RAMDOSII (c) 1996 by wilf rigter
Adapted for use with XTender (c)

The easy way for Sinclair ZX81 (and XTender) users to
organize storage of data and speed up program development.

RAMDOSII FEATURES

The Tiny RAMDisk Operating System (RAMDOSII) is a simple file management program to name, save, load, merge or erase BASIC programs and variables, MC programs, and data files in a 32K RAMDISK. It has 15 build-in single key functions and can be expanded with new "transient" commands.
Using RAMDOSII, the user can create a LIBRARY of programs, variables and data files in the on-board nonvolatile RAMDISK. The RAMDISK is organized as a directory of 32 files complete with file names, file type and file length. Simple cursor control selects the desired file, and with a single keystroke, the file is transferred between the RAMDISK memory and the system memory. Amazingly compact, RAMDOSII assembles to 512 bytes of relocatable object code and comes complete with a fully annotated source listing. RD2 Source Code
RAMDOS is normally used with a large Non-Volatile Memory pack or the ZX97. Here RAMDOS2 has been adapted for the XTender1 program for demonstration.

For a detailed description of each command read the section on RAMDOSII but for a short exploration of the RAMDOSII system continue with this section.

Here is a zip file with the RAMDOS2 program and banks: RAMDOS2.ZIP

XTender users should start XTenders and LOAD "RAMDOS2" and enter 00-03 for the bank number. The loading screen reminds users of the simple command set. Press any key and voila the RAMDISK directory. The > cursor points to the first file in the RAMDISK. Note that RAMDOSII commands only require the first letter of each command to be entered except "E"rase which must be shifted for security and the "N"ame command must be followed with 7 characters of text which includes numbers, letters, punctuation, and leading or trailing spaces.

A SHORT INTRODUCTION TO RAMDOS2

LOAD AND RUN BASIC PROGRAMS

First practice returning to the BASIC operating system by pressing Q and then press "T" (RAND) followed by " enter" (N/L) to return to the directory. While in the BASIC edit mode, the keyword RAND is used as a hot key to access RAMDOSII. Next practice loading and running programs. The first program that we will load from the RAMDISK is the DELPHIC program in directory 00. To select the DELPHIC file use the 5,6,7,8 keys. Once the cursor is positioned in front of the desired file just press "L" and "R" to load and run the program. All BASIC programs can be interrupted by pressing the "space" key to cause a break. Next practice saving a BASIC program. Select a position in the directory to save the program with keys 5 to 8. Notice how the cursor moves left, right, up and down and wraps around the ends of the directory.

SAVE AND NAME BASIC PROGRAMS

Saving a BASIC program is as simple as pressing "B" and when saved, the file type and file length pops up in the directory and the file is now in the RAMDISK ready to be loaded back into SYSTEM RAM when it is required. Note that the save operation nondestructively copies a file from SYSTEM MEMORY to the RAMDISK. To name this new file press "N" and then type "DEMOTWO" or any other 7 characters. The current letter to be typed is shown in reverse and after the 7th character, the screen flashes to indicate the name command is complete. Line edit functions or back space is not available for this procedure. If you make a mistake while entering the name, just continue to the 7th character and then you can RENAME by repeating the same procedure typing over an old name to change it.

BASIC EXERCISE

We have already learned to save, name and load an existing BASIC program (B file). Let's continue and experience writing and saving your own BASIC program. First break the current BASIC program by pressing space bar on the keyboard. Then press NEW to erase the System Memory and "Q"uit the directory. Next enter a few lines of BASIC code such as :

```
 10 PRINT " WELCOME TO UNLIMITED POSSIBILITIES"    N/L  20 GOTO 10    N/L
```

This is just a short BASIC program to show that you can save your own programs. Then press RAND and select directory 4. Move the cursor to the position in the directory where you wish to save it. Next press the "B" key to save this BASIC program and notice the file type and length change. Also notice that all other files are moved over to make room for the new file. Now press the "N" key and type in WELCOME. You have just saved and named your own BASIC program in RAMDISK. Now press Q to quit and go back to BASIC.

Next we try out the B file LOAD command. Press NEW to clear SYSTEM RAM and to return to the directory. Now position the cursor, using cursor keys (5,6,7,8), to point to the WELCOME file. Press L once to load WELCOME into the SYSTEM RAM. Then press "R" to demonstrate the autorunning quit command.

