Dataflow in Practice:
Transparent Dataflow Programming Model for Multicore and Many-core

Oleksandr Pochayevets

Introduction

The number of cores in modern Multicore/ Many-core computer systems grows and will continue to grow in the future up to hundreds and thousands. The parallel multithreading programming for multiple cores becomes a great challenge for those who would like to use multiple cores for speeding-up their applications. The community is getting more and more convinced that a revival of dataflow should close the gap between the evolving number of Multicores/ Many-cores and the difficulties of parallel programming for them.

How do we want to program Multicores/ Many-cores with dataflow? We want to program them like this:

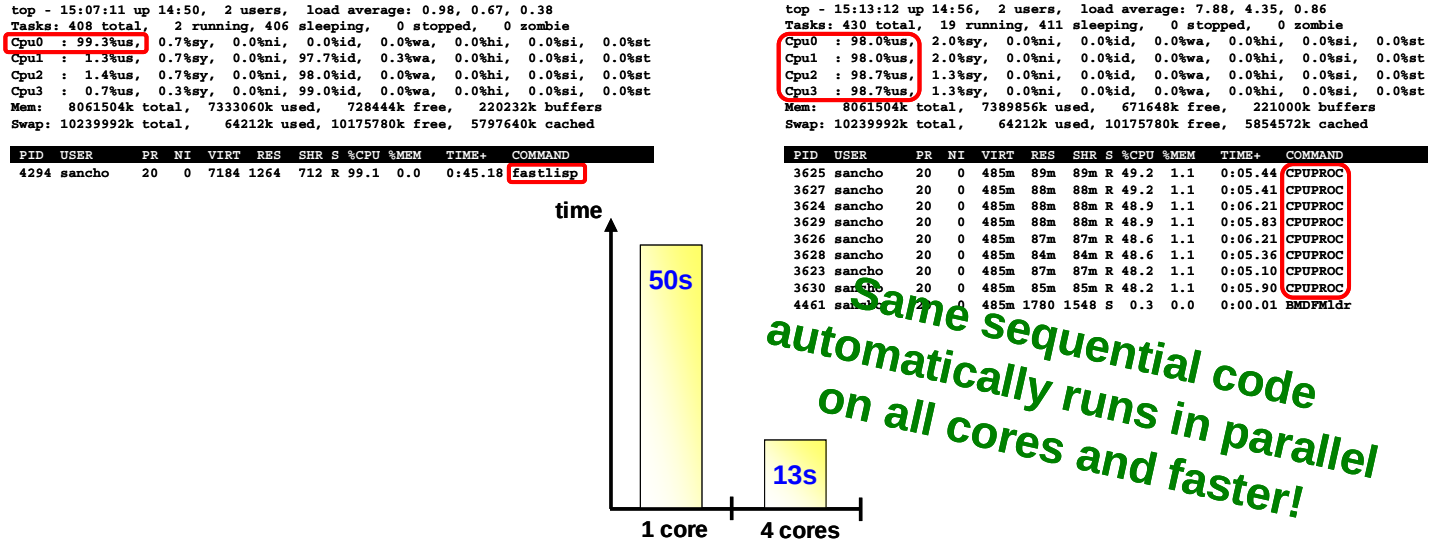
1. We do not want to use any unconventional programming paradigm. We want to use a normal traditional control flow, however, a dataflow engine will run our control flow in a different order according to the dataflow principle: **when operands are ready then operators are executed in parallel on the underlying Multicores/ Many-cores hiding all synchronization issues from us**:

```
a = foo0(i);
b = foo1(i+1);
b = b + 1;
c = foo2(b);
```

2. We do not want to be restricted with a single-assignment. **A dataflow engine should be able to create a different instance of a variable when the variable is re-assigned and then handle all instances correctly.**

Is there such a dataflow engine that can do this for us? Yes, BMDFM (Binary Modular Dataflow Machine; http://bmdfm.com) can do this. Further in this document, we provide a comprehensive test application example on how we program Multicores/ Many-cores using the BMDFM dataflow engine.

What do we want to achieve? We want to program our test applications sequentially with no special directives for parallel execution. We run our test using the BMDFM single-threaded engine that executes the test on a single processor core. Then we run our test using the BMDFM multithreaded engine that executes the test automatically on all available cores in parallel. **We expect to get a speedup that is equal to the number of cores as shown in the following picture!**



Test Applications

We have two test applications:

1. **dftest_prep:** prepares an input test file that contains multiple data clusters of different sizes (each data cluster may have a different number of records of a fixed record size):

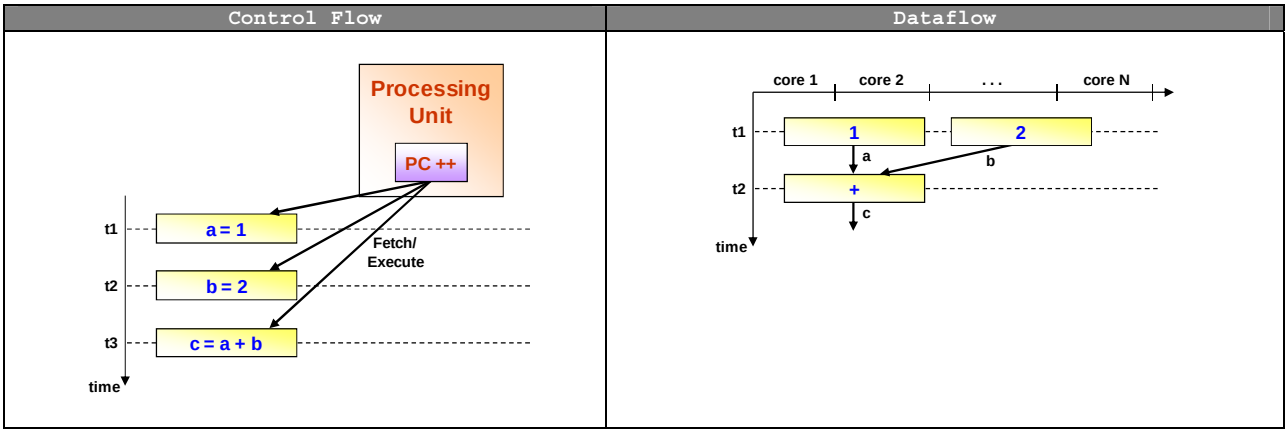
```
<SequenceId><TrailingSpaces>                    ^
<RecordsPerDataCluster><TrailingSpaces>     ^   |R
AOJJYPICIMGNENDSADMMPABZLALVZFNAUWJSLSV     ^   |e
UFCIJPLCPPPCFQDEBDQWDWJDQFNJRFELKHTUWFX     |R |c
MVNOADSFEWVBZRKCIQPRHVWTGDNAZSYMOLAPPTU     |e |o
TPQVOIFQQWGHEBEXHHKIHDGTRSUGHNBBCSWQACH     |c |r
RZOZDPDAXKKGRONKFFEMNSNPUGMLGOSXNGXRWAR     |o |d
ULCACQNNWTSIHKWWFCIQJXIGLPECLEUFPWGSNUG     |r |s
JNYRVJORPQAFZYOGKDKMPPHVFEBXRVDAJCSEMGW     |d |P
BXWGXVUEFYPSOEZJJDLHUHLVQNNVZURBROHOJCT     |s |e
PBIHPNGZWKKEERPZICNECHVDZKLNUOHJPPQFCXE     |P |r
ZHPDMGTMPWZTYGPCFZOTUCBDSQUXSSCSZRVMYPY     |e |D
NLXGJEWMKWAEQDFUWWPTOHWHHNDTMSRZDPGNTDZ     |r |a
EZAJPDOKZKZTZGQHNEKGQCYQFNXSHASMZSVPWJZ     |D |t
VUYPUFFBSJLZAOYQTLOMTOEFOXADTKDPFBEZGKA     |a |a
ZUMZUAXLUJZGCOLICIIGCTJRYKWXRHYRBLQWMNI     |t |C
GWHNZWZHYIPEKJNBHYYFQFEHHPXDBKMIHUWGQVN     |a |l
ODDTNMHPUFNZWSDDZTBDVLPETJAZAVMOYPIMCPB     |C |u
WVOWRHAVGUWKPIATBKTAKPNZNDHAFWBCSQZJXZF     |l |s
ETBOJKPCMZWMJLAJZDQZJNBMGRLPOKVSEXHNHXQ     |u |t
TWNGGYGPYKGYTUZGAQRQFBLYFIFTQCJKZWQGVXV     |s |e
UHCSBXSHXIYOOAZMGISAZVJJUGZACXWXFZQGWIO     |t |r
URNIFOIRUQJUPEEYZLYAOWXLBWBITKXNBKVGYEY     |e |+
TUIOKNTJMFINTEKFFHHOARLOSVKZUOYOJGCTTWC     |r |2
<------------- RecordSize ------------->
. . .
```

2. **dftest_exec:** reads multiple data clusters from the prepared input test file. The multiple data clusters are processed and written to an output test file.

We program our test applications sequentially with conventional control flow and let the BMDFM dataflow engine run everything (what is possible) in parallel on Multicores/ Many-cores.

Background (experts may skip this chapter)

1. **Control flow vs. dataflow:** control flow assumes that a processing unit has a Program Counter (PC) register pointing to executing instruction. The processing unit increments PC, fetches instruction that is pointed by PC and executes the instruction. Contrarily, dataflow tags operands with a token when they are ready. Operators of the dataflow graph process operands with ready-tokens.

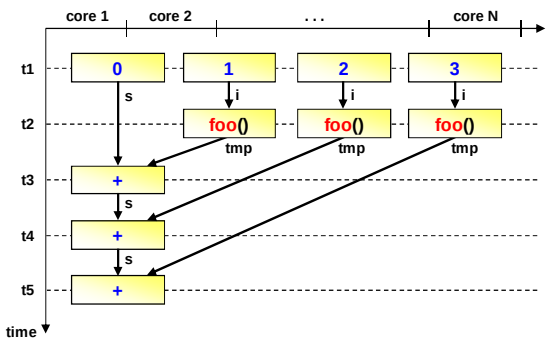


2. **Transparent dataflow semantics:** an assignment <***variable***> = <***expression_of_operators_constants_variables***> creates a new instance of the variable and adds new nodes with dependencies to the dataflow graph dynamically at runtime (later on, variable instances and nodes will be garbage collected from the dataflow graph).

Transparent Dataflow Semantics

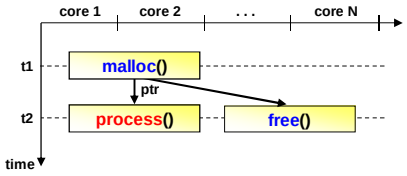
```
s = 0;
for(i = 1; i <= 3; i++)
  s += foo(i);

will be automatically reorganized to:

s = 0;
for(i = 1; i <= 3; i++){
  tmp = foo(i);  // coarse-grain processing
                 // can be done in parallel
  s += tmp;
}
```

Dataflow Graph



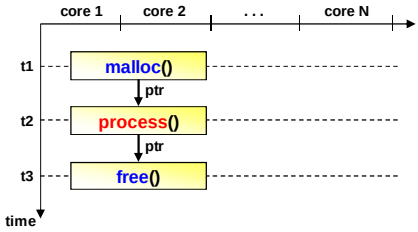
Transparent Dataflow Semantics and Dataflow Graph

```
// Dataflow might delete data earlier
// than data is processed.
ptr = malloc();
process(ptr);
free(ptr);
```



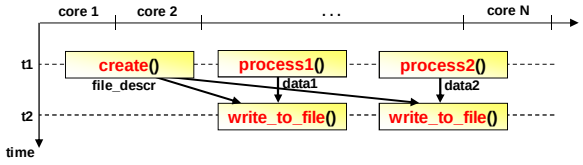
Transparent Dataflow Semantics and Dataflow Graph

```
// Dataflow is synchronized
// via ptr variable.
ptr = malloc();
ptr = process(ptr);
free(ptr);
```



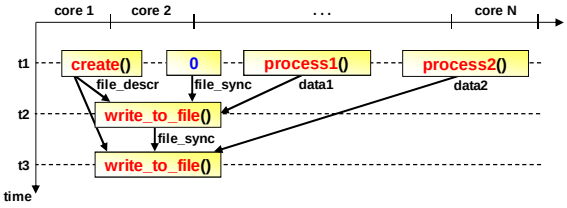
Transparent Dataflow Semantics and Dataflow Graph

```
// Dataflow might write data to file
// in a different order.
file_descr = create();
data1 = process1();
data2 = process2();
write_to_file(file_descr, data1);
write_to_file(file_descr, data2);
```



Transparent Dataflow Semantics and Dataflow Graph

```
// Dataflow is synchronized via file_sync variable.
file_sync = 0;
file_descr = create();
data1 = process1();
data2 = process2();
file_sync = write_to_file(file_descr, file_sync, data1);
file_sync = write_to_file(file_descr, file_sync, data2);
```



3. **C vs. LISP:** we program our applications in C and in a tiny subset of LISP in sake of convenience. We program our seamless helper functions in C. These are low-level coarse-grain functions. A dataflow engine does not apply any parallelization techniques to them. We program the rest of the code in LISP. This code is loaded into the dataflow engine for automatic parallelization. LISP programs are written in a prefix-form that is easy to understand from the following example (refer to the BMDFM comprehensive manual for more information; http://bmdfm.com/download.html).

C

```
for(i = 1; i <= N; i++){
  a = foo0(i);
  b = foo1(i + 1);
  b++;
  printf("a = %d\n", a);
  printf("b = %d\n", b);
}
```

LISP

```
(for i 1 1 N (progn
  (setq a (foo0 i))
  (setq b (foo1 (+ i 1)))
  (setq b (++ b))
  (outf "a = %d\n" a)
  (outf "b = %d\n" b)
))
```

Test Application dftest_prep

Here, we have a pseudo-code for **dftest_prep**:

Pseudo-Code

```
RecordSize               = 40;
MaxRecordsPerDataCluster = 10000;
NumberOfDataClusters     = 1000;
OutputTestFileName       = "dftest_in.dat";

OutputTestFileOrdering_sync = 0;

# The main processing for-loop
for(SequenceId = 1; SequenceId <= NumberOfDataClusters; SequenceId++){

  RecordsPerDataCluster = random([0...MaxRecordsPerDataCluster]);
  DataClusterSize = RecordSize * (RecordsPerDataCluster + 2);

  // We allocate memory for data cluster.
  DataClusterInMemory_ptr = malloc(DataClusterSize);

  // We define a seamless helper function write_data_to_cluster
  // that generates data cluster contents in the allocated memory.
  // We re-assign DataClusterInMemory_ptr in order to prevent the situation
  // where a dataflow engine might call e.g. asyncheap_delete earlier than
  // needed.
  DataClusterInMemory_ptr = write_data_to_cluster(DataClusterInMemory_ptr,
                                                  SequenceId,
                                                  RecordsPerDataCluster,
                                                  RecordSize);

  // We define a seamless helper function write_cluster_to_file
  // that writes data cluster contents from the allocated memory to a file.
  // We re-assign OutputTestFileOrdering_sync in order to prevent the situation
  // where a dataflow engine might create an out-of-order output.
  OutputTestFileOrdering_sync = DataClusterInMemory_ptr =
                                write_cluster_to_file(DataClusterInMemory_ptr,
                                                      DataClusterSize,
                                                      OutputTestFileName,
                                                      OutputTestFileOrdering_sync);

  // We free the allocated memory.
  free(DataClusterInMemory_ptr);
}
# The main processing for-loop implements the following dataflow:
#   for-loop:
#     (free (write_cluster_to_file (write_data_to_cluster (malloc DataClusterSize))))
```

And this is the real code of **dftest_prep**:

```
# dftest_prep.flp
# Refer to the BMDFM comprehensive manual for more information.

(progn
  # Configuration parameters:
  (setq RecordSize                              40)
  (setq MaxRecordsPerDataCluster             10000)
  (setq NumberOfDataClusters                  1000)
  (setq OutputTestFileName           "dftest_in.dat")

  # Sanity checks:
  (setq OutputTestFileName (cat OutputTestFileName
    (if (at "." OutputTestFileName) "" ".dat")))
  (setq OutputTestFileName (strtran OutputTestFileName "."
    (cat (if (id_taskjob) (str (id_taskjob)) "") ".")))
  (if (== -1 (setq fdescr (file_create OutputTestFileName)))
    (progn
      (outf "Error creating file %s\n" OutputTestFileName)
      (exit)
    )
    (setq fdescr (file_close fdescr))
  )
  (if (< RecordSize 40)
    (setq RecordSize 40)
    nil
  )
  (if (< MaxRecordsPerDataCluster 100)
    (setq MaxRecordsPerDataCluster 100)
    nil
  )
  (if (< NumberOfDataClusters 100)
    (setq NumberOfDataClusters 100)
    nil
  )

  # Processing begins here:
  (irnd -1)  # reset random number generator
  (setq OutputTestFileOrdering_sync fdescr)

  # main processing for-loop
  # for(SequenceId=1;SequenceId<=NumberOfDataClusters;SequenceId++){
  (for SequenceId 1 1 NumberOfDataClusters (progn

    (outf "Writing data to cluster %ld\n" SequenceId)

    (setq RecordsPerDataCluster (irnd MaxRecordsPerDataCluster))
    (setq DataClusterSize (* RecordSize (+ RecordsPerDataCluster 2)))

    (setq DataClusterInMemory_ptr
      (asyncheap_create DataClusterSize)  # allocate memory with malloc()
    )

    # We re-assign DataClusterInMemory_ptr in order to prevent the situation
    # where a dataflow engine might call e.g. asyncheap_delete earlier than
    # needed.

    (setq DataClusterInMemory_ptr
      (write_data_to_cluster DataClusterInMemory_ptr
                             SequenceId
                             RecordsPerDataCluster
                             RecordSize)
    )

    # We re-assign OutputTestFileOrdering_sync in order to prevent the situation
    # where a dataflow engine might create an out-of-order output.

    (setq OutputTestFileOrdering_sync (setq DataClusterInMemory_ptr
      (write_cluster_to_file DataClusterInMemory_ptr
                             DataClusterSize
                             OutputTestFileName
                             OutputTestFileOrdering_sync)
    ))

    (asyncheap_delete DataClusterInMemory_ptr)  # free memory with free()

  )) # } end main processing for-loop
  # Processing ends here:
  (space (& 0 OutputTestFileOrdering_sync))
)
# <EOF>
```

Test Application dftest_exec

Here, we have a pseudo-code for **dftest_exec**:

Pseudo-Code

```
RecordSize          = 40;
InputTestFileName   = "dftest_in.dat";
OutputTestFileName  = "dftest_out.dat";

InputFilePosition = 0;
OutputTestFileOrdering_sync = 0;

// We define a seamless helper function get_next_cluster_size
// that returns size of the next data cluster in the input file.
DataClusterSize = get_next_cluster_size(InputTestFileName,
                                        InputFilePosition,
                                        RecordSize);

# The main processing while-loop
while(0 < DataClusterSize){

  // We allocate memory for data cluster.
  DataClusterInMemory_ptr = malloc(DataClusterSize);

  // We define a seamless helper function read_data_to_cluster
  // that reads data cluster contents from the input file to the allocated memory.
  // We re-assign DataClusterInMemory_ptr in order to prevent the situation
  // where a dataflow engine might call e.g. asyncheap_delete earlier than
  // needed.
  DataClusterInMemory_ptr = read_data_to_cluster(DataClusterInMemory_ptr,
                                                 InputTestFileName,
                                                 InputFilePosition,
                                                 DataClusterSize);

  InputFilePosition += DataClusterSize;

  // We define a seamless helper function process_data_cluster
  // that processes data cluster contents in the allocated memory.
  DataClusterInMemory_ptr = process_data_cluster(DataClusterInMemory_ptr,
                                                 RecordSize,
                                                 DataClusterSize);

  // We re-use defined seamless helper function write_cluster_to_file
  // that writes data cluster contents from the allocated memory to a file.
  // We re-assign OutputTestFileOrdering_sync in order to prevent the situation
  // where a dataflow engine might create an out-of-order output.
  OutputTestFileOrdering_sync = DataClusterInMemory_ptr =
                                write_cluster_to_file(DataClusterInMemory_ptr,
                                                      DataClusterSize,
                                                      OutputTestFileName,
                                                      OutputTestFileOrdering_sync);

  // We free the allocated memory.
  free(DataClusterInMemory_ptr);

  // We re-use defined seamless helper function get_next_cluster_size
  // that returns size of the next data cluster in the input file.
  DataClusterSize = get_next_cluster_size(InputTestFileName,
                                          InputFilePosition,
                                          RecordSize);
}
# The main processing while-loop implements the following dataflow:
#   while-loop:
#     (free (write_cluster_to_file (process_data_cluster (read_data_to_cluster
#       (malloc (get_next_cluster_size file))))))
```

