P C O M M
Information
Systems Inc

ROBO, v0.37, November 19, 1988

An automated mass mailing program for FidoNet mail in Fido, Opus, BinkleyTerm, or compatible environments.

1411 Fort St, Suite 2001, Montreal, Que H3H-2N7, Cda

Contents

Requirements

Operating System ...:	DOS 3+ (not tested with lower vers)
Video ....:	Generic MSDOS
Keyboard ....:	Generic MSDOS
Memory ....:	Less than 100k when naked and wet.
Net Mail ....:	Fido format message system

Feature Summary

- Send inline or pre-written text to one or more destinations.
- Send one or more files to one or more destinations.
- Send both text and files to one or more destinations.
- Creates traditional message FILE ATTACH or direct FLO files.
- Optionally inserts text in only the 1st of multiple attaches.
- Variable maximum of files per file attach message.
- Run any one of several procedures or "groups" in a control file.
- Nested IF/ELSE/ENDIF conditional control file constructs.
- IF command can test for files, dates, Julian date, and time.
- Calc most recent Julian of any week day. (Eg, most recent Fri)
- Run DOS commands via Shelling.
- Can "Track" file sends to prevent subsequent duplicate mailings.
- Log activities to a log file.
- Append comments to Robo log file.
- Insert text entries into external text files or file listings.
- Display status message to the screen.
- Can display Track Log Report.
- Zone support for message addressing by IFNA kludge method.
- Different priorities for messages and optional direct FLO's.

Packing List

-Readme.1st	Brief unpacking instructions.
ROBO.PRN	This documentation (Printer format with form feeds)
ROBO.CTL	Sample control file with several examples.
ROBO.EXE	Main program
ROBO.REG	Product Registration Form

SEND.LST	Sample of address file.

Introduction

Robo is a batched oriented utility program for use by Fidonet compatible BBS operators to broadcast a message to multiple destinations with an optional set of file attaches. Its operation is controlled by the commands within its control file which is interpreted at run time in a sequential and linear fashion.

A simple set of commands are used to define and send the messages while a set of more powerful, but optional, commands can be used make message groups and allow conditional execution based on time or file presence.

Typically, Robo is used in two scenarios: To do a single mass-mailing for a special purpose or as an automated procedure to run automatically on a regular basis.

For a quick idea of what's need to do a mass-mailing, review the control file examples in the section: Sample Control Files.

Theory of Operation

As Robo scans its control file, it builds an "image" of the message in memory. A list of destinations, a list of any files to be sent, as well as the actual text to use as the body of the message are all stored in dynamically allocated memory.

When a SEND command is encountered, this image is then used to produce the multiple messages in the network mail directory. Any files to be sent are handled as file attachments to messages using the subject field for the file spec with the special FILE ATTACH flag set.

Robo will send one message to each destination specified. If files are specified, then there will be one message per file per destination. Since the file attachment process uses the SUBJECT field to hold the file specification, Robo inserts the subject text in a "Re:" line at the top of the message body for viewing by the recipient.

Robo's list of commands is quite extensive and one might easily think of it more as a batch file processor. With its SHELL command it can manage file movements and DOS executions quite well. For instance, it could easily check for the existence of an expected newsletter. If it was found in the inbound directory, it could then copy it to its normal resting place, check that it made it there, delete the original, generate a series of mass attaches to a list of destinations with accompanying text, ensure that the file wasn't already sent, and insert a file entry into the destination directory's file list.

Command Line Switches

Robo has a series of optional command line switches that tailor its operation. As they are all optional, it will run without any being specified using default values. Using the help switch, -H or -?, will display all available switches and their default values.

COMMAND LINE SWITCHES

-h, -?..............Display Help

Display command switch help summary.

-C{path}............Configuration file path

For when you wish to use a control file other than the default one as is convenient for multiple configurations.

-L{path}............Progress log file path

Sets an alternate to the default progress log file.

-T{path}............Send file track log path

Sets an alternate to the file tracking log as might be handy when you wish to test but don't want to destroy the current track log.

-E..................Control file scan echo

Echo the scan of control file to the screen.

-V{detail level}....Logging detail level

Sets the level of detail that will be written to the progress log file. A '0' suppresses all logging, a '1' will give minimal reporting and a '2' gives maximum detail.

-R..................Dump track log to screen

Displays the contents of the current File Tracking Log in the format, below, and then exits without running.

```
┌─────────────────────────────────────┬────────────────┐
│        Tracked File Information     │    Delivery    │
├─────────────────────────────────────┼────────────────┤
│ Pos Name        Date    Time  Size  │   Date    Time │
│   1 ROBO.CTL  88/09/19 21:54   2458 │ 88/09/19 22:30 │
│   2 ROBO.EXE  88/09/19 22:51  54376 │ 88/09/19 22:52 │
└─────────────────────────────────────┴────────────────┘
```

-F..................Force direct FLO file creation

For Opus, Fido, or Binkley style outbound message environments, this switch enables the optional, direct creation of file attach lists (FLO files) instead of using the traditional method of storing the file name in the message subject line and setting the message's FILE ATTACH flag.

