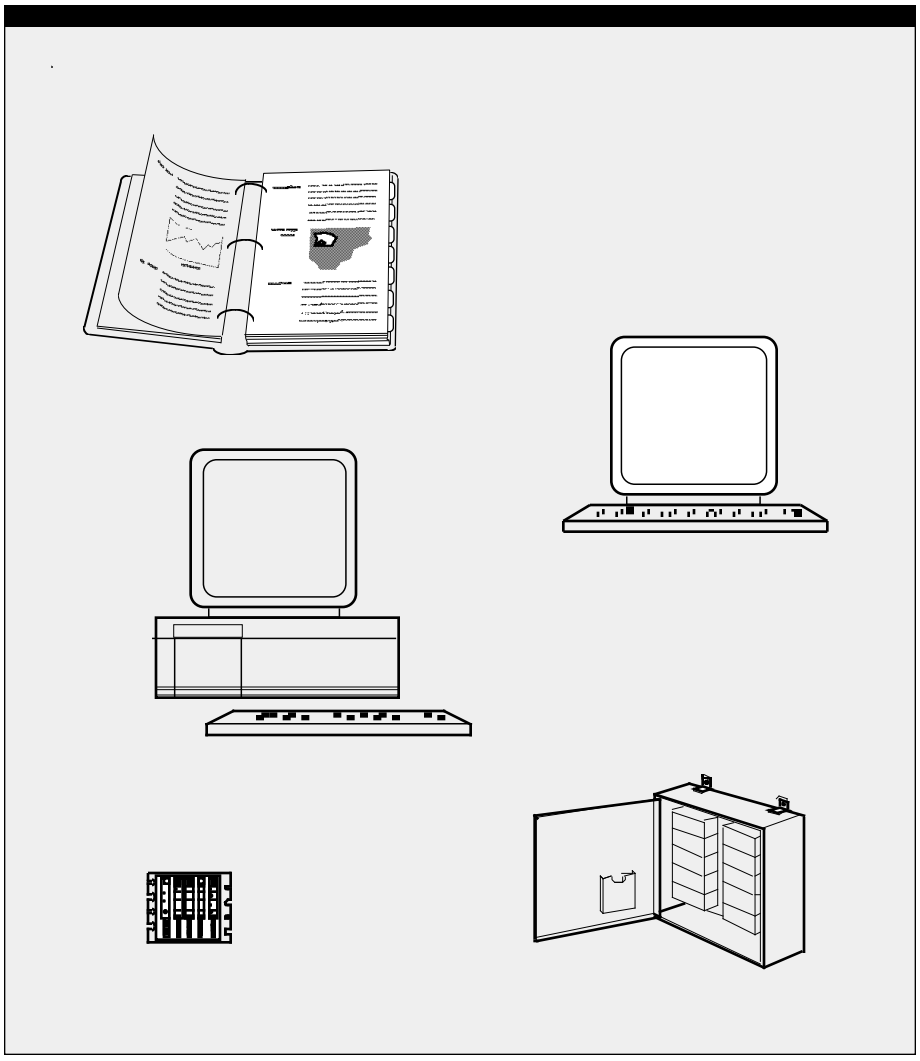
STREAMLINE ™

Software

Reference Documents

SQUARE D COMPANY
GROUPE SCHNEIDER



STREAMLINE ™

SOFTWARE

Reference

Documents

Introduction

General

This document details the CRISP®/32 tools available to the software application engineer to create data summary file from their CRISP/32 database.

This manual is divided into the following sections.

Section	Description
Operation *(page 3)*	Defines the operational details of the REPORTER™ system.
Configuration *(page 5)*	Defines the system license and configuration of the REPORTER system.
Commands *(page 7)*	Defines the EDL, RPC, and RPR commands.
Report Types *(page 9)*	Defines the different reports.
CRISP Variable Names *(page 11)*	Defines the access CRISP variables of REPORTER.
Distribution Files *(page 13)*	Defines the distribution files of REPORTER.
Directive Reference *(page 15)*	Defines the directive and the usage of REPORTER.
Directives *(page 19)*	Defines the directives of REPORTER.

Notes:

Overview

The REPORTER™ is a software package that provides users of CRISP/32 process control systems the ability to create data summary files from their CRISP/32 databases. These data files can contain values or averages of any data point in the CRISP® database and are formatted for direct printing or for importation into other database manipulation packages.

Some features of the package include the following.

- User selectable report headings and footers.
- Up to 3500 variables per report file.
- Up to 100 report files per CPU depending on performance requirements.
- Power fail error recovery.
- Event or time driven sampling.
- Event or time controlled report generation.
- User specified sample and average timing.
- User specified point names for each file.
- Tabular ASCII file format suitable for importation into other database and spreadsheet packages such as SAS, 1-2-3, DATATRIEVE, DBASE, 20/20, and others.
- Data files can be easily moved between DECnet nodes or to IBM/PC databases via simple asynchronous file transfer protocols.
- One time Ad-Hoc reports.
- Time of day sampling and closing of report files.
- Run different Reports with one REPORTER process.
- Functions on print directives.
- Multiple pages.
- Basic Math functions performed on `.PRINT` and `.STATS` directives
- ASCII file data can be transferred to the CRISP data base through the REPORTER.

Overview (cont)

- Output to System Print Queues.
- Write Report Data back to CRISP Real time database.
- Conditional printing of print directives.
- Recover a report between CRISP stop and start.
- Log data to a file if a CRISP data base variable's value increases, decreases or changes.
- DATE, TIME, YEAR, MONTH, DAY, HOUR, MINUTE, DOW (Day of Week) keywords.

Product Design

The REPORTER is simple to install and easy to use. An effort has been made to accommodate the novice CRISP user. Minimal DEC operating system experience is required, as the installation and configuration are performed with simple programs and command files. The REPORTER consists of two executables, a configuration command file and a support utility. The two executables are a REPORTER configuration file compiler (RPC) and a run time program (RPR) that takes the compiled configuration file and executes it. The user configuration command file is used to configure the reports for automatic start up when CRISP starts. The utility program (EDL) aids in linking the REPORTER variables to the CRISP databases. EDL is required for pre version 2.0 REPORTER. A sample CRISP C32 file, CHART II display file and REPORTER files have been included to aid the user by providing working examples.

System License

The REPORTER requires that a valid license file be present in the directory in which the REPORTER was installed (i.e. [CRISP.RPT]). The license is verified when a configuration file is compiled and also when the REPORTER is started. The following is contained in a license file. Note that you will receive the valid codes when the REPORTER is purchased. The information as it appears on the license document must be entered letter for letter and space for space. Upper and lower case letters must be used in the same way as they are on the hard copy license page. The license file is called LICENSE.RPT.

```
CUSTOMERNAME>>          ;"CUST NAME"
CUSTOMERLOCATION>>      ;"CUST LOCATION"
CUSTOMERSWLICENSE>>     ;100000000
CUSTOMERUSELIMIT>>      ;1
CUSTOMERCONFIGCODE>>    ;1234567890
CUSTOMERCONFIGID>>      ;1234567890
```

System Configuration

The report system is configured by executing the USER_CONFIG_RPT command file. USER_CONFIG_RPT will ask the user for the information needed to configure the report system. The command file should be run from the CRISP root directory (i.e. CRISP$CFG:).

Four basic questions are asked :

```
1.How many copies of Reporter should be automatically started [1] ?
```

This question determines how many reports are to be executed when CRISP is started.

```
        Information for Reporter Process 1
2.Report filespec  [CRISP$RPT:TIME.RPT]? [CRISP.RPT]DAILY.RPT
```

Enter the directory specification and report name at this prompt for each report to be started.

NOTE

Report file names should be kept to 8 characters or less. This name is used as the connect name on the Software bus. Connect names are eight or less characters.

System Configuration (cont)

3. Compile Report at Startup [YES]? **RETURN**

This will automatically run RPC before installing the report.

Question 2 and 3 will repeat for the number of automatic report file starts.

4. Do you want Reporter installed as shareable [YES]?

This will allow all common routines to use same the memory area. This will conserve memory.

There are several files created by the USER_CONFIG_RPT command file. Their location and purpose are described below.

USER_SETUP_RPT.COM This file sets up system logicals for the REPORTER and is located in the CRISP$CFG: directory of the device that CRISP is installed on. This command file is executed at system boot time.

USER_LOGIN_RPT.COM This file sets up global symbols for the REPORTER and is located in the [CRISP] directory of the device that CRISP is installed on. This command file is executed when the user logs in to the CRISP directory.

USER_START_RPT.COM This file starts REPORTER on a CRISP/32 system and is located in the [CRISP.EXE] directory of the device that CRISP is installed on. This command file is executed when the user starts CRISP.

Report Linker (EDL)

EDL is optional with RPT V2.0. This utility inserts the symbol information (i.e. variable type, variable offset) into the .SYMBOL section of the report configuration file (see the .SYMBOL directive for details on how to format the variables in the .SYMBOL section). This is accomplished by using the software bus locate and resolve call to look up the symbols in the .SYMBOL section of the report configuration file. Note that CRISP/32 must be running to accomplish this. After EDL process the report configuration file, it will place the time and date after the .SYMBOL directive.

To use EDL to process individual report configuration files use the following command.

```
EDL TIME.RPT
```

`TIME.RPT` is the name of the report configuration file.

NOTE

If EDL fails to locate a variable in the database, this problem should be resolved before RPC is executed.

Report Compiler (RPC)

RPC is the report compiler that reads the report configuration file and generates several files. A .RDT file is always created on a successful compile and contains the information required for any report to operate, such as sample rate, sampled variables, etc. A .HDR or .FTR file will only be created if a .HEADER or .FOOTER directive is encountered in the report configuration file.

If you desire to be compatible with EDL, it is suggested that .DSP be used as an extension for the report configuration file. To run RPC type the following.

```
RPC  TIME.RPT [/LIST]
```

`TIME.RPT` is the name of the report configuration file. The `/LIST` switch is optional and indicates that a listing is to be generated. If a listing is generated, it is sent to the terminal. A listing may be generated to aid in debugging an error in the configuration file. The default file extension for a report file is RPT.

Run Time Program (RPR)

RPR is the REPORTER run time program, it uses the .RDT file to configure itself to operate in the way that the report configuration file describes. The run time program is installed with unique task names to allow multiple reports to run concurrently. The run time program connects to the CRISP Software Bus using the report file name. The Software Bus connection provides the communication path to the CRISP database.

Power fail error recovery has been incorporated into the system. When the power fails on the system, the next time a report is started it will continue were it left off. Power fail recovery is made possible by using a recovery file (.RCV) that is updated with the statistics, number of writes, and file name data during every write to the report. All information needed to fully recover the report is kept in the (.RCV) file. To run RPR type the following.

```
RPR TIME.RPT
```

`TIME.RPT` is the name of the report configuration file.

NOTE

You do not necessarily have to manually run RPR since USER_START_RPT does it for you. Manual operation is useful for quick one time reports that you may not want to include in the automatic start up.

RPR Switches Optional

These optional switches are provided for user testing.

```
RPR MYRPT /NOCRISP /NOSWT /TSTTIM
```

`/NOCRISP` This switch will disable RPR from checking if CRISP is running.

`/NOSWT` This switch will cause the report to run and collect data Regardless of the logical state of the SAMPLE_WHILE_TRUE variable.

`/TSTTIM` This will cause time to accelerate. The time will increment by the seconds per sample every pass. With a value of 20 for SECONDS_PER_SAMPLE, time would advance by 20 seconds every pass of the report.

NOTE

Switches must be separated by spaces

Event Based Report

An event based report is used when you wish to have reports written based on some event in the CRISP database such as a high level alarm or a batch complete. These events are not predictable so the report generation is triggered by setting a Logical in the CRISP database that causes the report to be printed. The .WRITE_NOW directive is used to specify the Logical in the CRISP database. If the .WRITE_PER_FILE value is greater than 1, then a new record is printed to the same report until the number of records is equal to the .WRITE_PER_FILE value.

The .WRITE_AT and .CLOSE_AT directives are also available for event based reporting where the event is some time of the day, week, or month. These directives allow you to specify a set time at which a record should be written to the report or some set time at which the report should be closed.

Refer to the file EVENT.RPT for an example of an event based report using the .WRITE_NOW directive. See the file DAILY.RPT for an example of an event based report that uses the .WRITE_AT directive.

Time Based Report

Time based reports are usually used to build summary files of columns of data on a periodic basis. An example might be the flow rate, temperature, and pressure in a process. The .WRITE_PERIOD directive is used to specify the rate at which records should be written to the report. The .WRITE_PER_FILE directive will specify how many records should be written to the report. If .WRITE_PER_FILE is not specified, it defaults to 1 and a new report will be generated for every .WRITE_PERIOD.

Refer to the file TIME.RPT for an example of a time based report using the .WRITE_PERIOD directive.

One Time Report

The REPORTER is capable of running one time reports in an interactive fashion. This is useful for Ad Hoc reports that a user can create to retrieve summary information from the CRISP Real Time database. The report configuration file should contain a .FILE_NAME, .PRINT, and a .FORMAT directives at a minimum.

