D` Cursor Left (`CUB`)

Defaults: `Pn` = 1 Moves the cursor position left `Pn` columns from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Ps1 ; Ps2 sp D` Font Selection (`FNT`)

Defaults: `Ps1` = 0 Ps2 = 0 "sp" indicates a single space character. Sets font `Ps1` to be the one indicated by `Ps2`. Currently four fonts are supported. `Ps2` must be between 0 and 255. Not all output types support font selection. Only X11 and SDL currently do.

Table 4. Supported `Ps1` values

0	Default font
1	Font selected by the high intensity bit when `CSI ? 31 h` is enabled
2	Font selected by the blink intensity bit when `CSI ? 34 h` is enabled
3	Font selected by both the high intensity and blink bits when both `CSI ? 31 h` and `CSI ? 34 h` are enabled

Table 5. Currently included fonts

0	Codepage 437 English
1	Codepage 1251 Cyrillic, (swiss)
2	Russian koi8-r
3	ISO-8859-2 Central European
4	ISO-8859-4 Baltic wide (VGA 9bit mapped)
5	Codepage 866 (c) Russian
6	ISO-8859-9 Turkish

7	haik8 codepage (use only with armscii8 screenmap)
8	ISO-8859-8 Hebrew
9	Ukrainian font koi8-u
10	ISO-8859-15 West European, (thin)
11	ISO-8859-4 Baltic (VGA 9bit mapped)
12	Russian koi8-r (b)
13	ISO-8859-4 Baltic wide
14	ISO-8859-5 Cyrillic
15	ARMSCII-8 Character set
16	ISO-8859-15 West European
17	Codepage 850 Multilingual Latin I, (thin)
18	Codepage 850 Multilingual Latin I
19	Codepage 885 Norwegian, (thin)
20	Codepage 1251 Cyrillic
21	ISO-8859-7 Greek
22	Russian koi8-r (c)
23	ISO-8859-4 Baltic
24	ISO-8859-1 West European
25	Codepage 866 Russian
26	Codepage 437 English, (thin)
27	Codepage 866 (b) Russian
28	Codepage 885 Norwegian
29	Ukrainian font cp866u
30	ISO-8859-1 West European, (thin)
31	Codepage 1131 Belarusian, (swiss)
32	Commodore 64 (UPPER)
33	Commodore 64 (Lower)
34	Commodore 128 (UPPER)
35	Commodore 128 (Lower)
36	Atari
37	P0T NOoDLE (Amiga)
38	mO’sOul (Amiga)
39	MicroKnight Plus (Amiga)
40	Topaz Plus (Amiga)

41	MicroKnight (Amiga)
42	Topaz (Amiga)

Not all fonts are supported in all modes. If a font is not supported in the current mode, no action is taken, but there should be a non-zero 'Font Selection result' value in the Font State Report.

SOURCE: [ECMA-48]

`CSI Pn E` Cursor Next Line (`CNL`)

Defaults: `Pn` = 1
Moves the cursor to the first column of the line `Pn` down from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Pn F` Cursor Preceding Line (`CPL`)

Defaults: `Pn` = 1
Moves the cursor to the first column of the row `Pn` up from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Pn G` Cursor Character Absolute (`CHA`)

Defaults: `Pn` = 1
Movies the cursor to column Pn of the current row.

SOURCE: [ECMA-48]

`CSI Pn1 ; Pn2 H` Cursor Position (`CUP`)

Defaults: `Pn1` = 1 `Pn2` = 1
Moves the cursor to the `Pn2`th column of the `Pn1`th line.

SOURCE: [ECMA-48]

`CSI Pn I` Cursor Forward Tabulation (`CHT`)

Defaults: `Pn` = 1
Move the cursor to the Pn-th next tab stop. Basically the same as sending TAB Pn times.

SOURCE: [ECMA-48]

`CSI Ps J` Erase in Page (`ED`)

Defaults: `Ps` = 0
Erases from the current screen according to the value of `Ps`

0	Erase from the current position to the end of the screen.
1	Erase from the current position to the start of the screen.
2	Erase entire screen. As a violation of ECMA-048, also moves the cursor to position 1/1 as a number of BBS programs assume this behaviour.

Erased characters are set to the current attribute.

SOURCE: [ECMA-48], [BANSI]

`CSI Ps K` Erase in Line (`EL`)

Defaults: `Ps` = 0
Erases from the current line according to the value pf `Ps`

0	Erase from the current position to the end of the line.
1	Erase from the current position to the start of the line.
2	Erase entire line.

Erased characters are set to the current attribute.

SOURCE: [ECMA-48]

`CSI Pn L` Insert Line(s) (`IL`)

Defaults: `Pn` = 1
Inserts `Pn` lines at the current line position. The current line and those after it are scrolled down and the new empty lines are filled with the current attribute. If the cursor is not currently inside the scrolling margins, has no effect.

SOURCE: [ECMA-48]

`CSI Pn M` Delete Line(s) / "ANSI" Music (`DL`)

Defaults: `Pn` = 1 Deletes the current line and the `Pn` - 1 lines after it scrolling the first non-deleted line up to the current line and filling the newly empty lines at the end of the screen with the current attribute. If the cursor is not currently inside the scrolling margins, has no effect. If "ANSI" Music is fully enabled (CSI = 2 M), and no parameter is specified, performs "ANSI" music instead. See "ANSI" MUSIC section for more details.

SOURCE: [ECMA-48], [BANSI]

`CSI = Ps M` CTerm Set ANSI Music (`CTSAM`)

NON-STANDARD EXTENSION.
Defaults: `Ps` = 0
Sets the current state of ANSI music parsing. 0 - Only `CSI |` will introduce an ANSI music string. 1 - Both `CSI |` and `CSI N` will introduce an ANSI music string. 2 - `CSI |`, `CSI N`, and `CSI M` will all introduce an ANSI music string. In this mode, Delete Line will not be available.