This switch should only be used in traditional environments when the mail packetizer does not seem to be creating file attaches properly and you want Robo to create the file attach list for you. It is NOT otherwise needed. It's use in other environments is undefined and may result in no files being sent as the file lists (FLO) that are created may not be recognized or compatible.

Without this switch, the traditional message ATTACH is used and requires an external mail packetizer or "bundler" to detect any file attachments from the message flags and create the necessary file lists (FLO). In the Binkley and Opus environments, for example, the OMMM program performs this function.

Also note that this switch simply sets the MAXATT value (see commands) to zero and any MAXATT commands in the control file will override this setting.

Command Set Summary

Robo is essentially programmed via its control file. The control is scanned in a sequential manner, executing the commands it finds there. Although designed initially for mass-mailing, this programming can accomplish a good number of other activities. The command set is fairly large and can be thought of as falling into the categories of FLOW-CONTROL, CONFIGURATION, COMPOSITION, EXECUTION, and MACROS.

FLOW CONTROL

Comments	Remarks started by (;), blank lines ignored
IF / [ELSE] / ENDIF	IF DATE, Julian, DAY, HOUR, MINUTE, EXIST
GROUP name	Marks start of a command group

CONFIGURATION

OUTbound path	Sets the outbound path (for attach lists)
MAIL path	Sets the mail path (for netmail messages)
MAKEFLO	Obsolete and replaced by "MAXATT 0" command.
ONETEXT {YES│NO}	Limits text inclusion to first attach only.
MAXATT {0-8}	Sets maximum number of attaches per message.

COMPOSITION

SUBJect "subject"	Sets the Subject of message
FROM address "name"	Sets the Author's address and name
TO address "name"	Adds a destination address and name
TO LIST file name	Adds a list of dest addresses and names
FILE file name [Track]	Adds a file to send with optional tracking
TEXT textfile	Sets message text from a text file
TEXT /text/ ENDTEXT	Sets message text from control file
SET msgflag msgflag	Sets message flags
CLEAR item item	Clears flags, message items, etc.

EXECUTION

SHELL "DOS command"	Executes a DOS command
SEND	Triggers mail production
EXIT	Terminates processing
DISP "text remarks"	Displays remarks on local screen
LOG "text remarks"	Appends remarks to native log file
FLAP file targ new	Inserts "new" into "file" at "targ"

MACROS

Macros are expressions that, when encountered in control file statements, are replaced by their value as listed below. Note that the "fancy" brackets are part of the macro definition and must be included whenever inserting a macro.

The first group of macros, such as {fn}, are special in that they are reset by each true IF EXIST statement. If used before any true IF EXIST statement has executed, the substitution will be unpredictable.

{fn}	Trimmed non-pathed file name	"test.arc"
{ff}	Pathless filename padded to 12 chars	"test.arc"
{fr}	Trimmed root file name	"test"
{fe}	Trimmed file extension	"arc"
{fp}	Trimmed file path	"c:\files\"

{ymd}	Current date as yymmdd	"880131"
{mdy}	Current date as mmddyy	"013188"

The next two macros are substituted with either the current Julian date as two or three digits. The 3 digit form includes leading zeroes when necessary and the 2 digit form is the just LAST 2 digits of the three digit form.

{jd3}	Current Julian as ###	"032"	(Feb 1st)
{jd2}	Current Julian as ## (last 2)	"32"	(Feb 1st)

The next two macros are substituted with the Julian date of the most recent day of the week whose week position is specified, numerically, in the second character position (Monday = 0). If the indicated day is the same as the current day, then the current day's Julian is used, not the previous week's. For example, to insert the last two digits of the most recent Friday's Julian date, you would use "{j5n2}".

The following two examples fictitiously assume that the most recent Friday (day 5) was Julian 123.

{jnd3}	Julian ### most recent Nth day.	{j5d3} ==> "123"
{jnd2}	Julian ## most recent Nth day.	{j5d2} ==> "23"

Command Set Details

FLOW: Comments (inline comments)

```
; This is a comment
(some statement)      ; This is also a comment.
```

The control file can contain comments by preceding the comment with a semi-colon. Comments can also appear to the right of commands.

Blank lines are ignored and can be liberally used for visual balance.