MERGING TWO BASIC PROGRAMS

Next we can demonstrate the possibility to MERGE multiple B files by pressing Space, then RAND to return to the directory. Again position the cursor using cursor keys (5,6,7,8) to point to the WELCOME file. Press L once more to load a second copy of the WELCOME B file. Then use "Q" to quit and LIST to see 2 identical copies of the short BASIC program in the SYSTEM RAM. The clone was generated by MERGING the original with the copy loaded from the RAMDISK.

Simple LOADING occurs if the SYSTEM RAM had no previous BASIC program in it when LOADING a B file from RAMDISK. MERGE occurs, as we just demonstrated, when there is an existing BASIC program in the SYSTEM RAM, when loading any other B file from RAMDISK .

BASIC VARIABLE DATA

The procedure for saving and loading BASIC VARIABLES is identical to B file operations except that the letter "V" is used to save the VARIABLES to a V file. Experiment with DIM A$(1000) and then save this array to the RAMDISK with "V". Note the file size includes the variable name length etc. To do a simple V file LOAD, use the BASIC command CLEAR to erase the VARIABLES before loading a V file. To MERGE, simply DO NOT CLEAR existing VARIABLES before loading the V file.

The procedure for saving and loading MC programs to M files is identical to B file operations BUT MC PROGRAMS MUST BE FORMATTED before saving and loading. Read the example in INSTALLING RAMDOSII and RAMDOSII:SAVING M files before attempting any M file operations.

Continue as you like with this exercise by loading some of your favorite programs from EPROMDISK and saving them to RAMDISK until you feel familiar with the procedure for loading and saving, erasing and naming files in the RAMDISK. Then, for more information and program examples read the RAMDOSII section.

RAMDOSII DETAILED DESCRIPTION

The RAMDOSII software is designed to be compact, relocatable, and easy to use. It speeds up the process of saving and loading programs to "instantaneous". It is a tool for merging programs, subroutines and variables to aid program development.

RAMDOSII contains the following one key commands:

1.(B)asic Program Save to RAMDISK.
2.(V)ariable Area Save to RAMDISK.
3.(M)achine Code Save to RAMDISK.
4.(N)ame or rename file in Directory.
5.(E)rase (shift E) file from RAMDISK.
6.(L)oad file from RAMDISK.
7.(Q)uit RAMDOSII and return to calling program.
8.(R)un - quit RAMDOSII and GOTO 1 of BASIC program.
9.SPACE BAR - quit and break to BASIC edit mode.
10.Left, right, up, down arrows move the cursor.
11.Keys "0" and "1" select directories.

The letter in brackets is the single key used to execute the command , (N)ame must be followed by seven characters of text. The screen flashes as it is updated after each command is executed. For security (E)rase must be used together with the shift key.

REMEMBER THAT ALL KEYS ARE REPEATING AND COMMANDS WILL BE EXECUTED
MULTIPLE TIMES IF THE KEYS ARE HELD DOWN.

Additional commands are provided as transient utility routines as described under TRANSIENT COMMANDS.

Although there is no explicit Write Protect disable command before SAVING, NAMING and ERASING, the control register WRITE bit is set just before these commands modify RAMDISK memory and is immediately reset afterwards. Attempting to POKE a byte in the RAMDISK (32K-64K) from BASIC, will not work unless you first set the WRITE bit in the control register with POKE 5E3,128. However if DIP switch S1 is on then the RAMDISK memory is unconditionally write protected.

THE RAMDISK

The RAMDISK directories of 32K each are loaded into an area of the memory map normally not used by BASIC or ML programs.
The RAMDISK is used in the same way as a floppy or hard drive, but is much faster. The current directory is located between 32K and 64K as shown in FIG 1. The directories are organized as shown in FIG 2. Note the way that files are packed together with no waste of storage space. There are no minimum size sectors or fragmented memory to worry about. The actual location within the current directory is usually of no interest since the files are loaded into system memory before they are actually used. If required the memory address of the file to which the cursor is pointing, is passed back to BASIC in system variables 16507 (LSB) and 16508 (MSB) after quitting RAMDOSII and returning to BASIC.