And this is the real code of **dftest_exec**:

```
# dftest_exec.flp
# Refer to the BMDFM comprehensive manual for more information.

(progn
  # Configuration parameters:
  (setq RecordSize                              40)
  (setq InputTestFileName             "dftest_in.dat")
  (setq OutputTestFileName           "dftest_out.dat")

  # Sanity checks:
  (setq InputTestFileName (cat InputTestFileName
    (if (at "." InputTestFileName) "" ".dat")))
  (setq InputTestFileName (strtran InputTestFileName "."
    (cat (if (id_taskjob) (str (id_taskjob)) "") ".")))
  (if (== -1 (setq fdescr_ (file_open InputTestFileName)))
    (progn
      (outf "Error opening file %s\n" InputTestFileName)
      (exit)
    )
    (file_close fdescr_)
  )
  (setq OutputTestFileName (cat OutputTestFileName
    (if (at "." OutputTestFileName) "" ".dat")))
  (setq OutputTestFileName (strtran OutputTestFileName "."
    (cat (if (id_taskjob) (str (id_taskjob)) "") ".")))
  (if (== -1 (setq fdescr (file_create OutputTestFileName)))
    (progn
      (outf "Error creating file %s\n" OutputTestFileName)
      (exit)
    )
    (setq fdescr (file_close fdescr))
  )
  (if (< RecordSize 40)
    (setq RecordSize 40)
    nil
  )

  # Processing begins here:
  (setq InputFilePosition 0)
  (setq SequenceId 0)
  (setq OutputTestFileOrdering_sync fdescr)

  # main processing while-loop
  # while(0<(DataClusterSize=get_next_cluster_size())){
  (while (< 0
    (setq DataClusterSize
      (get_next_cluster_size InputTestFileName
                             InputFilePosition
                             RecordSize)
    )
  ) (progn

    (setq SequenceId (++ SequenceId))
    (outf "Processing data cluster %ld\n" SequenceId)

    (setq DataClusterInMemory_ptr
      (asyncheap_create DataClusterSize)  # allocate memory with malloc()
    )

    # We re-assign DataClusterInMemory_ptr in order to prevent the situation
    # where a dataflow engine might call e.g. asyncheap_delete earlier than
    # needed.

    (setq DataClusterInMemory_ptr
      (read_data_to_cluster DataClusterInMemory_ptr
                            InputTestFileName
                            InputFilePosition
                            DataClusterSize)
    )

    (setq InputFilePosition (+ InputFilePosition DataClusterSize))

    (setq DataClusterInMemory_ptr
      (process_data_cluster DataClusterInMemory_ptr
                            RecordSize
                            DataClusterSize)
    )

    # We re-assign OutputTestFileOrdering_sync in order to prevent the situation
    # where a dataflow engine might create an out-of-order output.

    (setq OutputTestFileOrdering_sync (setq DataClusterInMemory_ptr
      (write_cluster_to_file DataClusterInMemory_ptr
                             DataClusterSize
                             OutputTestFileName
                             OutputTestFileOrdering_sync)
    ))

    (asyncheap_delete DataClusterInMemory_ptr)  # free memory with free()

  )) # } end main processing while-loop
  # Processing ends here:
  (space (& 0 OutputTestFileOrdering_sync))
)
# <EOF>
```

Helper Functions

We write our helper functions in pure C. We keep them away from the dataflow engine (they are seamless for the dataflow engine) in order to avoid unnecessary dataflow scheduling:

```
#include <cflp_udf.h>  /* BMDFM C-interface */
/* Refer to the BMDFM comprehensive manual for more information. */

#define ULO unsigned long int
#define SLO signed long int
#define UCH unsigned char
#define CHR char
#define ECODE_RT__DFTEST_OUT_OF_MEMORY 247
#define ECODE_RT__DFTEST_FILE_IO_FAIL  248

void dftest_write_data_to_cluster(const ULO *dat_ptr, struct fastlisp_data *ret_dat){
  SLO DataClusterInMemory_ptr,SequenceId,RecordsPerDataCluster,RecordSize,i,j;
  CHR *heap_ptr;
  ret_ival(dat_ptr,  &DataClusterInMemory_ptr);  /* read arguments */
  ret_ival(dat_ptr+1,&SequenceId);               /* from the stack */
  ret_ival(dat_ptr+2,&RecordsPerDataCluster);
  ret_ival(dat_ptr+3,&RecordSize);
  if(noterror()){
    if(0==DataClusterInMemory_ptr)
      rise_error_info(ECODE_RT__DFTEST_OUT_OF_MEMORY,
        "dftest_write_data_to_cluster(): memory allocation failure");
    else{
      heap_ptr=(CHR*)DataClusterInMemory_ptr;

      j=sprintf(heap_ptr,"%ld",SequenceId);
      heap_ptr+=j;
      for(j++;j<RecordSize;j++)
        *heap_ptr++=' ';
      *heap_ptr++='\n';

      j=sprintf(heap_ptr,"%ld",RecordsPerDataCluster);
      heap_ptr+=j;
      for(j++;j<RecordSize;j++)
        *heap_ptr++=' ';
      *heap_ptr++='\n';

      for(i=0;i<RecordsPerDataCluster;i++){
        for(j=1;j<RecordSize;j++)
          *heap_ptr++=(CHR)((double)rand()/RAND_MAX*26)+'A';
        *heap_ptr++='\n';
      }
    }
    ret_dat->single=1;
    ret_dat->type='I';
    ret_dat->value.ival=DataClusterInMemory_ptr;  /* return value */
  }
  return;
}

void dftest_write_cluster_to_file(const ULO *dat_ptr, struct fastlisp_data *ret_dat){
  SLO DataClusterInMemory_ptr,DataClusterSize,OutputTestFileOrdering_sync;
  CHR *OutputTestFileName=NULL,**FileName_p=&OutputTestFileName;
  int outfile_descr=-1,*file_descr_p=&outfile_descr;
#ifdef DFTEST_USE_GLOBALS
  FileName_p=&dftest_globals_tsk_p[get_id_taskjob()].outfile_name;
  file_descr_p=&dftest_globals_tsk_p[get_id_taskjob()].outfile_descr;
#endif
  ret_ival(dat_ptr,  &DataClusterInMemory_ptr);  /* read arguments */
  ret_ival(dat_ptr+1,&DataClusterSize);          /* from the stack */
  ret_sval(dat_ptr+2,&OutputTestFileName);
  ret_ival(dat_ptr+3,&OutputTestFileOrdering_sync);
  if(noterror()){
    if(0==DataClusterInMemory_ptr)
      rise_error_info(ECODE_RT__DFTEST_OUT_OF_MEMORY,
        "dftest_write_cluster_to_file(): memory allocation failure");
    else
      if(((-1==*file_descr_p||!cmp(OutputTestFileName,*FileName_p)))&&
         (-1==(*file_descr_p=open(equ(FileName_p,OutputTestFileName),
         O_WRONLY|O_APPEND))))
        rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
          "dftest_write_cluster_to_file():"
          " file I/O failure while opening output file");
      else{
        lseek(*file_descr_p,0,SEEK_END);
        if(DataClusterSize!=write(*file_descr_p,(void*)DataClusterInMemory_ptr,
           DataClusterSize))
          rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
            "dftest_write_cluster_to_file():"
            " file I/O failure while writing to output file");
#ifndef DFTEST_USE_GLOBALS
        close(*file_descr_p);
#endif
      }
    ret_dat->single=1;
    ret_dat->type='I';
    ret_dat->value.ival=DataClusterInMemory_ptr;  /* return value */
  }
  free_string(&OutputTestFileName);
  return;
}
```

```
void dftest_get_next_cluster_size(const ULO *dat_ptr, struct fastlisp_data *ret_dat){
  SLO InputFilePosition,RecordSize,i,DataClusterSize=0;
  CHR *InputTestFileName=NULL,**FileName_p=&InputTestFileName,buff[21];
  int inpfile_descr=-1,*file_descr_p=&inpfile_descr;
#ifdef DFTEST_USE_GLOBALS
  ULO globidx;
  globidx=am_I_in_the_multithreaded_module()?get_id_taskjob()*get_n_cpuproc()+
    get_id_cpuproc():get_id_taskjob();
  FileName_p=&dftest_globals_thr_p[globidx].inpfile_name;
  file_descr_p=&dftest_globals_thr_p[globidx].inpfile_descr;
#endif
  ret_sval(dat_ptr,  &InputTestFileName);  /* read arguments */
  ret_ival(dat_ptr+1,&InputFilePosition);  /* from the stack */
  ret_ival(dat_ptr+2,&RecordSize);
  if(noterror()){
    if(((-1==*file_descr_p||!cmp(InputTestFileName,*FileName_p)))&&
       (-1==(*file_descr_p=open(equ(FileName_p,InputTestFileName),O_RDONLY))))
      rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
        "dftest_get_next_cluster_size():"
        " file I/O failure while opening input file");
    else{
      if(lseek(*file_descr_p,0,SEEK_END)>InputFilePosition)
        if(-1==lseek(*file_descr_p,InputFilePosition+RecordSize,SEEK_SET))
          rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
            "dftest_get_next_cluster_size():"
            " file I/O failure while seeking in input file");
        else{
          for(i=0;i<(SLO)sizeof(buff);i++)
            buff[i]=0;
          if(sizeof(buff)!=read(*file_descr_p,(void*)buff,sizeof(buff)))
            rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
              "dftest_get_next_cluster_size():"
              " file I/O failure while reading from input file");
          else
            DataClusterSize=RecordSize*(atol(buff)+2);
        }
#ifndef DFTEST_USE_GLOBALS
      close(*file_descr_p);
#endif
    }
    ret_dat->single=1;
    ret_dat->type='I';
    ret_dat->value.ival=DataClusterSize;  /* return value */
  }
  free_string(&InputTestFileName);
  return;
}

void dftest_read_data_to_cluster(const ULO *dat_ptr, struct fastlisp_data *ret_dat){
  SLO DataClusterInMemory_ptr,InputFilePosition,DataClusterSize;
  CHR *InputTestFileName=NULL,**FileName_p=&InputTestFileName;
  int inpfile_descr=-1,*file_descr_p=&inpfile_descr;
#ifdef DFTEST_USE_GLOBALS
  ULO globidx;
  globidx=am_I_in_the_multithreaded_module()?get_id_taskjob()*get_n_cpuproc()+
    get_id_cpuproc():get_id_taskjob();
  FileName_p=&dftest_globals_thr_p[globidx].inpfile_name;
  file_descr_p=&dftest_globals_thr_p[globidx].inpfile_descr;
#endif
  ret_ival(dat_ptr,  &DataClusterInMemory_ptr);  /* read arguments */
  ret_sval(dat_ptr+1,&InputTestFileName);        /* from the stack */
  ret_ival(dat_ptr+2,&InputFilePosition);
  ret_ival(dat_ptr+3,&DataClusterSize);
  if(noterror()){
    if(0==DataClusterInMemory_ptr)
      rise_error_info(ECODE_RT__DFTEST_OUT_OF_MEMORY,
        "dftest_read_data_to_cluster(): memory allocation failure");
    else
      if(((-1==*file_descr_p||!cmp(InputTestFileName,*FileName_p)))&&
         (-1==(*file_descr_p=open(equ(FileName_p,InputTestFileName),
         O_RDONLY))))
        rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
          "dftest_read_data_to_cluster():"
          " file I/O failure while opening input file");
      else{
        if(-1==lseek(*file_descr_p,InputFilePosition,SEEK_SET))
          rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
            "dftest_read_data_to_cluster():"
            " file I/O failure while seeking in input file");
        else
          if(DataClusterSize!=read(*file_descr_p,
             (void*)DataClusterInMemory_ptr,DataClusterSize))
            rise_error_info(ECODE_RT__DFTEST_FILE_IO_FAIL,
              "dftest_read_data_to_cluster():"
              " file I/O failure while reading from input file");
#ifndef DFTEST_USE_GLOBALS
        close(*file_descr_p);
#endif
      }
    ret_dat->single=1;
    ret_dat->type='I';
    ret_dat->value.ival=DataClusterInMemory_ptr;  /* return value */
  }
  free_string(&InputTestFileName);
  return;
}
```

```
void dftest_process_data_cluster(const ULO *dat_ptr, struct fastlisp_data *ret_dat){
  SLO DataClusterInMemory_ptr,RecordSize,DataClusterSize,i;
  ret_ival(dat_ptr,  &DataClusterInMemory_ptr);  /* read arguments */
  ret_ival(dat_ptr+1,&RecordSize);               /* from the stack */
  ret_ival(dat_ptr+2,&DataClusterSize);
  if(noterror()){
    if(0==DataClusterInMemory_ptr)
      rise_error_info(ECODE_RT__DFTEST_OUT_OF_MEMORY,
        "dftest_process_data_cluster(): memory allocation failure");
    else{
      /* Place your data cluster processing code here (e.g. sorting of the
         data cluster records).
         This is just a stub:
           - time complexity linearly depends on DataClusterSize;
           - data cluster remains unchanged in order to allow one to compare
             input and output test files. */
      for(i=0;i<DataClusterSize*6;i++)
        time(NULL);  /* dummy operation */
    }
    ret_dat->single=1;
    ret_dat->type='I';
    ret_dat->value.ival=DataClusterInMemory_ptr;  /* return value */
  }
  return;
}
```

```
/* This option changes stateless file descriptors to stateful ones: */
/* #define DFTEST_USE_GLOBALS */

#ifdef DFTEST_USE_GLOBALS
/* System may run multiple task job instances in parallel (max. N_IORBP jobs),
   thus, multiple copies of global entities might be needed.
   Additionally, system may run in multithreaded mode (N_CPUPROC threads),
   thus, multiple copies of global entities might be needed as well.
*/
/* Global entities that are sensitive to parallel task job instances
   only:
   - every global entity is an array of [0;get_n_taskjob()[,
     current entity is [get_id_taskjob()].
*/
  struct _dftest_globals_tsk{
    CHR *outfile_name;         /* only serial ordered WRITE operations */
    int outfile_descr;         /* are enabled in our dataflow program */
  } *dftest_globals_tsk_p=NULL;

/* Global entities that are sensitive to parallel task job instances
   and threads:
   - if !am_I_in_the_multithreaded_module() then
       every global entity is an array of [0;get_n_taskjob()[,
       current entity is [get_id_taskjob()];
   - if am_I_in_the_multithreaded_module() then
       every global entity is an array of [0;get_n_taskjob()*get_n_cpuproc()[,
       current entity is [get_id_taskjob()*get_n_cpuproc()+get_id_cpuproc()].
*/
  struct _dftest_globals_thr{
    CHR *inpfile_name;         /* parallel simultaneous READ operations */
    int inpfile_descr;         /* are enabled in our dataflow program */
  } *dftest_globals_thr_p=NULL;
#endif
```

```
void startup_callback(void){
#ifdef DFTEST_USE_GLOBALS
  /* Here we initialize our global entities once the system starts. */
  ULO i,globcnt;
  if(!dftest_globals_thr_p){
    globcnt=get_n_taskjob()*(am_I_in_the_multithreaded_module()?
      get_n_cpuproc():1);
    if((dftest_globals_thr_p=(struct _dftest_globals_thr*)malloc(globcnt*
       sizeof(struct _dftest_globals_thr)))==NULL){
      fprintf(stderr,
        "\nstartup_callback() for dftest: memory allocation failure\n");
      exit(1);
    }
    for(i=0;i<globcnt;i++){
      dftest_globals_thr_p[i].inpfile_name=NULL;
      dftest_globals_thr_p[i].inpfile_descr=-1;
    }
  }
  if(!dftest_globals_tsk_p){
    if((dftest_globals_tsk_p=(struct _dftest_globals_tsk*)malloc(
       get_n_taskjob()*sizeof(struct _dftest_globals_tsk)))==NULL){
      fprintf(stderr,
        "\nstartup_callback() for dftest: memory allocation failure\n");
      exit(1);
    }
    for(i=0;i<get_n_taskjob();i++){
      dftest_globals_tsk_p[i].outfile_name=NULL;
      dftest_globals_tsk_p[i].outfile_descr=-1;
    }
  }
#endif
  return;
}
```

```
void taskjob_end_callback(ULO id_taskjob){
#ifdef DFTEST_USE_GLOBALS
  /* Here we deinitialize our global entities each time a task job ends. */
  ULO i,globcnt,globidx;
  globcnt=am_I_in_the_multithreaded_module()?get_n_cpuproc():1;
  for(i=0;i<globcnt;i++){
    globidx=id_taskjob*globcnt+i;
    free_string(&dftest_globals_thr_p[globidx].inpfile_name);
    if(-1!=dftest_globals_thr_p[globidx].inpfile_descr){
      close(dftest_globals_thr_p[globidx].inpfile_descr);
      dftest_globals_thr_p[globidx].inpfile_descr=-1;
    }
  }
  free_string(&dftest_globals_tsk_p[id_taskjob].outfile_name);
  if(-1!=dftest_globals_tsk_p[id_taskjob].outfile_descr){
    close(dftest_globals_tsk_p[id_taskjob].outfile_descr);
    dftest_globals_tsk_p[id_taskjob].outfile_descr=-1;
  }
#endif
  return;
}
```

```
/* Register functions. */
INSTRUCTION_STRU INSTRUCTION_SET[]={
  {"WRITE_DATA_TO_CLUSTER",4,'I',(UCH*)"IIII",&dftest_write_data_to_cluster},
  {"WRITE_CLUSTER_TO_FILE",4,'I',(UCH*)"IISI",&dftest_write_cluster_to_file},
  {"GET_NEXT_CLUSTER_SIZE",3,'I',(UCH*)"SII", &dftest_get_next_cluster_size},
  {"READ_DATA_TO_CLUSTER", 4,'I',(UCH*)"ISII",&dftest_read_data_to_cluster},
  {"PROCESS_DATA_CLUSTER", 3,'I',(UCH*)"III", &dftest_process_data_cluster}
};
const ULO INSTRUCTIONS=sizeof(INSTRUCTION_SET)/sizeof(INSTRUCTION_STRU);
```

Running the Tests

We run our tests using the BMDFM single-threaded engine and multithreaded dataflow engine with the following batch shell-script:

```
#!/bin/sh

# Prepare dftest_in.dat
# with single-threaded engine and log
fastlisp dftest_prep.flp >dftest_prep.fastlisp

# Prepare dftest_in.dat
# with multithreaded dataflow engine and log
BMDFMldr dftest_prep.flp >dftest_prep.BMDFMldr

# Process dftest_in.dat to dftest_out.dat
# with single-threaded engine and log
fastlisp dftest_exec.flp >dftest_exec.fastlisp

# Process dftest_in.dat to dftest_out.dat
# with multithreaded dataflow engine and log
BMDFMldr dftest_exec.flp >dftest_exec.BMDFMldr

# Compare
diff dftest_in.dat dftest_out.dat
```

We tested both **dftest_prep** and **dftest_exec** on an affordable 64-way SMP x86-64 machine. The Linux OS reported in total 64 2.5GHz available processors (that actually are <***processors_on_dies***> multiplied by <***cores_per_processor_die***> multiplied by <***simultaneous_threads_per_core***>):

