OCSORT 1.0

[15 GEN 2015 Version]

User's Guide

1nd Edition, 15 Janury 2016

Sauro Menna
mennasauro@gmail.com

This work is dedicated to the memory of my niece Federica,
a strong young woman, sweet and resourceful.
You will always be in my heart and mind.

Summary of Changes

Edition	Date	Change Description
1st	15 Jan 2016	INITIAL RELEASE OF DOCUMENT

Table of Contents

1. Introduction

1.1. What is OCSort?

This document describes the features of the OCSORT utility.
OCSORT is an open-source tool for operations of sort/merge/copy files (Line Sequential, Sequential) produced by GNUCobol compiler.
The principal developers of OCSORT are Cedric Issaly and Sauro Menna.
This document was intended to serve as a full-function reference and user's guide for OCSORT utility.

2. Features

Version 1.0.0 of OCSort contains a follow constructs:

```
SORT/MERGE
      FIELDS (START,LENGTH,TYPE,[A|D], ...)           |
      FIELDS ((START,LENGTH,[A|D], ...),FORMAT=TYPE   |
      FIELDS=COPY

USE  / GIVE ORG [LS|SQ] RECORD [[F,Record Len] | [V,MIN,MAX Record Len]]
INCLUDE|OMIT COND=(CONDITION) [FORMAT=TYPE]
INREC FIELDS=(FIELD-SPEC...)
INREC BUILD=(FIELD-SPEC...)
SUM FIELDS = (pos,len,type, ...)
SUM FIELDS = (NONE)  or   SUM FIELDS = NONE

OUTREC FIELDS=(FIELD-SPEC...)
OUTREC BUILD=(FIELD-SPEC...)
OUTFIL
      FILES/FNAMES= (environment variable)
      STARTREC=nn
      ENDREC=nn
      [SAVE|[INCLUDE|OMIT] (CONDITION) [FORMAT=TYPE]]
      SPLIT
      OUTREC = (FIELD-SPEC...)
OPTION
      SKIPREC=nn        Skip nn records from input
      STOPAFT=nn        Stop read after nn records
      VLSCMP            0 disabled , 1 = enabled -- replace with binary zeros
                          any missing compare field bytes
      VLSHRT            0 disabled , 1 = enabled -- treat any comparison
                            involving a short field as false
      EQUALS           Command skipped
      REMOVECC         Command skipped
      CONVERT          Command skipped
      NODETAIL         Command skipped
```

3. Environment and first use

OCSort is a executable program written in 'C'.

Dependencies of executable OCSort are:

- **libcob** - GNUCobol
- **mpir** / **libgmp** - GNU MP

3.1. Following the steps for the first use

- Make executable OCSort
- Set environment variable to find library at runtime
- Run *ocsort <command line>*

3.2. Modify first environment variables

- Set ByteOrder (OCSORT_BYTEORDER) 0 = Native, 1 BigEndian
- Set Memory Allocation (OCSORT_MEMSIZE)
- Set Statistics (OCSORT_STATISTICS) to view details of execution

3.3. Use TAKE command

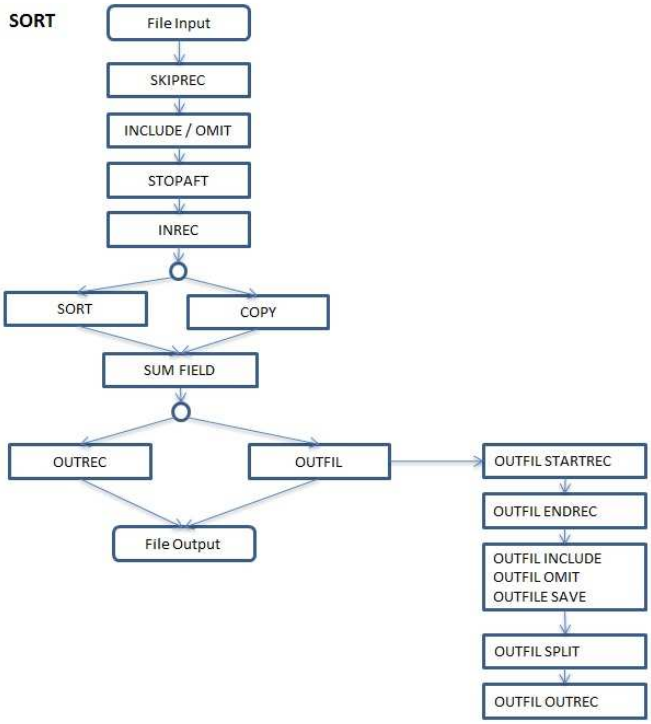
- Create file text
- Insert command. Single row o one row for command.
- In the file TAKE the '*' character is first column is a comment
- Run : *ocsort* TAKE *filename*

Example to create TAKE file with script sh.