When running this type of report the user simply compiles the report and runs it interactively from the DCL command line. Alternate methods for running one time reports are by using a command file that contains the commands to launch the report or by submitting this command file to the system batch queue.

Refer to the file ONE.RPT for an example of a one time report.

One Process Multiple Reports

In some reporting applications a need arises to print many different types of reports. The multiple report type is similar to the one time report except that the .RPT_UCF_FILE directive is used to specify a CRISP string variable that contains the name of the report to run and the .RUN_RPT_IF directive is used to specify a bit in the CRISP database that when set will cause the report to be run. The advantage of this mode of operation is that there is only one REPORTER process running on the VAX.

Refer to the example sections MULTI.RPT, MULTI1.RPT, and MULTI2.RPT for examples of this type of report.

Event Log

By using the LOG_IF directive an event log report can be generated. This directive can be used to detect a change in the value any CRISP database variable. The change can be a positive or a negative or unidirectional. In the case of STRING type variables the numerical value of each ASCII character is added and the total is stored. This stored value is compared to the current value for change. This directive can be used in EVENT or TIME based reports.

Refer to the example section LOG.RPT for an example of a event log reporting.

ASCII Input

The REPORTER has the ability to read ASCII data files and write information contained in those files to the CRISP Real Time database. Columns of data can be read into arrays of CRISP variables. Support is provided for indexing into the ASCII file on a record by record basis. The user can configure the ASCII read mode to parse the file by specifying white space, field delimiters, and record delimiters. A list of CRISP variables is specified as the destination for the ASCII data. Refer to the sample REPORTER configuration file input.rpt for an example of this mode of operation. The INPUT.RPT configuration file uses the ASCII text file input.txt to read data from. Three fields are read from input.txt, a float, a long, and a string. The field delimiter is specified as a tab so the columns in input.txt are separated by tabs. The record delimiter is specified as a new line so new line characters are used to separate the records in input.txt. A space is specified as white space although there are actually no spaces in the file input.txt. The file is then closed and can optionally be deleted. Refer to the REPORTER example sections INPUT.RPT and INPUT2.RPT for examples of this report type.

CRISP Variable Names

The REPORTER can access CRISP variables by name and database. Many of the directives require a CRISP variable name. The variables are used to control the operation of the REPORTER and also provide the data that gets filled into the report as it is generated. This method eliminates the need for the old .SYMBOL statement.

A CRISP variable name is specified in the REPORTER configuration file as follows.

```
#DATABASE:VAR_NAME
```

where :

Required part of the syntax

`DATABASE` is the CRISP database that contains the variable. This is not required when the `.APP_DB` directive is used.

`VAR_NAME` is the variable or tag in that database.

Notes:

Distribution Files

The CRISP/32 REPORTER software consists of the following files.

LICENSE.RPT	The REPORTER license.
EDL.EXE	Linker utility.
RPTC.EXE	Report compiler.
RPTCRT.EXE	Report Run time program.
USER_CONFIG_RPT.COM	Report configuration program.
AVG.RPT	Example of writing to CRISP database the AVERAGE value calculated by the REPORTER.
DAILY.RPT	Example DAILY report file using the .WRITE_AT and CLOSE_AT directives.
DAY.RPT	Example Daily Report.
EVENT.RPT	Example event based config file.
FORMAT.RPT	Example of single .FORMAT with multiple .PRINTS.
INPUT.RPT	Example ASCII Input File.
INPUT2.RPT	Example ASCII Input File.
LOG.RPT	Example EVENT logging report.
MATH.RPT	Example MATH functions.
MATH2.RPT	Example MATH functions.
MULTI.RPT	Example multiple report from one process.
MULTI1.RPT	Example report file run from MULTI.RPT.
MULTI2.RPT	Example report file run from MULTI.RPT.
ONE.RPT	Example one time report.
PAGE.RPT	Example of a multiple page report.
RPT32.C32	Example CRISP database for Report files.
RPT32.DSP	Example CHART/II display for Report files.
SYMBOL.DSP	Example of pre V2.0 .SYMBOL section.
SPS.RPT	Example SECONDS PER SAMPLE.
TIME.RPT	Example time based config file.

The REPORTER should be installed using the standard VMS installation procedure. Consult the Software Installation Guide for information regarding this product. The REPORTER files will be placed into the directory [CRISP.RPT] on the device specified at installation.

Notes:

Directive Reference

Directive	Usage
.APP_DB	Allows a default CRISP data base to be specified.
.BAD_DATA_VALUE	If a sample is equal to this value, ignore it.
.CHART	Used in CHART/II and REPORTER combination file. Allows the user to have one file where common CHART/II and REPORTER variables are used by both products.
.CLOSE_AT	Specify times to close the report file.
.CLOSE_INPUT_FILE_IF	If Logical is true, the ASCII input file is closed
.DATA_BASE	Specify the CRISP databases that will be used.
.END	End of file.
.EXIT_IF	If Logical is true, exit REPORTER process.
.FIELD_DELIMITER	Specify a list of field delimiter characters for ASCII input data.
.FILE_NAME	Specify the file name of the report.
.FILE_OPEN_STATUS	If Logical is true the report is open.
.FIRST_RECORD	Specifies the number of records to skip in the ASCII input file before data is read.
.FOOTER	Specify information to place in the footer section of the report.
.FORMAT	Specify the format of the output data in the .PRINT directive.
.FORMAT_MODE	Define the type of format to be used for the report.
.HEADER	Specify information to place in the header section of the report.
.IGNORE	Causes the REPORTER compiler RPC to skip over the CHART/II directives. Used with the .RESUME directive.

Directive Reference (cont)

.INPUT_ADB_VAR_LIST	Specifies a list of CRISP variables to read the ASCII data into.
.INPUT_FILENAME	Specifies the ASCII file to read.
.INPUT_FILE_OPEN_STATUS	Specifies a CRISP Logical that will be true when the ASCII input file is open .
.LOG_IF	Writes data when specified change has occurred
.OPEN_INPUT_FILE_IF	Specifies a CRISP Logical that when true will cause the ASCII input file to be opened.
.PAGE	Specify the page for output to the report.
.PRINT	Specify the information that will be printed in the report.
.PRINT_IF	If this Logical is false, the corresponding .PRINT directive is not printed.
.QUEUE	Specify the system print queue to print the report on.
.READ_IF set	Specifies a CRISP Logical that when will cause data to be read from the ASCII input file.
.READ_REC_COUNT	Specifies the number of records to read from the ASCII input file.
.RECORD_DELIMITER	Specify a list of record delimiter characters for ASCII input file.
.RECOVER_CRISP_STOP	Specify that the report file should be recovered through CRISP stop and start
.RESUME	Allows the REPORTER compiler RPC to skip over the CHART/II directives. Used with the .IGNORE directive.
.RPT_UCF_FILE	Use a CRISP string to specify the report configuration file name.

Directive Reference (cont)

.RUN_ONE_TIME	Force the report to be a one time report.
.RUN_RPT_IF	If this Logical is true, execute the report name contained in the .RPT_UCF_FILE string.
.SAMPLE_WHILE_TRUE	Set this Logical true to enable sampling.
.SECONDS_PER_SAMPLE	Specify the frequency to sample data.
.STATS_FORMAT	Define the format of the statistics specified with /STATS.
.STATS_HEADER	Specify a heading for the statistics.
.STATS_LABEL	Specify different labels for STATS lines.
.STR_WHITE_SPACE	Specify characters to be ignored when reading ASCII input files into CRISP strings.
.SYMBOL	Start of variable index section.
.VARS_PER_RECORD	Specifies the number of fields expected between records in the ASCII file.
.WHITE_SPACE	Specify a list of white space characters. White space is ignored when reading the ASCII file.
.WRITE_AT	Specify a time to write a record to the report.
.WRITE_NOW	If this Logical is true, write a record to the report.
.WRITE_PER_FILE	Specify the number of records to write to the report.
.WRITE_PERIOD	Specify the frequency to write records to the report.

Notes:

APP_DB

This directive allows a default data base to be specified. This eliminates the need to specify the data base name for each variable accessed. If the variable is in a database other then the default, that database must be specified with the variable name.

```
.APP_DB = "RPT32"
```

Bad_Data_Value

If a sample is equal to this value, it will be discarded from any processing that may be performed on it. If `/STATS` are specified for the printed variable, a bad count will be included at the end of the report.

```
.BAD_DATA_VALUE = number
```

Chart

This is used when the REPORTER file is included in as part of a CHART/II display. The `.IGNORE` and `.RESUME` must be used to prevent the RPC compiler from reading CHART/II directives.

Close_At

Specifies when the file is to be closed. Currently a maximum of 50 different close times can be specified per report. When the close time is reached, the file is closed regardless of the number of writes per file. Valid keywords are as follows.

```
END_OF_HOUR
END_OF_DAY
END_OF_MONTH
END_OF_YEAR
```

Or a specific time can be specified. The following is the basic syntax of the entry on the `.CLOSE_AT` directive.

```
[DAILY,MON,TUE,WED,THU,FRI,SAT,SUN]   @   HH:MM:SS
```

or

```
[MONTHLY,JAN,FEB, ... DEC] ## @ HH:MM:SS
```

where : ## is a day of the month
`HH` is the hour
`MM` is the minutes
`SS` is the seconds

NOTE

At least one item listed inside of the [ ] must be specified.

The entries on the .CLOSE_AT directive must be separated by commas.

Example :

```
.CLOSE_AT = DAILY @08:00:00,MON @09:00:00,JAN 1 @10:00:00
```

Close_Input_File_If

The CRISP Logical database variable specified will determine when the ASCII input file is closed. When this variable transitions from false to true (The result of some CRISP logic statement and input from a display) the ASCII input file is closed. Once the file is closed this variable is set false by the REPORTER to indicate the file has been closed.

```
.CLOSE_INPUT_FILE_IF = #RPT32:CLOSE_ASCII_FILE
```

Comment

Comments can be placed in the Report file by using a exclamation mark.

```
! THIS IS A COMMENT
```

Data_Base

(Required for pre version 2.0) This command identifies the CRISP/32 databases to be used. The first database listed is number zero, the second is database one. The database number is used along with the actual point name to uniquely identify the variable. Refer to the .SYMBOL command below. This command is only required if EDL is used to link the report.

```
.DATA_BASE= AB, CC, RI, SP
```

End

Mark the end of the report file. This directive or the old .SYMBOL directive is required.

```
.END
```

Exit_If

This CRISP Logical variable, when true, will cause the report process to exit.

```
.EXIT_IF = #DATABASE:STOP_REPORT
```

Field_Delimiter

This directive to specify a list of field delimiter characters. A CRISP string variable may be used to specify a field delimiter character. Field delimiters are used to separate the data in the ASCII file so that data can be read in pieces that match the .INPUT_ADB_VAR_LIST.

```
.FIELD_DELIMITER  = "\t"
```

This will use the tab as a field delimiter.

File_Name

The FILE_NAME directive controls the naming of the files created by the REPORTER. The report configuration file may contain more than one FILE_NAME directive. If more than one .FILE_NAME directive is used, the names specified will be used sequentially. For example, the first output file will be named according to the first .FILE_NAME directive. The second output file will be named according to the second .FILE_NAME directive and so on until the list is exhausted at that time the 1st one is used over again. The .FILE_NAME directive can consist of a CRISP variable and text enclosed in quotes or any valid file name keyword. Each element must be separated by commas. A grand total of 500 elements are allowed per Report file. This total is the sum of all .FILE_NAME directives.

```
.FILE_NAME = CRISP Variable
.FILE_NAME = "DISK$USER:[CRISP.LOG]MYREPORT.DAT"
.FILE_NAME = "DISK$USER:[Y", YEAR, ".", MONTH,"]",MONTH, DAY, ".DRP"
```

Keyword	Description
DAY	This will use the current two digit day number. If the day is 9 or less, a leading zero is inserted.
MONTH	This will use the 3 letter month abbreviation. ie; JAN FEB MAR
YEAR	This will use the current YEAR. ie; 1991 1992 1993
SEQ	This will cause the suffix to increment from 000000 to 999999 as each file is generated.
DT	This will cause the current date and time to be used to generate a 6 character string of the following format.

HHMMmDY

Component	Range	Represent
HH	0 to N	0 to 23 hours
MM	00 to 59	0 to 59 minutes
m	1 to C	Jan to Dec
D	1 to V	1st to 31st day
Y	0 to 9	0 to 9th year

File_Name (cont)

DATE — This will cause the current date and time to be used as the suffix of the report with the following format.

```
YYYYMMDDHHMMSS
```

The time used is the time the report is started.

TIME — This will cause the current time to be used as the suffix of the report with the following format.

HHMMSS — The time used is the time at which the report is started.

