

**Report Generator
Programming Information**

For Timenet V3.0

1.1 Overview

The report generator enables users to write or change report formats using simple text files without any changes to the main program.

1.2 Introduction

In its simplest form, a report file is a file which is processed once for each employee on the system. Any simple text present is copied directly to the output, and any symbols found are replaced with their values and sent to the output.

Symbols are words preceded by the dollar sign (\$) and there are many predefined symbols containing information about the current employee. E.g. \$empl_surname is a symbol which will be replaced by the actual surname of the current employee; \$curtime is a symbol containing the time at which the report is being generated.

Therefore, a report file to produce a list of all the employees could contain the line

```
$empl_num $empl_surname $empl_initials $empl_dname
```

Where

\$empl_num contains the current employee number

\$empl_surname contains the current employee surname

\$empl_initials contains the current employee initials

and \$empl_dname contains the current employee department name

This file would produce a report like this:

P12345	TURNER	JMW	Research and Development
P12346	CONSTABLE	J	Marketing
P12349	REYNOLDS	J	Accounts

1.3 File Structure

To make the report generator file more useful, we need to be able to add text which is only output at the beginning of the report, rather than for every employee, such as a general heading. Adding these extra instructions produces a fully functioning report file:

```

1:   Employee Listing
2:   $ask_empl=1
3:   $$start
4:   Employee Listing Report $curdate
5:
6:   Number   Surname   Initials   Department
7:   $$main
8:   $empl_num $empl_surname $empl_initials $empl_dname
9:   $$end
10:  $formfeed

```

This produces a report that looks like:

```

Employee Listing Report 23/09/93

Number Surname Initials Department
P12345 TURNER JMW Research and Development
P12346 CONSTABLE J Marketing
P12349 REYNOLDS J Accounts

```

Line 1 defines the name of the report to be displayed on the menu.

Line 2 - setting ask_empl tells the program to allow a section of employees - single, dept or all

Line 3 - the \$\$start directive defines the beginning of a section containing heading information

Line 7 - the \$\$main directive defines the beginning of the main report code.

Line 8 - this is the same line as before, to produce the information required.

Line 9 - the \$\$end directive defines the start of a section containing any finishing information.

Line 10 - \$formfeed is symbol containing just the formfeed character and ensures an end of page.

1.4 Formatting

Symbol values are stored internally in a raw form and when they are encountered in the report file, they are output according to a predefined-defined format which depends on the type of value stored. For example, dates are stored internally as a large number representing the number of seconds since 1st January 1970. When output, this information can be converted in a number of different ways depending on the format selected. A suitable format for \$curdate would be DD-MM-YY which would generate an output of 23-9-93. The default format for any symbol can be overridden by specifying a new format using a colon and the format name.

E.G.

```

$new_format="MM-DD-YY"
$curdate:new_format

```

will produce the American style date format 9-23-93

1.5 Creating Symbols

Symbols are created on a special line that does not produce any output. The symbol name should be at the beginning of the line followed by the equals sign (=) and the value which may be a string or a value.

E.G.

```
$count=1
$name="Bloggs"
```

When a symbol is created it uses the default format specification, \$value_format or \$string_format. You can attach a different format when it is created

E.G. if

```
$new_format="NNNN"
$count:new_format=0
```

then the line

```
***$count***
```

will produce

```
***0 ***
```

Symbols may only be made of alphabetic characters - numbers are NOT allowed as they are used for array indexing. Certain internal symbols are held as arrays - that is, as a list of values. You can use individual values by specifying the position in the list. For example, the rates associated with a clockout are held by the symbol \$RT as a list of rates from 0 to 8. To print the hours worked at rate 1 use the symbol \$RT1.

1.6 Calculations

It is possible to perform calculations on symbols by using mathematical expressions.

E.G.

```
$a=1
$b=2
$c=$a+$b
```

This will set the symbol \$c to 3. You can also use multiply "*", divide "/", minus '-'. Section 4. gives a full list of expressions possible. The calculations use normal mathematical priorities (multiply before addition etc) but you can use brackets to change the order:

```
$a=1+2*3    sets $a to 7
$a=(1+2)*3  sets $a to 9
```

1.7 Arrays

2. Format Definitions

When a symbol is encountered in the report generator file, its value is sent to the output using a defined format string. The format string is a set of characters which specify precisely how the value is to be represented. The characters in the format string can have different meanings depending on the type of value being produced (value, string, time etc).

The format specifiers are repeated to specify the default length of the output - for example, values printed with a format "NNNN" will always occupy at least 4 characters and will be padded with spaces if smaller. If the format is preceded with a zero, they will be padded with zeros. If followed by an exclamation mark, they will be truncated if longer than the specified number of characters.

E.g.

123 printed with format "N" will produce "123"
123 printed with format "NNNN" will produce " 123"
123 printed with format "ONNNN" will produce "0123"
123 printed with format "NN!" will produce " 12"

Any characters in the format string that are not recognised as formatting characters will be output unchanged.

E.g. 123 printed with format "<N>" will produce "<123>"

2.1 String specifiers

Symbols can be set to hold string values by enclosing text in quotation marks - \$test="ABCD". The string can be output to the left, to the right, or in the centre of the specified area:

L Left justify string

R Right justify string

C Centre string

E.g.

"ABCD" printed with format "LLLLLL" will produce "ABCD "
"ABCD" printed with format "RRRRRR" will produce " ABCD"
"ABCD" printed with format "CCCCCC" will produce " ABCD "

2.2 Value Specifiers

Most symbols hold values which are stored as 32 bit signed numbers. These numbers may be printed out in a number of different ways depending on the format:

- N Normal - print a normal value
- H Hours - treat the value as hours:minutes and print just hours
- i.e. format value/60 as specified
- R rates - treat the value as hours and minutes and print just hours
- i.e. format value/60 as specified
However, if the whole value is zero, print as just spaces
- M minutes - treat the value as hours and minutes and print just minutes
- i.e. format value%60 as specified
- X minutes - treat the value as hours and minutes and print just minutes in decimal
- i.e. format (value%60)*100/60 as specified
- B BCD - print the value as a BCD (or hexadecimal) number.
- K Card code
- L Pounds - treat the value as an amount stored in pence and print the numbers of pounds
- P Pence - treat the value as an amount stored in pence and print the number of pence
- C Calendar - treat the value as a calendar absence code. These are printed as a single character representing the absence code with type zero absences shown by a dot.
- D Decimal places
Treat the number as decimal using the number of digits specified.

E.g.

186 printed with format "N" will produce "186"
186 printed with format "HH:0MM" will produce " 3:06"
186 printed with format "H.0XX" will produce "3.10"
186 printed with format "L.0PP" will produce "£1.86"
186 printed with format "D.DD" will produce "1.86"

2.3 Time/Date specifiers

Values that are stored as time/date values are held as 32 bit numbers containing the number of seconds since 1st January 1970. Any part of the date information can be extracted using the appropriate format character. These values which are also used for clocking information, have extra status information which can be used to alter the appearance of the data (bold, underlined, etc)

The examples all show the output produced when formatting the time/date 9:30 1/10/93

Y Year - extract the year number. It will automatically be truncated to two digits and padded with zero if necessary.

