(specification)

CAD MEMO 2

MICROCOMPUTER WIREWRAP PROGRAM DISK

Stowe Keller
March 7, 1986

I. INTRODUCTION

This report describes a microcomputer version of a program designed to minimize the work involved in producing a wiring list for NRAO Shalloway wirewrap/ printed-circuit boards (shown in Figures 1 and 2). The program is intended to replace the wirewrap program for the IBM ("Pandora") mainframe, described in EDIR #163. That program, and the one described here, were written as tools to let a logic designer take a logic diagram and translate it into a wiring list for a Shalloway wirewrap card. This new version has greater flexibility in chip placement and pin specification, better error trapping and output format, automatic Vcc, GND and bypass capacitor wiring, and will output both used and unused pin cross-reference lists. Single Inline Packages can be entered, as well as being able to defeat the automatic Vcc and GND wiring for those packages that do not have standard Vcc and GND pins. In addition, descriptions are provided for related programs to prepare the input file, and to process the output files for conversion to other formats. This report also generally applies to the Mark III wirewrap program, known as WrapB, which is described in a separate document.

II. WIREWRAP DISK DESCRIPTION

The program disk is a stand-alone IBM-PC compatible PC-DOS/MS-DOS 5-1/4" floppy disk containing several files, including the PC-WRITE text editor (and its associated files), a brief help file, two versions of the wirewrap program (one for use those machines with an 8087 math chip installed, the other for non-8087 machines), and a program to convert the wirewrap output file to a compressed format for use by an automatic wirewrap company such as ELMA. A batch file is used to automate the wirewrap processing sequence; it is designed to catch errors and abort gracefully whenever possible. Due to memory constraints, the Turbo Pascal compiler and source code for most of the programs are actually stored on a separate disk; in general, when WireWrap or the WireWrap disk is mentioned, it refers to the dedicated WireWrap disk with the executable files, not the source code.

The PC-WRITE text editor is provided on the WireWrap disk for entering the wirelist input file; if the user wishes, any editor that produces DOS text files may be used instead. It is recommended that the user start with a blank formatted disk and maintain all his files on that disk instead of using the wirewrap disk. The wirelist file should contain the chip declarations and wirelist information in the format described later in this report.

After saving this file to his work disk, the user may then boot the wirewrap program disk (or simply invoke AUTOEXEC.BAT); a title screen will identify the disk and offer suggestions on the available programs. By entering "WW" (without quotes) at the DOS prompt, a batch file will execute and invoke the wirewrap program. The user then specifies the name of the wirelist input and output files. If no errors occur, there will normally be a wire list output file and a pin cross-reference file on disk, and the batch file will resume execution and ask the user if he wants to proceed to the conversion program to generate an ELMA-compatible (or other) wirelist file.

III. USING THE WIREWRAP PROGRAM

WireWrap is designed to work on the IBM PC and compatibles (including the AT&T) with DOS 2.0 or higher, and the format of the input and output files of

WireWrap are normal PC-DOS/MS-DOS ASCII text files. The computer should have at least 256K of memory in order to run WireWrap; 640K is required if a large RAM disk is going to be used with the program. To facilitate the generation of the input file, the PC-WRITE text editor is provided on the disk, but almost any other editor that generates a text file can be used.

The simplest way to use WireWrap is to boot the disk, and after the title page is displayed, invoke the PC-WRITE editor with "ED filename", where 'filename' is the name of the input file you wish to create. Working from the original schematics and circuit board layout, you are expected to enter a file in the format described in Section IV.

After you have entered the input file, save it to disk and exit the editor. From the DOS prompt, type 'WW' to start the wirewrap process. WW is a batch file that will determine if an 8087 numeric coprocessor is installed in your machine, and will invoke the appropriate version (8087 or non-8087) of WireWrap. WireWrap will display a few title lines, and then ask for an input filename, which may include a drive specification; if the file cannot be successfully opened, you will be prompted again. You will then be asked for the name of the output file for the wirelist and an output file for the pin cross-reference. In both these cases you may include a drive specification, and if a file cannot be successfully created, the program will prompt you for filenames until it is successful. It is also possible to specify 'CON:' or 'NUL' as a filename. Using CON: will send the output file to the screen (console) instead of the disk; if an output file is not desired, NUL can be used to act as a bit bucket by throwing away any data it receives. CON: and NUL are most commonly used during the debugging stages to quickly display errors in the wirelist, and for suppressing the generation of a cross-reference file, which can spare the user from several minutes of execution time.