Examples :

```
1).FILE_NAME = "DISK$USER:[CRISP.REPORTS]","BATCH_",SEQ,".DAT"
```

The files name will be the following.

```
1st time : DISK$USER:[CRISP.REPORTS]BATCH_000000.DAT

2nd time : DISK$USER:[CRISP.REPORTS]BATCH_000001.DAT
```

```
2).FILE_NAME = "UPDATE_" , TIME , ".SAV"
```

The file will be created in the `[CRISP.RPT]` directory (since a directory was not specified) and its name will be the following.

```
UPDATE_123013.SAV
```

Which would tell us that the report was started at `12:30:13.`

First_Record

The `.FIRST_RECORD` directive specifies the number of records to skip in the ASCII file before data is actually read. A CRISP numeric variable or constant can also be used.

```
.first_record     = #rpt32:first_row
```

This specifies that the contents of `first_row` are to be used to specify the number of records to skip.

File_Open_Status

This is a Logical that is set true by the REPORTER. When true, this indicates that the report file is open. If forced off or false, the report is closed.

```
.FILE_OPEN_STATUS = #TEST:FOS  CRISP variable
```

Footer

When this parameter is specified, contents are placed as a line in the report. Control characters can be used to produce desired results.

```
\n, \t, \v, \b as listed in the FORMAT parameter.
```

The .FOOTER directive can contain text enclosed in quotes or CRISP data base variable. If you wish to place the data in a position other then the left side of the report, spaces should be used instead of tabs inside of quotes.

```
.FOOTER = "\n"
.FOOTER = #USER1:DATE_STR
.FOOTER = "                                    ",TIME
.FOOTER = "                                    ", DATE
.FOOTER = "                                 text"
```

Format

This directive defines the format for the associated .PRINT directives. The 1 specifies that this is format directive number 1 and any .PRINT directive that references format directive 1 will use that format to print its output. The text can consist of valid 'C' language format specifiers, text, and special format characters. The text must end with a format specifier. Or the LB_FMT format method can be used. Refer to .FORMAT_MODE. Below are some examples.

```
%#s   string
%#d   numeric
%#.#f  float
```

is a field width specifier. For a float format specifier the first # is the total width and the second # is the number of decimal points. The following special characters may be used in a text string to provide for report formatting.

```
\n    Newline
\t    Horizontal Tab
\v    Vertical tab
\b    Back space
\r    Carriage Return
\f    Form feed
\0    NULL
\"    Quote
```

Refer to the documentation on the fprintf() function in a 'C' language reference manual for more information on valid formatting options. Below are some examples of .FORMAT and .PRINT directives. Each .FORMAT can be used many times.

```
.PRINT, 1 = TIME , 10 ,20,"YES"
.FORMAT, 1 = "%8s %7.2f %7.3f   \"%3s\""
.PRINT , 2= "the time :",TIME , 30 ," A variable :",40
.FORMAT, 2 = "%s %8s %3.2f  %s %5d"
```

Format (cont)

This sequence would print the following.

```
10:11:30 1234.67 123.456   "YES"
the date : 10:11:30 1.12   A variable : 12345
```

Format_Mode

This directive will allow the user to specify the optional LB_FMT. This will enable the user to specify formats using the following characters.

`S for string characters`	`#SSSSSSS` for a 8-character string.
`F for floats`	`#FFF.FF` for 4 digits to the left of the decimal point and 2 to the left.
`D for logical and numeric`	`#DDDD` for a 5-digit numerical.

Please note that the initial # character is counted as a character in the format.

```
.FORMAT_MODE = LB_FMT
```

NOTE

It is not advisable to use tabs since tab setting can vary from each device. Spaces will not vary and will provide a reliable output format.

Valid format modes are `LB_FMT` and `C_FMT`. `C_FMT` specifies that the default 'C' formatting mode is to be used.

Header

There are several methods available for annotating a report. The `.HEADER` and `.FOOTER` command allows a header and footer lines at the beginning and end of each report. A text string can be specified directly in this file or a CRISP string variable can be provided to allow for the specification of a header or footer from the CRISP application program and a `keyword` can be used.

Multiple headers and footers can be specified for each report.

NOTE

Headers are taken from the CRISP database when the file is created; while footers are taken just before the file is closed.

Header (cont)

Headers and footers are taken exactly as they are found. If they are to be centered, then center them. If they are a blank line, then provide a blank line.

```
.HEADER =    a CRISP variable  | "text"
```

Valid keywords for the header are as follows.

DATE
TIME
YEAR
MONTH
DAY

Ignore

This directive will cause the REPORTER RPC process to skip to the .RESUME directive. This directive is intended for use in a CHART/II display files that will use the same variables as the report file.

Input_Adb_Var_List

This directive specifies a list of CRISP variables to read into the ASCII data. Using the field and record delimiters the data in the ASCII file will be parsed and written to the CRISP tags. Data type conversion is done so that the data type of the CRISP variable is used to convert the ASCII data to the proper format. If the .READ_REC_COUNT is greater than 1, then the data is written to successive locations in the CRISP database so that columns of ASCII data can be written into arrays in the CRISP database.

```
.app_db     = "rpt32"
.input_adb_var_list  = #float1,#long1,#str1
```

This specifies the CRISP data base RPT32 and the tags float1, long1, and str1 as the location to write the ASCII data to.

Input_Filename

The .INPUT_FILENAME directive specifies the file to read. A CRISP string variable can also be used.

```
.INPUT_FILENAME   = "input.txt"
```

This will specify that the file INPUT.TXT should be used as the ASCII input file.

NOTE

The full path name may need to be specified.

Input_File_Open_Status

The .INPUT_FILE_OPEN_STATUS directive specifies a CRISP Logical that is true when the file specified in the .INPUT_FILENAME directive is open. The Logical specified will be reset when the file is closed.

```
.input_file_open_status = #rpt32:input_file_open
```

This specifies the Logical input_file_open in the rpt32 database as the open status bit.

Line Continuation

A line may be continued in the report file by using a \ (Back slash) character.

```
.PRINT = DATE \
#TEST:FLOW_TOTAL
```

Log_If

This directive allows a CRISP database variable to be monitor for the specified change. When this change is detected, the PRINT statement is written to the report.

```
.LOG_IF POSITIVE #TEST:MOTOR_101_START THEN DATE,"MOTOR 101 STARTED AT ", TIME
```

When MOTOR_101_START goes true, the text "MOTOR 101 STARTED AT" and the current time are written to the report.

Keywords are as follows.

POSITIVE	Monitors any CRISP variable for a Positive increase in value. Logical would be from FALSE to TRUE. A string would be if the NUMERICAL SUM of the characters has increased from previous value.
NEGATIVE	Monitors any CRISP variable for a Negative increase in value. Logical would be from TRUE to FALSE String would be if the NUMERICAL SUM of the characters has decreased from previous value.
CHANGE	Monitors any CRISP variable for a change Positive or Negative.

Open_Input_File_If

The .OPEN_INPUT_FILE_IF directive specifies a CRISP Logical that when true will cause the file specified in the .INPUT_FILENAME directive to open for reading. The Logical specified will be set when the file is closed.

```
.open_input_file_if  = #rpt32:open_input_file
```

This specifies the Logical open_input_file in the rpt32 database as the trigger to open the ASCII file.

Page

Allows the user to specify data to be placed on a page. Once the HEADER, PRINT, and FOOTER information for that page has been specified, another PAGE statement can be used to start the next page. The PAGE statement will proceed the data which is place on that page. PAGE directive numbers start at 1 with a maximum of 10 pages per Report file.

```
PAGE = 1
.HEADER ...
.FORMAT ...
.PRINT ...
.FOOTER ...

PAGE = 2
.HEADER ...
.FORMAT ...
.PRINT ...
.FOOTER ...
```

Print

This specifies which CRISP variables will be placed into the report when each sample is taken. The `1` identifies which .FORMAT statement to use. If the keyword `DATE` is used, the system date will be printed, and if `TIME` is used, the system time will be printed. The \ indicates line continuation. The format of the data is controlled by the corresponding `.FORMAT` directive. In the example below, the format comes from the closest format directive.

```
.PRINT, 1  =   TIME,                  \
      #TEST:VAR1  /STATS/AVERAGE,     \
      #TEST:VAR2  /STATS/AVERAGE,     \
      #TEST:VAR3  /STATS,             \
      DATE,                           \
      " some text string "
```

The user may also specify variables in the CRISP data base to be updated as the report is being generated. This will allow a CRISP Workstation screen to display data currently in the report. It is the responsibility of the user to clear out this data once the report file is closed. If not, the old data will remain in those variables until it is over written.

```
.PRINT, 1  = TIME,                                  \
   #TEST:VAR1  /STATS =#TEST:VAR1_STAT  /AVERAGE    \
   #TEST:VAR1_AVG,                                  \
   #TEST:VAR2  /STATS/AVERAGE,                      \
   #TEST:VAR3  /STATS/AVERAGE,                      \
   #TEST:VAR4  /STATS/VALUE = #TEST:VAR4_VALUE,     \
   DATE
```

Print (cont)

Where #TEST:VAR1_STAT is the first of 6 CRISP FLOAT variables to receive the STATS data on the samples currently collected. The order of this data is as follows.

```
AVERAGE
TOTAL
MAX
MIN
COUNT
BAD CNT
```

#TEST:VAR1_AVG will contain the average of the samples collected from TEST:VAR1 between the writes to the report. For example, if samples per second is 1 and write period is 10, then the average will be written as the average of the 10 samples that were gathered between the writes to the report file.

The /VALUE switch will write the current value of the variable to the CRISP database. For arrays if a 'W' is used as the subscript the current number of writes to the file will be used as an offset from the start of the array to write data back to the CRISP database. This feature allows the user to see the report on the screen as it is being built. This is useful in cases where the report is a data collection report and they want to see the current data in the report.

Example:

```
.PRINT, 1 = #TEST:FLOAT1 /VALUE = #TEST:F1_ARRAY(W)
```

Print Functions

These Keywords will provide the following functions.

Function Keywords for a single variable.

/AVERAGE	Prints the AVERAGE of all samples taken for a single a WRITE_PERIOD for time base reports. For event based reports the average of all samples taken up to the time of the WRITE_NOW variable going true is written.
/STATS	Prints the AVERAGE, TOTAL, MAXIMUM, MINIMUM, COUNT, and BAD COUNT of all WRITES to the file for the variable listed. The results are printed at the bottom of the report file.

Function Keywords for multiple variables.

AVG	Prints the average of the variables listed.
MAX	Prints the highest value of the variable or variables listed.
MIN	Prints the minimum value of the variable or variables listed.
SDEV	Prints the standard deviation of the variable or variables listed.

Print Functions (cont)

`SUM` Provide a sum of the variable or variables listed.

A single value would be printed for each of the functions listed below.

```
.PRINT = AGV ( #TEST:VAR1, #TEST:VAR2, #TEST:VAR3 )
.PRINT = MAX ( #TEST:VAR1, #TEST:VAR2, #TEST:VAR3 )
```

Print_If

This allows the user to specify a CRISP Logical variable which controls if the preceding print statement. When the variable is true, the line will be printed in the report.

```
.PRINT_IF  = #TESTDB:PRINT_LINE1
```

Queue

This allows the user to specify which print queue the report file is sent to when closed. Any valid print queue switches can be specified. The text passed with the switches must be enclosed in quotes.

NOTE

If a queue is not specified, SYS$PRINT is the default.

```
.QUEUE = "LA120_PRINT" /FORM = "10C6L"
```

`/FORM_NAME`	Determines which form is used by the print queue. These forms are defined when the System Manager defines the queue.
`/DELETE`	Will cause the report file to be deleted after printing.

Read_If

The `.READ_IF` directive specifies a CRISP Logical that when set will cause data to be read from the file specified in the `.INPUT_FILENAME.` The Logical specified will be reset when the read is complete. The following switches are valid on the `.READ_IF` directive.

```
Switch   Function
------   ------------------------------------------------

DELETE   Delete the input file when the read is complete.
         The CLOSE switch is required for this mode.

CLOSE    Close the input file after the read.
```

```
.read_if = #TESTDB:read_input_file /delete/close
```


Read_If (cont)

When `read_input_file` is true, the ASCII file is read in according to `.READ_REC_COUNT` and `.READ_ROW COUNT.` When the read operation is completed, the file is closed and deleted.

Read_Rec_Count

The `.READ_REC_COUNT` directive specifies the number of records to read from the ASCII file. Records are separated by the characters specified in the `.RECORD_DELIMITER` directive. A constant can also be used.

```
.read_rec_count   = #rpt32:row_count
```

This specifies that the record count comes from the CRISP variable `row_count` in the RPT32 database.

Record_Delimiter

Use this directive to specify a list of record delimiter characters. A CRISP string variable may be used to specify a record delimiter character. Record delimiters are used to separate the data in the ASCII file so that data can be read in pieces that match the `.INPUT_ADB_VAR_LIST.`

```
.RECORD_DELIMITER = "\n"
```

This will use the tab as a record delimiter.

Recover_Crisp_Stop

When this directive is specified, the existing file is open and appended when CRISP is stopped and restarted until the WRITES_PER_FILE or CLOSE_AT parameters have been meet. If a CLOSE_AT time has passed, then a new file will be started.

```
.RECOVER_CRISP_STOP
```

Resume

This directive will cause the REPORTER RPC process to start reading directives in the file. This directive is intended for use in a CHART/II display files that will use the same variables as the report file.