E.g. printing with format "Y" will produce "93"

E.g. printing with format "YY" will produce "93"

E.g. printing with format "YYYY" will produce "1993"

N Numeric month - extract the month number and print as digits

E.g. printing with format "DD/NN/YY" will produce " 1/10/93"

A Alphabetic month - extract the month number and print as text

E.g. format "D A Y" will produce "1 October 93"

E.g. format "D-AAA!-Y" will produce "1-Oct-93"

D Day - extract the day of the month and print as digits

T 'Th' - use the day number to print the suffix to a day (eg 5th)

E.g. format "DT A Y" will produce "1st October 93"

W Weekday - extract the day of the week and print as text

E.g. "W DT A Y" will produce "Friday 1st October 93"

H Hours - extract the hour number and print as digits. If the status is 'invalid', print as XX (e.g. not clocked out)

E.g. "HH:MM" will produce " 9:30"

M Minutes - extract the minute number and print as digits. If the status is 'invalid', print as XX (e.g. not clocked out)

E.g. "HH:MM" will produce " 9:30"

X Decimal minutes - extract the minute number and print as decimal digits. If the status is 'invalid', print as XX (e.g. not clocked out)

E.g. "HH.XX" will produce " 9.50"

S Seconds - extract the second number and print as decimal digits.

E.g. "HH:MM:SS" will produce " 9:50:59"

- To show if the status indicates a clockout, prefix with a -

E.g. "-0HH:MM" will produce "-09:55" if clocking out
or " 09:55" if clocking in.

C Insert format Code. If the status indicates a special case, insert the appropriate printer code. The code will toggle between on and off each time this character is encountered so there should always be an even number within a format string.

E.g. "CHH:MMC" will produce "{LATEON} 9:30{LATEOFF}"
if the time status indicates a lateness violation

The printer codes and screen colours produced for each special case can be specified within TIMENET.

The special cases available are:

Lateness
Core Time violation
Not clocked out
Auto clocking
Business In/Out
Clock In
Clock Out
Edited clocking

Z Weekday shown as a number 0-6 (Monday = 0)

h The same as H but ignores error codes.

3. PREDEFINED SYMBOLS

A large number of symbols have been defined within the report generator and set to values corresponding to the employee or other relevant information. These can be accessed at any point in the report generator file. There are also special symbols which affect how the report generator works or request specific operations during report generation.

3.1 Control Symbols

Control symbols affect the way the report generator operates. To take effect, they must be set before the `$$start` directive.

\$numperline Type: Numeric Max: 8 Min: 0 Default: 4

This symbol sets the maximum number of clockings reported on each line. If it is set to zero, the records are not read - this will increase speed for reports which are not concerned with clockings such as the employee listing.

e.g. `$numperline=8`

\$directory Type: String Default: ""

Setting this string informs the report generator that this file specifies a sub-menu, not a report. The value should be set to the name of a directory containing further report generator files.

e.g. `$directory="\reports\extras"`

\$ask_destination Type: value Default: 0

Controls the selection of output destination:

0	Same as 3
1	Output to screen
2	Output to printer
4	Output to export file

The destinations may be added together (e.g `$ask_destination=5` allows display or export)

\$ask_empl Type: value Default: 0

Set to 1 to instruct the report generator to allow selection of employee.

\$ask_jobs Type: value Default: 0

Set to 1 to instruct the report generator to allow selection of job numbers.
Either `ask_empl` or `ask_jobs` may be set for any report but not both.

\$ask_period Type: value Default: 0

Controls the selection of report date range:

- 0 No date range used
- 1 Provide a menu entry for selection
- 2 unused
- 3 Current Week
- 4 Previous week
- 5 Current Day
- 6 Yesterday
- 7 Start of employee/job
- 8 Current Pay Period
- 9 Previous pay period
- 10 Date Range

If \$ask_period is set to date range, \$reqbegdate and reqenddate may be set to specify the default period.

\$ask_pay Type: value Default: 0

Set to 1 to instruct the report generator to allow selection of pay period.

\$ask_sort Type: value Default: 0

Selects the method of sorting of employees/jobs:

Time/Attendance

- 0 No sorting
- 1 Prompt the user for selection of sort by employee name/number
- 2 Sort on Employee Number
- 3 Sort on Employee Name
- 4 Sort on Employee Department then employee Number
- 5 Sort on Employee Department then employee Name
- 6 Sort on workgroup then employee Number
- 7 Sort on workgroup then employee Name
- 8 Sort on fire department then employee number
- 9 Sort on fire department then employee name

Data Collection:

These settings select how data collection records are sorted within each job

- 21 Sort records on time
- 31-39 Sort records on Operation number, etc
- 42 Sort records on Employee number
- 43 Sort records on Employee name
- 44 Sort records on Department then Employee Number
- 45 Sort records on Department then Employee name
- 46 Sort records on Workgroup then Employee Number
- 47 Sort records on Workgroup then Employee name

- 61-69 As 0-9 used within \$\$recsort
To produce separate in and out clockings (need TA & DC records enabled)

\$en_deleted Type: value Default: 0

Set to 1 to instruct the report generator to include deleted employees in reports.

\$ask_absence Type: value Default: 0

Set to 1 to instruct the report generator to allow selection of absence reasons.

\$delimiters Type: string Default: "\$;~"

This can be set to change special characters used by the generator. It would not normally be used. The first character (\$) is the character used to indicate symbols and directives. The second character (;) is the character used to indicate comments. The third character (~) is the character used for padding.

\$records Type: value Default: 1

1 Time/Attendance Records
2 Data Collection Records
4 System records (access control etc)
+16 for original (un-edited records)
+32 for all records (including deleted)
+256 Only show in-clockings
+512 Only show out-clockings
+1024 Only show bad clockings

\$dc Type: value Default: 0

Set to 1 to indicate data collection report

\$enable Type: value Default: 1

Set to 0 to disable this report. If disabled, the report name will not appear in the report menu. It may be set to the result of a calculation, for example, to only enable a report if a certain field is set.

\$screen Type: value Default: 0

Set to 1 to enable an immediate mode screen display. This mode does not generate a normal report but instead allows the screen to continuously updated for attendance panels displays etc. See also \$\$display, \$\$pause, \$\$getchar.

\$reqbegdate Type: string

If ask_period has been set to date-range, this can be set to specify the starting date for the report. It may be an absolute date in dd/mm/yy format or an offset as given in "Winrep Command Line Parameters".

\$reqenddate Type: string

If ask_period has been set to date-range, this can be set to specify the ending date for the report. It may be an absolute date in dd/mm/yy format or an offset as given in "Winrep Command Line Parameters".

3.2 Format Symbols

The format symbols provide default formatting for other symbols. E.g. the `value_format` symbol specifies the formatting to be used when outputting any values that do not have a specified format. If the formats are changed, all outputs using the format will be affected.

