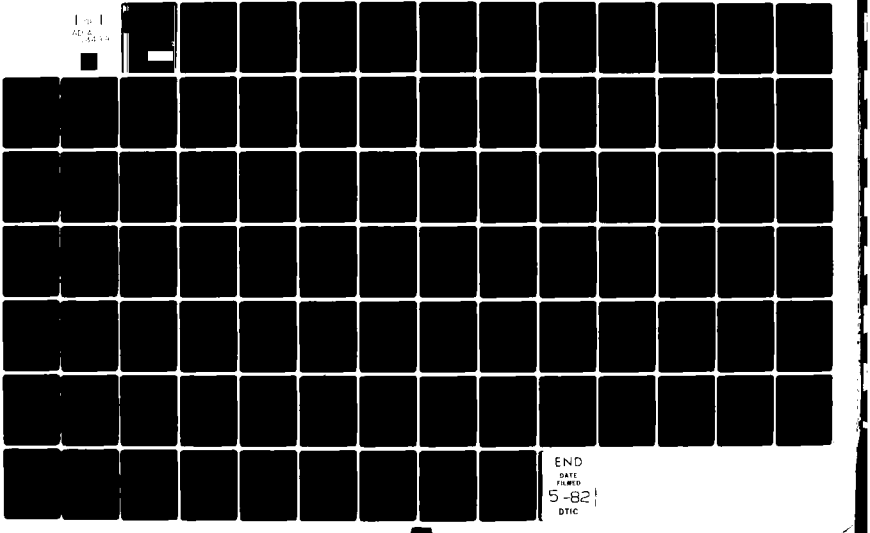
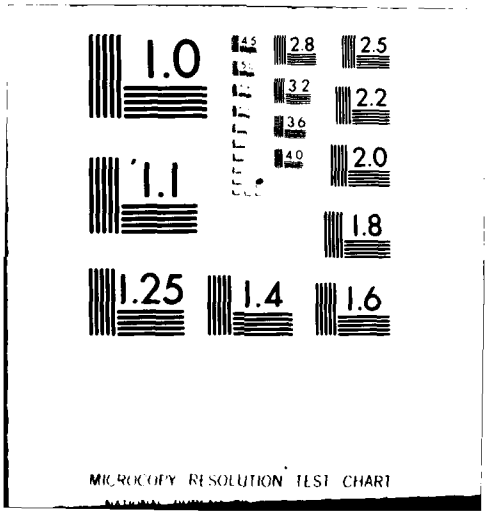
AD-A113 439 DEFENCE RESEARCH ESTABLISHMENT VALCARTIER (QUEBEC) F/G 9/5
COMPUTER-AIDED DESIGN AND FABRICATION OF WIRE-WRAP (TRADEMARK) --ETC(U)
FEB 82 B MONTMINY, R CARBONNEAU, A LAFLAMME
UNCLASSIFIED DREV-R-4245/82 NL



END
DATE FILMED
5-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

UNCLASSIFIED
UNLIMITED DISTRIBUTION

CRDV RAPPORT 4245/82
DOSSIER: 3633A-010
FÉVRIER 1982

DREV REPORT 4245/82
FILE: 3633A-010
FEBRUARY 1982

COMPUTER-AIDED DESIGN AND FABRICATION OF WIRE-WRAP® TYPE CIRCUIT BOARDS: A NEW SYMBOLISM AND ITS IMPLEMENTATION

B. Montminy
R. Carbonneau
V. Laflamme
M. Lessard
V. Blanchard

Centre de Recherches pour la Défense
Defence Research Establishment
Valcartier, Québec

BUREAU - RECHERCHE ET DEVELOPPEMENT
MINISTERE DE LA DEFENSE NATIONALE
CANADA

NON CLASSIFIÉ
DIFFUSION ILLIMITÉE

RESEARCH AND DEVELOPMENT BRANCH
DEPARTMENT OF NATIONAL DEFENCE
CANADA

CRDV R-4245/82 UNCLASSIFIED DREV R-4245/82
DOSSIER: 3633A-010 FILE: 3633A-010

COMPUTER-AIDED DESIGN AND FABRICATION OF WIRE-WRAP® TYPE CIRCUIT BOARDS: A NEW SYMBOLISM AND ITS IMPLEMENTATION

by

B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard and A. Blanchard

CENTRE DE RECHERCHES POUR LA DEFENSE

DEFENCE RESEARCH ESTABLISHMENT

VALCARTIER

Tel: (418) 844-4271

Québec, Canada February/février 1982

NON CLASSIFIE

RESUME

Nous définissons un symbolisme de codage permettant, à l'aide d'un nombre minimum d'énoncés, la description complète des nombreuses interconnexions de circuits électroniques complexes. Ce symbolisme est devenu, au Centre de Recherches pour la Défense, Valcartier, partie intégrante d'une méthode informatisée facilitant la transition entre les plans d'ingénierie électronique et la matrice pertinente d'interconnexions requise pour le montage par câblage enroulé Wire-Wrap®. Le développement de prototypes électroniques s'est vu considérablement accéléré par la préparation plus rapide des données d'interconnexions. (NC)

ABSTRACT

We define here an encoding symbolism that permits, with a minimum of statements, a thorough description of the numerous interconnections of complex electronic circuitry. This symbolism has been integrated at the Defence Research Establishment Valcartier in a computer-aided method that considerably eases the passage between an engineering electronic schematic and the related interconnection matrix required for Wire-Wrap® hardware. Electronics prototyping has been dramatically speeded up with this technique because of the time savings in the preparatory reduction of interconnection data. (U)

TABLE OF CONTENTS

1.0 INTRODUCTION

The packaging of electronic systems by conventional solder-based methods is now being rapidly phased out in numerous applications by the solderless wire-wrap technique. It basically consists in wrapping solid wires around interconnecting component socket pins thus achieving a reliable electrical contact.

At the Defence Research Establishment Valcartier (DREV), the wire-wrap technique is mainly used for construction of prototype electronic boards. The wire-wrap facility is based on a semi-automatic machine which positions with a cursor the wrapping gun at the exact location where a connection is to be made. This machine, described in Ref. 1, is now controlled (Ref. 2) by a microprocessor which communicates with the DREV central computer which is required to supply a listing of the necessary interconnections.

Although more rapid for prototyping than conventional methods such as printed circuit boards, the wire-wrap technique is handicapped by the time and effort required of the technician to generate from the engineering schematics the two-by-n interconnection matrix, containing the origin/destination of all n interconnecting wires. Moreover, considerable delays due to human errors are also quite frequent.

To fill the gap between the engineering schematics and the wire-wrap machine, a computer-aided method was devised. It is based on a symbolism which permits the full interconnection pattern of an electronic engineering diagram to be encoded with a minimum number of statements.

This symbolism is integrated into an efficient system capable of generating the interconnection matrix.

Section 2.0 of this report is devoted to the design philosophy of the method. Section 3.0 establishes the rationale behind the descriptive symbolism and proposes a syntax for it. Section 4.0 then describes the interpreter programs which decode the symbolic statements to obtain a raw interconnection matrix. Section 5.0 deals with the processing of the raw matrix to optimize the interconnection pattern and format the data for the microprocessor. Support programs for graphic display of the interconnecting data are presented in Sect. 6.0. Section 7.0 summarizes the overall implementation of the method and Sect. 8.0 presents a typical wiring session. A complete listing of all the APL programs appears in Appendix A.

This report covers part of DREV's work under PCN 33A10, Improvement to Equipment , from November 1976 to June 1981.

2.0 PHILOSOPHY

The installation of a semi-automatic wire-wrap facility at DREV provided a faster way of obtaining working prototypes from electronic schematic diagrams than the printed circuit method. However, the point-to-point manual wire-wrapping method, as initially implemented required tedious work before a circuit was ready for wiring. Part of this work, as detailed in Ref. 1, was to locate each pin to be wired on the wire-wrap board and to write the related coordinates on the electronic schematic diagram. From then on, a list showing the origin and destination of each circuit wire was generated and entered into the computer to produce a paper tape having the format required to feed the semi-automatic wire-wrap machine.

Although this method was quite acceptable for small circuits having less than 200 to 300 wires, the necessity for a computer-aided method of defining a circuit layout became more and more apparent as the size of the circuits increased. In fact, the technique was time-consuming and subject to many wiring errors which could not be discovered until the power to the board was turned on. A computer-aided method was thus devised to define the electrical circuit to be wired. This method had to produce error-free wire listings from a minimum number of definition statements while optimizing some wiring parameters, such as the length of the wires, and limiting the total number of connections per pin. Wiring errors (missing wires or wires connected to the wrong place) also had to be discovered before the circuit was wired.

The first step in the present computer-aided circuit definition eliminates the need for locating each pin to be wired on the board. Provided the computer has the pertinent information about the board and each chip's geometry, coordinate allocations for each chip's pins on the board can be straightforwardly defined by entering the coordinates of pin number one and the orientation of the chip on the board. Once all the chips' board positions have been entered into the computer, there is no further need to write each pin's location in board coordinates on the electrical drawing to be wired.

The second step required is to describe the connections to be made between the defined pins to the computer. A first approach to this consisted in entering the origin and the destination coordinates of each wire using as defining terms 1) the number assigned to a chip in the locating program and 2) the name of the pins, available from a chip data bank. This method turned out to be as tedious as the manual approach since the user still had to choose the exact placement of each wire on the board.

Many discussions about computer-aided circuit definition requirements brought out the concept of defining connection nodes. A connection node is a group of pins in an electronic circuit that have to be linked together. This definition is understood implicitly by the circuit designer as it is usual for him to identify the many points that have to be tied together on an electrical schematic diagram: starting at any point in the circuit, he follows all the routes starting from this point listing the starting point as well as the points encountered through the different routes. As will be shown later, a connection node obtained in this way may often be entered into the computer in one statement. The individual points within the node may be entered under the form of 1) a chip number and 2) either the number or the name of the pin. From a connection node statement, an interpretive program defines the actual coordinates of the pins and a link calculation program defines a circuit layout for this node so that the shortest wires are used and that no more than three connections are made on the same pin. The circuit designer can thus define a circuit to the computer without any further reference to the chip positions on the board.

On this basis, a complete language was developed to encode electronic circuits into a computer-understandable format. This language follows a few simple rules and permits an elaborate circuit diagram to be defined to the computer in the form of a computer program. These rules were established to ease circuit definition by permitting, for instance, the definition of many connection nodes through a single statement as well as the definition of one node in many statements. This last feature is particularly useful in defining connections to the power supplies since they are usually spread throughout the circuit diagram. Error identification programs were also developed to find missing wires as well as any unwanted links between two connection nodes.

3.0 ENCODING OF AN ELECTRICAL SCHEMATIC DIAGRAM INTO COMPUTER FORMAT

An electrical schematic diagram is ready to be encoded into a computer-understandable format as soon as the design is completed and the type of wire-wrap board receiving the circuit is specified. For the sake of clarity, the steps required to encode a circuit will be illustrated by an example which will be referred to throughout this chapter. This example is based on a circuit diagram (Fig. 1) containing three chips, one 40-pin connector, three discrete capacitors mounted on a 14-pin header, three power supply sources (± 5 V and + 12 V) located on an eight-pin header and an input two-pin header.

All the components of the circuit are first assigned a location on the board. For this purpose, all the chips and the sockets for discrete components are drawn to scale on a blank drawing of the receiving board. Each is assigned a number which will identify it in all the further steps of the circuit encoding. This is illustrated in Fig. 2 where the chips and the sockets are drawn on a blank matrix board and are numbered from 1 to 7. The different types of boards that are currently used at DREV are described in Sect. 3.1.

The next step is the writing of a location program which provides the computer with the location of all the pins of the circuit, using one statement for each chip. The syntax rules for the location program are given in Sect. 3.2.

The last step to encode the circuit is to specify to the computer the electrical connections of the circuit by means of a definition program whose syntax rules are explained in Sect. 3.3.

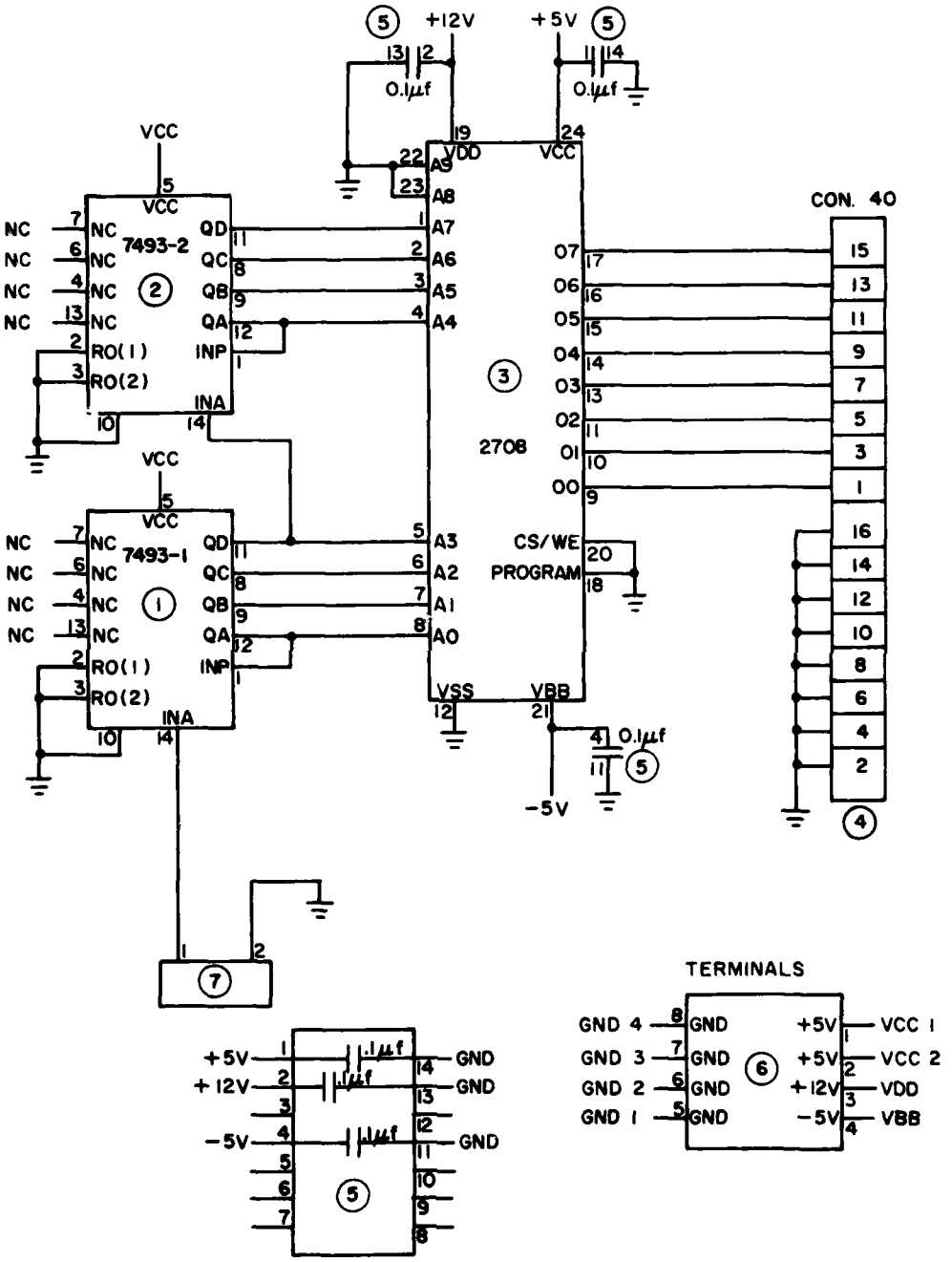


FIGURE 1 - Example of an electronic schematic diagram ready to be encoded into computer format

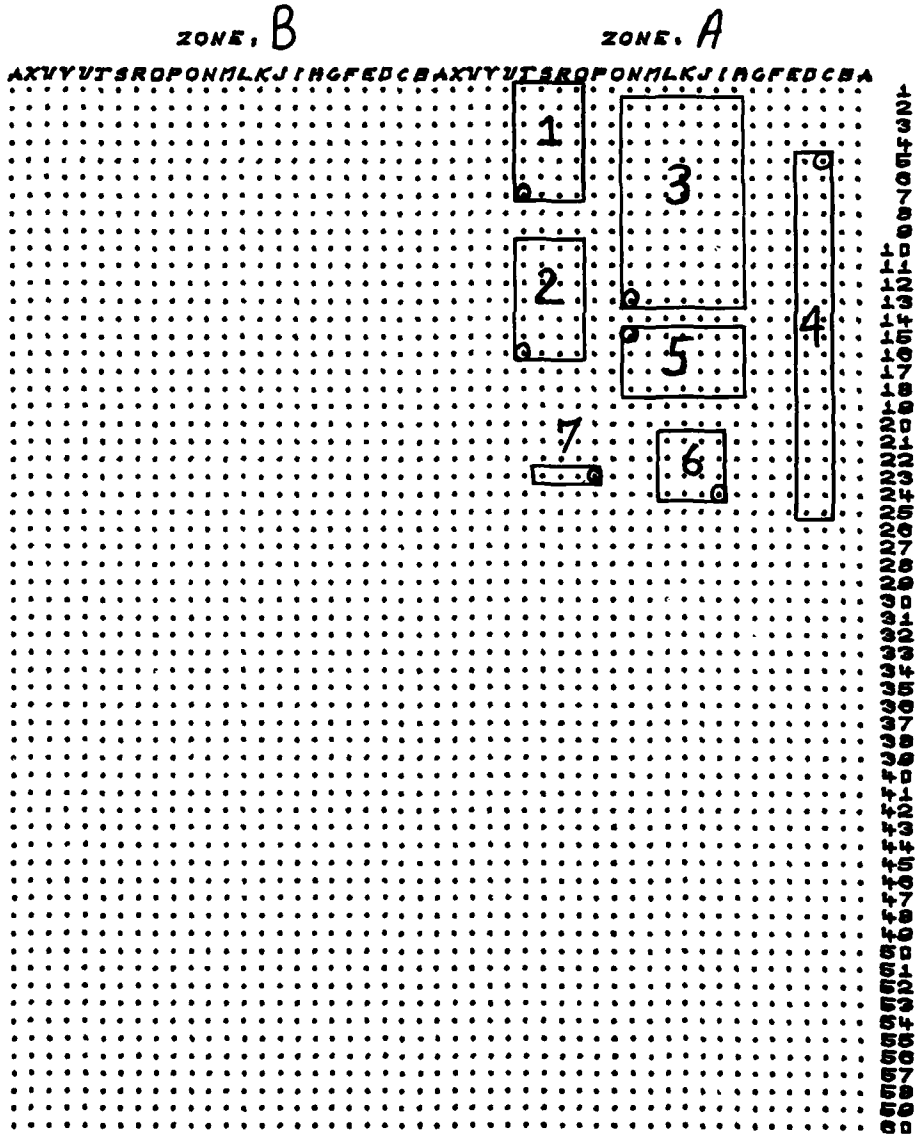


FIGURE 2 - Location of the components on a blank matrix board

3.1 Description of the Boards

The wire-wrap facilities implemented at DREV permit the use of three types of boards.

The type 1 or matrix board (Fig. 3) is a general-purpose pre-punched insulating board with a matrix of holes spaced 0.1 in center to center. Sockets are mounted on the board which may be cut to any size and shape to fit particular packaging constraints. Each hole can be exactly defined by means of three parameters: a zone, a column and a row. The dimensions of these panels permit a maximum of six zones identified by the letters A to F. Each zone may have up to 84 rows identified by numbers 1 to 84 and 24 columns identified by letters A to X.

The type 2 or Scanbe board (Fig. 4) is a Scanbe universal socket panel number LPU-54-M-A or C. Augat U-Series board number 8136-UG6-54 may also be used. The Scanbe board is divided into six zones identified by the letters A to F. Each zone contains nine 50-pin rows identified by the letters A to H and J. Each pin is exactly defined by the zone, the row and the pin number of the row (which ranges from 1 to 50). The two upper connectors on each zone have been assigned numbers 211 and 212 for zone A, 221 and 222 for zone B, up to 261 and 262 for zone F. The names of the pins for those connectors are P1 to P26. Also, VCC and GND busses on each zone have been assigned numbers 213 and 214 for zone A up to 263 and 264 for zone F. Each pin of these busses is identified by the name V1 to V12 and G1 to G12.

The type 3 or SBC board (Fig. 5) is an Intel SBC 905 universal prototype board for interfacing custom hardware to the iSBC series cards.