Rpt_Ucf_File

This directive allows a CRISP string to be specified. The contents of this string contains a report file FILE SPEC. This report will be started when the RUN_RPT_IF variable is set true in the CRISP data base. This CRISP Logical will then be set to a false state by the REPORTER when the report file is closed. This feature is intend to allow user selectable reports.

```
.RPT_UCF_FILE = #TEST:RUN_THIS_REPORT
```

Run_One_Time

This directive when specified will cause this report file process to exit. The intended use is for a report that manually starts using the RPR command, when required.

```
.RUN_ONE_TIME
```

Run_RPT_If

This directive allows a CRISP Logical to be specified. When this Logical is true, the report specified by the .RPT_UCF_FILE is generated. This is used in the master report file for a multiple report. See MULTI.RPT.

Sample_While_True

This is a required directive that specifies when sampling will start and stop. For event based sampling the `.SAMPLE_WHILE_TRUE` and the `.WRITE_NOW` variables must both be true. For time based sampling, if the `.SAMPLE_WHILE_TRUE` variable is true, sampling will occur based on the sample period specified with `.SECONDS_PER_SAMPLE.` `.SAMPLE_WHILE_TRUE` is basically a permissive that allows sampling to occur and initially causes the file to open. For time and event based sampling, this bit will close the file if it goes false.

```
.SAMPLE_WHILE_TRUE = CRISP Logical variable
```

Seconds_Per_Sample

The `.SECONDS_PER_SAMPLE` directive is used with time and event based reporting and specifies the time period between samples. If averaging is specified, the average is stored for each write to the file. If the directive is not specified, it defaults to 1. The `.SECONDS_PER_SAMPLE` value can not cause sampling to occur more frequently than once per second. The sec, min, hour qualifier must be specified if a CRISP Long variable is not specified. If a CRISP Long variable is specified, the default sampling rate is in seconds and a time qualifier is not used.

```
.SECONDS_PER_SAMPLE = number [SEC | MIN | HOUR]

.SECONDS_PER_SAMPLE = 1, MIN

.SECONDS_PER_SAMPLE = #TEST:SAMPLE_RATE
```

NOTE

For event based sampling that uses averaging, this value specifies the sample interval for the average. Frequent sampling for the purpose of averaging will increase the CPU loading.

Stats_Format

This specifies the format for the variables on which statistics are performed. The statistics variables are printed in the same order as in the .PRINT directive except that only the variables with the /STATS switch are printed. The same format specifiers apply as for .FORMAT.

```
.HEADER =  "Hourly AG   Flow     Flow     Flow     Flow"
.HEADER =  "Written at  A        B        C        D"
.HEADER =  "----------  ------   -----    ------   -----"
.FORMAT ="   #SSSSSSSS   #FF.F   #FF.F     #FF.F   #FF.F"
.STATS_FORMAT = "        #FF.F   #FF.F     #FF.F   #FF.F"
```

NOTE

The STATS keywords that describe the data, ie; TOTAL, MIN, MAX, will occupy the first nine characters of the line. Spaces outside of the quotes do not effect the position of the text. This is convenient in lining up data columns.

Stats_Header

This directive allows a text string to be specified. The specified data is printed above the statistics and may be used to provide headings for the statistics columns.

```
.STATS_HEADER = "text"
```

Stats_Label

This directive allows different strings to be used as labels for the STATS lines.

```
.STATS_LABEL = "TO",\
               "AV",\
               #RPT32:ASTR(0),\
               "MN",\
               "CN",\
               "BD"
```

The following is the result.

```
TO       60    75    150   300
AV       20    25     50   100
MAX(2)   50    60    110   210
MN       15    20     45    95
CN        3     3      3     3
BN        0     0      0     0
```

STR_White_Space

This directive allows characters to be specified as white space for ASCII input files being inputted into CRISP strings. The white space character is the example below is a ". All format characters require a preceding backslash. white space characters must be enclosed in quotes.

```
.STR_WHITE_SPACE = "\""
```

Symbol

(**Not required for REPORTER V2.0**) The .SYMBOL directive is used to identify the start of a variable list. Each entry in the variable list has the following general fields.

```
.SYMBOL
index/db:name/option=argument
index/db:name/option=argument
index/db:name/switch
```

Where:

index	is any unique numeric value that is used by other commands to reference a CRISP variable.
db	is a number specifying which database from the DATA_BASE command is to be used.
name	is any legal CRISP variable name in database db.
switch	specifies some optional information about the variable. See below.

The following options are available.

/AVERAGE	Average this variable between writes.
/STATS	Provide minimum, maximum, average, and total at the bottom of the column for this variable.

Vars_Per_Record

The .VARS_PER_RECORD specifies the number of fields expected between records in the ASCII file. A CRISP Numeric variable or Constant can also be used.

```
.vars_per_record  = 3
```

This specifies that we expect 3 fields for every record in the ASCII file.

White_Space

Use this directive to specify a list of white space characters. White space is ignored when the ASCII file is read. When the data type is a CRISP String, then the white space is not ignored. A CRISP string variable may be used to specify a white space character.

```
.WHITE_SPACE      = " ","\t"
```

This specifies a space and a tab as white space.

Write_At

The .WRITE_AT directive is used when the .PRINT directives are executed. This allows the user to specify a particular time of day, week, or month the information should be written to the file. Up to 200 entries can be placed on the .WRITE_AT directive. The following is the basic syntax of the entry on the .WRITE_AT directive. The following valid keywords can be used.

```
END_OF_HOUR or END_OF_DAY or END_OF_WEEK or END_OF_MONTH

[DAILY,MON,TUE,WED,THU,FRI,SAT,SUN]   @   HH:MM:SS

              or

[MONTHLY,JAN,FEB, ... DEC]   ##  @  HH:MM:SS
```

Where :

##	is a day of the month
HH	is the hour
MM	is the minutes
SS	is the seconds.

NOTE

One of the KEYWORDS must be specified.

The entries on the .WRITE_AT directive must be separated by commas.

Example :

```
.WRITE_AT = DAILY @08:00:00,MON @09:00:00,JAN 1 @10:00:00
```

Write_Now

When this directive is used, it specifies Event based sampling. When the CRISP Logical variable specified for .WRITE_NOW goes from false to true, a sample is placed in the file. The .WRITE_NOW variable is cleared when the sample is written to the file acknowledging to the CRISP application that the sample has been written. Note that each report should have its own .WRITE_NOW variable.

```
.WRITE_NOW = #TEST:WRITE_DATA
```

Write_Per_File

The file will close and a new one will open when the number of samples are placed into the file. This directive is optional for event or time based sampling. When this directive or the CLOSE_AT directive is satisfied, the file closes and a new one starts.

```
.WRITE_PER_FILE = CRISP Long variable
```

Other Keywords are as follows.

CUR_MONTH	Use the number of days in the current month.
APPEND_MODE	The data will always be appended to this file.

Write_Period

When this directive is used, it forces time based sampling and specifies the period at which records are written in either seconds, minutes, or hours.

```
.WRITE_PERIOD = number [SEC | MIN | HOUR]
```

NOTE

A time qualifier must be used if a CRISP Long variable is not specified.

If a CRISP Long variable is specified, then the default time period is in seconds and a time qualifier is not used.

```
.WRITE_PERIOD = #TEST:WRITE_PERIOD_IN_SECONDS
```