Some format symbols are set to different values depending on settings within Timenet. For example, default time formats are set differently depending on whether decimal times and rates are selected. Note that these settings only affect the default formats and can be overridden in the report generator files.

\$error_format Type: string Default: "?????"

This symbol is output when an unrecognised symbol or error is encountered.

\$date_format Type: string Default: "ODD/ONN/OYY"

This is the default format for dates. E.g. 23/09/93

\$value_format Type: string Default: "N"

This is the default format for all values.

\$string_format Type: string Default: "L"

This is the default format for all strings.

\$clock_format Type: string Default (decimal): "-CHH.XXC" (minutes): "-CHH:MMC"

This is the default format for all clockings.

\$time_format Type: string Default (decimal): "HH.XX" (minutes): "HH:MM"

This is the default format for all times.

\$rate_format Type: string Default (decimal): "RR.XX" (minutes): "RR:MM"

This is the default format for rates.

\$totl_format Type: string Default (decimal): "HHH.XX" (minutes): "HHH:MM"

This is the default format for rate totals.

\$gtot_format Type: string Default (decimal): "HHHH.XX" (minutes): "HHHH:MM"

This is the default format for rate grand totals

\$cal_format Type: string Default: "C"

This is the default format for calendar entries.

3.3 General Symbols

\$curtime	Type: Time	Current time
\$curdate	Type: Date	Current date
\$begdate	Type: Date	Report start date
\$enddate	Type: Date	Report end date
\$yeardate	Type: Date	Date of the start of year
\$lineno	Type: Value	Current Line number
\$pageno	Type: Value	Current Page number
\$version	Type: String	Version number "1.10"
\$version_date	Type: String	Date of version "01-Oct-93"
\$zero_rates	Type: Rate	Array of rates, all zero - used to initialise new symbols
\$parma	Type: Value	Internal symbol
\$parmb	Type: Value	Internal Symbol
\$site	Type: Value	The site number
\$weekno	Type: Value	The week number from the beginning of year
\$cflags	Type: Value	1=Data collection, 2=T/A, 4=Access enabled (additive)
\$hol_hours	Type: Value	Set to 1 if holidays are calculated in hours
\$numsites	Type: Value	Set to the number of sites set up
\$html_on	Type: Value	Set to indicate html formatting already present.
\$html_off	Type: Value	Set to indicate html formatting needed (default).
\$bands	Type: String[]	List of names for Pay Bands
\$form	Type: String{}	List of parameters from an html form

Printer Control Symbols

\$formfeed	Type: string	Formfeed (new page)
\$end_screen	Type: String	Force new page on display screen
\$bold_on	Type: String	Turn on bold (emphasized) printer typeface
\$bold_off	Type: String	Turn off bold (emphasized) printer typeface
\$italic_on	Type: String	Turn on italic typeface
\$italic_off	Type: String	Turn off italic typeface
\$ul_on	Type: String	Turn on underlining
\$ul_off	Type: String	Turn off underlining
\$pr_condensed	Type: String	Select condensed printing
\$pr_normal	Type: String	Select normal printing
\$pr_expanded	Type: String	Select expanded printing
\$bar	Type: String	Generate barcode Usage: \$bar,ht,wt<string> generates a barcode with height of ht in mm/10 and a barcode width of wt in mm/10 mm. The string is printed as a barcode. E.G. \$bar,80,4<\$j_name>

3.4 Directives

Directives instruct the report generator to take specific actions. Directives begin with two dollar signs and **MUST** be at the start of a line.

\$\$start

Start indicates the beginning of the start code. Code between the start and main directives is only executed once before generation of the main report. This is normally used to provide overall headings or initialise variables.

\$\$main

Main indicates the beginning of the main code. Code between the main and end directives is executed once for each employee selected.

\$\$end

End indicates the beginning of the end code. Code between the end directive and the end of file is only executed once after generation of the main report. This is normally used to output accumulated totals.

\$\$if expression

Lines following an 'if' directive are only processed if the expression was true (non-zero). Processing is restored after 'else' or 'endif' directives.

\$\$else

The else directive must only appear after an 'if' directive and causes the processing to be restored if the 'if' expression was false

\$\$elseif expression

The elseif directive must only appear after an if directive and causes a new if clause to be started only if the original 'if' expression was false.

\$\$endif

Finishes an if block. Normal processing is resumed after this directive.

\$\$while expression

The while directive causes the lines between the 'while' and the following 'endwhile' to be repeated until expression is false (zero). If the expression is initially zero, the lines will not be processed at all.

\$\$whilerecords

The whilerecords directive causes the lines between the 'whilerecords' and the following 'endwhile' to be repeated until all clocking records have been processed. Each time the whilerecords loop repeats, the records stored in symbols are advanced by numberline records.

\$\$endwhile

The endwhile directive ends the section of lines to be repeated in a while or whilerecords loop.

\$\$fetchempl expression val1 val2

Fetches the full information structure about the selected employee.

If expression is zero, fetch employee information for currently selected employee

If expression is positive, use as the employee index number

In either case, val1 specifies the calendar month number and the text info line number which is stored in \$empl_text

If expression is -1, fetch the next employee from the sorted list

if val1 is greater than or equal to 0 reset and re-sort the list of employees (val2 = site):

0 = no sort

2 = Employee number

3 = Employee name

4 = Department then Employee number

5 = Department then Employee name

6 = Workgroup then Employee number

7 = Workgroup then Employee name

8 = Fire Department then Employee number

9 = Fire Department then Employee name

\$\$storeempl 0 val1 text

Updates the extra information fields for the current employee

val1 = information field number

text = text to be stored (39 characters max)

\$\$fetchshift expression

The fetchshift directive fetches information about a single shift and stores it in the shift variables. The expression is formed by multiplying the shift number by 7 and adding the day number (Monday=0, Sunday=6). Thus all shifts can be processed by adding one to the expression each time.

\$\$fetchdept expression

The fetchdept directive fetches information about a department and stores it in the department variables (see 3.7)

\$\$fetchrecord val1 val2

Controls the fetching of records:

If val1 < 0, rewind and start again (does not fetch a record) - if val2 > 0, set numberline to val2
if val1 >= 0, fetch next line of records

For direct screen mode:

if val1 = 0, fetch next record (or RC_END if none available)
if val1 > 0, rewind back by val1 seconds

\$\$recsort val1 val2

Sorts the current list of records by val1, then val2

val1, val2 can be 0-9 for job number, opno, etc or 41 upwards for employee sort, then job
(Use 60-69 rather than 0-9 to prevent adding together all records for employees, must also have both TA & DC records selected).

\$\$fetchterminal expression eventnum

The fetchterminal directive fetches information about a terminal and stores it in the terminal variables (see 3.7). Information on programmed events 1-16 can be retrieved by setting eventnum.