`CSI N` BananaCom ANSI Music (`BCAM`)

"ANSI" Music / Not implemented. If "ANSI" Music is set to BananaCom (`CSI = 1 M`) or fully enabled (`CSI = 2 M`) performs "ANSI" music. See "ANSI" MUSIC section for more details.

SOURCE: [BANSI]

`CSI Pn P` Delete Character (`DCH`)

Defaults: `Pn` = 1
Deletes the character at the current position by shifting all characters from the current column + `Pn` left to the current column. Opened blanks at the end of the line are filled with the current attribute. If the cursor is not currently inside the scrolling margins, has no effect.

SOURCE: [ECMA-48]

`CSI Pn S` Scroll Up (`SU`)

Defaults: `Pn` = 1
Scrolls the screen up `Pn` lines. New lines emptied at the bottom are filled with the current attribute.

SOURCE: [ECMA-48]

`CSI ? Ps1 ; Ps2 S` XTerm Set or Request Graphics Attribute (`XTSRGA`)

If `Ps1` is 2, and `Ps2` is 1, replies with the graphics screen information in the following format: `CSI ? 2 ; 0 ; Px ; Py S` Where `Px` is the width of the screen in pixels and `Py` is the height.

SOURCE: [XTerm]

`CSI Pn T` Scroll Down (`SD`)

Defaults: `Pn` = 1
Scrolls all text on the screen down `Pn` lines. New lines emptied at the top are filled with the current attribute.

SOURCE: [ECMA-48]

`CSI Pn X` Erase Character (`ECH`)

Defaults: `Pn` = 1
Erase `p1` characters starting at the current character. Will not erase past the end of line. Erased characters are set to the current attribute. This can erase across scroll margins.

SOURCE: [ECMA-48]

`CSI Pn Y` Cursor Line Tabulation (`CVT`)

Defaults: `Pn` = 1
Move the cursor to the `Pn`-th next tab stop. Basically the same as sending TAB `Pn` times.

SOURCE: [ECMA-48]

`CSI Pn Z` Cursor Backward Tabulation (`CBT`)

Defaults: `Pn` = 1
Move the cursor to the `Pn`th preceding tab stop. Will not go past the start of the line.

SOURCE: [ECMA-48]

`CSI Pn ` ` Character Position Absolute (`HPA`)

Defaults: `Pn` = 1
Move the cursor to the specified position on the current row. Will not go past the end of the line.

SOURCE: [ECMA-48]

`CSI Pn a` Cursor Position Forward (`HPR`)

Defaults: `Pn` = 1
Moves the cursor position forward `Pn` columns from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Pn b` Repeat (`REP`)

Defaults: `Pn` = 1
Repeats the previous graphic character `Pn` times. Will not repeat escape sequences.

SOURCE: [ECMA-48]

`CSI Ps c` Device Attributes (`DA`)

Defaults: `Ps` = 0
If `Ps` is 0, CTerm will reply with the sequence: `CSI = 67;84;101;114;109;pN c` `67;84;101;114;109` is the ASCII values of the "CTerm" string. `pN` is the revision ID of CTerm with dots converted to semi-colons (e.g. "1;156"). Use the revision to detect if a specific feature is available. If you are adding features to a forked version of cterm, please do so by adding an extra parameter to the end, not by incrementing any existing one!

SOURCE: [ECMA-48]

`CSI < Ps c` CTerm Device Attributes (`CTDA`)

Defaults: `Ps` = 0
If `Pn` is 0, CTerm will reply with the sequence: `CSI < 0 ; Ps··· c`

Table 6. Possible values for `Ps`

1	Loadable fonts are availabe via Device Control Strings
2	Bright Background (ie: DECSET 32) is supported
3	Palette entries may be modified via an Operating System Command string

4	Pixel operations are supported (currently, sixel and PPM graphics)
5	The current font may be selected via `CSI Ps1 ; Ps2 sp D`
6	Extended palette is available
7	Mouse is available

`CSI PN d` Line Position Absolute (`VPA`)

Defaults: `Pn` = 1
Moves to row specified by `Pn`.

SOURCE: [ECMA-48]

`CSI Pn SP d` Tab Stop Remove (`TSR`)

Defaults: None
Removes a tab stop at postion `Pn`.

SOURCE: [ECMA-48]

`CSI Pn e` Line Position Forward (`VPR`)

Defaults: `Pn` = 1
Moves forward Pn rows.

SOURCE: [ECMA-48]

`CSI Pn1 ; Pn2 f` Character and Line Position (`HVP`)

Defaults: `Pn1` = 1 `Pn2` = 1
Moves the cursor to the `Pn`2th column of the `Pn`1th line.

SOURCE: [ECMA-48]

`CSI Ps g` Tabulation Clear (`TBC`)

Defaults: `Ps` = 0
Deletes tab stops according to the values of `Ps`:

0	Deletes tab stop at current position.
3	Deletes all tab stops.
5	Deletes all tab stops.

SOURCE: [ECMA-48]

`CSI = 255 h` (`BCSET`)

NON-STANDARD EXTENSION
Enable DoorWay Mode

SOURCE: [BANSI]

`CSI = 4 h` Enable Last Column Flag (`CTELCF`)

NON-STANDARD EXTENSION
Enable Last Column Flag mode

`CSI = 5 h` Force Last Column Flag (`CTFLCF`)

NON-STANDARD EXTENSION
Force Last Column Flag mode

`CSI ? Ps··· h` Set Mode (DECSET)

NON-STANDARD EXTENSION
Sets one or more mode. The following modes are supported:

6	Enable origin mode. In this mode, position parameters are relative to the top left of the scrolling region, not the screen. Defaults to reset. SOURCE: [VT102]
7	Enable auto wrap This is the normal mode in which a write to the last column of a row will move the cursor to the start of the next line triggering a scroll if required to create a new line. Defaults to set. SOURCE: [VT102]
9	X10 compatible mouse reporting Mouse button presses will send a CSI M <button> <x> <y> Where <button> is ' ' + button number (0-based) <x> and <y> are '!' + position (0-based) SOURCE: [XTerm]
25	Display the cursor. Defaults to set. SOURCE: [VT320]
31	Enable bright alt character set With this mode set, the bright (1) graphic rendition selects characters from an alternate character set. Defaults to reset.
32	Bright Intensity Disable This makes the bright intensity bit not control the intensity. Mostly for use with `CSI ? 31 h` to permit fonts in the same colours. Defaults to reset.

