Chapter 4

Dynamic Allocation

Three subroutines allocate CM arrays at run time, giving the programmer different levels of control over the array's layout. A fourth subroutine deallocates an array created by any of the other three.

```
  CALL CMF_ALLOCATE_ARRAY
&    ( FE_ARRAY, EXTENTS, RANK, TYPE )

  CALL CMF_ALLOCATE_LAYOUT_ARRAY
&    ( FE_ARRAY, EXTENTS, RANK, TYPE, ORDERS, WEIGHTS )

  CALL CMF_ALLOCATE_DETAILED_ARRAY
&    ( FE_ARRAY, EXTENTS, RANK, TYPE, ORDERS,
&      SUBGRIDS, PMASKS )

  CALL CMF_DEALLOCATE_ARRAY( FE_ARRAY )
```

The **`FE_ARRAY`** argument is an integer front-end vector whose length is the pre-defined constant **`CMF_SIZEOF_DESCRIPTOR`**. This array is treated as the *descriptor* of a CM array; the remaining arguments specify information to be placed in the slots of the descriptor. All three variants take as arguments:

- `EXTENTS` a front-end vector that contains dimension extents
- `RANK` a scalar integer that indicates rank
- `TYPE` a pre-defined integer constant that indicates type:
  `CMF_LOGICAL`
  `CMF_S_INTEGER`, `CMF_LONG_S_INTEGER`
  `CMF_FLOAT`, `CMF_DOUBLE`,
  `CMF_COMPLEX`, `CMF_DOUBLE_COMPLEX`

Copyright © 1994 Thinking Machines Corporation

The **FE_ARRAY** argument cannot be used as a CM array within the program unit that calls the allocation subroutine, since that program unit treats it as a front-end array. Instead, you pass the **FE_ARRAY** argument (that is, the descriptor) to another program unit that explicitly declares it a CM array. This method is illustrated in the following example.

4.1 Allocation Example (Canonical Layout)

```
 SUBROUTINE ALLOCATE()
 IMPLICIT NONE
 INTEGER NEW_ARRAY(CMF_SIZEOF_DESCRIPTOR)
 INTEGER EXTENTS(7), RANK, I
 PARAMETER (RANK=3)

 INCLUDE 'CMF_defs.h'

 DO I=1,RANK
    EXTENTS(I) = I * 10
 END DO

 CALL CMF_ALLOCATE_ARRAY
&   (NEW_ARRAY, EXTENTS, RANK, CMF_S_INTEGER)

 CALL PRINT_DIMS3D(NEW_ARRAY)

 CALL CMF_DEALLOCATE_ARRAY(NEW_ARRAY)
 END SUBROUTINE ALLOCATE


 SUBROUTINE PRINT_DIMS3D(IN)
 IMPLICIT NONE
 INTEGER IN(:,:,:)

 PRINT *,"Shape of DUMMY is (",UBOUND(IN,1),
&        ",",UBOUND(IN,2),
&        ",",UBOUND(IN,3),")"

 END SUBROUTINE PRINT_DIMS3D
```


Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

4.2 Controlling Array Layout

The "layout" and "detailed" variants of the allocation procedures take additional front-end vector arguments that contain layout information for each of the array dimensions. The significance of these arguments is comparable to the various forms of the **cmf** compiler directive **LAYOUT**.

- **ORDERS** contains symbolic constants indicating the ordering of each dimension: **CMF_SERIAL_ORDER** or **CMF_NEWS_ORDER**.
- **WEIGHTS** is a vector of integers indicating relative dimension weights.
- **SUBGRIDS** is a vector of integers indicating the desired subgrid length for each dimension (comparable to the **:BLOCK** item in the detailed-layout directive).
- **PMASKS** is a vector of integers that serve as bit-masks to indicate the desired processors (comparable to the **:PDESC** item in the detailed-layout directive). If **ORDERS** contains the value **CMF_SERIAL_ORDER** for any dimension, then **PMASKS** must contain 0 for that dimension.

There is no form directly comparable to the **:BLOCK :PROCS** form of the detailed **LAYOUT** directive. However, if **PMASKS** contains all zeros, the system computes the number of processors for each axis as extent / subgrid-length, rounded if necessary to the next power of 2.