\$\$fetchabsence expression

The fetchabsence directive fetches information about an absence reason and stores it in the absence variables (see 3.10)

\$\$fetchjob jobno field

The fetchjob directive fetches the details about specified job field (jobno = 0, opno = 1 etc) and stores it in the job variables (See section 3.7). Jobno should be set to the actual job index number, as reported in the clocking information. If jobno is zero, it fetches the next job from a sorted list of jobs.

\$\$fetchhr mode, tab, control

Fetch information from the human resources file

Tab and control can be numeric, in which case it returns the appropriate control information directly, or can be text, in which case it searches for the named tab and control

The results are returned in the variables:

\$hr_tab	Set to the name of the specified tab
\$hr_control	Set to the name of the specified control
\$hr_val	Set to the value of the specified control

E.G.

\$\$fetchhr 0, 1, 2 Returns the details of the second control on the first tab

\$\$fetchhr 0, "Medical", "Doctor" Returns the value of the control named "Doctor" on the tab called "Medical"

\$\$fetchsite site_number

Fetch information for the specified site

\$\$filename filename mode

This instructs the report generator to write the output a specified filename. This is normally used to specify a fixed filename for exports. If mode is set to zero, any existing file with the specified name is deleted first. If mode is non-zero, the output is appended to an existing file.

e.g. \$\$filename "f:\123\data.prn" 0

\$\$readparm mode, section, name, \$parm [, default [, filename]]

This reads or writes from your INI file.

mode=1 for read, mode=2 for write

section is the name of the section within the ini file (usually INIT)

name is the name of the parameter to read or write

\$parm is the variable that contains or will be set to the value in the INI file

default is the value that \$parm will be set to if the name is not found in the file (not used for write)

filename is the name of the file to be read/written

(if not specified C:\LWS\BIN\ADP.INI is used for the attendance panel
and C:\LWS\BIN\WINREP.INI is used for winrep))

E.G.

If ADP.INI contains:

```
[INIT]
TEST=abcd
```

then

\$\$readparm 1, "INIT", "TEST", \$val Sets \$val to "abcd"

\$\$findinfo name jobindex startn

Findinfo searches the list of extra information for the specified job. If an entry is found which matches the specified name, the variables \$j_extra... are loaded with the information from that field. A number may be specified instead of name to read directly from the specified field number. If startn is specified, the search will only start from that field number.

Variables loaded are:

\$j_extra1 Set to the name of the information field
 \$j_extra Set to the contents of the information field

If the field has been split into sections, they can be accessed by the variables

\$j_extraA Set to the contents of the first section
 ...
 \$j_extraE Set to the contents of the fifth section

E.g. \$\$findinfo Customer \$j_index 8 Find the first information field 8 or above which matches the name Customer

 \$\$findinfo 32 \$j_index 0 Read the information field 32

\$\$select initialn title option1 option2 ... optionn

Display a popup list containing the list of options and allow the user to select one option. The variable \$selval is set to the option number selected and \$selstr is set to the option selected. Title may be used to give the pop-up a heading. initialn specifies the option highlighted when the popup first appears.

Note Select must only be used within the start section (between \$\$start and \$\$main)

E.g. \$\$select 3 "Select Day" Monday Tuesday Wednesday Thursday Friday Saturday Sunday

 This displays a list of days with the cursor initially on Wednesday. If the cursor is moved to Saturday and enter pressed, \$selval will be set to 6 and \$selstr will be set to "Saturday"

\$\$input title

Pops up a window prompting for the user to type in data. Title may be used to provide a heading and prompt for the window. The variable \$instr is loaded with the data entered.

E.g. \$\$input "Enter Job Number"

\$\$printer expression

Directs all subsequent printed output to the terminal number specified.

\$\$array \$name, Num_bytes, Num_elements[, Num_rows]

Create an array symbol:

\$name	name of the array
Num_bytes	size of each element (in bytes)
Num_elements	Number of elements (values) in array
Num_rows	Number of rows (for 2 dimensional arrays)

E.G.

\$\$array \$table, 2, 10,4 Create an array of 2 byte (integer) values with 10 entries by 4 rows

Each entry in the array can be accessed using the index notation .e.g. \$table2 returns the third entry
Note that array numbering starts at zero.

To access two dimensional arrays, the index is specified as row and column separated by a decimal point

e.g. \$table2.3

\$\$import filename, array, mode

Read the specified file into the array. The file must contain comma separated data (csv)

Mode is not presently used and will be ignored

If there are too many columns of data in the file, the rest will be ignored.

If there are too many lines of data for the size of the array, an error is generated.

E.G.

```
$$array $table, 4, 5, 20
$$import "data.csv", $array, 0
```

\$\$sort array, mode, sorta[, sortb[, sortc[, sortd]]]

Sort the rows of the array according to the specified conditions

Each sort parameter is a five digit number abbcc where:

a = 0 for normal order, =1 for reverse order

bb = sort code 01=alphabetic, 02=alpha no case, 03=numeric, 04=date

cc = column

E.G.

\$\$sort \$table, 0, 00102, 10405 (alpha sort on column2, then reverse order sort column 5 on date)

\$\$search result, array, startrow, column, mode, "data"

Search the array for the specified string in the specified column.

The result variable is set to the row number

Mode: 1=match, 2=starts with, 3 = anywhere

E.G.

```
$$search $val, $table, 5, 2, 2, "ABC"
```

searches table for strings beginning with the letters ABC in column 2 and from row 5 onwards.

\$text <mode, parm1..parmn>

Change text setting for windows reports. These can be placed at any point within the rpg and do not need to be at the start of a new line

\$text<font, fontname>	Select a specific font
\$text>font, pointsize>	Change the font size
\$text<tcoulor, redn, grn, bluen>	Change the text colour - each colour 0-255
\$text	Restore any of the above font settings
\$text<align, mode>	Mode = left / centre / right
\$text</align>	Restore previous alignment
\$text<image, filename, w, h>	Display specified image on page If w<0 display actual size else stretch to fit [w,h]
\$text<bgimage, filename>	Displace specified image as background of page
\$text<hr>	Draw a horizontal line

\$\$tabset num, tab1, tab2, ... tabn

Set tabstops on page.

Num contains number of tabstops to set

tab1 is size of tab. If preceded by +, text will be centred within the tab
 If preceded by -, text will be left justified (normal)
 else text will be right justified.

\$\$include "filename", mode, filetype

Includes the specified file at this point in the report

mode	0	include all of file
	1	include header only (only applicable to html files)
	2	include body only "
	3	include tail only "
filetype	0	auto-detect
	1	rpg
	2	html
	3	rtf

\$\$graph Gtype, Gtitle, Xtitle, Ytitle, Ymin, Ymax, Yinc, Ydiv, Xpts, Xaxis, L1name, L1colour, L1vals

Generate a graph

Gtype: Type of graph
 0 = 2D line, 1 = 2D Bar chart 2 = 2D Pie chart
 3 = 3D Line 4 = 3D Bar 5 = 3D Pie
 6 = Sphere 7 = 2D Area 8 = 3D Area

Gtitle Title of graph
 Xtitle Title of x-axis
 Yaxis: Title of y-axis
 Ymin: Minimum (lower) value of y-axis
 Ymax: Maximum (upper) value of y-axis
 Yinc: Increment value for y-axis divisions
 Ydiv: Divisor for y-axis

Xpts: Number of points to display on x-axis
 Xaxis: String of labels for x-axis E.g. "Mon Tue Wed Thu Fri Sat Sun"

L1name: Name for first series
 L1colour: Colour for first series
 L1vals: First series values to be plotted (array of values)

... Up to 8 series may be specified

In DOS:

- (1) only Gtypes 0,1 or 2 may be used.
- (2) L1colour will be ignored

In Windows:

- (1) Yinc will be ignored.
- (2) Pie and Sphere graphs require two lines to be specified. The second series will be used as Series1 colours.