33	Blink to Bright Intensity Background With this mode set, the blink (5,6) graphic renditions cause the background colour to be high intensity rather than causing blink. Defaults to reset.
34	Enable blink alt character set With this mode set, the blink (5, 6) graphic renditions selects characters from an alternate character set. Defaults to reset
35	Blink Disabled This makes the blink (5, 6) graphic renditions not cause the character to blink. Mostly for use with `CSI ? 34 h` to permit fonts to be used without blinking. Defaults to reset.
67	When set, the backspace key sends a backspace character. Defaults to set.
69	DEC Left Right Margin Mode enabled Enables `CSI s` to set the left/right margins, and disables `CSI s` from saving the current cursor position.
80	Sixel Scrolling Enabled When this is set, the sixel active position begins in the upper-left corner of the currently active text position. When the sixel active position reaches the bottom of the page, the page is scrolled up. At the end of the sixel string, a sixel newline is appended, and the current cursor position is the one in which the bottom sixel is in. Defaults to set. SOURCE: [VT330/340]
1000	Normal tracking mode mouse reporting Mouse button presses will send a CSI M <button> <x> <y> Where <button> is ' ' + button number (0-based) Mouse button releases will use a button number of 4 <x> and <y> are '!' + position (0-based) SOURCE: [XTerm]
1001	Highlight tracking mode mouse reporting (Not supported by SyncTERM) SOURCE: [XTerm]

1002	Button-event tracking mode mouse reporting Mouse button presses and movement when a button is pressed will send a CSI M <button> <x> <y> Where <button> is ' ' + button number (0-based) 32 is added to the button number for movement events. Mouse button releases will use a button number of 4 <x> and <y> are '!' + position (0-based) SOURCE: [XTerm]
1003	Any-event tracking mode mouse reporting Mouse button presses and movement will send a CSI M <button> <x> <y> Where <button> is ' ' + button number (0-based) 32 is added to the button number for movement events. Mouse button releases will use a button number of 4 <x> and <y> are '!' + position (0-based) If no button is pressed, it acts as though button 0 is. SOURCE: [XTerm]
1004	Focus-event tracking mode mouse reporting (Not supported by SyncTERM) SOURCE: [XTerm]
1005	UTF-8 encoded extended coordinates (Not supported by SyncTERM) SOURCE: [XTerm]
1006	SGR encoded extended coordinates Instead of the CSI M method, the format of mouse reporting is changed to CSI < Pb ; Px ; Py M for presses and CSI < Pb ; Px ; Py m for releases. Instead of CSI M Px and Py are one-based. Pb remains the same (32 added for movement) Button 3 is not used for release (separate code) SOURCE: [XTerm]
1007	Alternate scroll mode (Not supported by SyncTERM) SOURCE: [XTerm]
1015	URXVT encoded extended coordinates (Not supported by SyncTERM) SOURCE: [XTerm]

2004	Set bracketed paste mode SOURCE: [XTerm]

`CSI Pn j` Character Position Backward (`HPB`)

Defaults: `Pn` = 1
Moves the cursor position left `Pn` columns from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI Pn k` Line Position Backward (`VPB`)

Defaults: `Pn` = 1 Moves the cursor position up `Pn` lines from the current position. Attempting to move past the screen boundaries stops the cursor at the screen boundary.

SOURCE: [ECMA-48]

`CSI = 255 l` Disable DoorWay Mode (`BCRST`)

NON-STANDARD EXTENSION

SOURCE: [BANSI]

`CSI = 4 l` (`CTDLCF`)

NON-STANDARD EXTENSION
Disable Last Column Flag mode

`CSI ? Ps··· l` Reset Mode (`DECRST`)

NON-STANDARD EXTENSION
Resets one or more mode. The following modes are supported:

6	Origin Mode With this mode reset, position parameters are relative to the top left of the screen, not the scrolling region. Defaults to reset. SOURCE: [VT102]
7	Disable auto wrap Resetting this mode causes a write to the last column of a to leave the cursor where it was before the write occurred, overwriting anything which was previously written to the same position. SOURCE: [VT102]
9	Disable X10 compatible mouse reporting

25	Hide the cursor. Defaults to set. SOURCE: [VT320]
31	Disable bright alt character set With this mode reset, the bright (1) graphic rendition does not select an alternative font. Defaults to reset.
32	Bright Intensity Enable When reset, bright intensity graphics rendition behaves normally. Defaults to reset.
33	Disable Blink to Bright Intensity Background With this mode set, the blink (5,6) graphic renditions do not affect the background colour. Defaults to reset.
34	Disable blink alt character set With this mode reset, the blink (5, 6) graphic renditions do not select characters from an alternate character set. Defaults to reset.
35	Blink Enable With this mode reset, the blink (5,6) graphic renditions behave normally (cause the characters to blink). Defaults to reset.
67	When reset, the backspace key sends a delete character. Defaults to set.
69	DEC Left Right Margin Mode disabled Disables CSI s from setting the left/right margins, and changes it back to saving the current cursor position. The current left/right margins are maintained.
80	Sixel Scrolling Disabled When this is reset, the sixel active position begins in the upper-left corner of the page. Any commands that attempt to advance the sixel position past the bottom of the page are ignored. At the end of the sixel string, the current cursor position is unchanged from where it was when the sixel string started. Defaults to set. SOURCE: [VT330/340]
1000	Disable Normal tracking mode mouse reporting SOURCE: [XTerm]