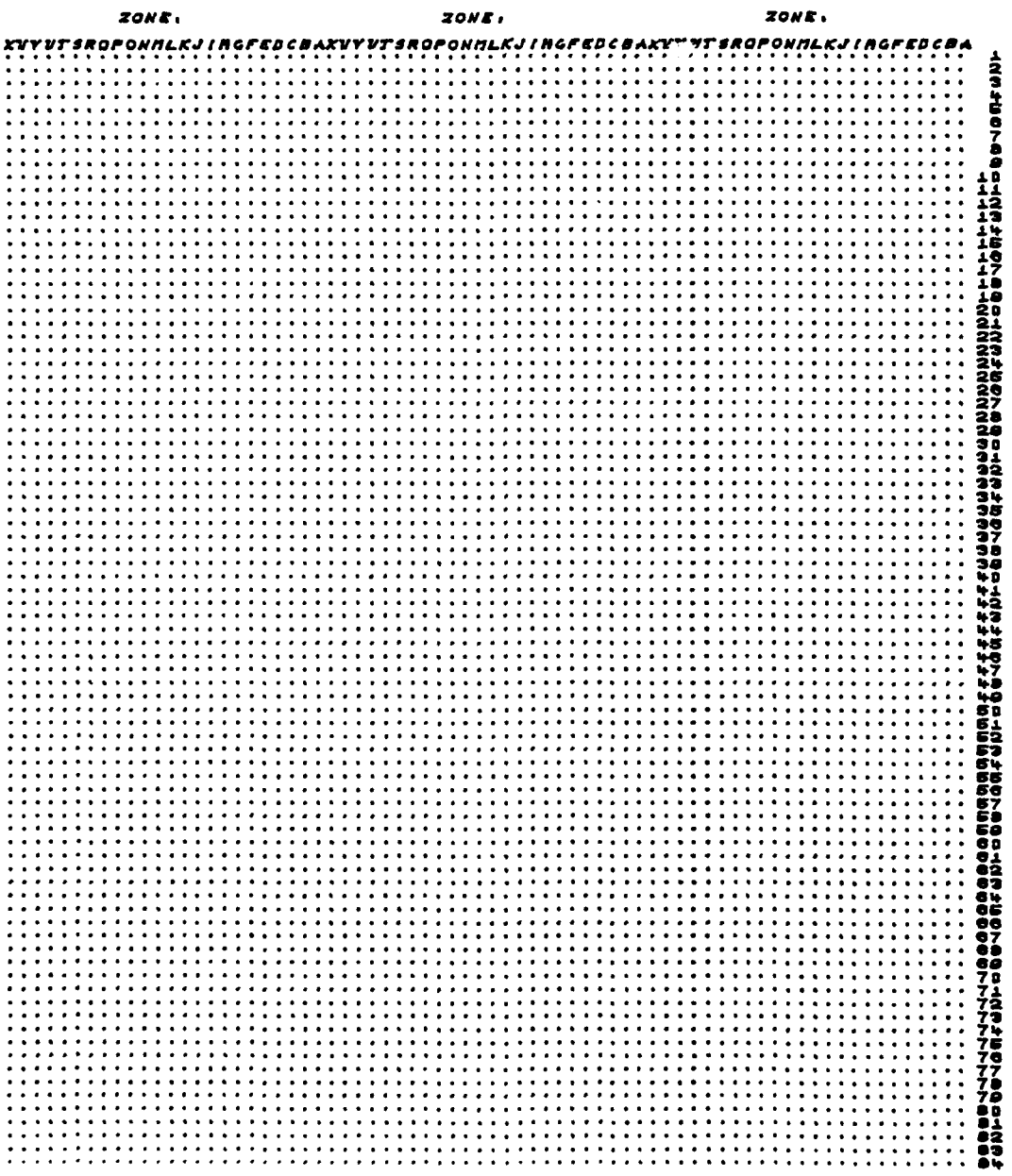


FIGURE 3 - Blank matrix board

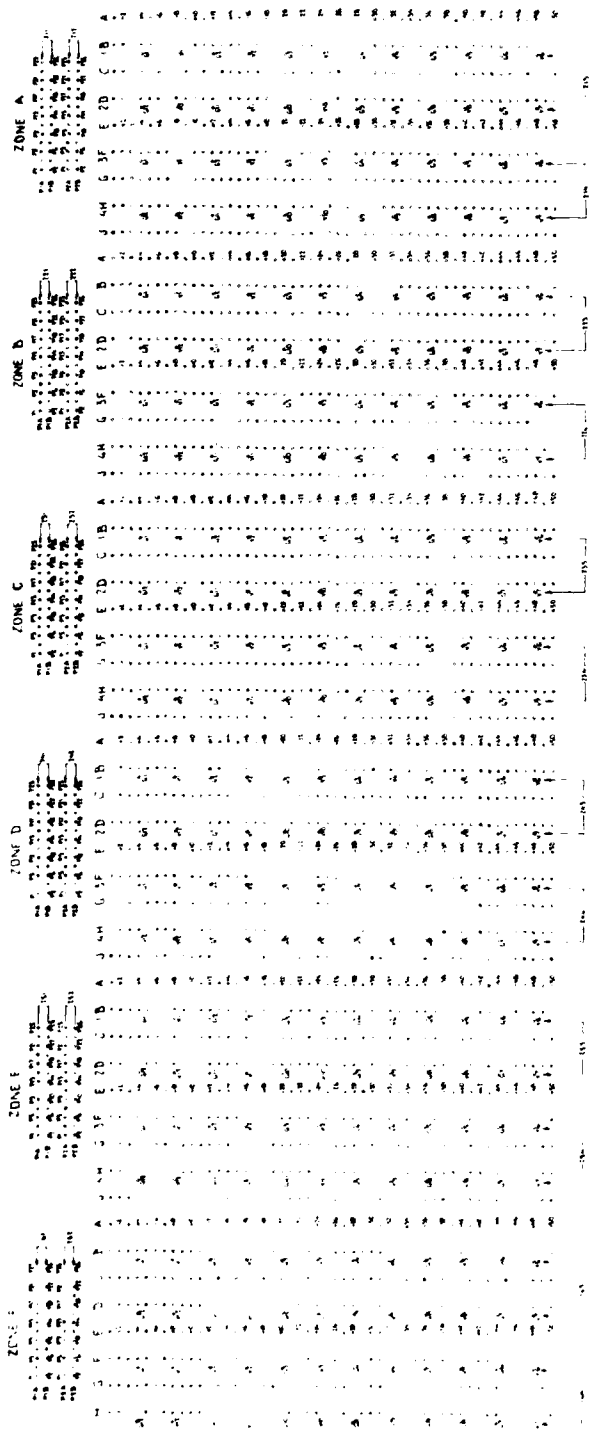


FIGURE 4 - Blank Scanbe board

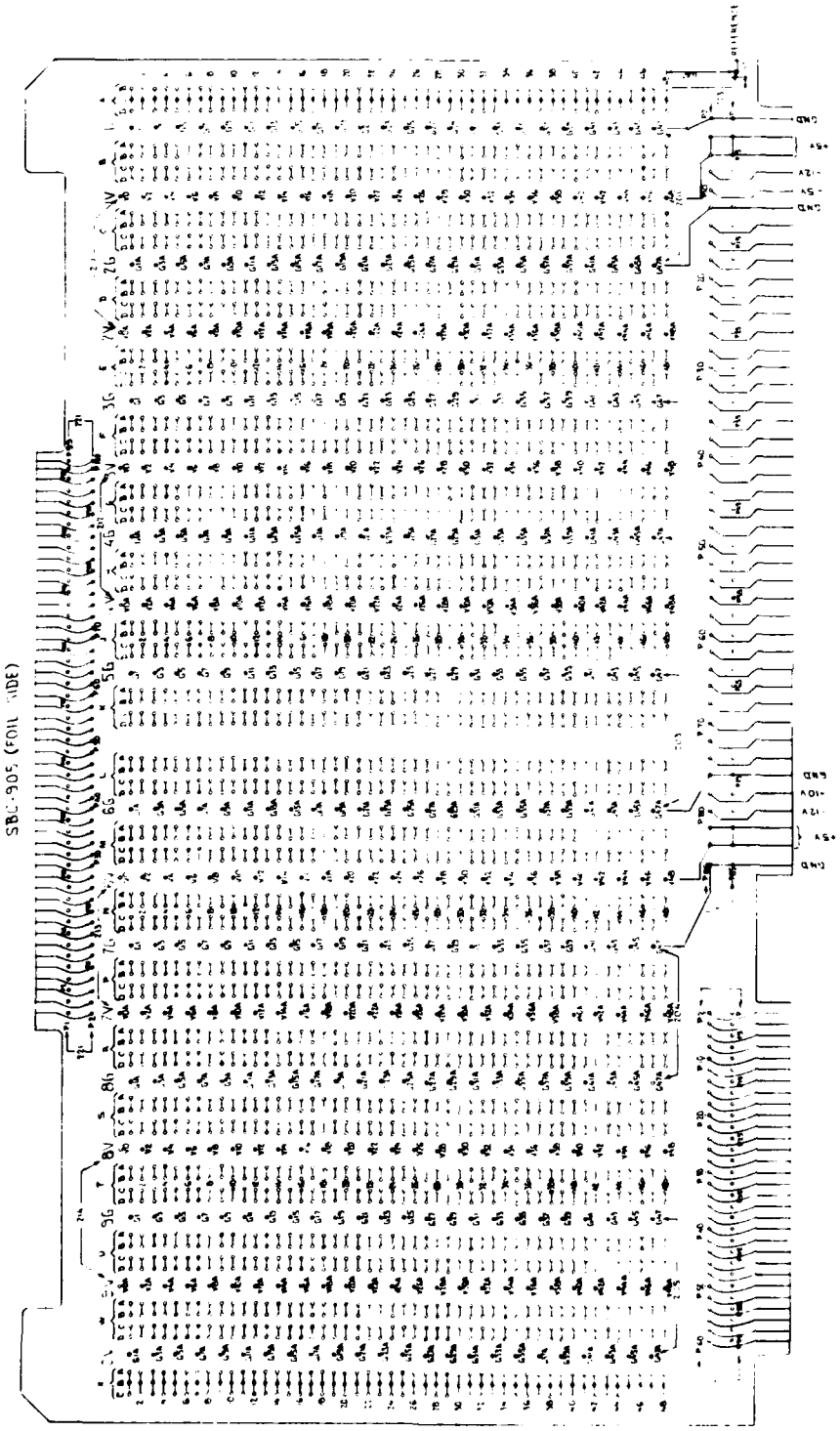


FIGURE 5 - Blank SBC board

The SBC board is divided into 20 zones identified by letters from A to X. Each zone contains four 48-pin rows identified by the letters A, B, C and D, except for zone A which contains only rows B, C, and D, as well as zone X which contains only rows A, B and C. Each pin is exactly defined by the zone, the row and the pin number in the row which ranges from 1 to 48.

The three connectors provided on this board are assigned the numbers 221, 222 and 223. The names of the pins for these connectors are P1 to P100 corresponding to pins 1 to 100. Each pin in these connectors may thus be referred to in the definition program by the connector number and the name of the pin.

The SBC board contains eight 25-pin rows of VCC sources and ten 24-pin rows of GND sources. The VCC sources are organized into four groups numbered from 211 to 214 whose pin names are VO, V2, V4,... V48 and VOA, V2A, V4A,... V48A. The GND sources are organized into five groups numbered from 201 to 205 whose pin names are G1, G3, G5,... G47 and G1A, G3A, G5A,... G47A. All these pins may thus be identified by the user in the definition program by the corresponding group number and the name of the pin.

The numbers for the connectors and VCC and GND busses that appear on the blank drawings of type 2 and 3 boards along with the names assigned to the pins are recognized by the computer without any user intervention in the location and definition programs. Moreover, VCC and GND busses have the status of terminals (see Sect. 3.3.5) named VCC and GND. They must not be redefined by the user in the definition program.

3.2 Identification and Location of the Components

All the circuit components that are to be wired as well as their positions on the board are defined in a location program written in APL and named by the user. It contains as many lines as there are chips on the board. In the following context, the meaning of the word 'chip' is extended to define a microcircuit, a connector or a header on which discrete components are mounted.

Each line of the location program shall be of the form: L'A,B,C,D' where L is itself an APL function which generates the necessary data to specify the board pins to be wired. Its action is detailed in Sect. 4.1.

Parameter A is an identification number assigned to a chip. It can be any positive integer except when a type 2 or type 3 board is used. In these cases, numbers between 200 and 265 are prohibited because many numbers in this range are already assigned to the connectors and power supply strips of these boards.

Parameter B represents the name of the chip. It can be any series of up to six characters, i.e.: 8085, 74S188, LM311, etc. Headers and connectors are assigned names as HEAD14 for a 14-pin header and CON26 for a 26-pin connector, for example.

Parameter C gives the coordinates of the location of pin number 1 of the chip as defined in Sect. 3.1.

Parameter D is an orientation code for the chip, represented by the numbers 1 to 4 and having the meaning shown in Fig. 6.

When the location program is executed, all the chips that are not already defined in the main data bank must be entered by the operator answering a series of questions asked by the program. Thus, the number of chips stored in the main bank increases each time new components are wired.

The location program for the circuit in the example is presented in Table I. The arbitrary name that has been chosen for the APL function is TESTLOC.

When a comment is to be inserted in the location program, the function L can accept an argument of the form: L'A COMMENT'.

3.3 Definition of the Connections

All the electrical connections to be performed are similarly defined in a definition program which is also an APL function whose specific name can be chosen by the user. Each line of the definition program must have the form W 'statement', where W is itself an APL function that will be called upon later to interpret the statement. Its action is detailed in Sect. 4.2.

The six types of statements are now described. The following syntax rules for defining these statements actually constitute a new language for encoding an electrical schematic diagram in a computer format.

TABLE I

Example of a location program TESTLOC

```
      ∇TESTLOC[□]∇
∇TESTLOC
[1]    L'1,74LS93,AT7,3'
[2]    L'2,74LS93,AT16,3'
[3]    L'3,2708,AN13,3'
[4]    L'4,CON40,AC6,1'
[5]    L'5,HEAD14,AN15,4'
[6]    L'6,HEAD8,AI24,2'
[7]    L'7,HEAD2,AP23,1'
∇
```

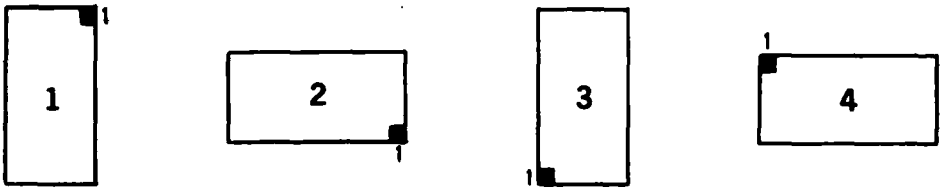


FIGURE 6 - Orientation codes for the chips

3.3.1 The Free Path Statement

The free path statement permits the definition of many points to be linked by the shortest possible path, the actual path being selected freely by the computer.

A typical free path statement is of the form:

$N_1\ P_1\ P_2,\ N_2\ P_3\ P_4,\ N_3\ -\ P_5$

where N_1, N_2 and N_3 are chip numbers, as specified in the location program, and P_1 to P_5 are either a pin number or the name of a pin appearing in the main bank. The hyphen between N_3 and P_5 is used as a no-connection indicator. The free path statement creates one or many connection nodes. This particular example defines two connection nodes: the first one is chip N_1, pin P_1 to chip N_2, pin P_3 and the second one is chip N_1, pin P_2 to chip N_2, pin P_4 to chip N_3, pin P_5. The first node contains two points to be linked while the second contains three points to be linked using a combination of the shortest wires possible. The free path statement is the one most commonly used to specify a circuit.

3.3.2 The Fixed Path Statement

The fixed path statement permits the definition of many points to be tied together following a route predetermined by the user.

This statement has exactly the same form and syntax as the free path statement except for the symbol u at the beginning: $u\ N_1\ P_1\ P_2,\ N_2\ P_3\ P_4,\ N_3\ -\ P_5$. This statement defines one or many nodes of connections and states the order in which each path will be

made by joining chip N_1, pin P_2, to chip N_2, pin P_4, to chip N_3, pin P_5 regardless of the length of the wires needed. Fixed path statements are used when it is necessary to control the order in which the pins are connected.

3.3.3 The Macro Statement

The macro statement permits the assignment of a long string of characters to a variable name which may be used many times in other types of statements.

A typical macro statement is of the form:

$DB \leftarrow D_0\ D_1\ D_2\ D_3\ D_4\ D_5\ D_6\ D_7.$

Its identifier is the specification arrow symbol (←). In the above example, it permits a series of characters (D_0 to D_7) to be represented by the variable DB. Each time this variable is encountered in another statement, it is eventually replaced by the specified string of characters. In this way, statements of a repetitive nature can be readily shortened.

Macro statements are most useful in the definition of data and address busses.

3.3.4 The Class Statement

The class statement enables the user to assign certain pins to a common class. All the pins so assigned, although they may appear in many statements, become part of the same connection node and will be connected.

A typical class statement is of the form:

3 GND, 5 OE EN, 8 A1 B1 C1: GND.

Its identifier is the colon (:). This statement assigns the pins defined in the left argument to a class whose name forms the right argument.

The number at the beginning of the left argument or numbers following a comma (3, 5 and 8 in the above example) are chip numbers that have been assigned in the location program. The data following each chip number identify the pins of the chip referred to. The pin identification may be either the pin number or name, as defined in the main bank. In the above example, chip 3 pin GND, chip 5 pins OE and EN, and chip 8 pins A1, B1, C1 are assigned to the GND class.

The class statement permits the definition of one connection node in several statements. It is very useful when many pins forming part of the same connection node are spread throughout the electrical schematic diagram, such as the pins going to the power supply sources.

3.3.5 The Terminal Statement

The terminal statement designates a certain number of pins in a circuit to act as terminals or sources.

A typical terminal statement is of the form:

VCC = 1 V1 V2 V3, 2 14 15, 14 6.

Its identifier is the equal sign (=). The right argument of the statement is written using the same syntax rules as for the left argument of the class statement.

The left argument of the statement is the collective name assigned to the series of terminals defined on the right side. This name is also normally assigned to a class (Sect. 3.3.4) in such a way that all the points of that class are linked to at least one terminal pin. The pins defined as terminals are not to be linked, as they are themselves sources and are already interconnected outside the wire-wrap circuit board.

3.3.6 The Comment Statement

The comment statement permits the user to insert comments within the definition program.

The comment statement is of the form:

'⍝ COMMENT'.

It is identified by the APL comment symbol ⍝ at the beginning. The comment begins right after this symbol.

The definition program for the circuit in the example is presented in Table II. The arbitrary name chosen for the APL function is TESTDEF.

TABLE II

Example of a definition program TESTDEF

```
        ∇TESTDEF[□]∇
∇TESTDEF
[1]    W'⍵EXAMPLE OF A DEFINITION PROGRAM FOR A CIRCUIT'
[2]    W'⍵MOUNTED ON A MATRIX BOARD AND WHOSE LOCATION'
[3]    W'⍵PROGRAM IS TESTLOC'
[4]    W'7 1,1 14',0⍴'**************FREE PATH******'
[5]    W'1 QD,2 INA,3 5'
[6]    W'1 1,1 12,3 8'
[7]    W'2 INB,2 QA,3 A4'
[8]    W'∪5 1,3 VCC',0⍴'************FIXED PATH*****'
[9]    W'∪5 14,3 12'
[10]   W'∪5 2,3 VDD'
[11]   W'∪5 4,3 21'
[12]   W'2 11,3 1'
[13]   W'5 11 13:GND2',0⍴'**********CLASS**********'
[14]   W'3 12 18 20 22 23:GND3'
[15]   W'1 5,2 5:VCC2'
[16]   W'M1←8 9',0⍴'****************MACRO**********'
[17]   W'M2←2 3 10'
[18]   W'M3←9 10 11 13 14 15 16 17'
[19]   W'M4←1 3 5 7 9 11 13 15'
[20]   W'1 M2,7 2,2 M2:GND1'
[21]   W'1 M1,3 6 7'
[22]   W'3 M2,2 8 9 ⍵'
[23]   W'3 M3,4 M4'
[24]   W'4 2 4 6 8 10 12 14 16:GND4'
[25]   W'3 24:VCC1'
[26]   W'3 19:VDD'
[27]   W'3 21:VBB'
[28]   W'VCC1=6 1',0⍴'**************TERMINAL*******'
[29]   W'VCC2=6 2'
[30]   W'VDD=6 3'
[31]   W'VBB=6 4'
[32]   W'GND1=6 5'
[33]   W'GND2=6 6'
[34]   W'GND3=6 7'
[35]   W'GND4=6 8'
∇
```

4.0 INTERPRETER PROGRAMS

The location and definition programs described in Sect. 3.0 contain all the encoded information required to generate the interconnection matrix. The location and definition statements must be decoded by two APL interpreters which will now be described.

4.1 Interpreter for the Location Program

The decoding of the location program is performed one statement at a time by the interpreter program L, whose listing may be found in Appendix A. The operator is first requested to supply the name of the location program and then the interpreter decodes each successive statement and gathers all geometrical and chip-related information about the circuit in six transfer vectors and one matrix. These serve as the basic information for interpreting the definition program (Sect. 4.2). The component drawing programs also use these vectors and this matrix (Sect. 6.0).

Each location statement is composed of a four-field alphanumeric argument on which acts the program L. The individual fields A, B, C and D are easily extracted because of the comma (,) separators.

The A field is a number arbitrarily assigned to identify each chip. The A fields of the successive location statements are accumulated to form the transfer vector LISTNUM. Assuming an N-component circuit, LISTNUM will be an N-element vector containing the identification number of each chip.

The B field contains the name of the chip referred to in the location statement. All names and pin identifications of the chips that have been used at least once are accessible to the program L in a main data bank and its directory (Sect. 7.0). The chip name decoded from the

B field can be immediately validated if this chip type belongs to the directory. On the other hand, if this chip is being used for the first time, control is transferred to an interactive interrogation program (INTERROGE, Appendix A) which asks for the new chip's specifications; that is, its geometry and an identification name for each pin. The chip's geometry is selected from a choice of preprogrammed configurations such as dual-in-line 14-pin. The new chip information is then inserted into the main chip bank and its directory, all located in APL files (Sect. 7.0).

A matrix called BANQUE is also formed by the program L to regroup those chip descriptions of the main chip bank that are in current use in the wire-wrap task. An N-element transfer vector, BASEVEC, is then generated to identify where the information pertinent to each chip begins in the unformatted matrix BANQUE. A 2N-element transfer vector, BASEMAT, is also formed to indicate the physical dimensions of each chip (in 0.1-in units) to the connection definition interpreter.

The C field contains the board coordinates for pin number 1 of the selected chip. The coordinate coding system is specific to each receiving board (Sect. 3.1) and is transformed by the program L to a numerical cartesian coordinate system common to all types of boards. The line and column numbers defining the location of pin number 1 of each chip is transferred to the definition interpreter by the 2N-element vector BASEPIN1.