E.g. The following lines will produce a bar-chart graph increasing by one each day of the week:

```

$$array $vals 2 7
$vals0=0
$vals1=1
$vals2=2
$vals3=3
$vals4=4
$vals5=5
$vals6=6
$$graph 1 "Graph Title", "Day of Week", "No. of Days", 0, 6, 1, 1, "Mon Tue Wed Thu Fri Sat Sun", "Line 1", 255, $vals
    
```

\$\$function functionname

This defines the start of a function body. This must be within the \$\$start section

\$\$return

This defines the end of a function body and must follow a \$\$function directive

\$\$call functionname

Calls a block of code with the name functionname. When \$\$return is encountered, control is returned to the line following the \$\$call.

E.G.

```
$$start
;
$$function demo
This is a demo
$$return
;
$$main
;
abcd
$$call demo
efgh
;
$$end
```

will produce the lines

```
abcd
This is a demo
efgh
```

Immediate Mode Screen Control

These directives control the output to the display screen when in immediate screen mode. If `$$screen` is not set, these directives will be ignored.

\$\$display mode x_val y_val

Controls the screen display.

Note - modes 1-4 are used only for DOS versions, and modes 6,7 only for Windows

mode = 1 Position the cursor at x_val, y_val (1,1 is at the top left corner of the screen)

mode = 2 Clear a portion of the screen
 x_val = 1, Clear to end of line
 x_val = 2, Clear to end of screen

mode = 3 Set screen colour
 x_val = colour of text
 y_val = colour of background

mode = 4 Scroll a portion of the screen from current cursor position
 x_val = number of lines to scroll

mode = 6 Update the screen

mode = 7 Re-position at start of screen

mode = 11 Position the cursor at pixel x_val, y_val (1,1 is at the top left corner of the screen)

\$\$getchar waitval

Returns a key press from the keyboard in variable `$inchar`

If waitval is zero, does not wait for key if nothing pressed, but return -1.
Otherwise returns ascii value of next key press.

\$\$pause seconds

Pause suspends the display for the specified number of seconds or until a key is pressed.

\$_select <1,keyval,startn,string1,string2,...stringn>

Insert a drop-down selection box at the current position. String1 to Stringn define the items in the list and startn the item number that is displayed (initial selection).. When an item is selected from the list, a key press is simulated with a key value of keyval, followed by the item number selected

E.G.

```
$_select<1,9999,3,"Monday","Tuesday","Wednesday", "Thursday", "Friday", "Saturday", "Sunday">
$$getchar 0
$$if ($inchar = 99)
$$getchar 1
Day of week changed - new day selected is $inchar
$$endif
```

will produce a drop-down list containing all the days of the week, with Wednesday initially shown. If the selection is then changed the new day of week number will be displayed.

\$_launch <keyval, val,text>

Add a hyperlink to the specified text.

When a mouse is clicked on the text, a key press is simulated with the value keyval, followed by the value val.

E.G.

```
$_launch<9998, 1, "test1">$_launch<9998, 2, "test2">$_launch<9998, 3, "test3">
$$getchar 0
$$if ($inchar = 9998)
$$getchar 1
Test message $inchar clicked
$$endif
```

\$\$execute 1, mode, reportname, arg1, ... argn

Run the specified report and pass it the arguments arg1 to argn.

If mode=0 wait for the program to finish before continuing with report

If mode=1 do not wait.

E.G. \$\$execute 1, 0, "ta\pay.rpg", "-rp", "3"

\$\$execute 2, mode, program, arg1, ... argn

Run the specified program and pass it the arguments arg1 to argn.

If mode=0 wait for the program to finish before continuing with report

If mode=1 do not wait.

E.G. \$\$execute 2, 1, "c:\lws\bin\announce", "test message"
The words "test message" should be spoken.

\$\$execute 4, mode, program, arg1, ... argn

Run the specified program and pass it the arguments arg1 to argn, after the report generator has finished. Mode is currently unused.

3.5 Employee Symbols

Employee symbols hold the information about the employee currently being reported.

\$empl_num	Format: String	Employee Number
\$empl_surname	Format: String	Employee surname
\$empl_initials	Format: String	Employee initials
\$empl_dept	Format: Value	Employee department number
\$empl_dname	Format: String	Name of employees department
\$empl_pp	Format: Value	Employee pay period
\$empl_pp_wday	Format: Value	Employee pay period day of week
\$empl_pp_mday	Format: Value	Employee pay period day of month
\$empl_tgroup	Format: Value	Group of valid terminals
\$empl_site	Format: Value	Employee site number
\$empl_shifts	Format: Value[]	List of valid shifts
\$empl_pay_mode	Format: Value	Pay mode
\$empl_status	Format: Value	Employee status
\$empl_emplcard	Format: Value	Employee card number
\$empl_jobcard	Format: Value	Employee job card
\$empl_acard	Format: Value	Employee access card
\$empl_group	Format: String	Name of employee workgroup
\$empl_mxcredit	Format: Rate	Maximum credit to carry forward for flexitime
\$empl_mxdebit	Format: Rate	Maximum debit to carry forward for flexitime
\$empl_bbf	Format: Rate	Balance brought forward from last pay period
\$empl_index	Format: Value	The index number of the current employee
\$record-count	Format: Value	Number of lines of records for this employee
\$empl_fire_dept	Format: Value	Employee fire department
\$empl_wkyrates	Format: Value[]	Weekly rates
\$empl_level	Format: Value	Access level
\$empl_ty_hols	Format: Value	Number of hours holiday this year
\$empl_ny_hols	Format: Value	Number of hours holiday next year
\$empl_slevel	Format: Value	Supervisor level
\$empl_sick	Format: Value[]	Number of hours sickness this/next year

Extra employee symbols (only available after a \$\$fetchempl)