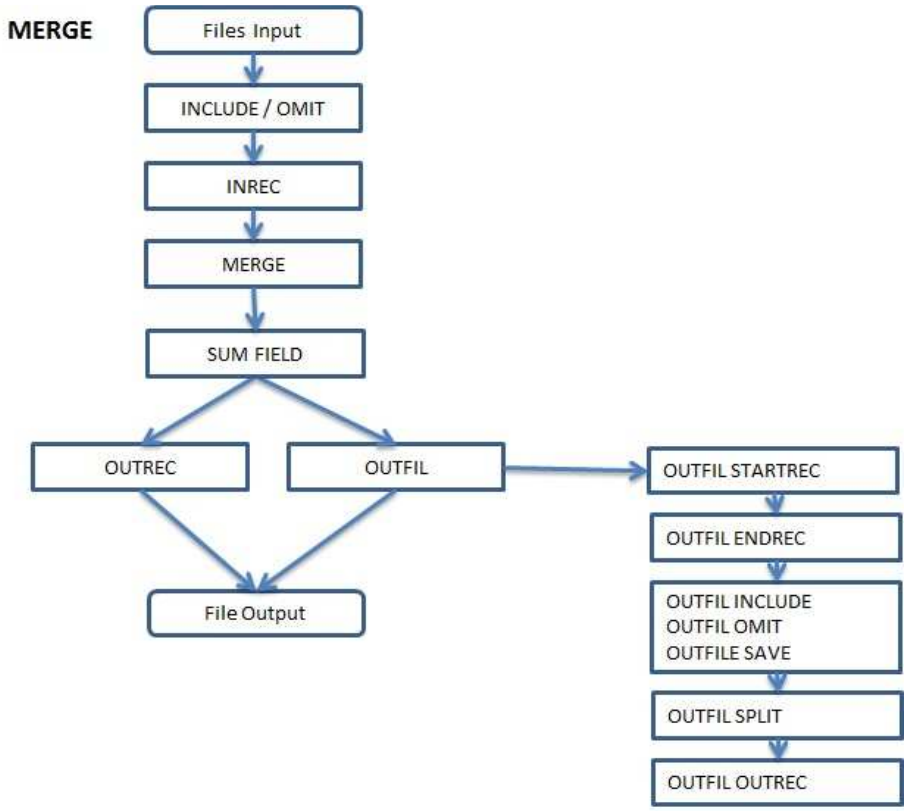
```
export LD_LIBRARY_PATH=/usr/local/lib
export OCSORT_MEMSIZE=1024000000
export OCSORT_BYTEORDER=0
export OCSORT_STATISTICS=2
echo "SORT FIELDS(4,1,CH,A) "                          >TAKEFILE.PRM
echo "SUM  FIELDS=(1,2,ZD,4,2,ZD,7,4,ZD,12,4,ZD)    " >>TAKEFILE.PRM
echo "USE  ../files/SQZD03 RECORD F,396 ORG SQ      " >>TAKEFILE.PRM
echo "GIVE ../files/SQZD03.SRT  RECORD F,396 ORG SQ " >>TAKEFILE.PRM
../bin/ocsort TAKE  TAKEFILE.PRM
```

4. Process Schema

This picture show logical schema of utility OCSort for SORT operations.



This picture show logical schema of utility OCSort for MERGE operations.



5. Sort

The purpose of SORT is read one or more files and create a output file with data ordered as indicated by the sort key fields.

6. Merge

The purpose of MERGE is read one or more files and create a output file with data ordered as indicated by the merge key fields.
It is mandatory that the input data is already sorted.

7. File Organization and Record Type

File organization identifies the type of file.
The types of file organization utility managed OCSORT are:

LS = Line Sequential
SQ = Sequential

Record type identifies the record structure
Record type are

F = Fixed
V = Variable (first 4 byte record len)

8. Field Type

Field type detects typology of field, Field type used are:

CH = Char
BI = Binary unsigned
FI = Binary signed
PD = Packed
ZD = Zoned max 18 digits

9. Command

9.1. SORT

SORT is command for ordering data.

Format 1 **SORT**

9.2.MERGE

MERGE is command for merging data.

Format 1 **MERGE**

9.3.FIELDS

This command specify fields for sort/merge operations. The fields are the key for order or merging data from files.

