

MICRO-MONITOR

An Embedded System Boot Platform

Host Based Tools

Revision date: November 15, 2002

MicroMonitor: Host Based Tools Table of Contents

This text contains a description of various tools, most of which can be built to run on a win32 or UNIX environment, and provide some type of support for the MicroMonitor platform. Some of the tools are provided and may or may not be of use depending on your environment; nevertheless, I've documented the tools that I've found useful at one point or another during embedded system development.

AOUT.....	3
BIN2ARRAY.....	4
BIN2SREC.....	5
CLINET.....	6
COFF.....	8
COM.....	10
DEFDATE.....	11
DHCPSRVR.....	13
ELF.....	16
F2MEM.....	18
FCMP.....	20
IBSI.....	21
MACCRYPT.....	22
MONCMD.....	23
MONSYM.....	24
NEWMON.....	26
TITLE.....	27
TTFTP.....	28
WHENCE.....	30

AOUT

NAME:

aout

SYNOPSIS:

Extract information from or compress sections of an AOUT file.

USAGE:

aout [-a:AB:cfMm:sS:v] {aout_filename}

DESCRIPTION:

This tool is one of three tools (coff, elf & aout) that basically perform the same tasks, but on different file formats. It allows the user to dump different portions of the file header to stdout in a readable format. Also, future plans include support for using zlib to compress each of the .text and .data sections so that TFS can store a compressed file in flash and decompress directly from TFS to the absolute-linked location for the application to run in.

OPTIONS:

- -a {filename}
append file to end of -S file;
- -A
print what was previously appended by -a
- -B {bin_filename}
convert file (aout_filename) to binary (bin_filename) for transfer to absolute memory space;
- -c
BE-to-LE convert... By default, this tool assumes the aout file was built with big-endian control structures. This option will convert to little endian control structures.
- -f
show AOUT file header;
- -M
show AOUT map with file offsets;
- -m
show AOUT map without file offsets;
- -p {filename}
pack to specified file (output file is also stripped)
This is soon to be replaced with zlib compression.
- -s
show AOUT section headers;
- -S {filename}
strip to specified file
- -v
verbose (debug) mode

NOTES:

- If, after running this tool, some type of file error (read, write, lseek, etc...) occurs, it is likely that the executable being operated on was built with control structures of a different endian-type. Try re-running with the -c option.

EXAMPLES:

- aout -a info -S app.str app
Strip the file *app*, place the result in *app.str*, then append the content of the file *info* to the end of *app.str*. The command line "strings app.str | tail -1" can then be run to extract the last string (content of *info* file) from the binary.

EXIT STATUS:

0 if successful
1 error

BIN2ARRAY

NAME:

bin2array

SYNOPSIS:

Convert a binary file to compile-able array in 'C'.

USAGE:

bin2array [-Aa:b:D:e:no:sw:V:] {input_file} [array-name]

DESCRIPTION:

This tool takes a binary file and converts it to an ascii file that contains a compile-able 'C' array that contains the ascii-coded hex equivalent of the content of the file. For example, if I have a file with the following content...

```
1234567890
```

Then bin2array would create something like this...

```
unsigned char some_array[] = {  
    0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x30, 0x0a  
};
```

Notice that the 0x31, 0x32, 0x33, etc...(from the output file) is the ascii-coded-hex equivalent of 123 etc... (from the input file).

OPTIONS:

- -A
use assembler byte format
- -a {arrayname}
specify the array name to be used. If not specified, then the array name will match the filename.
- -b {offset}
offset into binary file to begin at
- -D {define}
up to 16 "#define" statements inserted in output
- -e {offset}
offset into binary file to end at
- -n
null-terminate the created array.
- -o {filename}
output filename (default = stdout)
- -s
swap data
- -w {width}
unit width (1|2|4) (default = 1)
- -V
display version (build date) of tool

EXAMPLES:

- bin2array -a file_bin -o file_array file
Convert the entire content of 'file' in the array "file_bin[]" stored in file "file_array.c", .

EXIT STATUS:

0 if successful, 1 error

BIN2SREC

NAME:

bin2srec

SYNOPSIS:

Convert a binary file to an S3 record file.

USAGE:

bin2srec [-b:o:] {input_file}

DESCRIPTION:

This tool takes a binary file and converts it to S3-record format. The incoming file can be converted to exist at some address other than zero by specifying a base address; also, an offset into the input-file can be specified to "cut" into some binary.

OPTIONS:

- -b {base address}
override default base address of 0;
- -o {offset}
specify an offset (default is none) into the input_file at which point data conversion is started;

EXAMPLES:

- bin2srec file
Convert the entire content of 'file' to S-Record format.

EXIT STATUS:

0 if successful
1 error

CLINET

NAME:

clinet

SYNOPSIS:

A "**CLI-to-TELNET**" connection server.