DIRECTORY

When RAMDOSII prints the current directory to the screen, it reads the control register and prints the directory number to the top left hand corner of the screen. It reads and prints the first file name and type from start of 32K directory and masks bytes to make sure that the name and type are printable characters .(<CHR$64)

Next it reads and prints the file length and calculates the location of the next file header. It continues to print out all the file names it can read until it reaches the 32nd file. The order of the file names on the screen is left to right, top to bottom corresponding to the first to last position in memory. Depending on the contents of the directory, there may be some blank files names showing in the directory before or after any files which have actually been saved. For example, a empty directory generates a screen full of "   ;   0" files indicating that the current directory memory is all zeroes. (ie spaces). The RAMDISK memory can be initialized with NULL bytes using the CLRDISK utility. (see TRANSIENT COMMANDS)

CURSORS

The unshifted 5,6,7,8 keys are repeat action cursor keys. They move the inverted > cursor around the directory to select files by name. The unshifted 1 and 2 keys select 1 of 8 directories scanning forwards or backwards. Note that the cursor and directory keys roll over after reaching the first and last file or directory. The location in the current directory of the file pointed to by the cursor is stored in System Variables 16507/8 and this may be used as a parameter by other programs.

SAVING FILES

When saving files either the write enable push button must be held down or the write protect switch must be in the closed position. The memory map shown in

FIG 1 indicates the portions of memory which are saved to RAMDISK using the B, M, and V commands. The structure of the file types is shown in FIG 3-5 . The header for each file is 10 bytes : 7 bytes for name, 1 byte for type, and 2 bytes for length. Saving files is a simple procedure of entering the file name followed by the letter B, V, or M representing file type to be save. First the cursor is positioned in the desired location in the directory. After pressing B, V, or M, the files after the cursor will be moved over one position by the new file, thereby overwriting the 32nd file if it contained valid data. Only after a file has been saved can the file can be named.

BASIC SAVE

We save a BASIC file by simply pressing "B" and the directory will now show the "B" for BASIC type and the length of the file. The B file (BASIC file) is compact because it only includes BASIC lines but not the display or Variables.

VARIABLES SAVE

Saving BASIC VARIABLES separately provides a powerful means of transferring data files from one program to another, and allows a program to periodically backup it's BASIC VARIABLES without the need to save the program itself. Some BASIC programs don't initialize their BASIC VARIABLES and therefore won't operate if the VARIABLES are not retained. In that case both a B file and a V file must be saved, generally using the same NAME, and these must also be loaded together in order to properly execute the program. Remember NOT to use the RUN command since this will clear the BASIC VARIABLES. V files are saved by pressing "V" after which the "V" type and length are displayed in the directory. The VARIABLES in the SYSTEM RAM are only copied into the RAMDISK and are not cleared.

MC-SAVE

Saving and loading Machine Code programs presents some problems primarily because of the lack of standards for the structure and location of MC programs. MC programs reside and run anywhere below 32K , some above RAMTOP, some as part of a BASIC program in the first line of BASIC, some in the last line of BASIC. In addition many programs are written to run between 8K-16K. In order to reduce the complexity of dealing with MC program it is convenient to standardize three run locations.

1. The MC programs developed with the bundled Z80 assembler can be run from the BASIC area as part of a BASIC program. These M/C programs are stored in a 1 REM line and the transient commands and control demonstration programs are good examples. This type of MC program is very easy to deal with since it is by definition a B file and should be saved and loaded as such. This type of MC program is conceptually the simplest.

2. MC programs generated by the Z80 assembler which run from the 8K-16K area must be formatted as 1 REM lines, organized as shown in FIG 4. Note that the actual code is preceded with a 2 byte address which is the location of the

first byte of code when loaded into it's RUN LOCATION. After formatting this
program is saved as an M file. Note that the address used when calling the
program (RAND USR NN) is not necessarily the same as it's RUN LOCATION
address. A good example of this type of MC program is the Z80 disassembler
which is transferred to RUN LOCATION 8192 when loaded. The Z80 disassembler is
started with RAND USR 1E4.

3. Some M/C programs run above RAMTOP, the highest memory address used
by the BASIC program, which means that RAMTOP must be lowered prior to loading
the M file above RAMTOP. The Z80 assembler can generate such program by
preceding the code with the RUN LOCATION address as shown in FIG 4. The
program is saved as an M file. The Transient command RTPLOAD can be used to
lower RAMTOP.(see TRANSIENT COMMANDS) If already present above RAMTOP this
of
type MC program can copied from above RAMTOP into a formatted 1 REM line by
using the REMSAVE transient command and can then be saved as an M file. The
Z80 assembler is an example of this type of MC program. When saving an M file
the MC program must be in the following format: The Machine Code must be
located in the FIRST BASIC LINE as shown in FIG 4. This is the location of the
Machine Code product of some ASSEMBLER programs and the location for many MC
programs after they are loaded from tape. In addition the first 2 bytes of
the MC program right after REM, 118, 118 must be the RUN LOCATION address for
that MC program. This is no problem if you are the MC programmer and you can
simply precede the SOURCE CODE with the RUN LOCATION bytes before assembling
the OBJECT CODE. It is also possible to convert existing MC programs to M
files, by stripping off the relocating utility and adding the necessary RUN
LOCATION bytes. (Also see WRX16 HIRES, and Transient commands REMSAVE,
RTPLOAD)