If the wirelist output is sent to disk, WireWrap will print a period (".") to the screen for every wirelist string that is output, reassuring the user that the program is still functioning. (The length of the pauses between the printing of periods is directly related to the number of pins, and hence the amount of processing, in the current wire string.)

Almost all errors that can occur are sent to the wirelist output file, and are placed on lines by themselves, prefixed with "!", to make them easier to locate. WireWrap was designed to continue processing after errors occur to allow locating as many as possible in one run of the program. Be forewarned, though, that some errors can easily cascade into others: for example, a bad IC declaration will almost certainly cause bad chip references in the wirelist. To avoid overflowing a disk with error messages, though, WireWrap is designed to stop at 25 errors.

Most wirewrap runs are completed in 10 minutes are less. Using a RAM disk or hard disk will substantially reduce execution time; files can always be copied to another disk at a later time. If a cross-reference file is generated, it will contain an alphanumeric list of used pins with the input file line number, and the one, two, or three other pins to which it is connected. (Both IC pins and backplane notation pins will be output in this list.) Following that list is an alphanumeric list of unused IC pins (but no backplane pins, since such a list would be prohibitively long).

If errors occurred during WireWrap, the user will be notified on the screen with the error count, and both WireWrap and the batch file will terminate. If there were no wirelist errors, the batch file resumes control and will give the user an opportunity to proceed to a conversion program to convert the output file to ELMA format for automatic wirewrapping. It is intended that this conversion program will support new formats in the future if that should be required.

To view the output files, any one of several methods can be used: PC-WRITE, the DOS TYPE command, the text file list utility "L", or the DOS COPY command, which can send the output files to the printer ("PRN:"). All of these are available on the WireWrap disk.

IV. INPUT FILE FORMAT

The input text file is expected to have the following general format:

```
* Comment lines: any text that occurs after an asterisk in a line
*  is ignored. This allows commenting the file with date and project
*  name, as well as with details regarding each chip declaration or
*  other useful information regarding the wirelist commands.
DECLARE
* The command above, which must appear on a line by itself, informs
*  WireWrap to begin processing IC declarations, one to a line, in
*  the format below:
* Loc'n    # Pins    Pin #1 Loc'n    Type    Vcc    GND
1A          20          C1            1       20      1  * (EXAMPLE)
* Note: there must be at least one space separating each column;
*  additional spaces between columns are ignored, and can be used
*  to enhance readability. See the text for a discussion of the
*  information specified in IC declarations.
```

```
WIRELIST

* The command above informs WireWrap that the chip declarations are

*  complete, and that the list of wiring instructions follow.

S 1A2 1A7 @C2     * (Example wire string specification)

* Wire strings are sequences of two or more pins to be wired together;

*  the two string commands are 'S' and 'SE' (for 'String' and 'String

*  and 'String Exact'). See the text for a discussion of pin

*  specifications, wire string limitations, etc.

END

* The command END is used to mark the end of the entire input file; any

*  additional lines will be ignored.
```

IC declarations must contain the following information:

```
* Loc'n     # Pins    Pin #1 Loc'n    Type     Vcc    GND

1A           20           C1             1       20     1  * (EXAMPLE)
```

Loc'n : Frontplane location of the IC, specified as a number 1..11, followed by a letter

# Pins : Total number of pins on IC

Pin #1

Loc'n : Location, in backplane notation, of pin #1 on the IC. This is normally a letter C..Z (except I,O,Q) followed by a number 1..55. Note that single digit numbers are automatically converted to two digits by prefixing them with "0".

Type : Specifies the size of the chip in multiples of 0.3 inches: 1 means a .3 inch chip, 2 means .6 inch, etc. For Single Inline Packages, use 0.

>>>>>NOTE: SPECIFICATION OF VCC AND GND ARE OPTIONAL IN A RESTRICTED WAY:

YOU MAY OMIT BOTH OR INCLUDE BOTH, BUT YOU MAY NOT SPECIFY ONLY ONE. FOR THOSE CIRCUMSTANCES IN WHICH A VCC OR GND PIN DO NOT APPLY, SUCH AS A RESISTOR PACK, USE A "0" TO PREVENT AUTOMATIC WIRING OF THAT PIN.<<<<<

Vcc :	Pin number on IC of Vcc. If not specified, this value defaults to the highest numbered pin. Zero means there is no Vcc pin.
GND :	Pin number of GND. If not specified, this defaults to the value (highest numbered pin) / 2. Zero means there is no GND pin.