\$empl_dob	Format: String	Date of Birth
\$empl_Aaddr	Format: String	Address Line 1
\$empl_Baddr	Format: String	Address Line 2
\$empl_Caddr	Format: String	Address Line 3
\$empl_Daddr	Format: String	Address Line 4
\$empl_tapay	Format: Value[]	Pay in pence for each T/A pay Band
\$empl_jbpay	Format: Value[]	Pay in pence for each D/C pay Band
\$empl_strtdate	Format: Date	Start Date
\$empl_finishdate	Format: Date	Finish Date
\$ec	Format: Calendar	Calendar entries (single month)
\$empl_text	Format: String	Employee Additional details (single line)
\$empl_jparms	Format: Value[]	Current job parameter details
\$empl_location	Format: Value[]	[0]= current location, [1] = current site
\$empl_holiday	Format: Value[]	holiday booked hours [0] = this year, [1] = next year
\$empl_tyabs	Format: Value[]	Number of hours absence booked this year [0-9]
\$empl_nyabs	Format: Value[]	Number of hours absence booked next year [0-9]
\$empl_forename	Format: String	Forename
\$empl_title	Format: String	Title
\$empl_jbtitle	Format: String	Job Title
\$empl_email	Format: String	Email address
\$empl_phone	Format: String	Phone number
\$empl_lupdate	Format: Date	Last pay date

3.6 Department Symbols

These symbols are only defined after a \$\$fetchdept has been performed

\$dp_status	Format: Value	Department status
\$dp_name	Format: String	Department name
\$dp_printer	Format: Value	Department printer
\$dp_code	Format: Value	Department code

3.7 Shift Symbols

These symbols are only defined after a \$\$fetchshift has been carried out.

\$sh_number	Format: Value	Shift number
\$sh_type	Format: Value	Shift type
\$sh_name	Format: String	Shift name
\$sh_desc	Format: String	Shift description
\$sh_status	Format: Value	Shift status
\$sh_mins_due	Format: Time	Minutes due
\$sh_day	Format: String	Day of week
\$sh_minlen	Format: Time	Minimum shift length
\$sh_start	Format: Time	Scheduled start time
\$sh_end	Format: Time	Scheduled end time
\$sh_lstart	Format: Time	Latest start time
\$sh_lend	Format: Time	Latest end time
\$sh_dlyrates	Format: Time[]	Daily rates
\$sh_hourtimes	Format: Time[]	Hourly rate times
\$sh_hourrates	Format: Value[]	Hourly rates
\$sh_inclen	Format: Value[]	Increments
\$sh_inctime	Format: Time[]	Increment band times
\$sh_gi_len	Format: Value[]	Grace Time In length
\$sh_gi	Format: Time[]	Grace Time In times
\$sh_go_len	Format: Value[]	Grace Time Out length
\$sh_go	Format: Time[]	Grace Time Out times
\$sh_br_len	Format: Value[]	Break Times length
\$sh_br_fr	Format: Time[]	Break Times From
\$sh_br_to	Format: Time[]	Break Times To
\$sh_br_type	Format: Value[]	Break Times type
\$sh_ai_val	Format: Value[]	Autoclocking In flag
\$sh_ai_time	Format: Time[]	Autoclocking In Times
\$sh_ai_at	Format: Time[]	Autoclocking In at times
\$sh_ao_val	Format: Value[]	Autoclocking Out flag
\$sh_ao_time	Format: Time[]	Autoclocking Out Times
\$sh_ao_at	Format: Time[]	Autoclocking Out at times
\$sh_jinclen	Format: Value[]	Job Increments
\$sh_jinctime	Format: Time[]	Job Increment band times
\$sh_jgi_len	Format: Value[]	Job Grace Time In length
\$sh_jgi	Format: Time[]	Job Grace Time In times
\$sh_jgo_len	Format: Value[]	Job Grace Time Out length
\$sh_jgo	Format: Time[]	Job Grace Time Out times
\$sh_co_fr	Format: Time[]	Core Times From
\$sh_co_to	Format: Time[]	Core Times To
\$sh_ex_from	Format: Time[]	Exclusion band start time
\$sh_ex_to	Format: Time[]	Exclusion band end time
\$sh_ex_mode	Format: Value[]	Exclusion band mode

3.8 Data Collection Symbols

These symbols are predefined:

\$JA	Format: String	Name of first Data Collection field (Job Number)
\$JA_type	Format: Value	Type of first DC field
\$JA_numchars	Format: Value	Number of characters allowed in 1st DC field
\$JA_inout	Format: Value	Direction of clocking for input in 1st DC field

The above symbols are repeated for each of the ten fields where \$JB represents the 2nd field (op number) etc.

\$EX_A	Format: String	Name of 1st extra information field
\$EX_B	Format: String	Name of 2nd extra information field
\$EX_C	Format: String	Name of 3rd extra information field
\$EX_D	Format: String	Name of 4th extra information field
...		
\$EX_Z	Format: String	Name of 26th extra information field

These symbols are only defined after a \$\$fetchjob has been performed:

\$j_index	Format: Value	The index number of this job
\$j_name	Format: String	The job name
\$j_field	Format: Value	The field type (0=job, 1=opno etc)
\$j_status	Format: Value	The job status (1=deleted, 2=completed)
\$j_value	Format: Value[]	The values associated with the job
\$j_rates	Format: Rates[]	The accumulated hours worked on the job
\$j_card	Format: Value	The card number allocated to the job
\$j_starttime	Format: Time	Time that the job was added
\$j_usetime	Format: Time	Time that the job was last used

\$j_infoA	Format: String	Information field 1
\$j_infoB	Format: String	Information field 2
\$j_infoC	Format: String	Information field 3
\$j_infoD	Format: String	Information field 4
...		
\$j_infoZ	Format: String	Information field 26

These symbols are only defined after a \$\$find_info has been performed:

\$j_extral	Format: String	The name of the information field
\$j_extra	Format: String	The contents of the information field
\$j_extraA	Format: String	The contents of the first section
\$j_extraB	Format: String	The contents of the second section
\$j_extraC	Format: String	The contents of the third section
\$j_extraD	Format: String	The contents of the fourth section
\$j_extraE	Format: String	The contents of the fifth section

3.9 Record Symbols

Every time an employee clocks in or out, and adjustment is made or an absence occurs, a record is written which contains details of the event. These records can be accessed using predefined symbols.

The following information is available for each clocking on the line

\$RC	Format: value[numperline]	Record type
\$MO	Format: value[numperline]	Record mode
\$EM	Format: value[numperline]	Employee number
\$SU	Format: value[numperline]	Supervisor number
\$TM	Format: clock[numperline]	Time of clocking
\$AD	Format: clock[numperline]	Adjusted time of clocking
\$PA	Format: value[numperline]	Clock parameter 1
\$PB	Format: value[numperline]	Clock parameter 2
\$PC	Format: value[numperline]	Clock parameter 3
\$PD	Format: value[numperline]	Clock parameter 4
\$PE	Format: value[numperline]	Clock parameter 5
\$PF	Format: value[numperline]	Clock parameter 6
\$PG	Format: value[numperline]	Clock parameter 7
\$PH	Format: value[numperline]	Clock parameter 8
\$PI	Format: value[numperline]	Clock parameter 9
\$PJ	Format: value[numperline]	Clock parameter 10
\$SI	Format: value[numperline]	Site number of clocking
\$UN	Format: value[numperline]	Unit number of clocking
\$RE	Format: value[numperline]	Reason for absence
\$RT	Format: rate[numperline]	Hour Rates
\$TX	Format: String[numperline]	Free text description

The following symbols are only available for the first clocking on the line

\$late	Format: Time	Amount of lateness (minutes)
\$score	Format: Value	Coretime infringement
\$shift	Format: Value	Shift number
\$supr_name	Format: String	Name of supervisor making adjustment
\$supr_number	Format: String	Number of supervisor making adjustment
\$reason	Format: String	Reason for absence

For data collection clockings, the clock parameter symbols hold the information about the job. Parameter 1 contains either the index number for field 1 (job number), if set to alphanumeric, or the value entered if numeric. Parameter 2 contains the information for field 2 (op number) etc.