The D field is decoded to obtain the orientation code (1 to 4) of each chip. This code is transferred to the connection definition interpreter via the N-element vector BASEORI.

The last transfer vector is the N-element vector BASESPA. It gives the spacing increment (in 0.1-in units) between the pins of each chip. Most often, it is composed of unity elements except for the special cases mentioned below.

The variable N, used throughout the above description of the transfer vectors, may not be equal to the number of statements in the location program. A first case occurs if any comment statement is present in the location program. These statements are skipped by the program L. A second case occurs when either a Scanbe or a SBC-905 is the receiving board in use. For these boards, the geometrical information on the special connectors present (Sect. 3.1) is automatically added to the transfer vectors. The special connectors receive the same status as chips explicitly defined by statements in the location program. As several of these connectors do not have a 0.1-in increment between their pins, the use of the above-described transfer vector BASESPA is justified.

During the decoding of the location program, three types of errors may be reported.

Type 1 error message reports that the statement encountered comprises an incorrect chip identification number, either because of duplication or an incorrect syntax. It has the form:

INCORRECT CHIP IDENTIFICATION NUMBER AT LINE N

where N is the number of the statement in error.

Type 2 error message reports that the location selected for pin 1 of the concerned chip is incompatible with the receiving board coordinate system. No verification is done on the other pins of the chip and on compatibility with the location of the previously defined chips however. This error message has the form:

INCORRECT LOCATION AT LINE N.

Type 3 error message reports that the orientation code selected for the concerned chip is not valid, i.e. it is not 1, 2, 3 or 4. This error message has the form:

INCORRECT ORIENTATION AT LINE N.

4.2 Interpreter for the Definition Program

The definition program constitutes a formal description of the circuit interconnections required. The decoding of the definition program automatically follows that of the location program and is done by a group of functions listed in Appendix A, including the one named W which has been inserted into each line of the definition program. The operator is simply requested to supply the name of the definition program.

Section 3.0 has explained the six types of statements which can be found in the definition program, namely 'free path', 'fixed path', 'macro', 'class', 'terminal' and 'comment'. The W function first determines the type of statement by looking for the identifier and then stacks the argument of the statement in an appropriate character string. Since the comment statements are ignored when all statements have been processed, the result is a division of the definition program into five character strings, each containing all the statements of the same type.

The macro string is made up of all the definitions of the variables used in place of other characters, as explained in Sect. 3.3.3. The next step is to search the four other strings for occurrences of the variables' names and to replace the names with the actual characters they represent. The variables' names are processed one at a time and removed from the macro string when they are no longer present in the other four strings. When this action is completed, the number of strings has been reduced from five to four.

Defining a cluster as a group of pins which have to be interconnected, the four strings contain the following information:

1) 'free path' string: all the statements which define pins (belonging to one or more clusters) to be connected by the shortest path(s) within each cluster;

2) 'fixed path' string: all the statements which define pins (belonging to one or more clusters) to be connected in the stated order within each cluster;

3) 'terminal' string: all the statements which define pins belonging to a specified cluster but which are not to be connected between themselves;

4) 'class' string: all the statements which define pins belonging to a named cluster (the repeated use of a name forces a joining of the relevant clusters).

At this point, the program inspects each string in detail and extracts each pin's specification; with the help of the transfer vectors and matrix generated by the location interpreter (Sect. 4.1), it then turns this specification into pin coordinates, tags each pin with codes specifying whether it is a terminal or not and whether it belongs to a free or fixed path, and finally places it in the appropriate cluster, together with the line number of the definition program in which the pin was originally specified.

This table of pin coordinates and tags is then put on file for use by the circuit calculation program described in the following section.

During the decoding of the definition program, four types of errors may be reported.

Type 1 error message reports that the statement encountered cannot be classified in any of the defined types and has the form:

SYNTAX ERROR IN STATEMENT N

where N is the number of the statement in error.

Type 2 error message reports that an inconsistency has been encountered in a fixed or free path statement and has the form:

STATEMENT N IN ERROR.

Type 3 error message reports that a specified chip number is not among those generated by the positioning program and has the form:

ERROR AT LINE N

CHIP NUMBER M IS NOT DEFINED

where M is the erroneous chip number.

Type 4 error message reports that a specified pin name is not the same as any of the pin names of a specified chip number and has the form:

ERROR AT LINE N

CHIP NUMBER M

(name of pin No. 1 of specified chip No.)

(name of pin No. 2 of specified chip No.)

.

.

.

(name of pin No. L of specified chip No.)

PIN NAME

where NAME is the specified pin name, and L is the number of pins on the chip.

5.0 CIRCUIT CALCULATION PROGRAMS

Section 4.0 has described how the interpreter programs process the information contained in the location and definition programs into numerical data presented as a table of pin position coordinates, type of pins (terminals or not), cluster appartenance (class statement), type of connections (free or fixed) and origins (line numbers of the definition program which created the pins). In this section, we review the steps needed to convert this raw data into a final circuit layout and produce, in its final form, the information required by the operator and the microprocessor for the wiring of the board.

5.1 Node Determination

The interpreter programs have already assigned each pin to a cluster; however these clusters are not true nodes in the electrical sense of the word as the operator may specify indirectly connections between them simply by repeating on one line of the definition program a pin which has already been defined on another line. The first step is to examine the position coordinates of all pins in all clusters and to fuse clusters that have common pins while removing all pin repetitions. The result is a new grouping of the pins such that each new cluster is a complete set of the pins to be connected, with each pin being listed only once. These new clusters are now true nodes and will be referred to as such from here on. This step terminates with the printing of a list giving the line numbers of the definition program in which common pins have been discovered. This list greatly facilitates the operator's task of finding out if unwanted connections between different circuits have been accidentally introduced into the definition program.

5.2 Link Calculation

Now that all the nodes have been defined, the next step is to examine each of them individually to determine the configuration of the wires (links) which will connect all the pins in the node. This is a complex problem and the solution is of vital importance to the efficient production of satisfactory circuit boards.

5.2.1 The Link Problem

Consider a node comprised of n pins. The problem can be described simply as: link all n pins in such a way that:

a) the total length of the links is as short as possible and,
b) there are at the most three links attached to a given pin.

Condition b is a constraint arising from the finite physical length of the pins around which wires (links) are to be wrapped.

Condition a is a difficult one to meet. In fact, if the constraint expressed in condition b is changed to read that the maximum number of links attached to a pin is two instead of three, the resulting network of links will have the form of a single chain (without any branching). The problem then becomes similar to the familiar one of the traveling salesman who must visit n cities and return to his point of departure by the shortest route. It is well known that the number of paths to be investigated is $\frac{(n-1)!}{2}$ which becomes an excessively large task for large values of n (for n = 50, $n! = 3 \times 10^{64}$).

In our case, since the path remains open (a return to the starting point is not required), the number of possible paths actually rises to $\frac{n!}{2}$ since to each closed path of the salesman (in which the starting point is immaterial) there correspond n paths depending on the starting point. Furthermore, by permitting up to three links on a pin, branching can occur which introduces numerous new possibilities so that the number of paths increases even more rapidly.

Since it is common for real circuits to have nodes of more than 50 pins (the 'ground' node for example) and an exhaustive calculation of all possible paths to find the shortest one is clearly too lengthy, a shorter method to arrive at an acceptable result is required.

5.2.2 An Imperfect but Simple Solution

Consider once again a node comprised of n pins. Each pin has (n-1) neighbors to which it could potentially be linked, hence the total number of possible links is n(n-1). However, since direction does not have to be taken into account (the link from pin 1 to pin 2 is the same as that from pin 2 to pin 1), the number of distinct links to be examined is only $\frac{n(n-1)}{2}$. Since each link involves two pins, the number of links required to join all n pins is (n-1). The problem can then be restated as follows: select from the $\frac{n(n-1)}{2}$ possible links the (n-1) links which best satisfy the two conditions mentioned above.

The tactic adopted is as follows:

1) identify each possible link by its end points (pins) and calculate all distances between these pairs of points; even for a 50-pin node, this is a relatively small task, since it involves only $\frac{50\ (50-1)}{2}$ = 1225 distances;

2) tabulate the possible links in order of increasing distances (link lengths);

3) start examining links one by one from the top of the table (shorter links) and accept the link if it is 'valid'; the validity tests are:

   a) the link must not close any part of the path being defined on itself; otherwise redundant paths would be established which would evidently lengthen the total path without taking care of any new points;

   b) each end point of the link must have been used fewer than three times already;

4) stop the process when (n-1) valid links have been accepted; the path is completed and all pins are connected.

This method of assembling the path from the shortest valid links produces excellent results. In fact, if the (n-1) shortest links are all valid, the path is evidently the ideal or minimum-length one. Even when some of the early links do not pass the validity tests and the table has to be searched past the $(n-1)^{th}$ term to collect the (n-1) valid links required, the result appears to come very close to the ideal path length, or to equal it, in most cases. Unfortunately, except for trivial cases, the deviation from the ideal cannot be measured since the ideal path cannot be readily determined! However, it can be shown that for some special circuit configurations, the path is slightly longer than the ideal one and thus the method leads only to a near optimal, rather than an exact, solution.

Another indication that the solutions obtained are quite reliable comes from observations of path self-crossings. Appendix B demonstrates that paths which fold over themselves are not the shortest ones and can be readily shortened. Dozens of boards have been processed up to now and in the hundreds of node paths produced, none has yet been seen to cross itself, but again this is only an indication rather than a proof of the quality of the paths obtained. Appendix C shows a possible circuit configuration where the method would fail and actually produce both a crossing and a noncrossing path longer than the ideal one.

5.2.3 Treatment of 'Fixed Paths'

The method described above is obviously not required for connections defined as 'fixed paths' as explained in Sect. 3.3.2. The distances between these pins need not be calculated and these imposed paths are simply recognized as such and selected on a priority basis before any other selection is made. Nevertheless, the distances between the pins belonging to the fixed paths and the other pins are still calculated and the normal selection process is applied so that links from other pins can branch into the fixed paths if this is found advantageous.

5.2.4 Treatment of 'Terminals'

Some of the pins of any node can be defined as 'terminals' and have a special property: they must not be linked since they belong to sources and are already interconnected outside the wire-wrap board (see Sect. 3.3.5). The program handles these as follows:

a) distances between terminals are not calculated;
b) distances between terminals and other pins are calculated;
c) temporary links are defined between all terminals;
d) the normal link selection process is performed and the links defined in c assure that other links will not interconnect the terminals, whereas the other pins are linked to at least one of the terminals;
e) the temporary links between terminals are discarded.

5.2.5 Treatment of Large Nodes

When processing a large node, the manipulation of the $\frac{n\ (n-1)}{2}$ distances could require more storage space than currently provided in the APL workspace. To prevent the program from stopping for this reason, an arbitrary limit on node size has been set at 50 pins and when a node exceeds this limit, it is broken down into fragments and worked on piecewise.

To prevent wire crossings and unnecessarily long paths, this fragmentation of the node is done on a geographical basis, i.e. each fragment must be comprised of neighboring pins in a region of the board and the fragments must not overlap. However, the order in which the pin coordinates are supplied by the earlier programs depends only on the actual sequence used in the definition program, on the interpreter program action and on the 'cluster fusing' activity described in Sect. 5.1. The first step is to reorder all pins in both coordinates so that a clean split between the fragments can be effected. The number of fragments is chosen such that each fragment contains approximately the same number of pins, but fewer than 50.

Each fragment of the original node is then processed as a node by itself producing a sub-path and finally, there only remains the tying together of each of the sub-paths obtained. This is done by choosing approximately half the pins of two adjacent fragments, the selected pins being those closest to the common border. This creates a new piece which overlaps the two generator fragments and which again is processed as a node. In this case distances are calculated only between pins on opposite sides of the border and the link selection process is stopped as soon as one valid link is identified, since this is all that is required to effect the connection between the two sub-paths. This action is continued until all sub-paths have been connected to form a single path for the original large node. The program has been arranged so that this fragmentation can go along without interfering with the processing of 'fixed paths' or 'terminals'.

This piecewise processing of large nodes naturally introduces a further risk of producing nonminimum-length paths. In fact, there are large node configurations for which the path produced with this method are not ideal. However, the minimum size of a node fragment is 25 pins (when a node of 51 pins is split in half) and a node of this size still offers so many interconnection possibilities that the path produced usually comes very close to the ideal.

5.2.6 Node Layout Drawing

As mentioned above, the links are calculated node by node and features have been included for showing the actual path selected by the program on the screen, if the operator so wishes.

All paths are shown on a common scale which is automatically adjusted to cover the outermost pins of the board. Each node pin is plotted as a letter of the alphabet which changes when a new node is started and turn-around is provided if more nodes are processed than letters are available. The links appear as straight lines joining the letters. Terminals are shown as small rectangular boxes (the 'quad' symbol) irrespective of the node to which they belong. The operator also has a choice of displaying the nodes superimposed or one by one. This last feature requires the operator's intervention to pass from one node to the next and this affords him the possibility of printing each path individually if desired. These path prints can be used to verify the quality of the computed paths and are a preview of what the eventual board wiring will look like.

When the last node has been processed, a message tells the operator that the layout has been completed. This also means that the notion of nodes is no longer required since all links have been calculated and all the information is contained in a table consisting of four numbers per link, namely the coordinates of each link's end points. All further processing is done on the full set of links, irrespective of the node to which they formerly belonged.

5.3 Link Direction

It has been stated above that the link direction was not to be considered in the determination of paths. However, in the actual board wire-wrapping, the placement guide for the wrapping tool is attached to a moving arm, both of which may partially hide the board or the wire to be wrapped. Operators have found it more convenient to wrap a wire from left to right and from top to bottom. The next step consists in introducing the notion of direction and rearranging the link's coordinates in the proper sequence so that the movement pr ced will be as desired.

5.4 Wire Determination

The wire-wrap machine is equipped with 40 bins numbered 0 to 47 (in octal) to store the precut wires of standard lengths used in wiring a board. The program at this point goes through the following steps:

1) compute the length of each link in board units (0.1 in) and reorder all links so that they will be presented starting with the shorter ones;

2) convert the lengths to inches, adding the appropriate length required for the actual wrapping;

3) sort these lengths by bins, each length being assigned to the bin having the next longest standard length;

4) print a table of the number and of the lengths of wires in each bin, and of the bins' numbers.

The program pauses before printing the table and the operator is asked to intervene before the printing takes place. This allows the resetting of the CRT screen or of the paper in the typewriter to obtain a clean copy of the table. The information in this table can then be used to precut all the wires and distribute them into the proper bins in preparation for a wrapping session.

5.5 Wire-Wrap Codes

This is the final step in preparing the data required for wiring a board. The wire-wrap machine requires a number of special codes to:

a) set up the tool guide at the origin at the beginning of a session,
b) identify the beginning and the end of a wire to be wrapped,

c) sequentially identify each wire and the wire bin to be used,
d) sound a buzzer every time a new bin will be used, and
e) return to the origin at every 20 wires so that drift in the positioning of the tool guide can be corrected if necessary.

Furthermore, the machine works in 0.005-in steps and requires that the positioning information be supplied in the form of a displacement relative to the last position taken rather than in absolute coordinates. Hence the coordinates of the links have to be converted accordingly and all the appropriate codes inserted in the proper sequence. The resulting table is then put on file, ready for use in wiring the board.

The final action is to signal to the operator that the program is finished and advise him on how to obtain drawings of each component's connections, as explained in the following section.

6.0 COMPONENT DRAWING PROGRAMS

Once the link optimization programs (Sect. 5.0) have been executed, the user must have the means of determining the exact wiring pattern selected by the computer for linking each connection node. This must be provided in a form that can aid future reference.

For this purpose, a set of component drawing programs was written. These programs sketch a chip in a standardized presentation (Fig. 7) with all the information available from the main chip bank included. The location of each pin, in board coordinates, is displayed together with a list and location of all other pins (maximum three) to which the pin is hardwired. A complete information package on a wire-wrap job is thus formed by gathering hard copies of such chip drawings.

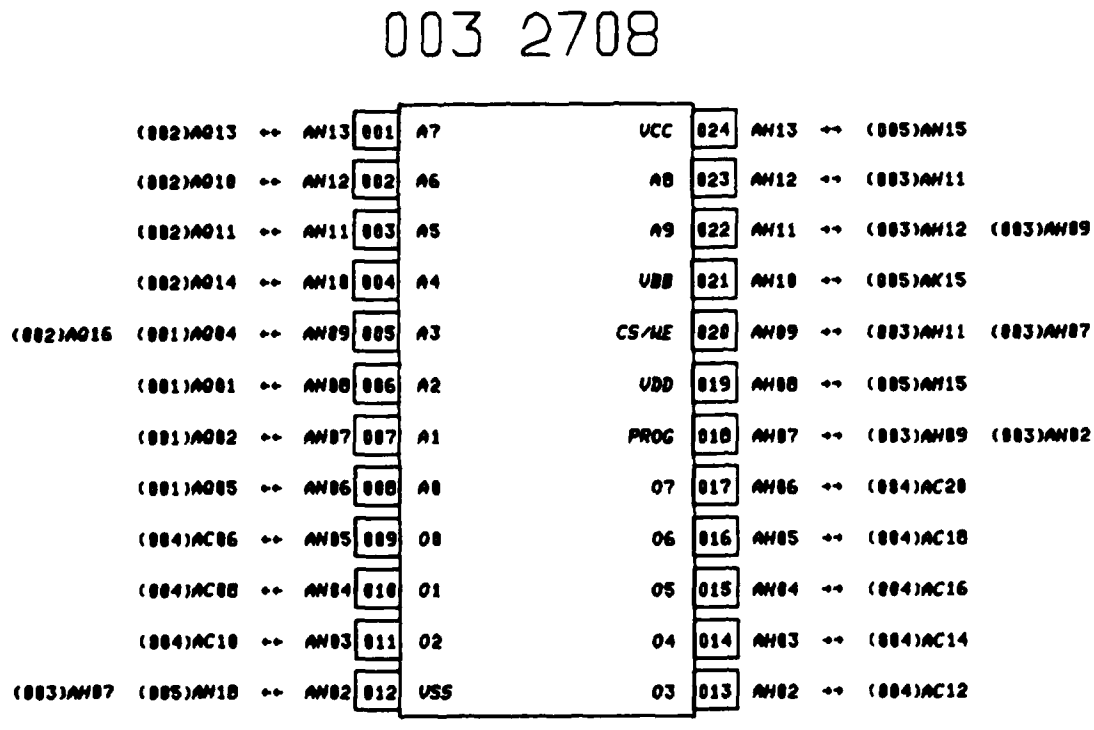


FIGURE 7 - Standardized presentation of a chip obtained with the drawing programs

```
CHIP 'CON50'
TERMINAL? (2-4015; 3-4662)
0:
      2
```

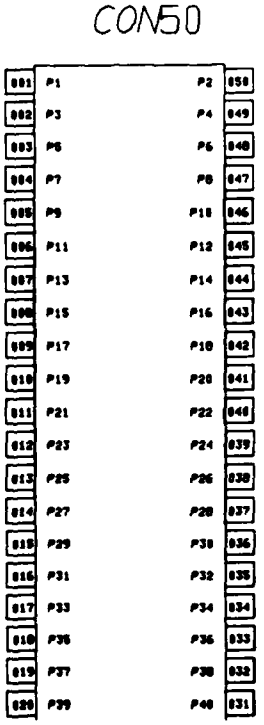


FIGURE 8a - Execution of a component drawing in several passes - first pass

```
CHIP 'CON50 22'
TERMINAL? (2-4015; 3-4662)
0:
      2
```

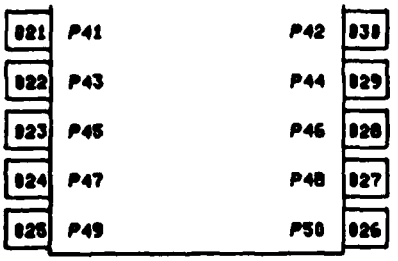


FIGURE 8b - Execution of a component drawing in several passes - second pass

```
CHIP '8085'
TERMINAL? (2-4815; 3-4662)
0:
      2
```

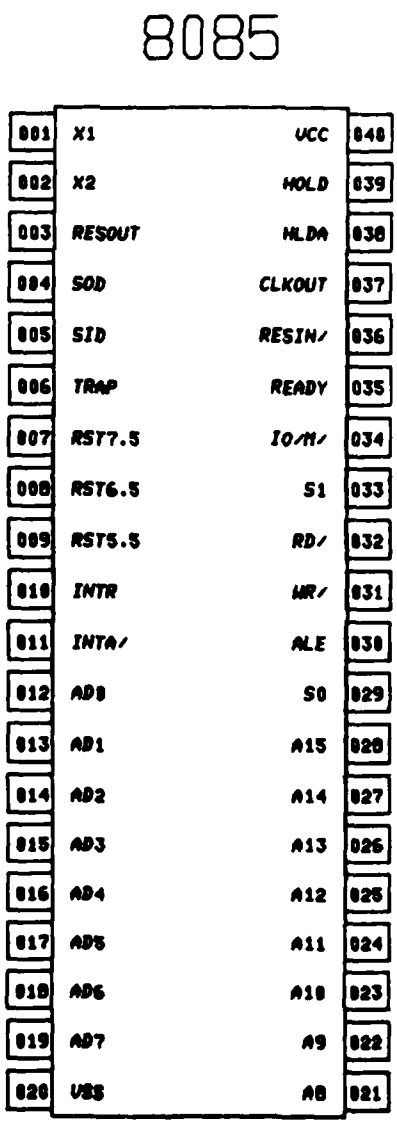


FIGURE 9 - Drawing of a chip in the bank

The program SETDRAWING must be executed once after the link is completed. It establishes a permanent record of the circuit-wide interconnection patterns for fast access by the drawing programs. The chip with identification number X may be drawn by submitting to the computer the command DRAW X. Components with up to 40 pins can be fully drawn this way in one execution of the program DRAW. Large components require drawing translation and further execution of the program DRAW. The applicable component format is DRAW X Y, where X is still the chip identification number and Y is a shift parameter, which translates the sketch upward by Y interpin units. A component drawing executed in several passes is shown in Figs. 8a and 8b.