Also note that single digit values are converted to 2-digit values (prefixed with zero) when appropriate, and will appear as such in the output files.

In the wirelist section of the file, there are two ways to wire up strings: "S" (for "String") specifies all pins in the string and lets WireWrap determine the optimal sequence in which to wire those pins so as to minimize the total length of wire used. The other command, "SE" (for "String Exact") specifies an exact sequence in which pins are to be wired. Up to 30 pins are allowed in a string; a single line in the input file may contain one or more wire strings, or a single wire string may cross two or more lines. In general, wiring will continue until an "S" or "SE" is encountered. In both cases, the order of the wires will be staggered to make repairs easier, but the ultimate sequence will be the same. If bad IC or pin references occur, there may not be enough pins to form a string, and WireWrap will generate a corresponding error message.

A pin specification may be either a frontplane coordinate (also called chip notation, since it refers to a pin number on a particular chip or package), or a backplane coordinate prefixed with "@", or a plug reference. Frontplane coordinates have the form of one or two digits, a letter, and another one or two digits. For example, "1A2" refers to IC 1A, pin number 2. Backplane coordinates are formed with an "@" followed by a letter B..Z (except I,O,Q) followed by a one or two digit number: "@C2" refers to backplane column C, row 2. (When WireWrap performs automatic Vcc and GND wiring, a "V" or "G" is appended to the backplane coordinate of the Vcc or GND pin.) Plug coordinates consist of the letter "P" followed by a one or two digit number, and are automatically converted to "@A" since that is the actual backplane coordinate of the plug connectors. NOTE: there is an important difference between a backplane coordinate, such as "@P5", and a plug coordinate, such as "P5"; this is an easy mistake to make, so it is recommended that the user check the outputs from WireWrap when in doubt.

V. OUTPUT FILE FORMATS

WireWrap can produce two files: a wirelist output file and a used/unused pin cross-reference list. The wirelist output file is a DOS text file which contains the following sections of information:

-Automatic GND and Vcc wiring list. A Vcc or GND pin will not be wired if it has a pin number of 0 in the IC declaration. WireWrap will locate the nearest GND pin first, and then the nearest Vcc pin to that GND pin.

Example:

```
GND and Vcc wire list:
  INPUT LINE    LENGTH   FROM    TO    COUNT
Auto      0       3.0    B04    B07G        1     01A14   GND
```

Where "Auto   0" means that this wire was not specified in the input file, but rather was generated automatically by WireWrap. Length is the wire length in inches (see note on wire length elsewhere). "FROM" and "TO" are the backplane coordinates of the wire to be connected, "COUNT" is the cumulative wire count for the wirelist, and the last two items consist of the frontplane Vcc or GND pin followed by "Vcc" or "GND".

-Bypass capacitor list. This list that is automatically generated from the Vcc and GND information for each IC and tells where bypass capacitors should be installed for filtering out noise. Backplane notation is used.

-Wirelist. This is almost identical in format to the GND and Vcc list described above, except that each wire string begins with the word "Start", followed by one or more lines containing each pair of pins to be wired together. Also, the "INPUT LINE" column contains the input line number from the WireWrap input text file, and the last two items on the line are the original pin specifications from the input file. Having these pin specifications displayed here is extremely useful in the debugging process. Note that wire length is calculated as the Pythagorean distance between the two pins, plus 2.5 inches to allow slack during wirewrapping, rounded up to the nearest half inch. If this length is less than 3 inches, it is set equal to 3 inches, to insure that there is enough wire to allow wrapping.

-The last line of a wirelist output file should be "END OF WIRELIST"; if this is not the case, then the file is probably missing the last part of its output (perhaps due to a disk error).

The pin cross-reference file is the other output file that WireWrap can generate. It consists of:

- -Used pin cross-reference, in alphanumeric order of pins. Both frontplane and backplane notation pins will appear. The first column contains the input file line number, then the pin currently being cross-referenced, and the one, two or three pins to which it is connected. An input file line number of 0 means that WireWrap had referenced that pin during the automatic Vcc and GND wiring. Vcc and GND pins are in backplane notation and have a suffix of "V" and "G", respectively.
- -Unused pin cross-reference (for IC pins only) in alphanumeric order. A carriage return is generated after each chip and/or after every ten pins output from a single chip. The last line of this file is "END OF XREF FILE".