To display the job name of the first clocking of the line, use the directive "\$\$fetch_job \$PA1 0" to load the symbol \$j_name. For the second clocking, replace PA1 with PA2 etc. For the op numbers, replace PA1 with PB1 etc.

3.10 Absence Symbols

\$a_reason	Format: String	Absence Reason
\$a_mins	Format: Value	Number of minutes allocated
\$a_rate	Format: Value	Rate paid
\$a_shift	Format: Value	Shift number allocated
\$a_type	Format: Value	Absence type
\$a_colour	Format: Value[]	Absence (calendar) colours
\$a_flags	Format: Value	2=Notdue, 4=DeductExpected, 8=PayExpected
\$a_dept	Format: Value	Department number

3.11 Terminal Symbols

\$t_status	Format: Value	Status of terminal
\$t_name	Format: String	Terminal name

If terminal number is less than 128 the following are also set

\$t_zone	Format: Value	Zone number that the terminal belong to
\$t_zname	Format: String	Zone name
\$t_level	Format: Value	Terminal minimum access level
\$t_datalog	Format: Value	Data logging setting

The following event information is loaded for the selected event 1-16

\$t_event	Format: Value	Event code
\$t_nevent	Format: String	Event name
\$t_eventn	Format: Value	Event subcode
\$t_ev_dest	Format: Value	Destination number of event action
\$t_ev_ndest	Format: String	Destination name
\$t_action	Format: Value	Action code
\$t_naction	Format: String	Action name
\$t_actnn	Format: Value	Action subcode
\$t_parm	Format: Value Array	Duration etc
\$t_tzone	Format: Value	Time band for event

If terminal number is greater than 128 (ie zone), the following are also set

\$t_time	Format: Value Array	List of Time bands for zone
\$t_state	Format: Value Array	List of states for each time band
\$t_level	Format: Value Array	List of access levels for each time band

3.19 Other symbols

rectype	Type: Value	
fetchrecord	Type: cmd	
debug	Type: cmd	
record_count	Type: Value	recnum
reason_type	Type: Value	absence.type

APPENDIX I

CONTROL CODES

Clocking records contain various control codes giving extra information about the type of record. These are used for internal data processing but may be accessed within the report generator. Note however that these codes may be changed from time to time without warning.

Record Types \$RCn

RC_CLOCK_IN	1
RC_CLOCK_OUT	2
RC_BREAK	3
RC_EOW	4
RC_ADJUST	5
RC_END	15
RC_JOB_IN	0x21
RC_JOB_OUT	0x22
RC_SYSTEM	0x31
RC_EVENT	0x40

Record Modes \$MOn

The value of the Mode variable depends on the type of record:

Clocking in/out	RC_CLOCK_IN, RC_CLOCK_OUT
RCS_OK	0
RCS_NEW	1
RCS_AUTO	2
RCS_BAD	3
RCS_BUSN	4
End of week modes	RC_EOW
RCS_WEEK	8
RCS_PAYPERIOD	9
RCS_PPAYPERIOD	10
RCS_REVISED	11
System modes	RC_SYSTEM
RCS_START	21
RCS_ERRORS	22

Parameters \$PA_n - \$PJ_n

Parameters within the record depend on the type of clocking:

Clock in/out RC_CLOCK_IN or RC_CLOCK_OUT
 \$PB_n shift
 \$PC_n department
 \$PD_n infringements
 \$PE_n shift day

End of Week RC_EOW
 \$PB_n 0=end of week, 1=end of payperiod
 \$PC_n brought forward
 \$PD_n carry forward (limited to max_debit/credit)
 \$PE_n carry forward

System start RC_SYSTEM RCS_START
 \$PA_n maxempl
 \$PB_n dc
 \$PC_n maxcompanies
 \$PD_n num sites
 \$PE_n version

Infringements (\$PD_n for clock in/out)

Infringement flags are available for in/out clockings:

INF_LATE 0x001
 INF_CORE 0x002
 INF_NULL 0x004
 INF_BAD 0x008
 INF_AUTO 0x010
 INF_BUSN 0x020
 INF_EDIT 0x100
 INF_IN 0x200
 INF_OUT 0x400

Events:

Event Types \$RC_n
 EV_NONE RC_EVENT+0
 EV_SWITCH RC_EVENT+1
 EV_RELAY RC_EVENT+2
 EV_SYSTEM RC_EVENT+3
 EV_ACCESS RC_EVENT+4
 EV_TIMES RC_EVENT+5
 EV_ALARM RC_EVENT+6
 EV_DATALOG RC_EVENT+9

Event Modes \$MOn

The mode value depends on the type of event record:

Switch	RC_EVENT+EV_SWITCH
EVS_SW1	1
EVS_SW2	2
EVS_SW3	3
EVS_SW4	4
EVS_SW5	5
EVS_SW6	6
EVS_SW7	7
EVS_SW8	8

Relay	RC_EVENT+EV_RELAY
EVS_RLO	0
EVS_RL1	1
EVS_RL2	2
EVS_RL3	3
EVS_RL4	4
EVS_RL5	5
EVS_RL6	6
EVS_RL7	7

System	RC_EVENT+EV_SYSTEM
EVS_SY_FIRE	1
EVS_SY_FIRE_END	2
EVS_SY_PWR_OFF	3
EVS_SY_PWR_ON	4
EVS_SY_COMS_OFF	5
EVS_SY_COMS_ON	6

Access	RC_EVENT+EV_ACCESS
EVS_AC_ACCESS 1	
EVS_AC_FAIL	2
EVS_AC_DURESS 3	
EVS_AC_DOOR_ER	4
EVS_AC_EXIT	5
EVS_AC_EXIT_SW	6
EVS_AC_DOOR_SW	7
EVS_AC_TAMPER	8

Alarm	RC_EVENT+EV_ALARM
EVS_AL_OFF	1
EVS_AL_RESET 2	
EVS_AL_SET	3
EVS_AL_ACTIVE 4	
EVS_AL_TRIGGER	5
EVS_AL_ALARM 6	
EVS_AL_FAIL	7
EVS_AL_DURESS 8	
EVS_AL_TEST	9
EVS_AL_MANUAL	10
EVS_AL_TAMPER 11	

Winrep Command Line Parameters

-r <filename>	Run the report <filename>, print it and exit
-rnp	Do not print - display on screen
-re <employee>	Run report for employee number <employee> only
-rd <department(s)>	Run report for specified departments only (comma separated list)
-rj <job>	Run report for job number <job> only
-rp <period>	Run report over period specified (as ask_period)
-rst <secs>	Report start time (if period is date range)
-ret <secs>	Report end time (if period is date range)
-ru <num>	Pass num as report parameter
-e <username/password>	Login as username and run the report generator
-exec <program>	Execute program when report is finished.