FLOW: IF/ [ELSE] /ENDIF (conditional blocks)

```
Syntax: IF [not] object [comparison] testvalue
             statements
             statements
          ELSE             \
             statements     >- Optional ELSE for negative cases
             statements    /
          ENDIF
```

Supported objects and their comparisons:

```
   IF [not] DATE       { LT | LE | EQ | GE | GT } yymmdd
   IF [not] Julian     { LT | LE | EQ | GE | GT } Julian (Julian day)
   IF [not] DAY        {Mon | Tue | Wed | Thu | Fri | Sat | Sun}
   IF [not] TIME       { LT | LE | EQ | GE | GT } {hhmm | hh:mm}
   IF [not] HOUR/HR    { LT | LE | EQ | GE | GT } hh
   IF [not] MINUTE/MIN { LT | LE | EQ | GE | GT } mm
   IF [not] EXIST      {filepath}
```

These commands control the flow of control file execution with the constructs of IF, ELSE, ENDIF, and GROUP.

The objects and comparisons can NOT be mixed freely as can be done with certain higher level languages. Only the combinations listed above are available.

The IF/ENDIF construct with optional ELSE subclause allows you to test various objects and, based upon the result, execute a set of statements if the test is successful or, using ELSE, if it fails.

IF blocks can be nested quite deeply and a NOT modifier is available to check for negative conditions.

When you wish to execute a set of commands when a condition is true, use IF/ENDIF.

When you wish to execute a set of commands when a condition is NOT true, then use IF NOT/ENDIF.

When you wish to execute one set of commands when a condition is true and another set when it is false then use IF/ELSE/ENDIF.

```
Ex:  IF DAY Friday
     DISP "It's Friday"
     IF exist c:\test\abc.arc
          DISP "ABC.ARC exists on Friday!"
        ELSE
          DISP "ABC.ARC is missing on Friday!"
        ENDIF
     ELSE
          DISP "It aint't Friday - Drats!
   ENDIF
```

FLOW: IF EXIST MACROS (macros set by IF EXIST)

Whenever an "IF EXIST file name" command is executed which evaluates to a TRUE condition, the file spec is parsed into a set of variables which are part of the set of macros available for inline insertion in any command.

Each subsequently true IF EXIST command resets those variables. If the IF EXIST is false then the file name macros are to be considered as undefined and unusable. The macros that are set by (true) IF EXIST commands are ...

{fn} ... The full file name without path.
{ff} ... The full file name padded with trailing blanks.
{fr} ... The root of the file name without extension.
{fe} ... The extension of the file name only.
{fp} ... The file path w/o trailing backslash.

FLOW: The GROUP (define GROUP label)

Syntax: GROUP label

A related, block control verb is GROUP. It basically allows you to label one or more point in the control file. When a text string is entered on the command line when running Robo, it is taken as the GROUP name to look for an execute.

What happens, when running with a group name is that all statements prior to the first GROUP name in the control file are executed. After that, all statements are skipped until the given group name is found. Then all subsequent statements up to the end of file or the next group name are executed.

```
 Ex:      statements  \__ Always executed
          statements  /

       GROUP Name1
          statements  \__ Only when Name1 given
          statements  /

       GROUP Name2
```

```
          statements  \__ Only when Name2 given
          statements  /
```

FLOW: EXIT (EXIT program)

Syntax: EXIT

EXIT causes an immediate termination of the control file. It is one of those commands that seems like it should be there but one whose use I've yet to see. <grin>.

CONFIG: OUTbound (define OUTbound mail path)

Syntax: OUT "outbound mail path"

Sets outbound mail path (aka holding area)

```
Ex: OUT c:\mail\out
```

CONFIG: MAIL (define MAIL message path)

Syntax: MAIL "network message area"

Sets the path where all netmail messages are stored. Also called the "netmail", "fidonet messages", or "inbound mail" area.

```
Ex: MAIL c:\mail\in
```

CONFIG: ONETEXT (limit text inclusion to first attach)

Syntax: ONETEXT {YES|NO}

This command controls the handling of message text insertion in those cases where multiple messages are required to send all attached files.

When set to YES (the default), any defined text will only be inserted into the first attach message. Any subsequent messages that are required for the balance of files will be the minimum necessary. This is useful when you have a large text that you do not want to be needlessly duplicated during a multiple file send wherein several messages are required to attach all files.

When set to NO, then the defined message text will always be inserted in every message to every destination. This might be useful when you have a small set of handling instructions or a notice to go out each and every attachments.

CONFIG: MAKEFLO (OBSOLETE) (use FLO file attachment method)

Syntax: MAKEFLO

This command is no longer supported as it has been completely replaced by the "MAXATT 0" command variant.

