Date	January 31, 2003	Court	Tokyo District Court 29th Civil Division
Case number	2001 (Wa) 17306		
– A case in which the court explained the criteria for determining whether a computer program is creative and whether a computer program is identical with another computer program. – A case in which the court denied the creativity of the plaintiff's programs used to draw the design of the contact wire for trains and also denied the similarity between the plaintiff's programs and the defendant's programs.			

References: Article 2, paragraph (1), Article 10, paragraph (1), item (ix), and paragraph (3), Article 21, Article 26-2, and Article 27 of the Copyright Act

Summary of the Judgment

1. The plaintiff alleged against the defendant that the defendant's acts of manufacturing and selling the defendant's products, which are loaded with the defendant's programs, infringe the plaintiff's copyrights (right of reproduction, adaptation right and right of transfer) for the plaintiff's programs, and based on such allegation, the plaintiff sought an injunction against the abovementioned defendant's acts as well as the payment of damages, etc.

Both the plaintiff's programs and the defendant's programs are programs used to draw the design of the contact wire (e.g. contact wire used to supply power electricity to electric trains), and they have been delivered to railway companies.

2. With respect to programs, the symbols for expression are limited and the language systems are rigid due to their nature. Moreover, the selection of the combination of instructions will be limited when intending to have the computer function economically and efficiently as much as possible. Thus, the specific descriptions of programs often become similar to each other. If the specific descriptions of programs are covered by the protection under the Copyright Act even if such descriptions are those that would be almost the same no matter who prepares them, those wherein simple contents are described by a very short notation or those that are extremely ordinary, the widespread use, etc. of computers will be hindered and significant problems will be posed to social lives and economic activities. In addition, the Copyright Act protects the specific expression of programs but not the functions or ideas thereof. Thus, if the specific descriptions of a program which performs a specific function are completely ordinary but are covered by the protection under the Copyright Act, this will result in protecting and monopolizing the function or idea per se. Accordingly, when the specific representation of a program, which is a combination of

instructions for the computer, consists of such descriptions, such specific representation should be found to lack creativity as the creator's individuality is not exhibited therein.

3. If the existence of the characteristics mentioned above in the program representation is to be taken into consideration, the determination on the identicalness of the programs should be made based on the standpoint of whether or not the programs are substantially identical by comparing the parts which are found to be creative among the specific descriptions of the programs or whether or not the creative characteristic parts can be directly perceived and not by merely finding whether or not the overall process and structure of the programs are similar.

4. The description of the shape definition (designation of the shape of a special character with a coordinate value) contained in the plaintiff's programs can be regarded as a program if it functions in cooperation with other programs that read it. However, since the creator has fewer options for describing the shape definition, the description of the plaintiff's shape definition is not creative. Even if it is possible to find that the description of the plaintiff's shape definition is creative, it differs from the specific description of the defendant's shape definition in terms of the coordinate values, etc. and therefore they cannot be deemed to be identical with each other, nor can the essential characteristic part of the description of the plaintiff's shape definition which has creativity be directly perceived from the description of the defendant's shape definition. Other descriptions of the plaintiff's programs also cannot be deemed to be creative or do not have similarity to the descriptions of the defendant's programs.

Judgment rendered on January 31,2003
2001 (Wa) 17306, Case of Seeking Injunction Against Infringement of Copyright, etc.
Date of conclusion of oral argument: October 2, 2002

Judgment

Plaintiff: YBM Co., Ltd.
Defendant: SATORI ELECTRIC CO., LTD.

Main text

1. All of the plaintiff's claims shall be dismissed.
2. The court costs shall be borne by the plaintiff.

Facts and reasons

No. 1 Claims
1. The defendant shall not manufacture, use, sell, distribute or export the products stated in the attached list of the defendant's products.
2. The defendant shall destruct the products stated in the preceding paragraph.
3. The defendant shall pay to the plaintiff 40,000,000 yen and money accrued thereon at the rate of 5% per annum for the period from September 4, 2001, until the date of completion of the payment.
No. 2 Outline of the case

The plaintiff alleged against the defendant that the defendant's acts of manufacturing and selling the products stated in the attached list of the defendant's products (hereinafter referred to as the "Defendant's Products"), which are loaded with the programs stated in the attached list of the defendant's programs (hereinafter collectively referred to as the "Defendant's Programs"), infringe the copyrights (right of reproduction, adaptation right and right of transfer) held by the plaintiff for the programs stated in the list of the plaintiff's programs (hereinafter collectively referred to as the "Plaintiff's Programs), and based on such allegation, sought an injunction against the abovementioned defendant's acts as well as the payment of damages, etc.
1. Facts on which the decision is premised (the evidence, etc. has been noted at the end of the sentences)
(1) The Plaintiff's Programs
A. Around September 1989, Yoshizawa Business Machines Kabushiki Kaisha (hereinafter referred to as "Yoshizawa Business Machines") developed the program stated in item (1) of the attached list of the plaintiff's programs that operates with MS-DOS3.1 and corresponds to the AutoCAD GX-III version (hereinafter referred to as "Plaintiff's Program 1") and started to

deliver the products in which said program is reproduced and stored to its clients including the Chiba branch of the East Japan Railway Company (hereinafter referred to as "JR-EAST") (Exhibit Ko 14 and the entire import of the oral argument).
A. Around November 1996, Yoshizawa Business Machines upgraded Plaintiff's Program 1 and developed the program stated in item (2) of the attached list of the plaintiff's programs that operates on Windows and corresponds to the AutoCAD R13J version (hereinafter referred to as "Plaintiff's Program 2") and started to deliver the products in which said program is reproduced and stored to its clients including the Akita branch of JR-EAST (the entire import of the oral argument).
C. On November 1, 2000, the plaintiff received the whole business including the development, sales and maintenance service of the Plaintiff's Programs from Yoshizawa Business Machines at 200,000,000 yen and also received the copyright of the Plaintiff's Programs and the right to claim damages that had emerged by that day due to infringement of the copyright (Exhibits Ko 11 and 12).
(2) The Defendant's Programs
A. Around March 1997, the defendant developed the program stated in item (1) of the attached list of the defendant's programs that operates on Windows and corresponds to the AutoCAD R13J version (hereinafter referred to as "Defendant's Program 1") and delivered the products in which said program is reproduced and stored to its clients including the Morioka branch of JR-EAST (Exhibit Otsu 3 and the entire import of the oral argument).
B. Around October 1998, the defendant upgraded Defendant's Program 1 and developed the program stated in item (2) of the attached list of the defendant's programs that corresponds to the AutoCAD R14 version (hereinafter referred to as "Defendant's Program 2"). Around October 2001, the defendant further upgraded Defendant's Program 2 and developed the program stated in item (3) of the attached list of the defendant's programs that corresponds to the AutoCAD 2000i version (hereinafter referred to as "Defendant's Program 3") and delivered the products in which the abovementioned programs are reproduced and stored to its clients including the Morioka branch of JR-EAST (Exhibit Otsu 3 and the entire import of the oral argument).
(3) Contents of the Plaintiff's Programs
A. The Plaintiff's Programs are computer-aided drafting and design software programs that operate on AutoCAD and create drawings for railway electrical design and equipment management. AutoCAD refers to a generalized CAD system (a platform that conducts creation, amendment, deletion, display and printing, etc. of two-dimensional or three-dimensional drawings) made by Autodesk Inc. that operates on an operating system (MS-DOS or Windows).

The part with respect to which the plaintiff alleges infringement of the right of reproduction is the part of the Plaintiff's Programs stated in B. and C. below.

B. The Plaintiff's "Contact Line - Base Line Creation Program"

(A) The plaintiff's "Contact Line - Base Line Creation Program" is part of the Plaintiff's Programs and is a program that draws the vertical and upper and lower base lines to aid the construction of the drawing for design of the contact line (contact wire, etc. used to supply power electricity to electric locomotive and trains) in accordance with the data input by users.

(B) The plaintiff's "Contact Line - Base Line Creation Program" is described for each function and is divided into the following five files (Exhibits Ko 17 and 27).

a. Main part: YBJ-TR68.lsp file (Attachment 1)

b. Input part: YBJ-TQ02.lsp file (Attachment 2)

c. Modification part: YBJ-TR80.lsp file

d. Drawing part: YBJ-TR79.lsp file (Attachment 3)

e. Explanation part: YBJ-TR78.lsp file

(C) The plaintiff's "Contact Line - Base Line Creation Program" is described in AutoLISP language (an interpreter programming language; the description in said language requires no compilation). The Plaintiff's Programs are reproduced and stored in the hard disk of the plaintiff's products in an encrypted source program form and there is no object program for the Plaintiff's Programs (the entire import of the oral argument).

C. Description regarding the shape definition

In the hard disk of the products in which the Plaintiff's Programs are reproduced and stored, a shape file (a binary data file with shx extension) concerning numerous special characters (font) and special graphics (shape) is stored (Exhibits Ko 19 and 24). A shape file is a file generated by translating (compiling) a shape definition file (a file with shp extension) into machine language by the command of AutoCAD. The shape definition file is described in accordance with the shape definition statements of AutoCAD (Exhibit Otsu 2).

(4) Contents of the Defendant's Programs

A. The Defendant's Programs are also computer-aided drafting and design software programs that operate on AutoCAD and create drawings for railway electrical design and equipment management.

B. The defendant's "Contact Line - Base Line Creation Program"

The defendant's "Contact Line - Base Line Creation Program" also constitutes part of the Defendant's Programs and is a program that draws the vertical and upper and lower base lines to aid the construction of the drawing for design of the contact line (contact wire, etc. used to supply power electricity to electric locomotive and trains) in accordance with the data input by users.

The defendant's "Contact Line - Base Line Program" is, as shown in Attachment 4, described in AutoLISP language on BASELINE.lsp file (Exhibit Ko 18).

C. Description regarding the shape definition

In the hard disk of the products in which the Defendant's Programs are reproduced and stored, a shape file (a binary data file with shx extension) concerning special characters (font) and special graphics (shape) is stored (Exhibits Otsu 1). The shape definition file is described in accordance with the shape definition statements of AutoCAD.

2. Issues

(1) Whether or not the Defendant's Programs are reproductions or adaptations of the Plaintiff's Programs (whether or not the defendant's acts of manufacturing, selling, or otherwise handling the Defendant's Products constitute infringement of the right of reproduction, adaptation right and right of transfer for the Plaintiff's Programs).

(2) Whether or not the defendant has reproduced and stored Plaintiff's Program 1 in the storage media in its computer in creating the Defendant's Programs.

(3) The amount of damages sustained by the plaintiff.

(omitted)

No. 3 Court decision

1. Regarding issue 1 (Whether or not the Defendant's Programs are reproductions or adaptations of the Plaintiff's Programs)

(1) Regarding the determination on the creativity and identicalness of the program

In order to find that a representation falls under the work covered by the protection under the Copyright Act (hereinafter sometimes referred to as the "Act"), it must be a production in which thoughts or sentiments are creatively expressed. In addition, in order to find that the thoughts or sentiments are creatively expressed, some kind of individuality of the creator must be exhibited in such expression although originality in a strict sense is not required.

This is no different in the case of a production of expression made in the form of programs (something expressed as a set of instructions written for a computer, which makes the computer function so that a specific result can be obtained). If the creator's individuality is expressed in the specific descriptions, programs will be protected by the Copyright Act as works.

With respect to programs, the symbols for expression are limited and the language systems are rigid due to their nature. Moreover, the selection of the combination of instructions will be limited when intending to have the computer function economically and efficiently as much as possible. Thus, the specific descriptions of programs often become similar to each other. If the specific descriptions of programs are covered by the protection under the Copyright Act even if such descriptions are those that would be almost the same no matter who prepares them, those wherein simple contents are described by a very short notation or those that are extremely ordinary, the widespread use, etc. of computers will be hindered and significant problems will be posed to social lives and economic activities. In addition, the Copyright Act protects the specific expression of programs but not the functions or ideas thereof. Thus, if the specific descriptions of a program which performs a specific function are extremely ordinary but are covered by the protection under the Copyright Act, the function or idea per se will be protected and monopolized. Accordingly, when the specific representation of a program, which is a combination of instructions for the computer, consists of such descriptions, such specific expression should be found to lack creativity as the creator's individuality is not exhibited.

Moreover, if the existence of the characteristics mentioned above in the program representation is to be taken into consideration, the determination on the identicalness of the programs should be made based on the standpoint of whether or not the programs are substantially identical by comparing the parts which are found to be creative among the specific descriptions of the programs or whether or not the creative characteristic parts can be directly perceived and not by merely finding whether or not the overall process and structure of the programs are similar.

By comprehensively taking into consideration the points mentioned above, this court will examine whether or not the Plaintiff's Programs are creative and compare them with the Defendant's Programs.

(2) Regarding the Contact Line - Base Line Creation Program

A. Menu display part

(A) Contents of the Plaintiff's Programs

The contents of the menu display part of the Contact Line - Base Line Creation Program contained in Plaintiff's Program 1 are as follows (Exhibits Ko 17 and 25).

The main program of the Contact Line - Base Line Creation Program contained in Plaintiff's Program 1 displays on the screen a menu list consisting of "1. Creation of data files", "2. Modification of data files", "3. Creation of base lines", "4. Grammar explanation for data files" and "0. End" and has the function of calling up (loading) the files that execute the functions in accordance with the menu number input by the user.