VI. PROGRAM DESCRIPTION

WireWrap is written in Turbo Pascal v3.01A for MS-DOS/PC-DOS 2.0 or higher. Both the regular (non-8087) and 8087 Turbo Pascal compilers were used to generate the two versions of the wirewrap program on the disk. The source code for wirewrap is approximately 50K long and generates a .COM file about 30K long. A variety of static and dynamic data structures are used to store the various data; constants are used where possible, and many variables are represented by structured variables.

Several goals were kept in mind when WireWrap was written: it needed to replace the mainframe version of the WireWrap program, but include features to handle situations that were not accounted for when the mainframe version was written. Source file formats had to be similar to allow easy conversion of mainframe files to the new program, but allow for flexible placement of chips and declaration of chips to handle different size packages, automatic Vcc and GND wiring (with the ability to override the defaults for the Vcc and GND pin numbers), and easier specification of backplane pin locations. In addition, a new format of pin cross-reference was desired along with a unused IC pin cross-reference. Any errors that occur during processing should not terminate WireWrap but rather give an appropriate error message, increment an error count and continue processing.

Rather than port the original program to the PC, it was decided to rewrite the entire program using a language and programming environment that would allow for easier program maintenance and future modification, as well as providing reasonable execution speed. IBM PC FORTRAN proved far too unwieldy and cumbersome in compiling and debugging, and left much to be desired in terms of the data structures that needed to be manipulated. After a brief evaluation Turbo Pascal was chosen as a remarkably fast and easy to use compiler that produces compact efficient code, and its availability on many machines as well as its commercial popularity offer the possibility of porting the new WireWrap program to other computers. Pascal, particularly Turbo's implementation, is a very effective language for many applications and is not difficult to learn, which should facilitate any future changes to WireWrap or one of its associated utilities.

A variety of data types and structures are used in the new WireWrap program, such as constants, boolean flags, text files, and enumerated data types. Character strings are used for storing such things as filenames, chipnames, text from the input file to process, and frontplane and backplane coordinates. Arrays of integers or of record structures are used for storing the list of IC's; the list of Vcc and GND pins across which to wire bypass capacitors; the array of all pins for storing their pin use and wire count, if appropriate. In addition, arrays are used for the list of pins and real-world coordinates for use in generating optimally short wiring sequences given a wire string, and for a few other useful things. Structured variables are a powerful construct in Pascal, and are used alone as well in some of the previously mentioned arrays for grouping related data together: for example, the StringList array not only stores the name of the specified pin, but also stores the input line number on which that pin was found. This input line number is output later in the program for the benefit of the user. For the pin cross-reference, which needs to be in alphanumeric (ASCII) order when output, the data is stored in a linked list structure and an insertion sort is used whenever a pin is referenced: if that pin entry already exists, an additional reference is appended to that entry. If it does not exist, it is inserted into the list at the point where it should alphanumerically occur. Through this method the list is always sorted, and needs no further processing when it comes time to print it out.

In accordance with the way in which Pascal programs are supposed to be written, WireWrap is broken down into numerous functions and procedures, which range from simple utilities to convert between data types to higher level subroutines for inputting text, processing chip declarations and wire strings, finding the shortest wiring paths and outputting a formatted wire list and pin cross-references. Several programming utilities are taken directly from the Turbo Tutor book and Turbo Pascal manual. The one-pass compilation technique of Pascal dictates that identifiers, which include functions and procedures, must be defined before they are used (with a few rare exceptions, such as pointers and forward procedure declarations); this means that lowlevel routines normally occur early in the program listing, with high-level routines occurring towards the end, just before the program itself. The algorithm for generating optimal wire strings is taken from the original mainframe program: try out all possible sequences, measure the total distance of wire that would be required, and remember which sequence proves to be the shortest. WireWrap will pause during the processing of long optimal wire strings because the number of possible sequences increases dramatically.

When a wirelist is output, be it an Exact String or an optimal String, all the odd-numbered entries are output first, followed by the even-numbered entries, so as to achieve a staggering of the wires at time of wirewrapping. This allows for easier repair and modification of a board after it has been wirewrapped.

VII. ADDITIONAL PROGRAMS

The WireWrap disk has several files on it in addition to the WireWrap program. There is a DOS batch file called 'WW' that is used in automating the wirewrap process; it makes use of several batch file commands in determining if desired files exist, as well as examining the 'ErrorLevel' upon exit of certain programs to determine flow of control in the batch file. For example, WW calls a .COM file to test for the presence of an 8087 numeric coprocessor. Based on the results of that test, WW will invoke the regular (non-8087) or 8087 version of WireWrap. After exiting WireWrap, an ErrorLevel of 0 indicates that no errors occurred, and WW will offer the user the opportunity to proceed to the ELMA conversion program. The WW batch file is presented in Listing 1.