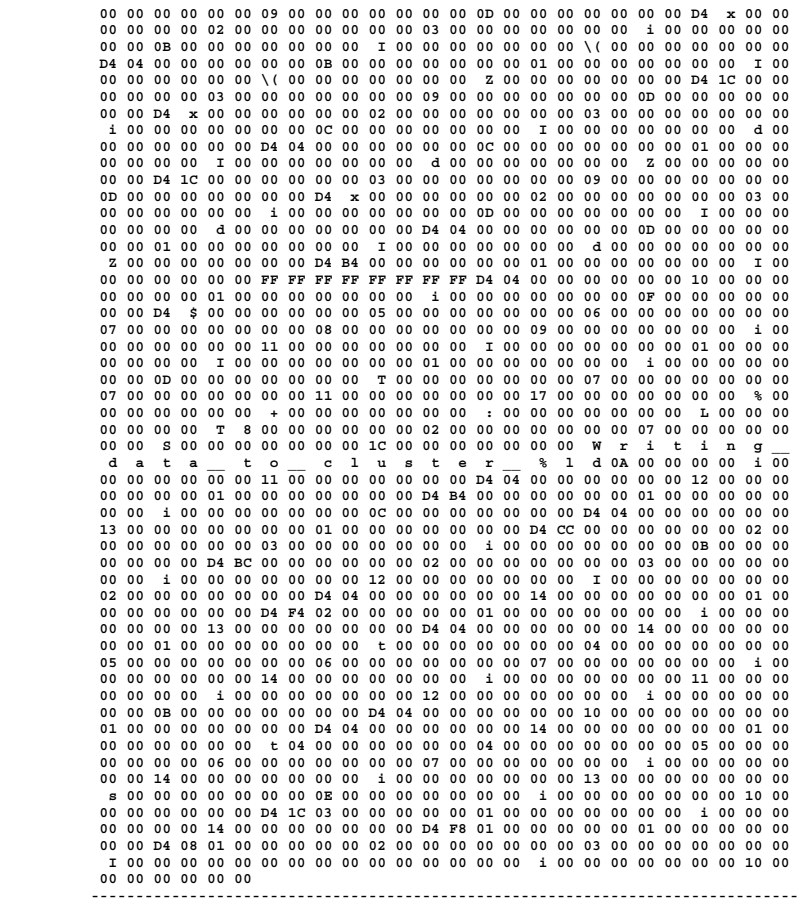
Test Application	Single-threaded Control Flow	Multithreaded Dataflow
dftest_prep (generates ~200MB of test data)	**~3-4sec.**	**~0.1-0.2sec.**
dftest_exec (processes generated test data)	**~50-60sec.**	**~1.5-2sec.**

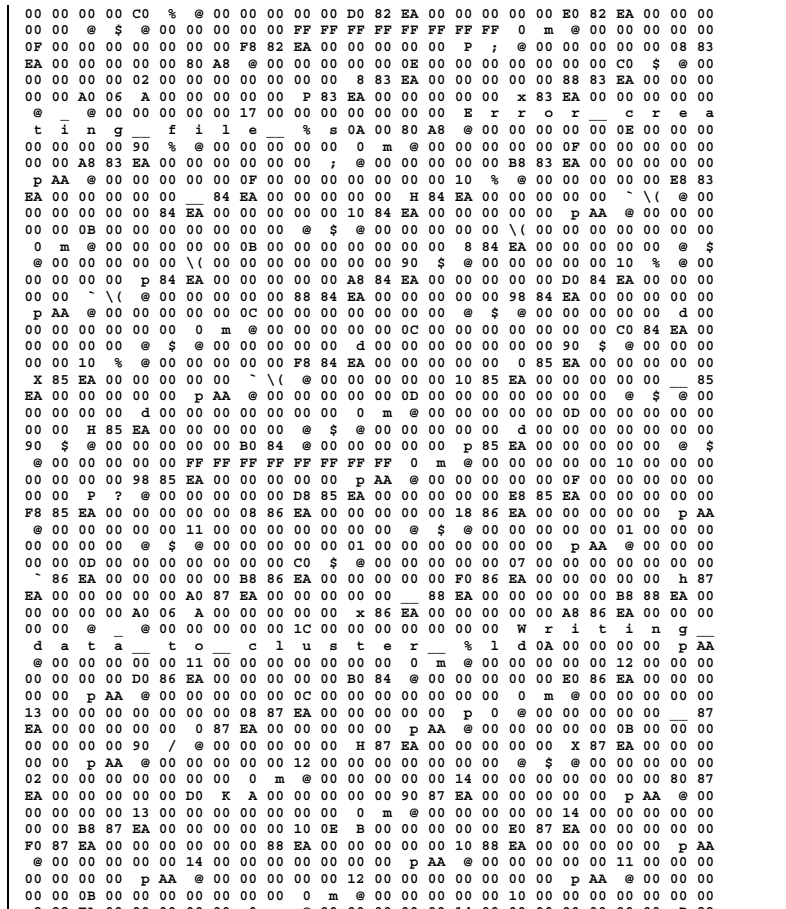
Appendix: Log Files

The log files are provided in this document for those who are interested in automatic control-flow-to-dataflow code transformations and time measurements:

dftest_prep.fastlisp

```
`Cursor invisible (vi)' capability was not found, default value is used!
`Cursor visible (ve)' capability was not found, default value is used!
Current termcap settings:
  TERM_TYPE=`vt100'; LINES_TERM=`24'; COLUMNS_TERM=`80';
  CLRSCR_TERM=`\e[H\e[J'; REVERSE_TERM=`\e[7m'; BLINK_TERM=`\e[5m';
  BOLD_TERM=`\e[1m'; NORMAL_TERM=`\e[0m'; HIDECURSOR_TERM=`';
  SHOWCURSOR_TERM=`'; GOTOCURSOR_TERM=`\e[%i%d;%dH'.
Checking whether the `dftest_prep.flp' file is already precompiled...
Reading the `fastlisp.cfg' configuration profile...
Checking the syntax of the configuration profile...
Squeezing the nested source PROGN statements in Global FastLisp function set...
  Redundant nested source PROGN statements removed: 0.
Looking for uninitialized variables/arrays in Global FastLisp function set...
Resolving data types in Global FastLisp function set...
Reading the `dftest_prep.flp' source FastLisp file...
*** Resetting time counters (first null assignment)... ***
Modifying the FastLisp code (PATTERN No# 1)...
  (PROGN <FastLisp_prog>)
Checking the syntax of the source FastLisp file...
Modifying the FastLisp code (PATTERN No# 2)...
  (PROGN {(SETQ <termcap_var> <termcap_val>) }<FastLisp_prog>)
Squeezing the nested source PROGN statements...
  Redundant nested source PROGN statements removed: 2.
Looking for uninitialized variables/arrays in the FastLisp code...
Resolving data types in the FastLisp code...
--------------------------------------------------------------------------------
(PROGN
  (SETQ@S TERM_TYPE@S "vt100")
  (SETQ@I LINES_TERM@I 24)
  (SETQ@I COLUMNS_TERM@I 80)
  (SETQ@S CLRSCR_TERM@S "\e[H\e[J")
  (SETQ@S REVERSE_TERM@S "\e[7m")
  (SETQ@S BLINK_TERM@S "\e[5m")
  (SETQ@S BOLD_TERM@S "\e[1m")
  (SETQ@S NORMAL_TERM@S "\e[0m")
  (SETQ@S HIDECURSOR_TERM@S "")
  (SETQ@S SHOWCURSOR_TERM@S "")
  (SETQ@S GOTOCURSOR_TERM@S "\e[%i%d;%dH")
  (SETQ@I RECORDSIZE@I 40)
  (SETQ@I MAXRECORDSPERDATACLUSTER@I 10000)
  (SETQ@I NUMBEROFDATACLUSTERS@I 1000)
  (SETQ@S OUTPUTTESTFILENAME@S "dftest_in.dat")
  (SETQ@S
    OUTPUTTESTFILENAME@S
    (CAT@J
      OUTPUTTESTFILENAME@S
      (IF@J (AT@J "." OUTPUTTESTFILENAME@S) "" ".dat")
    )
  )
  (SETQ@S
    OUTPUTTESTFILENAME@S
    (STRTRAN@J
      OUTPUTTESTFILENAME@S "."
      (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
    )
  )
  (IF@J
    (==@I -1 (SETQ@I FDESCR@I (FILE_CREATE@J OUTPUTTESTFILENAME@S)))
    (PROGN (OUTF "Error creating file %s\n" OUTPUTTESTFILENAME@S) (EXIT))
    (SETQ@I FDESCR@I (FILE_CLOSE@J FDESCR@I))
  )
  (IF@J (<@I RECORDSIZE@I 40) (SETQ@I RECORDSIZE@I 40) NIL)
  (IF@J
    (<@I MAXRECORDSPERDATACLUSTER@I 100)
    (SETQ@I MAXRECORDSPERDATACLUSTER@I 100)
    NIL
  )
  (IF@J
    (<@I NUMBEROFDATACLUSTERS@I 100)
    (SETQ@I NUMBEROFDATACLUSTERS@I 100)
    NIL
  )
  (IRND@J -1)
  (SETQ@I OUTPUTTESTFILEORDERING_SYNC@I FDESCR@I)
  (FOR@J
    SEQUENCEID@I 1 1 NUMBEROFDATACLUSTERS@I
    (PROGN
      (OUTF "Writing data to cluster %ld\n" SEQUENCEID@I)
      (SETQ@I RECORDSPERDATACLUSTER@I (IRND@J MAXRECORDSPERDATACLUSTER@I))
      (SETQ@I
        DATACLUSTERSIZE@I
        (*@J RECORDSIZE@I (+@J RECORDSPERDATACLUSTER@I 2))
      )
      (SETQ@I DATACLUSTERINMEMORY_PTR@I (ASYNCHEAP_CREATE@J DATACLUSTERSIZE@I))
      (SETQ@I
        DATACLUSTERINMEMORY_PTR@I
        (WRITE_DATA_TO_CLUSTER@J
          DATACLUSTERINMEMORY_PTR@I SEQUENCEID@I RECORDSPERDATACLUSTER@I
          RECORDSIZE@I
        )
      )
      (SETQ@I
        OUTPUTTESTFILEORDERING_SYNC@I
        (SETQ@I
          DATACLUSTERINMEMORY_PTR@I
          (WRITE_CLUSTER_TO_FILE@J
            DATACLUSTERINMEMORY_PTR@I DATACLUSTERSIZE@I OUTPUTTESTFILENAME@S
            OUTPUTTESTFILEORDERING_SYNC@I
          )
        )
      )
      (ASYNCHEAP_DELETE@J DATACLUSTERINMEMORY_PTR@I)
    )
  )
  (SPACE@J (&@J 0 OUTPUTTESTFILEORDERING_SYNC@I))
)
--------------------------------------------------------------------------------
(PROGN (SETQ@S TERM_TYPE@S "vt100") (SETQ@I LINES_TERM@I 24) (SETQ@I COLUMNS_TE
RM@I 80) (SETQ@S CLRSCR_TERM@S "\e[H\e[J") (SETQ@S REVERSE_TERM@S "\e[7m") (SET
Q@S BLINK_TERM@S "\e[5m") (SETQ@S BOLD_TERM@S "\e[1m") (SETQ@S NORMAL_TERM@S "\
e[0m") (SETQ@S HIDECURSOR_TERM@S "") (SETQ@S SHOWCURSOR_TERM@S "") (SETQ@S GOTO
CURSOR_TERM@S "\e[%i%d;%dH") (SETQ@I RECORDSIZE@I 40) (SETQ@I MAXRECORDSPERDATA
CLUSTER@I 10000) (SETQ@I NUMBEROFDATACLUSTERS@I 1000) (SETQ@S OUTPUTTESTFILENAM
E@S "dftest_in.dat") (SETQ@S OUTPUTTESTFILENAME@S (CAT@J OUTPUTTESTFILENAME@S (
IF@J (AT@J "." OUTPUTTESTFILENAME@S) "" ".dat"))) (SETQ@S OUTPUTTESTFILENAME@S
(STRTRAN@J OUTPUTTESTFILENAME@S "." (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJO
B)) "") ".")))) (IF@J (==@I -1 (SETQ@I FDESCR@I (FILE_CREATE@J OUTPUTTESTFILENAM
E@S))) (PROGN (OUTF "Error creating file %s\n" OUTPUTTESTFILENAME@S) (EXIT)) (S
ETQ@I FDESCR@I (FILE_CLOSE@J FDESCR@I))) (IF@J (<@I RECORDSIZE@I 40) (SETQ@I RE
CORDSIZE@I 40) NIL) (IF@J (<@I MAXRECORDSPERDATACLUSTER@I 100) (SETQ@I MAXRECOR
DSPERDATACLUSTER@I 100) NIL) (IF@J (<@I NUMBEROFDATACLUSTERS@I 100) (SETQ@I NUM
BEROFDATACLUSTERS@I 100) NIL) (IRND@J -1) (SETQ@I OUTPUTTESTFILEORDERING_SYNC@I
 FDESCR@I) (FOR@J SEQUENCEID@I 1 1 NUMBEROFDATACLUSTERS@I (PROGN (OUTF "Writing
 data to cluster %ld\n" SEQUENCEID@I) (SETQ@I RECORDSPERDATACLUSTER@I (IRND@J M
AXRECORDSPERDATACLUSTER@I)) (SETQ@I DATACLUSTERSIZE@I (*@J RECORDSIZE@I (+@J RE
CORDSPERDATACLUSTER@I 2))) (SETQ@I DATACLUSTERINMEMORY_PTR@I (ASYNCHEAP_CREATE@
J DATACLUSTERSIZE@I)) (SETQ@I DATACLUSTERINMEMORY_PTR@I (WRITE_DATA_TO_CLUSTER@
J DATACLUSTERINMEMORY_PTR@I SEQUENCEID@I RECORDSPERDATACLUSTER@I RECORDSIZE@I))
 (SETQ@I OUTPUTTESTFILEORDERING_SYNC@I (SETQ@I DATACLUSTERINMEMORY_PTR@I (WRITE
_CLUSTER_TO_FILE@J DATACLUSTERINMEMORY_PTR@I DATACLUSTERSIZE@I OUTPUTTESTFILENA
ME@S OUTPUTTESTFILEORDERING_SYNC@I))) (ASYNCHEAP_DELETE@J DATACLUSTERINMEMORY_P
TR@I))) (SPACE@J (&@J 0 OUTPUTTESTFILEORDERING_SYNC@I)))
--------------------------------------------------------------------------------
*You may recompile the `fastlisp' with commented `#define _NOISY_MODE_'
 to disable print of the FastLisp code.
Compiling the Global FastLisp function source code (Pass One)...
Compiled Global function bytecode size is 56bytes.
--------------------------------------------------------------------------------
 D5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  T 00
 00 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00  Z 00 00 00
 00 00 00 00
--------------------------------------------------------------------------------
*You may recompile the `fastlisp' with commented `#define _NOISY_MODE1_'
 to disable print of the compiled Global function bytecode.
Compiling the FastLisp source code (Pass One)...
Compiled bytecode size is 3048bytes.
--------------------------------------------------------------------------------
 D5 01 00 00 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  T 00
 00 00 00 00 00 00 19 00 00 00 00 00 00 00 19 00 00 00 00 00 00 00 1E 00 00 00
 00 00 00 00 \" 00 00 00 00 00 00 00  & 00 00 00 00 00 00 00  + 00 00 00 00 00
 00 00  0 00 00 00 00 00 00 00  5 00 00 00 00 00 00 00  : 00 00 00 00 00 00 00
  ? 00 00 00 00 00 00 00  D 00 00 00 00 00 00 00  I 00 00 00 00 00 00 00  O 00
 00 00 00 00 00 00  S 00 00 00 00 00 00 00  W 00 00 00 00 00 00 00  [ 00 00 00
 00 00 00 00  a 00 00 00 00 00 00 00  z 00 00 00 00 00 00 00 96 00 00 00 00 00
 00 00 BB 00 00 00 00 00 00 00 CB 00 00 00 00 00 00 00 DB 00 00 00 00 00 00 00
 EB 00 00 00 00 00 00 00 EE 00 00 00 00 00 00 00 F2 00 00 00 00 00 00 00  W 01
 00 00 00 00 00 00 00 D4 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00
 00 00 00 00  S 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00  v  t  1  0  0 00
 00 00 D4 04 00 00 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
  I 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00 D4 04 00 00 00 00 00 00 02 00
 00 00 00 00 00 00 01 00 00 00 00 00 00 00  I 00 00 00 00 00 00 00  P 00 00 00
 00 00 00 00 D4 05 00 00 00 00 00 00 03 00 00 00 00 00 00 00 01 00 00 00 00 00
 00 00  S 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 1B  [  H 1B  [  J 00 00
 D4 05 00 00 00 00 00 00 04 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  S 00
 00 00 00 00 00 00 04 00 00 00 00 00 00 00 1B  [  7  m 00 00 00 00 D4 05 00 00
 00 00 00 00 05 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00
 00 00 04 00 00 00 00 00 00 00 1B  [  5  m 00 00 00 00 D4 05 00 00 00 00 00 00
 06 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 04 00
 00 00 00 00 00 00 1B  [  1  m 00 00 00 00 D4 05 00 00 00 00 00 00 07 00 00 00
 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 04 00 00 00 00 00
 00 00 1B  [  0  m 00 00 00 00 D4 05 00 00 00 00 00 00 08 00 00 00 00 00 00 00
 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 D4 05 00 00 00 00 00 00 09 00 00 00 00 00 00 00 01 00 00 00
 00 00 00 00  S 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 D4 05 00 00 00 00 00 00 0A 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
  S 00 00 00 00 00 00 00 0A 00 00 00 00 00 00 00 1B  [  %  i  %  d  ;  %  d  H
 00 00 00 00 00 00 D4 04 00 00 00 00 00 00 0B 00 00 00 00 00 00 00 01 00 00 00
 00 00 00 00  I 00 00 00 00 00 00 00 \( 00 00 00 00 00 00 00 D4 04 00 00 00 00
 00 00 0C 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  I 00 00 00 00 00 00 00
 10  ' 00 00 00 00 00 00 D4 04 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 01 00
 00 00 00 00 00 00  I 00 00 00 00 00 00 00 E8 03 00 00 00 00 00 00 D4 05 00 00
 00 00 00 00 0E 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00
 00 00 0D 00 00 00 00 00 00 00  d  f  t  e  s  t  _  i  n  .  d  a  t 00 00 00
 D4 05 00 00 00 00 00 00 0E 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 D4 F4
 01 00 00 00 00 00 00 02 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00  s 00 00 00
 00 00 00 00 0E 00 00 00 00 00 00 00 D4 1C 00 00 00 00 00 00 03 00 00 00 00 00
 00 00 0A 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 00 D4 EC 01 00 00 00 00 00
 02 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 01 00
 00 00 00 00 00 00  . 00 00 00 00 00 00 00  s 00 00 00 00 00 00 00 0E 00 00 00
 00 00 00 00  S 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00  S 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  .  d  a  t 00 00 00 00
 D4 05 00 00 00 00 00 00 0E 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 D4 14
 02 00 00 00 00 00 00 03 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 06 00 00
 00 00 00 00  s 00 00 00 00 00 00 00 0E 00 00 00 00 00 00 00  S 00 00 00 00 00
 00 00 01 00 00 00 00 00 00 00  . 00 00 00 00 00 00 00 D4 F4 01 00 00 00 00 00
 02 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 00 D4 1C 00 00 00 00 00 00 03 00
 00 00 00 00 00 00 03 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00  T DC 02 00
 00 00 00 00 D4 C4 01 00 00 00 00 00 01 00 00 00 00 00 00 00  T DC 02 00 00 00
 00 00  S 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  S 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  . 00 00 00 00 00 00 00 D4 1C
 00 00 00 00 00 00 03 00 00 00 00 00 00 00 0E 00 00 00 00 00 00 00 1C 00 00 00
 00 00 00 00 D4  h 00 00 00 00 00 00 02 00 00 00 00 00 00 00 03 00 00 00 00 00
 00 00  I 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF D4 04 00 00 00 00 00 00
 0F 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 D4  < 00 00 00 00 00 00 01 00
 00 00 00 00 00 00  s 00 00 00 00 00 00 00 0E 00 00 00 00 00 00 00  T 00 00 00
 00 00 00 00 02 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 0B 00 00 00 00 00
 00 00  T  8 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 06 00 00 00 00 00 00
  S 00 00 00 00 00 00 00 17 00 00 00 00 00 00 00  E  r  r  o  r __  c  r  e  a
  t  i  n  g __  f  i  l  e __  %  s 0A 00  s 00 00 00 00 00 00 00 0E 00 00 00
 00 00 00 00  T  , 00 00 00 00 00 00 D4 04 00 00 00 00 00 00 0F 00 00 00 00 00
 00 00 01 00 00 00 00 00 00 00 D4  X 00 00 00 00 00 00 01 00 00 00 00 00 00 00
  i 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00 D4 1C 00 00 00 00 00 00 03 00
```





```
  BOLD_TERM=`\e[1m'; NORMAL_TERM=`\e[0m'; HIDECURSOR_TERM=`';
  SHOWCURSOR_TERM=`'; GOTOCURSOR_TERM=`\e[%i%d;%dH'.
Reading the `/tmp/.BMDFMsrv' BM_DFM connection file...
Opening the `/tmp/.BMDFMsrv_npipe' BM_DFM named FIFO pipe...
Accessing the BM_DFM Server...
Receiving the Global FastLisp function set from the BM_DFM Server...
Linked Global function bytecode size is 64bytes.
-------------------------------------------------------------------------------
  h  { A0  ; 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80  >
  @ 00 00 00 00 00 01 00 00 00 00 00 00 00 80  { A0  ; 00 00 00 00 00  >  @ 00
 00 00 00 00 01 00 00 00 00 00 00 00
-------------------------------------------------------------------------------
*You may recompile the `BMDFMldr' with commented `#define _NOISY_MODE1_'
 to disable print of the linked Global function bytecode.
Connection with the BM_DFM Server has been established but not yet registered.
Checking whether the `dftest_prep.flp' file is already precompiled...
Reading the `dftest_prep.flp' source FastLisp file...
*** Resetting time counters (first null assignment)... ***
Modifying the FastLisp code (PATTERN No# 1)...
  (PROGN <Global_FastLisp_function_set> <FastLisp_prog>)
Checking the syntax of the source FastLisp file...
Modifying the FastLisp code (PATTERN No# 3)...
  (PROGN {(SETQ <termcap_var> <termcap_val>) }<FastLisp_prog>)
Looking for uninitialized variables/arrays in the FastLisp code...
Checking the CODE STYLE RESTRICTIONS for the BM_DFM parallel processing...
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                             *
*  Summary of the BM_DFM CODE STYLE RESTRICTIONS:             *
*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~              *
*                                                             *
*  o Variable names within the inclusive range of             *
*    [`TMP__000000000'; `TMP__999999999'] are reserved.       *
*  o `SHADOW' is the reserved name for a UDF.                 *
*  o Array names should differ from ordinary variable names.  *
*  o Every variable should be initialized before use.         *
*    The following is an example of how to copy an array:     *
*      ...                                                    *
*      (arsetq a 0 1)                                         *
*      (arsetq a 1 5)                                         *
*      (alsetq b (alindex a 2))    # instead of `(setq b a)'  *
*      ...                                                    *
*  o The <step> and <limit> values of a <for> loop should be  *
*    the integer numeric constants, function arguments or     *
*    initialized variables which are not changed inside this  *
*    <for> loop.                                              *
*  o Second argument of the booleans <or> and <and> should    *
*    not include any assignments, I/O, conditional/           *
*    iteration processing and UDF calls.                      *
*                                                             *
*  NOTE: Any conventional program can be converted by a       *
*        formal procedure to the program that is compliant    *
*        with the above mentioned code style restrictions.    *
*                                                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*You may recompile BMDFMldr module with commented `#define EXPLAIN_RULE'
 to disable print of the code style restriction rule summary.
Squeezing the nested source PROGN statements...
  Redundant nested source PROGN statements removed: 2.
Modifying the FastLisp code (PATTERN No# 5)...
  (PROGN (OUTF (PRN_STRING_FMT) (CAT "" <FastLisp_prog>)) "")
Reorganizing the FastLisp code...
Resolving data types in the FastLisp code...
Registering in the BM_DFM Server Task Connection Zone...
Forking up the message queue listener...
Listener engine has been commenced.
The Loader/Listener pair is fully attached by the BM_DFM Server:
  Loader PID=3335, Listener PID=3335, SocketN# is 0.
-------------------------------------------------------------------------------
(PROGN
  (SETQ@S MAIN:TERM_TYPE@S "vt100")
  (SETQ@I MAIN:LINES_TERM@I 24)
  (SETQ@I MAIN:COLUMNS_TERM@I 80)
  (SETQ@S MAIN:CLRSCR_TERM@S "\e[H\e[J")
  (SETQ@S MAIN:REVERSE_TERM@S "\e[7m")
  (SETQ@S MAIN:BLINK_TERM@S "\e[5m")
  (SETQ@S MAIN:BOLD_TERM@S "\e[1m")
  (SETQ@S MAIN:NORMAL_TERM@S "\e[0m")
  (SETQ@S MAIN:HIDECURSOR_TERM@S "")
  (SETQ@S MAIN:SHOWCURSOR_TERM@S "")
  (SETQ@S MAIN:GOTOCURSOR_TERM@S "\e[%i%d;%dH")
  (SETQ@I MAIN:RECORDSIZE@I 40)
  (SETQ@I MAIN:MAXRECORDSPERDATACLUSTER@I 10000)
  (SETQ@I MAIN:NUMBEROFDATACLUSTERS@I 1000)
  (SETQ@S MAIN:OUTPUTTESTFILENAME@S "dftest_in.dat")
  (SETQ@S
    MAIN:OUTPUTTESTFILENAME@S
    (CAT@J
      MAIN:OUTPUTTESTFILENAME@S
      (IF@J (AT@J "." MAIN:OUTPUTTESTFILENAME@S) "" ".dat")
    )
  )
  (SETQ@S
    MAIN:OUTPUTTESTFILENAME@S
    (STRTRAN@J
      MAIN:OUTPUTTESTFILENAME@S "."
      (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
    )
  )
  (SETQ@I MAIN:FDESCR@I (FILE_CREATE@J MAIN:OUTPUTTESTFILENAME@S))
  (SETQ@I MAIN:TMP__000000002 (==@I -1 MAIN:FDESCR@I))
  (IF@J
    MAIN:TMP__000000002
    (PROGN
      (SETQ@S
        MAIN:TMP__000000002
        (OUTF "Error creating file %s\n" MAIN:OUTPUTTESTFILENAME@S)
      )
      (EXIT)
    )
    (SETQ@I MAIN:FDESCR@I (FILE_CLOSE@J MAIN:FDESCR@I))
  )
  (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:RECORDSIZE@I 40))
  (IF@J
    MAIN:TMP__000000003@I
    (SETQ@I MAIN:RECORDSIZE@I 40)
    (SETQ@Z MAIN:TMP__000000002 NIL)
  )
  (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:MAXRECORDSPERDATACLUSTER@I 100))
  (IF@J
    MAIN:TMP__000000003@I
    (SETQ@I MAIN:MAXRECORDSPERDATACLUSTER@I 100)
    (SETQ@Z MAIN:TMP__000000002 NIL)
  )
  (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:NUMBEROFDATACLUSTERS@I 100))
  (IF@J
    MAIN:TMP__000000003@I
    (SETQ@I MAIN:NUMBEROFDATACLUSTERS@I 100)
    (SETQ@Z MAIN:TMP__000000002 NIL)
  )
  (SETQ@I MAIN:TMP__000000001 (IRND@J -1))
  (SETQ@I MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:FDESCR@I)
  (FOR@J
    MAIN:SEQUENCEID@I 1 1 MAIN:NUMBEROFDATACLUSTERS@I
    (PROGN
      (SETQ@S
        MAIN:TMP__000000001
        (OUTF "Writing data to cluster %ld\n" MAIN:SEQUENCEID@I)
      )
      (SETQ@I
        MAIN:RECORDSPERDATACLUSTER@I
        (IRND@J MAIN:MAXRECORDSPERDATACLUSTER@I)
      )
      (SETQ@I
        MAIN:DATACLUSTERSIZE@I
        (*@J MAIN:RECORDSIZE@I (+@J MAIN:RECORDSPERDATACLUSTER@I 2))
      )
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (ASYNCHEAP_CREATE@J MAIN:DATACLUSTERSIZE@I)
      )
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (WRITE_DATA_TO_CLUSTER@J
          MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:SEQUENCEID@I
          MAIN:RECORDSPERDATACLUSTER@I MAIN:RECORDSIZE@I
        )
      )
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (WRITE_CLUSTER_TO_FILE@J
          MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:DATACLUSTERSIZE@I
          MAIN:OUTPUTTESTFILENAME@S MAIN:OUTPUTTESTFILEORDERING_SYNC@I
        )
      )
      (SETQ@I
        MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:DATACLUSTERINMEMORY_PTR@I
      )
      (SETQ@I
        MAIN:TMP__000000001
        (ASYNCHEAP_DELETE@J MAIN:DATACLUSTERINMEMORY_PTR@I)
      )
    )
  )
  (SETQ@S
    MAIN:TMP__000000001
    (OUTF
      (PRN_STRING_FMT)
      (CAT@J "" (SPACE@J (&@J 0 MAIN:OUTPUTTESTFILEORDERING_SYNC@I)))
    )
  )
  (SETQ@S MAIN:TMP__000000000@S "")
)
-------------------------------------------------------------------------------
(PROGN (SETQ@S MAIN:TERM_TYPE@S "vt100") (SETQ@I MAIN:LINES_TERM@I 24) (SETQ@I
MAIN:COLUMNS_TERM@I 80) (SETQ@S MAIN:CLRSCR_TERM@S "\e[H\e[J") (SETQ@S MAIN:REV
ERSE_TERM@S "\e[7m") (SETQ@S MAIN:BLINK_TERM@S "\e[5m") (SETQ@S MAIN:BOLD_TERM@
S "\e[1m") (SETQ@S MAIN:NORMAL_TERM@S "\e[0m") (SETQ@S MAIN:HIDECURSOR_TERM@S "
") (SETQ@S MAIN:SHOWCURSOR_TERM@S "") (SETQ@S MAIN:GOTOCURSOR_TERM@S "\e[%i%d;%
dH") (SETQ@I MAIN:RECORDSIZE@I 40) (SETQ@I MAIN:MAXRECORDSPERDATACLUSTER@I 1000
0) (SETQ@I MAIN:NUMBEROFDATACLUSTERS@I 1000) (SETQ@S MAIN:OUTPUTTESTFILENAME@S
"dftest_in.dat") (SETQ@S MAIN:OUTPUTTESTFILENAME@S (CAT@J MAIN:OUTPUTTESTFILENA
ME@S (IF@J (AT@J "." MAIN:OUTPUTTESTFILENAME@S) "" ".dat"))) (SETQ@S MAIN:OUTPU
TTESTFILENAME@S (STRTRAN@J MAIN:OUTPUTTESTFILENAME@S "." (CAT@J (IF@J (ID_TASKJ
OB) (STR@I (ID_TASKJOB)) "") "."))) (SETQ@I MAIN:FDESCR@I (FILE_CREATE@J MAIN:O
UTPUTTESTFILENAME@S)) (SETQ@I MAIN:TMP__000000002 (==@I -1 MAIN:FDESCR@I)) (IF@
J MAIN:TMP__000000002 (PROGN (SETQ@S MAIN:TMP__000000002 (OUTF "Error creating
file %s\n" MAIN:OUTPUTTESTFILENAME@S)) (EXIT)) (SETQ@I MAIN:FDESCR@I (FILE_CLOS
E@J MAIN:FDESCR@I))) (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:RECORDSIZE@I 40))
(IF@J MAIN:TMP__000000003@I (SETQ@I MAIN:RECORDSIZE@I 40) (SETQ@Z MAIN:TMP__000
000002 NIL)) (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:MAXRECORDSPERDATACLUSTER@I
 100)) (IF@J MAIN:TMP__000000003@I (SETQ@I MAIN:MAXRECORDSPERDATACLUSTER@I 100)
 (SETQ@Z MAIN:TMP__000000002 NIL)) (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:NUMB
EROFDATACLUSTERS@I 100)) (IF@J MAIN:TMP__000000003@I (SETQ@I MAIN:NUMBEROFDATAC
LUSTERS@I 100) (SETQ@Z MAIN:TMP__000000002 NIL)) (SETQ@I MAIN:TMP__000000001 (I
RND@J -1)) (SETQ@I MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:FDESCR@I) (FOR@J MAI
N:SEQUENCEID@I 1 1 MAIN:NUMBEROFDATACLUSTERS@I (PROGN (SETQ@S MAIN:TMP__0000000
01 (OUTF "Writing data to cluster %ld\n" MAIN:SEQUENCEID@I)) (SETQ@I MAIN:RECOR
DSPERDATACLUSTER@I (IRND@J MAIN:MAXRECORDSPERDATACLUSTER@I)) (SETQ@I MAIN:DATAC
LUSTERSIZE@I (*@J MAIN:RECORDSIZE@I (+@J MAIN:RECORDSPERDATACLUSTER@I 2))) (SET
Q@I MAIN:DATACLUSTERINMEMORY_PTR@I (ASYNCHEAP_CREATE@J MAIN:DATACLUSTERSIZE@I))
 (SETQ@I MAIN:DATACLUSTERINMEMORY_PTR@I (WRITE_DATA_TO_CLUSTER@J MAIN:DATACLUST
ERINMEMORY_PTR@I MAIN:SEQUENCEID@I MAIN:RECORDSPERDATACLUSTER@I MAIN:RECORDSIZE
@I)) (SETQ@I MAIN:DATACLUSTERINMEMORY_PTR@I (WRITE_CLUSTER_TO_FILE@J MAIN:DATAC
LUSTERINMEMORY_PTR@I MAIN:DATACLUSTERSIZE@I MAIN:OUTPUTTESTFILENAME@S MAIN:OUTP
UTTESTFILEORDERING_SYNC@I)) (SETQ@I MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:DAT
ACLUSTERINMEMORY_PTR@I) (SETQ@I MAIN:TMP__000000001 (ASYNCHEAP_DELETE@J MAIN:DA
TACLUSTERINMEMORY_PTR@I)))) (SETQ@S MAIN:TMP__000000001 (OUTF (PRN_STRING_FMT)
(CAT@J "" (SPACE@J (&@J 0 MAIN:OUTPUTTESTFILEORDERING_SYNC@I))))) (SETQ@S MAIN:
TMP__000000000@S ""))
-------------------------------------------------------------------------------
*You may recompile BMDFMldr module with commented `#define _NOISY_MODE_'
 to disable print of the FastLisp code.
Performing preliminary STATIC SCHEDULING (HARD_ARRAY_SYNCHRO=NO,
  EXT_IN_OUT_SYNCHRO=YES)...
Progress: *S*i*i*i*i*f
The translator module has finished the static scheduling.
The translator has returned the following exit code: 0(Success).
The following generated control sequence (so-called `BM_DFM UNICODE')
will be transferred to the BM_DFM kernel:
-------------------------------------------------------------------------------
(CTRL
  (N# 0)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 20 "MAIN:TERM_TYPE@S")
      (1 9 "MAIN:LINES_TERM@I")
      (2 3 "MAIN:COLUMNS_TERM@I")
      (3 2 "MAIN:CLRSCR_TERM@S")
```

```
  (4 17 "MAIN:REVERSE_TERM@S")
  (5 0 "MAIN:BLINK_TERM@S")
  (6 1 "MAIN:BOLD_TERM@S")
  (7 11 "MAIN:NORMAL_TERM@S")
  (8 8 "MAIN:HIDECURSOR_TERM@S")
  (9 19 "MAIN:SHOWCURSOR_TERM@S")
  (10 7 "MAIN:GOTOCURSOR_TERM@S")
  (11 15 "MAIN:RECORDSIZE@I")
  (12 10 "MAIN:MAXRECORDSPERDATACLUSTER@I")
  (13 12 "MAIN:NUMBEROFDATACLUSTERS@I")
  (14 13 "MAIN:OUTPUTTESTFILENAME@S")
  (15 13 "MAIN:OUTPUTTESTFILENAME@S")
  (16 13 "MAIN:OUTPUTTESTFILENAME@S")
)
(Fnc
  (N# 0)
  (FLP (SETQ@S MAIN:TERM_TYPE@S "vt100"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
    " v  t  1  0  0 00 00 00"
  )
  (Var_Ptrs 0)
)
(Fnc
  (N# 1)
  (FLP (SETQ@I MAIN:LINES_TERM@I 24))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " I 00 00 00 00 00 00 00" "18 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 1)
)
(Fnc
  (N# 2)
  (FLP (SETQ@I MAIN:COLUMNS_TERM@I 80))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " I 00 00 00 00 00 00 00" " P 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 2)
)
(Fnc
  (N# 3)
  (FLP (SETQ@S MAIN:CLRSCR_TERM@S "\e[H\e[J"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
    "1B  [  H 1B  [  J 00 00"
  )
  (Var_Ptrs 3)
)
(Fnc
  (N# 4)
  (FLP (SETQ@S MAIN:REVERSE_TERM@S "\e[7m"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    "1B  [  7  m 00 00 00 00"
  )
  (Var_Ptrs 4)
)
(Fnc
  (N# 5)
  (FLP (SETQ@S MAIN:BLINK_TERM@S "\e[5m"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    "1B  [  5  m 00 00 00 00"
  )
  (Var_Ptrs 5)
)
(Fnc
  (N# 6)
  (FLP (SETQ@S MAIN:BOLD_TERM@S "\e[1m"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    "1B  [  1  m 00 00 00 00"
  )
  (Var_Ptrs 6)
)
(Fnc
  (N# 7)
  (FLP (SETQ@S MAIN:NORMAL_TERM@S "\e[0m"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    "1B  [  0  m 00 00 00 00"
  )
  (Var_Ptrs 7)
)
(Fnc
  (N# 8)
  (FLP (SETQ@S MAIN:HIDECURSOR_TERM@S ""))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 8)
)
(Fnc
  (N# 9)
  (FLP (SETQ@S MAIN:SHOWCURSOR_TERM@S ""))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 9)
)
(Fnc
  (N# 10)
  (FLP (SETQ@S MAIN:GOTOCURSOR_TERM@S "\e[%i%d;%dH"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "0A 00 00 00 00 00 00 00"
    "1B  [  %  i  %  d  ;  %" " d  H 00 00 00 00 00 00"
  )
  (Var_Ptrs 10)
)
(Fnc
  (N# 11)
  (FLP (SETQ@I MAIN:RECORDSIZE@I 40))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " I 00 00 00 00 00 00 00" "\( 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 11)
)
(Fnc
  (N# 12)
  (FLP (SETQ@I MAIN:MAXRECORDSPERDATACLUSTER@I 10000))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " I 00 00 00 00 00 00 00" "10  ' 00 00 00 00 00 00"
  )
  (Var_Ptrs 12)
)
(Fnc
  (N# 13)
  (FLP (SETQ@I MAIN:NUMBEROFDATACLUSTERS@I 1000))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " I 00 00 00 00 00 00 00" "E8 03 00 00 00 00 00 00"
  )
  (Var_Ptrs 13)
)
(Fnc
  (N# 14)
  (FLP (SETQ@S MAIN:OUTPUTTESTFILENAME@S "dftest_in.dat"))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "0D 00 00 00 00 00 00 00"
    " d  f  t  e  s  t  _  i" " n  .  d  a  t 00 00 00"
  )
  (Var_Ptrs 14)
)
(Fnc
  (N# 15)
  (FLP
    (SETQ@S
      MAIN:OUTPUTTESTFILENAME@S
      (CAT@J
        MAIN:OUTPUTTESTFILENAME@S
        (IF@J (AT@J "." MAIN:OUTPUTTESTFILENAME@S) "" ".dat")
      )
    )
  )
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "D4 F4 01 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    "03 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
    "01 00 00 00 00 00 00 00" "D4 1C 00 00 00 00 00 00"
    "03 00 00 00 00 00 00 00" "0A 00 00 00 00 00 00 00"
    "0C 00 00 00 00 00 00 00" "D4 EC 01 00 00 00 00 00"
    "02 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " . 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
    "01 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    " .  d  a  t 00 00 00 00"
  )
  (Var_Ptrs 15 14)
)
(Fnc
  (N# 16)
  (FLP
    (SETQ@S
      MAIN:OUTPUTTESTFILENAME@S
      (STRTRAN@J
        MAIN:OUTPUTTESTFILENAME@S "."
        (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
      )
    )
  )
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "D4 14 02 00 00 00 00 00" "03 00 00 00 00 00 00 00"
    "04 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
    " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " S 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " . 00 00 00 00 00 00 00" "D4 F4 01 00 00 00 00 00"
    "02 00 00 00 00 00 00 00" "0C 00 00 00 00 00 00 00"
```

```
        "D4 1C 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
        " T DC 02 00 00 00 00 00" "D4 C4 01 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " T DC 02 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " . 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 16 15)
    )
  )
)
(CTRL
  (N# 1)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata
    (VarRef 13)
    (VarName "MAIN:OUTPUTTESTFILENAME@S")
    (Inq_Dest Ld)
  )
)
(CTRL (N# 2) (OpGroup 1) (COP 83) (<accum_chr> (dfmget_sdata)))
(CTRL (N# 3) (OpGroup 3) (COP 22) (<accum_slo> (FILE_CREATE <accum_chr>)))
(CTRL
  (N# 4)
  (OpGroup 1)
  (COP 71)
  (dfmput_idata <accum_slo> (VarRef 6) (VarName "MAIN:FDESCR@I"))
)
(CTRL
  (N# 5)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 6 "MAIN:FDESCR@I")
      (1 23 "MAIN:TMP__000000002")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:TMP__000000002 (==@I -1 MAIN:FDESCR@I)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  h 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "FF FF FF FF FF FF FF FF" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL
  (N# 6)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 23) (VarName "MAIN:TMP__000000002") (Inq_Dest Ld))
)
(CTRL (N# 7) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL
  (N# 8)
  (OpGroup 2)
  (COP 17)
  (IF_NOT <accum_slo> (GOTO 12))
  (REM "Pass over `MAIN:TMP__000000002' <if> conditional branch")
)
(CTRL
  (N# 9)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 13 "MAIN:OUTPUTTESTFILENAME@S")
      (1 23 "MAIN:TMP__000000002")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@S
          MAIN:TMP__000000002
          (OUTF "Error creating file %s\n" MAIN:OUTPUTTESTFILENAME@S)
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " T  8 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "06 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "17 00 00 00 00 00 00 00" " E  r  r  o  r __  c  r"
        " e  a  t  i  n  g __  f" " i  l  e __  %  s 0A 00"
        " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      )
      (Inq_Dest Ls)
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL (N# 10) (OpGroup 2) (COP 14) (GOTO 50) (REM "EXIT"))
(CTRL
  (N# 11)
  (OpGroup 2)
  (COP 14)
  (GOTO 16)
  (REM "Pass over `MAIN:TMP__000000002' <else> conditional branch")
)
(CTRL
  (N# 12)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 6) (VarName "MAIN:FDESCR@I") (Inq_Dest Ld))
)
(CTRL (N# 13) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL (N# 14) (OpGroup 3) (COP 29) (<accum_slo> (FILE_CLOSE <accum_slo>)))
(CTRL
  (N# 15)
  (OpGroup 1)
  (COP 71)
  (dfmput_idata <accum_slo> (VarRef 6) (VarName "MAIN:FDESCR@I"))
)
(CTRL
  (N# 16)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 15 "MAIN:RECORDSIZE@I")
      (1 24 "MAIN:TMP__000000003@I")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:RECORDSIZE@I 40)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  x 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "\( 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL
  (N# 17)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 24) (VarName "MAIN:TMP__000000003@I") (Inq_Dest Ld))
)
(CTRL (N# 18) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL
  (N# 19)
  (OpGroup 2)
  (COP 17)
  (IF_NOT <accum_slo> (GOTO 22))
  (REM "Pass over `MAIN:TMP__000000003@I' <if> conditional branch")
)
(CTRL
  (N# 20)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 15 "MAIN:RECORDSIZE@I"))
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:RECORDSIZE@I 40))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" "\( 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 21)
  (OpGroup 2)
  (COP 14)
  (GOTO 23)
  (REM "Pass over `MAIN:TMP__000000003@I' <else> conditional branch")
)
(CTRL
  (N# 22)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 23 "MAIN:TMP__000000002"))
    (Fnc
      (N# 0)
      (FLP (SETQ@Z MAIN:TMP__000000002 NIL))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " T 06 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " Z 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 23)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 10 "MAIN:MAXRECORDSPERDATACLUSTER@I")
      (1 24 "MAIN:TMP__000000003@I")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@I
          MAIN:TMP__000000003@I
          (<@I MAIN:MAXRECORDSPERDATACLUSTER@I 100)
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  x 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        " d 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL
  (N# 24)
  (OpGroup 1)
```