The descriptions of the program are as stated in Attachment 1 and consist of 28 lines (from line 1 to line 28) in total (there are blank lines; From line 1 to line 4 are notes on the contents and date of creation of the program).

a. Line 6 contains a description "(setq B1 0)" which instructs to initialize variable B1. The setq function is a basic assignment function in AutoLISP language.

b. In line 8 to line 20, mainly the syntaxes, "(princ "¥n[menu name]" and "(princ "¥n")" are repeated six times and the following menus are sequentially displayed in the part of "[Menu name] ": "<<JR-CAD>>[Creation of contact line km route and base line]", "1. Creation of data files", "2. Modification of data files", "3. Creation of base lines (after the creation of data files)", "4. Grammar explanation for data files" and "0. End". The princ function instructs to display on the screen the descriptions that are stated after the term "princ" and enclosed in the double quotation marks without any change. "¥n" is a linefeed code in the MS-DOS version of AutoCAD.

c. Line 21 contains a description "(setq B0 (getint "¥n Input the target number<0>:"))(if (=B0 nil)(setq B0 0))" which instructs to set the integer value (menu number) input by the user into variable B0 after displaying a message which reads "Input the target number <0>" on the screen.

d. In line 23 to line 27, the syntax "(if (=B0[Menu number])(load "[File name]"))" is repeated four times, and at the end, it is described "if (=B0 0)(setq B1 1))". In the part [File name], the file names that sequentially execute the data input part, modification part, drawing part and explanation part are described. This part instructs to open any of the files of the data input part, modification part, drawing part and explanation part in the AutoCAD according to the menu number input by the user and to set 1 into variable B1 and end the processing when the user selects the menu number "0" (end).

(B) Creativity

The program description of the menu display part in the Plaintiff's Programs is short on the whole and most of this part is a mere combination of simple instructions using general functions defined by AutoLISP language. Thus, the Plaintiff's Programs cannot be found to be an expression in which the creator's individuality is exhibited, and are not creative.

The flow of processing in the menu display part in the Plaintiff's Programs is as follows: [i] the menu messages are displayed in the order of data creation (input), modification, drawing, explanation and end, on the screen; [ii] the user is required to select (input) a menu number; and [iii] the file which executes the function is called up according to the menu number input by the user. This flow should be found as falling under the "algorithm" provided for in Article 10, paragraph (3), item (iii) of the Act and thus is not protected by copyright.

As described above, the menu display part in the Plaintiff's Programs is not creative.

B. Data input part for the base line

(A) Input part for the first value of the km route

a. Contents of the Plaintiff's Programs

The principal program description of the input part for the first value of the km route in the Plaintiff's Programs (the description alleged by the plaintiff as being identical with the corresponding part of the Defendant's Programs; the same shall apply in (B) through (E) below) consists of one line, "(setq V0 (getreal"¥nInput the first value of the ●km route in meters<0>:"))" (line 13 of Attachment 2). This description instructs to display on the screen a message instructing to input the first value of the km route and to set the input real number into variable V0 (Exhibits Ko 17 and 25).

The plaintiff's description part can be written in a syntax, "setq V0 (getreal"Message"))". A getreal function is used to have the real value input by AutoLISP language and the character string enclosed by the double quotation marks described after the function will be displayed on the screen without any change. ¥n is a code meaning a linefeed.

b. Creativity

The description of the input part for the first value of the km route in the Plaintiff's Programs expresses by a very short syntax the extremely easy content of displaying the character string of "input the first figure of the ●km route in meters<0>:" on the screen and then setting the real value input by the user into a variable by using the general functions of AutoLISP language.

Therefore, the description of the input part for the first value of the km route cannot be found to be an expression in which the creator's individuality is exhibited, and is not creative.

(B) Input part for the offset value of the km route

a. Contents of the Plaintiff's Programs

The principal program description of the input part for the offset value of the km route in the Plaintiff's Programs consists of two lines, "(princ"¥n¥n¥nInput the offset value of the ●km route" and "(setq V1(getreal"(the distance from the start value to the first mark) in meters<0>:"))" (line 17 and line 18 of Attachment 2). These descriptions instruct to display on the screen a message instructing the user to input the offset value of the ●km route and to set the input real number into variable V1 (Exhibits Ko 17 and 25).

The plaintiff's description part can be written into a syntax, "setq V0 (getreal"Message"))". The message to be displayed on the screen is not stated in full in the part enclosed by the double quotations following the getreal function and only part of the message is displayed on the screen using the princ function contained in the previous line.

b. Creativity

The description of the input part for the offset value of the km route in the Plaintiff's Programs expresses by a very short syntax the extremely easy content of displaying the character string of "input the offset value (the distance from the first value to the first mark) of the ●km route in meters <0>:" on the screen and then setting the real value input by the user into variables by using the general functions of AutoLISP language.

Therefore, the description of the input part for the offset value of the km route cannot be found to be an expression in which the creator's individuality is exhibited, and is not creative.

(C) Input part for the scale

a. Contents of the Plaintiff's Programs

The principal program description of the input part for the scale of the Plaintiff's Programs consists of one line, "(setq V2(getreal "¥n¥n¥nInput only the denominator of the ●scale (e.g. 500 in the case of 1/500)<500>:"))" (line 22 of Attachment 2). This description instructs to display on the screen a message instructing to input the denominator of the scale which reads "Input only the denominator of the ●scale (e.g. 500 in the case of 1/500)<500>:" and to set the input real number into variable V2 (Exhibits Ko 17 and 25).

The plaintiff's description part can be written in a syntax, "setq V0 (getreal"Message"))", which is the same description used in the "input part for the first value of the km route" and "input part for the offset value of the km route" mentioned above.

b. Creativity

The principal program description of the input part for the scale in the Plaintiff's Programs cannot be found to be an expression in which the creator's individuality is exhibited, and is not creative, as with the case of the descriptions of the input part for the first value of the km route and input part for the offset value of the km route.

(D) Input part for the paper size

a. Contents of the Plaintiff's Programs

The principal program description of the input part for the paper size in the Plaintiff's Programs consists of two lines, "(princ"¥n¥n¥nInput ●km paper size A4 (210mm high), A3 (297mm high)")" and (setq V3 (getstring"¥n or A2 (420mm high)<A4>:"))" (line 26 and line 27 of Attachment 2). This description instructs to display on the screen a message instructing to input the paper size which reads "input ●km paper size A4(201mm high), A3(297mm high) or A2(420mm high)" and to set the input character string into variable V3 (Exhibits Ko 17 and 25).

The plaintiff's description part can be written in a syntax, "setq V3 (getstring"Message"))". With respect to the paper size, the data to be input is a character string such as "A4" and "A3", and thus the getstring function, which is one of the functions of AutoLISP language corresponding to the input of character string, is used. In addition, part of the message is displayed on the screen using the princ function contained in the previous line.

b. Creativity

The description of the input part for the paper size in the Plaintiff's Programs expresses in a very short syntax the extremely easy content of displaying the character string of "Input ●km paper size A4 (210mm high), A3 (297mm high) or A2 (420mm high) <A4>:" on the screen and then setting the character string input by the user into variables by using the function normally used for inputting a character string in AutoLISP language.

Therefore, the description of the plaintiff's input part for the paper size cannot be found to an expression in which the creator's individuality is exhibited, and is not creative.

(E) Input part for the span distance

a. Contents of the Plaintiff's Programs

The principal program description of the input part for the span (the distance between the vertical base lines) consists of one line, "setq V0 (getstring "¥n¥n input the span<50>")) (line 36 of Attachment 2). This description instructs to display on the screen a message instructing to input the span which reads "input the span<50>" and to set the input character string into variable V0 (Exhibits Ko 17 and 25).

The plaintiff's description part can be written in a syntax, "setq V0 (getstring"Message"))", as with the case of the description of the input part for the paper size. With respect to the span, the data to be input is not limited to the span (real value) but includes the character string which instructs to draw auxiliary lines on either side of the span ("L" or "R") and thus, the getstring function corresponding to the input of character string is used.

b. Creativity

The description of the input part for the span of the Plaintiff's Programs expresses by a very short syntax the extremely easy content of displaying the character string of "Input the span<50>" on the screen and then setting the character string input by the user into variables by using the function normally used for inputting a character string in AutoLISP language.

Therefore, the description of the input part for the span distance cannot be found to an expression in which the creator's individuality is exhibited, and is not creative.

C. Drawing part of the base line

(A) Initializing part

a. Contents of the Plaintiff's Programs and the Defendant's Programs

(a) In the initializing part of the Plaintiff's Programs, the base lines are drawn based on the data input by the user in the input part and thus, the initializing part has the role of reading the data written in the data file in the input part and setting it into variables. The description of the initializing part of the Plaintiff's Programs consists of 32 lines (from line 22 to line 53) among the descriptions of the drawing part file (Attachment 3) and the structure thereof is as follows.

With respect to the "first value of the km route", processing is conducted in the following three steps: [i] by using the syntax,"(setq [variable] (read-line F1))", the data written in the data file is read from the beginning for one line and set into variables; [ii] by using rtos function and atof function (functions to translate character strings into real numbers), the variables mentioned in [i] above are transformed into real values or character strings that can be used in drawing figures and the results thereof are set again into new variables by the setq function; and [iii] the setting results of the new variables are displayed on the screen by the princ function. Following this, the processing consisting of the three steps mentioned above is repeated for each input item, "offset value of the km route", "scale" and "paper size".

In the initializing part of the Plaintiff's Programs, the order of setting variables is the same as the order of reading the data in a file in the input part, i.e. the order of "first value of the km route", "offset value of the km route", "scale" and "paper size". This is because, in the Plaintiff's Programs, the input data is sequentially written in the data file in the input order by using the syntax, "(write-line[variable]F0", in the input part (line 30 to line 34 of the input part file) and the read-line function used for reading the data in the initializing part can only read the data from the beginning in the order of being written in the file (Exhibits Ko 17 and 25).

(b) In contrast, the part of the Defendant's Programs corresponding to the initializing part mentioned above consists of eight lines (from line 284 to line 291) in the defendant's Contact Line - Base Line Creation File (Attachment 4) (described within the definition of the local function such as DrawBaseLine function) and is structured as follows.

First, with respect to the "first value of the km route", processing consisting of the following three steps is conducted: [i] necessary data is called up from among the list of IsBaseLineData (variables with multiple values); [ii] the data is evaluated by atof function and fix function (function that conducts conversion to an integer by rounding down the fraction portion), etc. without any change; and [iii] the evaluated numerical value is set into variables by the setq function. The processing consisting of the steps mentioned in [i] through [iii] above is sequentially conducted in the order of "offset value of the km route", "scale" and "paper size".

The specific description of the defendant's processing mentioned above principally consists of the single-line syntax which reads "(setq[variable] ([function (atof function, etc.)](nth[order in the list]IsBaseLineData))" and this syntax is repeated for each input item (variable). The part "(nth[order within the list]IsBaseLineData)" is an instruction to read the data in the contents of the list and can read the data not only from the beginning of the data file like the Read-line function but also the data in the position designated by the [order in the list] in the data file. In the defendant's Contact Line - Base Line Program, the data in the IsBaseLineData is retained in the order of scale, paper size, first value of the km route, offset value of the km route and span distance (Exhibits Ko 18, 25 and 27).

b. Creativity of the Plaintiff's Programs and comparison of the Plaintiff's Programs and the Defendant's Programs

(a) What is used as the input item in the Plaintiff's Programs is an idea which is not covered by the protection under the Copyright Act. In addition, the flow of processing wherein values are set into variables in the order of "first value of the km route", "offset value of the km route", "scale" and "paper size" falls under the "algorithm" provided for in Article 10, paragraph (3), item (iii) of the Act and will not be protected as works.

(b) Even if it is possible to construe that the specific description of the initializing part in the Plaintiff's Programs may have creativity, the scope of creativity should be found to be extremely narrow in light of the contents of the Plaintiff's Programs found above. The Defendant's Programs and the Plaintiff's Programs substantially differ in their specific descriptions due to the difference in the syntax used in the initializing part. The specific description of the initializing part in the Defendant's Programs cannot be found to be substantially identical with the description of the initializing part in the Plaintiff's Programs nor can the essential characteristic part of the Plaintiff's Programs which has creativity be directly perceived from the specific description of the initializing part in the Defendant's Programs.

(c) Therefore, it cannot be found that the right of reproduction or adaptation right has been infringed with respect to the initializing part of the Plaintiff's Programs.

(B) Drawing part of the span line

a. Contents of the Plaintiff's Programs and the Defendant's Programs

(a) The drawing part of the span line has the role of drawing the vertical base lines, right and left auxiliary lines, numerical values of the spans and span line (central horizontal base line) based on the data on the span (distance between the vertical base lines) input by the user and written in the data file (the numerical value that represents the distance between the spans or "L" or "R" which represents the drawing of the right or left auxiliary lines).

The description of the drawing part of the span line in the Plaintiff's Programs consists of 47 lines (from line 59 to line 105) in the plaintiff's drawing part file and is structured as follows.