Using equivalent software, it is possible to obtain a graphic display of any chip in the main chip bank, even if it is not in use in the current job, by executing CHIP 'name'. The component named is then drawn (Fig. 9) with its pertinent information extracted from the main chip bank.

7.0 OVERALL IMPLEMENTATION AT DREV

The computer-aided circuit definition described in this report was implemented at DREV using in-house APL facilities. The APL functions that have been generated to perform the work are distributed over three APL workspaces (WS) which are loaded in sequence by the user. Two APL files are used. A first one, the main data bank, is common to all users and contains the definition of the boards on which the circuits are to be wired as well as that of the chips entered in all previous circuit definitions. A second one, the temporary data bank, is particular to each user and is used to pass the information between the different WS's. This eliminates the need for transferring variables from one WS to another.

The name of the three WS's are WRAP1, WRAP2 and WRAP3. WRAP1 contains the interpreter programs and permits the drawing of a schematic of all the chips and terminal pins located on the board to be wired (BOARD). Additional functions allow the printing of the directory of all the chips in the main bank (DIRECTORY) to modify the data on any chip in the bank (MODIFY 'name') to draw any chip in the bank (CHIP 'name'). WRAP2 contains all the functions which perform the circuit layout calculations, list the lines of the definition program which contain a common pin, draw the connection layout for all the nodes if required, and print a table of the quantity of wires required of each length. WRAP3 produces a list of all the pins that have not been connected (FREEPIN) and contains the necessary functions to get the drawing of any of the chips on the board (DRAW N) with the complete information related to the origin and destination of each wire.

7.1 The Main Data Bank

The main data bank is an APL file containing the information about the connectors and power supply terminals of type 2 and type 3 boards as well as the definition of all the chips that were used in previous circuit definitions. The size of the file is incremented by one element each time a new chip is entered. The organization of the file is as follows:

a) the first element is a directory of all the chips in the bank;

b) the second element is a vector containing the element number where a particular chip is defined in the bank;

c) the third element contains the definition of all the connectors and power supply terminals of type 2 and type 3 boards;

d) each of the following elements contains the definition of one chip, which consists of the name of the chip, its geometry (24 PIN, 28 PIN, 40 PIN, etc) and the name of all its pins.

7.2 The Temporary Data Bank

The temporary data bank is a user file generated at the beginning of a session and is used by the three WS's to transfer the data from one WS to another and to store data blocks to prevent the WS from becoming full while executing functions manipulating large matrices. The organization of the file is given in Table III.

8.0 A WIRING SESSION

A complete wiring session for the circuit of Fig. 1 mounted on a matrix board is given in Appendix D. The different steps to go from the electrical schematic diagram to the complete wiring are as follows:

a) the chips are disposed on a drawing of the receiving board;

b) the location and definition programs are written according to the rules given in Sect. 3;

c) workspace WRAP1 is loaded and funtion START is executed. The chips that were not already in the main bank are entered following the program directives.

d) with WRAP1, a directory of all chips currently in the bank is produced by the function DIRECTORY and the stored data about any chip can be modified by using function MODIFY 'name' where 'name' is the name of the chip;

TABLE III

Organization of the temporary data bank

Element Number	Content
1	BASEORI
2	BASEPIN1
3	BASEMAT
4	BASESPA
5	LISTNUM
6	BASEFIND
7	RACINE
8	Definition of the chips used in the circuit being wired.
9	BASEVEC
10	CODEGEO
11	STRING1
12	STRING2
13	STRING3
14	STRING4
15	STRING5
16	STRING6
17	PINGEO
18	SECT
21	R
22	TYPE
101 to 109	XYM
150 to 159	Matrix of coordination in order of wire lengths.
301 to 309	Matrix for the wire-wrap machine.

e) after the completion of the interpreter programs, a computer drawing of the disposition of the chips on the board with the emplacement of the terminals may be obtained by executing the function BOARD;

f) the circuit layout calculation is then performed by loading workspace WRAP2 and executing the function START. A first output will be a list of the lines of the definition program that have common points. Then, the circuit layout for each node may be obtained separately or superimposed. A table of the quantity of wires of each standard length is printed;

g) to get a list of all the unused pins of the circuit, and drawings of the chips, the workspace WRAP3 is loaded and the function SETDRAWING is executed. The list of unused pins is obtained by the function FREEPIN. A chip is drawn with the function DRAW N where N is the number assigned to the chip in the location program.

9.0 CONCLUSION

This report has described a new 'language' for rapidly encoding a complex circuit in a computer-recognizable format, the interpreter programs which transform this coded description into a raw interconnection matrix, the circuit calculation programs which determine the final circuit layout and produce the wiring table, and the utility programs which produce the required reference sketches and tables. Together with the addition of the microprocessor control to the wiring machine itself (Ref. 2), these represent the introduction of computer assistance at every step required to go from a circuit diagram to a finished board.

A first advantage is the reduction in the time spent by qualified technicians to go through these steps, some of which are of a tedious and repetitive nature. The main advantage, however, stems from the 'relentless' attention which the computing machine can devote to such tasks, thus eliminating the major source of errors in the wiring of complex circuit boards. When full use is made of the error diagnostics produced during the processing phase by the built-in error detection schemes, it becomes possible to produce large boards which are practically error free. It is our experience that the number of wiring errors grows very rapidly as boards become more complex and the identification and correction of errors after the board is wired is a frustating and time-consuming task. The economy realized in this respect is ample justification for the effort spent in the design and implementation of the system.

10.0 REFERENCES

1. Grenier, C., "Câblage enroulé semi-automatique Wire-Wrap®", DREV M-2342/75, January 1975, UNCLASSIFIED

2. Bernier, N., "Automatisation de la machine à câblage enroulé WWM-600", DREV R-4187/81, February 1981, UNCLASSIFIED

APPENDIX A

Listing of Computer Programs

```
      )WSID
WRAP1
      )FNS
BASE     BOARD    CATEGORIE          CHIP     CHIP1    CO
CONV     CUMUL2   CUMUL4   CUMUL5    CUMUL6   DECODE   DEM
DIRECTORY          FAPPEND FCNUM     FCREATE FDROP     FERASE
FHOLD    FIND     FLIM     FNAMES    FNUMS    FP       FRDCIA
FRDCIDT  FREAD    FREPLACE            FSDROP  FSTIE    FTIE
FUNTIE   INFILE   INTERROGE           L        LIEDERN  LIEVIRG  LK
LOCATE   MAP      MIP      MODIFY    PAD      PIT      POSITION
READ     READ1    REGU     REGULARISE          SBCBOARD
SBC905   SC         SCANBE   SCANBEBOARD       STAGE    STAGE00
STAGE01  STAGE02  STAGE03  STAGE04  START    START1   SUBSTITUE
W          CA       CAL
      )VARS
CODEGEO  PINGEO   TEX      FA        IA       KA       NA
      ∇BASE[⎕]∇
      ∇BASE
[1]   H←,(((4×SECT),1)⍴B←⍉((SECT,4)⍴27 30 38 38)),((SECT×4
      ),1)⍴A←((4,SECT)⍴27×¯1+⍳SECT)+⍉(SECT,4)⍴18 18 5 17
      ∇
      ∇BOARD[⎕]∇
      ∇BOARD;S;Y1;Y2;A
[1]   →1×⍳~(A←⎕,0⍴⎕←'TERMINAL? (2=4015; 3=4662)')∈2 3
[2]   A←A⌷'GDV'
[3]   MIP
[4]   PIT
      ∇
      ∇CATEGORIE[⎕]∇
      ∇N←CATEGORIE S
[1]   N←1↑(('⍲'∈S),('←'∈S),(∧/'∪:'∈S),(∧/'∪,'∈S),(':'∈S),(
      '='∈S),','∈S)/7 1 3 5 2 6 4
      ∇
      ∇CHIP[⎕]∇
      ∇CHIP ABC;S;BANQUE;CHIP;NOM;TEX;TEXT;VERT;X;X1;X3;Y;
      YF;YH;OFFSET
[1]   CHIP1 ABC
      ∇
      ∇CHIP1[⎕]∇
      ∇CHIP1 CHOP;AD;COM;DIRECTNUM;DIRECTORY;I;N;NO;VEC;HO
      RI;LARG
[1]   →1×⍳~(S←⎕,0⍴⎕←'TERMINAL? (2=4015; 3=4662)')∈2 3
[2]   S←S⌷'GDV'
[3]   OFFSET←-1↑∈(CHOP⍳' ')↓CHOP
[4]   CHOP←(CHOP⍳' ')↓CHOP
[5]   BANQUE←6 1⍴'      '
[6]   FUNTIE FNUMS
[7]   'BAN.65888'FSTIE 1
```

```
[8]   'BON'FTIE 2
[9]   DIRECTORY←FREAD 1 1
[10]  DIRECTNUM←FREAD 1 2
[11]  CHIP←1
[12] DEBUT:S←1⎕'CTY'
[13]  TEX←'0123456789'
[14]  NOM←⍞1 6ρCHOP,'       '
[15]  →TRABAN×ι1=+/I←6=+/[2]DIRECTORY=(ρDIRECTORY)ρNOM
[16]  'CHIP ',(6ρ⍞NOM),'IS NOT IN THE BANK...'
[17]  →0
[18] TRABAN:I←I/ι(ρDIRECTORY)[1]
[19]  BANQUE←BANQUE,⍞(FREAD 1,DIRECTNUM[I])
[20]  VEC←6=+/[1]BANQUE=⍞(⍬ρBANQUE)ρNOM
[21] SUITE:Y←1
[22]  X3←X←0
[23]  VERT←+/.5×¯3+1↓ρBANQUE
[24]  OFFSET←OFFSET+VERT-20
[25]  HORI←5
[26]  S←¯10 15 0 22⎕'SCL'
[27]  ⍬←2 5ρ(0,0,HORI,HORI,0,0,VERT,VERT,0,0)-(5ρ0),5ρOFFS
      ET
[28]  LARG←.75
[29] SCAN2:⍬←2 5ρX1,X,X,X1,(X1←X-LARG),YF,YF,YH,(YH←YF+LAR
      G),YF←(Y+¯.5-LARG÷2)-OFFSET
[30]  →DROITE×ιX3=1
[31]  S←((¯10+X←.5×LARG),((-OFFSET)+¯.27+(1+VERT-Y)-.5×LAR
      G))⎕'CUR'
[32]  TEXT←(51ρ' '),TEX[1+10 10 10τY],'  ',6ρ⍞BANQUE[;Y+3]
[33] OKALL:⍬←TEXT
[34]  →SCAN2×ι(Y←Y+1)≤VERT
[35]  Y←1
[36]  X←X+HORI+LARG
[37]  →SCAN2×ι(X3←X3+1)≤1
[38]  →NAME
[39] DROITE:TEXT←6ρ⍞BANQUE[;(2×VERT)+4-Y]
[40]  TEXT←(-+/' '=TEXT)⌽TEXT
[41]  TEXT←TEXT,'  ',(,TEX[1+10 10 10τ⌊((2×VERT)-Y+¯1)])
[42]  S←((¯1.8+X←.5×LARG),((-OFFSET)+¯.27+(1+VERT-Y)-.5×LA
      RG))⎕'CUR'
[43]  →OKALL
[44] NAME:S←(¯.25,(-OFFSET)+VERT+1)⎕'CUR'
[45]  S←0⎕'CTY'
[46]  S←.6 1.2⎕'CSZ'
[47]  ⍬←'   ',CHOP
[48]  FUNTIE FNUMS
      ∇
      ∇CO[⎕]∇
      ∇R←CO 4
```

```
[1]    R←-/C←(0 1↓C),2 1↑A
        ∇
        ∇CONV[□]∇
        ∇M←W CONV P
[1]    M←(2,ρP)ρ1↓(W|P),L(P←P-1)↑W
        ∇
        ∇CUMUL2[□]∇
        ∇CUMUL2;CATEGO;PROVENANCE;TEXT;CLASSE;RES;U;I;PHRASE
        ;NUM;NOM;XY;STEP;STEPP;S
[1]    S←FREAD 2 12
[2]    →(1=ρS)/0
[3]    CATEGO←2
[4]    ⍝
[5]    ⍝--------ANALYSE DE LA STRING
[6]    ⍝
[7]   J3:S←'α'FP S
[8]    PROVENANCE←∈RES
[9]    S←'ω'FP S
[10]   TEXT←RES
[11]   CLASSE←':'FP TEXT
[12]   TEXT←RES
[13]   TEXT←TEXT,'.'
[14]   CLASSE←6↑CLASSE
[15]   U←CLASSE∧.=CLASSED
[16]   I←⍳¯1↑ρCLASSED
[17]   I←U/I
[18]   SETL←SETD[I]
[19]   →(0=ρI)/J4
[20]   →J2
[21]  J4:CLASSED←CLASSED,CLASSE
[22]   SETL←SET
[23]   SET←SET+1
[24]   SETD←SETD,SETL
[25]   ⍝
[26]   ⍝--------ANALYSE DU TEXTE
[27]   ⍝
[28]  J2:TEXT←','FP TEXT
[29]   PHRASE←RES
[30]   PHRASE←' 'FP PHRASE
[31]   NUM←∈RES
[32]   ⍝
[33]   ⍝--------ANALYSE DE LA PHRASE
[34]   ⍝
[35]  J1:PHRASE←' 'FP PHRASE
[36]   NOM←RES
[37]   XY←NUM MAP 6↑NOM
[38]   XY←XY,CATEGO,(-SETL),PROVENANCE
[39]   INFILE
```

```
[40]  →(0≠ρPHRASE)/J1
[41]  →(0≠ρTEXT)/J2
[42]  →(2≤ρS)/J3
[43]  →0
       ∇
       ∇CUMUL4[□]∇
       ∇CUMUL4;CATEGO;PROVENANCE;TEXT;CLASSE;RES;U;I;SETN;P
       HRASE;NUM;NOM;XY;STEP;STEPP;S
[1]    S←FREAD 2 14
[2]    →(1=ρS)/0
[3]    STEPP←0
[4]    CATEGO←4
[5]    ⍝
[6]    ⍝--------ANALYSE DE LA STRING
[7]    ⍝
[8]   J3:S←'α'FP S
[9]    PROVENANCE←∈RES
[10]   S←'ω'FP S
[11]   TEXT←RES
[12]   SETN←SET
[13]   ⍝
[14]   ⍝--------ANALYSE DU TEXTE
[15]   ⍝
[16]  J2:STEP←0
[17]   TEXT←',' FP TEXT
[18]   PHRASE←RES
[19]   PHRASE←' 'FP PHRASE
[20]   NUM←∈RES
[21]   ⍝
[22]   ⍝--------ANALYSE DE LA PHRASE
[23]   ⍝
[24]  J1:PHRASE←' 'FP PHRASE
[25]   NOM←RES
[26]   XY←NUM MAP 6↑NOM
[27]   →(0=1↑XY)/J5
[28]   XY←XY,CATEGO,SETN,PROVENANCE
[29]   INFILE
[30]  J5:STEP←STEP+1
[31]   SETN←SETN+1
[32]   →(0≠ρPHRASE)/J1
[33]   SETN←SETN-STEP
[34]   →(~(STEP=STEPP)∨STEPP=0)/ERREUR
[35]  ERREUR1:STEPP←STEP
[36]   →(0≠ρTEXT)/J2
[37]   SETN←SETN+STEPP
[38]   SET←SETN
[39]   STEPP←0
[40]   →(2≤ρS)/J3
```

```
[41]  →0
[42] ERREUR:'STATEMENT ';PROVENANCE;' IN ERROR'
[43]  →ERREUR1
      ∇
      ∇CUMUL5[□]∇
      ∇CUMUL5;CATEGO;PROVENANCE;TEXT;CLASSE;RES;U;I;SETN;P
      HRASE;NUM;NOM;XY;STEP;STEPP;S
[1]   S←FREAD 2 15
[2]   →(1=ρS)/0
[3]   STEPP←0
[4]   CATEGO←5
[5]   ⍝
[6]   ⍝-------ANALYSE DE LA STRING
[7]   ⍝
[8]  J3:S←'α'PP S
[9]   PROVENANCE←∈RES
[10]  S←'ω'PP S
[11]  TEXT←RES
[12]  TEXT←1↓TEXT
[13]  SETN←SET
[14]  ⍝
[15]  ⍝-------ANALYSE DU TEXTE
[16]  ⍝
[17] J2:STEP←0
[18]  TEXT←',PP TEXT
[19]  PHRASE←RES
[20]  PHRASE←' 'PP PHRASE
[21]  NUM←∈RES
[22]  ⍝
[23]  ⍝-------ANALYSE DE LA PHRASE
[24]  ⍝
[25] J1:PHRASE←' 'PP PHRASE
[26]  NOM←RES
[27]  XY←NUM MAP 6↑NOM
[28]  →(0=1↑XY)/J5
[29]  XY←XY,CATEGO,SETN,PROVENANCE
[30]  INFILE
[31] J5:STEP←STEP+1
[32]  SETN←SETN+1
[33]  →(0≠ρPHRASE)/J1
[34]  SETN←SETN-STEP
[35]  →(~(STEP=STEPP)∨STEPP=0)/ERREUR
[36] ERREUR1:STEPP←STEP
[37]  →(0≠ρTEXT)/J2
[38]  SETN←SETN+STEPP
[39]  SET←SETN
[40]  STEPP←0
[41]  →(2≤ρS)/J3
```

```
[42]  →0
[43] ERREUR:'STATEMENT ';PROVENANCE;' IN ERROR'
[44]  →ERREUR1
      ∇
      ∇CUMUL6[□]∇
      ∇CUMUL6;CATEGO;PROVENANCE;TEXT;CLASSE;RES;U;I;PHRASE
      ;NUM;NOM;XY;STEP;STEPP;S
[1]   S←FREAD 2 16
[2]   →(1=ρS)/0
[3]   CATEGO←6
[4]   ⍝
[5]   ⍝-------ANALYSE DE LA STRING
[6]   ⍝
[7]  J3:S←'α'FP S
[8]   PROVENANCE←∈RES
[9]   S←'ω'FP S
[10]  TEXT←RES
[11]  TEXT←'='FP TEXT
[12]  CLASSE←RES
[13]  TEXT←TEXT,','
[14]  CLASSE←6↑CLASSE
[15]  U←CLASSE∧.=CLASSED
[16]  I←⍳¯1↑ρCLASSED
[17]  I←U/I
[18]  →(0=ρI)/J4
[19]  SETL←SETD[I]
[20]  →J2
[21] J4:CLASSED←CLASSED,CLASSE
[22]  SETL←SET
[23]  SET←SET+1
[24]  SETD←SETD,SETL
[25]  ⍝
[26]  ⍝-------ANALYSE DU TEXTE
[27]  ⍝
[28] J2:TEXT←','FP TEXT
[29]  PHRASE←RES
[30]  PHRASE←' 'FP PHRASE
[31]  NUM←∈RES
[32]  ⍝
[33]  ⍝-------ANALYSE DE LA PHRASE
[34]  ⍝
[35] J1:PHRASE←' 'FP PHRASE
[36]  NOM←RES
[37]  XY←NUM MAP 6↑NOM
[38]  XY←XY,CATEGO,SETL,PROVENANCE
[39]  INFILE
[40]  →(0≠ρPHRASE)/J1
[41]  →(0≠ρTEXT)/J2
```

```
[42]  →(2≤ρS)/J3
[43]  →0
       ∇
       ∇DECODE[□]∇
       ∇DECODE K;VECGEO
[1]    VECGEO←CODEGEO LOCATE K
[2]    MAT←⍉VECGEO/[2]PINGEO
       ∇
       ∇DEM[□]∇
       ∇B←DEM A
[1]    ⍞←A
[2]    B←(ρA)↑⍞
       ∇
       ∇DIRECTORY[□]∇
       ∇DIRECTORY
[1]    FUNTIE FNUMS
[2]    'BAN.65888'FSTIE 1
[3]    'LIST OF CHIPS IN THE BANK'
[4]    ' '
[5]    FREAD 1 1
[6]    FUNTIE FNUMS
       ∇
       ∇FIND[□]∇
       ∇Z←Y FIND X
[1]    Z←Y LOCATE X
[2]    Z←(¯1⊖Z)/⍳(ρY)[2]
       ∇
       ∇PP[□]∇
       ∇SO←SEP PP SI;N
[1]    ⍝
[2]    ⍝ PLACE DANS RES LA PARTIE DE LA STRING SI QUI PRECE
       DE
[3]    ⍝ LE SEPARATEUR SEP ET PLACE DANS SO LE RESTE DE LA
       STRING
[4]    ⍝
[5]    N←SI⍳SEP
[6]    RES←(¯1+N)↑SI
[7]    SO←N↓SI
       ∇
       ∇INFILE[□]∇
       ∇INFILE
[1]    XYM←XYM,XY
[2]    →((((ρXYM)[2])>400))/AUGPE
[3]    →0
[4]   AUGPE:XYM PREPLACE(2,PE←PE+1)
[5]    XYM←5 0ρ⍳0
       ∇
       ∇INTERROGE[□]∇
       ∇INTERROGE;COM;COM1;PATTE
```

```
[1]   COM←NOM
[2]   ' GEOMETRY: DUAL IN LINE ←←←←←  2PIN'
[3]   '                                6PIN'
[4]   '                                8PIN'
[5]   '                               14PIN'
[6]   '                               16PIN'
[7]   '                               18PIN'
[8]   '                               20PIN'
[9]   '                               22PIN'
[10]  '                               24PIN'
[11]  '                               28PIN'
[12]  '                               40PIN'
[13]  '               CONNECTOR  ←←←←←C26PIN'
[14]  '                                C10PIN'
[15]  '                                C40PIN'
[16]  '                                C50PIN'
[17]  '               DISPLAY    ←←←←←←←AFF8'
[18]  '  DISPLAY LCD 40PIN X 1.3IN←←←←LCD1'
[19]  '  DISPLAY MAN6660,80 10PIN←←←MAN66X'
[20] GEODEM:COM1←DEM'GEOMETRY?  '
[21]  ''
[22]  COM1←⍞1 6⍴COM1,'        '
[23]  DECODE COM1
[24]  →GEODEM×⍳(×/⍴MAT)=0
[25]  COM←COM,COM1
[26]  PATTE←1
[27] PATDEM:COM1←DEM'IDENTIFICATION OF PIN',TRX[PATTE;],'?
       '
[28]  COM←COM,⍞1 6⍴COM1,'        '
[29]  →PATDEM×⍳(PATTE←PATTE+1)≤2×⍴MAT[1;1]
[30]  BANQUE←BANQUE,COM
[31]  (⍞COM)FAPPEND 1
[32]  (DIRECNUM←DIRECNUM,(¯1+FLIM 1)[2])FREPLACE 1 2
[33]  (DIRECTORY←DIRECTORY,[1]⍞NOM)FREPLACE 1 1
       ∇
       ∇L[⎕]∇
       ∇L A;G;NOM;NUM;NUMERO;OR;ORI;I;AD;NO;COM
[1]   ORDRE←A
[2]   CAR←0
[3]   G←⌶101,ERRNUM
[4]   NUMERO←LIEVIRG
[5]   →0×⍳(⍴NUMERO,' ')=1
[6]   NUM←|∈NUMERO
[7]   →ERRNUM×⍳0≠+/LISTNUM∈NUM
[8]   LISTNUM←LISTNUM,NUM
[9]   G←⌶101 0
[10]  NOM←LIEVIRG
[11]  NOM←⍞1 6⍴NOM,'        '
```

```
[12] TESTNOM:→NOMOK×⍳1=+/VEC←6=+/[1]BANQUE=⍉(⍬⍴BANQUE)⍴NOM
[13]  →TRABAN×⍳1=+/I←6=+/[2]DIRECTORY=(⍴DIRECTORY)⍴NOM
[14]  'CHIP ',(6⍴⍉NOM),'IS NOT IN THE BANK...'
[15]  INTERROGE
[16]  →TESTNOM
[17] TRABAN:I←I/⍳(⍴DIRECTORY)[1]
[18]  BANQUE←BANQUE,⍉(FREAD 1,DIRECNUM[I])
[19]  →TESTNOM
[20] NOMOK:VEC←¯1⍬VEC
[21]  DECODE⍉1 6⍴⍉VEC/BANQUE
[22]  BASEVEC←BASEVEC,(¯1⍬VEC)/⍳(⍴BANQUE)[2]
[23]  BASEMAT←BASEMAT,2⍴MAT
[24]  POSI←LIEVIRG
[25]  POSITION
[26]  →AVOR×⍳POSIOK=0
[27]  BASEPIN1←BASEPIN1,LI,COLN
[28]  OR←⌊∈ORI←LIEDERN
[29]  →ORIBAD×⍳0=+/'1234'∈ORI
[30]  BASEORI←BASEORI,OR
[31]  BASESPA←BASESPA,1
[32]  →0
[33] ORIBAD:'INCORRECT ORIENTATION AT LINE  ';(⍕27)[2]
[34]  →FIN
[35] ERRNUM:'INCORRECT CHIP IDENTIFICATION NUMBER AT LINE
        ';(⍕27)[2]
[36]  →FIN
[37] AVOR:'INCORRECT LOCATION AT LINE  ';(⍕27)[2]
[38] FIN:G←⍕101 0
[39]  ERREUR←1
      ∇
      ∇LIEDERN[⎕]∇
      ∇MOT←LIEDERN
[1]   MOT←⍳0
[2]  BOUC:CAR←CAR+1
[3]   MOT←MOT,ORDRE[CAR]
[4]   →0×⍳CAR=⍴ORDRE
[5]   →BOUC
      ∇
      ∇LIEVIRG[⎕]∇
      ∇MOT←LIEVIRG
[1]   MOT←⍳0
[2]   →0×⍳ORDRE[1]='⍝'
[3]  BOUC:CAR←CAR+1
[4]   →0×⍳ORDRE[CAR]=','
[5]   MOT←MOT,ORDRE[CAR]
[6]   →BOUC
      ∇
      ∇LK[⎕]∇
      ∇E←LK P;K;R;G
```

```
[1]    →FI×⍳0=⍴P←K↓P,E←0⍴G←(⍴R)⍴⍴R←((R⍳R)=⍳⍴R)/R←¯1↓(K←P⍳0
       )↓P
[2]   AG:→NE×⍳~∨/K←R∈E←((E⍳E)=⍳⍴E)/E,0⍴P←(1+⍴E←¯1↓(P⍳0)↓P)↓
       P
[3]    →(E←⍳0),(0<⍴P,0⍴G←((~K)/G),(⍴E)⍴G[(G∈K/G)/⍳⍴G]←1+⌈/G
       ,0⍴R←((~K)/R),E)⌽FI,AG
[4]   NE:→AG×E←⍳0<⍴P,0⍴R←R,E,0⍴G←G,(⍴E)⍴1+⌈/G
[5]   FI:→FI×⍳0<⍴G←K/G,0⍴R←K/R,0⍴E←E,((~K←G≠1+G)/R),0
         ∇
         ∇LOCATE[⎕]∇
         ∇Z←Y LOCATE X
[1]      Z←6=+/[1]Y=⍉(⌽⍴Y)⍴X
         ∇
         ∇MAP[⎕]∇
         ∇XY←NUM MAP NOM;I;MAT;Z;N;X;Y;DIST;NOMA;M;NOMB;IA;IB
[1]      M←2 2 4⍴1 0 ¯1 0 0 1 0 ¯1 0 ¯1 0 1 1 0 ¯1 0
[2]      I←LISTNUM⍳NUM
[3]      →(I>⍴LISTNUM)/J4
[4]      →(~∧/'-      '=NOM)/J1
[5]      XY←0 0
[6]      →0
[7]     J1:DECODE BANQUE[;¯1+BASEVEC[I]]
[8]      Z←MAT
[9]      Z←,Z
[10]     N←1+Z
[11]     X←⍳N
[12]     X←1+BASESPA[I]×X-1
[13]     X←X,⌽X
[14]     Y←(N⍴1),N⍴1+1+Z
[15]     XY←Y,[.5]X
[16]     XY←1+M[;;BASEORI[I]]+.×¯1+XY
[17]     DIST←BASEPIN1[(2×I-1)+1 2]
[18]     XY[1;]←XY[1;]+DIST[2]
[19]     XY[2;]←XY[2;]+DIST[1]
[20]     XY[1;]←1000-XY[1;]
[21]     XY[2;]←1000-XY[2;]
[22]     ⍝CUMU
[23]     NOMA←BANQUE[;¯1+BASEVEC[I]+⍳2×N]
[24]     IA←(NOM∧.=NOMA)/⍳2×N
[25]     →(0=⍴IA)/J2
[26]     XY←,XY[;IA]
[27]     →0
[28]    J2:NOMB←⍉'I2,X4'⍕⍳2×N
[29]     NOMB←(' '=NOMB[1;])⌽NOMB
[30]     IB←(NOM∧.=NOMB)/⍳2×N
[31]     →(0=⍴IB)/J3
[32]     XY←,XY[;IB]
[33]     →0
```

```
[34] J3:'ERROR AT LINE    ';PROVENANCE
[35]  'CHIP NUMBER ';NUM
[36]  ' '
[37]  ⍞BANQUE[;¯3+BASEVEC[I]+⍳N+2]
[38]  ' '
[39]  'PIN   ',NOM
[40]  XY←0 0
[41]  →0
[42] J4:'ERROR AT LINE  ';PROVENANCE
[43]  'CHIP NUMBER  ';NUM;'  IS NOT DEFINED'
[44]  XY←0 0
      ∇
      ∇MIP[⎕]∇
      ∇MIP;A;BANQUE;BASEVEC;BASEPIN1;BASEORI;LISTNUM;CODEG
      EO;PINGEO;BASESPA;NUM;M;I;S;Z;MAT;N;X;Y;Y;DIST;FE;XY
      M;XYH;XY
[1]   A←'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
[2]   FUNTIE FNUMS
[3]   'BON'FTIE 2
[4]   BANQUE←FREAD 2 8
[5]   BASEVEC←FREAD 2 9
[6]   BASEPIN1←FREAD 2 2
[7]   BASEORI←FREAD 2 1
[8]   A←(⍴LISTNUM←FREAD 2 5)⍴A
[9]   CODEGEO←FREAD 2 10
[10]  PINGEO←FREAD 2 17
[11]  BASESPA←FREAD 2 4
[12]  NUM←1
[13]  M←2 2 4⍴1 0 ¯1 0 0 1 0 ¯1 0 ¯1 0 1 1 0 ¯1 0
[14]  XYH←2 0⍴0
[15]  XYM←2 0⍴0
[16] J1:I←NUM
[17]  S←A[NUM]⍞'SYM'
[18]  DECODE BANQUE[;¯1+BASEVEC[I]]
[19]  Z←MAT
[20]  Z←,Z
[21]  N←1↑Z
[22]  X←⍳N
[23]  X←1+BASESPA[I]×X-1
[24]  X←(1↑X),¯1↑X
[25]  X←X,⌽X
[26]  Y1←(N⍴1)
[27]  Y1←(1↑Y1),¯1↑Y1
[28]  Y2←,N⍴1+1↑Z
[29]  Y2←(1↑Y2),¯1↑Y2
[30]  Y←Y1,Y2
[31]  XY←Y,[.5]X
[32]  XY←1+M[;;BASEORI[I]]+.×¯1+XY
```

```
[33]  DIST←BASEPIN1[(2×I-1)+1 2]
[34]  XY[1;]←XY[1;]+DIST[2]
[35]  XY[2;]←XY[2;]+DIST[1]
[36]  XY[1;]←1000-XY[1;]
[37]  XY[2;]←1000-XY[2;]
[38]  X←XY[1;]
[39]  Y←XY[2;]
[40]  X←1↓¯1↓,X,[1.5]X
[41]  Y←1↓¯1↓,Y,[1.5]Y
[42]  XY←X,[.5]Y
[43]  XYM←XYM,XY
[44]  XYH←XYH,2 1↑XY
[45]  NUM←NUM+1
[46]  →(NUM≤ρLISTNUM)/J1
[47]  S←21⍴'MOD'
[48]  S←'O'⍴'SYM'
[49]  XYM FREPLACE 2 24
[50]  XYH FREPLACE 2 25