The TEST8087 file is a machine language file for detecting the presence or absence of an 8087 numeric coprocessor in the system, and returns that information to the WW batch file described above. The program consists of two parts: a short Turbo Pascal program, and a machine language program. The machine language program was modified from a utility that was published in the June, 1985, PC TECH JOURNAL magazine, p. 181; it performs the actual test to see if the 8087 is installed, and passes the information back to the Turbo Pascal program, which displays a message to the user, and then passes the information to the batch file via the "Halt(0)" or "Halt(1)" commands. Listing 2 contains the two parts of the TEST8087 program.

The ELMA convert program (CONVWW) is a short (140 line) Turbo Pascal program that will take the wirelist output file from WireWrap and convert it into a new file in the format required for ELMA's automatic wirewrap equipment. The output file to be processed must not have any errors, or CONVWW will abort the conversion process. The user is asked for the names of the file to be processed and the filename for storing the converted information. In addition, the user will be asked for a one line text header to place at the beginning of the converted file. CONVWW will read lines from the file to be converted, ignoring any lines that do not contain wiring information (such as the column headers and 'Start' lines). The two pins to be wired and the wiring count will be extracted and output to the new file in the rigid ELMA format. Note that CONVWW expects such information to be in specific places in WireWrap's output file; any substantial change to that format will have to be accounted for in the CONVWW program. (Listing 3 shows the ELMA convert program.)

A separate program which has proved useful in transferring old wirewrap input files from the IBM mainframe to the PC is COPYFILE, which is a short

Pascal program on the PC for eliminating undesirable control characters, trailing spaces, and optionally removing other unwanted lines from files that have been downloaded. For example, some downloaded files are prefixed with line numbers, and/or contain blank lines and lines of asterisks for formatting purposes. COPYFILE gives the user the option to remove some or all of these groups of characters. Substantial amounts of disk space can be saved by processing files with this utility, as well as simplifying the conversion of old wirewrap source files to the new format.

VIII. CONCLUSION

This report has described the personal computer version of WireWrap, a program to facilitate the generation of wirewrap wiring lists from the original circuit schematics and IC placement diagrams. WireWrap was written to replace the wirewrap program on the IBM mainframe, as well as to offer capabilities and features not available in the mainframe version. These new features include more flexible chip placement, handling of different IC sizes including Single Inline Packages, more convenient backplane notation and better error trapping. Additional programs with WireWrap include a conversion program of WireWrap output to ELMA compatible file format, and a filter program that is useful for converting files downloaded from the mainframe into a DOS text file, with all control characters (other than carriage returns) and trailing spaces removed.

IX. ACKNOWLEDGMENTS

This project was done under the direction of Ray Escoffier, Walter Brown, Gene Runion and Andy Dowd, and the author would like to thank these people for their help.

Figure 1. Shalloway wirewrap card.