[i] After the processing such as the initialization of variables, drawing the figures of the base line on the extreme left of the screen is instructed; [ii] data concerning the span distance input in the data file is read for one line from the beginning of the file using the syntax, "(setq[variable](read-line F1))"; [iii] when the read data is not empty, it will be instructed to conduct the subsequent processing; [iv] if the read data is evaluated based on the instruction "(if(=(substr S0 1 1)"L")" as "L", it will be instructed to draw the left auxiliary line using the syntax, "command "line"(list X-coordinate 1, Y-coordinate 1)(list X-coordinate 1, Y-coordinate 2)"")" (X-coordinate 1 is a value obtained by offsetting -4.0 from the X-coordinate value of the previous base line and Y-coordinate values 1 and 2 are the Y-coordinate values of the points of both ends of the left end base line); [v] when the read data is "R", it will be instructed to similarly draw the right auxiliary line; [vi] when the read data is a real value (span distance), it will be instructed to draw a vertical base line in the position on the right for the span distance from the previous span position and to display the span distance; and [vii] after the abovementioned processing is completed, it will be instructed to further read the data from the data file and to set the variables, and if there is still any data, the processing stated in [iii] above will be repeated but if no data remains (the variables are empty), the processing will be ended (Exhibits Ko 17, 25 and 27).

(b) In contrast, the description of the span line drawing part in the Defendant's Programs consists of 34 lines (from line 329 to line 363) in the defendant's Contact Line - Base Line Creation File (Attachment 4) and is structured as follows.

[i] It will be instructed to draw the figures of the base line on the extreme left of the screen and the graphic information obtained as a result of the figure drawing will be renewed by using the putlayer function, which is one of the functions of AutoLISP language; [ii] the initial counter (variable) will be set to "4" and the fifth and subsequent data in the list (this is because, among the data input by the user and retained in the list, the fifth and subsequent data is related to the span) will be designated to be the processing object; [iii] it will be instructed to evaluate the overall length of the list and repeat the subsequent processing unless the counter reaches the length of the list; [iv] the data in the order of the counter (the first will be the fifth) will be read from the list and set into variables ("vltem") by the setq function; [v] if the variable is L as a result of the evaluation using the syntax, "(= (strcase vltem)", it will be instructed to draw the left auxiliary line; [vi] if the variable is R as a result of the same evaluation mentioned in [v] above, it will be instructed to draw a right auxiliary line; [vii] if the variable is a real number as a result of the evaluation using the syntax, "(<0.0(setq fSpan (atof vItem))" (numerical values will be transformed into real numbers), it will be instructed to draw the vertical base lines and numerical values of the spans; and [viii] finally, after adding one to the processing counter, the processing will return to the step stated in [iii] above and the steps thereafter will be repeated (Exhibits Ko 18, 25 and 27).
b. Creativity of the Plaintiff's Programs and comparison between the Plaintiff's Programs and the Defendant's Programs
(a) The flow of processing in the Plaintiff's Programs, wherein the read data is evaluated and drawn in the order of "L", "R" and "span", falls under the "algorithm" provided for in Article 10, paragraph (3), item (iii) of the Act and thus will not be protected as a work.
(b) Even if it is possible to construe that the specific description of the span drawing part in the Plaintiff's Programs may have creativity, the scope of creativity should be found to be extremely narrow in light of the contents of the Plaintiff's Programs found above. The Defendant's Programs and the Plaintiff's Programs differ in terms of the specific evaluation method of variables and the method and process to repeat the overall processing. The specific description of the span drawing part in the Defendant's Programs cannot be found to be substantially identical with the description of the span drawing part in the Plaintiff's Programs nor can the essential characteristic part of the Plaintiff's Programs which has creativity be directly perceived from the specific description of the span drawing part in the Defendant's Programs.
(c) Therefore, it cannot be found that the right of reproduction or adaptation right has been infringed with respect to the description of the span line drawing part in the Plaintiff's Programs.
(C) Drawing part of the upper and lower base lines in the drawings
a. Contents of the Plaintiff's Programs and the Defendant's Programs
(a) The drawing part of the upper and lower base lines has the function of drawing a total of

nine upper and lower base lines, i.e. three lines on the upper part of the screen and six lines on the lower part of the screen.

The description of the drawing part of the upper and lower base lines in the Plaintiff's Programs consists of 16 lines (from line 108 to line 123) in the plaintiff's drawing part file (Attachment 3). It is mainly structured by sequentially repeating nine times a syntax, "command "line" (list X-coordinate 1 Y-coordinate 1) (list X-coordinate 2 Y-coordinate 2)"")", for the base line in the lowermost part, base line in the uppermost part, the second lowest base line, and subsequently to the second top base line in ascending order. This description instructs to specify the starting point and ending point of each line by the coordinate values of X and Y and to draw a line between the two points (Exhibits Ko 17 and 25).

(b) In contrast, the description of the drawing part of the upper and lower base lines in the Defendant's Programs consists of 43 lines (from line 365 to line 407) in the defendant's Contact Line - Base Line Creation File (Attachment 4). The offset value of each base line (the distance between the base lines; the order of setting the variables is an ascending order starting from the base line in the lowermost part) and the value of half the height of the vertical line are set as variables in advance (line 293 to line 303). In addition, in the drawing part of the upper and lower base lines, the coordinate of the next base line (variable 1, variable 2) is obtained by repeating the instruction using the syntax, "command "Line" Variable 1 Variable 2 """ and adding the offset value (variable) to the coordinate of the base line drawn previously. By using this method, the upper and lower base lines are sequentially drawn; first, three base lines are drawn upward from the upper side of the central part of the screen (the third base line from the top) and then a total of six base lines are drawn downward from the lower side of the central part (the sixth base line from the bottom) (nine lines in total) (Exhibits Ko 18 and 25).

b. Creativity of the Plaintiff's Programs and comparison between the Plaintiff's Programs and the Defendant's Programs

(a) The flow of processing of setting the coordinate values of the upper and lower base lines in an ascending order from the lower base line to the upper base line and drawing them falls under the "algorithm" provided for in Article 10, paragraph (3), item (iii) of the Act and will not be protected as a work.

(b) Even if it is possible to construe that the specific description of the drawing part of the upper and lower base lines in the Plaintiff's Programs may have creativity, the scope of creativity should be found to be extremely narrow in light of the contents of the Plaintiff's Programs found above. The Defendant's Programs and the Plaintiff's Programs significantly differ in terms of the order of drawing the base lines as well as the description of the program concerning the method to calculate the coordinate values of the base lines. The specific description of the drawing part of the upper and lower base lines in the Defendant's Programs cannot be found to be substantially identical with the description of the drawing part of the upper and lower base lines in the Plaintiff's Programs nor can the essential characteristic part of the Plaintiff's Programs which has creativity be directly perceived from the description of the drawing part of the upper and lower base lines in the Defendant's Programs.

(c) Therefore, it cannot be found that the right of reproduction or adaptation right has been infringed with respect to the description of the drawing part of the upper and lower base lines in the Plaintiff's Programs.

(D) Drawing part of the km route

a. Contents of the Plaintiff's Programs and the Defendant's Programs

(a) The drawing part of the km route has the function of drawing a mark with respect to each 1km, 0.5km and 0.1km.

The drawing part of the km route of the Plaintiff's Programs consists of 36 lines (from line 129 to line 164) in the drawing part file (Attachment 3) and is structured as follows.

First, in order to set the mark at regular intervals (each 1km, 500m, or 100m): [i] the value of the km route that is currently processed will be divided by the numerical value of the regular intervals (for example, 1,000 for the 1km mark) and then the obtained result will be rounded down; the value so obtained will then further be multiplied by the same numerical value (1,000) and if the result of calculation agrees with the original value of the km route, the symbol of the 1km mark will be indicated (a graphic defined separately from the program will be read); [ii] when the result of calculation does not agree with the original value of the km route, it will be determined whether or not it falls under the case of indicating the 500m mark by the same method (provided that the numerical value to be used for division and multiplication is 500); [iii] when the result of calculation does not agree with the original value of the km route, it will be determined whether or not it falls under the case of indicating the 100m mark by the same method (provided that the numerical value to be used for division and multiplication is 100); and [iv] after ending the abovementioned processing, 0.1m will be added to the value of the km route that is currently processed and then the same processing will be repeated from the step mentioned in [i] above and the entire processing will end when the km route that is currently processed comes to agree with the "maximum km route" set as the variable in advance (when the km route reaches the right end of the screen) (Exhibits Ko 17 and 25).

(b) In contrast, the description of the drawing part of the km route in the Defendant's Programs consists of a total of 40 lines (from line 410 to line 449) in the defendant's Contact Line - Base Line Creation File (Attachment 4). It is structured wherein a rem function is used to determine the regular intervals (1km, 500m or 100m each) to establish a mark and uses a calculation method wherein the km route that is currently processed is divided by the numerical value of the regular intervals (for example, 1,000 for the 1km mark) and if the remainder is 0, the symbol of the 1km mark will be indicated (a graphic defined separately from the program will be read) (Exhibits Ko 18 and 25).

b. Creativity of the Plaintiff's Programs and the comparison between the Plaintiff's Programs and the Defendant's Programs

(a) The "algorithm" used in the Plaintiff's Programs will not be protected by copyright for programs pursuant to the provisions of Article 10, paragraph (3), item (iii) of the Act and the method of reading a graphic which has been defined (symbolized) separately from the program falls under the category of an idea and will not also be protected under the Copyright Act.

(b) Even if it is possible to construe that the specific description of the drawing part of the km route in the Plaintiff's Programs may have creativity, the scope of creativity should be found to be extremely narrow in light of the contents of the Plaintiff's Programs found above. The Defendant's Programs and the Plaintiff's Programs significantly differ in terms of the description of the program due to the difference in the method to evaluate the values of the km route. The specific description of the km route in the Defendant's Programs cannot be found to be substantially identical with the description of the km route of the Plaintiff's Programs nor can the essential characteristic part of the Plaintiff's Programs which has creativity be directly perceived from the specific description of the km route in the Defendant's Programs.

(c) Therefore, it cannot be found that the right of reproduction or adaptation right has been infringed with respect to the description of the km route of the Plaintiff's Programs.

D. Based on the abovementioned findings, the plaintiff's allegations claiming that the defendant's "Contact Line - Base Line Program" is a reproduction or adaptation of the plaintiff's "Contact Line - Base Line Program" are groundless.

(3) Regarding the description related to the shape definition

A. Special character data

According to the evidence (Exhibits Ko 6 to 8, 19, 24 and 33 and Exhibits Otsu 1, 2, 4 and 8 to 12), the following facts can be found.

(A) Outline of the AutoCAD character font, etc.

a. AutoCAD character font

(a) The AutoCAD character font can be largely categorized into three types, i.e. half size font, Big Font and TrueType font. The Big Font is a shape definition file in a special form to represent non ASCII characters such as Kanji characters. Generally, every character handled by computers is assigned with a "character code". However, basically, computers have only 256 character codes, which are insufficient to be assigned to a number of characters including Kanji characters, and thus, in the Big Font, characters can be represented by double-byte codes (two character codes that are concatenated to designate one character). When using the double-byte code, users can select a specific ASCII code which is rarely used as the "escape code (a character code used by the computer together with the next character to determine that the relevant code expresses a kanji character)" so as to prevent the computer from falsely recognizing that the first code is an individual character code.

AutoCAD is accompanied with BIGFONT. shx and EXTFONT. shx as the Big Font file in advance and the definition of characters within said file is disclosed and thus, users can alter or refer to it and customize the characters.

(b) The first line in the shape definition file of the Big Font is described (area declaration) in accordance with the syntax, "*BIGFONT nchars, nranges, b1, e1, b2, e2…".

The set of characters "nchras" represents the approximate value of the number of character definitions defined after such description.

The set of characters "nranges" designates the number of continuous ranges used as the escape code.

The part "b1, e1" designates the head and last code in the continuous ranges used as the escape code.

b. Description concerning the shape definition of AutoCAD

The shape definition file is described in accordance with the grammar prescribed in AutoCAD. Its protocol is stated in the manual attached to AutoCAD as standard. The font of AutoCAD is a vector font which shows the letter shape by a line instead of showing it by "a cluster of spots" as a bitmap.

The syntax of the shape description of the shape definition file of AutoCAD is as follows, and in each line of the shape definition file, a maximum 128 characters can be stated.

```
"*shapenumber, defbytes, shapename
   specbyte 1, specbyte 2, specbyte 3....0"
```

(a) The set of characters "*shapenumber" represents a unique number (shape number) assigned to a file and this shape number will be designated when calling up the shape file. In the case of a character font, a specific number corresponding to the value of each character in the ASCII code will be required (in the shape definition file, the shape of graphics, etc. other than characters can be described and in that case, an arbitrary number can be assigned).

(b) The set of characters "defbytes" refers to the number of data bytes required for describing the shape and the maximum for each shape is 2,000 bytes.

(c) The set of characters "shapename" refers to the name of the shape.

(d) The set of characters "specbyte" refers to the shape designating byte and each designating byte is a code that defines the length and direction of the vector or one of the special code numbers. The special code number has 15 types from 0 to E. For example, the following code number has the respective meanings mentioned below: [i] "0" or "000" means "end of shape definition"; [ii] "1" or "001" means "activate the drawing mode (pen down)"; [iii] "2" or "002" means "deactivate the drawing mode (pen up)"; [iv] "3" or "003" refers to the act of dividing the length of the vector by the byte to be described next; and [v] "4" or "004" refers to the act of multiplying the length of the vector by the byte to be described next.

