bcc	title WIRELISTING		prefix/class-number.revision WL/M-22	
checked [illegible]	authors Al Goodrich		approval date 10/19/70	revision date
checked [illegible]			classification Manual	
approved [illegible]			distribution Company Private	pages 14

ABSTRACT and CONTENTS

A description of the existing wirelisting programs.

TABLE OF CONTENTS

I. PROCESS

Three programs are now used in the wirelisting process. The central program is a sorter. It's purpose is to arrange a file of signal names in alphanumeric order so that all lines of the file pertaining to a given signal name appear together.

After a file has been sorted, it is fed to a 'wirelister.' This program takes the sorted file as input and outputs the final wirelist format. It simply wires the first pin it finds to the second, etc. until the signal name changes; then starts over.

Notice that routing is determined by the order in which the pins end up after the sort. This order will be correct only if each line of the file to be sorted is of a standard format. A 'preprocessor' is provided to check and correct bad format lines of a file.

The preprocessor and wirelister are more complicated than stated here, but their basic functions have been described.

II. INPUT FORMAT AND PREPROCESSOR DESCRIPTION

Numbers in an alphanumeric sort must have a standard number of digits. The preprocessor will insert leading zeros in some cases to even the numbers out.

Formally, an output line of the preprocessor has the following format as described in a memo from Peter Deutsch on May 11, 1970:

```
line = name " /" alpha beta;
name = garbage;
alpha = cage slot type "-" pin=number;
cage = l;
slot = n n n;
type = l;
pin:number = n n n;
l = letter A-Z;
n = digit Ø-9;
garbage = anything at all;
beta = [garbage load] [garbage wrap]
       [garbage sterm] garbage;
load = "(" )"x" / ["-"] $ n) ")";
wrap = "[B]" / "[T]"'
sterm = " 5 ";
```

Or, less formally, a typical line looks like:

```
I2CLOCK' /AØØ3Z-Ø24  (4)
```

The significance of each letter and number will be described later as this is a program description.

The preprocessor is designed to save the typist time; if not otherwise stated it will supply the following to a line given it:

A. Numbers in parenthesis are made at least two digits long.
i.e., (3) becomes (Ø3)

B. The number of spaces between the signal name and the '/' is made equal to one, regardless of the number in the input line.

C. The slot and pin number are padded to become three digits long.

D. If no letters appears before the '-' a 'Z' is assumed and inserted.

Thus the above line could be input as:

I2CLOCK'/A3-24 (4)

Because of the numerous ground and voltage pins, a standard name format for each has been assigned, and a short cut typing method for input is recognized for each by the preprocessor. As described in Peter's May 14 memo:

There are three special types of lines:

A. Lines beginning with '*' are completely ignored, as are blank lines.

B. Lines beginning with '$' are assumed to be ground lines. The rest of the signal name is ignored. The alpha field is of the form ⟨cage⟩ ⟨slot⟩ [⟨type⟩] "-" ⟨pin⟩ $("," ⟨pin⟩) ["g"]. For each pin, a ground pin is output of the form "G" ⟨cage⟩ ⟨slot⟩ "(" ⟨pin'⟩ ") /" ⟨cage⟩ ⟨slot⟩ ⟨type⟩ "-" ⟨pin''⟩ where ⟨pin'⟩ is the appropriate ground pin (i.e. 4n+1 where 4n+1 ≤ pin'' < 4n+4) and pin'' is, in turn, each pin from the list.

or:

```
     $/A23-1,2,5,6,9,23
```

becomes

```
     GAØ23(Ø1)  /AØ23Z-ØØ1
     GAØ23(Ø1)  /AØ232-ØØ2
     GAØ23(Ø5)  /AØ232-ØØ5
     GAØ23(Ø5)  /AØ232-ØØ7
     GAØ23(Ø9)  /AØ232-ØØ9
     GAØ23(21)  /AØ232-Ø23
```

C. Lines beginning with "#" are assumed to be voltage lines. The treatment is similar to ground lines, except that the signal name is "V" rest of original name ⟨cage⟩ ⟨slot⟩.

or:

```
     #+5P /A3-15,17,19,2Ø,21,46
```

becomes

```
     V+5PAØØ3  /AØØ32-Ø15
     V+5PAØØ3  /AØØ32-Ø17
     V+5PAØØ3  /AØØ32-Ø19
         .          .
         .          .
         .          .
```

Thus all the pins for a given voltage and slot are wired together. Later when the panel is installed each end of the wire gets its own connection to power from a common source.