USAGE:

clinet -[c:dr:s:] [config_file_name]

DESCRIPTION:

This tool allows a remote user to connect to a target system's command line interface (CLI) that is tied to a PC's COM port. At a very minimum, cliNet can be used as a terminal emulator. The intended configuration; however, is for this to be running on a networked PC whose COM port is tied to the serial port of some target embedded system. On the network side of the PC, cliNet looks like a telnet server running on port 8200 (default, can be overridden). At some remote location a user can access the target system's serial port using a telnet client to connect to the PC's IP address using port 8200. After the connection is established, the user of the telnet client is virtually connected to the serial port of the target; hence, characters typed by the user at the client are sent to the target's serial interface. Similarly, characters generated by the target system are sent to the user through the telnet connection.

At the local PC running this tool, cliNet is seen by the user as a simple terminal emulator. The local user interfaces to the target's serial port in the same way a remote user does. CliNet supports the ability to have multiple users connected to a specific COM port on the PC: one user can be working at the local PC and multiple additional users can be connected through telnet sessions. This implies some kind of coordination between users, of course.

For most up-to-date information, refer to the HELP menu items within the running tool.

OPTIONS:

- -c {column_count}
allows the user to override the default width of 80 columns (characters)
- -d
enable debug mode (for development only)
- -r ###[,###]
allows the user to override the default height of the console (40 rows) and trace (8 rows) windows. The presence of the comma tells cliNet that the number immediately following the comma is the new value to be used for the trace window height.
- -s {scriptname}
automatically run the script specified by "scriptname" at startup

FEATURES:

For complete details of the features mentioned below, refer to the help information that comes with cliNet.

Logging-

All interaction with the CLI can be logged to a file.

TimeStamping-

At the start of each line, a date/time stamp can be inserted.

Configuration Save/Restore-

The configuration established during a session can be saved and reloaded at a later time; thus, eliminating the need to reconfigure a "personality" for various cliNet sessions.

Programmable Function Keys-

F1 thru F12 are programmable to a key sequence you specify.

ASCII-Coded-Hex Output Option-

If you're not seeing what you think is being transmitted over the comport because some characters are non-printables, then you can enable a mode in which all characters are shown on the screen in ascii-coded-hex; plus, an ascii-to-hex conversion table is provided as one of the help menu items.

System Trace Window-

Various portions of the session can be traced and that trace output can be seen in a small trace window that becomes part of the session window through a menu-item click or accellerator key stroke (SHIFT-F1).

Scripting-

CliNet supports the ability to run scripts. The scripting language built into cliNet allows the user to not only execute a file of commands, but also establish a program flow using variables, goto/gosub and conditional branching. The script can be run during a cliNet session or it can be run as a startup script through the -s option of the command line.

Telnet or Serial Backend-

Typically, the back end of cliNet is a COM port connection on the PC; however, there are cases where the features of cliNet are useful during a telnet client session. The backend of cliNet can be telnet. This is typically used when a remote cliNet session is tied to a COM port and a user wants to connect to that session through a telnet client. The COM port or TELNET backend is established during the initial dialog with cliNet at startup (or through a configuration file that was saved with cliNet already in that mode).

File Transfer-

CliNet can transfer files through a TFTP client or XMODEM.

Built-in Servers-

CliNet is essentially a tool for embedded system developers. There are times when certain servers are needed, and they can be a pain in the neck to get up and running. CliNet provides versions of these servers that can be run as part of the session. The current philosophy of cliNet servers is that once the server is started, it can only be shutdown by terminating the cliNet session.

- Telnet: this server runs to allow multiple remote users to connect to the same COM port that cliNet is attached to locally. A telnet client can connect to cliNet (usually on port 8200) and see the same serial traffic as would be seen if that user was on the PC directly. In addition to providing a simple pipe to a remote user, the telnet server will treat "CTRL-p" (the control key and the 'p' character held down simultaneously) as a special request from the remote user to execute a command on the machine running the server.
- FTP: cliNet's FTP server is a superset of the "Minimum Implementation" specification of RFC959.
- TFTP: cliNet's TFTP server is a single threaded server (supporting one transfer at a time).
- DHCP/BOOTP: cliNet's DHCP/BOOTP server is easily configured to support most of the capabilities of a typical DHCP/BOOTP server.

Note that the above servers (not including telnet) are provided in cliNet as a convenience to an embedded system developer. They are not currently intended to run as full-blown servers on a system; however if traffic is minimal, then there's no reason why they won't work just fine.

EXAMPLES:

- cliNet
Bring up a startup dialog box that allows the user to configure the connections provided by cliNet.

COFF

NAME:

coff

SYNOPSIS:

Extract information from or compress sections of a COFF file.