With respect to the shape designating byte, there are two methods to describe each displacement point: [i] the method of representing the length and direction code of the vector by the character string consisting of three characters (the first character designates zero, the second character designates the length of the vector and the third character designates the direction of the vector); and [ii] the method of designating the coordinate value by "(X displacement, Y displacement)" after the special designated code "8" or "9". In the description method mentioned in [i] above, there are 15 types from 1 (1 unit length) to F (15 unit length) for the valid range which can be designated as the length of the vector, while there are 16 types from 0 to F (the direction equally and sequentially divided into 16 portions in a counterclockwise rotation from the 90-degree angle) for the valid range to designate the direction. In the description method mentioned in [ii] above, the range which can be designated as the XY displacement values is from -128 to +127.

(B) Contents of the description concerning the shape definition in the Plaintiff's Programs and in the Defendant's Programs

a. Descriptions concerning the font definitions used by the plaintiff and the defendant

(a) Description concerning the font definition used by the plaintiff

The initial declaration statement in the plaintiff's SUG-BIG1. shp (shape definition file) reads "BIGFONT 10369, 3, 05F, 060, 07B, 0A0, 0E0, 0FF" (Exhibit Ko 19). This means that three external character areas, "5F, 60", "7B, A0" and "E0, FF", have been set as the continuous range to be used as the escape code.

In its font definition file, the plaintiff has created a unique definition statement for Kanji characters, Alphabets, subscripts, superscripts and special symbols to be used in the text that are not attached to AutoCAD as standard.

(b) Description concerning the defendant's font definition

The special characters used in the Defendant's Programs include 118 characters with special fonts uniquely created by the defendant without using the font attached to AutoCAD. Among them, the number of characters for which the codes are designated within the three external character areas, "5F, 60", "7B, A0" and "E0, FF", (characters that have the same shape code as the plaintiff's special characters) and which are special characters that the plaintiff has also uniquely created without using the font attached to AutoCAD is 54 in total.

With respect to 10 characters out of such 54 characters, the shapes of the characters used by the plaintiff and the defendant differ from each other and thus the coordinate values of the changing points in the shape definition also differ. With respect to one character, the coordinate value of the changing point and the stroke order are almost the same but as a result of the difference in the starting point of the character, the shape of the character does not overlap between the one used by the plaintiff and the one used by the defendant. With respect to 38 characters, while the shape is identical and almost overlaps, respectively, the stroke order differs (the plaintiff's shape definition is described in an counterclockwise manner or from bottom to top, while the defendant's shape definition is described in a clockwise manner or from top to bottom), and thus the coordinate values of the changing points in the shape definition differ. One of the 38 characters is a character "(((" with the character code "0F27C" mentioned below. With respect to five characters, while both the shape and stroke order of the character match up, respectively, and thus the coordinate values of the changing points in the shape definition also match up, the specific description of the overall shape definition has different parts, respectively, such as the difference in the indication of the scaling factor of the vector described at the beginning of the shape definition. The shapes of the abovementioned five characters are "×", "/", "|", "^{l}(superscript small l)" and "$_{l}$(subscript small l)".

b. The descriptions of the character code "0F27C" used by the plaintiff and the defendant

(a) The plaintiff's shape description

The description of the plaintiff's shape definition for the character code "0F27C2 is as follows.

"0F27C, 107, [{B=Special(((]

2, 3, 5, 4, 3, 2, (8, 7, -3), 1, (8, -2, 2), (8, -2, 3), (8, -2, 4), (8, -1, 6), (8, 1, 6), (8, 2, 4), (8, 2, 3), (8, 2, 2), 2, (8, 0, -30), (8, 5, 2), 1, (8, -2, 2), (8, -2, 3), (8, -1, 2), (8, -1, 6), (8, 1, 6), (8, 1, 2), (8, 2, 3), (8, 2, 2), 2, (8, 0, -28), (8, 5, 4), 1, (8, -2, 2), (8, -2, 4), (8, -1, 4), (8, 0, 2), (8, 1, 4), (8, 2, 4), (8, 2, 2), 2, (8, 0, -26), (8, 6, 3), 3, 3, 4, 5, 0"

Accordingly, in the plaintiff's shape description, after the drawing mode is activated, the lines are drawn in the order of the changing points of "(-2, 2), (-2, 3), (-2, 4), (-1, 6), (1, 6), (2, 4), (2, 3), (2, 2), (0, -30), (5, 2), (-2, 2), (-2, 3), (-1, 2), (-1, 6), (1, 6), (1, 2), (2, 3), (2, 2), (0, -28), (5, 4), (-2, 2), (-2, 4), (-1, 4), (0, 2), (1, 4), (2, 4), (2, 2)." This will result in the move of drawing the "(" line from the bottom to upward and from the left side to the right direction on the screen.

(b) The defendant's shape description

The description of the defendant's shape definition for the character code "0F27C" is as follows.

"0F27C, 95, (((

3, 100, 4, 60, 2, 14, 8, (-8, -25), 2, 5, 8, (7, 27), 1, 9, (-2, -2), (-2, -3), (-2, -4), (-1, -6), (1, -6), (2, -4), (2, -3), (2, -2), (0, 0), 2, 8, (5, 28), 1, 9, (-2, -2), (-2, -3), (-1, -2), (-1, -6), (1, -6), (1, -2), (2, -3), (2, -2), (0, 0), 2, 8, (5, 24), 1, 9, (-2, -2), (-2, -4), (-1, -4), (0, -2), (1, -4), (2, -4), (2, -2), (0, 0), 2, 6, 8, (23, 0), 2, 14, 8, (-15, -6), 4, 100, 3, 60, 0"

Accordingly, in the defendant's shape description, after the drawing mode is activated, the lines are drawn in the order of the changing points of "(-2, -2), (-2, -3), (-2, -4), (-1, -6), (-1, -6), (2, -4), (2, -3), (2, -2), (5, 28), (-2, -2), (-2, -3), (-1, -2), (-1, -6), (1, -6), (1, -2), (2, -3), (2, -2), (5, 24), (-2, -2), (-2, -4), (-1, -4), (0, -2), (1, -4), (2, -4), (2, -2)." This will result in the move of drawing the "(" line from the top to downward and from the left side to the right direction on the screen.

(c) With respect to the line "(((" represented by the character code "0F27C," its shape on the screen is the same in both cases where it is indicated by the plaintiff's shape definition and the defendant's shape definition and the two lines so displayed overlap.

(C) Creativity of the description concerning the plaintiff's shape definition and comparison, etc. between the description concerning the plaintiff's shape definition and the description concerning the defendant's shape definition

a. Whether or not the description falls under the program provided for in Article 2, paragraph (1), item (x)-2 of the Act

This court will determine for confirmation whether or not the description of the plaintiff's shape definition falls under the program provided for in Article 2, paragraph (1), item (x)-2 of the Act.

The description of the plaintiff's shape definition consists of numbers such as "2" and "0" and thereby movements such as "pen up" or "end of shaping" will be conducted by a computer. Such description exists on AutoCAD and carries a meaning as an instruction for the computer only when it is read in the program that defines the shape description (for example, a program that defines that "2" means "pen up") and functions in cooperation. As such, the description of the shape definition is mere data in which the information to be read by AutoCAD is stated, and thus it may be possible to construe that the relevant description does not fall independently under "something expressed as a set of instructions written for a computer, which makes the computer function so that a specific result can be obtained." Yet, even if said description has no independence and cannot individually be used, it can be regarded as something expressed as a set of instructions written for a computer by functioning in cooperation with other programs that read the data portion, and thus it is permissible to construe that such description falls under the program provided for in said item.

Accordingly, the description of the plaintiff's shape definition should be covered by the protection under the Copyright Act so long as the specific description is creative.

b. Creativity

The method to describe the plaintiff's shape definition is prescribed by AutoCAD, which is a program that executes the shape definition file. Such description is represented by the combination of the coordinate values of the displacement points that specify the starting point and ending point of the vector (from -128 to +127) or the character string consisting of three characters that represent the length and direction code of the vector, and the special designated code from 1 to 10 that instructs the movement between such coordinate values. According to the grammar for shape definition, if the font of a specific shape or shape is to be stated by a normal stroke order, the creator will have fewer options for the method to describe the coordinate values of the displacement points and thus it cannot be found that the shape description will have creativity. The description of the plaintiff's shape definition of special characters follows the normal stroke order as the stroke order to draw the relevant character and thus the description of the coordinate values of the plaintiff's shape description cannot be found to have creativity.

c. Comparison

Even if it is possible to find that the specific expression used for the plaintiff's shape description is creative, in light of the fact that the coordinate values of the changing points or stroke orders differ for 49 characters out of the total 54 special characters uniquely created by the plaintiff and the defendant (for example, the stroke order of the shape description used by the plaintiff and the defendant for the lines "(((", which is alleged by the plaintiff as being identical, differs in that, while it is written from the top to downward in the defendant's shape description, it is written from the bottom to upward in the plaintiff's shape description and the specific description of the shape definition also differs from each other), it cannot be found that the specific expression methods used by the two parties are identical nor can the essential characteristic part of the description of the plaintiff's shape definition which has creativity be directly perceived from the description of the defendant's shape definition.

With respect to the five characters ("×", "/", "|", "^{l}(superscript small l)" and "$_{l}$(subscript small l)") out of the 54 characters mentioned above, the representation of the shape description mentioned above will be decided almost by necessity and thus there is no room for the shape definition for the five characters to have creativity.

c. Therefore, it cannot be found that the right of reproduction or adaptation right has been infringed with respect to the shape description of the plaintiff's special character codes.

B. Regarding the assignment area of the shape codes

The plaintiff alleges that the Defendant's Programs and the Plaintiff's Programs are substantially identical in terms of the assignment area of the shape codes. However, the selection of the assignment area of the shape codes in the Plaintiff's Programs is an idea and cannot serve as the basis for finding creativity in the Plaintiff's Programs or the description of the plaintiff's shape definition. Thus, the plaintiff's allegation in this regard is inappropriate.

2. Regarding issue 2 (Whether or not the defendant has reproduced and stored Plaintiff's Program 1 in the storage media in its computer in creating the Defendant's Programs)

The plaintiff alleges that the defendant has reproduced and stored the Plaintiff's Programs in the storage media, etc. in the defendant's computer on the grounds that there are common features between the Defendant's Programs and the Plaintiff's Programs.

However, for the following reasons, even if there are common features between the Plaintiff's Programs and the Defendant's Programs, the defendant could recognize the contents of the Plaintiff's Programs by a method other than reproducing the Plaintiff's Programs in the storage media, etc. in the defendant's computer, and thus the existence of such common features cannot serve as the basis for finding that the defendant has reproduced the Plaintiff's Programs.

(1) Regarding the matching of the coordinate values of the changing points of the special character data

The plaintiff alleges that the defendant has reproduced the Plaintiff's Programs based on the fact that the shapes of the special characters used in the Plaintiff's Programs and 90% of the shapes of the special characters used in the Defendant's Programs almost completely match.

However, the shapes of the special characters used in the Plaintiff's Programs can be recognized by executing the LISP program for the creation of the "font list" that is used in the Plaintiff's Programs and taking a look at the "font list" thus created (Exhibits Otsu 4, 8 and 10). Moreover, as found above, among the 54 characters uniquely created by the plaintiff and the defendant, only five characters completely match up in terms of the shape definition (coordinate values of the changing points) and such five characters have shapes that are simple enough to be natural even if the coordinate values agree. Thus, the fact that the shapes of the special characters used by the plaintiff and those by the defendant match does not serve as the basis for finding that the defendant has reproduced the Plaintiff's Programs.

(2) Matching of the assignment area of the shape codes

The plaintiff alleges that the defendant has reproduced the Plaintiff's Programs based on the fact that the assignment area of the shape codes used by the plaintiff and that by the defendant almost completely match and that all of the special external characters assigned to said assignment area, including the parts for which dual definition has been conducted, match.

However, according to the evidence (Exhibits Otsu 4, 8 and 10), the shape codes of the special characters used in the Plaintiff's Programs and the assignment area thereof can be recognized by taking a look at the "font list" that indicates the font(s) used in the Plaintiff's Programs. Moreover, the defendant can be found to have assigned the same shape codes as those assigned in the Plaintiff's Programs to the special characters corresponding to the special characters that exist in the Plaintiff's Programs based on the abovementioned font list in order to open the drawings created by the Plaintiff's Programs in the Defendant's Programs. Thus, the fact that the assignment area of the shape codes of the special characters used by the plaintiff and that by the defendant match does not serve as the basis for finding that the defendant has reproduced the Plaintiff's Programs.

(3) Matching of the number of auxiliary lines

The plaintiff alleges that the defendant has reproduced the Plaintiff's Programs based on the fact that the number of the horizontal auxiliary lines (upper and lower base lines) drawn in the defendant's Contact Line - Base Line Program and the number of the plaintiff's auxiliary lines match.

However, the number of the upper and lower lines can be easily recognized by taking a look at the drawings created by the Plaintiff's Programs (Exhibit Ko 3) and it is natural for the defendant to use the same number of lines as those used in the previous drawings for the convenience of JR-EAST. Thus, such fact cannot serve as the basis for finding that the defendant has reproduced the Plaintiff's Programs.

(4) Matching of the escape codes

The plaintiff alleges that the defendant has reproduced the Plaintiff's Programs based on the fact that the escape codes (""}"") used in the Defendant's Programs are the same as the codes used in the Plaintiff's Programs.