After preprocessing the file is ready to be sorted; however if the preprocessor is confused about a line, it will put it on a special file of errors which should be checked.

III. PROGRAM DESCRIPTION

The sorter is straightforward. It simply orders the lines of a file given it in alphanumeric order.

The wirelister takes the sorted file as input and outputs five files:

A. A wirelist

B. A Gardner-Denver list

All the wires to be done by the automatic wire-wrapping machine

C. A hand wire list

All the wires to be done by hand

D. A terminator list

E. An error file

For each signal the wirelist has the typical format shown here:

```
I2CLOCK':
    AØØ3Z-Ø24 (-5) TO AØØ4Z-124  (1)  [B]
    AØØ4Z-124 (1) TO AØØ9Z-ØØ3 (1)  [T]
```

The [B] and [T] designate the wrap height, bottom or top, respectively. When wiring by machine both ends of a wire must be at the same level.

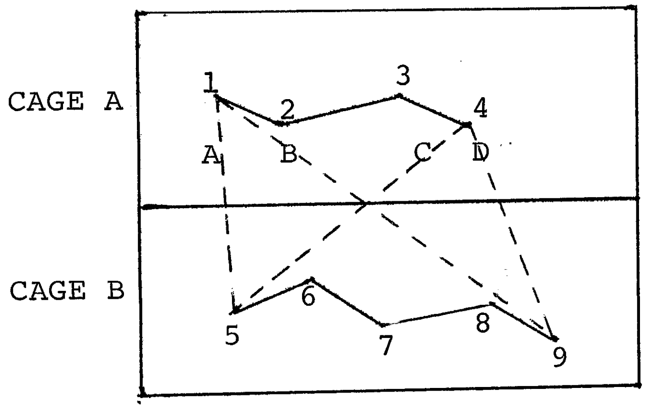
There are two exceptions to the rule that the wirelister orders the pins in the order it receives them.

A. A pin can be predefined as an end pin by putting a [B] or [T] after the pin number in the socket sheets.

K2X' /AØ23Z-Ø46 [B]

will be the end pin of all K2X' wires and it will have only a bottom wrap. The algorithm simply removes the predefined end from the sorted list of K2X', wires the remaining pins normally then wires the saved pin to the nearest then existing end of the wire. It is an error to have two or more predefined ends on one signal.

B. If the wires span two or more card cages, each cage is wired separately; then the shortest wire between the first two cages is chosen, then the next two, etc.



Out of the sorter wire 'C' would be chosen. The program however will choose 'A' as the shortest.

If there were a cage C the wire to it would originate from pin 9.

Type 'Z' and type 'G' pins are the only type pins which are accurately placed enough for the Gardner-Denver wire wrap machine. Therefore the wirelist is separated into two files:

A. Any wire whose pins are both 'Z' or 'G' go onto a 'Gardner-Denver' file to be machine wrapped.

B. Any wire which goes to a non-'Z' or 'G' pin is put on a 'hand' file to be wired by hand.

Both the Gardner-Denver file and the hand file are lists of wires only. That is all extraneous information has been removed, i.e. signal names.

Loading information may be supplied in the socket sheets in parenthesis.

```
    GM(23)' /AØ24Z-Ø94  (3)
```

The number in parenthesis is the number of T.I. 2 ma loads connected to that pin or outputs of gates, called sources, which have negative load numbers. In general, the end pins of a signal's connections which are not sources should be terminated.

The 'wirelister' will assign terminators as follows:

A. If there is no source or if some pin does not have loading information or if the load is '(X)', no terminator will be assigned.

B. A predefined terminator of ⟨5⟩ can be put in the socket sheets, in this case regular terminator analysis is bypassed and the ⟨5⟩ is assumed to be the terminator.