USAGE:

coff [-a:AB:cfMmp:sS:v] {coff_filename}

DESCRIPTION:

This tool is one of three tools (coff, elf & aout) that basically perform the same tasks, but on different file formats. It allows the user to dump different portions of the file header to stdout in a readable format. Also, future plans include support for using zlib to compress each of the .text and .data sections so that TFS can store a compressed file in flash and decompress directly from TFS to the absolute-linked location for the application to run in.

OPTIONS:

- -a {filename}
append file to end of -S file;
- -A
print what was previously appended by -a
- -B {bin_filename}
convert file (coff_filename) to binary (bin_filename) for transfer to absolute memory space;
- -c
BE-to-LE convert... By default, this tool assumes the coff file was built with big-endian control structures. This option will convert to little endian control structures.
- -f
show COFF file header;
- -M
show COFF map with file offsets;
- -m
show COFF map without file offsets;
- -p {filename}
pack to specified file (output file is also stripped)
This is soon to be replaced with zlib compression.
- -s
show COFF section headers;
- -S {filename}
strip to specified file
- -v
verbose (debug) mode
- -V
print the time/date the tool was built
- -z {zip level}
compress the COFF sections using zlib compression level "zip level". Refer to TFS decompression for more details on this.

NOTES:

- If, after running this tool, some type of file error (read, write, lseek, etc...) occurs, it is likely that the executable being operated on was built with control structures of a different endian-type. Try re-running with the -c option.

EXAMPLES:

- `coff -a info -S app.str app`
Strip the file *app*, place the result in *app.str*, then append the content of the file *info* to the end of *app.str*. The command line "`strings app.str | tail -1`" can then be run to extract the last string (content of *info* file) from the binary.

EXIT STATUS:

0 if successful
1 error

COM

NAME:

com

SYNOPSIS:

Very basic com port communication tool for connection to monitor's RS-232 port.

USAGE:

com [-b:p:\V]

DESCRIPTION:

This tool provides a console-based application for communication with a target's RS-232 port. It provides simple connection for command entry and xmodem file transfer.

OPTIONS:

- -p {##}
com port number (default is 1)
- -f {filename}
name of file to transfer
- -x {cmd}
xmodem command to send to target prior to starting transfer
- -i
show invisible characters
- -b {##}
baud rate (default is 9600)
- -V
display the version (build date) of the tool

EXAMPLES:

- com -b 57600 -p2
Connect to COM2 at 57600 baud.

DEFDATE

NAME:

defdate

SYNOPSIS:

Simple utility to generate the text to build a header file containing current time & date.

USAGE:

defdate [-fnV] [macro_name]

DESCRIPTION:

This tool provides an alternative to a compiler's `__DATE__` and `__TIME__` macros. The intent is that this be used within a makefile to update the content of a header file whose macro is used somewhere in source code to keep track of the time at which the code was built. If an argument is present, defdate will print out a string that can be part of a header file (`#define MACRO "datestring"`); if no argument is present, then only the date string itself is printed.

OPTIONS:

- -f {format}
override the default format of `%b_%d,%Y@%H_%M...`
 - %a: abbreviated weekday name
 - %A: full weekday name
 - %b: abbreviated month name
 - %B: full month name
 - %c: date and time representation appropriate for locale
 - %d: day of month as decimal number (01-31)
 - %H: hour in 24-hour format (00-23)
 - %I: hour in 12-hour format (01-12)
 - %j: day of year as decimal number (001-366)
 - %m: month as decimal number (01-12)
 - %M: minute as decimal number (00-59)
 - %p: current locale's AM/PM indicator for 12-hour clock
 - %S: second as decimal number (00-59)
 - %U: week of year as decimal number, with Sunday as first day of week (00-51).
 - %w: weekday as decimal number (0-6; Sunday=0)
 - %W: week of year as decimal number, with Monday as first day of week (00-51).
 - %x: date representation of current locale
 - %X: time representation for current locale
 - %y: year without century, as decimal number (00-99)
 - %Y: year with century, as decimal number
 - %z: time-zone name or abbreviation; no chars if time zone is unknown
 - %Z: all-caps version of %z
- -n don't append a newline character to the end of the date string;
- -V print the version of defdate.exe

EXAMPLES:

- defdate DATETIME info.h
Create a header file with a string formatted as follows: `#define DATETIME "May_21,1999@1159"`
- defdate
Print the current date in the format `May_21,1999@1159`.

EXIT STATUS:

MicroMonitor
Host Based Tools

0 if successful
1 if error

DHCP SRVR

NAME:

dhcpsvr

SYNOPSIS:

Run a basic dhcp or bootp server.