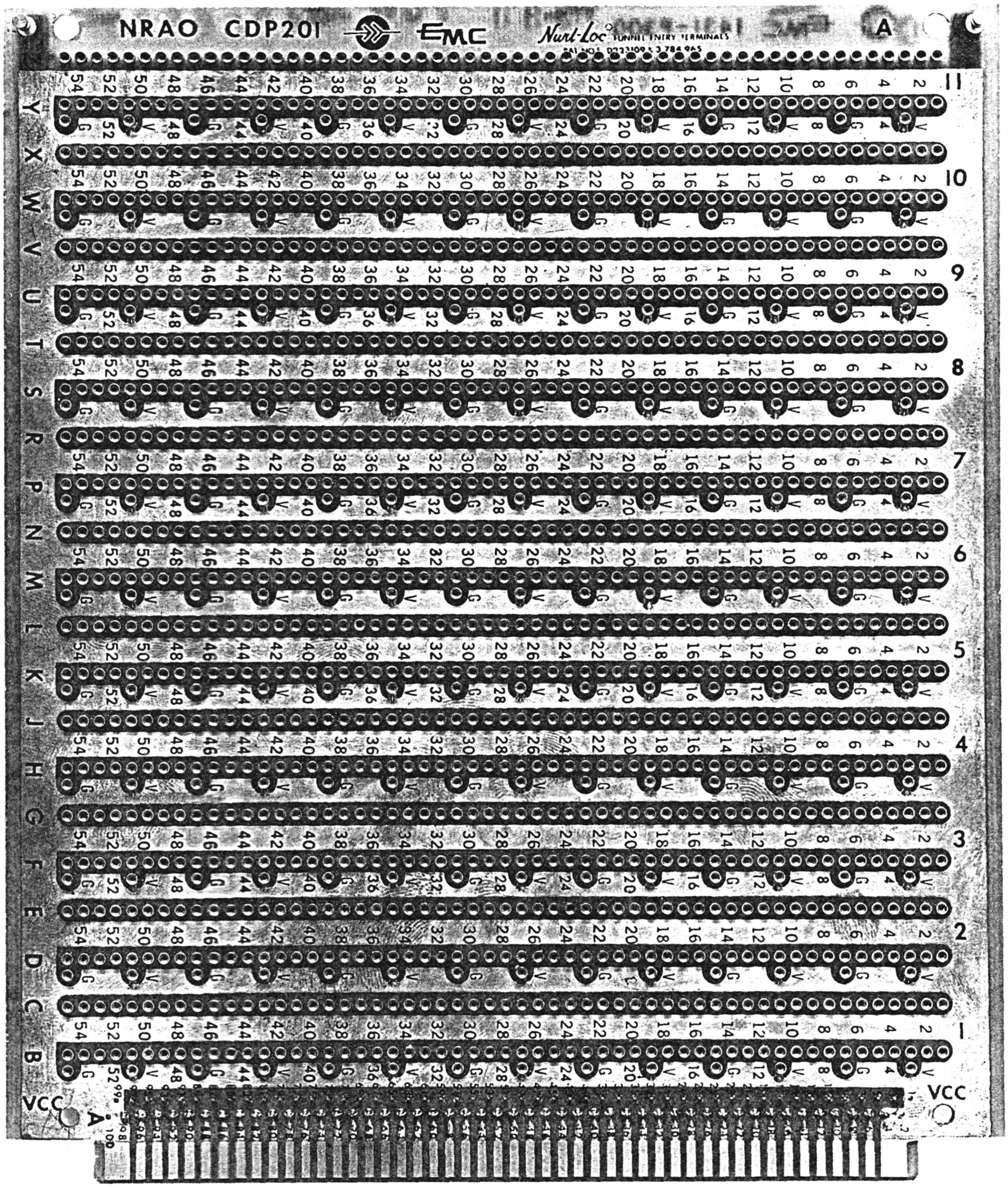
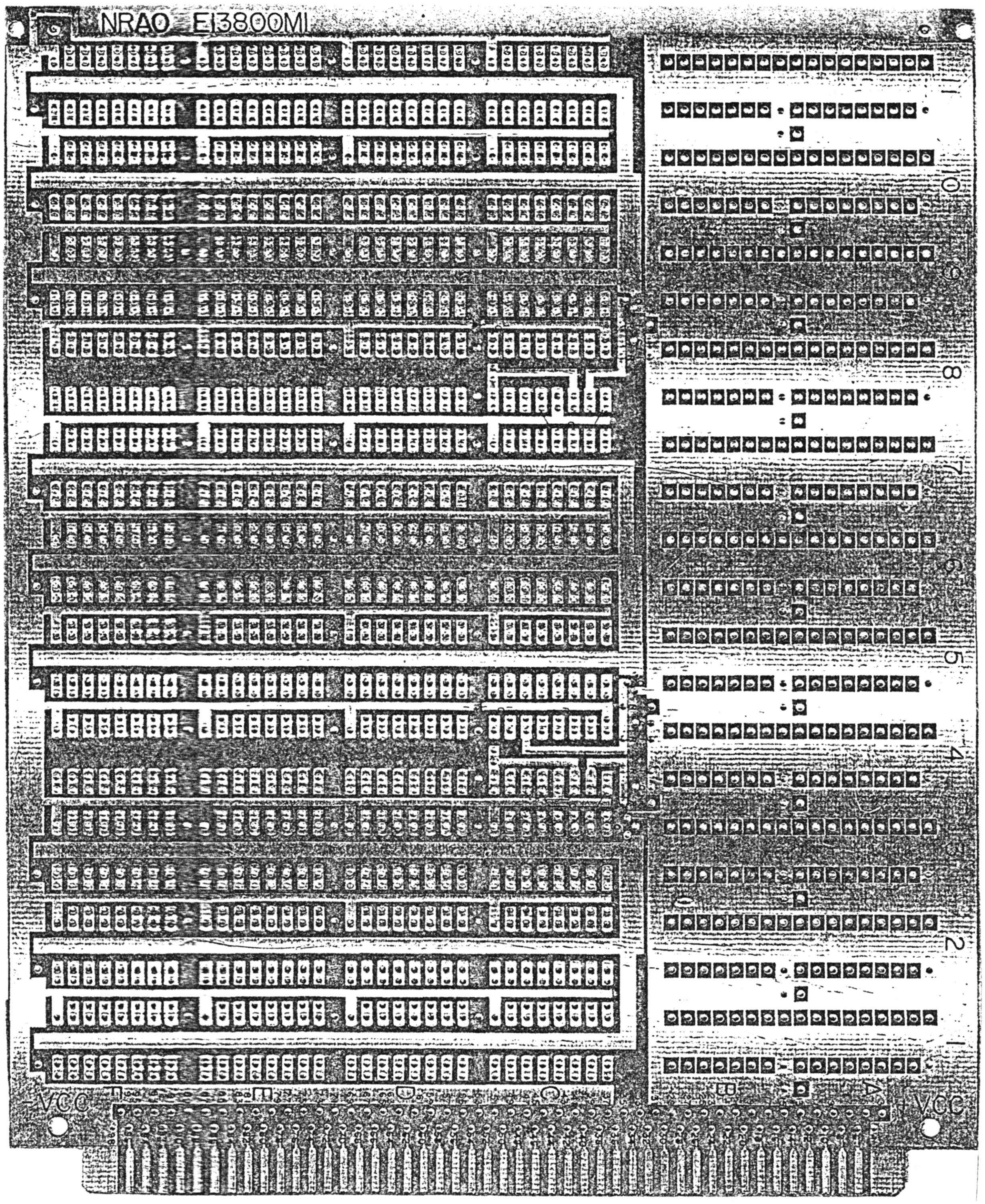