4.3 Allocation Example (Detailed Layout)

```
IMPLICIT NONE
INCLUDE 'CMF_defs.h'
INTEGER NEWARRAY(CMF_SIZEOF_DESCRIPTOR)
INTEGER EXTENTS(7),ORDERS(7),SUBGRIDS(7),PMASKS(7)
INTEGER RANK,I
INTEGER NPN,NPN_FRAC,FRAC,SG1,SG2

PARAMETER (RANK = 2)
PARAMETER (FRAC = 4)
PARAMETER (SG1 = 5, SG2 = 40)
```


Copyright © 1994 Thinking Machines Corporation

```
      NPN = CMF_NUMBER_OF_PROCESSORS()
      NPN_FRAC = NPN/FRAC

      PMASKS(1) = (NPN_FRAC - 1) * FRAC
      PMASKS(2) = FRAC - 1

      SUBGRIDS(1) = SG1
      SUBGRIDS(2) = SG2

      EXTENTS(1) = NPN_FRAC * SG1
      EXTENTS(2) = FRAC * SG2

      DO I = 1,RANK
        ORDERS(I) = CMF_NEWS_ORDER
      END DO

      CALL CMF_ALLOCATE_DETAILED_ARRAY
     &  (NEWARRAY,EXTENTS,RANK,CMF_FLOAT,ORDERS,
     &  SUBGRIDS,PMASKS)

      CALL USE_NEWARRAY(NEWARRAY,EXTENTS)
      CALL CMF_DEALLOCATE_ARRAY(NEWARRAY)
      END

      SUBROUTINE USE_NEWARRAY(A,EXT)
      INTEGER EXT(2)
      REAL A(EXT(1),EXT(2)), B(EXT(1),EXT(2))
CMF$  LAYOUT A(:,:)
CMF$  ALIGN  B(I,J) WITH A(I,J)

      B = CSHIFT(A,DIM=1,SHIFT=1)

C     Other operations on arrays A and B

      RETURN
      END
```


Copyright © 1994 Thinking Machines Corporation

4.4 Language Comparison

The dynamic allocation utility procedures are largely, but not completely, redundant with the CM Fortran statement **ALLOCATE**, which creates *deferred-shape* CM arrays. Some differences are:

- **ALLOCATABLE** arrays cannot appear in **COMMON**, so their names are not available to all program units. Array pointers and arrays created with **CMF_ALLOCATE_ARRAY** or one of its variants can be globally available.
- Data types and ranks of deferred-shape arrays must be known at compile time. With **CMF_ALLOCATE_ARRAY**, they can be decided at run time (although used only in subroutines where the appropriate type and rank are declared).
- Storage allocated for deferred-shape arrays is deallocated upon exit from the program unit. Storage allocated by a utility subroutine must be deallocated explicitly by a call to **CMF_DEALLOCATE_ARRAY**.
- If a deferred-shape array is subject to a **LAYOUT** directive, the directive must appear in the specification part of the program unit (before any executable code), and the values of any variables used in the directive are those established upon entry. If you use the utility **CMF_ALLOCATE_LAYOUT_ARRAY** or **CMF_ALLOCATE_DETAILED_ARRAY** instead, you can compute before the call to determine layout-related values, such as subgrid lengths. The variable values used are those established at the time of execution.

Neither the utility library nor the CM Fortran language provides for dynamic allocation of front-end arrays or scalars. For this purpose, use the CM Fortran subroutines **FMALLOC** and **FFREE** in **libcmf77.a** (described in Part III*).* These subroutines provide an interface to the standard **malloc** and **free** functionality that, together with the **%VAL** operator, enable you to manage front-end storage.