1001	Disable Highlight tracking mode mouse reporting (Not supported by SyncTERM) SOURCE: [XTerm]
1002	Disable Button-event tracking mode mouse reporting SOURCE: [XTerm]
1003	Disable Any-event tracking mode mouse reporting SOURCE: [XTerm]
1004	Disable Focus-event tracking mode mouse reporting (Not supported by SyncTERM) SOURCE: [XTerm]
1005	Disable UTF-8 encoded extended coordinates (Not supported by SyncTERM) SOURCE: [XTerm]
1006	Disable SGR encoded extended coordinates SOURCE: [XTerm]
1007	Disable Alternate scroll mode (Not supported by SyncTERM) SOURCE: [XTerm]
1015	Disable URXVT encoded extended coordinates (Not supported by SyncTERM) SOURCE: [XTerm]
2004	Disable bracketed paste mode SOURCE: [XTerm] [Paste64]

`CSI Ps… m` Select Graphic Rendition (SGR)

Defaults: `Ps1` = 0
Sets or clears one or more text attributes. Unlimited parameters are supported and are applied in received order. The following are supported:

Ps	Description	Blink	Bold	FG	BG	TF	TB

0	Default attribute, white on black	√	√	√	√	√	√
1	Bright Intensity		√			√	
2	Dim intensity		√			√	
5	Blink (By definition, slow blink)	√					√
6	Blink (By definition, fast blink) NOTE: Both blinks are the same speed.	√					√
7	Negative Image - Reverses FG and BG			√	√	√	√
8	Concealed characters, sets the foreground colour to the background colour.			√		√	√
22	Normal intensity		√			√	
25	Steady (Not blinking)	√					√
27	Positive Image - Restores FG and BG NOTE: This should be a separate attribute than 7 but this implementation makes them equal			√	√	√	√
30	Black foreground			√		√	
31	Red foreground			√		√	
32	Green foreground			√		√	
33	Yellow foreground			√		√	
34	Blue foreground			√		√	
35	Magenta foreground			√		√	
36	Cyan foreground			√		√	
37	White foreground			√		√	
38	Extended Foreground (see notes)					√	
39	Default foreground (same as white)			√		√	
40	Black background				√		√
41	Red background				√		√
42	Green background				√		√
43	Yellow background				√		√
44	Blue background				√		√
45	Magenta background				√		√
46	Cyan background				√		√
47	White background				√		√
48	Extended Background (see notes)						√

49	Default background (same as black)				√		√
91	Bright Red foreground		√	√		√	
92	Bright Green foreground		√	√		√	
93	Bright Yellow foreground		√	√		√	
94	Bright Blue foreground		√	√		√	
95	Bright Magenta foreground		√	√		√	
96	Bright Cyan foreground		√	√		√	
97	Bright White foreground		√	√		√	
100	Bright Black background	√			√		√
101	Bright Red background	√			√		√
102	Bright Green background	√			√		√
103	Bright Yellow background	√			√		√
104	Bright Blue background	√			√		√
105	Bright Magenta background	√			√		√
106	Bright Cyan background	√			√		√
107	Bright White background	√			√		√

All others are ignored.

Blink indicates the blink bit. Bold indicates the bold bit. FG indicates the foreground colour. BG indicates the background colour. TF indicates that the Tru Colour foreground is changed. TB indicates that the Tru Colour background is changed.

NOTE	For 90-97, there is no effect unless bright foreground colours are enabled.
NOTE	For 100-107, there is no effect unless bright background colours are enabled.
NOTE	For 38 and 48, two additional formats are supported, a palette selection and a direct colour selection.

For palette selection, an additional two parameters are required after that value. They are considered part of the 38/48, not separate values. The first additional parameter must be a 5. The second additional parameter specified the palette index to use. To set the foreground to orange, and

the background to a fairly dark grey, you would send: `CSI 38 ; 5 ; 214 ; 48 ; 5 ; 238 m`

The default palette is the XTerm 256-colour palette. [256colors]

For direct colour selection, an additional four parameters are required after that value. They are considered part of the 38/48, not separate values. The first additional parameter must be a 2. The second, third, and fourth specify the R/G/B values respectively. CTerm handles this with an internal temporary palette, so scrollback may not have the correct colours. The internal palette is large enough for all cells in a 132x60 screen to have unique foreground and background colours though, so the current screen should always be as expected.

SOURCE: [ECMA-48], [XTerm]

`CSI Ps n` Device Status Report (`DSR`)

Defaults: `Ps` = 0
A request for a status report. CTerm handles the following three requests:

5	Request a DSR CTerm will always reply with CSI 0 n indicating "ready, no malfunction detected"
6	Request active cursor position CTerm will reply with CSI y ; x R where y is the current line and x is the current row.
255	NON-STANDARD EXTENSION (BCDSR) Replies as though a CSI 6 n was received with the cursor in the bottom right corner. i.e.: Returns the terminal size as a position report.

SOURCE: [ECMA-48] (parameters 5 and 6 only) [BANSI] (parameter 255)

`CSI = Ps n` State/Mode Request/Report (`CTSMRR`)

NON-STANDARD EXTENSION
Defaults: `Ps` = 1
When `Ps` is 1, CTerm will respond with a Font State Report of the form `CSI = 1 ;pF ;pR ;pS0 ;pS1 ;pS2 ;pS3 n` `pF` is the first available loadable-font slot number `pR` is the result of the previous "Font Selection" request:

0	successful font selection
1	failed font selection
99	no font selection request has been received

`pS0` - `pS3` contain the font slots numbers of previously successful "Font Selection" requests into the 4 available alternate-font style/attribute values:

`pS0`	normal attribute font slot

pS1	high intensity foreground attribute font slot
pS2	blink attribute font slot
pS3	high intensity blink attribute font slot

When Ps is 2, CTerm will respond with a Mode Report of the form CSI = 2[;pN [;pN] [⋯]] n Where pN represent zero or more mode values set previously (e.g. via CSI ? pN h). Mode values cleared (disabled via CSI ? pN l) will not be included in the set of values returned in the Mode Report. If no modes are currently set, an empty parameter will be included as the first and only pN.