NAMING FILES

When naming files either the write enable push button must be held down or the write protect switch must be in the closed position. After saving a new file, the file length and file type appear in the directory but the name field is still blank. In order to name a file the user simply presses (N)ame and enters the 7 character name including any leading or trailing spaces. The current letter to be typed is an black square. All printable non-inverted characters may be used. After entering the 7th character the screen flashes to indicate the name command is finished. If you make a mistake while entering the name continue to the last character and then repeat the procedure to rename the file.(Backspace is not provided) To RENAME an old file, point cursor to the old file name and then use the same procedure as for a new file name. If NVM write protection DIP switch (S1) was OPEN or the write enable button not held down while entering the name, then the name will disappear after entering the 7th character.

FILE TYPE

The type of each file may be changed with the "T"ype command which highlights the file type and changes it to the next character entered with the keyboard. This is primarily useful to change M files to B files so they can be merged with existing BASIC programs. Another use is to lock files by changing the file type to a character other than B, V, or M. Such files can not be loaded or erased and are therefore considered locked. The last letter of the file name should be changed to the original type in order to remind the user when unlocking the file by returning the file to the original file type. The "T"ype command must be used with great caution aside from the two applications mentioned to avoid a system crash when attempting to load converted files.

LOADING and MERGING

The single "L" (LOAD) command transfers all types of files from the RAMDISK to various locations in the System Memory. Just move the cursor to point to the desired file name and then press L.   Like SAVING there are 3 types files that can be LOADED as follows:

BASIC FILES

BASIC files are loaded into the System Memory area between the System Variables and the Display. (FIG 1) The BASIC Variables are not affected by a Bfile Load which allows the user to combine existing BASIC Variables with new programs. There are two cases of loading a B file, namely LOADING and MERGING, which occur as a result of the initial conditions of the B file LOAD.

BASIC LOAD

If there is NO BASIC program in the BASIC area prior to Load (as would be the case if the user had deleted all lines or executed a NEW) then once Loaded. the B file will be the only BASIC program and it will Autorun from line 1 on exit from RAMDOSII. (Like GOTO 1)
 Remember that some BASIC programs may not execute properly if started at Line 1. If necessary add a line of BASIC such as 1 GOTO NN to the program where NN is the start of that BASIC program, before to Saving to RAMDISK.

BASIC MERGE

If there is already a BASIC program in the BASIC area before Loading a B file, then after Loading there will be two (or more) merged programs in the BASIC area with the latest B file MERGED ahead of the other programs. This is called BASIC MERGING and the combined program will NOT Autorun. B files are merged AHEAD of existing BASIC program lines rather than behind for several reasons:

1. Transient MC programs in 1 REM, which are used as commands, may be LOADED (merged), run with RAND USR 16523, and deleted with "1" without disturbing the existing basic program. Transient Commands usually are MC programs that were saved as B files and when loaded can run from the BASIC area. 2. When writing a BASIC program, universal subroutines stored in the RAMDISK can be loaded as required during the program development keeping in mind that subroutines run faster if located at the beginning of the basic program. 3. A merged program, before renumbering, can be easily Listed and Run if it is located at the start of the BASIC program area. A transient Command RENUMBR is required to give merged program lines unique line numbers. It is included in RAMDOSII BASIC and must be extracted and saved to an M file before it can be used as a transient command (see TRANSIENT COMMANDS).

VARIABLE FILES

Select the V file to be transferred from the RAMDISK to the System Memory using the cursor keys. The selected V file is transferred to the BASIC variable area between the end of the Display area (VARS) and the Work space (E-LINE) when pressing L. Variables are loaded or merged depending on initial conditions.

VARIABLE LOAD

Only one set of variables will exist after a V file LOAD if there were no previous variables in the variable area. This is true right after power-up ,or if no variables have been defined , or after CLEAR, RUN and NEW. BASIC programs are not affected by V file LOAD except for the impact that the new variables values may have on the program. Some BASIC programs were saved together with BASIC variables, in order to execute properly. Such programs must be loaded together with their variables. The order of loading is not important.

