SCHOOL OF PHYSICS AND ASTRONOMY
UNIVERSITY OF LEICESTER
Asterix Data Analysis
Programmer Note 004

PROG/004

RJ Vallance
19 July 1989

Dynamic Memory Access Routines

Contents

1 Introduction

The DYN_ routines are a set of routines for obtaining chunks of virtual memory dynamically. They are intended as a more tidy alternative to the DAT_TEMP - DAT_MAP method. For small amounts of memory they make calls to the relevant VMS system services to obtain virtual memory, whereas for larger amounts they create temporary mapped files on SYS$SCRATCH. No debris is left when the memory is released. The routines reside in ASTLIB.

2 Routine Descriptions

DYN_INIT — Initialise internal common blocks — DYN_INIT

Description: Initialises internal common blocks. Only necessary if to be used within a monolith ie. discrete programs need not call it.

Invocation: `CALL DYN_INIT ( )`

DYN_MAPR — Allocate memory for REAL data — DYN_MAPR

Description: Gets pointer to REAL array of specified size.

Invocation: `CALL DYN_MAPR ( NDIM, DIMS, PTR, STATUS )`

Arguments:

NDIM = INTEGER (Given)
The number of dimensions.

DIMS = INTEGER(*)
The dimensions.

PTR = POINTER (Returned)
The pointer to start of array.

STATUS = INTEGER (Given and Returned)
The global status/

DYN_MAPD — Allocate memory for DOUBLE PRECISION data — DYN_MAPD

Description: Gets pointer to DOUBLE PRECISION array of specified size.

Invocation: `CALL DYN_MAPD ( NDIM, DIMS, PTR, STATUS )`

Arguments:

NDIM = INTEGER (Given)
The number of dimensions.

DIMS = INTEGER(*)
The dimensions.

PTR = POINTER (Returned)
The pointer to start of array.

STATUS = INTEGER (Given and Returned)
The global status/

DYN_MAPI Allocate memory for INTEGER data DYN_MAPI

Description: Gets pointer to INTEGER array of specified size.

Invocation: `CALL DYN_MAPI ( NDIM, DIMS, PTR, STATUS )`

Arguments:

NDIM = INTEGER (Given)
The number of dimensions.

DIMS = INTEGER(*)
The dimensions.

PTR = POINTER (Returned)
The pointer to start of array.

STATUS = INTEGER (Given and Returned)
The global status/

DYN_MAPW Allocate memory for WORD data DYN_MAPW

Description: Gets pointer to WORD array of specified size.

Invocation: `CALL DYN_MAPW ( NDIM, DIMS, PTR, STATUS )`

Arguments:

NDIM = INTEGER (Given)
The number of dimensions.

DIMS = INTEGER(*)
The dimensions.

PTR = POINTER (Returned)
The pointer to start of array.

STATUS = INTEGER (Given and Returned)
The global status/

DYN_MAPB — Allocate memory for BYTE data — DYN_MAPB

Description: Gets pointer to BYTE array of specified size.

Invocation: `CALL DYN_MAPB ( NDIM, DIMS, PTR, STATUS )`

Arguments:

NDIM = INTEGER (Given)
The number of dimensions.

DIMS = INTEGER(*)
The dimensions.

PTR = POINTER (Returned)
The pointer to start of array.

STATUS = INTEGER (Given and Returned)
The global status/

DYN_MAPL — Allocate memory for LOGICAL data — DYN_MAPL

Description: Gets pointer to LOGICAL array of specified size.

Invocation: `CALL DYN_MAPL ( NDIM, DIMS, PTR, STATUS )`

Arguments:

NDIM = INTEGER (Given)
The number of dimensions.

DIMS = INTEGER(*)
The dimensions.

PTR = POINTER (Returned)
The pointer to start of array.

STATUS = INTEGER (Given and Returned)
The global status/

DYN_SIZE — Current size of pointer — DYN_SIZE

Description: Returns the total size of a dynamic array currently known to the DYN_ system. Zero is returned if the pointer is not known. For multi-dimensional arrays the product of the dimensions is returned.

Invocation: `CALL DYN_SIZE ( PTR, NVAL, STATUS )`

Arguments:

PTR = POINTER (Given)
The pointer to dynamic array.

NVAL = INTEGER (Returned)
The total number of values (elements) in array.

DYN_UNMAP — Unmap a pointer — DYN_UNMAP

Description: Frees virtual memory obtained with DYN_MAPx routines.

Invocation: `CALL DYN_UNMAP ( PTR )`

Arguments:

PTR = POINTER (Given)
The pointer to dynamic array.

DYN_CLOSE — Close down the system — DYN_CLOSE

Description: Frees all virtual memory obtained with DYN_MAPx routines.

Invocation: `CALL DYN_CLOSE ( )`