USAGE:

dhcpsvr [-A:a:bDdc:Chq:TvVw] {target ip-address} "target command string"

DESCRIPTION:

This tool provides a basic DHCP or BOOTP server for MicroMonitor clients. For DHCP, 'automatic allocation' mode is supported (no lease expiration). The intent of the program is STRICTLY for basic support of MicroMonitor clients. It expects to interact with only one client at a time, and simply provides a convenient alternative to setting up a "real" server somewhere. The server requires a configuration file to startup. It can be automatically generated via *dhcpsvr -C*. Refer to notes in that output for syntax information.

Complete documentation on the dhcpsvr tool can be dumped to a file via *dhcpsvr -h*. Where differences are detected, assume dhcpsvr -h output is more accurate.

OPTIONS:

- -A {arpcmdname}
self-extract and build an arp command for use by this server if not available on the system already;
- -a {arpcmdname}
use the specified string as the command name replacement for "arp";
- -b
broadcast reply to client (eliminates need for arp);
- -C
dump a template config file to stdout;
- -c {cfgfile_name}
override the default (dhcpsvr.cfg) config file name;
- -D
don't issue the arp;
- -d
don't check for the existence of an arp entry, just execute arp -d regardless;;
- -h
dump more help information to stdout;
- -q {n}
quit after 'n' discover/request handshakes;
- -T
start up a TFTP server (recommendation: use *tftp svr* because this is soon to be eliminated from dhcpsvr);
- -v
verbose mode;
- -V
print version of dhcpsvr tool (build date);
- -w
don't print warnings of incomplete .cfg entries;

EXAMPLES:

- dhcprsvr
Based on the file "dhcprsvr.cfg" in the current working directory; start up a dhcp/bootp server.
- dhcprsvr -C dhcprsvr.cfg
Create a template dhcprsvr config file.
- dhcprsvr -A myarp; dhcprsvr -a myarp
Build an "arp" tool called myarp and then tell the dhcp server to run with that version of arp.

WARNING:

When a DHCP server is to respond to a client that is making a request it has a catch-22 to deal with... The server's TCP/IP stack wants to talk to the client through the clients IP address, but the client doesn't have an IP address assigned to it yet (that's what the server is trying to give it). The server has two alternatives: one is to force an entry into the server's ARP cache (using the arp command mentioned above); the other is to broadcast the response. Ideally, the arp-cache modification is used because it reduces traffic on the network. This is the default mechanism used by this server. Users of this dhcp server may not have Administrator privileges on the machine (required to do an arp cache modification), so the broadcast method can be used (see -D & -b options above). On most NT/Win95 systems, if the server is not able to modify the arp cache, an error is returned when the attempt is made. The user sees this error message and knows that the server must be reconfigured for broadcast. Some users have found that on some NT/Win95 boxes the arp fails but there is no warning. Be aware of this, it appears to be an inconsistency across various Window's platforms. The resolution is to just use broadcast mode (-Db options) so that the server is not dependent on the arp cache entry being made.

EXIT STATUS:

0 if successful
1 if error

EXAMPLE CONFIG FILE (output of dhcprsvr -C):

```
# Configuration File Template....
# Notes:
# * Each line consists of an ID string followed by a PARAMETER string.
#   ID and PARAMETER must be white space delimited.
# * Blank lines are ignored, and lines starting with a '#' are ignored.
# * Specifying a client MAC address of 00:00:00:00:00:00 tells the server
#   to use this as a default. Note that this should be the last entry in
#   the config file because the server will scan this file from top to
#   bottom searching for a matching CLIENT_MAC entry, if a match is not
#   found by the time this entry is scanned, the default will be used.
# * A complete entry begins with the CLIENT_MAC ID. The server expects to
#   find all other entries associated with that MAC prior to finding the
#   next CLIENT_MAC ID.
# * The server will respond to BOOTP requests also. To signify
#   a BOOTP (instead of a DHCP) entry, use BOOTP_CLIENT_MAC instead of.
#   DHCP_CLIENT_MAC.
# * The subnet of the PC that is running this server must be the same
#   subnet that the CLIENT_IP entries are set to.
# * The XXX_OPTNO_NNN entries below demonstrate the fact that the server
#   can be told to load any option with a hex, ascii or IP type of value.
# * The XXX_VSOPTNO_NNN entries below demonstrate the fact that the server
#   can be told to load any vendor-specific option (#43) with a hex, ascii
#   or IP type of value.
# Valid DHCP entry:
DHCP_CLIENT_MAC: 00:60:1D:02:0B:FE
CLIENT_IP:      135.3.94.136
SERVER_IP:      135.3.94.76
RLYAGNT_IP:     135.3.94.3
```