NOTE

Also see WRITE_AT

Notes:

```
! AVG.RPT
! The following is the output of this report:
!-------------------------------------------------------------------------
! 14-NOV-1991 14:52:31.04
!
!               DATE        AN(0)    A        B        C      STR   astr(0)  swt    fos
!-------------------------- -----  -------- -------- -------- ----- -------  -----  -----
!  Thu Nov 14 14:52:32 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:35 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:38 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:41 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:44 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:47 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:50 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:53 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:56 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!  Thu Nov 14 14:52:59 1991     1      7.30     5.00     1.00 string 1 astr(0)      1      1
!           AN(0)    A        B        C
!TOTAL   :    10     73.00    50.00    10.00
!AVERAGE :     1      7.30     5.00     1.00
!MIN     :     1      7.30     5.00     1.00
!MAX     :     1      7.30     5.00     1.00
!COUNT   :    10     10.00    10.00    10.00
!BAD CNT :     0      0.00     0.00     0.00
!Total writes : 10
!
!----------DONE-------------
!
!-------------------------------------------------------------------------
! This report configuration defines a TIME based report.  When the
! .SAMPLE_WHILE_TRUE variables goes true a sample is taken at the interval
! determined by the .SECONDS_PER_SAMPLE (Once a second). A record
! is written when interval specified by .WRITE_PERIOD (3 seconds) has passed.
! The data in the record is the average of the samples collected once a second.
! When the file is closed STATS are written on four of the variables AN(0), A,
! B, and C. A new file is created when the number of records written are
! equal to the .WRITE_PER_FILE specification of 10. Each record
! is generated by one print statement. The print statement will place
! the DATE and TIME as a string then the value of AN(0), A, B, C, STR, ASTR(0),
! SWT and FOS on one line. This file also demonstrates the ability to write back
! to the CRISP data base the current value of the STAT data for variable A. The
! values collected by the reporter for each record are written back for
! variables A and B. The values for B use the W index which contains the number
! of the record just written to. The output is written to a file named AVG.TMP
!-------------------------------------------------------------------------
```

```
.DATA_BASE = RPT32
.FILE_NAME = "avg.tmp"
.WRITE_PERIOD = 3 SEC
.FORMAT_MODE = LB_FMT
.SECONDS_PER_SAMPLE = 1 SEC
.WRITE_PER_FILE = 10
.SAMPLE_WHILE_TRUE =  #rpt32:swt
.FILE_OPEN_STATUS =   #rpt32:fos
.BAD_DATA_VALUE = -9999

.header = #CRISP:csp_s_time
.HEADER =   "               DATE        AN(0)    A        B        C      STR   astr(0)  swt    fos  "
.HEADER =   "-------------------------- -----  -------- -------- -------- ----- -------  -----  -----"
.format,1 = "#SSSSSSSSSSSSSSSSSSSSSSSS #DDDD  #FFFF.FF #FFFF.FF #FFFF.FF #SSSS #ssssss  #dddd  #dddd"


.print,1 =                      DATE,\
       #rpt32:AN(0) /STATS             /AVERAGE,\
       #rpt32:A     /STATS=#rpt32:A_TOT_STAT   /AVERAGE = #RPT32:A_AVG,\
       #rpt32:B     /STATS             /AVERAGE = #RPT32:B_AVG(W),\
       #rpt32:C     /STATS             /AVERAGE,\
       #rpt32:STR1,\
       #rpt32:ASTR(0),\
       #rpt32:swt,\
       #rpt32:fos

.STATS_HEADER = " An(0)    A        B        C"
.STATS_FORMAT = "#DDDD  #FFFF.FF #FFFF.FF #FFFF.FF "
.footer = "\n----------DONE-------------\n"

.SYMBOL
```

```
! DAILY.RPT
! The following is the output of this report :
!-------------------------------------------------------------------------
!
!  time     float1 float2 float3 float4 float5 float6 float7
!  -------- ------ ------ ------ ------ ------ ------ ------
!  01:00:09   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  02:00:10   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  03:00:11   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  04:00:12   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  05:00:13   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  06:00:14   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  07:00:15   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  08:00:16   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  09:00:17   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  10:00:18   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  11:00:19   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  12:00:20   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  13:00:21   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  14:00:22   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  15:00:23   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  16:00:24   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  17:00:25   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  18:00:26   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  19:00:27   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  20:00:28   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  21:00:29   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  22:00:30   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  23:00:31   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  00:00:32   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!             FLOAT1    FLOAT5
!  TOTAL   :    266.64   1333.18
!  AVERAGE :     11.11     55.55
!  MIN     :     11.11     55.55
!  MAX     :     11.11     55.55
!  COUNT   :     24.00     24.00
!  BAD CNT :      0.00      0.00
!  Total writes : 24
!
!-------------------------------------------------------------------------
! This report configuration defines an EVENT based report.  When the
! .WRITE_AT condition is satisfied a record is written to the report. Every
! time the .SECONDS_PER_SAMPLE elapses a sample is taken. The average of FLOAT1
! and FLOAT5 taken each second are used for the record written. Since /STATS
! on FLOAT1 and FLOAT5 were specified statistics for those two variables
! will be written at the end of the report.  The output of the report will
! be written to a file named EVENT.TMP. The report is closed when the .CLOSE_AT
! condition is satisfied.
!-------------------------------------------------------------------------
```

```
!
! -------------------------------------
! the following section describes
! the control information for the report
! -------------------------------------

.FILE_NAME = "daily.TMP"            ! the report output file
.SECONDS_PER_SAMPLE = 1 SEC      ! check RPT32:SWT every second
.WRITE_AT= END_OF_HOUR           ! write a record every hour
.WRITE_PER_FILE = 24             ! close the file after 24 records
.CLOSE_AT = END_OF_DAY           ! close the file at midnight
.SAMPLE_WHILE_TRUE  = #RPT32:SWT    ! start reporting if this bit is true
.FILE_OPEN_STATUS   = #RPT32:FOS    ! set this when report output file is open
.FORMAT_MOD = C_FMT              ! select the 'C' language format specifiers

.BAD_DATA_VALUE  = -9999     ! if a value is equal to -9999 then do not average it



! ----------------------------------------------------------
! this section describes the look and contents of the report
! ----------------------------------------------------------

.PRINT,1 =  TIME ,\                   ! the data to be reported
       #RPT32:FLOAT1 /AVERAGE/STATS ,\
       #RPT32:FLOAT2           ,\
       #RPT32:FLOAT3           ,\
       #RPT32:FLOAT4           ,\
       #RPT32:FLOAT5 /AVERAGE/STATS ,\
       #RPT32:FLOAT6           ,\
       #RPT32:FLOAT7


.header =   "time     float1 float2 float3 float4 float5 float6 float7"
.header =   "-------- ------ ------ ------ ------ ------ ------ ------"
.FORMAT,1 = "%8s %6.1f %6.1f %6.1f %6.1f %6.1f %6.1f %6.1f "   ! the format

.STATS_HEADER = "FLOAT1    FLOAT5"  ! a header for the statistics
.STATS_FORMAT = "%9.2f %9.2f"       ! the format of the statistics

.END                             ! end of the report
```

```
! DAY.RPT
! The following is the output of this report :
!-------------------------------------------------------------------------------
!14-NOV-1991 14:56:24.04

!

!              DATE        long1  AN(0)     A          astr(0)       swt    fos

!------------------------- -----  --------  -------   -----------   -----  -----

!  Thu Nov 14 14:56:33 1991   111      1         7.30       astr(0)      1      1  <--line 1 record 1

!    Thu Nov 14 14:56:33 1991 14:56:33  1991   Nov    14    14    56   Thu         <--line 2 record 1

!  Thu Nov 14 14:56:43 1991   111      1         7.30       astr(0)      1      1  <--line 1 record 2

!    Thu Nov 14 14:56:43 1991 14:56:43  1991   Nov    14    14    56   Thu         <--line 2 record 2

!  Thu Nov 14 14:56:53 1991   111      1         7.30       astr(0)      1      1  <--line 1 record 3

!    Thu Nov 14 14:56:53 1991 14:56:53  1991   Nov    14    14    56   Thu         <--line 2 record 3

!  Thu Nov 14 14:57:03 1991   111      1         7.30       astr(0)      1      1  <--line 1 record 4

!    Thu Nov 14 14:57:03 1991 14:57:03  1991   Nov    14    14    57   Thu         <--line 2 record 4

!  Thu Nov 14 14:57:13 1991   111      1         7.30       astr(0)      1      1  <--line 1 record 5

!    Thu Nov 14 14:57:13 1991 14:57:13  1991   Nov    14    14    57   Thu         <--line 2 record 5

!

!----------DONE-------------

!-------------------------------------------------------------------------------
! This report configuration defines an TIME based report.  When the
! .SAMPLE_WHILE_TRUE variables goes true a sample is taken at the interval
! determined by the .SECONDS_PER_SAMPLE (RPT32 data base variable SPS). A record
! is written when interval specified by .WRITE_PERIOD (RPTC32 data base variable
! W_PERIOD) has passed. A new file is created when the number of records
! written are equal to the .WRITE_PER_FILE (RPTC32 data base variable WPF). Each
! record is generated by two print statements. The first print statement will
! place the DATE and TIME as a string then, the value of LONG1, AN(0), A,
! ASTR(0), SWT and FOS on one line. The next line will contain the DATE then
! TIME, YEAR, MONTH, DAY, HOUR, MIN, DOW (Day Of the Week).
!-------------------------------------------------------------------------------
```

```
.DATA_BASE = RPT32
.FILE_NAME = "day.tmp"
.WRITE_PERIOD = #RPT32:W_PERIOD
.FORMAT_MODE = LB_FMT

.SECONDS_PER_SAMPLE = #RPT32:SPS
.WRITE_PER_FILE = #RPT32:WPF

.SAMPLE_WHILE_TRUE =  #rpt32:swt
.FILE_OPEN_STATUS =   #rpt32:fos
.BAD_DATA_VALUE = -9999

.header = #CRISP:csp_s_time
.HEADER =   "               DATE        long1  AN(0)     A          astr(0)       swt
                          fos  "
.HEADER =   "-------------------------- -----  --------  -------   -----------   -----
                          -----"
.format,1 = "#SSSSSSSSSSSSSSSSSSSSSSSS #dddd  #DDDD     #FFFF.FF  #sssssssssss  #dddd
                          #dddd"


.print,1 =                DATE,\
      #rpt32:long1,\
      #rpt32:an(0),\
      #rpt32:A,\
      #rpt32:ASTR(0),\
      #rpt32:swt,\
      #rpt32:fos

.PRINT, 2 = date,time,year,month,day,hour,min,dow
.format,2 = "#sssssssssssssssssssssssssss #sssssss #ssss #ssss #ssss #ssss #ssss #ssss"
.footer = "\n----------DONE-------------\n"

.SYMBOL
```

```
! EVENT.RPT
! The following is the output of this report :
!-------------------------------------------------------------------------
!
!  time     float1 float2 float3 float4 float5 float6 float7
!  -------- ------ ------ ------ ------ ------ ------ ------
!
!  16:26:14   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  16:26:16   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  16:26:18   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  16:26:20   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!  16:26:22   11.1   22.2   33.3   44.4   55.5   66.7   77.8
!            FLOAT1    FLOAT5
!  TOTAL   :     55.55    277.75
!  AVERAGE :     11.11     55.55
!  MIN     :     11.11     55.55
!  MAX     :     11.11     55.55
!  COUNT   :      5.00      5.00
!  BAD CNT :      0.00      0.00
!  Total writes : 5
!
!-------------------------------------------------------------------------
! This report configuration defines an EVENT based report.  When the
! .WRITE_NOW variable goes true a record is written to the report.  Every
! time the .SECONDS_PER_SAMPLE elapses a sample is taken. The average of these
! samples taken of FLOAT1 and FLOAT5 are written to the record. Since /STATS on
! FLOAT1 and FLOAT5 were specified statistics for those two variables
! will be written at the end of the report.  The output of the report will
! be written to a file named EVENT.TMP.
!-------------------------------------------------------------------------
!
```

```
! --------------------------------------
! the following section describes
! the control information for the report
! --------------------------------------

.FILE_NAME = "EVENT.TMP"     ! the report output file
.SECONDS_PER_SAMPLE = 1 SEC     ! check RPT32:WNOW and RPT32:SWT every second
.WRITE_NOW                     = #RPT32:WNOW ! write a record if this bit is true
.WRITE_PER_FILE     = 5         ! close the file after 5 records
.SAMPLE_WHILE_TRUE  = #RPT32:SWT  ! start reporting if this bit is true
.FILE_OPEN_STATUS   = #RPT32:FOS  ! set this when report output file is open
.FORMAT_MODE                   = C_FMT   ! select the 'C' language format specifiers

.BAD_DATA_VALUE     = -9999     ! if a value is equal to -9999 then do not average it

! ----------------------------------------------------------
! this section describes the look and contents of the report
! ----------------------------------------------------------

.PRINT,1 =  TIME ,\                    ! the data to be reported
       #RPT32:FLOAT1 /AVERAGE/STATS ,\
       #RPT32:FLOAT2           ,\
       #RPT32:FLOAT3           ,\
       #RPT32:FLOAT4           ,\
       #RPT32:FLOAT5 /AVERAGE/STATS ,\
       #RPT32:FLOAT6           ,\
       #RPT32:FLOAT7


.header =   "time     float1 float2 float3 float4 float5 float6 float7"
.header =   "-------- ------ ------ ------ ------ ------ ------ ------"
.FORMAT,1 = "%8s %6.1f %6.1f %6.1f %6.1f %6.1f %6.1f %6.1f "   ! the format

.STATS_HEADER = "FLOAT1    FLOAT5"     ! a header for the statistics
.STATS_FORMAT = "%9.2f %9.2f"         ! the format of the statistics

.END                                    ! end of the report
```

```
! FORMAT.RPT
! The following is the output of this report :
!-------------------------------------------------------------------------------
!14-NOV-1991 14:59:03.04
!
!sub        AN(0)  astr(0)
!---        -----  -------------
!  0            1       astr(0)
!  1           12       astr(1)     <-- RECORD 1
!  2           13       astr(2)
! -----
!  0            1       astr(0)
!  1           12       astr(1)     <-- RECORD 2
!  2           13       astr(2)
! -----
!  0            1       astr(0)
!  1           12       astr(1)     <-- RECORD 3
!  2           13       astr(2)
! -----
!
!----------DONE-------------
!
!-------------------------------------------------------------------------------
! This report configuration defines an TIME based report.  When the
! .SAMPLE_WHILE_TRUE variable goes true a sample is collected at the interval
! specified by .SECONDS_PER_SAMPLE (every second). When
! the time period specified by .WRITE_PERIOD has passed the latest sample is
! written to the file. Each record contains values from AN(0), ASTR(0), AN(1),
! ASTR(1), AN(2), ASTR(2). The purpose of this file is to demonstrate that the
! same format statement can be used more than once. When three records have
! been written as specified by .WRITE_PER_FILE a new file is created. The output
! of the report will be written to a file named FORMAT.RPT.
!-------------------------------------------------------------------------------
```

```
.DATA_BASE = RPT32
.FILE_NAME = "format.tmp"
.WRITE_PERIOD = 3 SEC
.FORMAT_MODE = LB_FMT
.SECONDS_PER_SAMPLE = 1 SEC
.WRITE_PER_FILE = 3
.SAMPLE_WHILE_TRUE =  #rpt32:swt
.FILE_OPEN_STATUS =   #rpt32:fos
.BAD_DATA_VALUE = -9999

.header = #CRISP:csp_s_time
.HEADER =   "sub        AN(0)  astr(0)      "
.HEADER =   "---        -----  -------------"
.format,1 = " #S        #DDDD  #sssssssssss "
.format,2 = " #ssss"

.print,1 =                "0",\
      #rpt32:an(0),\
      #rpt32:ASTR(0)

.print,1 =                "1",\
      #rpt32:an(1),\
      #rpt32:ASTR(1)

.print,1 =                "2",\
      #rpt32:an(2),\
      #rpt32:ASTR(2)

.print,2 = "-----"

.footer = "\n----------DONE-------------\n"

.SYMBOL
```

```
!
!  file : INPUT.RPT
!
! sample configuration file for ASCII input
!-----------------------------------------
!
! The following is a copy of the input.txt ASCII text file
! for reference purposes.
!-----------------------------------------
!  1.1                    1  STR1
!  1.2                    2  STR2
!  1.3                    3  STR3
!  1.4                    4  STR4
!  1.5                    5  STR5
!  1.6                    6  STR6
!-----------------------------------------
! Move ASCII file data to a CRISP Database
!  Note that this file uses the rpt32 CRISP database.
!
! INPUT.TXT is the name of the ASCII file which will be opened when
! .OPEN_INPUT_FILE_IF variable is true. The Reporter will continue to try an
! open the file until success is obtained. Upon success the
! .INPUT_FILE_OPEN_STATUS is set TRUE. The ASCII file is closed when the
! .CLOSE_INPUT_FILE_IF variable goes TRUE or when a READ operation is performed
! because the /CLOSE switch has specified. The ASCII input file is also deleted
! once READ and CLOSED since the /DELETE switch has been specified with the
! .READ directive. The .OPEN_INPUT_FILE_IF and .INPUT_FILE_OPEN_STATUS
! variables are both cleared when the file is closed. When the .READ_IF
! variable is TRUE the value of the .FIRST_RECORD variable is used to determine
! the number of records skipped. The .READ_REC_COUNT determines
! how many records are read each time the .READ_IF variable goes TRUE. The
! number of variables in a record of the ASCII file is determined by
! .VARS_PER_RECORD. This must match the number of variables in the CRISP Data
! base specified by .INPUT_ADB_VAR_LIST.
!
.INPUT_FILENAME            = "input.txt"
.SECONDS_PER_SAMPLE        = 1 sec
.FIELD_DELIMITER           = "\t"
.RECORD_DELIMITER          = "\n"
.WHITE_SPACE                  = " "
.open_input_file_if        = #rpt32:open_input_file
.close_input_file_if       = #rpt32:close_input_file
.input_file_open_status    = #rpt32:input_file_open
.read_rec_count            = #rpt32:row_count
.read_if                      = #rpt32:read_if /close/delete
.vars_per_record           = 3
.first_record                 = #rpt32:first_row
.app_db                       = "rpt32"
.input_adb_var_list        = #float1,#long1,#str1
.end
```