Copyright © 1994 Thinking Machines Corporation

CMF_ALLOCATE_ARRAY

Allocates a CM array dynamically.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_ALLOCATE_ARRAY( ARRAY, EXTENTS, RANK, TYPE )
```

Arguments

ARRAY — An integer front-end array of size **CMF_SIZEOF_DESCRIPTOR** (a predefined constant). This argument will be modified to point to the allocated CM memory when the array is passed as an argument to a subprogram.

EXTENTS — An integer front-end vector that is at least **RANK(ARRAY)** in length and contains the dimension extents of the dynamic array. The first element specifies the length of axis 1, the second element specifies the length of the second axis, and so on. The axis ordering defaults to **CMF_NEWS_ORDER**.

RANK — An integer scalar indicating the rank of the dynamic array.

TYPE — A predefined integer constant that indicates the data type of the dynamic array. It must be one of:

CMF_LOGICAL
CMF_S_INTEGER
CMF_LONG_S_INTEGER
CMF_FLOAT
CMF_DOUBLE
CMF_COMPLEX
CMF_DOUBLE_COMPLEX

Result

None.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

Description

The subroutine **CMF_ALLOCATE_ARRAY** allocates CM storage to hold an array of the shape described by **RANK** and **EXTENTS** and of the type specified by **TYPE**. **ARRAY** is modified to serve as a descriptor for the array.

To use the array created by **CMF_ALLOCATE_ARRAY** as a CM array, you must pass **ARRAY** to a subprogram that explicitly declares it as a CM array. Since the program unit that calls **CMF_ALLOCATE_ARRAY** declares **ARRAY** as a front-end array, **ARRAY** cannot be used as a CM array in that program unit.

See Also

CMF_ALLOCATE_DETAILED_ARRAY
CMF_ALLOCATE_LAYOUT_ARRAY
CMF_DEALLOCATE_ARRAY

Copyright © 1994 Thinking Machines Corporation

CMF_ALLOCATE_LAYOUT_ARRAY

Allocates a CM array with a specified layout.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_ALLOCATE_LAYOUT_ARRAY( ARRAY, EXTENTS, RANK,
&                               TYPE, ORDERS, WEIGHTS )
```

Arguments

ARRAY — An integer front-end array of size **CMF_SIZEOF_DESCRIPTOR** (a predefined constant). This argument will be modified to point to the allocated CM memory when the array is passed as an argument to a subprogram.

EXTENTS — An integer front-end vector that is at least **RANK(ARRAY)** in length and contains the dimension extents of the dynamic array. The first element specifies the length of axis 1, the second element specifies the length of the second axis, etc.

RANK — An integer scalar indicating the rank of the dynamic array.

TYPE — A predefined integer constant that indicates the data type of the dynamic array. It must be one of:

- CMF_LOGICAL
- CMF_S_INTEGER
- CMF_LONG_S_INTEGER
- CMF_FLOAT
- CMF_DOUBLE
- CMF_COMPLEX
- CMF_DOUBLE_COMPLEX

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

ORDERS — A front-end array of integers that specify the ordering of each axis of the dynamic array. Each element of this array must be one of the following integer values:

```
CMF_NEWS_ORDER
CMF_SERIAL_ORDER
```

WEIGHTS — Front-end array of non-negative integers that specify the weight, or expected heaviness of use in communication operations, of each axis of the dynamic array. The **WEIGHTS** array should be initialized to all ones if no special weighting of axes is desired.

Result

None.

Description

The subroutine **CMF_ALLOCATE_LAYOUT_ARRAY** allocates the CM storage to hold an array of the shape described by **RANK** and **EXTENTS**, the type specified by **TYPE**, and with ordering and weights specified for each axis by **ORDERS** and **WEIGHTS**.

To use the array created by **CMF_ALLOCATE_ARRAY** as a CM array, you must pass **ARRAY** to a subprogram that explicitly declares it as a CM array. Since the program unit that calls **CMF_ALLOCATE_ARRAY** declares **ARRAY** as a front-end array, **ARRAY** cannot be used as a CM array in that program unit.