When Ps is 3, CTerm will respond with a Mode Report of the form CSI = 3 ; pH ; pW n Where pH is the height of a character cell in pixels, and pW is the width of a character cell in pixels.

When Ps is 4, CTerm will respond with a Mode Report of the form CSI = 4 ; pF n Where pF is 1 if LCF mode is enabled, and 0 if it is disabled.

When Ps is 5, CTerm will respond with a Mode Report of the form CSI = 5 ; pF n Where pF is 1 if LCF mode is forced, and 0 if it is not.

CSI ? Ps [; Pn] n Device Status Report (DECDSR)

When Ps is 62 (DECMSR) and there is no Pn, CTerm will respond with a Mode Report of the form CSI 32767 * { This indicates that 524,272 bytes are available for macro storage. This is not actually true, SyncTERM will use all available memory for macro storage, but some software checks this value, and some parsers don't allow more than INT16_MAX parameter values.

When Ps is 63 (DECCKSR) Pn defaults to 1, and CTerm will respond with a checksum of the defined macros in the form DCS Pn ! xxxx ST Where xxxx is the hex checksum.

SOURCE: [VT420]

CSI Pn1 ; Pn2 r Set Top and Bottom Margins (DECSTBM)

Defaults: Pn1 = 1 Pn2 = last line on screen
Selects top and bottom margins, defining the scrolling region. Pn1 is the line number of the first line in the scrolling region. Pn2 is the line number of the bottom line.

SOURCE: [XTerm]

CSI Ps1 ; Ps2 * r Select Communication Speed (DECSCS)

Set the output emulation speed. If Ps1 or Ps2 are omitted, causes output speed emulation to stop Ps1 may be empty. Sequence is ignored if Ps1 is not empty, 0, or 1. The value of Ps2 sets the output speed emulation as follows:

Value	Speed
empty, 0	Unlimited
1	300
2	600

Value	Speed
3	1200
4	2400
5	4800
6	9600
7	19200
8	38400
9	57600
10	76800
11	115200

SOURCE: [VT420]

`CSI ? Ps··· s` Save Mode Setting (`CTSMS`)

NON-STANDARD EXTENSION
Saves the current mode states as specified by `CSI ? l` and `CSI ? h`. If `Ps1` is omitted, saves all such states. If one or more values of `Ps` is included, saves only the specified states (arguments to `CSI ? l`/`h`).

`CSI Pn1 ; Pn2 s` Set Left and Right Margins (`DECSLRM`)

(Only when DEC Left Right Margin Mode - 69 - is enabled)

Defaults: `Pn1` = 1 `Pn2` = last column on screen
If either `Pn1` or `Pn2` is zero, the current setting is retained. Selects left and right margins, defining the scrolling region. `Pn1` is the column number of the first column in the scrolling region. `Pn2` is the column number of the right column.

SOURCE: [XTerm]

`CSI s` Save Current Position (`SCOSC`)

(Only when DEC Left Right Margin Mode - 69 - is disabled) NON-STANDARD EXTENSION Saves the current cursor position for later restoring with `CSI u` although this is non-standard, it's so widely used in the BBS world that any terminal program MUST implement it.

SOURCE: [ANSISYS]

`CSI Ps ; Pn1 ; Pn2 ; Pn3 t` Select a 24-bit colour (`CT24BC`)

NON-STANDARD EXTENSION

If `Ps` is 0, sets the background colour. If `Ps` is 1, sets the foreground colour. `Pn1`, `Pn2`, `Pn3` contains the RGB value to set. CTerm handles this with an internal temporary palette, so scrollback may not have the correct colours. The internal palette is large enough for all cells in a 132x60 screen to have

unique foreground and background colours though, so the current screen should always be as expected.

`CSI ? Ps… u` Restore Mode Setting (`CTRMS`)

NON-STANDARD EXTENSION

Restores the mode states as saved via `CSI ? s`. If `Ps` is omitted, restores all such states. If one or more values of `Ps` is included, restores all the specified states (arguments to `CSI ? l`/`h`)

`CSI u` Restore Cursor Position (`SCORC`)

Move the cursor to the last position saved by `CSI s`. If no position has been saved, the cursor is not moved.

SOURCE: [ANSISYS]

`CSI 2 $ w` Request Tab Stop Report (`DECTABSR`)

Requests a list of tab stops. The list is in the form: `DCS 2 $ u Pt ST`

The string `Pt` is a list of tab stops separated by `/`s.

SOURCE: [VT320]

`CSI Pn1 ; Ps ; Pn2 ; Pn3 ; Pn4 ; Pn5 * y` Request Checksum of Rectangular Area (`DECRQCRA`)

Returns a checksum for the specified rectangular area. `Pn1` is an ID that is returned in the response. `Ps` MUST be 1 `Pn2` specifies the top row of the rectangle `Pn3` specifies the left column of the rectangle `Pn4` specifies the bottom row of the rectangle `Pn5` specifies the right column of the rectangle The return value is in the format of `DCS Pn1 ! ~ xxxx ST` Where xxxx is the hex value of the checksum.

Source: [VT420]

`CSI Pn * z` Invoke Macro (`DECINVM`)

Invokes a macro. `Pn` specifies the macro number. If `Pn` is not 0..63, no action is taken.