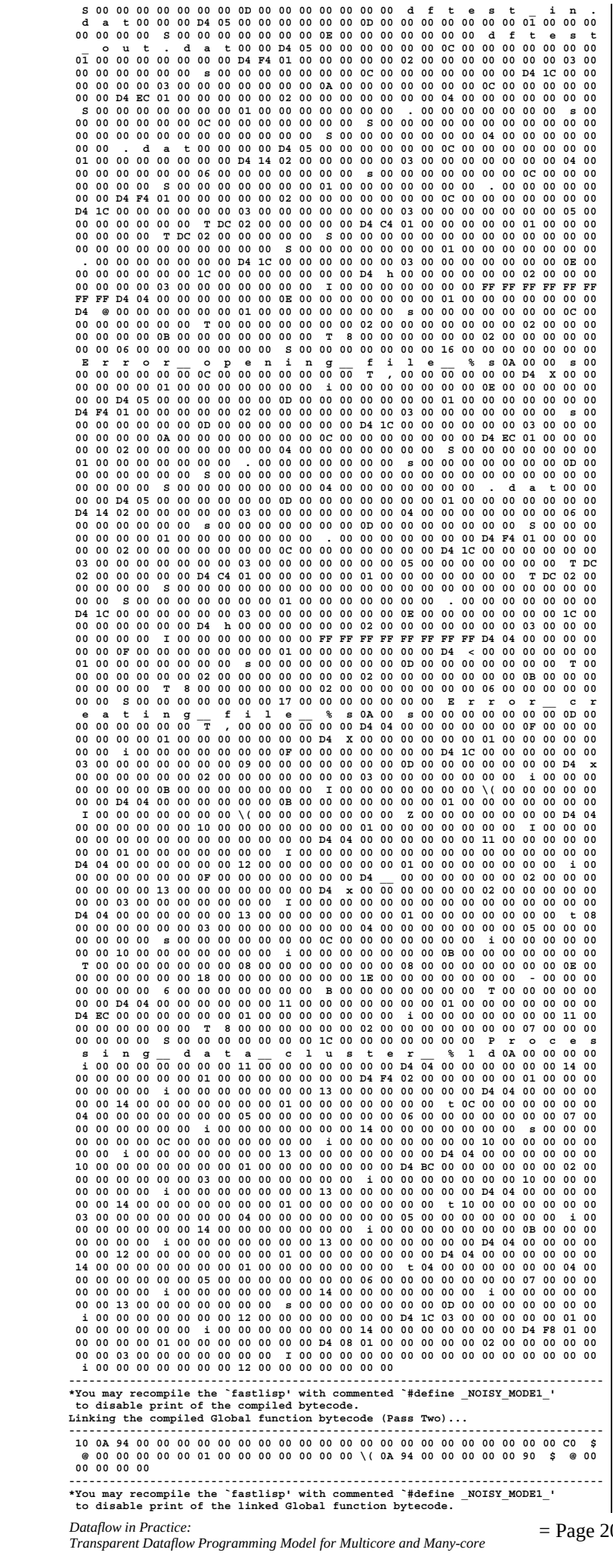
```
  (COP 70)
  (dfmput_zdata (VarRef 24) (VarName "MAIN:TMP__000000003@I") (Inq_Dest Ld))
)
(CTRL (N# 25) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL
  (N# 26)
  (OpGroup 2)
  (COP 17)
  (IF_NOT <accum_slo> (GOTO 29))
  (REM "Pass over `MAIN:TMP__000000003@I' <if> conditional branch")
)
(CTRL
  (N# 27)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 10 "MAIN:MAXRECORDSPERDATACLUSTER@I"))
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:MAXRECORDSPERDATACLUSTER@I 100))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" " d 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 28)
  (OpGroup 2)
  (COP 14)
  (GOTO 30)
  (REM "Pass over `MAIN:TMP__000000003@I' <else> conditional branch")
)
(CTRL
  (N# 29)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 23 "MAIN:TMP__000000002"))
    (Fnc
      (N# 0)
      (FLP (SETQ@Z MAIN:TMP__000000002 NIL))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " T 06 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " Z 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 30)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 12 "MAIN:NUMBEROFDATACLUSTERS@I")
      (1 24 "MAIN:TMP__000000003@I")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:NUMBEROFDATACLUSTERS@I 100))
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  x 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        " d 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL
  (N# 31)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 24) (VarName "MAIN:TMP__000000003@I") (Inq_Dest Ld))
)
(CTRL (N# 32) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL
  (N# 33)
  (OpGroup 2)
  (COP 17)
  (IF_NOT <accum_slo> (GOTO 36))
  (REM "Pass over `MAIN:TMP__000000003@I' <if> conditional branch")
)
(CTRL
  (N# 34)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 12 "MAIN:NUMBEROFDATACLUSTERS@I"))
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:NUMBEROFDATACLUSTERS@I 100))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" " d 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 35)
  (OpGroup 2)
  (COP 14)
  (GOTO 37)
  (REM "Pass over `MAIN:TMP__000000003@I' <else> conditional branch")
)
(CTRL
  (N# 36)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 23 "MAIN:TMP__000000002"))
    (Fnc
      (N# 0)
      (FLP (SETQ@Z MAIN:TMP__000000002 NIL))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " T 06 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " Z 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 37)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 22 "MAIN:TMP__000000001")
      (1 6 "MAIN:FDESCR@I")
      (2 14 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:TMP__000000001 (IRND@J -1)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 B4 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" "FF FF FF FF FF FF FF FF"
      )
      (Var_Ptrs 0)
    )
    (Fnc
      (N# 1)
      (FLP (SETQ@I MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:FDESCR@I))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 2 1)
    )
  )
)
(CTRL (N# 38) (OpGroup 2) (COP 10) (PUSHA))
(CTRL
  (N# 39)
  (OpGroup 4)
  (COP 90)
  (SubCOP 1)
  (<loop_slo> 1)
  (REM "<For> `MAIN:SEQUENCEID@I' loop initialization begins here")
)
(CTRL (N# 40) (OpGroup 4) (COP 90) (SubCOP 2) (<loopstep_slo> 1))
(CTRL
  (N# 41)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata
    (VarRef 12)
    (VarName "MAIN:NUMBEROFDATACLUSTERS@I")
    (Inq_Dest Ld)
  )
)
(CTRL (N# 42) (OpGroup 1) (COP 81) (SubCOP 3) (<loopto_slo> (dfmget_idata)))
(CTRL
  (N# 43)
  (OpGroup 4)
  (COP 100)
  (FOR <loop_slo> (STEP <loopstep_slo>) (TO <loopto_slo>) (BODY 47))
  (REM "Controlled by `MAIN:SEQUENCEID@I' variable")
)
(CTRL
  (N# 44)
  (OpGroup 1)
  (COP 71)
  (SubCOP 1)
  (dfmput_idata <loop_slo> (VarRef 18) (VarName "MAIN:SEQUENCEID@I"))
)
(CTRL
  (N# 45)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 18 "MAIN:SEQUENCEID@I")
      (1 22 "MAIN:TMP__000000001")
      (2 10 "MAIN:MAXRECORDSPERDATACLUSTER@I")
      (3 16 "MAIN:RECORDSPERDATACLUSTER@I")
      (4 15 "MAIN:RECORDSIZE@I")
      (5 5 "MAIN:DATACLUSTERSIZE@I")
      (6 4 "MAIN:DATACLUSTERINMEMORY_PTR@I")
      (7 4 "MAIN:DATACLUSTERINMEMORY_PTR@I")
      (8 13 "MAIN:OUTPUTTESTFILENAME@S")
      (9 14 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
      (10 4 "MAIN:DATACLUSTERINMEMORY_PTR@I")
      (11 14 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
      (12 22 "MAIN:TMP__000000001")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@S
          MAIN:TMP__000000001
          (OUTF "Writing data to cluster %ld\n" MAIN:SEQUENCEID@I)
        )
      )
```

```
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      " T  8 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      "07 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
      "1C 00 00 00 00 00 00 00" " W  r  i  t  i  n  g  __"
      " d  a  t  a  __  t  o  __" " c  l  u  s  t  e  r  __"
      " %  l  d 0A 00 00 00 00" " i 00 00 00 00 00 00 00"
      "01 00 00 00 00 00 00 00"
    )
    (Inq_Dest Ls)
    (Var_Ptrs 1 0)
  )
  (Fnc
    (N# 1)
    (FLP
      (SETQ@I
        MAIN:RECORDSPERDATACLUSTER@I
        (IRND@J MAIN:MAXRECORDSPERDATACLUSTER@I)
      )
    )
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      "D4 B4 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    )
    (Var_Ptrs 3 2)
  )
  (Fnc
    (N# 2)
    (FLP
      (SETQ@I
        MAIN:DATACLUSTERSIZE@I
        (*@J MAIN:RECORDSIZE@I (+@J MAIN:RECORDSPERDATACLUSTER@I 2))
      )
    )
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      "D4 CC 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "01 00 00 00 00 00 00 00" "D4 BC 00 00 00 00 00 00"
      "02 00 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
      " i 00 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      " I 00 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    )
    (Var_Ptrs 5 4 3)
  )
  (Fnc
    (N# 3)
    (FLP
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (ASYNCHEAP_CREATE@J MAIN:DATACLUSTERSIZE@I)
      )
    )
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      "D4 F4 02 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    )
    (Var_Ptrs 6 5)
  )
  (Fnc
    (N# 4)
    (FLP
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (WRITE_DATA_TO_CLUSTER@J
          MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:SEQUENCEID@I
          MAIN:RECORDSPERDATACLUSTER@I MAIN:RECORDSIZE@I
        )
      )
    )
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      " t 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
      "05 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
      "07 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "01 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "02 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "04 00 00 00 00 00 00 00"
    )
    (Var_Ptrs 7 6 0 3 4)
  )
  (Fnc
    (N# 5)
    (FLP
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (WRITE_CLUSTER_TO_FILE@J
          MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:DATACLUSTERSIZE@I
          MAIN:OUTPUTTESTFILENAME@S MAIN:OUTPUTTESTFILEORDERING_SYNC@I
        )
      )
    )
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      " t 04 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
      "05 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
      "07 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "01 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "02 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
      "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
      "04 00 00 00 00 00 00 00"
    )
    (Var_Ptrs 10 7 5 8 9)
  )
  (Fnc
    (N# 6)
    (FLP
      (SETQ@I
        MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:DATACLUSTERINMEMORY_PTR@I
      )
    )
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    )
    (Var_Ptrs 11 10)
  )
  (Fnc
    (N# 7)
    (FLP
      (SETQ@I
        MAIN:TMP__000000001
        (ASYNCHEAP_DELETE@J MAIN:DATACLUSTERINMEMORY_PTR@I)
      )
    )
    (FLP_COMPILED
      "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
      "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      "D4 1C 03 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    )
    (Var_Ptrs 12 10)
  )
 )
)
(CTRL
  (N# 46)
  (OpGroup 4)
  (COP 101)
  (SubCOP 1)
  (NEXT (BODY 43))
  (REM "Controlled by `MAIN:SEQUENCEID@I' variable")
)
(CTRL
  (N# 47)
  (OpGroup 1)
  (COP 71)
  (SubCOP 1)
  (dfmput_idata <loop_slo> (VarRef 18) (VarName "MAIN:SEQUENCEID@I"))
  (REM "<For> postloop `MAIN:SEQUENCEID@I' control variable value")
)
(CTRL (N# 48) (OpGroup 2) (COP 11) (POPA))
(CTRL
  (N# 49)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 14 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
      (1 22 "MAIN:TMP__000000001")
      (2 21 "MAIN:TMP__000000000@S")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@S
          MAIN:TMP__000000001
          (OUTF
            (PRN_STRING_FMT)
            (CAT@J "" (SPACE@J (&@J 0 MAIN:OUTPUTTESTFILEORDERING_SYNC@I)))
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " T  8 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "02 00 00 00 00 00 00 00" " T 80 02 00 00 00 00 00"
        "D4 F4 01 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "04 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "D4 F8 01 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 08 01 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00"
      )
      (Inq_Dest Ls)
      (Var_Ptrs 1 0)
    )
    (Fnc
      (N# 1)
      (FLP (SETQ@S MAIN:TMP__000000000@S ""))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 2)
    )
  )
)
(CTRL (N# 50) (OpGroup 4) (COP 200) (END) (REM "End of the control sequence"))
--------------------------------------------------------------------------------
*You may recompile BMDFMldr module with commented `#define _NOISY_MODE1_'
 to disable print of the BM_DFM control sequence.
*** Uploading and immediate running of the BM_DFM control sequence by
    the BM_DFM kernel will start here just after the time report!
Time spent to check and prepare the task approx.:
   Used by process: 0.020996sec.
   Used by  system: 0.004000sec.
   Total used time: 2.499600000000E-02sec.
Real absolute time: 2.509698736498E-02sec.
*** Resetting time counters (second event controlpoint)... ***
================================================================================
The task is being carried out on SocketN# 0.
================================================================================
Writing data to cluster 1
Writing data to cluster 2
```

```
Writing data to cluster 3
Writing data to cluster 4
Writing data to cluster 5
Writing data to cluster 6
Writing data to cluster 7
Writing data to cluster 8
Writing data to cluster 9
Writing data to cluster 10

                                    . . .

Writing data to cluster 991
Writing data to cluster 992
Writing data to cluster 993
Writing data to cluster 994
Writing data to cluster 995
Writing data to cluster 996
Writing data to cluster 997
Writing data to cluster 998
Writing data to cluster 999
Writing data to cluster 1000

================================================================================
Time spent to run the task (by PARENT loader and CHILD listener):
   Used by process: 0.015997sec.
   Used by  system: 0.183972sec.
   Total used time: 1.999690000000E-01sec.
Real absolute time: 2.009876496598E-01sec.
Task has been detached (logged out) from the BM_DFM Server.
The BM_DFM Task Loader/Listener pair has done its job decently and gracefully.
```