VARIABLE MERGE

If a set of BASIC variables existed prior to LOADING then the V file will be merged AHEAD of the existing BASIC variables. This means if two or more variables have the same name then only the latest variable can be accessed. There is no utility such as RENUMBER available for merged variables and the merged variables should therefore have unique names prior to saving. It is possible to separately delete array variables using DIM XX (0) while leaving the other BASIC variables intact. This may prove useful in some applications.

MC FILES

When loading a M file, two things are assumed :

1. That the M file was prepared in accordance with the M file SAVE instruction.

2. The area pointed to by the RUN LOCATION bytes is free.

Since any first BASIC line can be saved as a M file it is possible to generate a M file which has a random run-location, which if loaded may cause a system crash. M files which have a RUNLOCATION above RAMTOP may be loaded only after RAMTOP is lowered. A transient command, RTPLOAD, lowers RAMTOP to the value of the RUNLOCATION. M files do not necessarily contain MC programs but may also contain HIRES files, or 2040 printer pattern tables or any other data. Great care must be taken to avoid a system crash due to the inherent unconstrained addressing of the RUN LOCATION of MC programs and the way MC programs can alter critical unprotected areas of the system .

ERASING FILES

When erasing files either the write enable push button must be held down or the write protect switch must be in the closed position. In order to conserve space or to update files it is sometimes necessary to erase the old files. This is done by pointing the cursor at the file to be erased and pressing (E)rase while pushing the write protect button. The files located after the erased file are moved up to fill the gap and an equivalent new space is created at the end of the RAMDISK. If a file cannot be erased is possible that random data in the length field makes the file appear to be too long so that erase would cross the END of the RAMDISK. In that case push the file off the end of the directory by inserting small files in front of the long file or clear to END using the transient command CLRDISK.

QUIT RAMDOSII

When the user wants to exit RAMDOSII simply press Q. The position of the cursor can be peeked at 16507/8 and used to determine free RAMDISK or as a parameter for the transient programs. Newly loaded B files may autorun after Q, as described in BASIC LOAD. If RAMDOSII was called from within a BASIC program using for example 100 RAND USR 7685 then if a V file or M file was loaded it continues to run after Q or if a B file was loaded an error break will occur.

RAMDOSII IN OTHER PROGRAMS

You can use RAMDOSII and transfer files from within a BASIC or MC program.

BASIC PROGRAMS

It is simple to add a program line to any BASIC program and call RAMDOS by entering, for example, 100 RAND USR 32691. That address, as you can see, is 5 bytes more than 32K. This skips a test which RAMDOSII uses to filter the keyword RAND in BASIC programs while that program is in the RUN mode. On execution the Directory will appear and M files and V files can be loaded as required. Quit will return to the next BASIC program line which continues uninterrupted. If RAMDOSII is called from within a BASIC program and a B file is loaded, then this will result in a merge of the calling program and the B file which will result in an error break after you quit RAMDOSII. To return to the calling Basic program always use "Q"uit. If the call was a function like LET A=USR 7685 then the value which is returned is the address of the first byte of the current file name to which the cursor was pointed before quitting. MC PROGRAMS
MC programs can JUMP or CALL the RAMDOSII program at address 1E07 (7687d). Once in the DIRECTORY you can transfer files etc. B files can be LOADED but only if the calling MC program is not located in the BASIC area . Because RAMDOSII was written in relocatable Machine Code it can be merged with any other MC program and run from any location in memory. The various places for locating MC programs are: in the Enhanced ROM, 8K-16K, above RAMTOP or in any BASIC program line. To return to the calling program always use "Q"uit or "Space" since "R"un will force a restart from the first basic line. On return, the BC register pair holds the address of the first byte of the file header to which the cursor was pointed when quitting.

RAMDOSII TRANSIENT COMMANDS

Transient commands for RAMDOSII, which are commands or functions that are SAVED as B files and are LOADED or MERGED when needed to perform the required commands. The RENUMBR, UNMERGE, CLRDISK and FREERAM programs are examples of these transient command programs and are included with the development software. Any other utilities such as HEX/DEC conversion can be easily added by the user to suit his own particular requirements.
In general these transient commands are MC programs that are used infrequently, for a short time and are deleted after use. They may be MERGED with, but do not interfere with, existing programs in the SYSTEM RAM. If a transient command consists of a single 1 REM line with a MC program then it is deleted after use with 1, N/L. If the command is written in multiple BASIC lines then the UNMERGE command is used to reduce these multiple lines to a single 1 REM line which can be deleted with 2 keystrokes.