SOURCE: [VT420]

`CSI = Ps1 ; Ps2 {` (`CTOSF`)

NON-STANDARD EXTENSION (Deprecated)
Defaults: `Ps1` = 255 `Ps2` = 0
Indicates that a font block is following. `Ps1` indicates the font slot to place the loaded font into. This must be higher than the last default defined font (See `CSI sp D` for list of predefined fonts) `Ps2` indicates font size according to the following table:

0	8x16 font, 4096 bytes.
1	8x14 font, 3584 bytes.

2	8x8 font, 2048 bytes.

The DCS font string should be used instead as of CTerm 1.213

"ANSI" Music

This is the place where the BBS world completely fell on it's face in ANSI usage. A programmer with either TeleMate or QModem (the first two programs to support "ANSI" music as far as I can tell) decided they needed a method of playing music on a BBS connection. They decided to add an "unused" ANSI code and go their merry way. Since their product didn't implement `CSI M` (Delete line) they assumed it was unused and blissfully broke the spec. They defined "ANSI" music as: `CSI M <music string> 0x0e`

They used a subset of IBM BASICs PLAY statement functionality for ANSI music strings which often start with "MF" or "MB", so the M after the CSI was often considered as part of the music string. You would see things such as: `CSI MFABCD 0x0e` and the F would not be played as a note. This just added further confusion to the mess.

Later on, BananaCom realized the conflict between delete line and music, so they added **another** broken code `CSI N` (Properly, erase in field... not implemented in many BBS clients) which was to provide an "unbroken" method of playing music strings. They also used `CSI Y` to disambiguate delete line, `CSI Y` is supposed to be a vertical tab (also not implemented in very many clients). BananaCom also introduced many more non-standard and standard-breaking control sequences which are not supported by CTerm.

CTerm has further introduced a standard compliant ANSI music introducer `CSI |`

By default, CTerm allows both `CSI N` and `CSI |` to introduce a music string. Allowed introducers are set by `CSI = p1 M` as defined above.

The details of ANSI music then are as follows: The following characters are allowed in music strings: "aAbBcCdDeEfFgGlLmMnNoOpPsStT0123456789.-+#<> " If any character not in this list is present, the music string is ignored as is the introducing code.

If the introducing code is `CSI M` the first char is examined, and if it is a one of "BbFfLlSs" or if it is "N" or "n" and is not followed by a decimal digit, then the music string is treated as though an M is located in front of the first character.

The music string is then parsed with the following sequences supported:

`Mx`

sets misc. music parameters where x is one of the following:

`F`	Plays music in the foreground, waiting for music to complete playing before more characters are processed.
`B`	Play music in the background, allowing normal processing to continue.
`N`	"Normal" not legato, not staccato
`L`	Play notes legato

S	Play notes staccato

T###

Sets the tempo of the music where ### is one or more decimal digits. If the decimal number is greater than 255, it is forced to 255. If it is less than 32, it is forced to 32. The number signifies quarter notes per minute. The default tempo is 120.

O###

Sets the octave of the music where ### is one or more decimal digits. If the decimal number is greater than 6, it is forced to 6. The default octave is 4.

N###

Plays a single note by number. Valid values are 0 - 71. Invalid values are played as silence. Note zero is C in octave 0. See following section for valid note modifiers.

A, B, C, D, E, F, G, or P

Plays the named note or pause from the current octave. An "Octave" is the rising sequence of the following notes: C, C#, D, D#, E, F, F#, G, G#, A, A#, B The special note P is a pause. Notes may be followed by one or more modifier characters which are applied in order. If one overrides a previous one, the last is used. The valid modifiers are:

+ - Sharp

The next highest semitone is played. Each sharp character will move up one semitone, so "C++" is equivalent to "D".

 # - Sharp

The next highest semitone is played. Each sharp character will move up one semitone, so "C##" is equivalent to "D".

- - Flat

The next lowest semitone is played. Each flat character will move down one semitone, so "D--" is equivalent to "C".

. - Duration is 1.5 times what it would otherwise be

Dots are not cumulative, so C.. is equivalent to C.

- Notelength as a reciprocal of the fraction of a whole note to play the note for

For example, 4 would indicate a 1/4 note. The default note length is 4.

L###

Set the notelength parameter for all following notes which do not have one specified (ie: override the quarter-note default) Legal note lengths are 1-64 indicating the reciprocal of the fraction (ie: 4 indicates a 1/4 note).

<

Move the next lowest octave. Octave cannot go above six or below zero.

>

Move to the next highest octave. Octave cannot go above six or below zero.

The lowest playable character is C in octave zero. The frequencies for the six C notes for the seven octaves in rising order are: 65.406, 130.810, 261.620, 523.250, 1046.500, 2093.000, 4186.000

Purists will note that the lower three octaves are not exactly one half of the next higher octave in frequency. This is due to lost resolution of low frequencies. The notes **sound** correct to me. If anyone can give me an excellent reason to change them (and more correct integer values for all notes) I am willing to do that assuming the notes still sound "right".

NMOTE: If you are playing some ANSI Music then ask the user if they heard it, ALWAYS follow it with an 0x0f 0x0e is the shift lock character which **will** cause people with anything but an ANSI-BBS terminal (ie: *nix users using the bundled telnet app) to have their screen messed up. 0x0f "undoes" the 0x0e.

Sequences sent by SyncTERM

The following keys in SyncTERM result in the specified sequence being sent to the remote. This is not part of CTerm, but are documented here for people who want to maintain compatibility.