dftest_exec.fastlisp

```
`Cursor invisible (vi)' capability was not found, default value is used!
`Cursor visible (ve)' capability was not found, default value is used!
Current termcap settings:
  TERM_TYPE=`vt100'; LINES_TERM=`24'; COLUMNS_TERM=`80';
  CLRSCR_TERM=`\e[H\e[J'; REVERSE_TERM=`\e[7m'; BLINK_TERM=`\e[5m';
  BOLD_TERM=`\e[1m'; NORMAL_TERM=`\e[0m'; HIDECURSOR_TERM=`';
  SHOWCURSOR_TERM=`'; GOTOCURSOR_TERM=`\e[%i%d;%dH'.
Checking whether the `dftest_exec.flp' file is already precompiled...
Reading the `fastlisp.cfg' configuration profile...
Checking the syntax of the configuration profile...
Squeezing the nested source PROGN statements in Global FastLisp function set...
  Redundant nested source PROGN statements removed: 0.
Looking for uninitialized variables/arrays in Global FastLisp function set...
Resolving data types in Global FastLisp function set...
Reading the `dftest_exec.flp' source FastLisp file...
*** Resetting time counters (first null assignment)... ***
Modifying the FastLisp code (PATTERN No# 1)...
  (PROGN <FastLisp_prog>)
Checking the syntax of the source FastLisp file...
Modifying the FastLisp code (PATTERN No# 2)...
  (PROGN {(SETQ <termcap_var> <termcap_val>) }<FastLisp_prog>)
Squeezing the nested source PROGN statements...
  Redundant nested source PROGN statements removed: 2.
Looking for uninitialized variables/arrays in the FastLisp code...
Resolving data types in the FastLisp code...
--------------------------------------------------------------------------------
(PROGN
  (SETQ@S TERM_TYPE@S "vt100")
  (SETQ@I LINES_TERM@I 24)
  (SETQ@I COLUMNS_TERM@I 80)
  (SETQ@S CLRSCR_TERM@S "\e[H\e[J")
  (SETQ@S REVERSE_TERM@S "\e[7m")
  (SETQ@S BLINK_TERM@S "\e[5m")
  (SETQ@S BOLD_TERM@S "\e[1m")
  (SETQ@S NORMAL_TERM@S "\e[0m")
  (SETQ@S HIDECURSOR_TERM@S "")
  (SETQ@S SHOWCURSOR_TERM@S "")
  (SETQ@S GOTOCURSOR_TERM@S "\e[%i%d;%dH")
  (SETQ@I RECORDSIZE@I 40)
  (SETQ@S INPUTTESTFILENAME@S "dftest_in.dat")
  (SETQ@S OUTPUTTESTFILENAME@S "dftest_out.dat")
  (SETQ@S
    INPUTTESTFILENAME@S
    (CAT@J INPUTTESTFILENAME@S (IF@J (AT@J "." INPUTTESTFILENAME@S) "" ".dat"))
  )
  (SETQ@S
    INPUTTESTFILENAME@S
    (STRTRAN@J
      INPUTTESTFILENAME@S "."
      (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
    )
  )
  (IF@J
    (==@I -1 (SETQ@I FDESCR_@I (FILE_OPEN@J INPUTTESTFILENAME@S)))
    (PROGN (OUTF "Error opening file %s\n" INPUTTESTFILENAME@S) (EXIT))
    (FILE_CLOSE@J FDESCR_@I)
  )
  (SETQ@S
    OUTPUTTESTFILENAME@S
    (CAT@J
      OUTPUTTESTFILENAME@S
      (IF@J (AT@J "." OUTPUTTESTFILENAME@S) "" ".dat")
    )
  )
  (SETQ@S
    OUTPUTTESTFILENAME@S
    (STRTRAN@J
      OUTPUTTESTFILENAME@S "."
      (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
    )
  )
  (IF@J
    (==@I -1 (SETQ@I FDESCR@I (FILE_CREATE@J OUTPUTTESTFILENAME@S)))
    (PROGN (OUTF "Error creating file %s\n" OUTPUTTESTFILENAME@S) (EXIT))
    (SETQ@I FDESCR@I (FILE_CLOSE@J FDESCR@I))
  )
  (IF@J (<@I RECORDSIZE@I 40) (SETQ@I RECORDSIZE@I 40) NIL)
  (SETQ@I INPUTFILEPOSITION@I 0)
  (SETQ@I SEQUENCEID@I 0)
  (SETQ@I OUTPUTTESTFILEORDERING_SYNC@I FDESCR@I)
  (WHILE@J
    (<@I
      0
      (SETQ@I
        DATACLUSTERSIZE@I
        (GET_NEXT_CLUSTER_SIZE@J
          INPUTTESTFILENAME@S INPUTFILEPOSITION@I RECORDSIZE@I
        )
      )
    )
    (PROGN
      (SETQ@I SEQUENCEID@I (++@J SEQUENCEID@I))
      (OUTF "Processing data cluster %ld\n" SEQUENCEID@I)
      (SETQ@I DATACLUSTERINMEMORY_PTR@I (ASYNCHEAP_CREATE@J DATACLUSTERSIZE@I))
      (SETQ@I
        DATACLUSTERINMEMORY_PTR@I
        (READ_DATA_TO_CLUSTER@J
          DATACLUSTERINMEMORY_PTR@I INPUTTESTFILENAME@S INPUTFILEPOSITION@I
          DATACLUSTERSIZE@I
        )
      )
      (SETQ@I INPUTFILEPOSITION@I (+@J INPUTFILEPOSITION@I DATACLUSTERSIZE@I))
      (SETQ@I
        DATACLUSTERINMEMORY_PTR@I
        (PROCESS_DATA_CLUSTER@J
          DATACLUSTERINMEMORY_PTR@I RECORDSIZE@I DATACLUSTERSIZE@I
        )
      )
      (SETQ@I
        OUTPUTTESTFILEORDERING_SYNC@I
        (SETQ@I
          DATACLUSTERINMEMORY_PTR@I
          (WRITE_CLUSTER_TO_FILE@J
            DATACLUSTERINMEMORY_PTR@I DATACLUSTERSIZE@I OUTPUTTESTFILENAME@S
            OUTPUTTESTFILEORDERING_SYNC@I
          )
        )
      )
      (ASYNCHEAP_DELETE@J DATACLUSTERINMEMORY_PTR@I)
    )
  )
  (SPACE@J (&@J 0 OUTPUTTESTFILEORDERING_SYNC@I))
)
--------------------------------------------------------------------------------
(PROGN (SETQ@S TERM_TYPE@S "vt100") (SETQ@I LINES_TERM@I 24) (SETQ@I COLUMNS_TERM@I 80) (SETQ@S CLRSCR_TERM@S "\e[H\e[J") (SETQ@S REVERSE_TERM@S "\e[7m") (SETQ@S BLINK_TERM@S "\e[5m") (SETQ@S BOLD_TERM@S "\e[1m") (SETQ@S NORMAL_TERM@S "\e[0m") (SETQ@S HIDECURSOR_TERM@S "") (SETQ@S SHOWCURSOR_TERM@S "") (SETQ@S GOTOCURSOR_TERM@S "\e[%i%d;%dH") (SETQ@I RECORDSIZE@I 40) (SETQ@S INPUTTESTFILENAME@S "dftest_in.dat") (SETQ@S OUTPUTTESTFILENAME@S "dftest_out.dat") (SETQ@S INPUTTESTFILENAME@S (CAT@J INPUTTESTFILENAME@S (IF@J (AT@J "." INPUTTESTFILENAME@S) "" ".dat"))) (SETQ@S INPUTTESTFILENAME@S (STRTRAN@J INPUTTESTFILENAME@S "." (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") "."))) (IF@J (==@I -1 (SETQ@I FDESCR_@I (FILE_OPEN@J INPUTTESTFILENAME@S))) (PROGN (OUTF "Error opening file %s\n" INPUTTESTFILENAME@S) (EXIT)) (FILE_CLOSE@J FDESCR_@I)) (SETQ@S OUTPUTTESTFILENAME@S (CAT@J OUTPUTTESTFILENAME@S (IF@J (AT@J "." OUTPUTTESTFILENAME@S) "" ".dat"))) (SETQ@S OUTPUTTESTFILENAME@S (STRTRAN@J OUTPUTTESTFILENAME@S "." (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") "."))) (IF@J (==@I -1 (SETQ@I FDESCR@I (FILE_CREATE@J OUTPUTTESTFILENAME@S))) (PROGN (OUTF "Error creating file %s\n" OUTPUTTESTFILENAME@S) (EXIT)) (SETQ@I FDESCR@I (FILE_CLOSE@J FDESCR@I))) (IF@J (<@I RECORDSIZE@I 40) (SETQ@I RECORDSIZE@I 40) NIL) (SETQ@I INPUTFILEPOSITION@I 0) (SETQ@I SEQUENCEID@I 0) (SETQ@I OUTPUTTESTFILEORDERING_SYNC@I FDESCR@I) (WHILE@J (<@I 0 (SETQ@I DATACLUSTERSIZE@I (GET_NEXT_CLUSTER_SIZE@J INPUTTESTFILENAME@S INPUTFILEPOSITION@I RECORDSIZE@I))) (PROGN (SETQ@I SEQUENCEID@I (+@J SEQUENCEID@I)) (OUTF "Processing data cluster %ld\n" SEQUENCEID@I) (SETQ@I DATACLUSTERINMEMORY_PTR@I (ASYNCHEAP_CREATE@J DATACLUSTERSIZE@I)) (SETQ@I DATACLUSTERINMEMORY_PTR@I (READ_DATA_TO_CLUSTER@J DATACLUSTERINMEMORY_PTR@I INPUTTESTFILENAME@S INPUTFILEPOSITION@I DATACLUSTERSIZE@I)) (SETQ@I INPUTFILEPOSITION@I (+@J INPUTFILEPOSITION@I DATACLUSTERSIZE@I)) (SETQ@I DATACLUSTERINMEMORY_PTR@I (PROCESS_DATA_CLUSTER@J DATACLUSTERINMEMORY_PTR@I RECORDSIZE@I DATACLUSTERSIZE@I)) (SETQ@I OUTPUTTESTFILEORDERING_SYNC@I (SETQ@I DATACLUSTERINMEMORY_PTR@I (WRITE_CLUSTER_TO_FILE@J DATACLUSTERINMEMORY_PTR@I DATACLUSTERSIZE@I OUTPUTTESTFILENAME@S OUTPUTTESTFILEORDERING_SYNC@I))) (ASYNCHEAP_DELETE@J DATACLUSTERINMEMORY_PTR@I))) (SPACE@J (&@J 0 OUTPUTTESTFILEORDERING_SYNC@I)))
--------------------------------------------------------------------------------
*You may recompile the `fastlisp' with commented `#define _NOISY_MODE_'
 to disable print of the FastLisp code.
Compiling the Global FastLisp function source code (Pass One)...
Compiled Global function bytecode size is 56bytes.
--------------------------------------------------------------------------------
 D5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  T 00
 00 00 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00  Z 00 00 00
 00 00 00 00
--------------------------------------------------------------------------------
*You may recompile the `fastlisp' with commented `#define _NOISY_MODE1_'
 to disable print of the compiled Global function bytecode.
Compiling the FastLisp source code (Pass One)...
Compiled bytecode size is 3656bytes.
--------------------------------------------------------------------------------
 D5 01 00 00 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  T 00
 00 00 00 00 00 00 1A 00 00 00 00 00 00 00 1A 00 00 00 00 00 00 00 1F 00 00 00
 00 00 00 00  # 00 00 00 00 00 00 00  ' 00 00 00 00 00 00 00  , 00 00 00 00 00
 00 00  1 00 00 00 00 00 00 00  6 00 00 00 00 00 00 00  ; 00 00 00 00 00 00 00
  @ 00 00 00 00 00 00 00  E 00 00 00 00 00 00 00  J 00 00 00 00 00 00 00  P 00
 00 00 00 00 00 00  T 00 00 00 00 00 00 00  Z 00 00 00 00 00 00 00  ` 00 00 00
 00 00 00 00  y 00 00 00 00 00 00 00 95 00 00 00 00 00 00 00 B7 00 00 00 00 00
 00 00 D0 00 00 00 00 00 00 00 EC 00 00 00 00 00 00 00 11 01 00 00 00 00 00 00
  ! 01 00 00 00 00 00 00 00  % 01 00 00 00 00 00 00 \) 01 00 00 00 00 00 00  - 01
 00 00 00 00 00 00 A2 01 00 00 00 00 00 00 D4 05 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 05 00 00 00 00 00
 00 00  v  t  1  0  0 00 00 00 D4 04 00 00 00 00 00 00 01 00 00 00 00 00 00 00
 01 00 00 00 00 00 00 00  I 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00 D4 04
 00 00 00 00 00 00 02 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  I 00 00 00
 00 00 00 00  P 00 00 00 00 00 00 00 D4 05 00 00 00 00 00 00 03 00 00 00 00 00
 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00
 1B  [  H 1B  [  J 00 00 D4 05 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 01 00
 00 00 00 00 00  S 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 1B  [  7  m
 00 00 00 00 D4 05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00 01 00 00 00 00 00
 00 00  S 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 1B  [  5  m 00 00 00 00 00
 D4 05 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  S 00
 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 1B  [  1  m 00 00 00 00 D4 05 00 00
 00 00 00 00 07 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00
 00 00 04 00 00 00 00 00 00 00 1B  [  0  m 00 00 00 00 D4 05 00 00 00 00 00 00
 08 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D4 05 00 00 00 00 00 00 09 00 00 00
 00 00 00 00 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 D4 05 00 00 00 00 00 00 0A 00 00 00 00 00 00
 01 00 00 00 00 00 00 00  S 00 00 00 00 00 00 00 0A 00 00 00 00 00 00 00 1B  [
  %  i  %  d  ;  %  d  H 00 00 00 00 00 00 00 D4 04 00 00 00 00 00 00 0B 00 00
 00 00 00 00 01 00 00 00 00 00 00 00  I 00 00 00 00 00 00 00 \( 00 00 00 00 00
 00 00 D4 05 00 00 00 00 00 00 0C 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
```


*You may recompile the `fastlisp' with commented `#define _NOISY_MODE1_'
to disable print of the compiled bytecode.
Linking the compiled Global function bytecode (Pass Two)...
*You may recompile the `fastlisp' with commented `#define _NOISY_MODE1_'
to disable print of the linked Global function bytecode.
Linking the compiled bytecode (Pass Two)...

```
 00 00 00 00  p AA  @ 00 00 00 00 00 13 00 00 00 00 00 00 00  0  m  @ 00 00 00
 00 00 14 00 00 00 00 00 00 00 B0  H 94 00 00 00 00 00 10 0A  B 00 00 00 00 00
 D8  H 94 00 00 00 00 00 E8  H 94 00 00 00 00 00 F8  H 94 00 00 00 00 00 08  I
 94 00 00 00 00 00  p AA  @ 00 00 00 00 00 14 00 00 00 00 00 00 00 80 A8  @ 00
 00 00 00 00 0C 00 00 00 00 00 00 00  p AA  @ 00 00 00 00 00 10 00 00 00 00 00
 00 00  p AA  @ 00 00 00 00 00 13 00 00 00 00 00 00 00  0  m  @ 00 00 00 00 00
 10 00 00 00 00 00 00 00  0  I 94 00 00 00 00 00 90  /  @ 00 00 00 00 00  H  I
 94 00 00 00 00 00  X  I 94 00 00 00 00 00  p AA  @ 00 00 00 00 00 10 00 00 00
 00 00 00 00  p AA  @ 00 00 00 00 00 13 00 00 00 00 00 00 00  0  m  @ 00 00 00
 00 00 14 00 00 00 00 00 00 00 80  I 94 00 00 00 00 00  p 09  B 00 00 00 00 00
 A0  I 94 00 00 00 00 00 B0  I 94 00 00 00 00 00 C0  I 94 00 00 00 00 00  p AA
  @ 00 00 00 00 00 14 00 00 00 00 00 00 00  p AA  @ 00 00 00 00 00 0B 00 00 00
 00 00 00 00  p AA  @ 00 00 00 00 00 13 00 00 00 00 00 00 00  0  m  @ 00 00 00
 00 00 12 00 00 00 00 00 00 00 E8  I 94 00 00 00 00 00  0  m  @ 00 00 00 00 00
 14 00 00 00 00 00 00 00 00  J 94 00 00 00 00 00 F0 0C  B 00 00 00 00 00 \(  J
 94 00 00 00 00 00  8  J 94 00 00 00 00 00  H  J 94 00 00 00 00 00  X  J 94 00
 00 00 00 00  p AA  @ 00 00 00 00 00 14 00 00 00 00 00 00 00  p AA  @ 00 00 00
 00 00 13 00 00 00 00 00 00 00 80 A8  @ 00 00 00 00 00 0D 00 00 00 00 00 00 00
  p AA  @ 00 00 00 00 00 12 00 00 00 00 00 00 00 F0 8C  @ 00 00 00 00 00  x  J
 94 00 00 00 00 00  p AA  @ 00 00 00 00 00 14 00 00 00 00 00 00 00  `  5  A 00
 00 00 00 00 98  J 94 00 00 00 00 00  @  2  @ 00 00 00 00 00 B0  J 94 00 00 00
 00 00 C0  J 94 00 00 00 00 00  @  $  @ 00 00 00 00 00 00 00 00 00 00 00 00 00
  p AA  @ 00 00 00 00 00 12 00 00 00 00 00 00 00
```


```
--------------------------------------------------------------------------------
*You may recompile the `fastlisp' with commented `#define _NOISY_MODE1_'
 to disable print of the linked bytecode.
*** Immediate running of the compiled and linked bytecode will start
    here just after the time report!
Time spent to check and prepare the task approx.:
   Used by process: 0.014997sec.
   Used by  system: 0.001000sec.
   Total used time: 1.599700000000E-02sec.
Real absolute time: 1.600986665728E-02sec.
*** Resetting time counters (second event controlpoint)... ***
================================================================================
Processing data cluster 1
Processing data cluster 2
Processing data cluster 3
Processing data cluster 4
Processing data cluster 5
Processing data cluster 6
Processing data cluster 7
Processing data cluster 8
Processing data cluster 9

                                               . . .

Processing data cluster 991
Processing data cluster 992
Processing data cluster 993
Processing data cluster 994
Processing data cluster 995
Processing data cluster 996
Processing data cluster 997
Processing data cluster 998
Processing data cluster 999
Processing data cluster 1000

================================================================================
Time spent to run the task:
   Used by process: 47.419791sec.
   Used by  system: 1.668747sec.
   Total used time: 4.908853800000E+01sec.
Real absolute time: 5.400968757646E+01sec.
```