Notes:

```
!
! Sample ASCII input file INPUT2.RPT
!
!  1,"fred van eijk",3.4
!
! Move ASCII file data to a CRISP Database
!
! This file demonstrates how to handle white space in a string. This string has
! a double quotation character that requires the white space spec "\"" to be
! used. The other characters which require a backslash are:
!  All special formatting characters i.e. \n, \t, \v, \b, etc. Reference the
! Directives section of the REPORTER User's Guide for a complete list.
!
.INPUT_FILENAME            = "t.txt"
.SECONDS_PER_SAMPLE        = 1 sec
.FIELD_DELIMITER           = ","
.RECORD_DELIMITER          = "\n"
.WHITE_SPACE                  = " "
.str_WHITE_SPACE           = "\""
.open_input_file_if        = #rpt32:open_input_file
.close_input_file_if       = #rpt32:close_input_file
.input_file_open_status    = #rpt32:input_file_open
.read_rec_count            = #rpt32:row_count
.read_if                      = #rpt32:read_if
.vars_per_record           = 3
.first_record                 = #rpt32:first_row
.app_db                       = "rpt32"
.input_adb_var_list        = #long1,#str1,#float1
.end
```

Notes:

```
! LOG.RPT
!-------------------------------------------------------------------------
! The following report configuration file will generate a report
! that looks as follows :
!-------------------------------------------------------------------------
!                STATUS LOG REPORT
!
!Report started at : Thu Nov 14 15:01:29 1991
!-----------------------------------------------
!Thu Nov 14 15:01:36 1991 PRINT_IF_1 went true      FLOAT1 = 11.110000
!Thu Nov 14 15:01:42 1991 PRINT_IF_2 changed state  FLOAT2 = 22.219999
!Thu Nov 14 15:01:44 1991 PRINT_IF_2 changed state  FLOAT2 = 22.219999
!Thu Nov 14 15:01:50 1991 PRINT_IF_4 changed state  FLOAT4 = 44.439999
!Thu Nov 14 15:01:52 1991 PRINT_IF_4 changed state  FLOAT4 = 44.439999
!Thu Nov 14 15:01:58 1991 PRINT_IF_3 went false     FLOAT3 = 33.330002
!Thu Nov 14 15:02:06 1991 A changed VALUE      A = 9.100000
!Thu Nov 14 15:02:18 1991 LONG1 changed VALUE  LONG1 = 123
!Thu Nov 14 15:02:26 1991 AN(0) changed VALUE  AN(0) = 777
!Thu Nov 14 15:02:38 1991 STR1 changed VALUE   STR1 = new str
!-----------------------------------------------
!Report completed at : 15:02:52
!
!
!-------------------------------------------------------------------------
! This is a TIME based report and creates a file named LOG.TMP.
! The header consists of three lines and is printed when the
! RPT32:SWT CRISP logical variable first goes true.
! Each of the log lines are scanned  every 2 seconds.
! This whole process is repeated as long as
! the CRISP logical variable RPT32:SWT remains TRUE.
! Note that the .SECONDS_PER_SAMPLE = 2 SEC statement controls how often
! the CRISP logical variable RPT32:SWT is checked and also
! the rate at which the log bits are checked.
!
!-------------------------------------------------------------------------
! --------------------------------------
! the following section describes
! the control information for the report
! --------------------------------------

.FILE_NAME                    = "log.TMP"   ! write report to file TIME.TMP
.SECONDS_PER_SAMPLE = 2 SEC                 ! check .sample_while_true every 2 sec
.WRITE_PER_FILE     = 10                    ! write 10 lines to the report
.WRITE_NOW          = #rpt32:wnow           ! write a line now
.SAMPLE_WHILE_TRUE  = #rpt32:swt            ! the sample while true bit
.FILE_OPEN_STATUS   = #rpt32:fos            ! the file open status bit

.FORMAT_MODE = LB_FMT                       ! use the #sss format mode

.BAD_DATA_VALUE     = -9999      ! if a value = -9999 then it is
                                 ! ignored for averaging and statistics
```

```
! ---------------------------------------------------------
! this section describes the look and contents of the report
! ---------------------------------------------------------

.HEADER   = "                STATUS LOG REPORT\n"              ! 3 lines of header
.HEADER   = "Report started at : ",DATE
.HEADER   = "-----------------------------------------------"
.FORMAT,1 = "#ssss"
.FOOTER   = "-----------------------------------------------" ! 2 lines of footer
.FOOTER   = "Report completed at : ",time

.print,1  = "-----"

! ---------------------------------------
! Below is the logging section of the report
! ---------------------------------------

.LOG_IF POSITIVE #RPT32:PRINT_IF_1 THEN DATE," PRINT_IF_1 went true      ","FLOAT1 = ",#RPT32:FLOAT1

.LOG_IF CHANGE   #RPT32:PRINT_IF_2 THEN DATE," PRINT_IF_2 changed state  ","FLOAT2 = ",#RPT32:FLOAT2

.LOG_IF NEGATIVE #RPT32:PRINT_IF_3 THEN DATE," PRINT_IF_3 went false     ","FLOAT3 = ",#RPT32:FLOAT3

.LOG_IF CHANGE   #RPT32:PRINT_IF_4 THEN DATE," PRINT_IF_4 changed state  ","FLOAT4 = ",#RPT32:FLOAT4


.LOG_IF CHANGE   #RPT32:A     THEN DATE," A changed VALUE      ","A = ",#RPT32:A          ! A FLOAT

.LOG_IF CHANGE   #RPT32:LONG1 THEN DATE," LONG1 changed VALUE  ","LONG1 = ",#RPT32:LONG1  ! A LONG

.LOG_IF CHANGE   #RPT32:AN(0) THEN DATE," AN(0) changed VALUE  ","AN(0) = ",#RPT32:AN(0)  ! A NUMERIC

.LOG_IF CHANGE   #RPT32:STR1  THEN DATE," STR1 changed VALUE   ","STR1 = ",#RPT32:STR1    ! A STRING


.END                                    ! end of the report
```

```
! MATH.RPT
! Sample Output file
!
!14-NOV-1991 15:04:21.04
!
!              MATH.RPT
!               DATE        A        B
!-------------------------- -------- --------
!  Thu Nov 14 15:04:21 1991    18.20     2.00
!  Thu Nov 14 15:04:21 1991    18.20     2.00
!  Thu Nov 14 15:04:22 1991    18.20     2.00
!  Thu Nov 14 15:04:23 1991    18.20     2.00
!  Thu Nov 14 15:04:24 1991    18.20     2.00
!             A         B
!TOT * 3   273.00     30.00
!AVG        18.20      2.00
!MIN        18.20      2.00
!MAX        18.20      2.00
!CNT         5.00      5.00
!BADCNT      0.00      0.00
!Total writes : 5
!
!----------DONE-------------
!
! This file is a TIME based report which uses the REPORTER Math functions.
! When the .SAMPLE_WHILE_TRUE variable is TRUE sample are collected at a 1
! Second rate as specified by the .SECONDS_PER_SAMPLE. A record is written at
! the .WRITE_PERIOD value of once a second. The DATE, RTP32 variable A's
! current value is multiplied by 2. This value is then placed in the Report.
! The value of B is multiplied by 2. This value is placed in the Report and
! written back to the RPT32 data base variable B_AVG(W). The B_AVG is an array
! W is used as the INDEX and will be the current record. This also demonstrates
! the ability to rename the STATS labels.
!
.DATA_BASE = RPT32
.FILE_NAME = "math.tmp"
.WRITE_PERIOD = 1 SEC
.FORMAT_MODE = LB_FMT
.SECONDS_PER_SAMPLE = 1 SEC
.WRITE_PER_FILE = 5
.SAMPLE_WHILE_TRUE =  #rpt32:swt
.FILE_OPEN_STATUS =   #rpt32:fos
.BAD_DATA_VALUE = -9999

.header = #CRISP:csp_s_time
.HEADER =   "              MATH.RPT              "
.HEADER =   "               DATE        A        B        "
.HEADER =   "-------------------------- -------- -------- "
.format,1 = "#SSSSSSSSSSSSSSSSSSSSSSSS #FFFF.FF #FFFF.FF "
```

```
.print,1 =                 DATE                      ,\
      #rpt32:A/TOT = CUR_VAL * 3;/MAXIMUM/MINIMUM/AVG/CNT/BCNT/CALC = CUR_VAL * 2; ,\
      #rpt32:B/TOT = CUR_VAL * 3;/MAXIMUM/MINIMUM/AVG/CNT/BCNT/calc=cur_val  * 2;/value
                           = #rpt32:b_avg(W)

.STATS_LABEL =             "TOT * 3 ",\
      "AVG     ",\
      "MIN     ",\
      "MAX     ",\
      "CNT     ",\
      "BADCNT  "

.STATS_HEADER = "   A         B"
.STATS_FORMAT = "#FFFF.FF  #FFFF.FF"
.footer = "\n----------DONE-------------\n"

.SYMBOL
```

```
! MATH2.RPT
! Sample Output File
!
!14-NOV-1991 15:07:24.04
!
!              MATH2.RPT
!    A
!---------
!  779.30
!  779.30
!  779.30
!  779.30
!  779.30
!
!----------DONE-------------
!
! This file is a TIME base Report. When the .SAMPLE_WHILE_TRUE variable SWT is
! TRUE samples are collected at the rate specified by .SECONDS_PRE_SAMPLE. A
! record is written to the file every second. The value of A is calculated
! by RPT32 data base variables B times C times AN(0) divided by PRINT_IF_1
! and 2.3 is added. The result of this calculation is written back to the
! RPT32 data base variable A. When the number of writes to the file equals
! the .WRITE_PER_FILE variable a new file will be created.
!
.DATA_BASE = RPT32
.FILE_NAME = "math2.tmp"
.WRITE_PERIOD = 1 SEC
.FORMAT_MODE = LB_FMT
.SECONDS_PER_SAMPLE = 1 SEC
.WRITE_PER_FILE = 5
.SAMPLE_WHILE_TRUE =  #rpt32:swt
.FILE_OPEN_STATUS =   #rpt32:fos
.BAD_DATA_VALUE = -9999

.header = #CRISP:csp_s_time
.HEADER =   "              MATH2.RPT              "
.HEADER =   "    A "
.HEADER =   "---------"
.format,1 = "#FFFF.FF "


.print,1 =                #rpt32:A/CALC = #rpt32:b * #rpt32:c * #rpt32:an(0) /
                          #rpt32:print_if_1 + 2.3 ;/value = #rpt32:a

.footer = "\n----------DONE-------------\n"

.SYMBOL
```

Notes:

```
! MULTI.RPT
!-------------------------------------------------------------------------
! This report configuration file is typical of one that can be used
! for reports that do no require any averaging or statistics
! over a period of time.  Typical applications might be :
!
!  1) - Batch Reports
!  2) - System Status Reports
!  3) - Periodic Reports
!
! This report is executed by setting a bit that causes a report that is
! named in a CRISP string variable to be executed.
!
!
! The following is a sample report configuration file which will use
! MULTI1.RPT and MULTI2.RPT.
!-------------------------------------------------------------------------


.file_name = "multi.tmp"      ! location of report output
.format_mode = LB_FMT                      ! use #sss format

.run_rpt_if = #rpt32:run_rpt_bit
.rpt_ucf_file = #rpt32:rpt_ucf_filename

.print,1  = #CRISP:CSP_S_TIME
.format,1 = "The current CRISP time     : #sssssssssssssssssssssss "

.print,2  = #rpt32:float1
.format,2 = "The value of rpt32:float1  : #ffff.ff "

.end
```

Notes:

```
! Sample Output File MULTI1.RPT
!
!The current CRISP time     :   14-NOV-1991 15:06:03.04
!The value of rpt32:float1  :    11.11
!
!
!-------------------------------------------------------------------------
! This report configuration file is typical of one that can be used
! for reports that do no require any averaging or statistics
! over a period of time.  Typical applications might be :
!
!  1) - Batch Reports
!  2) - System Status Reports
!  3) - Periodic Reports
!
! This report is executed by setting the .RUN_RPT_IF variable in MULTI.RPT and
! the .RPT_UCF_FILE variable contains the name of this file. This will cause
! this report file to run and exit when the file is closed. This file will
! print the CRISP data base string CSP_S_TIME (Current VMS Time) and the value
! of RPT32 data base variable FLOAT1.
!
! The following is a sample report configuration file.
!-------------------------------------------------------------------------


.file_name = "multi1.tmp"     ! location of report output
.format_mode = LB_FMT                       ! use #sss format

.print,1  = #CRISP:csp_s_time
.format,1 = "The current CRISP time     : #sssssssssssssssssssssss "

.print,2  = #rpt32:float1
.format,2 = "The value of rpt32:float1  : #ffff.ff "

.end
```

Notes:

```
! Sample Output File MULTI2.RPT
!
!The current CRISP time     :   14-NOV-1991 15:06:24.04
!The value of rpt32:float2  :    22.22
!
!-------------------------------------------------------------------------
! This report configuration file is typical of one that can be used
! for reports that do no require any averaging or statistics
! over a period of time.  Typical applications might be :
!
!  1) - Batch Reports
!  2) - System Status Reports
!  3) - Periodic Reports
!
! This report is executed by setting the .RUN_RPT_IF variable in MULTI.RPT and
! the .RPT_UCF_FILE variable contains the name of this file. This will cause
! this report file to run and exit when the file is closed. This file will
! print the CRISP data base string CSP_S_TIME (Current VMS Time) and the value
! of RPT32 data base variable FLOAT2.
!
!
! The following is a sample report configuration file.
!-------------------------------------------------------------------------


.file_name = "multi2.tmp"     ! location of report output
.format_mode = LB_FMT                       ! use #sss format

.print,1  = #CRISP:CSP_S_TIME
.format,1 = "The current CRISP time     : #sssssssssssssssssssssss "

.print,2  = #rpt32:float2
.format,2 = "The value of rpt32:float2  : #ffff.ff "

.end
```