Figure 2. ECL card.



Listing 1. The WW Batch File

```
echo off
echo testing for 8087 math chip
test8087
if errorlevel 1 goto ww

if not exist ww87.com goto fileerr
echo To invoke the 8087 WireWrap program,
pause
ww87
if errorlevel 1 goto exit
goto convert

:ww
if not exist wwpgm.com goto fileerr
echo To invoke the regular WireWrap program,
pause
wwpgm
if errorlevel 1 goto exit
goto convert

:fileerr
echo ERROR: CAN'T FIND THAT FILE OR PROGRAM
goto exit
:convert
echo To convert output to ELMA format,
pause
convww

:exit
echo termination of batch file.
```

Listing 2. Listing of TEST8087 Programs

```
program testfor8087;    {Turbo Pascal program}

function test8087 : integer; external 'test'; {use machine language test}

begin {program}
  if test8087 = 0 then
    begin
      Writeln('False--8087 or 80287 not installed.');
      Halt(1);  {exit with errorlevel = 1}
    end
  else
    begin
      Writeln('True--8087 or 80287 installed.');
      Halt(0);   {exit with errorlevel = 0}
    end;
end. {program}


; Assembly language utility to detect 8087:
; Check for 8087 or 80287 math coprocessor
; modified from june 1985 pc tech journal, p. 181

code    segment public
assume cs:code

        org     100h                    ;set up as a com file
        .8087                           ;tell assembler to process 8087 stuff
start:  jmp     begin

control dw      0

;
; test if math coprocessor is present and return integer (0 or 1) to
;    the turbo pascal program

begin:
        fninit                          ;initialize math chip
        xor     ah,ah                   ;clear ah
        mov     byte ptr CS:control+1,ah ;clear memory byte
        fnstcw  control                 ;store control word
        mov     ah,byte ptr CS:control+1 ;hi byte is 03h if 8087
        cmp     ah,03h                  ;   or 80287 installed
        jne     none                    ;
        mov     ax,01h                  ;math chip is present
        jmp     exit                    ;
none:
        mov     ax,00h                  ;math chip not present
exit:
        ret                             ;return
code    ends                            ;
        end     start                   ;start is entry point
```