Notes:

For start and end times, the time may be given as relative by prefixing with + or -. It may also be given in different units by adding the character D for day, W for week, M for month or Y for year. (where +0W is the start of the current week)

EG to run the holiday report for the next two weeks use:

```
winrep -r personel/calmonth -rnp -rst +0W -ret +2W
```

E.G. to produce a pay report for employee E1001 for the current week, use:

```
winrep -r ta\pay.rpg -re E1001 -rp 3
```

E.G. To produce a pay report for departments 3 and 4 and email to Fred.

```
winrep -r ta\pay -rnp -rd "3,4" -exec "blat -body report -subject Report -to fred@hotmail.com -attach"
```

<u>Absence Symbols.</u>	<u>27</u>
<u>array .</u>	<u>18</u>
<u>ask_absence.</u>	<u>9</u>
<u>ask_destination.</u>	<u>8</u>
<u>ask_empl.</u>	<u>8</u>
<u>ask_jobs.</u>	<u>8</u>
<u>ask_pay.</u>	<u>9</u>
<u>ask_period.</u>	<u>9</u>
<u>ask_sort.</u>	<u>9</u>
<u>bar.</u>	<u>12</u>
<u>begdate.</u>	<u>12</u>
<u>bold_off.</u>	<u>12</u>
<u>bold_on.</u>	<u>12</u>
<u>cal_format.</u>	<u>11</u>
<u>Calculations.</u>	<u>3</u>
<u>cflags.</u>	<u>12</u>
<u>clock_format.</u>	<u>11</u>
<u>Command Line Parameters.</u>	<u>31</u>
<u>CONTROL CODES.</u>	<u>28</u>
<u>Control Symbols.</u>	<u>8</u>
<u>Creating Symbols.</u>	<u>3</u>
<u>curdate.</u>	<u>12</u>
<u>curtime.</u>	<u>12</u>
<u>Data Collection Symbols.</u>	<u>25</u>
<u>date_format.</u>	<u>11</u>
<u>dc.</u>	<u>10</u>
<u>delimiters.</u>	<u>10</u>
<u>Department Symbols.</u>	<u>24</u>
<u>Directives.</u>	<u>13</u>
<u>directory.</u>	<u>8</u>
<u>display.</u>	<u>21</u>
<u>else.</u>	<u>13</u>
<u>elseif.</u>	<u>13</u>
<u>Employee Symbols.</u>	<u>23</u>
<u>en_deleted.</u>	<u>9</u>
<u>enable.</u>	<u>10</u>
<u>end.</u>	<u>13</u>
<u>end_screen.</u>	<u>12</u>
<u>enddate.</u>	<u>12</u>
<u>endif.</u>	<u>13</u>
<u>endwhile.</u>	<u>14</u>
<u>error_format.</u>	<u>11</u>
<u>execute.</u>	<u>22</u>
<u>fetchabsence.</u>	<u>15</u>
<u>fetchdept.</u>	<u>15</u>
<u>fetchempl.</u>	<u>14</u>
<u>fetchhr.</u>	<u>16</u>
<u>fetchjob.</u>	<u>15</u>
<u>fetchrecord.</u>	<u>15</u>
<u>fetchshift.</u>	<u>14</u>
<u>fetchsite.</u>	<u>16</u>
<u>fetchterminal.</u>	<u>15</u>
<u>File Structure.</u>	<u>2</u>
<u>filename.</u>	<u>16</u>
<u>findinfo.</u>	<u>17</u>
<u>Format Definitions.</u>	<u>4</u>
<u>Format Symbols.</u>	<u>11</u>
<u>Formatting.</u>	<u>2</u>
<u>formfeed.</u>	<u>12</u>
<u>getchar.</u>	<u>21</u>
<u>graph.</u>	<u>20</u>
<u>gtot_format.</u>	<u>11</u>
<u>hol_hours.</u>	<u>12</u>

<u>if.</u>	<u>13</u>
<u>Immediate Mode Screen Control.</u>	<u>21</u>
<u>import.</u>	<u>18</u>
<u>input.</u>	<u>17</u>
<u>Introduction.</u>	<u>1</u>
<u>italic_off.</u>	<u>12</u>
<u>italic_on.</u>	<u>12</u>
<u>launch.</u>	<u>22</u>
<u>lineno.</u>	<u>12</u>
<u>main.</u>	<u>13</u>
<u>numberline.</u>	<u>8</u>
<u>numsites.</u>	<u>12</u>
<u>Overview.</u>	<u>1</u>
<u>pageno.</u>	<u>12</u>
<u>Parameters.</u>	<u>29</u>
<u>parma.</u>	<u>12</u>
<u>parmb.</u>	<u>12</u>
<u>pause.</u>	<u>21</u>
<u>pr_condensed.</u>	<u>12</u>
<u>pr_expanded.</u>	<u>12</u>
<u>pr_normal.</u>	<u>12</u>
<u>PREDEFINED SYMBOLS.</u>	<u>8</u>
<u>printer.</u>	<u>17</u>
<u>Printer Control Symbols.</u>	<u>12</u>
<u>rate_format.</u>	<u>11</u>
<u>readparm.</u>	<u>16</u>
<u>Record Symbols.</u>	<u>26</u>
<u>records.</u>	<u>10</u>
<u>recsort.</u>	<u>15</u>
<u>screen.</u>	<u>10</u>
<u>search.</u>	<u>18</u>
<u>select.</u>	<u>17, 22</u>
<u>Shift Symbols.</u>	<u>24</u>
<u>site.</u>	<u>12</u>
<u>sort.</u>	<u>18</u>
<u>start.</u>	<u>13</u>
<u>storeempl.</u>	<u>14</u>
<u>String specifiers.</u>	<u>4</u>
<u>string_format.</u>	<u>11</u>
<u>tabset.</u>	<u>19</u>
<u>Terminal Symbols.</u>	<u>27</u>
<u>text.</u>	<u>19</u>
<u>Time/Date specifiers.</u>	<u>6</u>
<u>time_format.</u>	<u>11</u>
<u>totl_format.</u>	<u>11</u>
<u>ul_off.</u>	<u>12</u>
<u>ul_on.</u>	<u>12</u>
<u>Value Specifiers.</u>	<u>5</u>
<u>value_format.</u>	<u>11</u>
<u>version.</u>	<u>12</u>
<u>version_date.</u>	<u>12</u>
<u>weekno.</u>	<u>12</u>
<u>while.</u>	<u>14</u>
<u>whilerecords.</u>	<u>14</u>
<u>yeardate.</u>	<u>12</u>
<u>zero_rates.</u>	<u>12</u>