However, if the Plaintiff's Programs created by using the escape codes used by the plaintiff are opened in the Defendant's Programs, such escape codes are displayed on the screen as the characters of the relevant codes as they are, without playing the role of escape codes (Exhibit Otsu 4). Moreover, they can be recognized without reproducing the Plaintiff's Programs and it is reasonable for the defendant to use the same escape codes as those used in the Plaintiff's Programs for the convenient operation of the user, or JR-EAST. Thus, the matching of the escape codes cannot serve as the basis for finding that the defendant has reproduced the Plaintiff's Programs.

(5) Matching of the linefeed codes

The plaintiff alleges that the defendant has reproduced the Plaintiff's Programs based on the fact that the linefeed codes and the amount of linefeed used in the Defendant's Programs are the same as the linefeed codes and the amount of linefeed used in the Plaintiff's Programs.

However, when the Plaintiff's Programs created by using the linefeed codes used by the plaintiff are opened in the Defendant's Programs, such linefeed codes will be displayed on the screen as characters which cannot be read (" ・ ") but such codes can be found by pasting such characters into another application (Exhibit Otsu 4). In addition, with respect to the amount of linefeed (in the shape definition of the plaintiff's linefeed codes, the coordinate value of the displacement point is (0, -30) and this means that it moves 30 units to the downward direction), it is possible to find the same amount of linefeed as that when the plaintiff's linefeed codes are used. Moreover, it is natural for the defendant to set the same codes as those used in the Plaintiff's Programs as the escape codes and to match the amount of linefeed for the operational

convenience of JR-EAST. Thus, the fact that the linefeed codes are matching does not serve as the basis for finding that the defendant has reproduced and analyzed the Plaintiff's Programs.
(6) Matching of the order of setting the values into variables

The plaintiff alleges that the defendant has reproduced the Plaintiff's Programs based on the fact that the order of setting the values into variables in the programs in the initializing part of the defendant (the order of "initial value of the km route", "offset value of the km route", "scale" and "paper size") is the same as the order for setting the values into variables in the Plaintiff's Programs.

However, the abovementioned order is a natural order for setting variables, and the fact that such order is matching cannot serve as the basis for finding that the defendant has reproduced the Plaintiff's Programs.
(7) As found above, the common features alleged by the plaintiff cannot serve as the basis for finding that the defendant has reproduced the Plaintiff's Programs and there is no other evidence sufficient to uphold the plaintiff's allegation.

Accordingly, the plaintiff's allegation in this regard is groundless.
3. Based on the abovementioned findings, all of the plaintiff's claims are groundless without the need to make determination on other points. Thus, the judgment shall be rendered in the form of the main text.

Tokyo District Court, 29th Civil Division

Presiding judge: IIMURA Toshiaki

Judge: IMAI Hiroaki

Judge: OOYORI Asayo

(Attachments)
List of the Defendant's Products
Product name: Quite (Kuwaito) Railroad Edition
Function: Computer-aided drafting and design software programs for railroads
Registered holder/person: SATORI ELECTRIC CO., LTD.

List of the Plaintiff's Programs
1. AutoCAD GXIII version JR-CAD

Among the "Contact Line - Base Line Creation Program," the program of the menu display part, the program of the input part and the program of the drawing part are as stated in Attachments 1 through 3, respectively.

2. AutoCAD R13 version JR-CADII

Attachment 1, Attachment 2, Attachment 3

（別紙１）

```
YBJ-TR68.DEC      Page 1

   (princ "¥nybj-TR68  v1.1¥n")
 ;  内容:電車線*a行程、基準線作成
 ;  作成:89-04-20   サブ:無し  備考:
 ;

(setq B1 0)

(princ "¥n¥n¥n¥n¥n¥n¥n¥n¥n¥n¥n")
(princ "¥n《ＪＲ－ＣＡＤ》【電車線*a行程、基準線作成】")
(princ "¥n")
(princ "¥n１．データファイルの作成")
(princ "¥n")
(princ "¥n２．データファイルの修正")
(princ "¥n")
(princ "¥n３．基本線作成（データファイル作成後）")
(princ "¥n")
(princ "¥n４．データファイルの文法説明")
(princ "¥n")
(princ "¥n０．終了")
(princ "¥n")
(setq B0 (getint "¥n目的の番号を入力（0）: "))(if (= B0 nil)(setq B0 0))

(if (= B0 1)(load "Z:YBJ-TQ02"))
(if (= B0 2)(load "Z:YBJ-TR80"))
(if (= B0 3)(load "Z:YBJ-TR79"))
(if (= B0 4)(load "Z:YBJ-TR78"))
(if (= B0 0)(setq B1 1))

```

（別紙２）

```
YBJ-TQ02.DEC     Page 1

 ;  (princ "¥nybj-TQ02  v1.0¥n")
 ;  内容::  電車線基本データ作成
 ;  作成:90-02-06  備考:点データ入力サブルーチン
 ;

(setq W0 1)

(princ "¥n¥n¥n《ＪＲ－ＣＡＤ》【電車線"キロ行程、基準線データ登録】")
(princ "¥n¥n電車線基本データ登録¥n")

(setq F1 "T:TEST.DAT")

(setq V0 (getreal "¥n●キロ行程の最初の値をにm単位で入力〈0〉："))
(if (= V0 nil)(setq V0 0.0))
(setq V0 (rtos V0 2 2))

(princ        "¥n¥n¥n●キロ行程のオフセット値（スタート値から最")
(setq V1 (getreal "¥n初のマークまでの距離）をm単位で入力〈0〉："))
(if (= V1 nil)(setq V1 0.0))
(setq V1 (rtos V1 2 2))

(setq V2 (getreal "¥n¥n¥n●縮尺の分母のみ（例:1/500は500）入力〈500〉："))
(if (= V2 nil)(setq V2 500.0))
(setq V2 (rtos V2 2 0))

(princ         "¥n¥n¥n●キロ用紙サイズ Ａ４（高さ210mm）Ａ３（高さ297mm）")
(setq V3 (getstring "¥n                     Ａ２（高さ420mm）を入力〈A4〉："))
(if (= V3 "")(setq V3 "A4"))

(setq F0 (open F1 "w"))
(write-line V0 F0)
(write-line V1 F0)
(write-line V2 F0)
(write-line V3 F0)

(setq V0 (getstring "¥n¥nスパンを入力〈50〉"))
(if (= V0 "")(setq V0 "50"))

(princ "¥n¥nL又はRを入力すると最近の基準線の左右に補助線が作成されます")

(while (/= V0 "999")

(write-line V0 F0)


(princ "¥n¥nスパンを入力 (終了は999)〈")(princ V0)
(setq B0 (getstring "〉"))(if (/= B0 "")(setq V0 B0))

),w

(close F0)

```

（別紙３）

```
'YBJ-TR79.DEC     Page 1

   (princ "¥eybj-TR79  v3.2¥n")
 ;  内容:電車線*ｏ工程．基準線作成（サブ  作成）
 ;  作成:89-04-20   サブ:無し  備考:
 ;

(textscr)

(princ "¥n¥n¥n¥n¥n¥n¥n¥n¥n¥n¥n")
(princ "¥n【ＪＲ－ＣＡＤ】［電車線*ｏ工程．基準線作成］")
(princ "¥n")
(setq F0 "T:TEST.DAT")
(setq F1 (open F0 "r"))

(if (= F1 nil)
(while 1
(princ "¥n¥nデータファイルがありません...")
(getint "¥n¥n［命令中断］を押して下さい：")
);w
);i

(close F1)
(setq F1 (open F0 "r"))

;-----*ｏ行程----------------
(setq V0 (read-line F1))
(setq V0 (atof V0) V1 (rtos (/ V0 100.0) 2 0))
(setq L1 (strlen V1) V2 (substr V1 1 (- L1 1)) V3 (substr V1 L1))

(setq B0 (strcat V2 "K" V3 "00m"))
(princ "¥n¥n¥nスタートの*ｏ行程は［")(princ B0)(princ "］です")

;-----*ｏ行程オフセット--------------
(setq V9 (read-line F1))
(setq V9 (atof V9) B0 (rtos V9 2 0))
(princ "¥n¥n¥nスタートのマークのオフセット距離は［")(princ B0)(princ "m］です")

;-----縮尺---------------
(setq Z0 (read-line F1))
(setq Z0 (atof Z0) Z1 (rtos Z0 2 0) Z5 (/ 1.0 Z0))

(setq B0 (strcat "1/" Z1))
(princ "¥n¥n¥n縮尺は［")(princ B0)(princ "］です")

;-----サイズ----------------
(setq A0 (read-line F1))
(if (= (substr A0 1 2) "A4")
(setq Y0 105.0 Y1 10.0 Y2 30.0 Y3 180.0 Y4 185.0 Y5 195.0 Y6 210.0 B0 "A4(高さ210mm)")
(if (= (substr A0 1 2) "A3")
(setq Y0 145.0 Y1 15.0 Y2 40.0 Y3 260.0 Y4 270.0 Y5 280.0 Y6 297.0 B0 "A3(高さ297mm)")
(if (= (substr A0 1 2) "A2")
(setq Y0 210.0 Y1 15.0 Y2 40.0 Y3 385.0 Y4 395.0 Y5 405.0 Y6 420.0 B0 "A2(高さ420mm)")
(setq B0 "不適当")
)));iii

(princ "¥n¥n¥n用紙サイズは［")(princ B0)(princ "］です")

(setq P0 (getpoint "¥n¥n¥nスタート位置を指示してください<0,0>:"))

(if (= P0 nil)(setq P0 (list 0 0)))

;-------------------------
(setq X0 (car P0) X1 X0 X4 X0 N0 0)

(command "LAYER" "M" "S" "")
(command "line" (list X0 Y2)(list X0 Y3) "")

(setq S0 (read-line F1))

(while (/= S0 nil)

;-----左基準記入---------
(if (= (substr S0 1 1) "L")
  (progn
  (command "LAYER" "M" "1" "")
  (setq X3 (- X0 4.0))
  (command "line" (list X3 Y2)(list X3 Y3) "")
  );p
;-----右基準記入---------
(if (= (substr S0 1 1) "P")
  (progn
  (command "LAYER" "M" "    ")
  (setq X3 (+ X0 4.0))
  (command "line" (list X3 Y2)(list X3 Y3) "")
  );p
(progn
;----すぱん記入--------
(setq X1 (atof S0))
(if (/= X1 nil)
  (progn
  (setq X1 (* X1 1000.0 Z0))
  (setq X0 (+ X0 X1))
  (setq X2 (- X0 (/ X1 2.0)))
  (command "LAYER" "M" "5" "")
  (command "line" (list X0 Y2)(list X0 Y3) "")
  (command "LAYER" "M" "1" "")
  (command "TEXT" "M" (list X2 Y0) 2.5 0.0 S0)
  );p
);p
);i
);i
);i

(setq S0 (read-line F1))
);w

(close F1)

(command "LAYER" "M" "4" "")
(command "line" (list X4 (/ Y6 2.0))(list X0 (/ Y6 2.0)) "")
(command "LAYER" "M" "3" "")
(command "line" (list X4 0.0)(list X0 0.0) "")
(command "line" (list X4  Y6)(list X0  Y6) "")
(command "LAYER" "M" "2" "")
(command "line" (list X4  Y1)(list X0  Y1) "")
(command "line" (list X4 (+ Y1  5))(list X0 (+ Y1  5)) "")
(command "LAYER" "M" "1" "")
(command "line" (list X4 (+ Y1 10))(list X0 (+ Y1 10)) "")
(command "line" (list X4 (+ Y1 15))(list X0 (+ Y1 15)) "")
(command "line" (list X4 (+ Y1 20))(list X0 (+ Y1 20)) "")
(command "LAYER" "M" "6" "")
(command "line" (list X4  Y4)(list X0  Y4) "")
(command "LAYER" "M" "1" "")
(command "line" (list X4  Y5)(list X0  Y5) "")

; X1:図面上の距離       X9:図面上のオフセット距離(mm)
; Y1:表示される距離(M)  Y9:offset(M)   S1:表示内容(S)
; Y8:表示上の最終位置(M) Y0:最初の距離(M)

(setq X9 (* Y9 Z0 1000.0))
(setq Y8 (+ (/ (- (+ X0 X9) X4) Z0 1000.0) Y0))
(setq Y1 Y0)

(command "LAYER" "M" "1" "")

(while (<= Y1 Y8)

(setq X1 (+ (* (- Y1 Y0) Z0 1000.0) X9 X4))
(if (= Y1 (* (fix(/ Y1 1000.0)) 1000.0))
(progn
(command "insert" "S:YBD-ET02" (list X1 Y5) K2B "" 0.0)
(setq S0 (rtos (/ Y1 100.0) 2 0) L1 (strlen S0) S2 (substr S0 1 (- L1 1)))
(setq S3 (substr S0 L1) S1 (strcat S2 "]K" S3 "00]M"))
(command "TEXT" "C" (list X1 (+ Y5 5.0)) (* K2B 2.5) 0.0 S1)
);p
(if (= Y1 (* (fix(/ Y1 500.0)) 500.0))
(progn
(command "insert" "S:YBD-ET04" (list X1 Y5) K2B "" 0.0)
(setq S0 (rtos (/ Y1 100.0) 2 0) L1 (strlen S0) S2 (substr S0 1 (- L1 1)))
(setq S3 (substr S0 L1) S1 (strcat S2 "]K" S3 "00]M"))
(command "TEXT" "C" (list X1 (+ Y5 5.0)) (* K2B 2.5) 0.0 S1)
);p
(progn
(command "insert" "S:YBD-ET03" (list X1 Y5) K2B "" 0.0)
(setq S0 (rtos (/ Y1 100.0) 2 0) L1 (strlen S0))
(setq S3 (substr S0 L1))
(command "TEXT" "C" (list X1 (+ Y5 5.0)) (* K2B 2.5) 0.0 S3)
```

```
);w
);i
);i

(setq Y1 (+ Y1 100.0))

);w
(command "LAYER" "M" "4" "")

(setq P0 nil A0 nil F0 nil F1 nil L1 nil B0 nil)
(setq Y0 nil Y1 nil Y2 nil Y3 nil Y8 nil Y9 nil)
(setq Z0 nil Z1 nil)
(setq Y0 nil Y1 nil Y2 nil Y3 nil Y4 nil Y5 nil Y6 nil)
(setq X0 nil X1 nil X2 nil X3 nil X4 nil x9 nil N0 nil)
(setq S0 nil S1 nil S2 nil S3 nil)
```

List of the Defendant's Programs

1. AutoCAD R13J version Quite Railroad Edition

Among the "Contact line - Base line Creation Program," the program of the menu display part, the program of the input part and the program of the drawing part are as stated in Attachment 4.

