ProTEX guidelines

Purnendu Chakraborty

January 2, 2013

Contents

1 Introduction

This doc is a general guideline to document F90 source files (with ProTeX tags). For questions, contact: purnendu.chakraborty@nasa.gov.

Note:

- Many of the source lines are reproduced verbatim by ProTeX. So it is best, if we stick to the 80-column rule (including comments).
- A method can refer to both a subroutine or a function.

In a module/main program, we want to document the following:

1. a description of the module
2. USES (e.g. use ESMF_Mod)
3. list of PUBLIC TYPES
4. list PUBLIC MEMBER FUNCTIONS (subroutines/functions)
5. interface, arguments and description of each public method

In the following sections we use an example to illustrate uses of the ProTeX tags.

2 Prologue

Documentation starts right after the module name (line 4) with the `!BOP` (begin-of-prologue) tag (line 5), followed by the module name and a brief description of the module (line 6 - note the two !'s with a space between them). This is the name that ProTeX reads and prints out.

```
1   #include "MAPL_Generic.h"
2   #define MPI_NULL_TAG 99
3
4   module MAPL_CFIOMod
5     !BOP
6     ! !MODULE: MAPL_CFIOMod --- CF Compliant I/O for ESMF
```

Next is a detailed description of the module following the tag `!DESCRIPTION`. We can either write the description in the code itself (as is the case here) or include a TeX file (without preamble) as

`\input{filedesc.tex}`

In the case of include-ing a file, the file should be added/committed to CVS. Also note that the comment character '!' for the text following `!DESCRIPTION` should always be in column 1 (line 8). This is probably a quirk of ProTeX - and I hope to fix it sometime.

```
 7    !DESCRIPTION:
 8  !  Description of what this module does.
 9  !  blah blah etc. The comment character '!'
10  !  should always be in column 1.
```

The tag !USES is followed by all modules loaded and used.

```
11    !USES:
12    use ESMF_Mod
13    use MAPL_BaseMod
14
15    implicit none
```

The public methods are listed next followed by the end of prologue tag (!EOP).

```
16    private
17
18    !PUBLIC MEMBER FUNCTIONS:
19    public MAPL_CFIOclose
20    public MAPL_CFIOCreate
21    public MAPL_CFIOWrite
22    !EOP
23
24  contains
```