CONFIG: MAXATT {0-8} (set maximum attaches per message)

Syntax: MAXATT {0-8}

This command sets the maximum number of file paths that will be attached to each attached file message and overrides the default which is set to 3.

If the subject field, where attach file paths are stuffed, becomes full before this maximum is reached, then as many additional messages are created as necessary to complete the list of attached files.

When set to zero, it has the special meaning of forcing the direct creation of file attach lists in the outbound directory and then only creates a single covering message.

> !WARNING! This option is ONLY applicable to Opus, Binkley, and compatible message environments and controls the way in which the file sending process is accomplished. It is generally NOT necessary to use this option.
>
> Normally, WITHOUT this option being used, all files are sent by marking one or more message with a "FILE ATTACH" notation and having a separate mail packetizing function take care of the rest.
>
> Using this option in Binkley, Opus, or compatible message systems takes the file sending process one step further with the direct creation of file lists or "FLO" files. This bypasses the need for that function in the mail packetizer. Any accompanying message text is sent as a cover letter without any file attachment information but with a leading text remark listing the accompanying files. The files will be still be sent, however, since the FLO file is all that is needed to send the file.
>
> Use of this option is ONLY recommended when you wish to bypass the file sending logic of your mail packetizer for some reason, eg, when the packetizer doesn't handle file attaches properly.

COMPOSE: SUBJECT (define SUBJECT field)

Syntax: SUBJECT "subject of message"

The SUBJECT command sets the subject field of the message until it is CLEAR'd or another SUBJECT command is encountered.

```
Ex: SUBJECT "Region 99 Bulletin"
```

COMPOSE: FROM (define FROM: field)

```
Syntax: FROM address "sender's name"
```

The FROM statement sets your network address and name which remains set until it is cleared with a 'CLEAR FROM' statement or reset by a subsequent 'FROM' statement.

The address format is of the form {zone}:{district}/{node}.

The name field is either enclosed in double-quotes or has any spaces replaced with underscores.

```
Ex: FROM 1:167/1 "Tom Kashuba"

Ex: FROM 1:167/1 Tom_Kashuba
```

COMPOSE: TO (define TO: field)

The 'TO' command is additive and adds the given address and name or list of such to an internal destination table. When the SEND command is encountered, then the mail is sent to everyone in that list.

```
Syntax 1: TO {address} {name}

Syntax 2: TO LIST {filename}
```

In both forms, the address is the FidoNet address of the destination with optional zone field. Excluding the zone defaults it to the zone in the last 'FROM' statement. The address field is either enclosed in double-quotes or has its spaces replaced with underscores.

Syntax 1 adds the given destination address and name to the internal destination list.

Syntax 2 adds all the addresses and names from the given disk file to the internal destination list.

Multiple 'TO' statements are allowed and can be a mix of both types.

CAVEAT: A SEND statement does *NOT* clear the destination list. Use the "CLEAR TO" statement for that purpose. For example:

```
TO 167/9999 "Jane Doe"

TO LIST list1.adr

TO 2:167/8888 John_Deer

TO LIST list2.adr
```

A destination file list is a pure text file with one FidoNet address and user name per line. Eg, a destination file with three addresses might look like:

```
999/101 "Allan Always"

2:999/201 Roberta_Rubson

1:999/301 "Wanda Waxer"
```

The use of underscores to represent spaces in the names of address lists allows the direct use of Opus 'bombing run' address files.

COMPOSE: FILE (define FILEs to send)

Syntax: FILE file name [TRACK]

Each FILE command adds a file name to an internal list that is appended to with each subsequent FILE command. This file list is then used to create the required file attaches when a SEND command is encountered.

TRACK Option

If the optional "TRACK" verb follows the file name, then the given file is logged in a special file send tracking log so see if it has already been sent. If it has, a message is output and the file is skipped. If it has not, then the file is sent and some identifying info is stored in the track log for future reference.

Tracking Algorithm

The track log stores the name, size, date, time, and random check sum of up to 64 files. To check for dupes, Robo first compares the file sizes. If the same, it then compares the random check sums. Since Robo always needs the check sum for future cases where the file sizes might be the same, it always computes it.

The currently used algorithm for the random checksum is to take a cumulative summing of the bytes at 100 equally spaced points throughout the target file. If the file is equal to, or smaller than 100 bytes, then all bytes are summed.

This type of check summing combines speed with a moderate level of uniqueness as there are some 255 * 100 combinations, ie, 25,500 possible values. The cases where the file size and all 100 test points are the same should be quite rare. In this way, the check sum process will not take any longer for very large files.

Example Use of Track

The TRACKing function will only be of use when the name of a file (without path) is explicit (without wild cards) and is meant for use in specific scenarios where strict file naming conventions are used as exemplified by the FidoNet "NodeList.A##", "NodeDiff.A##", and Fnews###.ARC files which are distributed each week.