2. AutoCAD R14 version Quite Railroad Edition Ver. 1.1

3. AutoCAD 2000i version Quite 2000 Railroad Edition Ver. 1.0

Attachment 4

（別紙４）

```
;------------------------------------------------------------------------------
; 基準線作成コマンド  Version 1.30
;------------------------------------------------------------------------------
(defun c:BaseLine(
  (
    /
    fTanBase                ; 入力距離の単位(kmなら1000.0)

    lsWriteData lsReadData

    sDefaultNo
    sInputNo
    bLoopFlg

    sBasePath sDataFile
    sInsDwg1 sInsDwg2 sInsDwg3        ; キロ程記号のブロック名
    sMenuID
    sEditorName                ; エディタの実行ファイル名
    lsPaList
    fWork1
    ptWork1

    X Y


    InputBaseLineData            ; 基準線のデータ入力部分
    ListDataWrite               ; リストのないようをファイルに出力する
    ListDataRead                ; ファイルの内容をリストにする
    DrawBaseLine                ; 基準線を作図する
    DelLastZero                 ; 最後のゼロを取り除く
    Tos                    ; 関数名がないので
    GetUserSize
    GetUserScale
  )


;------------------------------------------------------------------------------
; 基準線のデータ入力部
;------------------------------------------------------------------------------
(defun InputBaseLineData
  (
    /
    fStartKiro                ; キロ程の最初の値
    fOffsetKiro               ; キロ程のオフセット値
    fScScaleM fScScaleC          ; 縮尺(分子、分母)
    sPaSize fPaHeight            ; キロ用紙のサイズ(高さ)
    fSpan                ; スパン距離
    fOldSpan lsSpanList


    bChangeFlg
    bLeftFlg bRightFlg
    bLoopFlg

    lsRet
  )

  (setq bChangeFlg nil)
  (if (+GetDwgInfo)
    ; 図面の設定が既にされている(SETUPコマンドが実行されている)
    (progn
      (initget "Y N")
      (princ (strcat "¥n用紙サイズ: " (nth Setup:iSz Setup:sSize) "   図面尺度: " Setup:sScm "/" Setup:sSc "で設定されています"))
      (if (= (getkword "¥nそのままの設定でいきますか <Y>: ") "N")
        (setq bChangeFlg T)   ; 変更する
      )
    )
    (setq bChangeFlg T)
  )
  (if bChangeFlg
    (progn
      ; 尺度を入力させる
      (setq fScScaleM 1.0)
      (initget 6)
      (setq fScScaleC (getreal "¥n尺度の分母のみ(1/500のときは500)を入力 <500>: "))
      (if (null fScScaleC)
        (setq fScScaleC 500.0)
      )

        キロ用紙のサイズを入力させる
      (initget 6 "A0 A1 A2 A3 A4")
      (setq sPaSize (getreal "¥n用紙サイズ(A4, A3, A2, A1, A0)または高さをmm単位で入力 <A4>: "))
      (if (or (= (type sPaSize) 'REAL) (= (type sPaSize) 'INT))
        ; 高さが入力されたとき
        (progn
          (setq fPaHeight (* sPaSize 1.0))
          (setq sPaSize (rtos sPaSize))
        )
        ; 用紙サイズが入力されたとき
        (progn
          (if (not sPaSize)
            (setq sPaSize "A4")
          )
          (setq fPaHeight (cdr (assoc sPaSize lsPaList)))
        )
      )
    )
    (progn
      (setq fScScaleM (atof Setup:sScM))
      (setq fScScaleC (atof Setup:sScC))
      (setq fPaHeight (/ (abs (- (cadr (getvar "LIMMAX")) (cadr (getvar "LIMMIN")))) Setup:fSc))
      (setq sPaSize (rtos fPaHeight))
    )
  )

  ; キロ行程の最初の値を入力させる(負数入力禁止)
  (initget 4)
  (setq fStartKiro (getreal "¥nキロ行程の最初の値をm単位で入力 <0>: "))
  (if (not fStartKiro)
    (setq fStartKiro 0.0)
  )

  ; キロ行程のオフセット値を入力させる(負数入力禁止)
  (initget 4)
  (setq fOffsetKiro (getreal "¥nキロ行程のオフセット値(スタート値から最初のマークまでの距離)をm単位で入力 <0>: "
))
  (if (not fOffsetKiro)
    (setq fOffsetKiro 0.0)
  )

  (setq bLeftFlg nil)
  (setq bRightFlg nil)
  (setq bLoopFlg T)
  (setq fOldSpan 50.0)
  (setq lsSpanList '())

  ; スパン距離入力(ゼロと負数入力禁止)
  (initget 6)
  (setq fSpan (getreal (strcat "¥nスパン距離をm単位で入力 <" (DelLastZero fOldSpan) ">: ")))
  (while bLoopFlg
    (if (not fSpan)
      (setq fSpan fOldSpan)
    )
    (if (and (/= fSpan "L") (/= fSpan "R"))
      (setq fOldSpan fSpan)
    )
    (cond
      ((= fSpan "L")
        (setq bLeftFlg nil)
      )
      ((= fSpan "R")
        (setq bRightFlg nil)
      )
      ((= fSpan "E")
        (setq bLoopFlg nil)
      )
      (T
        (setq bLeftFlg T)
        (setq bRightFlg T)
      )
    )
    (if bLoopFlg
      (progn
        (setq lsSpanList (append lsSpanList (list fSpan)))
        (cond
          ((and bLeftFlg bRightFlg)
            (princ "¥nLまたはRを入力すると最後の基準線の左右に補助線が作成さ
            (initget 4 "E L R")
          )
          ((and (not bLeftFlg) bRightFlg)
            (princ "¥nRを入力すると最後の基準線の右に補助線が作成されます")
            (initget 4 "E R")
          )
          ((and bLeftFlg (not bRightFlg))
            (princ "¥nLを入力すると最後の基準線の左に補助線が作成されます")
            (initget 4 "E L")
          )
          (T
            (initget 4 "E")
          )
        )

        ; スパン距離入力(ゼロと負数入力禁止)
        (setq rSpan (getreal (strcat "¥nスパン距離をm単位で入力 (E=終了) <" (DefLastZero rOldSpan) ">: ")))
      )
    )
  )
  (append (list (/ rScScaleL rScScaleW) sPaSize rStartKiro rOffsetKiro) lsSpanList)
)


;------------------------------------------------------------------------------
; リストの内容をファイルに出力する
;------------------------------------------------------------------------------
(defun ListDataWrite
  (
    sWriteFile          ; データを出力するファイル名(フルパス)
    lsWriteList         ; ファイルに出力するリスト
    iStrNo iEndNo       ; ファイルに出力するリストの開始番号と終了番号
    iWriteMode          ;
    /
    hFp sBuff          ; ファイルポインタ, 出力データ
    xItemData          ; リストの項目
    i
  )

  (and
    (if (not (setq hFp (open sWriteFile "w")))
      (alert "ファイルがオープンできません")
      T
    )
    (progn
      (setq i iStrNo)
      (while (and (<= i iEndNo) (< i (length lsWriteList)))
        (setq xItemData (nth i lsWriteList))
        (princ xItemData hFp)
        (princ "¥n" hFp)
        (setq i (1+ i))
      )
      (close hFp)
      T
    )
  )
)


;------------------------------------------------------------------------------
; ファイルの内容を読み出しリストにする
;------------------------------------------------------------------------------
(defun ListDataRead
  (
    sReadFile          ; データを入力するファイル名(フルパス)
    /
    lsReadList          ; データを出力するリスト
    hFp sBuff          ; ファイルポインタ, 出力データ
    xItemData          ; リストの項目
    i
  )

  (setq lsReadList '())

  (and
    (if (not (setq hFp (open sReadFile "r")))
      (alert "ファイルがオープンできません")
      T
    )
    (progn
      (while (setq sBuff (read-line hFp))
        (setq lsReadList (append lsReadList (list sBuff)))
      )
      (close hFp)
      T
    )
  )
  lsReadList
)
```

```
;------------------------------------------------------------------------------------
; 基準線を作図する
;------------------------------------------------------------------------------------
(defun DrawBaseLine
  (
    lsBaseLineData              ; 作図する基準線のデータリスト
    ptP0                      ; 基準線の起点
    /
    fOffsetX fOffsetY
    fHigh00 fHigh01 fHigh02 fHigh03
    fHigh04 fHigh05 fHigh06 fHigh07
    fHigh08 fHigh09 fHigh10
    fHojoSplt
    sLineLay0 sLineLay1 sLineLay2
    sLineLay3 sLineLay4 sLineLay5
    sLineLay6
    sTextLay0 sTextLay1
    fTextOffY0 fTextOffY1
    fTextH0 fTextH1
    ptP1 ptP2 ptP3 ptP4
    ptP5 ptP6 ptP7 ptP8
    ptSP ptEP
    sPaSize fPaHeight fPaWidth
    fSpan
    fPaScale
    fSpace

    iSelKiro iOffKiro

    sIni

    i j sItem
  )

  ; 変数の設定
  (setq sIni (findfile "DRAWING.INI"))
  (setq iSelKiro (fix (/ (atof (nth 2 lsBaseLineData)) 100)))
  (setq iOffKiro (atof (nth 3 lsBaseLineData)))
  (setq fPaScale (atof (nth 0 lsBaseLineData)))
  (setq sPaSize (nth 1 lsBaseLineData))
  (if (assoc sPaSize lsPaList)
    (setq fPaHeight (cdr (assoc sPaSize lsPaList)))
    (setq fPaHeight (atof sPaSize))
  )
  (setq fPaHeight (* fPaHeight fPaScale))
  (setq fOffsetX (* 15.0 fPaScale))
  (setq fOffsetY (* 15.0 fPaScale))
  (setq fHigh00 (* 15.0 fPaScale))
  (setq fHigh01 (* 5.0 fPaScale))
  (setq fHigh02 (* 5.0 fPaScale))
  (setq fHigh03 (* 5.0 fPaScale))
  (setq fHigh04 (* 5.0 fPaScale))
  (setq fHigh05 (* 5.0 fPaScale))
  (setq fHigh08 (* 10.0 fPaScale))
  (setq fHigh09 (* 10.0 fPaScale))
  (setq fHigh10 (* 15.0 fPaScale))
  (setq fHigh06 (/ (- fPaHeight (+ (* fOffsetY 2.0) fHigh00 fHigh01 fHigh02 fHigh03 fHigh04 fHigh05 fHigh08 fHigh09 fHigh10)) 2.0))
  (setq fHigh07 fHigh06)
  (setq fHojoSplt (* 4.0 fPaScale))
  (setq sLineLay0 "TEXT")
  (setq sLineLay1 "SYMBOL")
  (setq sLineLay2 "THIN")
  (setq sLineLay3 "HOJO")
  (setq sLineLay4 "KAIL")
  (setq sLineLay5 "BUILD")
  (setq sLineLay6 "THIN")
  (setq sTextLay0 "THIN")
  (setq sTextLay1 "THIN")
  (setq fTextOffY0 (* 0.0 fPaScale))
  (setq fTextOffY1 (* 5.0 fPaScale))
  (setq fTextH0 (* 3.5 fPaScale))
  (setq fTextH1 (* 2.5 fPaScale))
  (setq fSpace (* 5.0 fPaScale))

  (command "UNDO" "BE")

  ; 縦線を作図
  (setq ptP2 (list (+ (X ptP0) fOffsetX) (+ (Y ptP0) fOffsetY fHigh00 fHigh01 fHigh02 fHigh03 fHigh04 fHigh05) 0.0))
  (setq ptP5 (list (X ptP2) (+ (Y ptP2) fHigh06) 0.0))
  (setq ptP1 (list (X ptP2) (+ (Y ptP5) fHigh07) 0.0))
  (setq ptSP ptP5)
  (command "LINE" ptP1 ptP2 "")
  (*putlayer (entlast) sLineLay3)
  (setq i 4)
  (while (< i (length lsBaseLineData))
    (setq vItem (nth i lsBaseLineData))
    (cond
      ; 最後の基準線の左に補助線を作図する
      ((= (strcase vItem) "L")
        (command "LINE" (list (- (X ptP1) fHojoSpl) (Y ptP1) 0.0) (list (- (X ptP2) fHojoSpl) (Y
)
        (*putlayer (entlast) sLineLay6)
      )
      ; 最後の基準線の右に補助線を作図する
      ((= (strcase vItem) "R")
        (command "LINE" (list (+ (X ptP1) fHojoSpl) (Y ptP1) 0.0) (list (+ (X ptP2) fHojoSpl) (Y
)
        (*putlayer (entlast) sLineLay6)
      )
      ; 指示されたスパン距離で基準線とスパン距離を作図する
      ((< 0.0 (setq fSpan (atof vItem)))
        (setq ptP3 (list (+ (X ptP1) (* fSpan fTanBase)) (Y ptP1) 0))
        (setq ptP4 (list (X ptP3)                           (Y ptP2) 0))
        (setq ptP6 (list (X ptP3)                           (Y ptP5) 0))
        (setq ptEP ptP6)
        (setq ptP7 (*midp ptP5 ptP6))
        (setq ptP8 (list (X ptP7) (+ (Y ptP7) fTextOffY0) 0))
        (command "LINE" ptP3 ptP4 "")
        (*putlayer (entlast) sLineLay3)
        (command "TEXT" "J" "MC" ptP8 fTextH0 0 (DelLastZero fSpan))
        (*putlayer (entlast) sTextLay0)
        (setq ptP1 ptP3)
        (setq ptP2 ptP4)
        (setq ptP5 ptP6)
      )
    )
    (setq i (1+ i))
  )
  (if ptEP
    (progn
      ; 桁線を作図する
      (command "LINE" ptSP ptEP "")
      (*putlayer (entlast) sLineLay4)
      (setq ptP1 (list (X ptSP) (+ (Y ptP1) fHigh08) 0))
      (setq ptP3 (list (X ptEP) (Y ptP1)             0))
      (command "LINE" ptP1 ptP3 "")
      (*putlayer (entlast) sLineLay5)