Format 1 FIELDS (pos,len,type,order, ...) |
Format 2 FIELDS ((pos,len, order, ...),FORMAT=TYPE |
Format 3 FIELDS=COPY

FIELDS (pos, len, type, order,….)

pos specifies the first byte of a control field relative to the beginning of the input record.
The first data byte of a fixed-length record has relative position 1.
The first data byte of a variable-length record has relative position 1.

len specifies the length of the field. Values for all fields must be expressed in integer numbers of bytes.

type specifies the format of the data of field.

Format	Description
CH	Characters
ZD	Signed zoned decimal
PD	Signed packed decimal
BI	Unsigned binary
FI	Signed fixed-point

order specifies how the field is to be ordered. The valid codes are:
A ascending order
D descending order

FIELDS ((pos,len,order, ...),FORMAT=type

FORMAT=type can be used to specify a particular format for one or more control fields. f from FORMAT=f is used for p,m,s fields.

FIELDS=COPY or FIELDS=(COPY)

Causes OCSORT to copy a file input to the output data sets. Records can be edited INCLUDE/OMIT, INREC, OUTREC, and OUTFIL statements; and SKIPREC and STOPAFT parameters.

9.4.USE

USE command declare input file for SORT and MERGE operation.

Format for USE:
USE <filename > ORG <organization> RECORD [<record format>,< length>]
USE <filename > ORG <organization> RECORD [<record format>, <lenght min>,< length max>]

filename Input file name, with or without pathname

organization LS = Line Sequential
SQ = Sequential

record format F = Fixed
V = Variable

length Length of record

length min Minimun length of record

length max Maximum length of record

9.5.GIVE

GIVE command declare output file for SORT and MERGE operation.

Format for USE:

Fixed GIVE <filename > ORG <organization> RECORD [<record format>,< length>]
Variable GIVE <filename > ORG <organization> RECORD [<record format>, <length min>,< length max>]

filename Input file name, with or without pathname
organization LS = Line Sequential
SQ = Sequential

record format F = Fixed
V = Variable

length Length of record

length min Minimun length of record

length max Maximum length of record

9.6.INCLUDE/OMIT

INCLUDE condition statement is used for ***select*** records to insert in the file output.
OMIT condition statement is used for ***exclude*** certain records from the file input.

INCLUDE/OMIT COND=(condition) [FORMAT=type]

condition
Format 1 (pos , len , type , cond, pos , len , type)
Format 2 (pos , len , type , cond, [X|C|Z]'[value]')
Format 3 (condition , relcond , condition)

Format 1 (pos , len , type , cond, relcond , pos , len , type)

pos specifies the first byte of a control field relative to the beginning of the input record.
The first data byte of a fixed-length record has relative position 1.
The first data byte of a variable-length record has relative position 1.

len specifies the length of the field. Values for all fields must be expressed in integer numbers of bytes.

type specifies the format of the data of field.

Format	Description
CH	**Characters**
ZD	**Signed zoned decimal**
PD	**Signed packed decimal**
BI	**Unsigned binary**
FI	**Signed fixed-point**

cond Comparison operators are as follows:

EQ Equal to
NE Not equal to
GT Greater than
GE Greater than or equal to
LT Less than
LE Less than or equal to.

Format 2 (pos , len , type , cond, [X|C]'[value]')|[+/-nnnn]

pos specifies the first byte of a control field relative to the beginning of the input record.
The first data byte of a fixed-length record has relative position 1.
The first data byte of a variable-length record has relative position 1.

len specifies the length of the field. Values for all fields must be expressed in integer numbers of bytes.

type specifies the format of the data of field.

Format	Description
CH	**Characters**
ZD	**Signed zoned decimal**
PD	**Signed packed decimal**
BI	**Unsigned binary**
FI	**Signed fixed-point**

cond Comparison operators are as follows:

EQ Equal to
NE Not equal to
GT Greater than
GE Greater than or equal to
LT Less than
LE Less than or equal to.

C'cc…c' **Character String Format .** The value c is a ASCII character/string.

X'hh..hh' **Hexadecimal String Format.** The value hh represents any pair of hexadecimal digits.

+/- nnnn.. **Decimal Number Format**

Format 3 (condition , relcond , condition)

condition Format 1 or Format 2

relcond Relational conditions can be logically combined, with AND or OR.
The relational condition specifies that a comparison test be performed.
Relational conditions can be logically combined, with AND or OR.

9.7.INREC/OUTREC

INREC redefines the structure of record input. This operation is executed after read file input e before all operations.
The INREC control statement reformat the input records **before** they are sorted, merged, or copied.
All fields specifications presents in OUTREC, Sort Key, … must be referred to a new structure defined by INREC.

Format 1 INREC FIELDS=(FIELD-SPEC...)
Format 2 INREC BUILD=(FIELD-SPEC...)