C. If the source is an end the other end is terminated with a ⟨1⟩ or ⟨3⟩ for sources greater or less than (-11) respectively.

D. If the source is not an end, both ends are terminated with a ⟨2⟩ or ⟨4⟩ for strong or weak sources (defined as above).

A typical terminator list entry looks like:

```
K2'/ AØØ2Z-Ø23  <2>
K2'/ AØ23Z-ØØ2  <2>
```

The terminator list is in signal order. It can easily be put in pin order by doing an 'INVERTED SORT' on the list (see Operations section for a description of 'INVERTED SORT').

Various format errors are detected and the following logical errors:

A. A signal in which every pin has loading information but no source.

B. A signal with more than one source, predefined end or terminator.

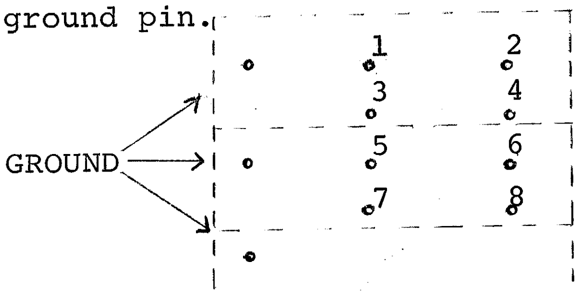
C. A signal where every pin has loading information but the positive loads add to greater magnitude than the negative source.

D. Signals with only one entry. Not put on error file, but denoted with a double asterisk on the line after the signal name in the wirelist.

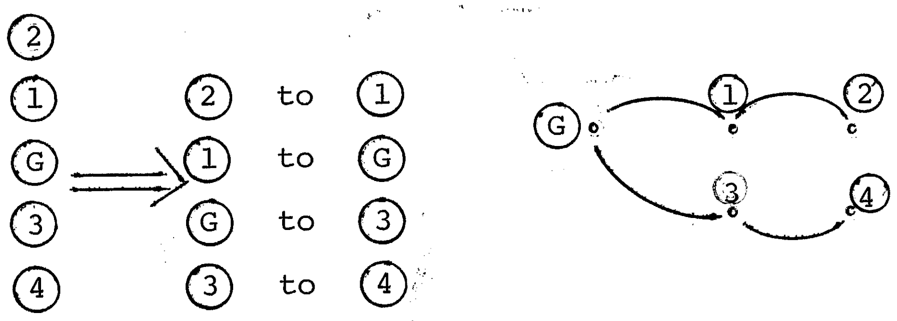
Any signal put on the error file will not appear anywhere else.

The wirelister also checks to see if the signal name is a ground pin. If it is it handles it as follows.

In the Elco panel there are four signal pins for every ground pin.



In the sorted list, only the signal pins exist so the program takes all the pins of the group of four that are grounds for a given pin and if any exist the appropriate ground pin is created and the pins are reordered and wired as follows.



Notice that the programs we have are not a general wire-listing system. They are heavily tied to our Elco panels. For this reason wiring non-standard things may be difficult. Witness Stan's master control panel; many kludges were required because the pin format is very inflexible.

IV OPERATION

The above programs now exist as a subsystem called 'WIRE'; it has Y as its herald. The commands are as follows:

```
PREPROCESS
SORT
WIRELIST
INVERTED SORT
FINISHED
```

INVERTED SORT sorts a file on pins only. This is so that a perfect set of socket sheets is obtainable from one which was not typed in perfect order and for terminator lists.

Each command is terminated with a period and then asks for all input and output files before doing its job.

Typically, a wirelist will take about 30 minutes to run.