Left Arrow	"\033[D"
Right Arrow	"\033[C"
Up Arrow	"\033[A"
Down Arrow	"\033[B"
Home	"\033[H"
End	"\033[K"
Select	"\033[K" (Same as End due to termcap weirdness)
Delete	"\x7f"
Page Down	"\033[U"
Page Up	"\033[V"
F1	"\033[11~"
F2	"\033[12~"
F3	"\033[13~"
F4	"\033[14~"
F5	"\033[15~"
F6	"\033[17~" (Note the jump from 15 to 17 here)
F7	"\033[18~"
F8	"\033[19~"
F9	"\033[20~"
F10	"\033[21~"

F11	"\033[23~" (Note the jump from 21 to 23 here)
F12	"\033[24~"
Shift + F1	"\033[11;2~"
Shift + F2	"\033[12;2~"
Shift + F3	"\033[13;2~"
Shift + F4	"\033[14;2~"
Shift + F5	"\033[15;2~"
Shift + F6	"\033[17;2~"
Shift + F7	"\033[18;2~"
Shift + F8	"\033[19;2~"
Shift + F9	"\033[20;2~"
Shift + F10	"\033[21;2~"
Shift + F11	"\033[23;2~"
Shift + F12	"\033[24;2~"
Alt + F1	"\033[11;3~"
Alt + F2	"\033[12;3~"
Alt + F3	"\033[13;3~"
Alt + F4	"\033[14;3~"
Alt + F5	"\033[15;3~"
Alt + F6	"\033[17;3~"
Alt + F7	"\033[18;3~"
Alt + F8	"\033[19;3~"
Alt + F9	"\033[20;3~"
Alt + F10	"\033[21;3~"
Alt + F11	"\033[23;3~"
Alt + F12	"\033[24;3~"
Control + F1	"\033[11;5~"
Control + F2	"\033[12;5~"
Control + F3	"\033[13;5~"
Control + F4	"\033[14;5~"
Control + F5	"\033[15;5~"
Control + F6	"\033[17;5~"
Control + F7	"\033[18;5~"
Control + F8	"\033[19;5~"

Control + F9	"\033[20;5~"
Control + F10	"\033[21;5~"
Control + F11	"\033[23;5~"
Control + F12	"\033[24;5~"
Insert	"\033[@"
Back Tab	"\033[Z"

References

- [STD-070] Digital Equipment Corporation. Video Systems Reference Manual. 1989-04-14.
- [ECMA-48] ECMA. Control Functions for Coded Character Sets. June 1991
- [XTerm] Edward May. XTerm Control Sequences. University of California, Berkeley. 2024/09/19
- [Paste64] Thomas E. Dickey. XTerm — bracketed paste. 2022
- [BANSI] Paul Wheaton. BANSI.TXT. 1999
- [VT102] Digital. VT102 Video Terminal User Guide. 1982.
- [VT330/340] Digital. VT330/VT340 Programmer Reference Manual, Volume 2: Graphics Programming. May 1988.
- [VT320] Digital. Installing and Using the VT320 Video Terminal. June 1987.
- [256colors] Jonas Jarad Jacek. 256 colors cheat sheet. 2023-12-24.
- [VT420] Digital. Installing and Using the VT420 Video Terminal. June 1990.
- [ANSISYS] Wikipedia. ANSI.SYS.

Ciolib Manual

Introduction

Ciolib originated as a FreeBSD/Linux implementation of the Borland conio library for use by the Synchronet User InterFaCe library (UIFC). Since then, it has grown more complete by being used to port other software that was originally implemented using Borland C.

The use of ciolib in SyncTERM however, kicked off an explosion in capabilities, and SyncTERM has been the primary driver of ciolib development ever since. Graphics, multiple font, TruColor, window scaling and more have been added to ciolib to extend SyncTERM. In addition, the ANSI parsing and displaying code CTerm is part of ciolib, and not SyncTERM.

Output Modes

An output mode specifies the manner in which ciolib displays the content to the user. There are twelve ciolib output modes that can be broadly grouped into two categories:

Text Output Modes

These modes use a text based library or interface to display character cells and attributes. These

modes are incapable of graphics.

Curses Modes

In one of the curses modes, the curses (or ncurses if available) API provided by the OS is used for output. This means it requires the TERM variable be set appropriately, and needs to run inside of another terminal emulator such as XTerm or Kitty. There are three curses modes.

Curses

This uses the wide char curses API and supports unicode input and output, translating as needed. This provides the highest quality in almost all terminals.

Curses IBM

When in this mode, ciolib assumes that any characters displayed on the screen will be in IBM codepage 437, and translates to and from that as appropriate. This mode can work well inside of a CP437 BBS connection, but makes many assumptions that are suspect.

Curses ASCII

In this mode, output is restricted to the ASCII character set, and everything is translated to that. This is the least capable curses mode.

ANSI Mode

Ciolib will output ANSI control sequences on stdout in this mode. In general, the sequences used are in line with "ANSI-BBS". This is similar to curses mode, except the TERM variable has no impact, and all characteristics of the terminal emulator used for display are simply assumed. This is the best mode to use from inside a BBS connection.

Windows Conio

Only available on Windows, this uses the old Windows NT console API to output text. Newer versions of Windows have changed this API considerably, so until ciolib is updated to support the new console, this is of limited usefulness.

Graphical Modes

In graphical modes, ciolib controls every pixel that is displayed. As a result of this, every feature of ciolib is available to every graphical mode. The main reasons to choose one over another is portability and OS.

Graphical modes usually also have a fullscreen variant.

SDL Mode

SDL mode is the most portable of the modes as it uses libsdl, a library designed for writing cross-platform games. SDL supports many more platforms than ciolib does, and is usually the first graphical mode supported on a new platform.

Unfortunately, SDL mode tends to be more complex and usually somewhat slower or more CPU intensive than other modes, so is usually only used as a fallback.

X mode

This uses libX11 to communicate with an X server. Historically, the GUI for most *nix systems

were provided via an X server. While Linux distributions are moving to Wayland, evne Wayland still support libX11 applications.

GDI mode

This directly uses the Win32 Graphics Display Interface (GDI) API.