OUTREC defines structure record output for output file.
Format 1 OUTREC FIELDS=(FIELD-SPEC...)
Format 2 OUTREC BUILD=(FIELD-SPEC...)

Field specification is the same for INREC and OUTREC.

BUILD or FIELDS are synonymous.

FIELD-SPEC (pos, len | posOut:pos,len | n:X | n:Z |C'constant' | nX | nZ, |X'hh')
One or more occurrence of follow elements, separated by comma.

pos, len	**pos** = position input record, **len** = length of field
posOut:pos,len	**posOut** = position output, **pos** = position input record, **len** = length of field
n:X	Filling with Blank character (0x20) from last position to **n** (absolute position of output record).
n:Z	Filling with zero Binary (0x00) character from last position to **n** (absolute position of output record).
C'constant'	constant character value.
nC'constant'	repeat **n** times constant character value.
nX	repeat **n** times Blank character.
nZ	repeat **n** times Binary (0x00) character.
X'hh…hh'	hexdecimal string .
nX'hh…hh'	repeat **n** times hexdecimal string .

9.8.SUM FIELDS

SUM FIELDS is command for aggregate record and summarize value for numeric fields.
All fields present in SUM FIELDS are aggregate when more records has same key.

Format 1 SUM FIELDS = (pos,len,type, ...)
Format 2 SUM FIELDS = (NONE) or SUM FIELDS = NONE

There are two formats for SUM FIELD, the first summarize numeric fields, the send NOT summarize, but eliminate duplicate key.

Format 1 SUM FIELDS = (pos,len,type, ...)

pos specifies the first byte of a control field relative to the beginning of the input record.
The first data byte of a fixed-length record has relative position 1.
The first data byte of a variable-length record has relative position 1.
len specifies the length of the field. Values for all fields must be expressed in integer numbers of bytes.
type specifies the format of the data of field.

Format	Description
ZD	**Signed zoned decimal**
PD	**Signed packed decimal**
BI	**Unsigned binary**
FI	**Signed fixed-point**

Format 2 SUM FIELDS = (NONE) or SUM FIELDS = NONE

In this case Format2 insert into output file one occurrence of same key specified by SORT KEY.
The record output contains the first record in order of reading.
For identify a first occurrence of data, OCSORT verified the value of pointer of record into file input, selecting the lowest value.

9.9.OUTFIL

OUTFIL is command to create one or more output file for a sort, copy, or merge operation.
Each file output is defined from OUTFIL command

FORMAT

OUTFIL

FILES/FNAMES= (environment variable)
STARTREC=nn
ENDREC=nn
[SAVE|[INCLUDE|OMIT] (CONDITION) [FORMAT=TYPE]]
SPLIT
OUTREC = (FIELD-SPEC...)

OUTFIL

FILES/FNAMES=filename	filename = Identify a environment variable the contain the file name
STARTREC=nn	Start write after **nn** records
ENDREC=nn	Stop write after **nn** records
SAVE	Save records that not used by command INCLUDE/OMIT.
INCLUDE/OMIT (CONDITION) [FORMAT=TYPE]]	Same definition for COND-FIELD (INCLUDE/OMIT)
SPLIT	Split record data for each OUTFIL definition
OUTREC = (FIELD-SPEC...)	Define structure output data. Same definition for (FIELD-SPEC...).

If the environment variable filename for FILES/FNAMES is not defined, OCSort writes output file in local folder assuming the name equal at value of identifier filename (FILES/FNAMES=*filename*).

9.10. OPTION

This command allows you to change the behavior of the utility.

Format1 **OPTION** [**SKIPREC**=nn]|[**STOPAFT**=nn]|[**VLSCMP**]|[**VLSHRT**]

SKIPREC=nn	Skip nn records from input
STOPAFT=nn	Stop read after nn records
VLSCMP	0 disabled , 1 = enabled -- temporarily replace any missing compare field bytes with binary zeros
VLSHRT	0 disabled , 1 = enabled -- treat any comparison involving a short field as false
EQUALS	Command skipped
REMOVECC	Command skipped
CONVERT	Command skipped
NODETAIL	Command skipped

The commands 'skipped' are recognized from OCSort, but not used.

10. Environment Variables

10.1. Byte Order

OCSort can treat numeric fields in both binary format BigEndian or Native. To indicate a byte order is used environment variable OCSORT_BYTEORDER that assume 0 for Native or 1 for BigEndian. This value affects the treatment of SORT and SUM KEY FIELDS.