Notes:

```
! ONE.RPT
! This report configuration file is typical of one that can be used
! for reports that do no require any averaging or statistics
! over a period of time.  Typical applications might be :
!
!  1) - Batch Reports
!  2) - System Status Reports
!  3) - Periodic Reports
!  4) - Ad-Hoc reports
!
! This report is executed by :
!
!  1) - typing a command at DCL
!  2) - executing a DCL command file
!  3) - submitting the command file to the Batch Queue
!
! The advantage to using this type of report is that there is no requirement
! to have a process running continuously that monitors some bit that
! controls when the report is to run.
!
! To execute this type of report from CRISP the SUBMIT_BATCH CRISP LOGIC
! FUNCTION CALL can be used to execute a command file that runs the report.
!
! The following is a sample report configuration file.


.file_name = "one.tmp"        ! location of report output
.format_mode = LB_FMT         ! use #sss format

.print,1  = #CRISP:CSP_S_TIME
.format,1 = "The current CRISP time     : #sssssssssssssssssssssss "

.print,2  = #rpt32:float1
.format,2 = "The value of rpt32:float1  : #ffff.ff "

.end
```

Notes:

```
! PAGE.RPT
!-------------------------------------------------------------------------
! The following report configuration file will generate a report
! that looks as follows :
!-------------------------------------------------------------------------
!
!page 1
!14-NOV-1991 15:09:25.04
!
!               DATE         AN(0)    A        B        C      STR   astr(0)
! -------------------------- -----  -------- -------- -------- ----- -------
!   Thu Nov 14 15:09:27 1991   777     11.11    22.22    33.33 newst astr(0)
!   Thu Nov 14 15:09:30 1991   777     11.11    22.22    33.33 newst astr(0)
!   Thu Nov 14 15:09:33 1991   777     11.11    22.22    33.33 newst astr(0)
!   Thu Nov 14 15:09:36 1991   777     11.11    22.22    33.33 newst astr(0)
!   Thu Nov 14 15:09:39 1991   777     11.11    22.22    33.33 newst astr(0)
!           An(0)  FLOAT1   FLOAT2   FLOAT3
!TOTAL   :  3885     55.55   111.10   166.65
!AVERAGE :   777     11.11    22.22    33.33
!MIN     :   777     11.11    22.22    33.33
!MAX     :   777     11.11    22.22    33.33
!COUNT   :     5      5.00     5.00     5.00
!BAD CNT :     0      0.00     0.00     0.00
!Total writes : 5
!
!----------DONE------------- page 1
!
!
```

```
                          page 2
!14-NOV-1991 15:09:26.04
!
!               DATE          AN(0)    A        B        C      STR   astr(0)
! ------------------------- -----  -------- -------- -------- ----- -------
!   Thu Nov 14 15:09:27 1991    12     44.44    55.55    66.66 strin astr(1)
!   Thu Nov 14 15:09:30 1991    12     44.44    55.55    66.66 strin astr(1)
!   Thu Nov 14 15:09:33 1991    12     44.44    55.55    66.66 strin astr(1)
!   Thu Nov 14 15:09:36 1991    12     44.44    55.55    66.66 strin astr(1)
!   Thu Nov 14 15:09:39 1991    12     44.44    55.55    66.66 strin astr(1)
!        An(1)  FLOAT4   FLOAT5   FLOAT6
!TO      60    222.20   277.75   333.30
!AV      12     44.44    55.55    66.66
!MN      12     44.44    55.55    66.66
!astr(0) 12     44.44    55.55    66.66
!CN       5      5.00     5.00     5.00
!BD       0      0.00     0.00     0.00
!Total writes : 5
!
!----------DONE------------- page 2
!
!
```

```
                          page 3
!14-NOV-1991 15:09:26.04
!
!               DATE        AN(2)   FLOAT7   FLOAT8   FLOAT9   STR3  astr(2)
! -------------------------- -----  -------- -------- -------- ----- -------
!   Thu Nov 14 15:09:27 1991    13     77.77    88.88    99.99 strin astr(2)
!   Thu Nov 14 15:09:30 1991    13     77.77    88.88    99.99 strin astr(2)
!   Thu Nov 14 15:09:33 1991    13     77.77    88.88    99.99 strin astr(2)
!   Thu Nov 14 15:09:36 1991    13     77.77    88.88    99.99 strin astr(2)
!   Thu Nov 14 15:09:39 1991    13     77.77    88.88    99.99 strin astr(2)
!           An(2)  FLOAT7   FLOAT8   FLOAT9
!TO          65    388.85   444.40   499.95
!AV          13     77.77    88.88    99.99
!MN          13     77.77    88.88    99.99
!MX          13     77.77    88.88    99.99
!CN           5      5.00     5.00     5.00
!BD           0      0.00     0.00     0.00
!Total writes : 5
!
!----------DONE------------- page 3
!
!
!
!
!-------------------------------------------------------------------------
! This is an time based report and creates a file named PAGE.TMP.
! The report uses the .PAGE keyword to cause data to be printed on multiple
! pages (i.e. in this case pages 1,2 and 3).  Each page will have a header
! and a footer as described in the .PAGE section for that page of the
! report.  Note that averaging and statistics are also calculated for
! for some of the variables in the report.  We will write 5 records
! (i.e. .WRITE_PER_FILE = 5) every 3 seconds with samples taken every second.
! The output is written to a file named PAGE.TMP.
!-------------------------------------------------------------------------

! ---------------------------------------
! the following section describes
! the control information for the report
! ---------------------------------------

.FILE_NAME = "page.tmp"      ! the output file
.WRITE_PERIOD = 3 SEC           ! write every 3 seconds
.FORMAT_MODE = LB_FMT           ! use #sss format mode
.SECONDS_PER_SAMPLE = 1 SEC     ! sample every second
.WRITE_PER_FILE = 5             ! write 35 lines to the report
.SAMPLE_WHILE_TRUE =  #RPT32:swt  ! the sample_while_true bit
.FILE_OPEN_STATUS =   #RPT32:fos  ! the file_open_status_bit
.RECOVER_CRISP_STOP             ! recover this report if CRISP stops
.BAD_DATA_VALUE = -9999      ! if a value is -9999 then ignore it for average and stats

! .QUEUE = "SYS$PRINT" /FORM_NAME = "10C6L"
```

```
! ------------------------------------------------------------
! this section describes the look and contents of the report
! ------------------------------------------------------------


!-------------------------------------------------------------------------------
.page = 1
.header = "page 1"
.header = #CRISP:csp_s_time
.HEADER =   "               DATE       AN(0)    A        B        C      STR   astr(0)"
.HEADER = " ------------------------- -----  -------- -------- -------- ----- -------"
.format,1=" #SSSSSSSSSSSSSSSSSSSSSSSS #DDDD  #FFFF.FF #FFFF.FF #FFFF.FF #SSSS #ssssss"


.print,1 =                DATE,\
      #RPT32:AN(0)/STATS/AVERAGE,\
      #RPT32:FLOAT1/STATS/AVERAGE,\
      #RPT32:FLOAT2/STATS/AVERAGE,\
      #RPT32:FLOAT3/STATS/AVERAGE,\
      #RPT32:STR1,\
      #RPT32:ASTR(0)

.STATS_HEADER = " An(0)  FLOAT1   FLOAT2   FLOAT3 "
.STATS_FORMAT = "#DDDD  #FFFF.FF #FFFF.FF #FFFF.FF "
!.STATS_LABEL =           "AV ",\
!     "TO ",\
!     "MX ",\
!     "MN ",\
!     "CN ",\
!     "BD "
.footer = "\n----------DONE------------ page 1\n"

!-------------------------------------------------------------------------------
.page = 2
.header = "\fpage 2"
.header = #CRISP:csp_s_time
.HEADER =   "               DATE       AN(0)    A        B        C      STR   astr(0)"
.HEADER = " ------------------------- -----  -------- -------- -------- ----- -------"
.format,2=" #SSSSSSSSSSSSSSSSSSSSSSSS #DDDD  #FFFF.FF #FFFF.FF #FFFF.FF #SSSS #ssssss"


.print,2 =                DATE,\
      #RPT32:AN(1)/STATS/AVERAGE,\
      #RPT32:FLOAT4/STATS/AVERAGE,\
      #RPT32:FLOAT5/STATS/AVERAGE,\
      #RPT32:FLOAT6/STATS/AVERAGE,\
      #RPT32:STR2,\
      #RPT32:ASTR(1)
```

```
.STATS_HEADER = " An(1)  FLOAT4   FLOAT5   FLOAT6 "
.STATS_FORMAT = "#DDDD  #FFFF.FF #FFFF.FF #FFFF.FF "
.STATS_LABEL =             "AV ",\
      "TO ",\
      #rpt32:astr(0),\
      "MN ",\
      "CN ",\
      "BD "
.footer = "\n----------DONE------------- page 2\n"

!-------------------------------------------------------------------------------
.page = 3
.header = "\fpage 3"
.header = #CRISP:csp_s_time
.HEADER = "               DATE         AN(2)  FLOAT7   FLOAT8   FLOAT9   STR3  astr(2)"
.HEADER = " ------------------------- -----  -------- -------- -------- ----- -------"
.format,3=" #SSSSSSSSSSSSSSSSSSSSSSS #DDDD  #FFFF.FF #FFFF.FF #FFFF.FF #SSSS #ssssss
                        "


.print,3 =                 DATE,\
      #RPT32:AN(2)/STATS/AVERAGE,\
      #RPT32:FLOAT7/STATS/AVERAGE,\
      #RPT32:FLOAT8/STATS/AVERAGE,\
      #RPT32:FLOAT9/STATS/AVERAGE,\
      #RPT32:STR3,\
      #RPT32:ASTR(2)

.STATS_HEADER = " An(2)  FLOAT7   FLOAT8   FLOAT9 "
.STATS_FORMAT = "#DDDD  #FFFF.FF #FFFF.FF #FFFF.FF "
.STATS_LABEL =             "AV ",\
      "TO ",\
      "MX ",\
      "MN ",\
      "CN ",\
      "BD "
.footer = "\n----------DONE------------- page 3\n"

.END
```