Text Modes

Ciolib operates in exactly one text mode. This is distinct from the Text Output Mode mentioned above. Internally, a text mode is defined via fourteen values:

Mode Number

A unique ID for each mode. Many of these were defined by Borland, but the list has been extended by various parties over the years.

Palette

The palette is a mapping of attribute values to TruColor RGB values. For historical DOS modes, this defines the standard sixteen colours (or up to three intensities for monchrome modes). For other modes such as the Commodore 64 and Atari modes, the palette is very different.

Columns

The number of columns on the display.

Rows

The number of rows on the display.

Cursor Start

The pixel row number the default cursor starts on. DOS modes tend to use a two pixel high underline cursor, while Commodore uses a full block cursor.

Cursor End

The pixel row number the default cursor ends on.

Character Height

The height of a single cell on the display. This indirectly sets the allowed fonts, since only fonts with the specified height can be used.

Character Width

The width of a character cell. In all except one mode, this is 8. However, in the VGA80X25 mode, this is 9. When this is 9, an 8 pixel wide font may still be used if the `VIDMODES_FLAG_EXPAND` flag is set.

Default Attribute

The attribute used when a screen is initialized. Light Gray on Black for DOS modes, but varies for other modes.

Flags

Flags can be set to control on/off behaviours in a mode. There are currently two flags that can be

set

CIOLIB_VIDEO_EXPAND

This flag is used to add an extra pixel to the right side of each character cell that is not present in the font data.

CIOLIB_VIDEO_LINE_GRAPHICS_EXPAND

An algorithm from IBM graphics cards is used to fill in an extra pixel column. This makes line drawing characters connect across the space, but leaves a single pixel gap between block drawing characters.

Aspect Ratio Width

See next item.

Aspect Ratio Height

Aspect Ratio Height and Aspect Ratio Width controls the aspect ratio the mode is scaled to. Most historical text modes did not use square pixels, but ciolib assumes that its output does use square pixels. It's simplest to describe these old modes using the aspect ratio of the display and the pixel resolution. Historical display were almost universally 4:3 aspect ratio. The Commodore 64 however used large borders on the sides, and the aspect ratio is actually 6:5. There is a small number of additional modes that use aspect ratios with square pixels. These are:

ST132X37_16_9

This is a 132x37 mode with square pixels and a 16:9 aspect ratio.

ST132X37_5_4

This is a 132x52 mode with square pixels and a 5:4 aspect ratio.

LCD80X25

This is an 80X25 mode with square pixels and an 8:5 aspect ratio. This mode was added to provide a way of avoiding scaling and the resulting "blurriness".

X Resolution

The width in pixels of the display.

Y Resolution

The height in pixels of the display.

There is a custom mode defined where the values can be modified by the program to create exactly the desired mode.

Fonts

There are a large number of built in fonts that ciolib supports. Most are codepage fonts, but there is also Commodore, Atari, and Amiga fonts included in the default set. Not all builtin fonts are available in every size. The following table summarizes the available fonts.

Name	8x16	8x12	8x8	Character Set
Codepage 437 English	√	√	√	CP437
Codepage 1251 Cyrillic, (swiss)	√			CP1251
Russian koi8-r	√	√	√	KOI8_R
ISO-8859-2 Central European	√	√	√	ISO_8859_2
ISO-8859-4 Baltic wide (VGA 9bit mapped)	√			ISO_8859_4
Codepage 866 (c) Russian	√			CP866M
ISO-8859-9 Turkish	√			ISO_8859_9
haik8 codepage (use only with armscii8 screenmap)	√	√	√	HAIK8
ISO-8859-8 Hebrew	√	√	√	ISO_8859_8
Ukrainian font koi8-u	√	√	√	KOI8_U
ISO-8859-15 West European, (thin)	√			ISO_8859_15
ISO-8859-4 Baltic (VGA 9bit mapped)	√	√	√	ISO_8859_4
Russian koi8-r (b)	√			KOI8_R
ISO-8859-4 Baltic wide	√			ISO_8859_4
ISO-8859-5 Cyrillic	√	√	√	ISO_8859_5
ARMSCII-8 Character set	√	√	√	ARMSCII8
ISO-8859-15 West European	√	√	√	ISO_8859_15
Codepage 850 Multilingual Latin I, (thin)	√		√	CP850
Codepage 850 Multilingual Latin I	√	√	√	CP850
Codepage 865 Norwegian, (thin)	√		√	CP865
Codepage 1251 Cyrillic	√	√	√	CP1251
ISO-8859-7 Greek	√	√	√	ISO_8859_7
Russian koi8-r (c)	√			KOI8_R
ISO-8859-4 Baltic	√	√	√	ISO_8859_4
ISO-8859-1 West European	√	√	√	ISO_8859_1
Codepage 866 Russian	√	√	√	CP866M2
Codepage 437 English, (thin)	√		√	CP437
Codepage 866 (b) Russian	√			CP866M2
Codepage 865 Norwegian	√	√	√	CP865
Ukrainian font cp866u	√	√	√	CP866U
ISO-8859-1 West European, (thin)	√			ISO_8859_1
Codepage 1131 Belarusian, (swiss)	√			CP1131
Commodore 64 (UPPER)	√		√	PETSCIIU

Name	8x16	8x12	8x8	Character Set
Commodore 64 (Lower)	√		√	PETSCIIL
Commodore 128 (UPPER)	√		√	PETSCIIU
Commodore 128 (Lower)	√		√	PETSCIIL
Atari	√		√	ATASCII
P0T NOoDLE (Amiga)	√	√		ISO_8859_1
mO’sOul (Amiga)	√		√	ISO_8859_1
MicroKnight Plus (Amiga)	√			ISO_8859_1
Topaz Plus (Amiga)	√			ISO_8859_1
MicroKnight (Amiga)	√		√	ISO_8859_1
Topaz (Amiga)	√	√		ISO_8859_1
Prestel	√			ISO_8859_1