10.2. Temporary Files

When dimension of files input is greater of memory available, OCSort creates temporary files for sort operation. Temporary files is created in pathname specified from OCSORT_TMPFILE environment variable, if this value is not available, OCSort use TMP/TEMP environment variable or use current directory. For Windows the filename is composed from:

- Prefix = Srt
- Name = name (created from GetTempFileName())
- Extension = .tmp

For Linux file name is composed from:

- Prefix = Srt
- Name = PID of process OCSort
- Num = Progressive of file
- Extension = .tmp

Temporary files are destroyed after sort operation.

10.3. Memory Allocation

The environment variable OCSORT_MEMSIZE specify amount of memory that OCSORT will use for sort operation.

OCSort analyze the value and made two area for sort operation:

(1) Key Area : this area is used for sort in memory
(2) Data Area : this area contains data record

The optimization for use of memory OCSort check dimension of key and record.

Key Area = [OCSORT_MEMSIZE] * ((Key Length + 8 + 4 + 8) / Record Length)

Data Area = [OCSORT_MEMSIZE] - Key Area

(8 + 4 + 8) 8 is pointer of record into file, 4 record length, 8 pointer to record area in memory.

If value of ((Key Length + 8 + 4 + 8)/ Record Length) is minor of 15% or major of 50%, OCSORT force this value to 15%.

10.4. Statistics

OCSort produce in output a lot of information about execution.

You can setting OCSORT_STATISTICS environment variable to three values:

0 = minimal information

Example:

```
=====================================================
OCSort Version 01.00.00
=====================================================
TAKE file name
D:\GNU_COBOL\OCSort_1_0_0\ocsort_testcase\take\par_SORT_debug.par
=====================================================
File : D:\OCSORTTEST\OCFILES\TEST9\INP000.txt
Size : 1194
=====================================================
Record Number Total       : 15
Record Write Sort Total   : 0
Record Write Output Total : 15
=====================================================
Start    : Mon Jan 25 11:17:55 2016
End      : Mon Jan 25 11:17:55 2016
Elapsed  Time 00hh 00mm 00ss 000ms
```

```
Sort OK
```

1 = medium information

Example

```
=========================================================
OCSORT
File TAKE : D:\GNU_COBOL\OCSort_1_0_0\ocsort_testcase\take\par_SORT_debug.par
=========================================================
SORT FIELDS(3,1,CH,A)
USE D:\OCSORTTEST\OCFILES\TEST9\INP000.txt ORG LS RECORD V,1,27990
GIVE D:\OCSORTTEST\OCFILES\TEST9\OUT000.SRT ORG LS RECORD V,1,27990

=========================================================
OCSort Version 01.00.00
=========================================================
TAKE file name
D:\GNU_COBOL\OCSort_1_0_0\ocsort_testcase\take\par_SORT_debug.par
=========================================================
Operation  : SORT

INPUT FILE :
        D:\OCSORTTEST\OCFILES\TEST9\INP000.txt VARIABLE (1,27990) LS
OUTPUT FILE :
        D:\OCSORTTEST\OCFILES\TEST9\OUT000.SRT VARIABLE (1,27990) LS
SORT FIELDS : (3,1,CH,A)
=========================================================
File : D:\OCSORTTEST\OCFILES\TEST9\INP000.txt
Size : 1194
=========================================================
Record Number Total       : 15
Record Write Sort Total   : 0
Record Write Output Total : 15
=========================================================
Start    : Mon Jan 25 11:20:01 2016
End      : Mon Jan 25 11:20:01 2016
Elapsed  Time 00hh 00mm 00ss 000ms

Sort OK
```

2 = details information

```
=========================================================
OCSORT
File TAKE : D:\GNU_COBOL\OCSort_1_0_0\ocsort_testcase\take\par_SORT_debug.par
=========================================================
SORT FIELDS(3,1,CH,A)
USE D:\OCSORTTEST\OCFILES\TEST9\INP000.txt ORG LS RECORD V,1,27990
GIVE D:\OCSORTTEST\OCFILES\TEST9\OUT000.SRT ORG LS RECORD V,1,27990

=========================================================
OCSort Version 01.00.00
=========================================================
TAKE file name
D:\GNU_COBOL\OCSort_1_0_0\ocsort_testcase\take\par_SORT_debug.par
=========================================================
Operation  : SORT

INPUT FILE :
        D:\OCSORTTEST\OCFILES\TEST9\INP000.txt VARIABLE (1,27990) LS
```

```
OUTPUT FILE :
        D:\OCSORTTEST\OCFILES\TEST9\OUT000.SRT VARIABLE (1,27990) LS
SORT FIELDS : (3,1,CH,A)
=====================================================
File : D:\OCSORTTEST\OCFILES\TEST9\INP000.txt
Size : 1194
After  job_loadFiles      - Mon Jan 25 11:21:44 2016
After  job_sort           - Mon Jan 25 11:21:44 2016
After  job_save           - Mon Jan 25 11:21:44 2016
=====================================================
Record Number Total       : 15
Record Write Sort Total   : 0
Record Write Output Total : 15
=====================================================

Memory size for OCSort data     :  133875000
Memory size for OCSort key      :   23625000
BufferedReader MAX_BUFFER       :    4063232
MAX_SIZE_CACHE_WRITE            :    4063232
MAX_SIZE_CACHE_WRITE_FINAL      :    4063232
MAX_MLTP_BYTE                   :         63
BYTEORDER                       :          0
=============================================

Start    : Mon Jan 25 11:21:44 2016
End      : Mon Jan 25 11:21:44 2016
Elapsed  Time 00hh 00mm 00ss 000ms

Sort OK
```