The UNMERGE routine is for use with multiple line BASIC transient programs to allow such multiple line BASIC programs to be deleted with only a few keystrokes while leaving the original BASIC program intact. UNMERGE is MERGED with the transient command program which must end with 9998 RAND USR 16523 and 9999 STOP When used the Transient command is executed with GOTO 1 and when finished, the UNMERGE command will take all lines between 1 and 9999 and include them in a single 1 REM line which can then be deleted with 1, N/L.

The RENUMBR command will renumber merged BASIC program lines in increments of 10 but will not change GOTO's or GOSUB's. Just LOAD RENUMBR and RAND USR 16523
After that press 10 and N/L to delete the RENUMBR utility and remember to change all those GOTO's and GOSUB's.

The FREERAM command returns the space remaining in the RAMDISK from the cursor to the end of the RAMDISK. After Loading FREERAM from the directory, position the cursor after the last file of interest, Quit and then use PRINT USR 16523 to print the number of bytes remaining from the end of the last file to the end of the RAMDISK.

The CLRDISK command is used to clear all or part of the RAMDISK to 00. There is no way to recover the data after it has been cleared and therefore CLRDISK must be used with care and attention. After Loading CLRDISK, position the cursor to point to the location in the directory after the last file of interest and Quit.

MAKE SURE YOU HAVE POSITIONED THE CURSOR CORRECTLY.

Use RAND USR 16523 to run the CLRDISK command. Remove the 1 REM line which contains CLRDISK with 1 and N/L. Go to the directory to reset the write bit.
The REMSAVE command is used to transfer an MC program from above RAMTOP to a 10 REM line, automatically formatted with the appropriate RUNLOCATION bytes and ready to be saved as a M file. Use RAND USR 16523 to run.

The RTPLOAD command is a companion to REMSAVE when loading M files above RAMTOP. In that case load RTPLOAD and leave the cursor pointing to the M file to be transferred to above RAMTOP. This M file must have a VALID RUNLOCATION ie somewhere between RAMTOP and 32K and must have been saved using the REMSAVE command . Next Quit and RAND USR 16523 will lower RAMTOP to the required value followed by an automatic NEW. Then LOAD the M file in the usual way.

BUNDLED SOFTWARE

ASSEMBLER

The Z80 Assembler program is "L"oaded into 8k-16K memory as a Mfile.
Thereafter it can be executed with RAND USR 8192.
The generated source code in 2 REM can be saved to a Bfile before assembly.
The executable code in 1 REM can be saved as a Mfile or as a Bfile. Appendix A provides the full instructions.

DISASSEMBLER

The disasssembler is installed with a simple load to the area between 8192 and xxx. The program is executed with RAND USR 1E4. The following functions are available:

1. "D"isassemble.
2. "L"ist data in HEX and Sinclair format.
3. "S"earch for a two byte string.
4. "C"ontinue listing or disassembling.
5. "Z"=copy disassembled code to a 2 REM file.

The disassembler was written by Harvey Taylor and slightly modified by me to dump the output to an assembler source code file. Before using the "Z" function, the user must first load a Assembler compatible 2 REM line. If necesary, install and run the Assembler program to create a 2 REM line. The disassembled code which appears on the screen is transferred when pressing "Z" and is appended to any existing source code in the 2 REM line. Multiple screens can be transferred this way. The only changes the user must make before assembling this disassembled source code is to translate the hex offset that accompanies the IX and IY instructions to decimal code. code. This minor shortcoming stems from the peculiar requirement of the assembler for decimal offsets. The 2REM2REM utility can be loaded and run with RAND USR 8192 to merge two 2 REM lines into one 2 REM line. This is a great way to create macros or a library of subroutines which can be merged into a single source file before editing with the assembler.

COMMUNICATION

The ZXTERM80 terminal program was written by Fred Nachbaur and includes ZMODEM upload and download capability . It is a HIRES program which displays up to 80 columns of ASCII or Sinclair Code. The instructions are in appendix B.

MULTITASKING

The program NOVA 1000 is included which provides a BASIC
line trace utility and a real time clock which run concurrently with any other application program. This makes it easy to write time or clock oriented control programs. The details to NOVA are provided in the NOVA article .