Tracking will most often be used in those scenarios where you have to pass files (that you receive on a regular basis) on to others but stand the chance of receiving duplicate sends from one or more potential suppliers. Since there are only one or two suppliers but you may need to pass on to many, the resulting and needless phone charge is amplified by the number of destinations that you send to and a large transfer cost can be saved by stopping duplicate sends.

COMPOSE: TEXT (define TEXT to send)

```
Syntax 1: TEXT textfilename

Syntax 2: TEXT
          lines of text
          ENDTEXT
```

Sets the text to be sent in the body of the mass-mailed message. The text can be placed in the control file between a TEXT/ENDTEXT statement pair or it can be taken from an external file. There should only be one TEXT statement per message. Multiple TEXT verbs would only re-assign the same text over again, replacing the previous text with the last setting.

```
Ex: TEXT
       Hello.
       How'ya doin'
       Good Boy!
    ENDTEXT

Ex: TEXT bulletin.001
```

"A carriage-return by any other name ..."

When Robo inserts the defined message text, it tries its best to convert the text file into standard FidoNet message format.

Each input text line is inserted into the message body with a trailing space plus "soft" carriage return (Hex 8D) unless ...

- The current line is null (empty).
- The next line begins with a space or control character.

... in which cases it terminates the line with a "hard" end of line sequence (Hex 0D0A). This should allow for tabular or formatted message text to appear as intended on most systems.

COMPOSE: SET (SET message attributes)

`Syntax: SET [Hold | Crash | Norm] [Priv] [Kill]`

Crash	Make message highest priority.
Hold	Hold message for pickup.
Normal ...	Removes any Crash or Hold status.
Private ..	Make message private.
Kill	Delete message when sent.

The SET command allows the setting of the Fido type message flags until they are CLEAR'd. As they are boolean in nature, they cannot be changed with another SET. SET only turns flags ON.

`Ex: SET Private Crash Kill`

COMPOSE: FSET (SET FLO attributes)

`Syntax: FSET [Crash | Hold | Norm | Same]`

Crash	Make direct FLO file as CRASH.
Hold	Make direct FLO file as HOLD.
Normal ...	Make direct FLO file as NORMAL.
Same	Make direct FLO file same as message (initial default).

The FSET command is ONLY used when the special "MAXATT 0" option is set and allows the setting of the directly created file attach lists (FLO's) to a different priority than the matching message. This is used in those cases where one might like to immediately send out a covering message about some files that are being held for the subsequent call back and pick up of message's recipient.

`Ex: FSET Hold`

COMPOSE: CLEAR (CLEAR some or all message fields)

`Syntax: CLEAR [Subj] [To] [File] [Priv] [Attr] [Text] [All]`

The CLEAR command clears (resets) individually set items as well as certain groups of items.

Subj	Clear current Subject
To	Clear current destination list
File	Clear current file list
Text	Clear current message text
Priv	Turn off PRIVATE flag
Kill	Turn off KILL-after-SEND flag
Attr	Resets message attributes [Priv,Kill,Crash,Hold]
ALL	Clear everything except the FROM field

```
Ex: CLEAR File    ; Clears current list of files to send
Ex: CLEAR To      ; Clears current list of destinations
```

EXECUTION: SHELL   (SHELL to dos and run command)

Syntax: SHELL "dos command line"

The SHELL command invokes a second copy of the command interpreter and executes the given command. Its action is immediate.

Ex: SHELL "Copy c:\netfiles\Nodelist.* k:\fidonet*.*"

EXECUTION: SEND   (SEND what's been constructed)

Syntax: SEND

The SEND command commences an actual mass mailing based upon everything that has been SET prior to its invocation. It does not alter any of the settings so, for example, two SEND commands in a row would cause two identical mass-mailings. It is typically the last command in a mass mailing command set.

Ex: SEND

EXECUTION: DISP   (DISPlay text to console)

Syntax: DISP "text to display"

Displays the given text string on the console for running comments.

Ex: DISP "Executing NodeList Processing"

EXECUTION: LOG   (add text to LOG)

Syntax: LOG "text to display"

Write the given text string to the log file, if it is open, with the current date and time prefixed in Robo format.

Ex: LOG "NodeList Processing"

EXECUTION: FLAP (File List APpend)

Syntax: `FLAP filelist "target text" "insert text"`

Appends, prefixes, inserts or replaces entries in a text file list or log. The "target" is searched for in the given file and the "insert" is placed before, in place of, or after it.

If the "target" is prefixed with "<", "=", ">" then the "insert" is placed before, in place of, or after line in which the "target" was found. The matching of the "target" text always starts at the left column.

If the "target" consists only of "<" or ">" without any other text, then the "insert" text will be inserted before the first line or appended after the last line in the file, respectively.

Ex: `FLAP "k:\fidonet\files.bbs" "<NODELIST" "Latest Nodelist -robo"`

... the above would insert "Latest Nodelist -robo" BEFORE the first line in the file that started with "NODELIST" thereby maintaining a most-recent-first order.

Ex: `FLAP "k:\fidonet\files.bbs" ">" "Latest Nodelist -robo"`

... the above would just add the given text to the end of the file.

Ex: `FLAP "k:\fidonet\files.bbs" "=NODELIST" "Latest Nodelist -robo"`

... the above would overwrite (replace) the first NODELIST line in those cases you might wish to keep the most current file the given file area.

Control File Examples

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Example 1:  Send message (without files) to a list of destinations
;
;  Notes:  1. A minimal message is still created for "Encl: " record.
;          2. Shows how to use inline text definition.
;          3. Sets destinations manually with multiple "TO" commands.
;
MAIL  c:\mail_in\                         ; Where to put messages
OUTBOUND c:\mail_out\                     ; Where to put file lists
FROM  1:167/1 "Tom Kashuba"               ; YOU (Zone required)
;
TO    999/101 "Allan Allgood"             ; Def a destination.
TO    2:999/201 "Cynthia Comely"          ; Def a destination.
TO    999/333 "Rona Roxboro"              ; Def a destination.
SUBJ  "Sample Broadcast Message"          ; Set Subj:
TEXT                                      ; Start body text (inline)
  Here's your weekly update
  Please apply as necessary!
ENDTEXT                                   ; End of inline text
SET PRIVATE KILL                          ; 'Priv' & 'Kill'
SEND                                      ; Initiate actual send
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Example 2:  Send files (without text) to a list of destinations
;
;  Notes:  1. Sets destinations using a file of addresses and names.
;
MAIL  c:\mail_in\                         ; Where to put messages
OUTBOUND c:\mail_out\                     ; Where to put file lists
FROM  1:167/1 "Tom Kashuba"               ; YOU (Zone Required)
;
TO    FILE  c:\netdata\dest.lst           ; Def destination list.
FILE  c:\files\yoo.arc                    ; Def a file to send.
FILE  c:\files\hoo.arc                    ; Def a file to send.
SUBJ  "Sample Broadcast Message"          ; Set Subj:
SET PRIVATE KILL                          ; 'Priv' & 'Kill'
SEND                                      ; Initiate actual send
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Example 3:  Send files AND text to a list of destinations
;
;  Notes:  1. Sets destinations using a file of addresses and names.
;          2. Shows us of external text file for message text.
;
MAIL  c:\mail_in\                         ; Where to put messages
OUTBOUND c:\mail_out\                     ; Where to put file lists
FROM  1:167/1 "Tom Kashuba"               ; YOU (Zone Required)
;
TO    FILE  c:\netdata\dest.lst           ; Def destination list.
FILE  c:\files\yoo.arc                    ; Def a file to send.
FILE  c:\files\hoo.arc                    ; Def a file to send.
SUBJ  "A mass mail example"               ; Set Subj:
TEXT c:\netdata\bull.txt                  ; Use external msg text.
SET PRIVATE KILL                          ; Set PRIV, Kill after send
SEND                                      ; Compose msgs/file lists
```

```
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;
   ;  Example 4:  Heavy-Duty example that acts as an automated nodediff
   ;              distribution system.
   ;
   ;  An example of using a "group" of commands that might be one
   ;  of several in a larger master control file for general use.  The
   ;  group is labelled "Diff" and would execute when a command line of
   ;  "Robo Diff" was used.
   ;
   ;  This particular example demonstrates how Robo might be used as a
   ;  daily event to look for a newly received "NodeDiff.*" file for
   ;  further redistribution to a list of destinations.
   ;
   ;  Note the extensive use of the IF EXIST command and the found file
   ;  macros of {fp}, {fn}, etc.
   ;
   MAIL  c:\mail_in\                         ; Where to put messages
   OUTBOUND c:\mail_out\                     ; Where to put file lists
   FROM  1:167/1 "Tom Kashuba"               ; YOU (Zone Required)
   ;
   GROUP Diff                                ; Set GROUP name
   ;
   IF EXIST C:\Fnet\NodeDiff.A*              ; Has a NodeDiff arrived?
                                             ; Yes, set macros like {fn}
   ;
      SHELL "Copy C:\Net\NodeDiff C:\Files\*.*" ; copy to dest dir

      IF EXIST C:\Files\{fn}                ; Is it there after copy?
                                            ; {fn} => last found file
                                            ; Resets file macros eg {fn}

         SHELL "Del C:\Net\{fn}"            ; Yes: Delete Orig but DON'T
                                            ; use {fp} (the GOOD path).
         FILE  C:\Files\{fn} Track          ; Set file to send & "Track"
                                            ; & avoid future dupe sends.
         TO    LIST other.adr               ; Set list of destinations.
         SUBJ  "Weekly NodeList Update"     ; Set subject

         TEXT                               ; Set msg body text (inline)
    .------------------------------------.
    |  H o t   O f f   T h e   W i r e ! |
    |------------------------------------|
    | Here's the weekly nodelist update. |
    | Please, pass it on to your nodes.  |
    `------------------------------------'
         ENDTEXT                            ; end msg body define

         SET PRIVATE KILL                   ; Set PRIV, Kill after send
         SEND                               ; Compose msgs/file lists

; Now we use the File List Update feature to automatically insert the
; new nodediff in a file list in the destination directory.  The "<" in
; the "<NODEDIFF" search phrase inserts the text BEFORE
; the first line found with "NODEDIFF" in it.

         FLAP "C:\Files\Files.Bbs" "<NODEDIFF" "{ff} {ymd} NodeDiff"

      ENDIF ; End of file found/moved/sent block
```

```
   ENDIF ; End of file found block
```

Licensing Agreement

This is a commercial product which may be freely distributed to any person, group, or organization, on a trial basis, as long as it is distributed in its original, unadulterated form with all of the files listed in the section, Packing List.

All persons, groups, or organizations which come to use this product are required to pay a licensing fee of $10 for each computer, work-station, or terminal from, or on, which it is run. The entire fee is applied against the costs of its development, support, future enhancements, and the development of other related products.

Those using the product and, therefore, accepting these terms are asked to complete and return the registration questionnaire in this package with their license fee.

Bulletin boards, information systems, Special Interest Groups, clubs, and software libraries may also carry and distribute it as long as no fee or compensation is charged for any part of it other than minimal and unrelated charges for diskette distribution or on-line fees.

At no time is it to be presented as having any value to anyone other than the author nor may it be presented as an added value to any products or services that are not the author's unless prior and explicit arrangements have been made.

Due to the wide range of possible target environments, the minimal cost of the product, and the fact that a free trial is allowed, the author disavows all responsibility for any and all damages or liabilities resulting from its possession or use but is eager to receive any problem reports for possible correction in subsequent product releases.

Revision History

v0.37 88/11/19

Fixed a bug indigenous to version 0.36 wherewith the sending of text without any files resulted in the setting of the FILE ATTACH flag on the outgoing messages. Mail bundlers would then take each word in the subject line as a file specification. As they were not, the bundlers understandably would complain about files being missing. The messages still went out so it was a rather benign, albeit noisy, bug.

Corrected the documentation to explain that zones can be specified in the 'TO' statements and is now mandatory in the 'FROM' statement.

v0.36 88/11/07

MAXATT {0-8} command (MAXimum ATTaches)

The new command, MAXATT {0-8}, overrides the maximum number of attached files (default of 3) that will be stuffed into the message subject field when using the normal file attach method.

If MAXATT is set to zero, then it has the identical effect as the former MAKEFLO command had in that it forces the direct creation of FLO files.

Attach stuffing

ROBO will only stuff as many attached file paths as will fit in the subject field up to the MAXATT value. If the subject field becomes full before the MAXATT value is reached, it will use as many additional messages as is necessary to attach them all.

For example, if the MAXATT value was at its default of 3 and you had the following six files to send ...