11. Command Line

OCSort command line accepts the following parameters:

ocsort	print version and options.
ocsort --help	print help.
ocsort --version	print version.
ocsort --config	print the value of environment variables.
ocsort ***command line***	execute command line.
ocsort TAKE ***filename***	read filename where are present commands for Sort/Merge.

The file used in the TAKE command is free format.
If in the first column is presents the character '*', the line is a comment.

12. Padding and Truncating

OCSort verify in write output operation the length of record output and value size for padding or truncating data record.

Follow schema presents logical operation.

EOL is End Of Line, terminator for Line Sequential ('0D0A' for Windows,'0A' for Linux).

Line Sequential

Verify Len	Record Input	Record Output	Action
LenIn = LenOut	Input Fixed	Output Fixed	Use LenOut + Append EOL
LenIn < LenOut	Input Fixed	Output Fixed	Use LenOut + PADDING with space + append EOL
LenIn > LenOut	Input Fixed	Output Fixed	Use LenOut + TRUNCATE + append EOL
LenIn = LenOut	Input Fixed	Output Fixed	Force LenIn into LenOut + append EOL
LenIn < LenOut	Input Fixed	Output Fixed	Force LenIn into LenOut + append EOL
LenIn > LenOut	Input Fixed	Output Fixed	Use LenOut + TRUNCATE + append EOL
LenIn = LenOut	Input Variable	Output Fixed	Use LenOut + Append EOL
LenIn < LenOut	Input Variable	Output Fixed	Use LenOut + PADDING with space + append EOL
LenIn > LenOut	Input Variable	Output Fixed	Use LenOut + TRUNCATE + append EOL
LenIn = LenOut	Input Variable	Output Fixed	Force LenIn into LenOut + append EOL
LenIn < LenOut	Input Variable	Output Fixed	Force LenIn into LenOut + append EOL
LenIn > LenOut	Input Variable	Output Fixed	Use LenOut + TRUNCATE + append EOL

Sequential

Verify Len	Record Input	Record Output	Action
LenIn = LenOut	Input Fixed	Output Fixed	Use LenOut
LenIn < LenOut	Input Fixed	Output Fixed	Use LenOut + PADDING with null
LenIn > LenOut	Input Fixed	Output Fixed	Use LenOut + TRUNCATE
LenIn = LenOut	Input Fixed	Output Fixed	Force LenIn into LenOut

LenIn < LenOut	Input Fixed	Output Fixed	Force LenIn into LenOut
LenIn > LenOut	Input Fixed	Output Fixed	Use LenOut + TRUNCATE
LenIn = LenOut	Input Variable	Output Fixed	Use LenOut
LenIn < LenOut	Input Variable	Output Fixed	Use LenOut + PADDING with null
LenIn > LenOut	Input Variable	Output Fixed	Use LenOut + TRUNCATE
LenIn = LenOut	Input Variable	Output Fixed	Force LenIn into LenOut
LenIn < LenOut	Input Variable	Output Fixed	Force LenIn into LenOut
LenIn > LenOut	Input Variable	Output Fixed	Use LenOut + TRUNCATE

13. Retun Code

OCSort has two values for return code:

0 for success
16 for failure

14. File Conversion

OCSort permit to specify 'ORGANIZATION' and 'RECORD TYPE' for output data different structure from input data, to permit the conversion of file format.

In this case OCSort convert data from a structure to another structure, for example, from Sequential to Line Sequential or vice versa.

If you want sort a text file (LS) and you don't know the record length, you can specify RECORD V with max len very large, example:

```
SORT KEY (1,20,CH,A)
USE F1.TXT ORG LS RECORD V,1,3000
GIVE F1.TXT.OUT ORG LS RECORD V,1,3000
```

15. Performance and Tuning

For tuning performance of OCSort is good practices modify the settings of value for memory allocation and modify dimension of area for Memory Mapped File.