See Also

```
CMF_ALLOCATE_ARRAY
CMF_ALLOCATE_DETAILED_ARRAY
CMF_DEALLOCATE_ARRAY
```


Copyright © 1994 Thinking Machines Corporation

CMF_ALLOCATE_DETAILED_ARRAY

Allocates a CM array dynamically with a specified detailed layout.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_ALLOCATE_DETAILED_ARRAY( ARRAY, EXTENTS, RANK,
&                         TYPE, ORDERS, SUBGRIDS, PMASKS )
```

Arguments

ARRAY — An integer front-end array of size **CMF_SIZEOF_DESCRIPTOR** (a predefined constant). This argument will be modified to point to the allocated CM memory when the array is passed as an argument to a subprogram.

EXTENTS — An integer front-end vector that is at least **RANK(ARRAY)** in length and contains the dimension extents of the dynamic array. The first element specifies the length of axis 1, the second element specifies the length of the second axis, etc.

RANK — An integer scalar indicating the rank of the dynamic array.

TYPE — A predefined integer constant that indicates the data type of the dynamic array. It must be one of:

- CMF_LOGICAL
- CMF_S_INTEGER
- CMF_LONG_S_INTEGER
- CMF_FLOAT
- CMF_DOUBLE
- CMF_COMPLEX
- CMF_DOUBLE_COMPLEX

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

ORDERS — A front-end array of integers that specify the ordering of each axis of the dynamic array. Each element of this array must be one of the following integer values:

CMF_NEWS_ORDER

CMF_SERIAL_ORDER

For any axis described with **CMF_SERIAL_ORDER**, the corresponding **PMASKS** value must be 0, and the **SUBGRIDS** value must be the axis extent. Use **CMF_NEWS_ORDER** for axes described by other **SUBGRIDS** and **PMASKS** values.

SUBGRIDS — Front-end array of integers that specify the subgrid length for each axis.

PMASKS — Front-end array of integers that serve as bitmasks to indicate the processors on which to allocate each axis. If the **ORDERS** argument contains the value **CMF_SERIAL_ORDER** for any axis, the **PMASKS** argument must contain 0 for that axis.

Result

None.

Description

The subroutine **CMF_ALLOCATE_DETAILED_ARRAY** allocates the CM storage to hold an array of the shape described by **RANK** and **EXTENTS** and the type specified by **TYPE**.

The **SUBGRIDS** and **PMASKS** arguments enable you to specify in detail how the CM array is laid out on the parallel processing elements (vector units or nodes, depending on the execution model). For each array axis, the value in the **SUBGRIDS** argument specifies the number of elements in the per-processor subgrid; the value in **PMASKS** is a bit-mask that specifies which processing elements are used. The **ORDERS** argument can specify either serial or NEWS order.

To use the array created by **CMF_ALLOCATE_ARRAY** as a CM array, you must pass **ARRAY** to a subprogram that explicitly declares it as a CM array. Since the program unit that calls **CMF_ALLOCATE_ARRAY** declares **ARRAY** as a front-end array, **ARRAY** cannot be used as a CM array in that program unit.

Copyright © 1994 Thinking Machines Corporation

See Also

CMF_ALLOCATE_ARRAY
CMF_ALLOCATE_LAYOUT_ARRAY
CMF_DEALLOCATE_ARRAY

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_DEALLOCATE_ARRAY

Deallocates a CM array allocated with **CMF_ALLOCATE_ARRAY**, **CMF_ALLOCATE_DETAILED_ARRAY**, or **CMF_ALLOCATE_LAYOUT_ARRAY**.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_DEALLOCATE_ARRAY( ARRAY )
```

Arguments

ARRAY	Front-end array. The **ARRAY** argument to one of the dynamic allocation utility routines that has been modified to serve as a descriptor for a CM array. The CM array represented by this argument will be deallocated.

Result

None.

Description

CMF_DEALLOCATE_ARRAY deallocates a CM array that has been allocated with **CMF_ALLOCATE_ARRAY**, **CMF_ALLOCATE_LAYOUT_ARRAY**, or **CMF_ALLOCATE_DETAILED_ARRAY**. Only **ARRAY** arguments modified by one of these allocation subroutines should be passed to this deallocation subroutine. The contents of the CM array represented by **ARRAY** cannot be accessed after a call to this subroutine.

Copyright © 1994 Thinking Machines Corporation

See Also

CMF_ALLOCATE_ARRAY
CMF_ALLOCATE_DETAILED_ARRAY
CMF_ALLOCATE_LAYOUT_ARRAY

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation