4315-0413-11

PUBNET System Documentation

# YMS Primer

4315-0413-11

PUBNET System Documentation

# YMS Primer

# PUBNET System Documentation
# YMS Primer

Document Number 4315-0413-11

February 26, 2026

Evie Cooper
Michel Hack
C. J. Stephenson

This document originated from the HSnet Document Management System. It is stored on the *PUBDOCS* TOOLS minidisk in PDF and source form.

For comments or requests related to this document, please send a note to **WEC AT EVIEVM**!

Revision history:

- First Edition, February 26, 2026
- Second Edition, March 6, 2026
- Third Edition, March 10, 2026
- Fourth Edition, March 18, 2026

**Fourth Edition**

**Note:** This is composed as a synthesis of notes from Michel and documentation written by C. J. All credit goes to them for this amazing OS, and Michel for his maintenance of this amazing OS on EVIEVM.

# Contents

# Figures

# Starting YMS

If CMS's system disk has the address of 190 within your virtual machine, YMS's system disk has the address 290. To gain access to YMS, enter the following CP commands on your VM:

```
LINK YMS 290 290 RR
TERM CONMODE 3270
<CMS will die if you were running it>
IPL 290
```

You will now be using YMS. You will want to read the next “Whirlwind Tour” section, then go to the disk initialization section to get started.

If you are in need of a YMS 291 minidisk, you should tell Evie and she will hopefully get you one. I mean, I definitely will give you one!

# A Whirlwind Tour

**Welcome to ED and EM/YMS!**

You are now in ED, which, besides being the standard editor in EM/YMS, also serves as the primary user environment for issuing system commands. EM/YMS runs either on a bare 370 or on a virtual machine under CP. In general you should not have to issue CP commands (other than LOGOFF) as equivalent function is provided by YMS.

ED supports a large number of requests. To find out about all its facilities you first need to know about a few very primitive requests. You can scroll forward (toward the bottom) and backward (toward the beginning) with the **S** and **-S** requests. You can look at other files with the editor without losing your place in the one you are now looking at by issuing a request of the form **ED name type**. You can return to this file with the **Q** request or by issuing **NEWUSER** again.

One of ED's features is a general UNDO capability. Except when issuing commands that change or erase EM files you can do no wrong. If you don't like the results of what you just did type **UNDO**.

If you look at the screen you will see that there is a header at the top which contains a mode indicator at the left (normally 'Edit:'); an indicator of where you are in the file in the center; and a number followed by the name of the file at the right. There is an input area on the bottom two lines of the screen. (There is a scale line immediately above that.) If ED has messages to show you it will temporarily use some of the lines just above the scale line.

You can return to this memo exactly where you left off by issuing **NEWUSER** again. To see the documentation for any ED request, is **TELL** *request*; to see introductory information about ED, issue **INTRO**; to see a list of all the ED primitive requests issue **PRIMITIVES**.

## Basic Commands

A short description of some of the ED requests follows.

| | |
|---|---|
| **1,2,...** | View a file that is already being edited |
| **$** | Execute a YMS command |
| **-** | Reverse direction of request following **-** |
| **?** | Recall the last request(s) |
| **"** | Execute previous request again |
| **BOTTOM** | Make the last line the current line |
| **CHANGE** | Replace one string with another over given range |
| **COPY** | Copy lines from one place to another |
| **CP** | Execute a CP command |
| **DELETE** | Delete line(s) |
| **EDIT** | Edit specified file or file identified by current line |
| **FILE** | Save current edit file in EM library and abandon edit file |
| **FIND** | Advance to next line that matches column for column |

| | |
|---|---|
| **HISTORY** | Show the last few commands issued |
| **INSERT** | Insert new line or lines |
| **LIST** | List the files currently being edited |
| **LOCATE** | Advance to the next line with matching string |
| **MOVE** | Move lines from one place to another |
| **NEXT** | Move toward end of file |
| **PARAGRAPH** | Format a paragraph of text |
| **QUIT** | Abandon current edit file |
| **REPLACE** | Replace current line |
| **SAVE** | Save current file in EM library |
| **SCROLL** | Advance current line to next display screen |
| **SHOWPF** | Edit a file showing the current PF key settings |
| **TELL** | Edit a file explaining specified part of ED |
| **TELLY** | Edit a file explaining specified part of YMS |
| **TOP** | Make the top line the current line |
| **UNDO** | Undo effect of one or more previous requests |
| **UP** | Move toward top of file |
| **VIEW** | Split the screen into multiple views |

Some of the more common of these requests can be issued by using the PF keys. Issue **SHOWPF** to see what they are set to. Now would be a good time to try some of the requests. Issue: **TELL** *request* to get more information on some of them. Remember you can always issue **NEWUSER** to return here.

## Libraries and Files

One of the most important services provided by the computer is the maintenance of the user's documents, which are stored in files. These are saved by the computer between sessions, even though there may be a power failure, or the computer may be shut down for some other reason. It is because of this memory that the user can continue from where he left off yesterday (or last week, or last year), without having to start again from scratch. In fact, the only things which survive from one session to the next (apart from the equipment itself) are the files, together with the 'libraries' in which the files are stored.

To see what libraries you already have issue 'LIBS'. This will show at least a SYSTEM library and perhaps your own 'A' library. To see what files are available on a library make that library the current line (with suitable NEXT or UP requests) and issue **LL**. To see the contents of a particular file make one of those lines the current line and issue **E**.

Besides the currently active libraries there may be others which can be activated. There may be even more which can be made available by attaching disks or tapes. Disks or tapes can be divided into one or more areas each of which may be a library that can be activated. To see a list of the currently attached areas, issue **AREAS**. To activate one of these, issue **ACTIVATE**. When running under CP, to see a list of othe disks (and the areas on them) edit the file *KNOWN AREAS*. While pointing at a particular line of this file you can issue **ATTACH**. If that succeeds (it may fail because the disk requires a password), you can then issue **ACTIVATE** to make it active.

If you are running under CP, the following may apply. Since you are a new user, you may not have an 'A' library. You can create a temporary A library on a temporary 291 disk by issuing**$INIT291**. Any files created on it will be lost when you log off. You should at your earliest convenience get a disk for permanent storage of files.

# Advanced Editing

A generally powerful way of doing things in YMS is to make lists of objects of various kinds in an edit file (e.g. libraries or files). When editing such a file there are many requests which operate on the current line, e.g. edit the file referred to.

## Lister Commands

Requests that generate lists of objects:

| | |
|---|---|
| **AREAS** | Edit a list of the attached devices and the areas on them |
| **LF** | Edit a list of files matching given pattern |
| **LIBS** | Edit a list of the currently active EM libraries |
| **LISTSPOOL** | Edit a list of spool files *(useful synonym:* **LS***)* |
| **LL** | Edit a list of files in library designated by the current line |
| **LM** | Edit a list of the members of a collection |
| **PROG** | Edit a list of the currently loaded YMS programs |

## Spool File Lists

When in a file of type **SPOOLS** created by LISTSPOOL:

| | |
|---|---|
| **DISCLOSE** | Display the network characteristics of the designated spool file |
| **FORWARD** | Reroute the designated spool file to some other user |
| **LOAD** | Load the designated spool file into an EM file |
| **LOOK** | Edit the designated spool file |
| **PEEK** | Edit the raw form of the designated spool file |
| **PURGE** | Purge the designated spool file |
| **REVISE** | Change the network characteristics of the designated spool file |
| **SORT** | Sort the spool files |

## File Lists

When in a file of type **LIST** created by LF or LL:

| | |
|---|---|
| **COPYFILE** | Make a copy of the designated EM file (useful synonym: CF) |
| **EDIT** | Edit the designated EM file |
| **ERASE** | Erase the designated EM file |
| **EXAMINE** | Examine (a read only editor) the designated EM file |

**GATHER** Collect several designated EM files into a collection

**KOMPARE** Compare two files to determine their differences

**MOVEFILE** Move the designated EM file

**PRINTFILE** Print the designated EM file

**PRUNE** Prune list to those containing a particular string

**RENAME** Rename the designated EM file

**RENEW** Regenerate the infomation about the designated EM file

**SAMEFILE** List other files related to the designated EM file

**SEARCH** Search through the files for particular strings

**SENDFILE** Send the designated EM file to another user

**SORT** Sort the list of files on any of the columns

## Member Lists

When in a file of type **MEMBERS** created by LM:

**EDMEMBER** Edit the designated member

## Disk Areas Lists

When in a file of type **AREAS** created by AREAS:

**ACTIVATE** Activate the designated area

## Disk Library Lists

When in a file of type **LIBS** created by LIBS:

**ACTIVATE** Activate another library

**DEACTIVATE** Deactivate an area

**DETACH** Deactivate and detach an area

**REACTIVATE** Reactivate an area (to see recent changes or change R/W status)

## Programs Lists

When in a file of type **PROGS** created by PROG:

**BREAK** Set a breakpoint on the designated program

**PROWL** Enter PROWL and set origin at beginning of program

**UNLOAD** Unload the designated program

# MAIL and MESSAGES

The following commands work with mail:

**KEEP** Record the (received) mail file currently being edited

**MAIL** Begin editing a mail file

**SEND** Send the mail file currently being edited

**SPEAK** Send a message to another user

# Documents

The current EM and YMS documents are listed in the file *DOCUMENT LIST*. Enter **EDIT DOCUMENT LIST** to see that list. To read a particular document, go to the appropriate line in that file and enter **E** alone. A summary of useful documents is given here: the same procedure may be used to edit one of these documents.

```
INTRODUCTION  MEMO    *     General Introduction to YMS
ED            INTRO   *     General Introduction to ED
EXEC-2        MEMO    *     EXEC 2  Language Reference Manual
EXEC2-IN-YMS  MEMO    *     EXEC 2  Implementation Notes for YMS
REXX          MEMO    *     Documentation for REXX interpreter
YMS           MEMO    *     Some notes on the structure of YMS
```

# Tailoring

Both ED and YMS may be tailored to suit each user's personal tastes with regard to command abbreviations and defaults. Also, you can define your own YMS commands by writing EXEC files and your own ED commands by writing ED files. These may be programmed in the EXEC 2 or the REX language.

The following file allows you to tailor your YMS environment:

```
YMS           INIT      A     General environment setup (must be in A library)
```

The following files allow you to tailor your ED environment:

```
ED            SYNS      *     Table of synonyms
ED            MACROS    *     Table of macro abbreviations
ED            INITS     *     Table of initial file characteristics
ED            BAD       *     EXEC file which determines what to do
                              with illegal ED requests
```

Look at the files below to see synonyms and abbreviations that are already set up for you:

```
YMS           SYNS      *     Table of YMS synonyms
ED            SYNS'     *     Table of synonyms
ED            MACROS'   *     Table of abbreviations
ED            INITS'    *     Table of file characteristics
```

# Tell

The programs which run with YMS contain a certain amount of their own documentation. If you do not have your manual at hand, or if the system has been changed since your manual was printed, you may be able to obtain instruction from the computer. The 'magic word' to remember for this is TELL. You may either issue **$TELL** *command* or **TELLY** *command* from the editor. (TELL from the editor will tell you about ED requests.) In most contexts, the command alone (without any arguments) provides some general information on the commands available, and instructions on how to obtain more detailed information.

It should be made clear that TELL will not necessarily 'help'. At best, it provides a summary of the rules. It does not examine the user's particular situation, or try to guess at what is giving him trouble.

# Fixing IPL Errors

***OBSOLETE - 2026-02-26***

YMS will not IPL on certain DASD arrays.[1] To address this upon being thrown into PRY, enter the following one-liner:

```
break .getaio+162 <45r40#go> # -loc 1b001b22 # go
```

[1] Multiprise 3000 Integrated Disk Array or the Bustech zDASD

# Initializing Disks

The section above is not entirely accurate on disk formatting for YMS. First, you need to acquire a 291 minidisk; this can either be a permament minidisk or a TDISK. Then, format it:

1. **E** - enter the editor
2. **$PREP 291** - invoke the PREP program
3. **N 1** - go to the next line of the first area
4. **INIT** - create the area on the disk
5. **Y** - confirm that the area should be created (and that the disk can be modified)
6. **FORMAT 4096** - format the area with a blocksize of 4096, a good blocksize for a 3390
7. **RESERVE FOR LIBRARY** - permit you to place a library onto the area
8. **NAME** *userid*/**A** - give the new area a name that corresponds to your userid
9. **Q** - quit the PREP screen and return to the plain editor
10. **ATTACH 291** - mount the disk you just formatted
11. **ACTIVATE** *userid*/**A** - activate the LIBRARY as R/W mode, library A

# Networking

## Using TOOLS

YMS includes a full **TOOLS ED** macro that can perform almost any TOOLS function; it is written in REXX.

First, set the TOOLS disk and the destination:

```
TOOLS SENDTO TOOLS/PUBVM
TOOLS DISK WWCHAT
```

To retrieve a *FORUM* file:

```
TOOLS GET GENERAL FORUM
```

You will then want to go to **LISTSPOOL**, select the line of the *FORUM* file, then **LOAD** it (which deposits it into a YMS disk file). From then, you can display it from **LL** with**E** like normal.

To create a file on TOOLS:

```
TOOLS CREATE SOMEPROG ASM
```

To append to a *FORUM*:

```
E GENERAL APPEND
<enter the file contents>
FILE
TOOLS APPEND GENERAL FORUM
```

Easy as that!

## Using MAIL

Before you can use the mail system, you need to create a nicknames file that is named *userid* **NICKNAMES A**. The first line corresponds to **your user**, and matches this form:

```
--|------------|------------|--------|--------|---|---------
  aaaa         bbbb         cccc     dddd     IBM eeee
```

The fields are as follows, listed in alphabetic order:

- **aaaa** is the nickname file entry for your userid. You should make this match your login userid.
- **bbbb** is the mail file that emails from the user will be stored in; since this is the first line that corresponds to *your* userid, you should make this the same as the first field.
- **cccc** is your userid on the machine you are running YMS on.
- **dddd** is the RSCS node ID of the machine you are running YMS on.
- **eeee** is an underscore-delimited version of your name.

Here is an example for the author's first line:

```
--|------------|------------|--------|--------|---|---------
  WEC          WEC          WEC      EVIEVM   IBM Evie_Cooper
```

Subsequent lines are for **other users.** Here is a sample for another user:

```
--|------------|------------|--------|--------|---|---------
  PUBMAINT     PUBMAINT     MAINT    PUBVM    IBM Jane_Doe
```

Next, you will need to run the **VNET** program that loads in the *NICKNAMES* file:

```
VNET REMEMBER userid NICKNAMES A
```

You will almost definitely want to put this line in your **YMS INIT** file.

# Using SENDFILE and LISTSPOOL

If you wish to send files to another user, use the **SENDFILE** macro from a *LL* or *LF* screen. You will want to specify a nickname on the same line as the command, so:

```
SENDFILE WEC
```

would send the highlighted file to me.

In order to receive files, you will want to use the **LOAD** command from a *LISTSPOOL* (of which *LS* is a good synonym from the editor) screen. Simply highlight the line you want, and use **LOAD** with a filename to deposit that into a YMS disk file. It will default to the A library, which is accessible to pretty much anyone that knows its password (which is RYMS).

# Using RSCS

The **RSCS** macro provides an easy-to-use interface to RSCS commands. It will execute a command, so the macro

```
RSCS PUBVM CPQ CPLEVEL
```

is equivalent to

```
CP SMSG RSCS CMD PUBVM CPQ CPLEVEL
```

It is highly recommended that you loaded Colloque to trap CP messages, else your terminal will flip between CP and YMS if you are using a 3270 console!

## Chatting With Other Users

You can use the **SPEAK** command to chat with other online users. You can use it in two ways:

```
speak wec hello
speak maint/pubvm can you fix TOOLS?
```

This also works with Colloque, see the section on this for more detail!

# Customizing YMS

## Tailoring YMS INIT

The **YMS INIT A** file is equivalent to **PROFILE EXEC** on CMS, **AUTOEXEC.BAT** on MS-DOS, and **.login** (or **.profile**) on UNIX. You will want to update this file to do several things for you:

1. Set some PF hotkeys.
2. Load the *WISCIO* extension to permit interfacing with the TCP/IP stack.
3. Load the *IUCV* extension to trap messages.
4. Prime your terminal with good settings to permit the typing of brackets and other special characters.
5. Load the Colloque program (works in conjunction with entry #3 in this list).[2]
6. Load the **NICKNAMES** file.

This file can be written in EXEC 2. Here is a sample file:

```
&PRESUME &CMD
PF GROUP 12 ;;N;
PF GROUP 11 ;;U;
PF STRING 9 "
PF GROUP 8 ;;COVERTLY SCROLL;
PF GROUP 7 ;;COVERTLY -SCROLL;
PF GROUP 1 ;;EDITX;
PF GROUP 2 ;;BACK;
PF GROUP 3 ;;QUIT

INSTALL WISCIO
INSTALL IUCV
SETUP3279
COLLOQUE
CPX QUIET MSG IUCV
VNET REMEMBER WEC NICKNAMES A
```

Of course, you will want to change **WEC NICKNAMES A** to *youruserid* **NICKNAMES A** as described in the above section.

**SETUP3279** will properly change your terminal settings.

[2] Colloque is a much more advanced messaging system than shown in this example; you will want to read the section on that for more.

**Typing Quirks: Special Characters**

The **SETUP3270** program, which can be regenerated from its assembler source, will change the terminal characteristics and special keys to the following:

| | |
|---|---|
| **!** | Immediate command *and* the literalizer. To type a ! in an editor document, you will need to type two bangs, so !! |
| **{** | Line-end character |
| **¢** | Tab character |
| **** | Backspace |

## An Ed Dark Color Scheme

To reverse the color scheme to save an OLED monitor running a 3270 emulator, create a file named **DARK ED A** with the following contents:

```
$PRYSET .CRTIO#L 2842002842#N11#R F5
$PRYSET .CRTIO#L 2841F2#R 284100
```

To run it, simply type **DARK** on the Ed command line! You can also include this in your editor init file if you would like this to run at start.

# Programming

This section explains how to write YMS programs in assembler and other high-level languages.

## Assembler H (IEV90)

Assembler H, or **IEV COMPILER S**, is the IBM System/370 assembler. It is provided on YMS and runs under the OS simulator.

### Entering the Program

Create an editor file and populate it with your program source. Name it **MYPROG ASM A**. Here is a sample listing:

```
---------|----|-----------------
IEVDEMO  CSECT
         SR   15,15
         BR   14
         END
```

Figure 1. Sample Dummy Assembler Program

### Preparing Macros

Since assembler programming is not nearly as fun as it could be without macros, you will certainly want to learn to create/edit macro files. These come in the form of a **CATALOG** file; these can, in-turn, make either *plain macro files* or *macro collections*[3] available to the assembler.

Let's say you have a macro catalog that expands upon the basic **SYSLIB** catalog (see the next subsection). That might look like this:

```
--|-----------------------------------
  INCLUDE OS MACROS *
  DECLARE $LOW MACRO * AS $LOW
  DECLARE PDPEPIL MACRO * AS PDPEPIL
  DECLARE PDPPRLG MACRO * AS PDPPRLG
  DECLARE PDPMAIN MACRO * AS PDPMAIN
  DECLARE PDPTOP MACRO * AS PDPTOP
```

As the contents might imply, this is for the PDPCLIB runtime library, which GCCYMS uses. Alas, you can see two important things here:

1. The **INCLUDE** line loads a *collection* of macros
2. The **DECLARE** lines work just like the way they do in the C/370 section below

### Assembling

While looking at the file in a *LL* or *LF* editor output session, type in the following command to run the assembler:

[3] See the **GATHER** and **LM** Ed macros for more information on this

```
ASM
```

You can also highlight the line in **FL** or **LL** and assemble it there; upon completion, the date will be replaced with *(ASM)* in its row.

If there are any errors, they will be made available to you to view when you close the current editor session or run **DIAG** to display the corresponding *filename* **DIAG** file.

If you wish to **use an alternate macro catalog,** you can create one and then specify it on the ASM macro:

```
ASM MYMACS
```

The default catalog is *SYSLIB CATALOG S*! See the note above about creating a macro catalog if you wish to do this.

## Binding

The binder is the YMS term for what other mainframe OSes call the link editor.[4] To link your program, enter these commands from Ed:

```
$bind myprog
insert myprog text a
finish
save
quit
```

## Running

Unlike the C compiler or any other compiler, these programs link to straight YMS programs with the filetype **370.** As such, you can run it directly from YMS by typing the name, or can run it from Ed like this:

```
$myprog
```

# Phantasm

Phantasm is the native YMS assembler that is arguably much more suited to YMS itself than the IEV90 assembler. You should already know how to write an assembler program if you've made it this far.[5] As such, I will jump straight to assembling and linking.

## Assembling

Since there is no helper macro, you can just directly invoke the assembler:

```
$phantasm myprog asm a
```

## Linking

To link your program, enter these commands from Ed:

---

[4] YMS does have an OS/VS compatible link editor, see the section on ***C/370*** to see how to use it.

[5] If not, see the YMS Assembler Programming Tutorial book (that I have not yet written).

```
$bind myprog
insert myprog text a
finish
save
quit
```

# C/370

YMS does indeed have a working C compiler, named **EDC COMPILER S**. As the name implies, this is IBM C/370 V1R2, from about 1992 or so. This program runs under the control of the OS/VS1 runtime environment; as such, programs that are compiled with this are limited to a 24-bit address space.

## Entering the Program

Start by creating your program. Edit a file named **TESTPROG C A**:

```
Edit:                       At line 6 of 6                    1.  TESTPROG


#include <stdio.h>

int main()
    puts("YMS C/370 Test");
    return 0;
}









0...:....1....:....2....:....3....:....4....:....5....:....6.. ..:....7...
                                                              EMZv0   RUNNING
```

Figure 2. Sample C/370 Program

Note that if you found that you could not type in the **#** characters, you forgot to run **SETUP3279** probably in your *YMS INIT* file. Likewise, if you could not type **{**, simply type two, so, **{{** to make it enter.

## Header Files

Next, it helps to understand how *header files* work on the OS/VS1 emulator. If you were running C/370 on real OS/VS1 or MVS, you would have a partitioned data set full of headers. Since we don't have these available to us, we must construct our own. Any header files that you will use will need to be placed into a *CATALOG* file, one of which our **EDCC** macro will always load: **C CATALOG**.

**If you wish to use the default C/370 headers,** use the following **C CATALOG** file contents:

```
INCLUDE EDC INCLUSIONS *
```

This *EXTENSIONS* file is actually a YMS collection -- as such, you can use the **LM** editor macro to inspect each entry.

If you wish to create a collection of your own, you can use the **GATHER** command!

**If you wish to deviate and use custom header files** without using a collection, use the following contents instead:

```
DECLARE STDIO H * AS STDIO
DECLARE MYHEADER H * AS MYHEADER
```

**STDIO H** is the standard C/370 header; these are available for you on the WEC 291 minidisk.[6] Whenever the compiler opens **STDIO**, the OS runtime monitor will fetch **STDIO H *** and feed the contents back to the compiler.

Now that you have created your *CATALOG* file (or used the existing one), you can now compile the program.

While you are looking at your source code in your editor (just like in the screen capture above), run the macro that compiles it:

```
EDCC
```

Type **QUIT** (or press PF3 if you used the example *YMS INIT* file in this manual), and you should see the compiler output:

---

[6] Subject to change. Please check for updates to this section in this document! Note that the default **C CATALOG** mentioned before is also present here -- the only reason this document is telling you how to create this file at all is if you wish to add your own header files!

```
Edit:                    At line 3 of 3                  13.  EDCC  REP




*-------------------------------------------------------------------
Compiling TESTPROG C A1 with
comp r;    21:29:13  vt = 29 ms  tt = 38 ms  wt = 0 ms  et = 137 ms









0...:....1....:....2....:....3....:....4....:....5....:....6.. ..:....7...

                                                          EMZv0   RUNNING
```

Figure 3. C/370 Compiler Output

Now, you need to link-edit the program. Normally, on YMS, you would us the binder, but this is for native YMS programs -- we are making an OS/VS1 program. As such, we must instead use **LINKEDIT**!

To use this, you will want to type the following commands *back-to-back* from Ed:

```
$linkedit testprog c-370 a
insert testprog c-text a
finish
save
quit
```

Finally, go to a file list (if you don't know how to get here or what this means, you need to go read the rest of this guide) and run **RUNC** on the *TESTPROG C-370* line. Hopefully it works!

# Advanced Topics

## Using SMTP with MAIL

You can use SMTP with the **MAIL** macro fairly easily. First, start mailing the PUBNET SMTP proxy service machine:

```
MAIL SMTP/PUBNET
```

Ensure the first line underneath the bar of ¬ symbols says this:

```
¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
To: myuser@mymail.net
```

Once composed, type **SEND** on the command line and it should send!

## PEDIT

This is an alternate editor. Check back later for more information!

Part Number
0131-14

File Number
YMSPRMR

Printed in U.S.A.