dftest_exec.BMDFMldr

```
`Cursor invisible (vi)' capability was not found, default value is used!
`Cursor visible (ve)' capability was not found, default value is used!
Current termcap settings:
  TERM_TYPE=`vt100'; LINES_TERM=`24'; COLUMNS_TERM=`80';
  CLRSCR_TERM=`\e[H\e[J'; REVERSE_TERM=`\e[7m'; BLINK_TERM=`\e[5m';
  BOLD_TERM=`\e[1m'; NORMAL_TERM=`\e[0m'; HIDECURSOR_TERM=`';
  SHOWCURSOR_TERM=`'; GOTOCURSOR_TERM=`\e[%i%d;%dH'.
Reading the `/tmp/.BMDFMsrv' BM_DFM connection file...
Opening the `/tmp/.BMDFMsrv_npipe' BM_DFM named FIFO pipe...
Accessing the BM_DFM Server...
Receiving the Global FastLisp function set from the BM_DFM Server...
Linked Global function bytecode size is 64bytes.
--------------------------------------------------------------------------------
```


```
  h  { A0  ; 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80  >
  @ 00 00 00 00 00 01 00 00 00 00 00 00 00 80  { A0  ; 00 00 00 00 00  >  @ 00
 00 00 00 00 01 00 00 00 00 00 00 00
```


```
--------------------------------------------------------------------------------
*You may recompile the `BMDFMldr' with commented `#define _NOISY_MODE1_'
 to disable print of the linked Global function bytecode.
Connection with the BM_DFM Server has been established but not yet registered.
Checking whether the `dftest_exec.flp' file is already precompiled...
Reading the `dftest_exec.flp' source FastLisp file...
*** Resetting time counters (first null assignment)... ***
Modifying the FastLisp code (PATTERN No# 1)...
  (PROGN <Global_FastLisp_function_set> <FastLisp_prog>)
Checking the syntax of the source FastLisp file...
Modifying the FastLisp code (PATTERN No# 3)...
  (PROGN {(SETQ <termcap_var> <termcap_val>) }<FastLisp_prog>)
Looking for uninitialized variables/arrays in the FastLisp code...
Checking the CODE STYLE RESTRICTIONS for the BM_DFM parallel processing...
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*  Summary of the BM_DFM CODE STYLE RESTRICTIONS:               *
*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~               *
*                                                               *
*  o Variable names within the inclusive range of               *
*    [`TMP__000000000'; `TMP__999999999'] are reserved.         *
*  o `SHADOW' is the reserved name for a UDF.                   *
*  o Array names should differ from ordinary variable names.    *
*  o Every variable should be initialized before use.           *
*    The following is an example of how to copy an array:       *
*      ...                                                      *
*      (arsetq a 0 1)                                           *
*      (arsetq a 1 5)                                           *
*      (alsetq b (alindex a 2))     # instead of `(setq b a)'   *
*      ...                                                      *
*  o The <step> and <limit> values of a <for> loop should be    *
*    the integer numeric constants, function arguments or       *
*    initialized variables which are not changed inside this    *
*    <for> loop.                                                *
*  o Second argument of the booleans <or> and <and> should      *
*    not include any assignments, I/O, conditional/             *
*    iteration processing and UDF calls.                        *
*                                                               *
*  NOTE: Any conventional program can be converted by a         *
*        formal procedure to the program that is compliant      *
*        with the above mentioned code style restrictions.      *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*You may recompile BMDFMldr module with commented `#define EXPLAIN_RULE'
 to disable print of the code style restriction rule summary.
Squeezing the nested source PROGN statements...
  Redundant nested source PROGN statements removed: 2.
Modifying the FastLisp code (PATTERN No# 5)...
  (PROGN (OUTF (PRN_STRING_FMT) (CAT "" <FastLisp_prog>)) "")
Reorganizing the FastLisp code...
Resolving data types in the FastLisp code...
Registering in the BM_DFM Server Task Connection Zone...
Forking up the message queue listener...
Listener engine has been commenced.
The Loader/Listener pair is fully attached by the BM_DFM Server:
  Loader PID=3394, Listener PID=3394, SocketN# is 0.
--------------------------------------------------------------------------------
(PROGN
  (SETQ@S MAIN:TERM_TYPE@S "vt100")
  (SETQ@I MAIN:LINES_TERM@I 24)
  (SETQ@I MAIN:COLUMNS_TERM@I 80)
  (SETQ@S MAIN:CLRSCR_TERM@S "\e[H\e[J")
  (SETQ@S MAIN:REVERSE_TERM@S "\e[7m")
  (SETQ@S MAIN:BLINK_TERM@S "\e[5m")
  (SETQ@S MAIN:BOLD_TERM@S "\e[1m")
  (SETQ@S MAIN:NORMAL_TERM@S "\e[0m")
  (SETQ@S MAIN:HIDECURSOR_TERM@S "")
  (SETQ@S MAIN:SHOWCURSOR_TERM@S "")
  (SETQ@S MAIN:GOTOCURSOR_TERM@S "\e[%i%d;%dH")
  (SETQ@I MAIN:RECORDSIZE@I 40)
  (SETQ@S MAIN:INPUTTESTFILENAME@S "dftest_in.dat")
  (SETQ@S MAIN:OUTPUTTESTFILENAME@S "dftest_out.dat")
  (SETQ@S
    MAIN:INPUTTESTFILENAME@S
    (CAT@J
      MAIN:INPUTTESTFILENAME@S
      (IF@J (AT@J "." MAIN:INPUTTESTFILENAME@S) "" ".dat")
    )
  )
  (SETQ@S
    MAIN:INPUTTESTFILENAME@S
    (STRTRAN@J
      MAIN:INPUTTESTFILENAME@S "."
      (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
    )
  )
  (SETQ@I MAIN:FDESCR_@I (FILE_OPEN@J MAIN:INPUTTESTFILENAME@S))
  (SETQ@I MAIN:TMP__000000003@I (==@I -1 MAIN:FDESCR_@I))
  (IF@J
    MAIN:TMP__000000003@I
    (PROGN
      (SETQ@S
        MAIN:TMP__000000002
        (OUTF "Error opening file %s\n" MAIN:INPUTTESTFILENAME@S)
      )
      (EXIT)
    )
    (SETQ@I MAIN:TMP__000000002 (FILE_CLOSE@J MAIN:FDESCR_@I))
  )
  (SETQ@S
    MAIN:OUTPUTTESTFILENAME@S
    (CAT@J
      MAIN:OUTPUTTESTFILENAME@S
      (IF@J (AT@J "." MAIN:OUTPUTTESTFILENAME@S) "" ".dat")
    )
  )
  (SETQ@S
    MAIN:OUTPUTTESTFILENAME@S
    (STRTRAN@J
      MAIN:OUTPUTTESTFILENAME@S "."
      (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
    )
  )
  (SETQ@I MAIN:FDESCR@I (FILE_CREATE@J MAIN:OUTPUTTESTFILENAME@S))
  (SETQ@I MAIN:TMP__000000002 (==@I -1 MAIN:FDESCR@I))
  (IF
    MAIN:TMP__000000002
    (PROGN
      (SETQ@S
        MAIN:TMP__000000002
        (OUTF "Error creating file %s\n" MAIN:OUTPUTTESTFILENAME@S)
      )
      (EXIT)
    )
    (SETQ@I MAIN:FDESCR@I (FILE_CLOSE@J MAIN:FDESCR@I))
  )
  (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:RECORDSIZE@I 40))
  (IF@J
    MAIN:TMP__000000003@I
    (SETQ@I MAIN:RECORDSIZE@I 40)
    (SETQ@Z MAIN:TMP__000000002 NIL)
  )
  (SETQ@I MAIN:INPUTFILEPOSITION@I 0)
  (SETQ@I MAIN:SEQUENCEID@I 0)
  (SETQ@I MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:FDESCR@I)
  (SETQ@I
    MAIN:DATACLUSTERSIZE@I
    (GET_NEXT_CLUSTER_SIZE@J
      MAIN:INPUTTESTFILENAME@S MAIN:INPUTFILEPOSITION@I MAIN:RECORDSIZE@I
    )
  )
  (SETQ@I MAIN:TMP__000000001 (<@I 0 MAIN:DATACLUSTERSIZE@I))
  (WHILE@J
    MAIN:TMP__000000001
    (PROGN
      (SETQ@I MAIN:SEQUENCEID@I (++@J MAIN:SEQUENCEID@I))
      (SETQ@S
        MAIN:TMP__000000002
        (OUTF "Processing data cluster %ld\n" MAIN:SEQUENCEID@I)
      )
      (SETQ@I
```

```
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (ASYNCHEAP_CREATE@J MAIN:DATACLUSTERSIZE@I)
      )
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (READ_DATA_TO_CLUSTER@J
          MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:INPUTTESTFILENAME@S
          MAIN:INPUTFILEPOSITION@I MAIN:DATACLUSTERSIZE@I
        )
      )
      (SETQ@I
        MAIN:INPUTFILEPOSITION@I
        (+@J MAIN:INPUTFILEPOSITION@I MAIN:DATACLUSTERSIZE@I)
      )
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (PROCESS_DATA_CLUSTER@J
          MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:RECORDSIZE@I
          MAIN:DATACLUSTERSIZE@I
        )
      )
      (SETQ@I
        MAIN:DATACLUSTERINMEMORY_PTR@I
        (WRITE_CLUSTER_TO_FILE@J
          MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:DATACLUSTERSIZE@I
          MAIN:OUTPUTTESTFILENAME@S MAIN:OUTPUTTESTFILEORDERING_SYNC@I
        )
      )
      (SETQ@I
        MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:DATACLUSTERINMEMORY_PTR@I
      )
      (SETQ@I
        MAIN:TMP__000000002
        (ASYNCHEAP_DELETE@J MAIN:DATACLUSTERINMEMORY_PTR@I)
      )
      (SETQ@I
        MAIN:DATACLUSTERSIZE@I
        (GET_NEXT_CLUSTER_SIZE@J
          MAIN:INPUTTESTFILENAME@S MAIN:INPUTFILEPOSITION@I MAIN:RECORDSIZE@I
        )
      )
      (SETQ@I MAIN:TMP__000000001 (<@I 0 MAIN:DATACLUSTERSIZE@I))
    )
  )
  (SETQ@S
    MAIN:TMP__000000001
    (OUTF
      (PRN_STRING_FMT)
      (CAT@J "" (SPACE@J (&@J 0 MAIN:OUTPUTTESTFILEORDERING_SYNC@I)))
    )
  )
  (SETQ@S MAIN:TMP__000000000@S "")
)
--------------------------------------------------------------------------------
(PROGN (SETQ@S MAIN:TERM_TYPE@S "vt100") (SETQ@I MAIN:LINES_TERM@I 24) (SETQ@I
MAIN:COLUMNS_TERM@I 80) (SETQ@S MAIN:CLRSCR_TERM@S "\e[H\e[J") (SETQ@S MAIN:REV
ERSE_TERM@S "\e[7m") (SETQ@S MAIN:BLINK_TERM@S "\e[5m") (SETQ@S MAIN:BOLD_TERM@
S "\e[1m") (SETQ@S MAIN:NORMAL_TERM@S "\e[0m") (SETQ@S MAIN:HIDECURSOR_TERM@S "
") (SETQ@S MAIN:SHOWCURSOR_TERM@S "") (SETQ@S MAIN:GOTOCURSOR_TERM@S "\e[%i%d;%
dH") (SETQ@I MAIN:RECORDSIZE@I 40) (SETQ@S MAIN:INPUTTESTFILENAME@S "dftest_in.
dat") (SETQ@S MAIN:OUTPUTTESTFILENAME@S "dftest_out.dat") (SETQ@S MAIN:INPUTTES
TFILENAME@S (CAT@J MAIN:INPUTTESTFILENAME@S (IF@J (AT@J "." MAIN:INPUTTESTFILEN
AME@S) "" ".dat"))) (SETQ@S MAIN:INPUTTESTFILENAME@S (STRTRAN@J MAIN:INPUTTESTF
ILENAME@S "." (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") "."))) (SETQ@I
 MAIN:FDESCR_@I (FILE_OPEN@J MAIN:INPUTTESTFILENAME@S)) (SETQ@I MAIN:TMP__00000
0003@I (==@I -1 MAIN:FDESCR_@I)) (IF@J MAIN:TMP__000000003@I (PROGN (SETQ@S MAI
N:TMP__000000002 (OUTF "Error opening file %s\n" MAIN:INPUTTESTFILENAME@S)) (EX
IT)) (SETQ@I MAIN:TMP__000000002 (FILE_CLOSE@J MAIN:FDESCR_@I))) (SETQ@S MAIN:O
UTPUTTESTFILENAME@S (CAT@J MAIN:OUTPUTTESTFILENAME@S (IF@J (AT@J "." MAIN:OUTPU
TTESTFILENAME@S) "" ".dat"))) (SETQ@S MAIN:OUTPUTTESTFILENAME@S (STRTRAN@J MAIN
:OUTPUTTESTFILENAME@S "." (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".
"))) (SETQ@I MAIN:FDESCR@I (FILE_CREATE@J MAIN:OUTPUTTESTFILENAME@S)) (SETQ@I M
AIN:TMP__000000002 (==@I -1 MAIN:FDESCR@I)) (IF MAIN:TMP__000000002 (PROGN (SET
Q@S MAIN:TMP__000000002 (OUTF "Error creating file %s\n" MAIN:OUTPUTTESTFILENAM
E@S)) (EXIT)) (SETQ@I MAIN:FDESCR@I (FILE_CLOSE@J MAIN:FDESCR@I))) (SETQ@I MAIN
:TMP__000000003@I (<@I MAIN:RECORDSIZE@I 40)) (IF@J MAIN:TMP__000000003@I (SETQ
@I MAIN:RECORDSIZE@I 40) (SETQ@Z MAIN:TMP__000000002 NIL)) (SETQ@I MAIN:INPUTFI
LEPOSITION@I 0) (SETQ@I MAIN:SEQUENCEID@I 0) (SETQ@I MAIN:OUTPUTTESTFILEORDERIN
G_SYNC@I MAIN:FDESCR@I) (SETQ@I MAIN:DATACLUSTERSIZE@I (GET_NEXT_CLUSTER_SIZE@J
 MAIN:INPUTTESTFILENAME@S MAIN:INPUTFILEPOSITION@I MAIN:RECORDSIZE@I)) (SETQ@I
MAIN:TMP__000000001 (<@I 0 MAIN:DATACLUSTERSIZE@I)) (WHILE@J MAIN:TMP__00000000
1 (PROGN (SETQ@I MAIN:SEQUENCEID@I (++@J MAIN:SEQUENCEID@I)) (SETQ@S MAIN:TMP__
000000002 (OUTF "Processing data cluster %ld\n" MAIN:SEQUENCEID@I)) (SETQ@I MAI
N:DATACLUSTERINMEMORY_PTR@I (ASYNCHEAP_CREATE@J MAIN:DATACLUSTERSIZE@I)) (SETQ@
I MAIN:DATACLUSTERINMEMORY_PTR@I (READ_DATA_TO_CLUSTER@J MAIN:DATACLUSTERINMEMO
RY_PTR@I MAIN:INPUTTESTFILENAME@S MAIN:INPUTFILEPOSITION@I MAIN:DATACLUSTERSIZE
@I)) (SETQ@I MAIN:INPUTFILEPOSITION@I (+@J MAIN:INPUTFILEPOSITION@I MAIN:DATACL
USTERSIZE@I)) (SETQ@I MAIN:DATACLUSTERINMEMORY_PTR@I (PROCESS_DATA_CLUSTER@J MA
IN:DATACLUSTERINMEMORY_PTR@I MAIN:RECORDSIZE@I MAIN:DATACLUSTERSIZE@I)) (SETQ@I
 MAIN:DATACLUSTERINMEMORY_PTR@I (WRITE_CLUSTER_TO_FILE@J MAIN:DATACLUSTERINMEMO
RY_PTR@I MAIN:DATACLUSTERSIZE@I MAIN:OUTPUTTESTFILENAME@S MAIN:OUTPUTTESTFILEOR
DERING_SYNC@I)) (SETQ@I MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:DATACLUSTERINME
MORY_PTR@I) (SETQ@I MAIN:TMP__000000002 (ASYNCHEAP_DELETE@J MAIN:DATACLUSTERINM
EMORY_PTR@I)) (SETQ@I MAIN:DATACLUSTERSIZE@I (GET_NEXT_CLUSTER_SIZE@J MAIN:INPU
TTESTFILENAME@S MAIN:INPUTFILEPOSITION@I MAIN:RECORDSIZE@I)) (SETQ@I MAIN:TMP__
000000001 (<@I 0 MAIN:DATACLUSTERSIZE@I)))) (SETQ@S MAIN:TMP__000000001 (OUTF (
PRN_STRING_FMT) (CAT@J "" (SPACE@J (&@J 0 MAIN:OUTPUTTESTFILEORDERING_SYNC@I)))
)) (SETQ@S MAIN:TMP__000000000@S ""))
--------------------------------------------------------------------------------
*You may recompile BMDFM1dr module with commented `#define _NOISY_MODE_'
 to disable print of the FastLisp code.
Performing preliminary STATIC SCHEDULING (HARD_ARRAY_SYNCHRO=NO,
  EXT_IN_OUT_SYNCHRO=YES)...
Progress: *S*i*i*i*w
The translator module has finished the static scheduling.
The translator has returned the following exit code: 0(Success).
The following generated control sequence (so-called `BM_DFM UNICODE')
will be transferred to the BM_DFM kernel:
--------------------------------------------------------------------------------
(CTRL
  (N# 0)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 20 "MAIN:TERM_TYPE@S")
      (1 12 "MAIN:LINES_TERM@I")
      (2 3 "MAIN:COLUMNS_TERM@I")
      (3 2 "MAIN:CLRSCR_TERM@S")
      (4 17 "MAIN:REVERSE_TERM@S")
      (5 0 "MAIN:BLINK_TERM@S")
      (6 1 "MAIN:BOLD_TERM@S")
      (7 13 "MAIN:NORMAL_TERM@S")
      (8 9 "MAIN:HIDECURSOR_TERM@S")
      (9 19 "MAIN:SHOWCURSOR_TERM@S")
      (10 8 "MAIN:GOTOCURSOR_TERM@S")
      (11 16 "MAIN:RECORDSIZE@I")
      (12 11 "MAIN:INPUTTESTFILENAME@S")
      (13 14 "MAIN:OUTPUTTESTFILENAME@S")
      (14 11 "MAIN:INPUTTESTFILENAME@S")
      (15 11 "MAIN:INPUTTESTFILENAME@S")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@S MAIN:TERM_TYPE@S "vt100"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
        " v  t  1  0  0 00 00 00"
      )
      (Var_Ptrs 0)
    )
    (Fnc
      (N# 1)
      (FLP (SETQ@I MAIN:LINES_TERM@I 24))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" "18 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1)
    )
    (Fnc
      (N# 2)
      (FLP (SETQ@I MAIN:COLUMNS_TERM@I 80))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" " P 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 2)
    )
    (Fnc
      (N# 3)
      (FLP (SETQ@S MAIN:CLRSCR_TERM@S "\e[H\e[J"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
        "1B  [  H 1B  [  J 00 00"
      )
      (Var_Ptrs 3)
    )
    (Fnc
      (N# 4)
      (FLP (SETQ@S MAIN:REVERSE_TERM@S "\e[7m"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        "1B  [  7  m 00 00 00 00"
      )
      (Var_Ptrs 4)
    )
    (Fnc
      (N# 5)
      (FLP (SETQ@S MAIN:BLINK_TERM@S "\e[5m"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        "1B  [  5  m 00 00 00 00"
      )
      (Var_Ptrs 5)
    )
    (Fnc
      (N# 6)
      (FLP (SETQ@S MAIN:BOLD_TERM@S "\e[1m"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        "1B  [  1  m 00 00 00 00"
      )
      (Var_Ptrs 6)
    )
    (Fnc
      (N# 7)
      (FLP (SETQ@S MAIN:NORMAL_TERM@S "\e[0m"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        "1B  [  0  m 00 00 00 00"
      )
      (Var_Ptrs 7)
    )
    (Fnc
      (N# 8)
      (FLP (SETQ@S MAIN:HIDECURSOR_TERM@S ""))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 8)
    )
    (Fnc
      (N# 9)
```

```
      (FLP (SETQ@S MAIN:SHOWCURSOR_TERM@S ""))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 9)
    )
    (Fnc
      (N# 10)
      (FLP (SETQ@S MAIN:GOTOCURSOR_TERM@S "\e[%i%d;%dH"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "0A 00 00 00 00 00 00 00"
        "1B  [  %  i  %  d  ;  %" " d  H 00 00 00 00 00 00"
      )
      (Var_Ptrs 10)
    )
    (Fnc
      (N# 11)
      (FLP (SETQ@I MAIN:RECORDSIZE@I 40))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" "\( 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 11)
    )
    (Fnc
      (N# 12)
      (FLP (SETQ@S MAIN:INPUTTESTFILENAME@S "dftest_in.dat"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "0D 00 00 00 00 00 00 00"
        " d  f  t  e  s  t  _  i" " n  .  d  a  t 00 00 00"
      )
      (Var_Ptrs 12)
    )
    (Fnc
      (N# 13)
      (FLP (SETQ@S MAIN:OUTPUTTESTFILENAME@S "dftest_out.dat"))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "0E 00 00 00 00 00 00 00"
        " d  f  t  e  s  t  _  o" " u  t  .  d  a  t 00 00"
      )
      (Var_Ptrs 13)
    )
    (Fnc
      (N# 14)
      (FLP
        (SETQ@S
          MAIN:INPUTTESTFILENAME@S
          (CAT@J
            MAIN:INPUTTESTFILENAME@S
            (IF@J (AT@J "." MAIN:INPUTTESTFILENAME@S) "" ".dat")
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 F4 01 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" "D4 1C 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" "0A 00 00 00 00 00 00 00"
        "0C 00 00 00 00 00 00 00" "D4 EC 01 00 00 00 00 00"
        "02 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " . 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        " .  d  a  t 00 00 00 00"
      )
      (Var_Ptrs 14 12)
    )
    (Fnc
      (N# 15)
      (FLP
        (SETQ@S
          MAIN:INPUTTESTFILENAME@S
          (STRTRAN@J
            MAIN:INPUTTESTFILENAME@S "."
            (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 14 02 00 00 00 00 00" "03 00 00 00 00 00 00 00"
        "04 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
        " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " . 00 00 00 00 00 00 00" "D4 F4 01 00 00 00 00 00"
        "02 00 00 00 00 00 00 00" "0C 00 00 00 00 00 00 00"
        "D4 1C 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
        " T DC 02 00 00 00 00 00" "D4 C4 01 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " T DC 02 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " . 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 15 14)
    )
  )
)
```

```
(CTRL
  (N# 1)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 11) (VarName "MAIN:INPUTTESTFILENAME@S") (Inq_Dest Ld))
)
(CTRL (N# 2) (OpGroup 1) (COP 83) (<accum_chr> (dfmget_sdata)))
(CTRL (N# 3) (OpGroup 3) (COP 23) (<accum_slo> (FILE_OPEN <accum_chr>)))
(CTRL
  (N# 4)
  (OpGroup 1)
  (COP 71)
  (dfmput_idata <accum_slo> (VarRef 7) (VarName "MAIN:FDESCR_@I"))
)
(CTRL
  (N# 5)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 7 "MAIN:FDESCR_@I")
      (1 24 "MAIN:TMP__000000003@I")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:TMP__000000003@I (==@I -1 MAIN:FDESCR_@I)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  h 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "FF FF FF FF FF FF FF FF" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL
  (N# 6)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 24) (VarName "MAIN:TMP__000000003@I") (Inq_Dest Ld))