       ∇
       ∇MODIFY[□]∇
       ∇MODIFY NOM;I;B;CHIP;DIRECTORY;DIRECNUM;R;N
[1]   FUNTIE FNUMS
[2]   'BAN.65888'FSTIE 1
[3]   DIRECTORY←FREAD 1 1
[4]   DIRECNUM←FREAD 1 2
[5]   NOM←⍴1 6ρNOM,'      '
[6]   →L1×ι1=+/I←6=+/[2]DIRECTORY=(ρDIRECTORY)ρNOM
[7]   'CHIP ',(6ρ⍴NOM),' IS NOT IN THE BANK'
[8]   →0
[9]  L1:I←I/ι(ρDIRECTORY)[1]
[10]  CHIP←FREAD 1,DIRECNUM[I]
[11]  R←(ρCHIP)[1]
[12]  N←1
[13] L2:B←DEM CHIP[N;],' CHANGED FOR: '
[14]  →(0=ρB)/REC
[15]  CHIP[N;]←6ρB,'      '
[16] REC:→((N←N+1)≤R)/L2
[17]  CHIP FREPLACE 1,DIRECNUM[I]
[18]  DIRECTORY[I;]←CHIP[1;]
[19]  DIRECTORY FREPLACE 1 1
[20]  FUNTIE FNUMS
       ∇
       ∇PAD[□]∇
       ∇SO←PAD SI;UP;UA;UC
[1]   ⍝
[2]   ⍝ PLACE DES BLANCS AUTOUR DES SYMBOLES-TYPE
[3]   ⍝
[4]   UP←SI∈'←:U=,⍺⍵'
```

```
[5]    UA←(1+2×ρUP)ρ0 1
[6]    UP←UA\UP
[7]    SI←UA\SI
[8]    UC←∨/¯1 0 1∘.UP,UA,[1.5]UP
[9]    SI←UC/SI
[10]   SO←SI
        ∇
        ∇PIT[□]∇
        ∇PIT;D;XYM;XYH
[1]    FUNTIE 2
[2]    'BON'FSTIE 2
[3]    XYM←FREAD 2 24
[4]    XYH←FREAD 2 25
[5]    S←1⍎'CTY'
[6]    D←(D×⌊/36 24⊤D←|-/D)⍎'FRS',0ρ(,D←⍉2 2ρ(⌊/XYM),⌈/XYM)
        ⍎'SCL'
[7]    S←31⍎'MOD'
[8]    □←XYM
[9]    S←21⍎'MOD'
[10]   □←XYH
        ∇
        ∇POSITION[□]∇
        ∇POSITION;COL;G;DESCA;DESCB;DESCC;DESCD;DESCE
[1]    POSIOK←0
[2]    →AVORTE×⍳(ρPOSI)>4
[3]    COL←POSI[1 2]
[4]    →AVORTE×⍳((ρCOL,'1')≠3)
[5]    PAT1←'ABCDEFGHJKLMNPRSTUWX'
[6]    DESCA←6,SECT,20
[7]    DESCB←24 9,4-COL[1]='X'
[8]    →AVORTE×⍳(~COL[1]∈PAT1←DESCA[TYPE]↑PAT1)∨~COL[2]∈PAT
        2←DESCB[TYPE]↑'ABCDEFGHJIKLMNOPQRSTUVWX'
[9]    →AVORTE×⍳(COL[1]='A')∧(COL[2]='A')∧TYPE=3
[10]   G←⌶101,ERROR
[11]   DESCE←0 34 30
[12]   DESCC←84 50 48
[13]   DESCD←24 27 6
[14]   LI←(LI1←∈POSI[3 4])+DESCE[TYPE]
[15] LIGNEOK:G←⌶101 0
[16]   →AVORTE×⍳(LI<1)∨LI1>DESCC[TYPE]
[17]   RO←ρPAT1
[18] ICI:COLN←DESCD[TYPE]×(PAT1∈ROρCOL[1])/(⍳RO)-1
[19]   →ICI+2×TYPE
[20]   COLN←COLN+(('ABCDEFGHIJKLMNOPQRSTUVWX'∈24ρCOL[2])/⍳2
        4)
[21]   →JOB2
[22]   COLN←(COLN-2)+3×('ABCDEFGHJ'∈9ρCOL[2])/⍳9
[23]   →JOB2
```

```
[24]  COLN←COLN+('ABCD'∈4ρCOL[2])/ι4
[25] JOB2:POSIOK←1
[26] AVORTE:→0
[27] ERROR:G←⊥101,ERROR2
[28]  LI←(LI1←∈POSI[3])+DESCE[TYPE]
[29]  →LIGNEOK
[30] ERROR2:→0
       ∇
       ∇READ[□]∇
       ∇R←READ A
[1]    R←5 0ρι0
[2]    ∈(∧/A=0)/'A←ι¯1+1↓FLIM 2'
[3]    A←(A<1↓FLIM 2)/A
[4]   ON:→0×ι0=ρA
[5]    R←R,[2]FREAD 2,1ρA
[6]    →ON,ρA←1↓A
       ∇
       ∇READ1[□]∇
       ∇R←READ1 A
[1]    R←2 0ρι0
[2]    ∈(∧/A=0)/'A←ι¯1+1↓FLIM 2'
[3]    A←(A<1↓FLIM 2)/A
[4]   ON:→0×ι0=ρA
[5]    R←R,[2]FREAD 2,1ρA
[6]    →ON,ρA←1↓A
       ∇
       ∇REGU[□]∇
       ∇SO←REGU SI;UP;UB;UC
[1]    ⍝
[2]    ⍝ ENLEVE LES BLANCS AUTOUR DES SYMBOLES-TYPE
[3]    ⍝
[4]    UB←' '=SI
[5]    UP←SI∈'←:∪=,⍺⍵'
[6]    UC←~(UB∧1⌽UP)∨UB∧¯1⌽UP
[7]    SO←UC/SI
       ∇
       ∇REGULARISE[□]∇
       ∇SO←REGULARISE SI;UP;UB;UC
[1]    ⍝
[2]    ⍝ PLACE UN BLANC DEVANT LA CHAINE SI
[3]    ⍝
[4]    SI←' ',SI
[5]    ⍝
[6]    ⍝ TROUVE LA POSITION DES BLANCS DANS LA CHAINE SI
[7]    ⍝
[8]    UB←' '=SI
[9]    ⍝
[10]   ⍝ LIMITE LA SUITE DE BLANCS CONSECUTIFS DANS SI A UN
```

```
[11]  ⍝
[12]  UC←~UB∧1⌽UB
[13]  SI←UC/SI
[14]  ⍝
[15]  ⍝ PLACE UN BLANC APRES LA CHAINE SI
[16]  ⍝
[17]  SI←SI,' '
[18]  ⍝
[19]  ⍝ TROUVE LA POSITION DES BLANCS DANS LA CHAINE SI
[20]  ⍝
[21]  UB←' '=SI
[22]  ⍝
[23]  ⍝ TROUVE LA POSITION DES SYMBOLE-TYPE DANS LA CHAINE
       SI
[24]  ⍝
[25]  UP←SI∈'←:⊃=,⍺⍵'
[26]  ⍝
[27]  ⍝ ENLEVE LES BLANCS QUI ENCADRENT LES SYMBOLES-TYPE
       DANS LA CHAINE SI
[28]  ⍝
[29]  UC←~(UB∧1⌽UP)∨UB∧¯1⌽UP
[30]  SI←UC/SI
[31]  SO←SI
       ∇
       ∇SBCBOARD[⎕]∇
       ∇SBCBOARD
[1]   W'MBVCC←V0 V2 V4 V6 V8 V10 V12 V14 V16 V18 V20 V22 V
       24 V26 V28 V30 V32 V34 V36 V38 V40 V42 V44 V46 V48'
[2]   W'MBVCCA←V48A V46A V44A V42A V40A V38A V36A V34A V32
       A V30A V28A V26A V24A V22A V20A V18A V16A V14A V12A
       V10A V8A V6A V4A V2A V0A'
[3]   W'MBGR←G1 G3 G5 G7 G9 G11 G13 G15 G17 G19 G21 G23 G2
       5 G27 G29 G31 G33 G35 G37 G39 G41 G43 G45 G47'
[4]   W'MBGRA←G47A G45A G43A G41A G39A G37A G35A G33A G31A
       G29A G27A G25A G23A G21A G19A G17A G15A G13A G11A G9
       A G7A G5A G3A G1A'
[5]   W'VCC=211 MBVCC MBVCCA,212 MBVCC MBVCCA,213 MBVCC MB
       VCCA,214 MBVCC MBVCCA'
[6]   W'GND=201 MBGR MBGRA,202 MBGR MBGRA,203 MBGR MBGRA,2
       04 MBGR MBGRA,205 MBGR MBGRA'
       ∇
       ∇SBC905[⎕]∇
       ∇SBC905
[1]   LISTNUM←LISTNUM,(200+⍳5),(210+⍳4),220+⍳3
[2]   BASEVEC←BASEVEC,1+(5⍴BANQUE FIND'CONGR '),(4⍴BANQUE
       FIND'CONVCC'),(BANQUE FIND'CON100'),BANQUE FIND'CON6
       0 '
[3]   BASEVEC←BASEVEC,1+BANQUE FIND'CON86 '
```

```
[4]   BASEPIN1←BASEPIN1,31 5.5 31 29.5 31 53.5 31 77.5 31
       101.5 30 11.5 30 35.5 30 71.5 30 95.5 26 84 84 83.25
       84 4.75
[5]   BASEORI←BASEORI,(9ρ1),4,2ρ2
[6]   BASESPA←BASESPA,(9ρ2),(2ρ1),1.56
[7]   BASEMAT←BASEMAT,(10ρ24 12),(8ρ25 12),50 2 30 2 43 2
       ∇
       ∇SC[⎕]∇
       ∇V←L SC P;M;W;I;J;Q;G;K;Z;K
[1]    →OT×ιV/Z=2,1+N←1↓ρV←N,2 0ρZ←1↓ρP←0 ¯1↓P
[2]    →DI×ιZ=L←⌈Z÷⌈Z÷L←M←W←ρ0ρG←,H
[3]    P←K,P[;⍋|(Z-N)↑,P],2 0↑P←P[;⍋|(Z-N)↑,P],2 0↑P←(0 1×ρ
       K←(2,N)↑P)↑P
[4]   DI:J←M+∈(14×W)ρ',((W-J←J-1)ρJ)',0ρJ←W←¯1+L←L⌊Z-M
[5]    I←M+∈(17×W)ρ',((1+W-I)+ιI←I+1)',0ρI←0
[6]   NE:V←((|W)⌊M+L-N+1)TY(2,ρQ)ρI[Q],J[Q←⍋+/[1]Q×Q←(|P[;I
       ])-|P[;J]]
[7]    →((¯1=W),Z=M←M+L×W≠¯1)/DI,OT
[8]    →NE,(ρI←,I),ρJ←,⍉(I←(2ρK)ρM+ιK)-K←⌊(Z-1↓ρV)⌊L↑3+W←-Q
       ←1
[9]   OT:K←(ρV←⍉((.5××/ρV),2)ρV),ρT←T,[1]V←|(v/0<V)/[1]V←((
       Z-1),4)ρ3 2 1⍉P[;V,(2,0=1↓ρV)ρι2]
       ∇
       ∇SCANBE[⎕]∇
       ∇SCANBE
[1]    LISTNUM←LISTNUM,,(⍉(SECT,4)ρι4)+(4,SECT)ρ200+10×ιSEC
       T
[2]    BASEVEC←BASEVEC,1+((2×SECT)ρBANQUE FIND'CON26 '),(2×
       SECT)ρBANQUE FIND'CONVG '
[3]    BASE
[4]    BASEPIN1←BASEPIN1,H
[5]    BASEORI←BASEORI,,⍉(SECT,4)ρ4 4 1 1
[6]    BASESPA←BASESPA,((2×SECT)ρ1),(2×SECT)ρ4
[7]    BASEMAT←BASEMAT,((4×SECT)ρ13 1),(4×SECT)ρ12 6
       ∇
       ∇SCANBEBOARD[⎕]∇
       ∇SCANBEBOARD
[1]    W'BUSVCC←V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12'
[2]    W'BUSGND←G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12'
[3]    →(¯1+2×7-SECT)+I26
[4]    W'VCC=263 BUSVCC,264 BUSVCC'
[5]    W'GND=263 BUSGND,264 BUSGND'
[6]    W'VCC=253 BUSVCC,254 BUSVCC'
[7]    W'GND=253 BUSGND,254 BUSGND'
[8]    W'VCC=243 BUSVCC,244 BUSVCC'
[9]    W'GND=243 BUSGND,244 BUSGND'
[10]   W'VCC=233 BUSVCC,234 BUSVCC'
[11]   W'GND=233 BUSGND,234 BUSGND'
```

```
[12]  W'VCC=223 BUSVCC,224 BUSVCC'
[13]  W'GND=223 BUSGND,224 BUSGND'
[14]  W'VCC=213 BUSVCC,214 BUSVCC'
[15]  W'GND=213 BUSGND,214 BUSGND'
      ∇
      ∇STAGE[⎕]∇
      ∇STAGE ARGU;LIGNE
[1]   FUNTIE FNUMS
[2]   'BON'FTIE 2
[3]   ⍝
[4]   ⍝-------INITIATION DES PARAMETRES.
[5]   STAGE00
[6]   ⍝
[7]   ⍝-------CLASSIFICATION EN CATEGORIES(STRINGS).
[8]   STAGE01
[9]   ⍝
[10]  ⍝-------REGULARIZATION DES STRINGS.
[11]  STAGE02
[12]  ⍝
[13]  ⍝-------SUBTITUTION DES MACROS.
[14]  STAGE03
[15]  ⍝
[16]  ⍝-------COMPILATION ET IDENTIFICATION
[17]  ⍝-------DES COORDONNEES POUR CHAQUE CATEGORIE.
[18]  STAGE04
[19]  ⍝
[20]  ⍝-------COMPILATION DES LIENS ENTRE COORDONNEES.
[21]  ⍝ STAGE05
[22]  FUNTIE 2
      ∇
      ∇STAGE00[⎕]∇
      ∇STAGE00;STRING1;STRING2;STRING3;STRING4;STRING5;STR
      ING6
[1]   ⍝-------INITIATION DES PARAMETRES.
[2]   LIGNE←0
[3]   STRING1←⍳0
[4]   STRING2←⍳0
[5]   STRING3←⍳0
[6]   STRING4←⍳0
[7]   STRING5←⍳0
[8]   STRING6←⍳0
[9]   STRING1 FREPLACE 2 11
[10]  STRING2 FREPLACE 2 12
[11]  STRING3 FREPLACE 2 13
[12]  STRING4 FREPLACE 2 14
[13]  STRING5 FREPLACE 2 15
[14]  STRING6 FREPLACE 2 16
      ∇
```

```
      ∇STAGE01[□]∇
      ∇STAGE01;STRING1;STRING2;STRING3;STRING3;STRING4;STR
      ING5;STRING6;SECT;TYPE
[1]   STRING1←FREAD 2 11
[2]   STRING2←FREAD 2 12
[3]   STRING3←FREAD 2 13
[4]   STRING4←FREAD 2 14
[5]   STRING5←FREAD 2 15
[6]   STRING6←FREAD 2 16
[7]   SECT←FREAD 2 18
[8]   ∈ARGU
[9]   →((TYPE←FREAD 2 22)=1)/ST2
[10]  →(TYPE=3)/ST1
[11]  SCANBEBOARD
[12]  →ST2
[13] ST1:SBCBOARD
[14] ST2:STRING1 FREPLACE 2 11
[15]  STRING2 FREPLACE 2 12
[16]  STRING3 FREPLACE 2 13
[17]  STRING4 FREPLACE 2 14
[18]  STRING5 FREPLACE 2 15
[19]  STRING6 FREPLACE 2 16
      ∇
      ∇STAGE02[□]∇
      ∇STAGE02;STRING1;STRING2;STRING3;STRING4;STRING5;STR
      ING6
[1]   STRING1←FREAD 2 11
[2]   STRING2←FREAD 2 12
[3]   STRING3←FREAD 2 13
[4]   STRING4←FREAD 2 14
[5]   STRING5←FREAD 2 15
[6]   STRING6←FREAD 2 16
[7]   STRING1←REGULARISE STRING1
[8]   STRING2←REGULARISE STRING2
[9]   STRING3←REGULARISE STRING3
[10]  STRING4←REGULARISE STRING4
[11]  STRING5←REGULARISE STRING5
[12]  STRING6←REGULARISE STRING6
[13]  STRING1 FREPLACE 2 11
[14]  STRING2 FREPLACE 2 12
[15]  STRING3 FREPLACE 2 13
[16]  STRING4 FREPLACE 2 14
[17]  STRING5 FREPLACE 2 15
[18]  STRING6 FREPLACE 2 16
      ∇
      ∇STAGE03[□]∇
      ∇STAGE03;STRING1;STRING2;STRING3;STRING4;STRING5;STR
      ING6
```

```
[1]   STRING1←FREAD 2 11
[2]   STRING2←FREAD 2 12
[3]   STRING3←FREAD 2 13
[4]   STRING4←FREAD 2 14
[5]   STRING5←FREAD 2 15
[6]   STRING6←FREAD 2 16
[7]   STRING2←STRING1 SUBSTITUE STRING2
[8]   STRING3←STRING1 SUBSTITUE STRING3
[9]   STRING4←STRING1 SUBSTITUE STRING4
[10]  STRING5←STRING1 SUBSTITUE STRING5
[11]  STRING6←STRING1 SUBSTITUE STRING6
[12]  STRING2 FREPLACE 2 12
[13]  STRING3 FREPLACE 2 13
[14]  STRING4 FREPLACE 2 14
[15]  STRING5 FREPLACE 2 15
[16]  STRING6 FREPLACE 2 16
      ∇
      ∇STAGE04[□]∇
      ∇STAGE04;BANQUE;BASEVEC;BASEPIN1;BASEORI;LISTNUM;COD
      EGEO;PINGEO;SET;SETD;CLASSED;XYM;BASESPA;SETL;FE
[1]   FUNTIE FNUMS
[2]   'BON'FTIE 2
[3]   ⍝
[4]   ⍝ BANQUE CONTIENT DES DESCRIPTIONS DE CHIPS
[5]   ⍝
[6]   BANQUE←FREAD 2 8
[7]   ⍝
[8]   ⍝ BASEVEC INDIQUE LA POSITION D'UN CHIP DANS BANQUE
[9]   ⍝
[10]  BASEVEC←FREAD 2 9
[11]  ⍝
[12]  ⍝ BASEPIN1 DONNE LA COORDONNEE DE LA PREMIERE PIN
[13]  ⍝
[14]  BASEPIN1←FREAD 2 2
[15]  ⍝
[16]  ⍝ BASEORI DONNE L'ORIANTATION DU CHIP
[17]  ⍝
[18]  BASEORI←FREAD 2 1
[19]  ⍝
[20]  ⍝ LISTNUM DONNE LA LISTE DES NUMEROS DES CHIPS
[21]  ⍝
[22]  LISTNUM←FREAD 2 5
[23]  CODEGEO←FREAD 2 10
[24]  PINGEO←FREAD 2 17
[25]  BASESPA←FREAD 2 4
[26]  SET←1
[27]  SETD←⍳0
[28]  CLASSED←6 0⍴''
```

```
[29]  ⍝
[30]  ⍝ LES COLONNES 1 ET 2 DE XYM SONT LES COORDONNEES
[31]  ⍝ LA COLONNE 3 INDIQUE LE TYPE D'ENONCE
[32]  ⍝ LA COLONNE 4 INDIQUE LA GANG
[33]  ⍝ LA COLONNE 5 INDIQUE LA PROVENANCE
[34]  ⍝
[35]  XYM←5 0⍴0
[36]  FE←100
[37]  CUMUL6
[38]  CUMUL2
[39]  ⍝CUMUL3
[40]  CUMUL4
[41]  CUMUL5
[42]  ⍝ CUMUL7
[43] FIN:XYM FREPLACE 2,FE+1
       ∇
       ∇START[⎕]∇
       ∇START;BANKBASE;BANQUE;BASEORI;BASEPIN1;BASEMAT;BASE
       SPA;LISTNUM;BASEVEC;DIRECTORY;DIRECNUM
[1]    START1
       ∇
       ∇START1[⎕]∇
       ∇START1;N;CAR;COLN;ERREUR;LI;LI1;MAT;ORDRE;PAT1;PAT2
       ;POSI;POSIOK;RO;SECT;TYPE;VEC;B;H;Y1;Y2
[1]    FUNTIE FNUMS
[2]    'BAN.65888'FSTIE 1
[3]    'BON'FTIE 2
[4]    FDROP 2 400
[5]    DIRECTORY←FREAD 1 1
[6]    DIRECNUM←FREAD 1 2
[7]    FHOLD 1
[8]    LISTNUM←BASESPA←BASEVEC←BASEPIN1←BASEORI←BASEMAT←⍳0
[9]    SECT←ERREUR←0
[10]   'TYPE OF BOARD?'
[11]   'MATRIX:1'
[12]   'SCANBE:2'
[13]   'SBC-905:3'
[14] TYPEDEM:TYPE←⍞DEM'TYPE?'
[15]   →TYPEDEM×⍳1≠+/TYPE=⍳3
[16]   TYPE FREPLACE 2 22
[17]   →NEXT×⍳TYPE≠2
[18] SECDEM:SECT←⍞DEM'NUMBER OF SECTIONS?    '
[19]   →SECDEM×⍳1≠/SECT=⍳6
[20]   SECT FREPLACE 2 18
[21] NEXT:'NAME OF THE LOCATION PROGRAM:'
[22]   BANQUE←6 1⍴'        '
[23]   →(TYPE=1 2 3)/MATR,SCAN,SBC
[24] SCAN:BANQUE←⍞(FREAD 1 3)[⍳54;]
```

```
[25]  SCANBE
[26]  →MATR
[27] SBC:BANQUE←⍞FREAD 1 3
[28]  SBC905
[29] MATR:∈⍞
[30]  →0×⍳ERREUR=1
[31]  'NO ERROR IN THE LOCATION PROGRAM    '
[32]  BASEORI FREPLACE 2 1
[33]  BASEPIN1 FREPLACE 2 2
[34]  BASEMAT FREPLACE 2 3
[35]  BASESPA FREPLACE 2 4
[36]  LISTNUM FREPLACE 2 5
[37]  BANQUE FREPLACE 2 8
[38]  BASEVEC FREPLACE 2 9
[39]  CODEGEO FREPLACE 2 10
[40]  PINGEO FREPLACE 2 17
[41]  'NAME OF THE DEFINITION PROGRAM:'
[42]  STAGE⍞
[43]  MIP
[44]  'NO ERROR IN THE DEFINITION PROGRAM'
[45]  ' '
[46]  'TO OBTAIN A DRAWING OF THE BOARD,TYPE ''BOARD'' .'
[47]  'THE CIRCUIT LAYOUT CALCULATION IS OBTAINED BY:'
[48]  ')LOAD WRAP2 ,AND EXECUTING ''START'''
[49]  FUNTIE FNUMS
       ∇
       ∇SUBSTITUE[⍞]∇
       ∇SO←STRING1 SUBSTITUE SI;TEXT;RES;M;N3;I;U;N1;N2
[1]    ⍝
[2]    ⍝ PLACE DES BLANCS AUTOUR DES SYMBOLES-TYPE
[3]    ⍝
[4]    SI←PAD SI
[5]   J3:SO←⍳0
[6]    STRING1←'α'FP STRING1
[7]    STRING1←'ω'FP STRING1
[8]    TEXT←RES
[9]    TEXT←'←'FP TEXT
[10]   M←RES
[11]   M←' ',M,' '
[12]   N3←⍴M
[13]   I←¯1+⍳N3
[14]  J2:U←∧⌿I∘.=M∘.=SI
[15]   N1←⍴U
[16]   N2←U⍳1
[17]   →(N2>N1)/J1
[18]   SO←SO,(N2↑SI),TEXT
[19]   SI←(¯2+N2+N3)↓SI
[20]   →J2
```

```
[21] J1:SO←SO,SI
[22]  →(2≥ρSTRING1)/J4
[23]  SI←SO
[24]  →J3
[25]  ⍝
[26]  ⍝ ENLEVE LES BLANCS AUTOUR DES SYMBOLES-TYPE
[27]  ⍝
[28] J4:SO←REGU SO
       ∇
       ∇W[⎕]∇
       ∇W STRING;N
[1]    LIGNE←LIGNE+1
[2]    N←CATEGORIE STRING
[3]    →(N=0 1 2 3 4 5 6 7)/J0,J1,J2,J3,J4,J5,J6,J7
[4]    ⍝
[5]    ⍝--------ERREUR:CATEGORIE INEXISTANTE.
[6]    ⍝
[7]   J0:'SYNTAX ERROR IN STATEMENT  ';LIGNE
[8]    'UNDEFINED STATEMENT'
[9]    →0
[10]   ⍝--------MACRO(←)
[11]   ⍝
[12] J1:STRING1←STRING1,('I4'⍺LIGNE),'⍺',STRING,'⍵'
[13]   →0
[14]   ⍝--------CLASSES LIBRE(:)
[15]   ⍝
[16] J2:STRING2←STRING2,('I4'⍺LIGNE),'⍺',STRING,'⍵'
[17]   →0
[18]   ⍝--------CLASSES FERMES(∪:)
[19]   ⍝
[20] J3:STRING3←STRING3,('I"'⍺LIGNE),'⍺',STRING,'⍵'
[21]   →0
[22]   ⍝--------CHEMIN LIBRES(,)
[23]   ⍝
[24] J4:STRING4←STRING4,('I4'⍺LIGNE),'⍺',STRING,'⍵'
[25]   →0
[26]   ⍝--------CHEMINS FERMES(∪,)
[27]   ⍝
[28] J5:STRING5←STRING5,('I4'⍺LIGNE),'⍺',STRING,'⍵'
[29]   →0
[30]   ⍝--------BORNES(=)
[31]   ⍝
[32] J6:STRING6←STRING6,('I4'⍺LIGNE),'⍺',STRING,'⍵'
[33]   →0
[34]   ⍝--------LIGNE DE COMMENTAIRES
[35]   ⍝
[36] J7:→0
       ∇
```

```
      )WSID
WRAP2
      )FNS
CO        CONV     DIAG     FIL      FLIM     FREAD    FREPLACE
FSTIE     FUNTIE   LAY      LK       OPC      PLOT     READ     RP
SC        START    TABLE    TY       WIREWRAP          CA