MicroMonitor
Host Based Tools

```
GATEWAY:          135.3.94.1
NETMASK:          255.255.255.0
SERVER_NAME:     server_name_here
BOOTFILE:        some_filename_here
STR_OPTNO_131:   some_ascii-string_here
HEX_OPTNO_132:   AABBCDDEEFF
IPA_OPTNO_133:   4.8.12.16
STR_VSOPTNO_11:  ascii_string
HEX_VSOPTNO_132: 112233
IPA_VSOPTNO_13:  1.2.3.4
#
# Note that in the above example, NETMASK is the same as IPA_OPTNO_1
# and GATEWAY is the same as IPA_OPTNO_3
#
# Valid BOOTP entry:
BOOTP_CLIENT_MAC: 00:60:1D:02:0B:FC
CLIENT_IP:        135.3.94.131
SERVER_IP:        135.3.94.76
RLYAGNT_IP:      135.3.94.1
GATEWAY:         135.3.94.1
NETMASK:         255.255.255.0
SERVER_NAME:     server_name_here_too
BOOTFILE:        some_other_filename_here
# Default..
# Uncomment this entire entry if a default is to be specified.
# It is shown here for example purposes only.
#DHCP_CLIENT_MAC: 00:00:00:00:00:00
#CLIENT_IP:      135.3.94.148
#SERVER_IP:      135.3.94.76
#GATEWAY:        135.3.94.1
#NETMASK:        255.255.255.0
#SERVER_NAME:    servername_again
#BOOTFILE:       yet_another_filename_here
```

ELF

NAME:

elf

SYNOPSIS:

Extract information from or compress sections of an ELF file.

USAGE:

elf [-a:B:cfMm:P:sS:vz] {elf_filename}

DESCRIPTION:

This tool is one of three tools (coff, elf & aout) that basically perform the same tasks, but on different file formats. It allows the user to dump different portions of the file header to stdout in a readable format. Also, future plans include support for using zlib to compress each of the .text and .data sections so that TFS can store a compressed file in flash and decompress directly from TFS to the absolute-linked location for the application to run in.

OPTIONS:

- -a {filename}
append file to end of -S file;
- -B {bin_filename}
convert file (elf_filename) to binary (bin_filename) for transfer to absolute memory space;
- -c
BE-to-LE convert... By default, this tool assumes the elf file was built with big-endian control structures. This option will convert to little endian control structures.
- -f
show ELF file header;
- -M
show ELF map with file offsets;
- -m
show ELF map without file offsets;
- -P {pad-byte}
by default, 0xff is used, this provides an override;
- -s
show ELF section headers;
- -S {filename}
strip to specified file
- -v
verbose (debug) mode
- -V
print the time/date the tool was built
- -z {zip level}
compress the COFF sections using zlib compression level "zip level". Refer to TFS decompression for more details on this.

NOTES:

- If, after running this tool, some type of file error (read, write, lseek, etc...) occurs, it is likely that the executable being operated on was built with control structures of a different endian-type. Try re-running with the -c option.

EXAMPLES:

- `elf -z6 app`
Compress the elf file "app" using zlib compression level 6. The output is placed in the file app.ezip.

EXIT STATUS:

0 if successful

1 error

F2MEM

NAME:

f2mem

SYNOPSIS:

Convert a series of files into a memory image for transfer to a flash device

USAGE:

f2mem [a:B:b:f:m:O:o:p:s:S:T:t:Vv]

DESCRIPTION:

f2mem (files to memory translator) is a tool that allows the user to take a group of files and generate a file that is suitable for a flash programmer. The intent is to feed it a monitor binary followed by a list of files that are destined to exist in TFS (Tiny File System). The monitor binary is placed at the base of the memory, then, starting at the offset specified by -t, each of the remaining files are processed and made to look as if they were residing in flash memory under TFS.

OPTIONS:

- -a {tfsname}
ascii file for TFS
For all files that are destined for TFS, and are considered ascii files, this option is used. The conversion made here is to replace "\r\n" with "\r" in the ascii data.
- -B {address}
address flash base in CPU memory (default=0)
This address is the actual address that the base of FLASH will be residing at when the device is part of the system CPU's memory map.
- -b {tfsname}
binary file for TFS
Similar to the -a option, but there is no conversion of any kind made on the file. It is placed in TFS just as it exists on the host.
- -f {hexbyte}
byte to be used as filler (default=0xff)
The default of 0xff is used so that all of the unwritten flash can be written (1-to-0 transition in flash space). If this is not the desirable case, then use this option to override the default.
- -m {filename}
monitor file
This is a required option, since this tool is assumed to be building a bootable memory image and the monitor is what does the booting.
- -O 's|b'
output type: S3-record or binary (default=S3)
- -o {filename}
output file (default = mem.bin | mem.srec)
- -p {pad-to size}
pad the file out to the specified size using the fill byte specified by -f as the pad (default is no padding);
- -s {m|a}
swap bytes in monitor binary (m) or all bytes (a)
- -S [address]
s-record base address (default=0)
In some cases, the S-record file will be used to program a device using a flash programmer that is external to the actual system that the flash will reside in. In this case, it is likely that the S-record base will be 0 because the addresses will be relative to the base of the flash device. In other cases, the S-record is used to program memory while it is part of the CPU's memory space, so the base address