OCSORT_MEMSIZE Indicate amount of memory for sort.

OCSORT_MLT Indicate the number of views for MMF in temporary files. This number is multiplied by Page Size of system (example 65536). Increasing this value the view for read file in memory is more greater and can reduce the elapsed time.(Temporary files).

By default OCSORT_MLT assume 63 (Example: 63 * 65536 = 4Mbyte dimension of view for MMF).

16. Limits

The max numbers of input files for Merge is 16.

The max numbers of temporary files is 16. The temporary files is reused when the size of files input is more of size of (Memory OCSORT_MEMSIZE * 16 files).

17. Errors and Warnings

OCSORT produces two types of messages:

- Error format '*OCSort*Snnn'
- Warning format '*OCSort*Wnnn'

For Error message OCSort break execution and terminate operation with message and return code.

For Warning message OCSort continue execution and continue operation with message.

The message string identify a specific condition of error o warning, in the of warning print a specific action.

18. OCSort by examples

18.1. SORT

SORT single file

```
=====================================================================
SORT   FIELDS(3,1,CH,A)
USE    ../PJTestCaseSort/SQBI01          RECORD F,51 ORG SQ
GIVE   ../PJTestCaseSort/SQBI01.SRT.TST  RECORD F,51 ORG SQ
=====================================================================
```

SORT single file with INCLUDE condition

Order KEY

1) Position 37, Len 1, Character, Descending
2) Position 18, Len 17, Character, Ascending

Filter only records with character in position 37 Equal 'C'.

```
=================================================
SORT FIELDS=(37,1,CH,D,18,17,CH,A)
INCLUDE COND=(37,1,EQ,C'C') FORMAT=CH
USE  FIL_100.TXT            RECORD F,3000 ORG LS
GIVE FIL_100.TXT.SRT        RECORD F,3000 ORG LS
=================================================
```

18.2. MERGE

MERGE

Merge files with KEY Position 1, Len 50, Char, Ascending

Input files sorted

Input Record Variable from 1 to 27990 ORGanization Sequential

Output Record Variable from 1 to 27990 ORGanization Sequential

```
=====================================================================
MERGE FIELDS(1,50,CH,A)
      USE   D:\OCSORTTEST\OCFILES\RGX10.DAT     RECORD V,1,27990 ORG SQ
      USE   D:\OCSORTTEST\OCFILES\RGX10.DAT     RECORD V,1,27990 ORG SQ
      USE   D:\OCSORTTEST\OCFILES\RGX10.DAT     RECORD V,1,27990 ORG SQ
      GIVE  D:\OCSORTTEST\OCFILES\RGX10.DAT.MRG RECORD V,1,27990 ORG SQ
=====================================================================
```

MERGE

```
FIELDS=COPY
Copy records from input to output.
Include condition check binary value (low-value)
     Pos     Len    Condition   Value
from 305     04     Not Equal   Hex '00000000'
=====================================================================
USE D:\OCSORTTEST\FilesT\FIL_OUTFIL_500.TXT ORG LS RECORD F,3000
GIVE D:\OCSORTTEST\FilesT\FIL_OUTFIL_500_023.TXT.SRT ORG LS RECORD F,3000
OPTION  VLSHRT,VLSCMP,EQUALS
```

```
MERGE   FIELDS=COPY
INCLUDE COND=(305,4,NE,X'00000000'),FORMAT=CH
=====================================================================
```

18.3. COPY

COPY

Copy data from input to output with record filter.
Input FIXED Line Sequential, Output FIXED Line Sequential
Omitted (not insert in output file) records with condition:

a) Position 1, Len 12, EQual , Character '000000006060'
OR
b) Position 1, Len 12, EQual , Character '000000000030'
OR
c) Position 1, Len 12, EQual , Character '000000000051'

```
=================================================
USE   F1IN.DAT            RECORD F,3000 ORG LS
GIVE  F1IN.DAT_002.SRT    RECORD F,3000 ORG LS
MERGE FIELDS=COPY
OMIT    COND=(01,12,EQ,C'000000006060',OR,
              01,12,EQ,C'000000000030',OR,
              01,12,EQ,C'000000000051'),FORMAT=CH
=================================================
```

SORT without duplicates

Sort Key Pos 5, len 6, Ascending
SUM FIELDS = (NONE) delete duplicates

```
=================================================
USE  FIL_OUTFIL_100.TXT         ORG LS RECORD F,3000
GIVE FIL_OUTFIL_100_020.TXT.SRT ORG LS RECORD F,3000
SORT FIELDS=(5,6,A),FORMAT=CH,EQUALS
SUM FIELDS=(NONE)
=================================================
```