CAL
      )VARS
FA        IA       KA       NA
      ∇CO[□]∇
      ∇R←CO A
[1]   R←-/C←(0 1↓C),2 1↑A
      ∇
      ∇CONV[□]∇
      ∇M←W CONV P
[1]   M←(2,ρP)ρ1↓(W|P),⌊(P←P-1)↑W
      ∇
      ∇DIAG[□]∇
      ∇DIAG Q;A;X;J
[1]   0ρ□←'LINES WITH COMMON POINTS:',0ρQ←ρ1ρX←⌈V
[2]   PR:→PR×ι0<ρX←(ρA)↓X,0ρJ←∈(1<ρJ)/'□←J',0ρJ←RP(~Q←~(R←Q
      /R)∈A←(X⍳0)↑X)/L←Q/L
      ∇
      ∇FIL[□]∇
      ∇R←FIL A
[1]   G FREPLACE 2,A
[2]   G←0 4ρR←ι0
      ∇
      ∇LAY[□]∇
      ∇T←LAY W;Q;X;J;K;N;D;H;I;L;S
[1]   X←'ABCDEFGHIJKLMNOPQRSTUVWXYZ',F←0ρT←0 4ρ0
[2]   S←(D×⌊/36 24↑D←⌈-/D)⍎'FRS',0ρ(,D←⍉2 2ρ(⌊/S),⌈/S←⌈(50
      0<⌈,1 0↓W)/W)⍎'SCL'
[3]   ON:→OV×ι0=Q←,1 ¯1↓J,2 0ρH←1,0ρW←(0 1×ρJ←(2,(,1 0↓W)ι1
      ↓ρN←2 0ρ0)↓W)↓W
[4]   S←(ρH←G),ρQ TY N,V←2 0ρG←1,0ρW←(0,Q)↓W,2 0ρN←(2,Q)↑W
[5]   OV:→ON×ι(ρ0ρX←1ϕX)<1↓ρW,2 0ρPLOT J
      ∇
      ∇LK[□]∇
      ∇E←LK P;K;R;G
[1]   →FI×ι0=ρP←K↓P,E←0ρG←(ρR)ρρR←RP R←¯1↓(K←Pι0)↓P
[2]   AG:→NE×ι~∨/K←RϵE←RP E,0ρP←(1↓ρE←¯1↓(Pι0)↓P)↓P
[3]   →(E←ι0),(0<ρP,0ρG←((~K)/G),(ρE)ρG[(GϵK/G)/ιρG]←1+⌈/G
      ,0ρR←((~K)/R),E)ϕFI,AG
[4]   NE:→AG×E←ι0<ρP,0ρR←R,E,0ρG←G,(ρE)ρ1+⌈/G
[5]   FI:→FI×ι0<ρG←K/G,0ρR←K/R,0ρE←E,((~K←G≠1↑G)/R),0
      ∇
      ∇OPC[□]∇
      ∇F←OPC R;I;C;D;BN;NB;S;O;ID;K;A;G
```

```
[1]    A←300,0ρG←0 4ρO←2 1ρ(⌈/¯1 0↓S),⌊/1 0↓S←20×S←⍵S←(2×(1
        ↑ρS),1)ρS←FREAD 2 150
[2]    G←G,[1]0 69,D←-CO C←O,O,2 0ρI←ρ0ρBN←(FREAD 2 151)[1]
[3]   NE:K←∈(0=20|I←I+1)/'G←G,[1]0 69,D←-CO O'
[4]    K←∈(BN≠NB←(FREAD 2 151)[I])/'G←G,[1]I,59,2ρρ0ρBN←NB'
[5]    K←(ρG←G,[1]I,68,D←-CO S←0 1↓S),ρG←G,[1]I,BN,-CO S
[6]    →NE×⍳~F,∈((F←0=1↓ρS←0 1↓S)∨0=100|I)/'FIL A←A+1'
        ∇
        ∇PLOT[□]∇
        ∇G←PLOT A
[1]    →0×⍳G←Y,∈Y/'0ρ50 SC A'
[2]    □←A,2 0ρ(1↑X)□'SYM',0ρ21□'MOD'
[3]    □←-A,2 0ρ'□'□'SYM'
[4]    □←50 SC A,2 0ρG←31□'MOD'
[5]    ∈Z/'Z←□,0ρ□←''0 OU 1'''
        ∇
        ∇READ[□]∇
        ∇R←READ A;B;I;J
[1]    R←0ρJ←I←1
[2]    ∈(∧/A=0)/'A←⍳¯1+1↓FLIM 2'
[3]    A←(A<1↓FLIM 2)/A
[4]   BK:R←R,,(B←(¯2↑5,ρB)ρB←FREAD 2,A[J])[I;]
[5]    →BK×⍳(ρA)≥J←J+1
[6]    R FREPLACE 2,49↑I
[7]    →BK×⍳5≥I←I+J←1↑ρR←⍳0
[8]    R←⍳0
        ∇
        ∇RP[□]∇
        ∇R←⍵P X
[1]    R←((X⍳X)=⍳ρX)/X
        ∇
        ∇SC[□]∇
        ∇V←L SC P;M;W;I;J;Q;G;K;Z
[1]    →OT×⍳∨/Z=2,1↑N←1↓ρV←N,2 0ρZ←1↓ρP←0 ¯1↓P
[2]    →DI×⍳Z=L←⌈Z↑⌈Z↑L+M←W←ρ0ρG←,H
[3]    P←P[1 2;],2 0ρV←P[3;]⍳V,2 0ρP←P[;⍋|Z↑,P],3 0ρP←P[;⍋|
        Z↑Z↓,P],3 0ρP←P,[1]⍳Z
[4]   DI:J←M↑∈(14×W)ρ',((W-J←J-1)ρJ)',0ρJ←W←¯1↑L←L⌊Z-M
[5]    J←Q/J,0ρI←(Q←(P[1;J]>0)∨P[2;I]>0)/I←M↑∈(17×W)ρ',((1↑
        W-I)↑⍳I←I+1)',0ρI←0
[6]   NE:V←(|W)TY(2,ρQ)ρI[Q],J[Q←⍋↑+/[1]Q×Q←(|P[;I])-|P[;J]]
[7]    →((¯1=W),Z←M←M+L×W≠¯1)/DI,OT
[8]    →NE,(ρI←,I),ρJ←,⍵(I←(2ρK)ρM+⍳K)-K←⌊(Z-1↑ρV)⌊L↑3↑W←-Q
        ←1
[9]   OT:K←(ρV←⍵((.5××/ρV),2)ρV),ρT←T,[1]V←|(∨/0<V)/[1]V←((
        Z-1),4)ρ3 2 1⍵P[;V,(2,0=1↑ρV)ρ⍳2]
        ∇
        ∇START[□]∇
        ∇START;A
```

```
[1]    →1×⍳~(A←⍞,0ρ⍞←'TERMINAL? (2=4015; 3=4662)')∈2 3
[2]    A←A,'GDV'
[3]    FUNTIE 2
[4]    'BON'FSTIE 2
[5]    (WIREWRAP READ(100+⍳10))FREPLACE 2 20
[6]    FUNTIE 2
[7]    'COMPLETED'
[8]    'TO OBTAIN CHIP DRAWINGS AND A LIST OF UNUSED PINS,
        TYPE:'
[9]    ')LOAD WRAP3'
[10]   'AND EXECUTE: SETDRAWING'
        ∇
        ∇TABLE[⍞]∇
        ∇R←TABLE T;D;B;Q;C;N
[1]    T←(2×(T[;1]>T[;3])∨(T[;1]=T[;3])∧T[;2]<T[;4])⌽T
[2]    Q←D=D←⌈4.5+((D×D←-/T[;2 4])+D×D←-/T[;1 3])*.5
[3]    C←B←(1++,B<⌈/D)↑B←9 11,(10+2×⍳45),100+3×⍳32
[4]   BK:→BK×⍳0<ρB←(ρ1ρQ←Q←D>1↑B)↓B
[5]    D←⍳0
[6]    (10⊥0 8⊤Q←Q[B←⍋Q])FREPLACE 2 151
[7]    T[B;]FREPLACE 2 150
[8]    T←⍳0
[9]    0ρ⍞←(18↑'NUMBER'),(18↑'BIN'),'LENGTH'
[10]   0ρ⍞←⍉(3,ρB)ρB,(10⊥0 8⊤N),2+.1×C[N←¯1↓D/Q],0ρB←1↓B-¯1
        ⌽B←(D←B=Q⍳Q)/B←⍳ρQ←Q,0
        ∇
        ∇TY[⍞]∇
        ∇R←W TY Q;E;K;L;S;Y
[1]    K←∈K/'R←V',∈(S←~K←(Y←0)≠1↓ρV)/'0ρQ←(1=ρR←2 1↑Q)↓Q'
[2]   LP:→0×⍳(∨/0∈E),0ρQ←(1=ρE←2 1↑Q)↓∈'Q',(∨/Y)/'←(~∨/[1]Q
        ∈Y/,E)/Q'
[3]    →LP×⍳∨/(Y←3=(ρK),ρL),(K←(∨/[1]R∈1↑,E)/G)∈L←(∨/[1]R∈1
        ↑,E)/G
[4]    →LP×W>S←S↓ρ1ρG←G,∈(13×0=ρK←K,L)↓'G[(G∈K)/⍳ρG]←1+⌈/G,
        0ρR←R,E'
        ∇
        ∇WIREWRAP[⍞]∇
        ∇R←WIREWRAP Z;X;P;W;V;U;B;C;N;I;E;J;S;K;F;A;D;L;G;Q;
        T;Y
[1]    V←0ρX←FREAD 2 53,0ρP←(¯1*D←6=Z←FREAD 2 52)×R←X←(W←⌈/
        X←FREAD 2 50)×¯1+FREAD 2 51
[2]    X←|X,0ρC←U/X,0ρB←(U←(5=Z)∨6=Z)/P,0ρA←|D/P,0ρL←,FREAD
        2 54
[3]   ON:→ON×⍳0<ρX←Z/X,0ρP←Z/P,0ρV←V,((~Z←X≠1↑X)/P),0
[4]    DIAG⍞←F←2 0ρT←' COMPLETED',⍞←0ρV←(¯1*V∈A)×V←LK|V
[5]   NE:→KP×⍳0=∨/(0ρS←×E),J←(|E←¯1↓(I←V⍳0)↑V)∈|B,0ρN←2 1ρ0
[6]    S←×E←(((|E)⍳|E)=⍳ρE)/E←(RP A←(A/B)[K]),((~J)/E),0ρX←
        X[K←⍋X←(A←(|B)∈|J/E)/C]
```

```
[7]   N←((1↓ρN),0),N←((X⍳X)≠⍳ρX)/(2,ρN)ρ(¯1φN),N←(|E)⍳|4
[8]  KP:→NE×⍳0<ρV←I↓V,0ρF←F,(((2,ρE)ρS)×W CONV|E),N
[9]   →OV×⍳Y←~□,0ρ□←'GRAPHICS? 0 OU 1',□←0ρQ←'TABLE'
[10]  Z←□,0ρ□←'COMPLETE(0) OR BY SEGMENT(1)?'
[11] OV:Q,T,□←0ρOPC TABLE R,[1]0 4ρ□,0ρ□←'TYPE 1 FOR ',Q,0
       ρ□←'LAYOUT',T,0ρR←LAY F
       ∇
```

```
      )WSID
WRAP3
      )FNS
CUEILLE DEM     DESSIN1 DRAW    ESSAI   ESSAIG  FAPPEND
FCNUM   FCREATE FDROP   FERASE  FHOLD   FLIM    FNAMES
FNUMS   FRDCIA  FRDCIDT FREAD   FREEPIN FREPLACE
FSDROP  FSTIE   FTIE    FUNTIE  PLANTE  POT1    POT3
POUSSE  PRAR    SETDRAWING       TRAITE  CA      CAL
      )VARS
COARB   STOP    TYPE    FA       IA      KA      NA
      ∇CUEILLE[⎕]∇
      ∇CUEILLE;N;R
[1]   R←FREAD 2 21
[2]   R←R+(1↑ρRACINE)×R=0
[3]   RACINE←FREAD 2 7
[4]   RACINE←RACINE,[1]1 3ρ0
[5]   RACINE←RACINE,((1↑ρRACINE),9)ρ0
[6]   N←0
[7]  BOUC:→BOUC×ι(N≠1↑ρR)×1+ρ0ρRACINE[R[N;1];3+ι9]←,RACINE
       [R[(N←N+1);2 3 4];ι3]
[8]   RACINE FREPLACE 2 7
      ∇
      ∇DEM[⎕]∇
      ∇B←DEM A
[1]   ⍝←A
[2]   B←(ρA)+⍝
      ∇
      ∇DESSIN1[⎕]∇
      ∇DESSIN1 CHOP;HORI;LARG;S;TEX2;X;X1;X3;YF;YH;TEX;TEX
      T;VERT;BANQUE;A;B1
[1]   →1×ι~(S←⎕,0ρ⎕←'TERMINAL? (2=4015; 3=4662)')∈2 3
[2]   S←S⍪'GDV'
[3]   CHOP←2ρCHOP,0
[4]   OFFSET←-CHOP[2]
[5]   CHOP←CHOP[1]
[6]   FUNTIE FNUMS
[7]   'BON'FTIE 2
[8]   TYPE←FREAD 2 22
[9]   TRAITE CHOP
[10]  →0×ιSTOP=1
[11]  CODESOUS←0 34 30
[12]  CHIP←(CHOP=FREAD 2 5)/ι(ρFREAD 2 5)
[13] DEBUT:S←1⍪'CTY'
[14]  TEX←TEX1←'0123456789'
[15]  NOM←6ρ⍵(BANQUE←FREAD 2 8)[;¯2+(FREAD 2 9)[CHIP]]
[16] SUITE:Y←1
[17]  X3←X←0
[18]  VERT←|(FREAD 2 3)[¯1+2×CHIP]
```

```
[19]  OFFSET←OFFSET+VERT-20
[20]  HORI←5
[21]  S←¯10 15 0 22⎕'SCL'
[22]  ⎕←2 5⍴(0,0,HORI,HORI,0,0,VERT,VERT,0,0)-(5⍴0),5⍴OFFS
      ET
[23]  LARG←.75
[24] SCAN2:⎕←2 5⍴X1,X,X,X1,(X1←X-LARG),YF,YF,YH,(YH←YF+LAR
      G),YF←(Y+¯.5-LARG÷2)-OFFSET
[25]  →DROITE×⍳X3=1
[26]  S←((¯10+X-.5×LARG),((-OFFSET)+¯.27+(1+VERT-Y)-.5×LAR
      G))⎕'CUR'
[27]  ESSAI Y-1
[28]  TEXT←SUP,' ',TEX[1+10 10 10⍴Y],'  ',6⍴⍞BANQUE[;Y+¯1+
      (FREAD 2 9)[CHIP]]
[29] OKALL:⎕←TEXT
[30]  →SCAN2×⍳(Y←Y+1)≤VERT
[31]  Y←1
[32]  X←X+HORI+LARG
[33]  →SCAN2×⍳(X3←X3+1)≤1
[34]  →NAME
[35] DROITE:TEXT←6⍴⍞BANQUE[;(2×VERT)+(FREAD 2 9)[CHIP]-Y]
[36]  ESSAIG(2×VERT)-Y
[37]  TEXT←(-¯1+(' '≠⌽TEXT)⍳1)⌽TEXT
[38]  TEXT←TEXT,'  ',(,TEX[1+10 10 10⍴((2×VERT)-Y+¯1)]),SU
      P
[39]  S←((¯1.8+X-.5×LARG),((-OFFSET)+¯.27+(1+VERT-Y)-.5×LA
      RG))⎕'CUR'
[40]  →OKALL
[41] NAME:S←(¯.25,(-OFFSET)+VERT+1)⎕'CUR'
[42]  S←0⎕'CTY'
[43]  S←.6 1.2⎕'CSZ'
[44]  ⎕←(TEX1[1+10 10 10⍴CHOP]),' ',NOM
[45]  FUNTIE 2
      ∇
      ∇DRAW[⎕]∇
      ∇DRAW ABC;CHIP;TEX1;NOM;Y;SUP;CODESOUS;B;C;D;DATA;FL
      AG;PAT1;PAT2;Z;M;OFFSET
[1]   DESSIN1 ABC
      ∇
      ∇ESSAI[⎕]∇
      ∇ESSAI H;K
[1]   FLAG←0
[2]   SUP←''
[3]   →AJOU×⍳COARB='N'
[4]   DATA←,M[1+H;]
[5]   DATA←DATA-12⍴0,CODESOUS[TYPE],0
[6]   →CONHO×⍳(DATA[2]<0)∨(DATA[2]>50)∧TYPE≠1
[7]  NEXT:SUP←'  ↔  ',(2⍴POT1 1⍴DATA[3]),TEX1[1+10 10⍴DAT
```

```
      A[2]]
[8]   K←1
[9]  HERE:→AJOU×ιDATA[4+3×K-1]=0
[10]  SUP←'  ',(Z←PRAR K),SUP
[11]  →HERE×ι(K←K+1)≤3
[12] AJOU:SUP←((50-ρSUP)ρ' '),SUP
[13]  →0
[14] CONHO:→CONSBC×ιTYPE=3
[15]  FLAG←1
[16]  A←POT1 1ρDATA[3]
[17]  B←7 2ρ'    2B2A  1B1A'
[18]  DATA[1 2 3]←|DATA[1 2 3]
[19]  B1←,B[DATA[2];]
[20]  4←(B1,,A)[3 4 1 2 5 6]
[21]  SUP←'  ↔  ',A
[22]  FLAG←0
[23]  →NEXT+1
[24] CONSBC:SUP←'  ↔  ',POT3(DATA[2 3])
[25]  →NEXT+1
      ∇
      ∇ESSAIG[□]∇
      ∇ESSAIG H;K
[1]   FLAG←0
[2]   →FIN×ιCOARB='N'
[3]   DATA←,M[1+H;]
[4]   DATA←DATA-12ρ0,CODESOUS[TYPE],0
[5]   →CONHO×ι(DATA[2]<0)∨(DATA[2]>50)∧TYPE≠1
[6]  NEXT:SUP←'  ',(2ρPOT1 1ρDATA[3]),TEX1[1+10 10τDATA[2]
      ],'  ↔  '
[7]   K←1
[8]  HERE:→0×ιDATA[4+3×K-1]=0
[9]   SUP←SUP,(Z←PRAR K),'  '
[10]  →HERE×ι(K←K+1)≤3
[11]  →0
[12] FIN:SUP←''
[13] CONHO:→CONSBC×ιTYPE=3
[14]  FLAG←1
[15]  A←POT1 1ρDATA[3]
[16]  B←7 2ρ'    2B2A  1B1A'
[17]  DATA[1 2 3]←|DATA[1 2 3]
[18]  B1←,B[DATA[2];]
[19]  A←(B1,,A)[3 4 1 2 5 6]
[20]  SUP←'  ',A,'  ↔  '
[21]  FLAG←0
[22]  →NEXT+1
[23] CONSBC:SUP←'  ',(POT3 DATA[2 3]),'  ↔  '
[24]  →NEXT+1
      ∇
```

```
      ∇FREEPIN[□]∇
      ∇FREEPIN;A;BASEFIND;BASEVEC;DATA;F;FLAG;LISTNUM;M;N;
      PAT1;PAT2;R;TEX1;D;CHLEN;K1
[1]   FLAG←0
[2]   TEX1←'0123456789'
[3]   FUNTIE FNUMS
[4]   'BON'FTIE 2
[5]   TYPE←FREAD 2 22
[6]   R←~(A←⍳(1+⍴FREAD 2 7))∈(FREAD 2 21)[;1]
[7]   M←R/[1]FREAD 2 7
[8]   M[;2]←(M[;2]-34×(TYPE=2))-30×(TYPE=3)
[9]   A←¯1+(A≠0)/A←R×A
[10]  N←0
[11]  BASEFIND←FREAD 2 6
[12]  BASEVEC←FREAD 2 9
[13]  LISTNUM←FREAD 2 5
[14] LI:→LI×⍳N<⍴A×1+⍴A[N]←1+A[N]-1+(1⌽A[N←N+1]<BASEFIND)/B
      ASEFIND
[15]  'THE FOLLOWING PINS ARE NOT USED:'
[16]  CHLEN←¯1+(1⌽BASEFIND)-BASEFIND
[17]  N←0
[18]  K1←1
[19] LO:DATA←0 0 0,M[N+1;]
[20]  →VIDE×⍳(LISTNUM[K1]≠M[N+1;1])
[21]  →CHIFIN×⍳(N+CHLEN[K1])>⍴A
[22]  →CHIPLI×⍳M[N+1;1]=M[N+CHLEN[K1];1]
[23] CHIFIN:F←6⍴⍉(FREAD 2 8)[;A[N←N+1]+¯1+BASEVEC[(DATA[4]
      =LISTNUM)/⍳(⍴LISTNUM)]]
[24]  F←(PRAR 1),'  PIN  ',TEX1[1+10 10 10⊤A[N]],'    ',F
[25]  K1←K1+1×(M[N+1;1]≠M[N;1])
[26]  F
[27]  →LO×⍳N<⍴A
[28]  FUNTIE 2
[29]  →0
[30] CHIPLI:'CHIP (',(TEX1[1+10 10 10⊤DATA[4]]),') NOT USE
      D'
[31]  N←N+CHLEN[K1]
[32]  K1←K1+1
[33]  →LO×⍳N<⍴A
[34]  FUNTIE 2
[35]  →0
[36] VIDE:K1←K1+1
[37]  →LO
      ∇
      ∇PLANTE[□]∇
      ∇PLANTE;N;BASEOR;BASEPIN;MAT;D;VEC;TEMPO;RACETTE;RAC
      INE;BASEFIND;BASEORI;BASEPIN1;BASEMAT;BASESPA;LISTNU
      M
```

```
[1]    BASEORI←FREAD 2 1
[2]    BASEPIN1←FREAD 2 2
[3]    BASEMAT←FREAD 2 3
[4]    BASESPA←FREAD 2 4
[5]    LISTNUM←FREAD 2 5
[6]    BASEFIND←1⍴1
[7]    RACINE←0 3⍴0
[8]    N←1
[9]   DEBU:BASEOR←BASEORI[N]
[10]   BASEPIN←BASEPIN1[((2×N)-1),2×N]
[11]   MAT←BASEMAT[((2×N)-1),2×N]
[12]   D←1
[13]   →(BASEOR=⍳4)/UN,DEUX,TROIS,QUAT
[14]  DEUX:D←0
[15]   MAT←1 ¯1×MAT
[16]   BASEPIN←1⌽BASEPIN
[17]   →UN
[18]  TROIS:MAT←¯1×MAT
[19]   →UN
[20]  QUAT:D←0
[21]   MAT←¯1 1×MAT
[22]   BASEPIN←1⌽BASEPIN
[23]  UN:VEC←(((2×|MAT[1]),2)⍴BASEPIN)+⍉(2,2×|MAT[1])⍴(TEMP
         0,⌽TEMPO←((×MAT[1])×(BASESPA[N]×⍳|MAT[1])-BASESPA[N]
         )),(((|MAT[1])⍴0),(|MAT[1])⍴MAT[2]
[24]   →FIN×⍳D=1
[25]   VEC←⌽VEC
[26]  FIN:RACETTE←0 1 1\VEC
[27]   RACETTE[;1]←LISTNUM[N]
[28]   RACINE←RACINE,[1]RACETTE
[29]   BASEFIND←BASEFIND,(¯1↑BASEFIND)+1↑⍴RACETTE
[30]   →DEBU×⍳(N←N+1)≤⍴LISTNUM
[31]   BASEFIND FREPLACE 2 6
[32]   RACINE←RACINE,[1]1 3⍴0
[33]   RACINE FREPLACE 2 7
       ∇
       ∇POT1[⍞]∇
       ∇H←POT1 K;K1;K2;POT
[1]    PAT1←'ABCDEF'
[2]    →(TYPE=⍳3)/MATRI,SCAN,SBC
[3]   SCAN:D←27
[4]    PAT2←'A  B1 C  D2 E  F3 G  44 J  '
[5]    →OUT×⍳FLAG=0
[6]    PAT2←⍉27 3⍴((15⍴' '),('P13P12P11P10P9 P8 P7 P6 P5 P4
         P3 P2 P1 '),27⍴' ')
[7]    →OUT
[8]   MATRI:D←24
[9]    PAT2←'ABCDEFGHIJKLMNOPQRSTUVWX'
```

```
[10] OUT:POT←(6,D)⊤⍞K-1
[11]  →OUT2×⍳FLAG≠0
[12] OUT1:K1←((⍴K1),1)⍴K1←PAT1[1+POT[1;]]
[13]  H←K1,((⍴K2),1)⍴K2←PAT2[1+POT[2;]]
[14]  →0
[15] OUT2:K1←((⍴K1),1)⍴K1←PAT1[1+POT[1;]]
[16]  H←K1,[1]((⍴K2),1)⍴K2←,PAT2[;(1+POT[2;])]
[17]  →0
[18] SBC:PAT2←'ABCD  '
[19]  PAT1←'ABCDEFGHJKLMNPRSTUWX'
[20]  →SBCCON×⍳K≠⌊K
[21]  POT←20 6⊤⍞K-1
[22]  →OUT1
[23] SBCCON:PAT1←'1234567890'
[24]  PAT2←'    G     V '
[25]  POT←10 12⊤⌊⍞K-1
[26]  →OUT1
      ∇
      ∇POT3[□]∇
      ∇H←POT3 K;X;J;PAT2;A
[1]   PAT2←⍞6 6⍴'(221)P(221)P(222)P(222)P(223)P(223)P'
[2]   X←1+((K[1]=¯4 ¯2)/⍳2),(((K[2]>75)×A)/4 3),(A←K[1]=52
      54)/6 5
[3]   PAT2←,PAT2[;X]
[4]   J←((K[1]=¯2)+1+2×|K[2]-84),((K[1]=52)+1+2×(K[2]-83.2
      5)),(K[1]=52)+1+2×(K[2]-4.75)÷1.56
[5]   J←J[⌈X÷2]
[6]   H←PAT2,,TEX1[1+10 10 10⊤J]
      ∇
      ∇POUSSE[□]∇
      ∇POUSSE;ALB2;RACINE2;I;O;M;N;ALB1;R;RACINE
[1]   ALB2←((1+(⍴ALB1←999-(FREAD 2 20)[;2 1 4 3])),2)⍴0
[2]   RACINE2←(1000×(FREAD 2 7)[;2])+(FREAD 2 7)[;3]
[3]   ALB2[;1]←(ALB1[;1]×1000)+ALB1[;2]
[4]   ALB2[;2]←(ALB1[;3]×1000)+ALB1[;4]
[5]   RACINE2←ALB1←ALB2←0⍴M←⍞RACINE2⍳ALB2
[6]   R←0 4⍴I←⍴0⍴O←O[▲O←((O⍳O)=⍳⍴O)/O←,M]
[7]  ON:→ON×⍳(⍴O)>1+⍴R←R,[1]4↑O[I],(O[I]≠N)/N←((N⍳N)=⍳⍴N)/
      N←,(∨/[1]O[I←I+1]=M)/M
[8]   R FREPLACE 2 21
      ∇
      ∇PRAR[□]∇
      ∇Z←PRAR K;B
[1]   B←7 2⍴'    2B2A  1B1A'
[2]   →SPEC×⍳(DATA[5+3×K-1]<0)∨(DATA[5+3×K-1]>50)∧TYPE≠1
[3]   Z←'(',TEX1[1+10 10 10⊤DATA[4+3×K-1]],')',(2⍴POT1 1⍴D
```

```
      ATA[6+3×K-1]),TEX1[1+10 10⍴DATA[5+3×K-1]]
[4]    →0
[5]   SPEC:→CONSBC×⍳TYPE=3
[6]    DATA[(4 5 6)+3×K-1]←|DATA[(4 5 6)+3×K-1]
[7]    FLAG←1
[8]    Z←((,B[(DATA[5+3×K-1]);]),,POT1 1⍴DATA[6+3×K-1])[3 4
       1 2 5 6]
[9]    FLAG←0
[10]   →0
[11]  CONSBC:Z←POT3 DATA[5 6+3×K-1]
       ∇
       ∇SETDRAWING[⎕]∇
       ∇SETDRAWING
[1]    FUNTIE FNUMS
[2]    'BON'FTIE 2
[3]    TYPE←FREAD 2 22
[4]    PLANTE
[5]    POUSSE
[6]    FUNTIE 2
[7]    'COMPLETED'
[8]    'THE FUNCTION ''FREEPIN''LISTS ALL UNUSED PINS OF TH
       E CIRCUIT'
[9]    'THE FUNCTION ''DRAW X''DRAWS THE CHIP NUMBER X.'
[10]   'TO WIREWRAP A CIRCUIT,CONTACT M. LESSARD AT 4315.'
       ∇
       ∇TRAITE[⎕]∇
       ∇TRAITE CHIP;A;D;N;R;R1
[1]    STOP←0
[2]    A←,(FREAD 2 7)[;1]=CHIP
[3]    →NONO×⍳0=+/A
[4]    M←A/[1]FREAD 2 7
[5]    D←⌊/A/⍳(⍴FREAD 2 7)[1]
[6]    M←M,[1]1 3⍴0
[7]    M←M,((1↑⍴M),9)⍴0
[8]    R←(,((FREAD 2 21)[;1])∈(⍳⍴A)×A)/[1]FREAD 2 21
[9]    R1←R←(1↑⍴FREAD 2 7)×R=0
[10]   R←R←(1↑⍴M)×R=0
[11]   N←0
[12]   →0×⍳0=×/⍴R
[13]  BOUC:→BOUC×⍳(N≠1↑⍴R)×1+⍴0⍴M[R[N;1]+1-D;3+⍳9]←,(FREAD
       2 7)[R1[(N←N+1);2 3 4];⍳3]
[14]   →0
[15]  NONO:'CHIP  ';CHIP;'  UNDEFINED'
[16]   STOP←1
       ∇
```