Notes:

```
!  SAMPLE CRISP/32 FILE CONFIGURED FOR SAMPLE REPORT FILES RPT32.C32

INTERMEDIATE;             VFLAG:TRUE, RSFLG:TRUE, ACTIVE, CACDSA, \
                          INDONE    , PRTFLG    , FCNBIT, CONBIT  , \
                          FLASH     , OVFLG     , IOTEST, ICCDSA  , \
                          PAUSE     , COMDSA    ,       ,
!
LOGICAL;                  SWT,FOS,WNOW,OPEN_INPUT_FILE,INPUT_FILE_OPEN,READ_IF,\
                          CLOSE_INPUT_FILE

LOGICAL;                  PRINT_IF_1
LOGICAL;                  PRINT_IF_2
LOGICAL;                  PRINT_IF_3
LOGICAL;                  PRINT_IF_4

LOGICAL;                  RUN_RPT_BIT

NUMERIC;                  ROW_COUNT:1
NUMERIC;                  FIRST_ROW:0

STRING;                   RPT_UCF_FILENAME[]:"MULTI1"


LONG;                     W_PERIOD:10
LONG;                     SPS:1
LONG;                     WPF:5

FLOAT;                    A,B,C

FLOAT;                    A_AVG_STAT,A_TOT_STAT,A_MAX_STAT,A_MIN_STAT,A_CNT_STAT, \
                          A_BAD_STAT

FLOAT;                    A_AVG

FLOAT;                    B_AVG(50)

FLOAT;                    FLOAT1:11.11
FLOAT;                    FLOAT2:22.22
FLOAT;                    FLOAT3:33.33
FLOAT;                    FLOAT4:44.44
FLOAT;                    FLOAT5:55.55
FLOAT;                    FLOAT6:66.66
FLOAT;                    FLOAT7:77.77
FLOAT;                    FLOAT8:88.88
FLOAT;                    FLOAT9:99.99
FLOAT;                    FLOAT10:100.101
```

```
LONG;                   LONG1:111
LONG;                   LONG2:222
LONG;                   LONG3:333
LONG;                   LONG4:444
LONG;                   LONG5:555
LONG;                   LONG6:666
LONG;                   LONG7:777
LONG;                   LONG8:888
LONG;                   LONG9:999
LONG;                   LONG10:101010

NUMERIC;                AN(10):11:12:13:14:15

STRING;                 STR1[40]:"STRING 1"
STRING;                 STR2[40]:"STRING 2"
STRING;                 STR3[40]:"STRING 3"
STRING;                 STR4[40]:"STRING 4"
STRING;                 STR5[40]:"STRING 5"
STRING;                 STR6[40]:"STRING 6"
STRING;                 STR7[40]:"STRING 7"
STRING;                 STR8[40]:"STRING 8"
STRING;                 STR9[40]:"STRING 9"
STRING;                 STR10[40]:"STRING 10"

STRING;                 ASTR(10)[]:"ASTR(0)":"ASTR(1)":"ASTR(2)":"ASTR(3)"

TABLES;

RESTART;

END;
```

```
! CHART DISPLAY FILE for SAMPLE Report file Variables RPT32.DSP

.CHART
.<ESC>[3;0H
.<ESC>[?3N
              RPT32.DSP           A                                  B_AVG
SWT {10}  WNOW        {11}      AVG  {201}      A {301}        (0)  {221}
FOS {12}  RUN_RPT_BIT {13}      TOT  {202}      B {302}        (1)  {222}
SPS {15}  W_PERIOD    {16}      MAX  {203}      C {303}        (2)  {223}
WPF {17}                        MIN  {204}                     (3)  {224}
RPT_UCF_FILENAME {14}           CNT  {205}                     (4)  {225}
A_AVG {211}                     BAD  {206}                     (5)  {226}
OIF   {990} IFO  {991}  READ_IF {992}   ROW_CNT {993}          (6)  {227}
FLOAT1  {31}     LONG1  {91}     STR1 {51}      AN(0) {71}     (7)  {228}
FLOAT2  {32}     LONG2  {92}     STR2 {52}      AN(1) {72}     (8)  {229}
FLOAT3  {33}     LONG3  {93}     STR3 {53}      AN(2) {73}     (9)  {230}
FLOAT4  {34}     LONG4  {94}     STR4 {54}      AN(3) {74}     (10) {231}
FLOAT5  {35}     LONG5  {95}     STR5 {55}                     (11) {232}
FLOAT6  {36}     LONG6  {96}     ASTR(0) {61}                  (12) {233}
FLOAT7  {37}     LONG7  {97}     ASTR(1) {62}                  (13) {234}
FLOAT8  {38}     LONG8  {98}     ASTR(2) {63}                  (14) {235}
FLOAT9  {39}     LONG9  {99}     ASTR(3) {64}                  (15) {236}
PRINT_IF_1  {81}
PRINT_IF_2  {82}
PRINT_IF_3  {83}
PRINT_IF_4  {84}


.AUTO_LINK
.FALSE= OFF
.TRUE= ON
.DATA_BASE= RPT32
```

```
.SYMBOL

10 /0:SWT
11 /0:WNOW
12 /0:FOS
13 /0:RUN_RPT_BIT
14 /0:RPT_UCF_FILENAME
15 /0:SPS
16 /0:W_PERIOD
17 /0:WPF

31 /0:FLOAT1
32 /0:FLOAT2
33 /0:FLOAT3
34 /0:FLOAT4
35 /0:FLOAT5
36 /0:FLOAT6
37 /0:FLOAT7
38 /0:FLOAT8
39 /0:FLOAT9

51 /0:STR1
52 /0:STR2
53 /0:STR3
54 /0:STR4
55 /0:STR5

61 /0:ASTR(0)
62 /0:ASTR(1)
63 /0:ASTR(2)
64 /0:ASTR(3)

71 /0:AN(0)
72 /0:AN(1)
73 /0:AN(2)
74 /0:AN(3)

81 /0:PRINT_IF_1
82 /0:PRINT_IF_2
83 /0:PRINT_IF_3
84 /0:PRINT_IF_4

91 /0:LONG1
92 /0:LONG2
93 /0:LONG3
94 /0:LONG4
95 /0:LONG5
96 /0:LONG6
97 /0:LONG7
98 /0:LONG8
99 /0:LONG9
```

```
201 /0:A_AVG_STAT
202 /0:A_TOT_STAT
203 /0:A_MAX_STAT
204 /0:A_MIN_STAT
205 /0:A_CNT_STAT
206 /0:A_BAD_STAT

211 /0:A_AVG

221 /0:B_AVG(0)
222 /0:B_AVG(1)
223 /0:B_AVG(2)
224 /0:B_AVG(3)
225 /0:B_AVG(4)
226 /0:B_AVG(5)
227 /0:B_AVG(6)
228 /0:B_AVG(7)
229 /0:B_AVG(8)
230 /0:B_AVG(9)
231 /0:B_AVG(10)
232 /0:B_AVG(11)
233 /0:B_AVG(12)
234 /0:B_AVG(13)
235 /0:B_AVG(14)
236 /0:B_AVG(15)

301 /0:A
302 /0:B
303 /0:C

990 /0:OPEN_INPUT_FILE
991 /0:INPUT_FILE_OPEN
992 /0:READ_IF
993 /0:ROW_COUNT
```

Notes:

```
! SPS.RPT
! Sample Output File
!
!14-NOV-1991 15:14:07.05
!
!               DATE        long1  AN(0)     A          astr(0)       swt    fos
!------------------------- -----  --------  -------   -----------   -----  -----
!  Thu Nov 14 15:14:17 1991   123    777       779.30       astr(0)      1      1
!  Thu Nov 14 15:14:27 1991   123    777       779.30       astr(0)      1      1
!  Thu Nov 14 15:14:37 1991   123    777       779.30       astr(0)      1      1
!  Thu Nov 14 15:14:47 1991   123    777       779.30       astr(0)      1      1
!  Thu Nov 14 15:14:57 1991   123    777       779.30       astr(0)      1      1
!
!----------DONE-------------
!
! This file is a TIME based report.
!
! Example of using CRISP VARIABLES for SECONDS_PER_SAMPLE, WRITE_PERIOD,
! WRITE_PER_FILE directives. The output is written to a file named SPS.TMP.
!

.DATA_BASE = RPT32
.FILE_NAME = "SPS.TMP"
.WRITE_PERIOD = #RPT32:W_PERIOD
.FORMAT_MODE = LB_FMT

.SECONDS_PER_SAMPLE = #RPT32:SPS
.WRITE_PER_FILE = #RPT32:WPF

.SAMPLE_WHILE_TRUE =  #rpt32:swt
.FILE_OPEN_STATUS =   #rpt32:fos
.BAD_DATA_VALUE = -9999

.header = #CRISP:csp_s_time

.HEADER =   "              DATE        long1  AN(0)     A          astr(0)       swt    fos  "
.HEADER =   "------------------------- -----  --------  -------   -----------   -----  -----"
.format,1 = "#SSSSSSSSSSSSSSSSSSSSSSSS #dddd  #DDDD     #FFFF.FF  #sssssssssss  #dddd  #dddd"



.print,1 =                 DATE,\
      #rpt32:long1,\
      #rpt32:an(0),\
      #rpt32:A,\
      #rpt32:ASTR(0),\
      #rpt32:swt,\
      #rpt32:fos

.footer = "\n----------DONE-------------\n"

.SYMBOL
```

Notes:

```
! TEST.RPT
.DATA_BASE = RPT32
.FILE_NAME = "test.tmp"
.WRITE_PERIOD = #RPT32:W_PERIOD
.FORMAT_MODE = LB_FMT

.SECONDS_PER_SAMPLE = #RPT32:SPS
.WRITE_PER_FILE = #RPT32:WPF

.SAMPLE_WHILE_TRUE =  #rpt32:swt
.FILE_OPEN_STATUS =   #rpt32:fos
.BAD_DATA_VALUE = -9999

.header = #CRISP:csp_s_time

.HEADER =   "              DATE        long1  AN(0)     A          astr(0)       swt    fos  "
.HEADER =   "------------------------- -----  --------  -------   -----------   -----  -----"
.format,1 = "#SSSSSSSSSSSSSSSSSSSSSSSS #dddd  #DDDD     #FFFF.FF  #sssssssssss  #dddd  #dddd"


.print,1 =                  DATE,\
      #rpt32:long1,\
      #rpt32:an(0),\
      #rpt32:A,\
      #rpt32:ASTR(0),\
      #rpt32:swt,\
      #rpt32:fos

.footer = "\n----------DONE-------------\n"

.SYMBOL
```

Notes:

```
! TIME.RPT
!-------------------------------------------------------------------------
! The following report configuration file will generate a report
! that looks as follows :
!-------------------------------------------------------------------------
!
!                 TIME BASED REPORT
!
!  Report started at : Mon Sep 16 10:58:12 1991
!  ---------------------------------------------
!
!  10:58:13    11.110   111   string 1 text string
!  10:58:13    11.110   111   string 1 text string
!  10:58:14    11.110   111   string 1 text string
!  10:58:15    11.110   111   string 1 text string
!  ---------------------------------------------
!  Report completed at : 10:58:15
!
!
!
!-------------------------------------------------------------------------
! This is a TIME based report and creates a file named TIME.TMP.
! The header consists of three lines and is printed when the
! RPT32:SAMPLE_WHILE_TRUE CRISP logical variable first goes true.
! Each of the lines (i.e. .PRINT,1 = ...) are printed every 6 seconds
! until 4 lines (i.e. .WRITE_PER_FILE = 4) have been printed.  After
! the 4 lines are printed the footer is written to TIME.TMP and the report
! is complete. This whole process is repeated as long as
! the CRISP logical variable RPT32:SAMPLE_WHILE_TRUE remains TRUE.
! Note that the .SECONDS_PER_SAMPLE = 2 SEC statement controls how often
! the CRISP logical variable RPT32:SAMPLE_WHILE_TRUE is checked and also
! the rate at which samples are read if we had specified averaging for
! some of the variables. The output is written to a file named TIME.RPT.
!
!-------------------------------------------------------------------------

! --------------------------------------
! the following section describes
! the control information for the report
! --------------------------------------

.FILE_NAME          = "time.TMP" ! write report to file TIME.TMP
.WRITE_PERIOD       = 6 SEC      ! write a line every 6 seconds
.SECONDS_PER_SAMPLE = 2 SEC      ! check .sample_while_true every 2 sec
.WRITE_PER_FILE     = 4          ! write 4 lines to the report
.SAMPLE_WHILE_TRUE  = #rpt32:swt ! the sample while true bit
.FILE_OPEN_STATUS   = #rpt32:fos ! the file open status bit

.FORMAT_MODE        = LB_FMT     ! use the #sss format mode

.BAD_DATA_VALUE     = -9999      ! if a value = -9999 then it is
                                 ! ignored for averaging and statistics
```

```
! ---------------------------------------------------------
! this section describes the look and contents of the report
! ---------------------------------------------------------

.HEADER   = "               TIME BASED REPORT\n"        ! 3 lines of header
.HEADER   = "Report started at : ",DATE
.HEADER   = "-----------------------------------------------"
.FORMAT,1 = "#sssssss #ffff.fff #dddd #sssssssss #sssssssss" ! format for the print
                         statement
.FOOTER   = "-----------------------------------------------" ! 2 lines of footer
.FOOTER   = "Report completed at : ",time

.PRINT,1  =  time,\                          ! 5 values for the .FORMAT
        #rpt32:float1,\
        #rpt32:long1,\
        #rpt32:str1,\
        "text string"

.END                                ! end of the report
```

RPT V2.1 December 1993

- The CRISP/32 REPORTER V2.1 has been enhanced to support the following.

  - CRISP Symbol Aliasing

  - Variable Subscript Support

  This allows alias names for CRISP variables and alias names for CRISP array variables with variable subscripts to be accessed for reporting. The specific CRISP variable to access is resolved once during symbol resolution.

  Although this version of the REPORTER can run on CRISP/32 version 2.8 systems, CRISP/32 version 2.8 does not support alias names for CRISP variables. A user that wants to use alias names in the REPORTER must upgrade to CRISP/32 version 3.0.

- The "One Process Multiple Reports" report type has been fixed so that it does not run out of process AST quotes after running for a while.

RPT V2.0 September 1991

This version of the REPORTER has many new features to make its use more flexible and suited for a larger spectrum of applications. Some notable enhancements are as follows.

- EDL is no longer required.
- The .SYMBOL section is no longer required.
- New format mode (using # signs as field substitution characters).
- One time Ad-Hoc reports.
- Time of day sampling and closing of report files.
- Run different Reports with one REPORTER process.
- Functions on .PRINT directives.
- Multiple pages.
- Output file name can be composed from multiple CRISP data base variables and text.
- ASCII text files can be opened and contents move to CRISP data base variables.
- Basic Math functions.
- Output to System Print Queues with FORM and DELETE specifications.
- Write Report Data back to CRISP Real time database.
- Conditional printing of .PRINT directives.
- Recover a report between CRISP stop and start.
- Support for arrays
- LOG_IF directive for event logging
- Enable combined CHART/II and report file
- Added minute, hour, month, day, year and dow

The enhancements are compatible with previous versions of the REPORTER and should not affect the operation of existing applications. The period (.) delimiter between the file name and extension should be specified.

RPT V1.7 October 1990

This version of the REPORTER includes improvements to reduce CPU loading when a large number of data points are used in a report without averaging. This release of RPT is compatible with version 2.6 and 2.7 of CRISP and does not include support for CRISP 2.7 arrays.