18.4. SUMFIELDS

SUMFIELDS

Sort Key Pos 1, len 1, Ascending
SUM FIELDS Binary fields

```
=====================================================================
SORT  FIELDS(3,1,CH,A)
SUM   FIELDS=(1,2,BI,7,3,BI,15,4,BI,20,3,BI,29,4,BI,34,8,BI,43,8,BI)
USE   ../PJTestCaseSort/SQBI01 RECORD F,51 ORG SQ
GIVE  ../PJTestCaseSort/SQBI01.SRT.TST  RECORD F,51 ORG SQ
```

18.5. OUTREC

OUTREC FIELDS/BUILD

SORT FIELDS = COPY (copy record NO Sort)

Format output : OUTREC

Output structure

```
     Pos      Len     Value
     01       16      Record input  Pos:1,Len 16
     17       2       Blank ('X' = blank)
     19       2       Record input  Pos:18,Len 2
     21       1       Character '-'
     23       2       Record input  Pos:20,Len 2
     25       1       Character '-'
     26       2       Record input  Pos:22,Len 2
     28       2       2 blank
=====================================================================
USE  ../Files/FIL_OUTFIL_200.TXT              ORG LS RECORD F,3000
GIVE ../Files/FIL_OUTFIL_200_007.TXT.SRT      ORG LS RECORD F,3000
SORT FIELDS=COPY
OUTREC=(01,16,2X,18,2,C'-',20,2,C'-',22,2,2X)
END
```

OUTREC FIELDS=(8,2, 20:5,10,3C'ABC',80:X)

Position Input	Len Input	Position output	Len output	Value
8	2	1	2	
5	10	20	10	Characters from pos 5, len10 from input
		30	9 (3 times x 3 char)	'ABCABCABC'
		80		Padding from 39 to 80

OUTREC FIELDS=(5C'LITERAL -',10X'414243',3X'525558',120,18)

Position Input	Len Input	Position output	Len output	Value
		1	45 (5 time x 9 char)	'LITERAL -LITERAL -LITERAL LITERAL-LITERAL-'
		46	30 (10 times 1 char hex)	'ABCABCABCABCABCABCABCABCABC'
		76	9 (3 times x 3 char hex)	'RUXRUXRUX'
80	18	85	18	Input record from 80 for 18 characters

OUTREC FIELDS=(1,40,60:Z,81:X)

Position Input	Len Input	Position output	Len output	Value
1	40	1	40	Input record from 1 for 40 characters
		41	20 (60 abs position – 40 current position)	20 characters with '00' binary
		61	20	21 characters with '20' space

18.6. OUTFIL

OUTFIL INCLUDE

```
Example with more  files for OUTFIL
Each file output with Include condition
The purpose is merge files and write four output.
FNAMES=FOUT201_1
FOUT201_1       Environment Variable
FOUT201_2       Environment Variable
FOUT201_3       Environment Variable
FOUT201_SAVE    Environment Variable
=============================================================================
USE  ../FIL_OUTFIL_001.TXT     ORG LS RECORD F,3000
GIVE ../FIL_OUTFIL_001.TXT.OUT ORG LS RECORD F,3000
MERGE   FIELDS=COPY
OUTFIL INCLUDE=(01,03,CH,EQ,C'201',AND,24,03,CH,LE,C'999'),FNAMES=FOUT201_1
OUTFIL INCLUDE=(01,03,CH,EQ,C'210',AND,24,04,CH,GT,C'0000',AND,24,04,CH,LE,C'9999'),FNAMES=FOUT201_2
OUTFIL INCLUDE=(01,03,CH,EQ,C'230',AND,36,04,CH,GT,C'0000',AND,36,04,CH,LE,C'9999'),FNAMES=FOUT201_3
OUTFIL SAVE,FNAMES=FOUT201_SAVE
=============================================================================
```

OUTFIL OMIT

```
Format output record
OMIT Condition for input.
FOUTKEY_YES       Environment Variable
FOUTKEY_NO        Environment Variable
=============================================================================
USE D:\OCSORTTEST\FilesT\FIL_OUTFIL_050.txt ORG LS RECORD F,3000
GIVE D:\OCSORTTEST\FilesT\FIL_OUTFIL_050.txt.OUT ORG LS RECORD F,3000
  SORT FIELDS=COPY
  OUTFIL OMIT=(156,15,CH,LT,141,15,CH,AND,005,10,CH,EQ,C'KEYMAX800E'),FNAMES=FOUTKEY_YES
  OUTFIL SAVE,FNAMES=FOUTKEY_NO
  END
=============================================================================
```