APPENDIX B

Self-Crossing Paths

Any path which folds over itself can be depicted as three sub-paths (A, B and C) plus the two links which actually cross each other, as shown in Fig. B-1.

Each of the crossing links must have one end (in this case pins 2 and 3) connected to the same sub-path (in this case to B) for the ensemble to be a single path and not two disconnected paths.

Interchanging the other two ends of the links will produce a configuration as in Fig. B-2 in which the links do not cross and the ensemble is a single new path.

The four pins constitute a quadrilateral and the sum of two opposite sides (new links) is shorter than the sum of the two diagonals (previous links). Hence the new path is shorter than the original and a self-crossing path is not optimum.

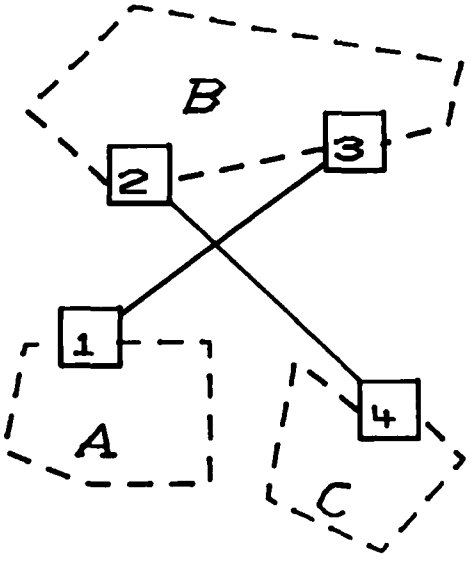


FIGURE - B-1

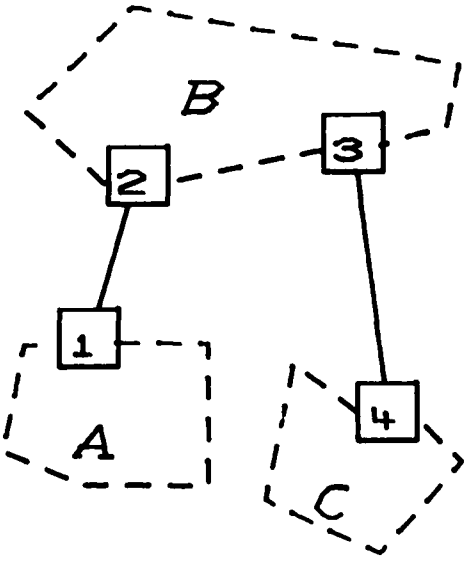


FIGURE - B-2

APPENDIX C

Nonoptimum Paths

Figure C-1 shows a node made up of pins disposed in two concentric circular segments whose path would be established by first linking (radially) each pin of one segment to the opposing pin on the other segment (the shorter links), then all the pins of the inner segment together (the next set of shorter links).

Once this is done, a pin such as A sees its immediate inner segment neighbors occupied by three links and has to go to the last pin for a connection, while the dotted link could have permitted the removal of one of the bothersome links onto its nearest neighbor.

A pin such as B, for the same reason, would end up being linked to an outer segment pin, producing a crossing.

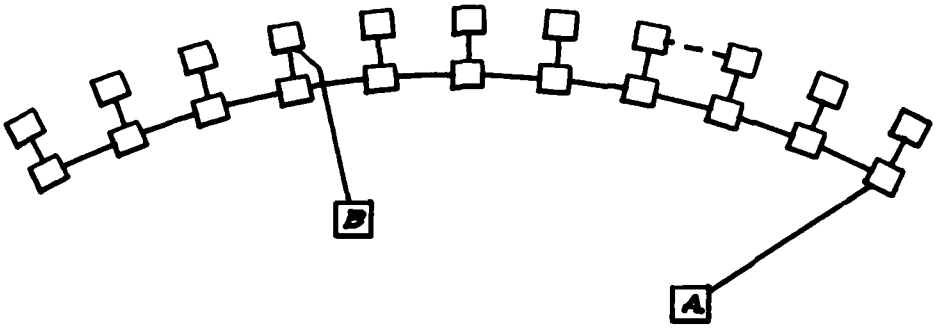


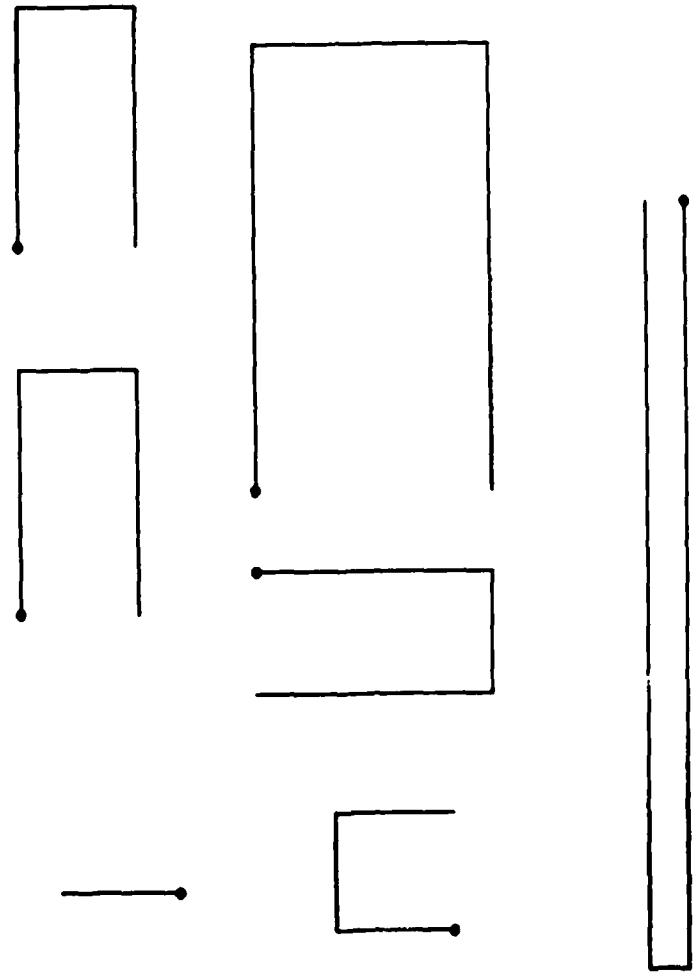
FIGURE - C-1

APPENDIX D

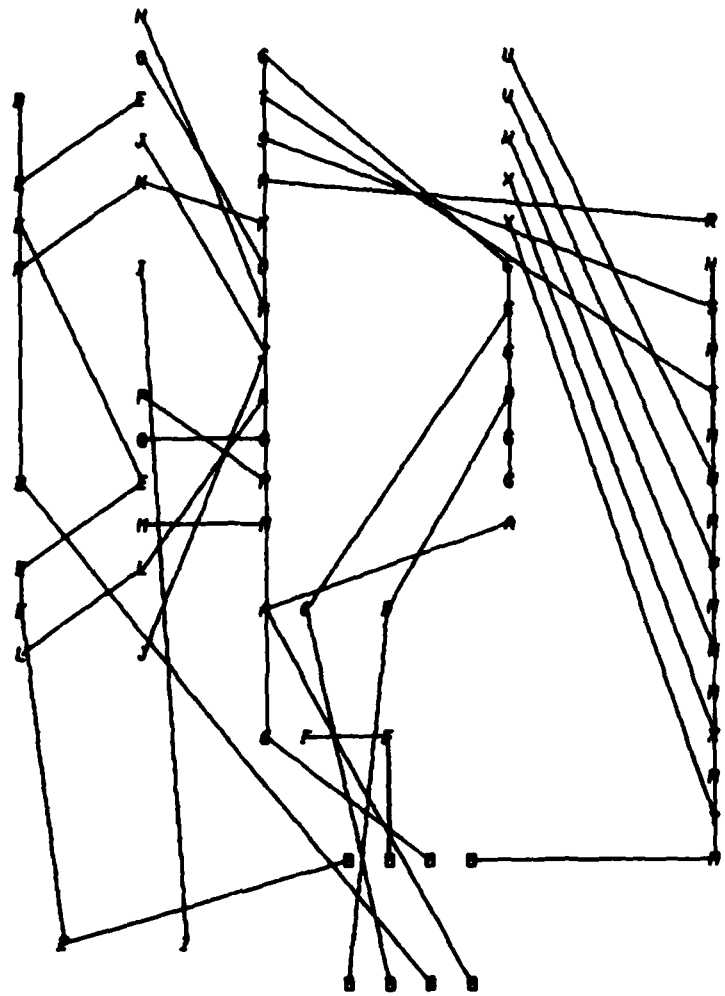
Wiring Session

```
      )LOAD WRAP1
SAVED 15:57 SEP 14,'81
      )COPY PWW1 TESTLOC TESTDEF
SAVED 14:03 SEP 29,'81
      START
TYPE OF BOARD?
MATRIX:1
SCANBE:2
SEC-905:3
TYPE?1
NAME OF THE LOCATION PROGRAM:
TESTLOC
CHIP 74LS93 IS NOT IN THE BANK...
 GEOMETRY:  DUAL IN LINE <<<<<  2PIN
                                6PIN
                                8PIN
                               14PIN
                               16PIN
                               18PIN
                               20PIN
                               22PIN
                               24PIN
                               28PIN
                               40PIN
            CONNECTOR  <<<<<C26PIN
                           C10PIN
                           C40PIN
                           C50PIN
            DISPLAY    <<<<<<<AFF8
   DISPLAY LCD 40PIN X 1.3IN<<<<LCD1
   DISPLAY MAN6660,80 10PIN<<<MAN66X
GEOMETRY?   14PIN

IDENTIFICATION OF PIN 1?   INB
IDENTIFICATION OF PIN 2?   RO(1)
IDENTIFICATION OF PIN 3?   RO(2)
IDENTIFICATION OF PIN 4?   NC
IDENTIFICATION OF PIN 5?   VCC
IDENTIFICATION OF PIN 6?   NC
IDENTIFICATION OF PIN 7?   NC
IDENTIFICATION OF PIN 8?   QC
IDENTIFICATION OF PIN 9?   QB
IDENTIFICATION OF PIN10?   GND
IDENTIFICATION OF PIN11?   QD
IDENTIFICATION OF PIN12?   QA
IDENTIFICATION OF PIN13?   NC
IDENTIFICATION OF PIN14?   INA
```

```
NO ERROR IN THE LOCATION PROGRAM
NAME OF THE DEFINITION PROGRAM:
TESTDEF
NO ERROR IN THE DEFINITION PROGRAM

TO OBTAIN A DRAWING OF THE BOARD,TYPE 'BOARD' .
THE CIRCUIT LAYOUT CALCULATION IS OBTAINED BY:
)LOAD WRAP2, AND EXECUTING 'START'

BOARD
TERMINAL? (2-4815; 3-4662)
0:
      2
```



```
)LOAD MRAP2
SAVED 15:08 SEP 14,'81
      START
TERMINAL? (2=4815; 3=4662)
0:
      2


LINES WITH COMMON POINTS:

28  25  8
29  15
30  26  10
31  27  11
32  28
33  13
34  14  9
35  24
22  23

GRAPHICS? 0 OU 1
0:
      1
COMPLETE(0) OR BY SEGMENT(1)?
0:
      0


LAYOUT COMPLETED
TYPE 1 FOR TABLE
0:
      1




NUMBER          BIN          LENGTH

  22             1           2.8
   7             2           3.1
   2             3           3.2
   8             4           3.4
   3             5           3.6
   4             6           3.8
   2             7           4
   3            10           4.2


TABLE COMPLETED
COMPLETED
TO OBTAIN CHIP DRAWINGS AND A LIST OF UNUSED PINS, TYPE:
)LOAD MRAP3
AND EXECUTE: SETDRAWING
```



```
)LOAD WRAP3
SAVED 09:52 JUN 22,'81
      SETDRAWING
COMPLETED
THE FUNCTION 'FREEPIN'LISTS ALL UNUSED PINS OF THE CIRCUIT
THE FUNCTION 'DRAW X'DRAWS THE CHIP NUMBER X.
TO WIREWRAP A CIRCUIT.CONTACT M. LESSARD AT 4315.
      FREEPIN
THE FOLLOWING PINS ARE NOT USED:
(001)AT04  PIN  004   NC
(001)AT02  PIN  006   NC
(001)AT01  PIN  007   NC
(001)AA06  PIN  013   NC
(002)AT13  PIN  004   NC
(002)AT11  PIN  006   NC
(002)AT10  PIN  007   NC
(002)AA15  PIN  013   NC
(004)AC22  PIN  017   P33
(004)AC23  PIN  018   P35
(004)AC24  PIN  019   P37
(004)AC25  PIN  020   P39
(004)AD25  PIN  021   P40
(004)AD24  PIN  022   P38
(004)AD23  PIN  023   P36
(004)AD22  PIN  024   P34
(004)AD21  PIN  025   P32
(004)AD20  PIN  026   P30
(004)AD19  PIN  027   P28
(004)AD18  PIN  028   P26
(004)AD17  PIN  029   P24
(004)AD16  PIN  030   P22
(004)AD15  PIN  031   P20
(004)AD14  PIN  032   P18
(004)AD13  PIN  033   P16
(004)AD12  PIN  034   P14
(004)AD11  PIN  035   P12
(004)AD10  PIN  036   P10
(004)AD09  PIN  037   P8
(004)AD08  PIN  038   P6
(004)AD07  PIN  039   P4
(004)AD06  PIN  040   P2
(005)AL15  PIN  003   3A
(005)AJ15  PIN  005   5A
(005)AI15  PIN  006   6A
(005)AH15  PIN  007   7A
(005)AH10  PIN  008   7B
(005)AI10  PIN  009   6B
(005)AJ10  PIN  010   5B
(005)AL10  PIN  012   3B
```

DREV R-4245/82 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"Computer-Aided Design and Fabrication of Wire-Wrap® Type Circuit Boards: A New Symbolism and its Implementation"
by B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard and A. Blanchard

We define here an encoding symbolism that permits, with a minimum of statements, a thorough description of the numerous interconnections of complex electronic circuitry. This symbolism has been integrated at the Defence Research Establishment Valcartier in a computer-aided method that considerably eases the passage between an engineering electronic schematic and the related interconnection matrix required for Wire-Wrap® hardware. Electronics prototyping has been dramatically speeded up with this technique because of the time savings in the preparatory reduction of interconnection data. (U)

DREV R-4245/82 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"Computer-Aided Design and Fabrication of Wire-Wrap® Type Circuit Boards: A New Symbolism and its Implementation"
by B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard and A. Blanchard

We define here an encoding symbolism that permits, with a minimum of statements, a thorough description of the numerous interconnections of complex electronic circuitry. This symbolism has been integrated at the Defence Research Establishment Valcartier in a computer-aided method that considerably eases the passage between an engineering electronic schematic and the related interconnection matrix required for Wire-Wrap® hardware. Electronics prototyping has been dramatically speeded up with this technique because of the time savings in the preparatory reduction of interconnection data. (U)

DREV R-4245/82 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"Computer-Aided Design and Fabrication of Wire-Wrap® Type Circuit Boards: A New Symbolism and its Implementation"
by B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard and A. Blanchard

We define here an encoding symbolism that permits, with a minimum of statements, a thorough description of the numerous interconnections of complex electronic circuitry. This symbolism has been integrated at the Defence Research Establishment Valcartier in a computer-aided method that considerably eases the passage between an engineering electronic schematic and the related interconnection matrix required for Wire-Wrap® hardware. Electronics prototyping has been dramatically speeded up with this technique because of the time savings in the preparatory reduction of interconnection data. (U)

DREV R-4245/82 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"Computer-Aided Design and Fabrication of Wire-Wrap® Type Circuit Boards: A New Symbolism and its Implementation"
by B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard and A. Blanchard

We define here an encoding symbolism that permits, with a minimum of statements, a thorough description of the numerous interconnections of complex electronic circuitry. This symbolism has been integrated at the Defence Research Establishment Valcartier in a computer-aided method that considerably eases the passage between an engineering electronic schematic and the related interconnection matrix required for Wire-Wrap® hardware. Electronics prototyping has been dramatically speeded up with this technique because of the time savings in the preparatory reduction of interconnection data. (U)

CRDV R-4245/82 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. GOA 1RO

"Conception et fabrication automatisées de circuits par câblage enroulé: un nouveau symbolisme et son application"
par B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard et A. Blanchard

Nous définissons un symbolisme de codage permettant, à l'aide d'un nombre minimum d'énoncés, la description complète des nombreuses interconnexions de circuits électroniques complexes. Ce symbolisme est devenu, au Centre de Recherches pour la Défense, Valcartier, partie intégrante d'une méthode informatisée facilitant la transition entre les plans d'ingénierie électronique et la matrice pertinente d'interconnexions requise pour le montage par câblage enroulé Wire-Wrap®. Le développement de prototypes électroniques s'est vu considérablement accéléré par la préparation plus rapide des données d'interconnexions. (NC)

CRDV R-4245/82 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. GOA 1RO

"Conception et fabrication automatisées de circuits par câblage enroulé: un nouveau symbolisme et son application"
par B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard et A. Blanchard

Nous définissons un symbolisme de codage permettant, à l'aide d'un nombre minimum d'énoncés, la description complète des nombreuses interconnexions de circuits électroniques complexes. Ce symbolisme est devenu, au Centre de Recherches pour la Défense, Valcartier, partie intégrante d'une méthode informatisée facilitant la transition entre les plans d'ingénierie électronique et la matrice pertinente d'interconnexions requise pour le montage par câblage enroulé Wire-Wrap®. Le développement de prototypes électroniques s'est vu considérablement accéléré par la préparation plus rapide des données d'interconnexions. (NC)

CRDV R-4245/82 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. GOA 1RO

"Conception et fabrication automatisées de circuits par câblage enroulé: un nouveau symbolisme et son application"
par B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard et A. Blanchard

Nous définissons un symbolisme de codage permettant, à l'aide d'un nombre minimum d'énoncés, la description complète des nombreuses interconnexions de circuits électroniques complexes. Ce symbolisme est devenu, au Centre de Recherches pour la Défense, Valcartier, partie intégrante d'une méthode informatisée facilitant la transition entre les plans d'ingénierie électronique et la matrice pertinente d'interconnexions requise pour le montage par câblage enroulé Wire-Wrap®. Le développement de prototypes électroniques s'est vu considérablement accéléré par la préparation plus rapide des données d'interconnexions. (NC)

CRDV R-4245/82 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. GOA 1RO

"Conception et fabrication automatisées de circuits par câblage enroulé: un nouveau symbolisme et son application"
par B. Montminy, R. Carbonneau, A. Laflamme, M. Lessard et A. Blanchard

Nous définissons un symbolisme de codage permettant, à l'aide d'un nombre minimum d'énoncés, la description complète des nombreuses interconnexions de circuits électroniques complexes. Ce symbolisme est devenu, au Centre de Recherches pour la Défense, Valcartier, partie intégrante d'une méthode informatisée facilitant la transition entre les plans d'ingénierie électronique et la matrice pertinente d'interconnexions requise pour le montage par câblage enroulé Wire-Wrap®. Le développement de prototypes électroniques s'est vu considérablement accéléré par la préparation plus rapide des données d'interconnexions. (NC)