MicroMonitor
Host Based Tools

used in the S-record file will be relative to the CPU space and will typically be the same value used for the -B option

- -T {#}
TFS header version (default is 0).
- -t {###}
offset relative to base of flash for TFS offset.
This is a required option that tells f2mem where the beginning of TFS space is relative to the base of the flash.
- -V
show version
- -v
verbose

NOTES:

- The -m and -t options are required.
- Syntax for tfsname: name,flags[,info]
- Currently, this tool assumes the monitor is the first piece of binary destined for the flash part. As the need arises, we will support other configurations.

EXAMPLES:

- f2mem -v -Ob -m monppc.bin -a monrc,e -t 0x40000 -B 0x80000000
Build a memory image with the monitor binary being monppc.bin, and a monrc file as the only file in TFS. TFS starts at offset 0x40000 relative to the base of the flash and the base of the flash is at address 0x80000000 relative to the CPU's address bus. The output is binary, and stored in the file mem.bin. Note that the file specification of monrc contains a ",e" after it. This is the notation used by TFS to indicate that it is to be stored in TFS with the "e" flag set (executable). Note also that since monrc is an ascii file, it is an argument to the '-a' option.
- f2mem -v -Os -o ias.srec -m monppc.bin -a monrc,e -b ias,eEB -t -x40000 -b 0x80000000
Build a memory image with the monitor, monrc file and an ELF based executable called "ias". The output is in S3-record format and will be stored in the file ias.srec. Note that the ias file is a binary file, so it is an argument to the -b option.
- f2mem -v -Os -o ias.srec -m monppc.bin -a monrc,e -t -x40000 -b 0x80000000 -S 0x80000000
Build a memory image with monitor and monrc file, but set up the S-record so that the base address is relative to the CPU's memory space, not the FLASH device.

EXIT STATUS:

0 if successful
1 error

FCMP

NAME:

fcmp

SYNOPSIS:

Compare two binary files.

USAGE:

fcmp {file1} {file2}

DESCRIPTION:

Simple file comparison tool if nothing else is available.

EXAMPLES:

- fcmp fileA fileB
Make a byte-for-byte comparison of fileA to fileB. If identical, print "files are identical" else indicate the position at which they differ.

EXIT STATUS:

0 if files are identical

1 if files are same size, but different content

2 if files are different size

IBSI

NAME:

ibsi

SYNOPSIS:

Insert the necessary boot sector information into the binary image that is destined to exist as a monitor after the boot-sector safety code runs.

USAGE:

ibsi [-c:i:lo:v] {filename}

DESCRIPTION:

This tool is used to insert the necessary data into a monitor binary image so that the boot-sector safety code can determine if the image is valid. The structure for this data is

{PRIVATE}ulong	starting point of the crc32 calculation
ulong crcstart	
ulong crcsize	size of the crc32 calculation
ulong crcvalue	expected result of the crc32 calculation
ulong entrypoint	entrypoint into the image if crc32 passes

The tool simply calculates a crc32 on a portion of the space within the binary image and places that information back into the image. The boot-sector safety code can then assume that this data exists at some fixed point within the image and use it to see if the image is safe to jump into.

OPTIONS:

- -b {####}
base address at which the .bsi monitor is to exist
- -c {####}
offset into the file at which the CRC calculation begins (default = 0);
- -i {####}
offset into the file at which the CRC is inserted (default = 0);
- -l
target is little endian (default = big endian);
- -o {filename}
name of output file (default is infile.bsi);
- -v
verbose mode
- -V
print the time/date the tool was built

NOTES:

- This tool is only used on a monitor binary if it is being inserted into a system that is using the boot-sector safety

EXAMPLES:

- `ibsi -b 0x4000 -i 0x50 -c 0x60 -o monitor.bsi monitor.bin`
The monitor.bin file has been built to exist starting at location 0x4000. The point of insertion into the file is 0x50 bytes from the start and the point at which to start the CRC calculation is 0x60 bytes from the start of the file. The new file is called monitor.bsi.

EXIT STATUS:

0 if successful

1 error

MACCRYPT

NAME:

maccrypt

SYNOPSIS:

Convert a MAC address to an encrypted string.

USAGE:

maccrypt [-f:uV] "MAC Address"

DESCRIPTION:

This tool provides the "backdoor" access to a password protected MicroMonitor environment. Typically, this would be used if a MicroMonitor based system was configured with password protection but the users forgot the passwords.