APPENDIX

DESIGNATION:

A Ø 1 Ø Z - ØØ4

Bin Letter, Slot Number, Pin Type, Pin Number

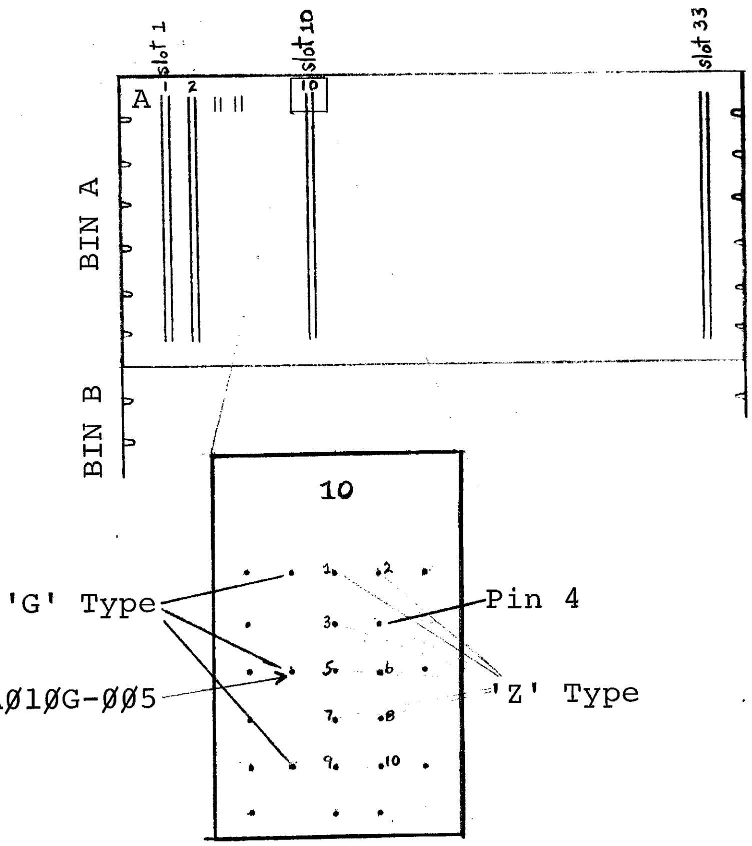
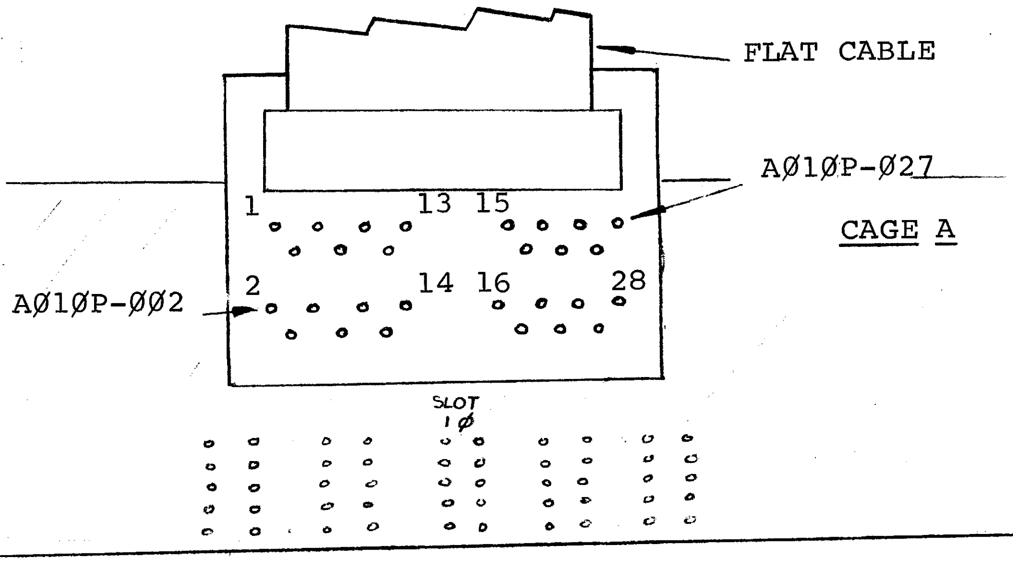


Figure 1.

Figure 1 contains a picture of the Elco panel and the basic translation from physical pin to pin designators.

The pins on a flat-cable connector are designated type 'P' instead of 'Z' or 'G' and is given the slot number over which it is centered.



At the present time there are two kinds of sources: strong (-30) and weak (-1∅). T. I.'s SN74H4∅ is at present the only strong source gates popularly used. Most inputs have a load of (1) except for 74H4∅s which has a (2) load for each input.