```
c:\fidonet\netmail\out\somefile.arc

c:\fidonet\netmail\out\junkfile.arc

c:\fidonet\netmail\out\testfile.arc

c:\fidonet\netmail\out\thatfile.arc

c:\fidonet\netmail\out\thisfile.arc

c:\fidonet\netmail\out\somefile.arc
```

... then three messages would be generated, not 2, because only two such paths would fit in the standard SUBJ field width.

MAKEFLO command removed.

The MAKEFLO command has been retired. The new "MAXATT 0" command accomplishes the same thing since zero attaches forces the direct creation of FLO files. Replace any occurrences of "MAKFLO" with the "MAXATT 0" statement for the same effect.

ONETEXT {YES|NO} command

This command affects the handling of any defined message text when there are multiple files to send by the normal file attach method.

When set to YES (default), only the first message to each destination will contain the defined text and any subsequent messages that may be necessary to send the remaining attached files for that destination will be the minimum necessary for an attach. This is useful when you have a large text which you don't want included with other than the first message to each destination.

When set to NO, then the defined text will be repeatedly included in all of the messages sent to each destination. This is useful when you want to send a covering message with all attach messages, eg, to explain the reason for the send or instructions on the handling of the attached files.

Unreported, tracked file Bug

A minor, unreported, bug was found in the FILE TRACKing logic that may have, in some instances, allowed a duplicate file to be sent rather than skipping it as it should.

v0.35 88/10/31

Changed FROM command

The FROM command which previously only specified the district and node segments of your address and used for the origin address in various mail objects has been expanded to include the zone segment for use in the zone support.

Zone Support

Added the currently standard zone support method whereby inter-zone messages are redirected to the zone host address but with an "IFNA Kludge" line as the first line in the message body (hidden behind a Ctrl-A) with the true out-of-zone destination address. Any direct FLO files created via the "MAXATT 0" command will be readdressed to the other zone's host "gateway" in the current zone. It is up to such gateway's to handle such re-routed attach lists as there is no standard for such handling at the present time.

TBBS Kill-Dupe Quirk-Around

Added a fix for TBBS's DUPE KILLER which only checks a message DATE and SUBJ for DUPE detection and not the file size. Robo now increments the seconds, by 1, for each subsequent message SEND.

This quirk only rears it's head when you send multiple text messages in rapid succession, using multiple SENDS, as might be the case when sending a multi-part bulletin as a series of messages.

SeaDog IN-TRANSIT Quirk-Around

Forced the setting of the LOCAL flag on messages so that SeaDog doesn't mistake a message as InTransit when that flag is not set. Even though a message is addressed as being from you and destined for elsewhere, SeaDog will consider it as InTransit unless the LOCAL flag is set.

InterZone File Hold Quirk-Around

Added the ability to make a FLO file with a different priority then the covering message. This is only applicable when "MAXATT 0" is set and is, therefore, not compatible with SeaDog whose environment is not compatible with direct FLO file creation.

This feature was added for those cases where a file distributor might want to HOLD one or more files for Long Distance or interzone pickups yet send the covering message as a notification, without attached files.

New FSET command

The new command "FSET { HOLD | CRASH | NORM | SAME} assigns the priority of the outbound FLO file if, and only if, you are using the direct make flo option -F or "MAXATT 0" command in the control file. This is for support of the interzone file hold feature previously described.

Each FSET command sets the FLO priority as per the given value (CRASH, etc). The "FSET SAME" command restores the default operation whereby the FLO matches the covering message's priority as set by the standard "SET" command.

CLEAR command bug fixed

Fixed a problem whereby the use of the CLEAR command to clear message attributes did not always do so correctly or refused to accept certain CLEAR commands with an "unknown object" message.

New SET options

Added the "SET NORM" command to set a message's priority to NORMAL. This replaces the former (and obsolete) commands of CLEAR CRASH and CLEAR HOLD.

Obsoleted CLEAR options

The "CLEAR HOLD" and "CLEAR CRASH" commands are now obsolete as the "SET NORM" command does the same job.

v0.34 88/10/18

Now defaults sending files by the traditional method of inserting the filespec in the subject line and setting the ATTACH flag. This method is more generic and should work with more environments. It does not directly create a send file list (FLO) but depends on your message bundler to do that as it sees fit. The previous version's direct FLO file method can be re-enabled by using the F switch.

NEW: "IF TIME" construct for time checking.

NEW: "IF HOUR" construct for hour checking.

NEW: "IF MIN" construct for minute checking.

NEW: "ELSE" clause for the IF/ENDIF construct to allow conditional execution of a block of statements when the test condition is NOT true.

NEW: Rewrote, expanded, or corrected several sections of the documentation - most notable "Theory of Operation", "Commands", and "Credits".

FIX: Minor text formatting problem that caused external text that had a double space between two consecutive paragraphs to end up as a single space when the following paragraph did not start with a space.

v0.33 88/10/01

CHG: Rewrote and enhanced the sample control file.

NEW: Now allows comments in 'TO FILE' lists.

FIX: Minor bug with maximum checking.

FIX: Minor bug in file list line parsing.

v0.32 88/09/19

NEW: File Tracking Log Report option, -r.

FIX: Tracking of "Sent Date" in tracking log.

FIX: Help display when using the -h switch.

CHG: Revised the help screen text.

CHG: Send logic now aborts when files aren't found.

CHG: Revised documentation, typos, etc.

v0.31 88/09/17

FIX: Allows message send without files.

FIX: Added missing CRLF at end of inline text body.

v0.30 88/09/15

FIX: Display of 'Encl' files in message text.

NEW: Msg text lines now end with 'soft returns' (0x8D).

ADD: Rewrote docs with more clarity and examples.

v0.29 88/09/01 First general release. Working well.