OPTIONS:

- -f {filename}
This option overrides the default data table used by the proprietary scrambler.
- -u
Use UNIX crypt() instead of proprietary. Use of this option depends on whether the monitor was built to use the proprietary version or standard Unix version of crypt().
- -V
display the version (build date) of the tool

EXAMPLES:

- maccrypt 00:60:1D:02:0B:FD
Print out an encrypted string that will be accepted as a valid password by a target running MicroMonitor with that MAC address.

EXIT STATUS:

0 if successful

1 if error

MONCMD

NAME:

moncmd

SYNOPSIS:

Interface to the monitor's UDP command interface port.

USAGE:

moncmd [-p:w:qrV] {target ip-address} "target command string"

DESCRIPTION:

This tool allows the user to communicate with a target running MicroMonitor through ethernet/UDP. The monitor has a server waiting for incoming command strings on port 777 (or whatever port is assigned in the MCMDPORT shell variable). It will process the command string sent via moncmd just as it would process a command from the console interface.

OPTIONS:

- -p {##}
override default port number of 777;
- -w {##}
number of seconds to wait for response, after which a timeout will occur (default is 10 seconds);
- -q
quiet mode (don't print the timeout message);
- -r
retry if command-not-received. Note that if the command is received, but not completed the retry does not apply.
- -v
verbose mode (print the 'Sending...' string);
- -V
display the version (build date) of the tool

EXAMPLES:

- moncmd 135.3.94.136 "tfs ls"
Send the command string "tfs ls" to a target at IP address 135.3.94.136
- moncmd -w0 135.3.94.136 "reset"
Tell the target at IP address 135.3.94.136 to reset. Note that waittime is set to zero because there will be no response, since we are resetting the target firmware.

EXIT STATUS:

- 0 if successful
Command was received, executed and completed.
- 1 if timeout waiting for command completion
In this case, moncmd verified that the command was received by the target, but there was no command completion handshake. This would be the type of timeout that would occur if moncmd was used to issue a target reset.
- 2 if error
Some kind of usage error typically.
- 3 if timeout waiting for command reception
In this case, moncmd could not even verify that the command was received by the target.

MONSYM

NAME:

monsym

SYNOPSIS:

Convert a file of tabulated data into the format used by MicroMonitor's command-line symbol retrieval.

USAGE:

monsym [-d:l:p:s:vV] {filename}

DESCRIPTION:

This tool does some very simple column re-arranging of tabulated data in files. The monitor has the ability to process "symbols" (see discussion on shell variables and symbols). This is done through a file in TFS that is assumed to contain lines of data where each line is of the format:

{symbol} whitespace {data}

Typically, the output of nm or some other tool that generates a readable listing of symbols contains a lot of information that is of no value for basic symbol lookup. This tool simply extracts the data field and symbol-name field from each line of the input file and prints to stdout in the format above. This file can then be transferred to a target and used by the monitor for symbol lookup.

OPTIONS:

- -d {col #}
column number from which the data is to be extracted from the incoming file (default = 1);
- -l {size}
maximum size of an incoming line (default = 64);
- -s {col #}
column number from which the data is to be extracted from the incoming file (default = 3);
- -p {data prefix string}
a prefix attached to the data prior to output;
- -S {x|d}
sort the output based on the content of the data column being hex (x) or decimal (d).
- -v
verbose (debug) mode
- -V
print the time/date the tool was built

NOTES:

- This tool is typically used on a file that was created by "nm" (or some equivalent).

EXAMPLES:

- monsym -p0x ias.sym
From the incoming file ias.sym, output column 3 followed by whitespace and column 1. The data from column 3 is prefixed by 0x. For example if the ias.sym file contained...

```
00018000 T _sysInit
00018000 t gcc2_compiled.
0001805c t vxpbil
0001809c t gcc2_compiled.
0001809c T sysSerialIntEnable
0001811c T sysSerialIntDisable
```

MicroMonitor
Host Based Tools

```
0001819c T sysSerialHwInit  
00018308 T sysSerialHwInit2  
00018364 T sysSerialChanGet  
000183bc T sysSerialReset
```

then the result would be...

```
_sysInit 0x00018000  
gcc2_compiled. 0x00018000  
vxpbil 0x0001805c  
gcc2_compiled. 0x0001809c  
sysSerialIntEnable 0x0001809c  
sysSerialIntDisable 0x0001811c  
sysSerialHwInit 0x0001819c  
sysSerialHwInit2 0x00018308  
sysSerialChanGet 0x00018364  
sysSerialReset 0x000183bc
```

EXIT STATUS:

0 if successful

1 error

NEWMON

NAME:

newmon

SYNOPSIS:

Automatically update a monitor's boot flash to a new version.

USAGE:

newmon -[A:B:p:w:vV] {target ip-address} {binary filename}

DESCRIPTION:

This tool allows the user to upgrade a target's boot monitor.

<blink>WARNING!!!</blink> This command will take the requested file and assume it is a binary file destined for the boot sector of the target device. If you give it the wrong file, you will destroy your boot sector!!!

OPTIONS:

Note that these options only apply to the .exe version. The script does not have any options.

- -A {###}
override default use of \$APPRAMBASE shell variable as location for tftp transfer and source for flash ewrite.
- -B {###}
override default use of \$BOOTROMBASE shell variable as target for flash ewrite.
- -p {##}
override default port number of 777;
- -w {##}
number of seconds to wait for response, after which a timeout will occur (default is 10 seconds);
- -v
verbose mode (print the 'Sending...' string);
- -V
display the version (build date) of the tool

EXAMPLE:

- newmon 135.3.94.136 mon403.bin
Place the content of mon403.bin into the boot flash of the target at address 135.3.94.136

EXIT STATUS:

Note that these exit status apply to the.exe version only.

- 0 if successful
Command was received, executed and completed.
- 1 if timeout waiting for command completion
In this case, moncmd verified that the command was received by the target, but there was no command completion handshake. This would be the type of timeout that would occur if moncmd was used to issue a target reset.
- 2 if error
Some kind of usage error typically.
- 3 if timeout waiting for command reception
In this case, moncmd could not even verify that the command was received by the target.

TITLE

NAME:

title

SYNOPSIS:

Place a string of text in the title bar of a WIN95 console window.

USAGE:

title "string"

DESCRIPTION:

This tool allows the user place a string of text in the title bar of the current WIN95 console window. This is useful when multiple console windows are opened for various purposes and some or all have been minimized; such that only the content of the title bar is visible. If no argument is specified, then title will look for the content of the shell variable "TITLE" and use that as the text; if not found, then the title bar is cleared.

EXAMPLES:

- title "IAD development"
Place the string *IAD development* into the title bar.
- title
If the shell variable TITLE exists, then place the content of that variable into the title bar; else, simply clear the title bar.

EXIT STATUS:

- 0 if successful
- 1 if timeout waiting for command completion
- 2 if error
- 3 if timeout waiting for command reception

TFTP

NAME:

tftp

SYNOPSIS:

Run a tftp client or server.

USAGE:

tftp [-ad:qr:t:T:vV] {target ip_address} {get | put | svr} [source] [destination]

DESCRIPTION:

This tool is similar to many other tftp clients. It runs standard TFTP, but with a few added features that make it more useful. For example, the status of the download is available so that the user can determine if it is in progress, or hung. Also, a single-user server is built into the same executable. Note that the Sun-Solaris version of this does not currently support the 'svr' capability.

OPTIONS:

- -a
use netascii mode for 'put'; default is octet (binary);
- -d {per-packet-delay}
per-packet-delay (milliseconds) inserted between each client-based outgoing packet .
- -q
quiet mode (without this, tftp will print one dot per block transferred);
- -r {rfc2349_tsize_off} use this to disable the client's use of the TSIZE extension as is specified in RFC 2349. This is applicable to "put" only.
- -t {timeout}
inactivity timeout used by server; default is 60 seconds after which any transaction in progress is aborted.
- -T {test_setup_string} Used for debugging tftp clients/servers.
- -v
verbose mode (display details of each TFTP packet)
- -V
display the version (build date) of the tool

EXAMPLES:

- tftp svr
Run as a single-user server with the current working directory being relative to the directory in which the server was started. Note that this server will definitely get confused if multiple clients are thrown at it. Its purpose is of single-system lab testing only. Also note that this server option is not available on the SUN version of tftp.
- tftp -t 120 svr
Run as a single-user server with the current working directory being relative to the directory in which the server was started. Allow for up to 120 seconds of inactivity before timing out. Inactivity refers to an active transaction (WRQ or RRQ) but no data transfer.
- tftp -a 135.3.94.136 put monrc monrc,e
Copy the file named monrc (currently on the host) to TFS on the target at IP address 135.3.94.136. The file flags on the destination will be 'e' (executable), meaning that it is an executable script.
- tftp -r rfc2349_tsize_off -a 135.3.94.136 put monrc monrc,e
Same as above, but without the "tsize" extension.
- tftp 135.3.94.136 get afile
Retrieve the file 'afile' from the target and place it on the host with the same name.

EXIT STATUS:
0 if successful
1 if error

WHENCE

NAME:

whence

SYNOPSIS:

Determine the execution path (based on PATH shell var) of an executable.

USAGE:

whence {program_name}

DESCRIPTION:

Similar to the "whence" capability built into KSH, but not readily available on WIN95 machines.

EXAMPLES:

- whence vi
Print the full path (if found) of the vi.exe executable.

EXIT STATUS:

0 if successful

1 if error