)
(CTRL (N# 7) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL
  (N# 8)
  (OpGroup 2)
  (COP 17)
  (IF_NOT <accum_slo> (GOTO 12))
  (REM "Pass over `MAIN:TMP__000000003@I' <if> conditional branch")
)
(CTRL
  (N# 9)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 11 "MAIN:INPUTTESTFILENAME@S")
      (1 23 "MAIN:TMP__000000002")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@S
          MAIN:TMP__000000002
          (OUTF "Error opening file %s\n" MAIN:INPUTTESTFILENAME@S)
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " T  8 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "06 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "16 00 00 00 00 00 00 00" " E  r  r  o  r __  o  p"
        " e  n  i  n  g __  f  i" " l  e __  %  s 0A 00 00"
        " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      )
      (Inq_Dest Ls)
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL (N# 10) (OpGroup 2) (COP 14) (GOTO 48) (REM "EXIT"))
(CTRL
  (N# 11)
  (OpGroup 2)
  (COP 14)
  (GOTO 16)
  (REM "Pass over `MAIN:TMP__000000003@I' <else> conditional branch")
)
(CTRL
  (N# 12)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 7) (VarName "MAIN:FDESCR_@I") (Inq_Dest Ld))
)
(CTRL (N# 13) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL (N# 14) (OpGroup 3) (COP 29) (<accum_slo> (FILE_CLOSE <accum_slo>)))
(CTRL
  (N# 15)
  (OpGroup 1)
  (COP 71)
  (dfmput_idata <accum_slo> (VarRef 23) (VarName "MAIN:TMP__000000002"))
)
(CTRL
  (N# 16)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 14 "MAIN:OUTPUTTESTFILENAME@S")
      (1 14 "MAIN:OUTPUTTESTFILENAME@S")
      (2 14 "MAIN:OUTPUTTESTFILENAME@S")
    )
    (Fnc
      (N# 0)
      (FLP
```

```
        (SETQ@S
          MAIN:OUTPUTTESTFILENAME@S
          (CAT@J
            MAIN:OUTPUTTESTFILENAME@S
            (IF@J (AT@J "." MAIN:OUTPUTTESTFILENAME@S) "" ".dat")
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 F4 01 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" "D4 1C 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" "0A 00 00 00 00 00 00 00"
        "0C 00 00 00 00 00 00 00" "D4 EC 01 00 00 00 00 00"
        "02 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " . 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        " .  d  a  t 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
    (Fnc
      (N# 1)
      (FLP
        (SETQ@S
          MAIN:OUTPUTTESTFILENAME@S
          (STRTRAN@J
            MAIN:OUTPUTTESTFILENAME@S "."
            (CAT@J (IF@J (ID_TASKJOB) (STR@I (ID_TASKJOB)) "") ".")
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 14 02 00 00 00 00 00" "03 00 00 00 00 00 00 00"
        "04 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
        " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " . 00 00 00 00 00 00 00" "D4 F4 01 00 00 00 00 00"
        "02 00 00 00 00 00 00 00" "0C 00 00 00 00 00 00 00"
        "D4 1C 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
        " T DC 02 00 00 00 00 00" "D4 C4 01 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " T DC 02 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " . 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 2 1)
    )
  )
)
(CTRL
  (N# 17)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata
    (VarRef 14)
    (VarName "MAIN:OUTPUTTESTFILENAME@S")
    (Inq_Dest Ld)
  )
)
(CTRL (N# 18) (OpGroup 1) (COP 83) (<accum_chr> (dfmget_sdata)))
(CTRL (N# 19) (OpGroup 3) (COP 22) (<accum_slo> (FILE_CREATE <accum_chr>)))
(CTRL
  (N# 20)
  (OpGroup 1)
  (COP 71)
  (dfmput_idata <accum_slo> (VarRef 6) (VarName "MAIN:FDESCR@I"))
)
(CTRL
  (N# 21)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 6 "MAIN:FDESCR@I")
      (1 23 "MAIN:TMP__000000002")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:TMP__000000002 (==@I -1 MAIN:FDESCR@I)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  h 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "FF FF FF FF FF FF FF FF" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL
  (N# 22)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 23) (VarName "MAIN:TMP__000000002") (Inq_Dest Ld))
)
(CTRL (N# 23) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL
  (N# 24)
  (OpGroup 2)
  (COP 17)
  (IF_NOT <accum_slo> (GOTO 28))
  (REM "Pass over `MAIN:TMP__000000002' <if> conditional branch")
)
(CTRL
  (N# 25)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 14 "MAIN:OUTPUTTESTFILENAME@S")
      (1 23 "MAIN:TMP__000000002")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@S
          MAIN:TMP__000000002
          (OUTF "Error creating file %s\n" MAIN:OUTPUTTESTFILENAME@S)
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " T  8 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "06 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "17 00 00 00 00 00 00 00" " E  r  r  o  r __  c  r"
        " e  a  t  i  n  g __  f" " i  l  e __  %  s 0A 00"
        " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      )
      (Inq_Dest Ls)
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL (N# 26) (OpGroup 2) (COP 14) (GOTO 48) (REM "EXIT"))
(CTRL
  (N# 27)
  (OpGroup 2)
  (COP 14)
  (GOTO 32)
  (REM "Pass over `MAIN:TMP__000000002' <else> conditional branch")
)
(CTRL
  (N# 28)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 6) (VarName "MAIN:FDESCR@I") (Inq_Dest Ld))
)
(CTRL (N# 29) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL (N# 30) (OpGroup 3) (COP 29) (<accum_slo> (FILE_CLOSE <accum_slo>)))
(CTRL
  (N# 31)
  (OpGroup 1)
  (COP 71)
  (dfmput_idata <accum_slo> (VarRef 6) (VarName "MAIN:FDESCR@I"))
)
(CTRL
  (N# 32)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 16 "MAIN:RECORDSIZE@I")
      (1 24 "MAIN:TMP__000000003@I")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:TMP__000000003@I (<@I MAIN:RECORDSIZE@I 40)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  x 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "\( 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
  )
)
(CTRL
  (N# 33)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 24) (VarName "MAIN:TMP__000000003@I") (Inq_Dest Ld))
)
(CTRL (N# 34) (OpGroup 1) (COP 81) (<accum_slo> (dfmget_idata)))
(CTRL
  (N# 35)
  (OpGroup 2)
  (COP 17)
  (IF_NOT <accum_slo> (GOTO 38))
  (REM "Pass over `MAIN:TMP__000000003@I' <if> conditional branch")
)
(CTRL
  (N# 36)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 16 "MAIN:RECORDSIZE@I"))
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:RECORDSIZE@I 40))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" "\( 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 37)
  (OpGroup 2)
  (COP 14)
  (GOTO 39)
  (REM "Pass over `MAIN:TMP__000000003@I' <else> conditional branch")
)
(CTRL
  (N# 38)
  (OpGroup 1)
  (COP 50)
```

```
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array] (0 23 "MAIN:TMP__000000002"))
    (Fnc
      (N# 0)
      (FLP (SETQ@Z MAIN:TMP__000000002 NIL))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " T 06 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " Z 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
  )
)
(CTRL
  (N# 39)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 10 "MAIN:INPUTFILEPOSITION@I")
      (1 18 "MAIN:SEQUENCEID@I")
      (2 6 "MAIN:FDESCR@I")
      (3 15 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
      (4 11 "MAIN:INPUTTESTFILENAME@S")
      (5 16 "MAIN:RECORDSIZE@I")
      (6 5 "MAIN:DATACLUSTERSIZE@I")
      (7 22 "MAIN:TMP__000000001")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:INPUTFILEPOSITION@I 0))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 0)
    )
    (Fnc
      (N# 1)
      (FLP (SETQ@I MAIN:SEQUENCEID@I 0))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " I 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1)
    )
    (Fnc
      (N# 2)
      (FLP (SETQ@I MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:FDESCR@I))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 3 2)
    )
    (Fnc
      (N# 3)
      (FLP
        (SETQ@I
          MAIN:DATACLUSTERSIZE@I
          (GET_NEXT_CLUSTER_SIZE@J
            MAIN:INPUTTESTFILENAME@S MAIN:INPUTFILEPOSITION@I MAIN:RECORDSIZE@I
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " t 08 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
        "04 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
        " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " i 00 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        " i 00 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 6 4 0 5)
    )
    (Fnc
      (N# 4)
      (FLP (SETQ@I MAIN:TMP__000000001 (<@I 0 MAIN:DATACLUSTERSIZE@I)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4  x 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 7 6)
    )
  )
)
(CTRL (N# 40) (OpGroup 2) (COP 10) (PUSHA))
(CTRL
  (N# 41)
  (OpGroup 1)
  (COP 70)
  (dfmput_zdata (VarRef 22) (VarName "MAIN:TMP__000000001") (Inq_Dest Ld))
  (REM "<While> `MAIN:TMP__000000001' loop body begins here")
)
(CTRL (N# 42) (OpGroup 1) (COP 81) (SubCOP 1) (<loop_slo> (dfmget_idata)))
(CTRL
  (N# 43)
  (OpGroup 2)
  (COP 17)
  (SubCOP 1)
  (IF_NOT <loop_slo> (GOTO 46))
  (REM "Exit <while> loop")
)
(CTRL
  (N# 44)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 18 "MAIN:SEQUENCEID@I")
      (1 18 "MAIN:SEQUENCEID@I")
      (2 23 "MAIN:TMP__000000002")
      (3 5 "MAIN:DATACLUSTERSIZE@I")
      (4 4 "MAIN:DATACLUSTERINMEMORY_PTR@I")
      (5 11 "MAIN:INPUTTESTFILENAME@S")
      (6 10 "MAIN:INPUTFILEPOSITION@I")
      (7 4 "MAIN:DATACLUSTERINMEMORY_PTR@I")
      (8 10 "MAIN:INPUTFILEPOSITION@I")
      (9 16 "MAIN:RECORDSIZE@I")
      (10 4 "MAIN:DATACLUSTERINMEMORY_PTR@I")
      (11 14 "MAIN:OUTPUTTESTFILENAME@S")
      (12 15 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
      (13 4 "MAIN:DATACLUSTERINMEMORY_PTR@I")
      (14 15 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
      (15 23 "MAIN:TMP__000000002")
      (16 5 "MAIN:DATACLUSTERSIZE@I")
      (17 22 "MAIN:TMP__000000001")
    )
    (Fnc
      (N# 0)
      (FLP (SETQ@I MAIN:SEQUENCEID@I (++@J MAIN:SEQUENCEID@I)))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 EC 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 1 0)
    )
    (Fnc
      (N# 1)
      (FLP
        (SETQ@S
          MAIN:TMP__000000002
          (OUTF "Processing data cluster %ld\n" MAIN:SEQUENCEID@I)
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " T  8 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "07 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "1C 00 00 00 00 00 00 00" " P  r  o  c  e  s  s  i"
        " n  g __  d  a  t  a __" " c  l  u  s  t  e  r __"
        " %  l  d 0A 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00"
      )
      (Inq_Dest Ls)
      (Var_Ptrs 2 1)
    )
    (Fnc
      (N# 2)
      (FLP
        (SETQ@I
          MAIN:DATACLUSTERINMEMORY_PTR@I
          (ASYNCHEAP_CREATE@J MAIN:DATACLUSTERSIZE@I)
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 F4 02 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 4 3)
    )
    (Fnc
      (N# 3)
      (FLP
        (SETQ@I
          MAIN:DATACLUSTERINMEMORY_PTR@I
          (READ_DATA_TO_CLUSTER@J
            MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:INPUTTESTFILENAME@S
            MAIN:INPUTFILEPOSITION@I MAIN:DATACLUSTERSIZE@I
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " t 0C 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
        "05 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
        "07 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
        "02 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "04 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 7 4 5 6 3)
    )
    (Fnc
      (N# 4)
      (FLP
        (SETQ@I
          MAIN:INPUTFILEPOSITION@I
          (+@J MAIN:INPUTFILEPOSITION@I MAIN:DATACLUSTERSIZE@I)
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 BC 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "02 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 8 6 3)
    )
    (Fnc
      (N# 5)
```

```
  (FLP
    (SETQ@I
      MAIN:DATACLUSTERINMEMORY_PTR@I
      (PROCESS_DATA_CLUSTER@J
        MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:RECORDSIZE@I
        MAIN:DATACLUSTERSIZE@I
      )
    )
  )
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " t 10 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
    "04 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
    " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " i 00 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    " i 00 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 10 7 9 3)
)
(Fnc
  (N# 6)
  (FLP
    (SETQ@I
      MAIN:DATACLUSTERINMEMORY_PTR@I
      (WRITE_CLUSTER_TO_FILE@J
        MAIN:DATACLUSTERINMEMORY_PTR@I MAIN:DATACLUSTERSIZE@I
        MAIN:OUTPUTTESTFILENAME@S MAIN:OUTPUTTESTFILEORDERING_SYNC@I
      )
    )
  )
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " t 04 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    "05 00 00 00 00 00 00 00" "06 00 00 00 00 00 00 00"
    "07 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
    "01 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
    "02 00 00 00 00 00 00 00" " s 00 00 00 00 00 00 00"
    "03 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
    "04 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 13 10 3 11 12)
)
(Fnc
  (N# 7)
  (FLP
    (SETQ@I
      MAIN:OUTPUTTESTFILEORDERING_SYNC@I MAIN:DATACLUSTERINMEMORY_PTR@I
    )
  )
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 14 13)
)
(Fnc
  (N# 8)
  (FLP
    (SETQ@I
      MAIN:TMP__000000002
      (ASYNCHEAP_DELETE@J MAIN:DATACLUSTERINMEMORY_PTR@I)
    )
  )
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "D4 1C 03 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " i 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 15 13)
)
(Fnc
  (N# 9)
  (FLP
    (SETQ@I
      MAIN:DATACLUSTERSIZE@I
      (GET_NEXT_CLUSTER_SIZE@J
        MAIN:INPUTTESTFILENAME@S MAIN:INPUTFILEPOSITION@I MAIN:RECORDSIZE@I
      )
    )
  )
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "04 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " t 08 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
    "04 00 00 00 00 00 00 00" "05 00 00 00 00 00 00 00"
    " s 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    " i 00 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    " i 00 00 00 00 00 00 00" "03 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 16 5 8 9)
)
(Fnc
  (N# 10)
  (FLP (SETQ@I MAIN:TMP__000000001 (<@I 0 MAIN:DATACLUSTERSIZE@I)))
  (FLP_COMPILED
    "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "D4 04 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
    "D4  x 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
    "03 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
    "00 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
    "01 00 00 00 00 00 00 00"
  )
  (Var_Ptrs 17 16)
)
  )
)
(CTRL
  (N# 45)
  (OpGroup 2)
  (COP 14)
  (SubCOP 1)
  (GOTO 41)
  (REM
    "Continue <while> `MAIN:TMP__000000001' loop, <while> loop body ends here"
  )
)
(CTRL (N# 46) (OpGroup 2) (COP 11) (POPA))
(CTRL
  (N# 47)
  (OpGroup 1)
  (COP 50)
  (dfmput_marshaled_cluster
    (Vars_N#_Ref_Name_[Array]
      (0 15 "MAIN:OUTPUTTESTFILEORDERING_SYNC@I")
      (1 22 "MAIN:TMP__000000001")
      (2 21 "MAIN:TMP__000000000@S")
    )
    (Fnc
      (N# 0)
      (FLP
        (SETQ@S
          MAIN:TMP__000000001
          (OUTF
            (PRN_STRING_FMT)
            (CAT@J "" (SPACE@J (&@J 0 MAIN:OUTPUTTESTFILEORDERING_SYNC@I)))
          )
        )
      )
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " T  8 00 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "02 00 00 00 00 00 00 00" " T 80 02 00 00 00 00 00"
        "D4 F4 01 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "04 00 00 00 00 00 00 00" " S 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "D4 F8 01 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "D4 08 01 00 00 00 00 00" "02 00 00 00 00 00 00 00"
        "03 00 00 00 00 00 00 00" " I 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" " i 00 00 00 00 00 00 00"
        "01 00 00 00 00 00 00 00"
      )
      (Inq_Dest Ls)
      (Var_Ptrs 1 0)
    )
    (Fnc
      (N# 1)
      (FLP (SETQ@S MAIN:TMP__000000000@S ""))
      (FLP_COMPILED
        "D5 01 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "D4 05 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00" "01 00 00 00 00 00 00 00"
        " S 00 00 00 00 00 00 00" "00 00 00 00 00 00 00 00"
        "00 00 00 00 00 00 00 00"
      )
      (Var_Ptrs 2)
    )
  )
)
(CTRL (N# 48) (OpGroup 4) (COP 200) (END) (REM "End of the control sequence"))
```

```
--------------------------------------------------------------------------------
*You may recompile BMDFMldr module with commented `#define _NOISY_MODE1_'
 to disable print of the BM_DFM control sequence.
*** Uploading and immediate running of the BM_DFM control sequence by
    the BM_DFM kernel will start here just after the time report!
Time spent to check and prepare the task approx.:
   Used by process: 0.012998sec.
   Used by  system: 0.001999sec.
   Total used time: 1.499700000000E-02sec.
Real absolute time: 1.500889796387E-02sec.
*** Resetting time counters (second event controlpoint)... ***
================================================================================
The task is being carried out on SocketN# 0.
================================================================================
Processing data cluster 1
Processing data cluster 2
Processing data cluster 3
Processing data cluster 4
Processing data cluster 5
Processing data cluster 6
Processing data cluster 7
Processing data cluster 8
Processing data cluster 9
Processing data cluster 10

                                      . . .

Processing data cluster 991
Processing data cluster 992
Processing data cluster 993
Processing data cluster 994
Processing data cluster 995
Processing data cluster 996
Processing data cluster 997
Processing data cluster 998
Processing data cluster 999
Processing data cluster 1000

================================================================================
Time spent to run the task (by PARENT loader and CHILD listener):
   Used by process: 0.064990sec.
   Used by  system: 0.207968sec.
   Total used time: 2.729580000000E-01sec.
Real absolute time: 1.700989369432E+00sec.
Task has been detached (logged out) from the BM_DFM Server.
The BM_DFM Task Loader/Listener pair has done its job decently and gracefully.
```

<EOF>