      (setq ptP1 (list (X ptP1) (+ (Y ptP1) fHigh09) 0))
      (setq ptP3 (list (X ptP3) (Y ptP1)             0))
      (setq ptSP ptP1)
      (setq ptEP ptP3)
      (command "LINE" ptP1 ptP3 "")
      (*putlayer (entlast) sLineLay2)
      (setq ptP1 (list (X ptP1) (+ (Y ptP1) fHigh10) 0))
      (setq ptP3 (list (X ptP3) (Y ptP1)             0))
      (command "LINE" ptP1 ptP3 "")
      (*putlayer (entlast) sLineLay0)

      (setq ptP2 (list (X ptSP) (- (Y ptP3) fHigh05) 0))
      (setq ptP4 (list (X ptEP) (Y ptP2)             0))
      (command "LINE" ptP2 ptP4 "")
      (*putlayer (entlast) sLineLay2)
      (setq ptP2 (list (X ptP2) (- (Y ptP2) fHigh04) 0))
      (setq ptP4 (list (X ptP4) (Y ptP2)             0))
      (command "LINE" ptP2 ptP4 "")
      (*putlayer (entlast) sLineLay2)
      (setq ptP2 (list (X ptP2) (- (Y ptP2) fHigh03) 0))
      (setq ptP4 (list (X ptP4) (Y ptP2)             0))
      (command "LINE" ptP2 ptP4 "")
      (*putlayer (entlast) sLineLay2)
      (setq ptP2 (list (X ptP2) (- (Y ptP2) fHigh02) 0))
      (setq ptP4 (list (X ptP4) (Y ptP2)             0))
      (command "LINE" ptP2 ptP4 "")
      (*putlayer (entlast) sLineLay1)
      (setq ptP2 (list (X ptP2) (- (Y ptP2) fHigh01) 0))
      (setq ptP4 (list (X ptP4) (Y ptP2)             0))
      (command "LINE" ptP2 ptP4 "")
      (*putlayer (entlast) sLineLay1)
      (setq ptP2 (list (X ptP2) (- (Y ptP2) fHigh00) 0))
      (setq ptP4 (list (X ptP4) (Y ptP2)             0))
      (command "LINE" ptP2 ptP4 "")
      (*putlayer (entlast) sLineLay0)

      ; キロ程を作図する
```

```
        (setq ptP1 (list (+ (X ptSP) (* 10(iKiro fTanBase)) (Y ptSP) 0))
        (setq ptP2 (list (X     1) (+ (Y ptP1) (TextOffY)) 0))
        (setq vItem (itoa i     .iro))
        (setq bLoopFlg T)
        (while bLoopFlg
          (if (< (X ptEP) (X ptP1))
            (setq bLoopFlg nil)
            (progn
              (cond
                ; 1kmごと
                ((= (rem iSrtKiro 10) 0)
                  (command "INSERT" sInsDwg1 ptP1 fPaScale fPaScale 0)
                  (if (<= 10 iSrtKiro)
                    (setq vItem (strcat (itoa (/ iSrtKiro 10)) "|K000|M"))
                    (setq vItem "0|K000|M")
                  )
                )
                ; 500mごと
                ((= (rem iSrtKiro 5) 0)
                  (command "INSERT" sInsDwg2 ptP1 fPaScale fPaScale 0)
                  (if (<= 10 iSrtKiro)
                    (setq vItem (strcat (itoa (/ iSrtKiro 10)) "|K500|M"))
                    (setq vItem "0|K500|M")
                  )
                )
                ; 100mごと
                (T
                  (command "INSERT" sInsDwg3 ptP1 fPaScale fPaScale 0)
                  (setq vItem (itoa (rem iSrtKiro 10)))
                )
              )
              (#putlayer (entlast) sLineLay2)
              (command "TEXT" "J" "MC" ptP2 fTextH 0 vItem)
              (#putlayer (entlast) sLineLay2)
              (setq iSrtKiro (1+ iSrtKiro))
              (setq ptP1 (list (+ (X ptP1) (* 100.0 fTanBase)) (Y ptSP) 0))
              (setq ptP2 (list (X ptP1) (Y ptP2) 0))
            )
          )
        )

        ; 表示画面を作図図形に合せる
        (command "ZOOM" "E")

        (if (not (#GetDwgInfo))
          ; 図面がセットアップされていないとき
          (and
            (progn
              (initget "Y N")
              (/= (getkword "\n図面の設定を作図した基準線に合せますか <Y>: ") "N")
            )
            (progn
              ; 図面情報ファイルを挿入
              (command "INSERT" "DWGINFO" "0,0" "" "" "")

              ; 図面のスケールを設定
              (setq Setup:iScln (GetUserScale))          ; スケールリストの番号
              (setq Setup:sScM "1")                      ; 図面尺度の分子
              (setq Setup:sScC (DelLastZero fPaScale))    ; 図面尺度の分母
              (setq Setup:rScl fPaScale)                  ; 尺度分母/尺度分子
              (setvar "LTSCALE" (* (atof (GetIni "General" "LTSCALE" sIni)) Setup:rScl))
              (setvar "LUPREC"      (atoi (GetIni "General" "LUPREC"  sIni)))

              ; 図面の大きさを設定
              (setq Setup:iSz (GetUserSize))                ; 用紙サイズリストの番号
              (setq ptP1 (setvar "LIMMIN" (list (X ptP0) (Y ptP0))))         ; 図面の左下点
              (setq ptP2 (setvar "LIMMAX" (list (+ (X ptP0) (* 10(fix(X 2.0)) (distance ptSP ptEP)) (+ (Y ptP0) fFrHeight))))
              (setvar "SNAPBASE" (list (X ptP0) (Y ptP0)))               ; スナップの基点

              (initget "Y N")
              (if (/= (getkword "\n図枠を挿入しますか <Y>: ") "N")
                ; 図面に外枠を挿入する
                (progn
                  (setq vItem (getvar "PLINEWID"))
                  (setvar "PLINEWID" vItem)
                  (command "PLINE"
                    (list (+ (X ptP1) fSpace) (+ (Y ptP1) fSpace))
                    (list (+ (X ptP1) fSpace) (- (Y ptP2) fSpace))
                    (list (- (X ptP2) fSpace) (- (Y ptP2) fSpace))
                    (list (- (X ptP2) fSpace) (+ (Y ptP1) fSpace))
                    "C"
                  )
```

```
                 (*putlayer (entlast) "BORDER")
                 (*putlayer (entlast) "BTLAYER")
               )
             )

             ; 寸法線の設定
             (setvar "DIMASZ"    (* (atof (foo "DIMENSION" "DIMASZ"   sIni)) Setup:rScl))
             (setvar "DIMBLK1"             (foo "DIMENSION" "DIMBLK"   sIni)             )
             (setvar "DIMBLK2"             (foo "DIMENSION" "DIMBLK"   sIni)             )
             (setvar "DIMTIH"       (atoi (foo "DIMENSION" "DIMTIH"   sIni))            )
             (setvar "DIMTOH"       (atoi (foo "DIMENSION" "DIMTOH"   sIni))            )
             (setvar "DIMASO"       (atoi (foo "DIMENSION" "DIMASO"   sIni))            )
             (setvar "DIMEXO"    (* (atof (foo "DIMENSION" "DIMEXO"   sIni)) Setup:rScl))
             (setvar "DIMEXE"    (* (atof (foo "DIMENSION" "DIMEXE"   sIni)) Setup:rScl))
             (setvar "DIMDLE"    (* (atof (foo "DIMENSION" "DIMDLE"   sIni)) Setup:rScl))
             (setvar "DIMDLI"    (* (atof (foo "DIMENSION" "DIMDLI"   sIni)) Setup:rScl))
             (setvar "DIMTXT"    (* (atof (foo "DIMENSION" "DIMTXT"   sIni)) Setup:rScl))
             (setvar "DIMTAD"       (atoi (foo "DIMENSION" "DIMTAD"   sIni))            )
             (setvar "DIMSAH"       (atoi (foo "DIMENSION" "DIMSAH"   sIni))            )
             (setvar "DIMTVP"       (atoi (foo "DIMENSION" "DIMTVP"   sIni))            )
             (setvar "DIMTOFL"      (atoi (foo "DIMENSION" "DIMTOFL"  sIni))            )
             (setvar "DIMTIX"       (atoi (foo "DIMENSION" "DIMTIX"   sIni))            )
             (setvar "DIMSOXD"      (atoi (foo "DIMENSION" "DIMSOXD"  sIni))            )
             (setvar "DIMTXSTY"            (foo "DIMENSION" "DIMTXSTY" sIni)             )

             ; 図面情報を保存
             (*putdwginfo)

             ; 図面の大きさで表示する
             (command "ZOOM" "A")
             (command "VIEW" "S" "UNIVERSE")
             T
           )
         )
       )
     )
   )
   (command "UNDO" "E")
   T
 )


;----------------------------------------------------------------------------
; 数字文字列の後ろのゼロをとる
;----------------------------------------------------------------------------
(defun DelLastZero ( rVal          ; 変換対象の数値
            /
            sVal
            i
            )

  (if (= (rem rVal 1))
    (rtos rVal 2 0)
    (progn
      (setq sVal (rtos rVal))
      (setq i (strlen sVal))
      (while (and (< 0 i) (= (substr sVal i 1) "0"))
        (setq i (1- i))
      )
      (substr sVal 1 i)
    )
  )
)


;----------------------------------------------------------------------------
; 関数名が長いので
;----------------------------------------------------------------------------
(defun foo ( sApp sTit sIni / )
  (*GetPrivateProfileString sApp sTit sIni)
)


;----------------------------------------------------------------------------
; INIファイルのユーザーサイズをサイズリストの番号を取得する
;----------------------------------------------------------------------------
(defun GetUserSize
  (
    /
    i j k
    sWork1 sWork2
    iWork1
    bLoopFlg
  )

  (setq bLoopFlg T)
  (setq i 0)
  (while (and bLoopFlg (/= (setq sWork1 (ioo "Size" (strcat "SizeData" (itoa i)) sIni)) ""))
    (setq j (strlen sWork1))
    (while (and (< 0 j) (/= (setq sWork2 (substr sWork1 j 1)) ","))
      (setq j (1- j))
    )
    (setq iWork1 (atoi (substr sWork1 (1+ j))))
    (if (= (Boole 1 iWork1 3) 2)
      (setq bLoopFlg nil)
      (setq i (1+ i))
    )
  )
  i
)


;--------------------------------------------------------------------------
; INIファイルのユーザースケールをスケールリストの番号を取得する
;--------------------------------------------------------------------------
(defun GetUserScale
  (
    /
    i j k
    sWork1 sWork2
    iWork1
    bLoopFlg
  )

  (setq bLoopFlg T)
  (setq i 0)
  (while (and bLoopFlg (/= (setq sWork1 (ioo "Scale" (strcat "ScaleData" (itoa i)) sIni)) ""))
    (setq j (strlen sWork1))
    (while (and (< 0 j) (/= (setq sWork2 (substr sWork1 j 1)) ","))
      (setq j (1- j))
    )
    (setq iWork1 (atoi (substr sWork1 (1+ j))))
    (if (= (Boole 1 iWork1 1) 1)
      (setq bLoopFlg nil)
      (setq i (1+ i))
    )
  )
  i
)


;--------------------------------------------------------------------------
; めいん
;--------------------------------------------------------------------------
  (setq fTanBase 1000.0)
  (setq sDefaultNo "0")

  (setq X car)
  (setq Y cadr)

  ; 用紙サイズのリスト
  (setq lsPaList '(("A0" . 841.0) ("A1" . 594.0) ("A2" . 420.0) ("A3" . 297.0) ("A4" . 210.0)))

  (and
    ; キロ程記号のブロック名を設定
    (setq sBasePath (findfile "BaseLine.Lsp"))
    (setq sBasePath (Epathname sBasePath))
    (setq sDataFile "kirodat.dat")
    (setq sEditorName (findfile "QEDIT.EXE"))
    (setq sInsDwg1 (findfile "Dwg¥¥Qpkr0000.Dwg"))          ; 1kmごと
    (setq sInsDwg2 (findfile "Dwg¥¥Qpkr0001.Dwg"))          ; 500mごと
    (setq sInsDwg3 (findfile "Dwg¥¥Qpkr0002.Dwg"))          ; 100mごと

    (setq bLoopFlg T)
    (while bLoopFlg
      (textscr)
      (princ "¥n¥n【電車線キロ行程、基準線作成】")
      (princ "¥n¥n    1.基準線のデータ作成")
      (princ "¥n¥n    2.作成済データの修正")
      (princ "¥n¥n    3.基準線作図")
      (princ "¥n¥n    0.終了")
      (princ (strcat "¥n¥n番号を入力 <" sDefaultNo ">: "))