Listing 3. Listing of CONVWW Program

```
program ConvertWWOutputFile;
{by Stowe Keller, NRAO CV Ivy Rd              11/15/85 FRI 03:00pm}
{program to convert wirewrap output to ELMA format file}

{$R+}     {Range error checking--remove for faster execution}
{$U+}     {User interrupt--remove for faster execution}

{-----------------------------------------------------------------------------}
type
  MaxString = string[255];
  FileName = string[14];
{-----------------------------------------------------------------------------}
var
  FilIn,FilOut:             Text;   {text file on disk or other device (CON:)}
  Line, Line1 :             string[255];
  InputFile,OutputFile :    FileName;
{-----------------------------------------------------------------------------}
procedure PrintTitle;
  begin
    ClrScr;
    Writeln('This program converts WireWrap (PC version) output files');
    Writeln('to the format required for automatic wirewrap.');
    Writeln;
    Writeln;
  end;

procedure InputFileHeader(var Line : MaxString);
  begin
    Writeln;
    Write('Enter file name to place at beginning of file: ');
    Readln(Line);
  end; {procedure}

function SubStr(Line : MaxString; I,J : integer) : MaxString;
  {extract and return substring}
  begin
    SubStr := Copy(Line,I,J - I + 1);      {new way}
  end; {function}

procedure GetFileInfo;  {get input, output and xref file information}
  var Parm1,Parm2,Parm3 : FileName;
    OK : boolean;
  begin
    repeat
      Write('Enter name of INPUT text file : '); {Set up input file:}
      Readln(InputFile);
      Assign(FilIn,InputFile);
      {$I-} Reset(FilIn); {$I+}  {trap i/o errors}
      OK := (IOResult = 0);
      if not OK then Writeln('ERROR: CAN''T FIND FILE.');
    until OK;
    Writeln;
    repeat
      Write('Enter name of OUTPUT text file : '); {Set up output file:}
      Readln(OutputFile);
      Assign(FilOut,OutputFile);
      {$I-} Rewrite(FilOut); {$I+}  {trap i/o errors}
      OK := (IOResult = 0);
      if not OK then Writeln('ERROR: CAN''T CREATE FILE.');
    until OK;
  end;  {procedure}

procedure Strip(var Line : MaxString);   {maybe redo as function?}
  var
    I      :  integer;
  begin                   {Remove extraneous spaces and ctrl chrs:}
    I := Length(Line);
    if Line[I] = chr(8) then Line := SubStr(Line,1,I-1);
    I := Length(Line);
    while Line[I] = ' ' do
     {find last non-space in string}
      I := I - 1;
    Line := SubStr(Line,1,I);; {update string}
    I := Length(Line);
    if Line[I] = chr(8) then Line := SubStr(Line,1,I-1);
  end; {procedure}

procedure GetLine(var Line : MaxString);
  {get a non-comment line from input file}
  begin
    repeat
      if not Eof(FilIn) then
        begin
          Readln(FilIn,Line);  {read line from input}
          if Line[1] = '!' then
            begin
              Writeln('Input file must not have any errors!');
              Writeln('Conversion process terminated.');
              Halt; {stop program}
            end;
        end
      else
        Line:= chr(26); {ctrl-Z for end of file marker}
    until ((Length(Line) > 50) and
          (Line <> '') and (Line <> 'Start')) or Eof(FilIn);
  end; {procedure}

procedure ConvertLine(Line : MaxString; var Line1 : MaxString);
  {convert ww output to auto ww format}
  var From,PinTo : string[4];
    Count : string[5];
    WC,Error : integer;
  begin
    From := SubStr(Line,27,31); {pin from}
    PinTo := SubStr(Line,35,39); {pin to}
    Count := SubStr(Line,44,47); {wire count, type string}
    while ((Count[1] = ' ') and (Length(Count) > 0)) do
      Count := SubStr(Count,2,Length(Count));
    Val(Count,WC,Error); {wire count as integer}
    Str(10000+WC,Count); {make wirecount into string}
    Count := SubStr(Count,2,5); {extract 4-digit # with leading zeroes}
    Line1 := ' '+From+' '+PinTo+' !'+Count; {form new line}
  end;

{******************************************************************************}
begin {program}
  PrintTitle;
  GetFileInfo;  {get input,output file info}
  InputFileHeader(Line); {get name to place at beginning of file}
  Writeln(FilOut,Line); {output it}

  {Now process file:}
  repeat
    GetLine(Line);         {get a compressed, non-comment line}
    if (Line <> '') and (Line <> chr(26)) and
        (Line <> 'END OF WIRELIST') then
      begin
        ConvertLine(Line,Line1);  {convert to auto ww format}
        Writeln(FilOut,Line1);  {output}
        Write('.'); {active status}
      end;
  until ((Line = chr(26)) or (Eof(FilIn)));

  Write(FilOut,chr(26)); {write out end of file}
  Close(FilIn);                {Close the files:}
  Close(FilOut);
  Writeln;
  Writeln('End of program.');  {to console}
end. {program}
```