      (initget "1 2 3 0")
      (if (not (setq sInputNo (getkword)))
        (setq sInputNo sDefaultNo)
      )
      (cond
        ; データ作成
        ((= sInputNo "1")
          (and
            (setq lsWriteData (inputBaseLineData))
            (ListDataWrite (strcat sBasePath sDataFile) lsWriteData 0 (1- (length lsWriteData)) 0)
          )
        )
        ; データ修正
        ((= sInputNo "2")
          (and
            (if (not (findfile sDataFile))
              (alert "基準線のデータが作成されていません")
              T
            )
            (progn
              (setq sMenuID (#GetPrivateProfileString "MenuID" "QEdit" (findfile "QCITE.INI")))
              (if (/= "" sMenuID)
                (progn
                  (princ "¥n¥n【縦断線キロ行程、基準線修正】")
                  (princ "¥n¥n    テキストエディタを使用して作成データを修正します")
                  (princ "¥n¥n    修正が終了したら更新終了して終了してください")
                  (initget "Y N")
                  (if (/= (getkword "¥n¥nよろしいですか <Y>: ") "N")
                    (#WaitProcess sEditorName (strcat "/F=" sBasePath sDataFile " /ID=" sMenuID))
                  )
                  (graphscr)
                )
              )
              T
            )
          )
        )
        ; 基準線作図
        ((= sInputNo "3")
          (and
            ; ファイルから基準線のデータを読み込む
            (if (not (findfile sDataFile))
              (alert "基準線のデータが作成されていません")
              (setq lsReadList (ListDataRead (strcat sBasePath sDataFile)))
            )
            (progn
              ; 基準線の作成データを表示する
              (princ "¥n¥n【縦断線キロ行程、基準線作図】")
              (princ "¥n¥n    キロ行程の開始値 : ")
              (setq fWork1 (atof (nth 3 lsReadList)))
              (if (<= 1000.0 fWork1)
                (princ (strcat (itoa (fix (/ fWork1 1000.0))) "K" (itoa (rem (fix (/ fWork1 100.0)) 10)) "00M"))

                (princ (strcat "K" (itoa (rem (fix (/ fWork1 100.0)) 10)) "00M"))
              )
              (setq fWork1 (atof (nth 3 lsReadList)))
              (princ (strcat "¥n¥n    キロ行程のオフセット距離 : " (DelLastZero fWork1) "M"))
              (princ "¥n¥n    測量の尺度 : ")
              (setq fWork1 (atof (nth 0 lsReadList)))
              (princ (DelLastZero fWork1))
              (princ "¥n¥n    用紙のサイズ : ")
              (setq fWork1 (strcase (nth 1 lsReadList)))
              (if (assoc fWork1 lsPaList)
                (princ (strcat fWork1 "(高さ" (DelLastZero (cdr (assoc fWork1 lsPaList))) "mm"))
                (princ (strcat "(高さ" (DelLastZero (atof fWork1)) "mm"))
              )
              (if (not (#GetDwgInfo))
                ; 図面がセットアップされているとき
                (if (not (setq ptWork1 (getpoint "¥n基準線を作図する左下点を指示 <0,0>: ")))
                  (setq ptWork1 '(0 0 0))
                )
                ; 図面がセットアップされていないとき
                (progn
                  (initget "L")
                  (setq ptWork1 (getpoint "¥n基準線を作図する左下点を指示(L=設定図面の左下点) <0,0>: "))
                  (if (or (= ptWork1 "L") (not ptWork1))
                    (setq ptWork1 (getvar "LIMMIN"))
                  )
                )
              )
              (graphscr)
              (#popvar)
              (#pushvar "CMDECHO" 0)
              (#pushvar "BLIPMODE" 0)
              (#pushvar "TEXTSTYLE" "SIMPLEX")
              (DrawBaseLine lsReadList ptWork1)
              (#popvar)
```

```
                T
              )
            )
          )
          ; 終了
          (if (= sInputNo "Q")
            (graphscr)
            (setq bLoopFlg nil)
          )
        )
      )
    )
  (princ)
)


(defun c:BaseLine2
  (
    /
    iSrtKiro iEndKiro         ; キロ程の開始値と終了値
    fTxtWidth               ; キロ程同士の間隔

    sLineLay0 sLineLay1 sLineLay2
    sLineLay3 sLineLay4 sLineLay5   ; 線分の作図画層

    fTextH                  ; 文字高
    fTextOffX fTextOffY          ; 文字列のオフセット間隔

    ptP0 fBaseOffX fBaseOffY       ; 基準点、基準点からのオフセット値

    fOffY00 fOffY01 fOffY02 fOffY03
    fOffY04 fOffY05 fOffY06 fOffY07
    fOffY08 fOffY09 fOffY10 fOffY11
    fOffY12 fOffY13 fOffY14        ; 各基準線の間隔
    fOffSum

    ptP1 ptP2 ptP3 ptP4          ; 基準線を作図するときの点
    ptP5 ptP6 ptP7 ptP8
    ptTP0 ptTP1               ; 文字列を作図する点

    i                    ; カウンタ変数

    X Y
  )

  (setq X car)
  (setq Y cadr)

  ; 作図画層の設定
  (setq sLineLay0 "TEXT")
  (setq sLineLay1 "SYMBOL")
  (setq sLineLay2 "THIN")
  (setq sLineLay3 "HOJO")
  (setq sLineLay4 "RAIL")
  (setq sLineLay5 "BUILD")

  ; 文字列の作図設定
  (setq fTextH (* 2.5 setup:rscl))
  (setq fTextOffX (* 1.0 setup:rscl))
  (setq fTextOffY (* 0.5 setup:rscl))

  ; 基準点との間隔
  (setq fBaseOffX (* 1.0 setup:rscl))
  (setq fBaseOffY (* 1.0 setup:rscl))

  ; 基準線の間隔
  (setq fOffY00 (* 5.0 setup:rscl))
  (setq fOffY01 (* 10.0 setup:rscl))
  (setq fOffY02 (* 5.0 setup:rscl))
  (setq fOffY03 (* 10.0 setup:rscl))
  (setq fOffY04 (* 40.0 setup:rscl))
  (setq fOffY05 (* 10.0 setup:rscl))
  (setq fOffY06 (* 10.0 setup:rscl))
  (setq fOffY07 (* 40.0 setup:rscl))
  (setq fOffY08 (* 10.0 setup:rscl))
  (setq fOffY09 (* 40.0 setup:rscl))
  (setq fOffY10 (* 10.0 setup:rscl))
  (setq fOffY11 (* 20.0 setup:rscl))
  (setq fOffY12 (* 10.0 setup:rscl))
  (setq fOffY13 (* 10.0 setup:rscl))
  (setq fOffY14 (* 5.0 setup:rscl))
  (setq fOffSum (+ fOffY00 fOffY01 fOffY02 fOffY03 fOffY04 fOffY05 fOffY06 fOffY07
            fOffY08 fOffY09 fOffY10 fOffY11 fOffY12 fOffY13 fOffY14))

  (and
    ; 最初のキロ程を入力させる
    (progn
      (initget 4)
      (setq iSrtKiro (getint "¥n最初のキロ程を入力: "))
    )

    ; 最後のキロ程を入力させる
    (progn
      (initget 4)
      (setq iEndKiro (getint "¥n最後のキロ程を入力: "))
      (while (and iEndKiro (< iEndKiro iSrtKiro))
        (alert "最初のキロ程より大きな値を入力してください")
        (setq iEndKiro (getint "¥n最後のキロ程を入力"))
      )
      iEndKiro
    )

    ; 文字列の間隔を入力させる
    (progn
      (initget 6)
      (if (setq rTxtWidth (getreal "¥nキロ程文字列の間隔を入力 <40>: "))
        (setq rTxtWidth (* rTxtWidth setup:rscl))
        (setq rTxtWidth (* 40.0 setup:rscl))
      )
    )

    ; 基準点を指示させる
    (if (not (setq ptP0 (getpoint "¥n基準線を作図する位置を指示 <0,0>: ")))
      (setq ptP0 '(0 0))
      T
    )

    ; 基準線を作図する開始点を求める
    (setq ptP1 (list (+ (X ptP0) rBaseO[(X)) (+ (Y ptP0) rBaseO[(Y))))
    (setq ptP2 (list (X ptP1)                (+ (Y ptP1) rO[(Y00))))
    (setq ptP3 ptP1)
    (setq ptP4 ptP2)
    (setq ptP5 (list (X ptP1)                (+ (Y ptP1) rO[(Sum))))
    (setq ptP6 (list (X ptP5)                (- (Y ptP5) rO[(Y14))))
    (setq ptP7 ptP5)
    (setq ptP8 ptP6)

    (setq ptTP0 (list (+ (X ptP1) rTextO[(X)) (+ (Y ptP1) rTextO[(Y))))
    (setq ptTP1 (list (+ (X ptP6) rTextO[(X)) (+ (Y ptP6) rTextO[(Y))))

    (progn
      (*popvar)
      (*pushvar "CMDECHO" 0)
      (command "UNDO" "BE")
      (*pushvar "TEXTSTYLE" "SIMPLEX")
      (*pushvar "BLIPMODE" 0)

      ; 縦の基準線を作図する
      (setq i iSrtKiro)
      (while (<= i iEndKiro)
        (command "LINE" ptP3 ptP4 "")
        (*putlayer (entlast) sLineLay3)
        (command "LINE" ptP7 ptP8 "")
        (*putlayer (entlast) sLineLay3)
        (command "TEXT" ptTP0 rTextH 0 i)
        (*putlayer (entlast) sLineLay0)
        (command "TEXT" ptTP1 rTextH 0 i)
        (*putlayer (entlast) sLineLay0)

        (setq ptP3 (list (+ (X ptP3) rTxtWidth) (Y ptP3)))
        (setq ptP4 (list (X ptP3)               (Y ptP4)))
        (setq ptP7 (list (X ptP3)               (Y ptP7)))
        (setq ptP8 (list (X ptP3)               (Y ptP8)))
        (setq ptTP0 (list (+ (X ptTP0) rTxtWidth) (Y ptTP0)))
        (setq ptTP1 (list (X ptTP0)               (Y ptTP1)))
        (setq i (1+ i))
      )

      ; 横の基準線を作図する
      (setq ptP3 (list (- (X ptP3) rTxtWidth) (Y ptP1)))
      (setq ptP4 (list (- (X ptP4) rTxtWidth) (Y ptP2)))

      (command "LINE" ptP1 ptP3 "")
      (*putlayer (entlast) sLineLay3)
      (command "LINE" ptP2 ptP4 "")
      (*putlayer (entlast) sLineLay3)
```

```
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY01)))
        (setq ptP4 (list (X [illegible]) (Y ptP2)))
        (command "LINE" ptP[illegible]P4 "")
        (*putlayer (entlast) sLineLay2)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY02)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay2)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY03)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay3)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY04)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay5)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY05)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay5)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY06)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay4)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY07)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay4)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY08)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay4)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY09)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay4)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY10)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay5)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY11)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay5)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY12)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay2)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY13)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay3)
        (setq ptP2 (list (X ptP2) (+ (Y ptP2) fOffY14)))
        (setq ptP4 (list (X ptP4) (Y ptP2)))
        (command "LINE" ptP2 ptP4 "")
        (*putlayer (entlast) sLineLay3)

        (command "UNDO" "E")
        (*popvar)
       )
     )
  )
  (princ)
)
(princ)


;-------------------------------------------------------------------------------
; へんこうりれき
;-------------------------------------------------------------------------------
;
; 1997/08/13  Version 1.00
;       Quick数値編集力用コマンド
;       新規作成
;       By Kou
;
; 1997/09/01  Version 1.10
;       ゼロ入力できるところができなかったのを修正
;       基準線作図後、表示画面を作図図形に合せてズーム表示する
;       By Kou
;
; 1997/12/15  Version 1.20
;       コマンド名「BaseLine」を「BaseLine2」に変更
;       メニュー選択一覧の表示時に、テキストスクリーンに強制的に切替えるようにする
;       「KOOOM」を「OKOOOM」に変更する
;       By Kou
;
```