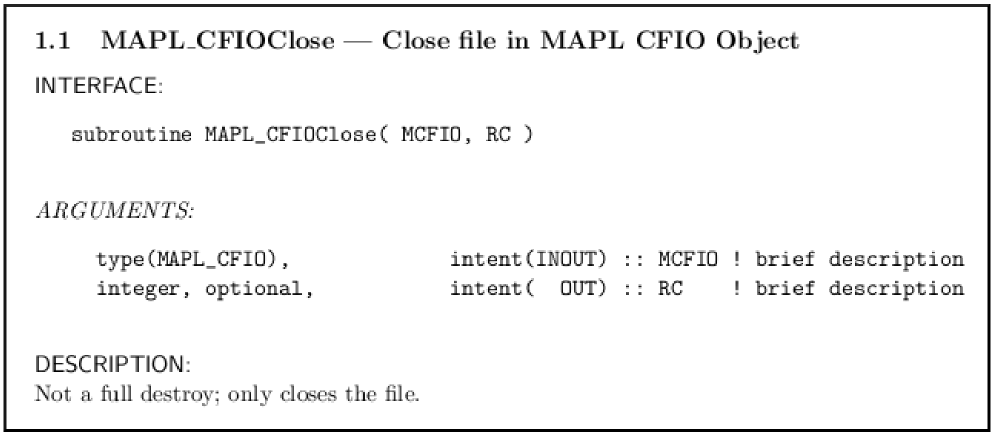
3 Contained method

Documentation (!IROUTINE, !INTERFACE, !ARGUMENTS and !DESCRIPTION tags) is enclosed inside a !BOPI/!EOPI (begin/end of prologue - internal) block. Documentation starts with the routine name and a brief description (line 26 - note the two '!' separated by a space). !INTERFACE precedes the function/subroutine name and !ARGUMENTS is followed by the list of in/out/inout arguments in the interface. It is a good practice to include a brief description of each argument. As before, The text following !DESCRIPTION should have the comment character in column 1.

```
25    !BOPI
26    ! !IROUTINE: MAPL_CFIOClose --- Close file in MAPL CFIO Object
27
28    !INTERFACE:
29    subroutine MAPL_CFIOClose( MCFIO, RC )
30
31      !ARGUMENTS:
32      type(MAPL_CFIO),             intent(INOUT) :: MCFIO ! brief description
33      integer, optional,           intent(  OUT) :: RC    ! brief description
34
35      !DESCRIPTION:
36  !   Not a full destroy; only closes the file.
37
```

```
38       !EOPI
39       ... rest of the code
40
41     end subroutine MAPL_CFIOClose
```

The result is shown in Figure (1).



1.1 MAPL_CFIOClose — Close file in MAPL CFIO Object

INTERFACE:

```
   subroutine MAPL_CFIOClose( MCFIO, RC )
```

ARGUMENTS:

```
     type(MAPL_CFIO),            intent(INOUT) :: MCFIO ! brief description
     integer, optional,          intent(  OUT) :: RC    ! brief description
```

DESCRIPTION:

Not a full destroy; only closes the file.

Figure 1: ProTEX-ed version of MAPL_CFIOClose

4 Overloaded routines

ProTEX uses the tags `!IROTUINE` and `!IIROUTINE` to document overloaded methods. It is important to code the individual methods (that are part of the generic interface) consecutively. The individual methods are tagged with `!IIROUTINE` (lines 44, 68) instead of `!IROUTINE`), `!INERFACE`, `!ARGUMENTS` and `!DESCRIPTION`. The first of these routine includes an additional `!IROUTINE` tag (line 43) that defines the generic interface name. In this example, the generic interface is called `MAPL_CFIOCreate` that overloads two individual routines, the first of which is documented as follows:

```
42     !BOPI
43     ! !IROUTINE: MAPL_CFIOCreate --- Creates a MAPL CFIO Object
44     ! !IIROUTINE: MAPL_CFIOCreateFromBundle --- Creates MAPL CFIO Object from a Bundle
45
46     !INTERFACE:
47     subroutine MAPL_CFIOCreateFromBundle( MCFIO, NAME, CLOCK, BUNDLE,          &
48                                           OFFSET, RESOLUTION, FREQUENCY,       &
49                                           LEVELS, DESCR, XYOFFSET, VCOORD,     &
50                                           VUNIT, VSCALE, SOURCE, INSTITUTION,  &
51                                           COMMENT, CONTACT, FORMAT, EXPID,     &
52                                           DEFLATE, GC, ORDER, NumCores, nbits, &
53                                           RC )
54
55       !ARGUMENTS:
    type(MAPL_CFIO),             intent(OUT)   :: MCFIO ! brief description
    character(LEN=*),            intent(IN)    :: NAME  ! brief description
    ... more arguments

    !DESCRIPTION:
!   Description of this routine and arguments in more detail

    !EOPI
    ... rest of the code

  end subroutine MAPL_CFIOCreateBundle
```

The second routine is documented as

```
  !BOPI
  ! !IIROUTINE: MAPL_CFIOCreateFromState --- Creates MAPL CFIO Object from a State

  !INTERFACE:
  subroutine MAPL_CFIOCreateFromState ( MCFIO, NAME, CLOCK, STATE, OFFSET,   &
                                        RESOLUTION, LEVELS, DESCR, BUNDLE,   &
                                        XYOFFSET, VCOORD, VUNIT, VSCALE,     &
                                        SOURCE, INSTITUTION, COMMENT,        &
                                        CONTACT, FORMAT, EXPID, DEFLATE, GC, &
                                        ORDER, NumCores, nbits, &
                                        RC )

    !ARGUMENTS:
    type(MAPL_CFIO),             intent(OUT) :: MCFIO  ! brief description
    character(LEN=*),            intent(IN)  :: NAME   ! brief description
    ... more arguments

    !DESCRIPTION:
!   more detailed description

    !EOPI
    ... rest of the code

  end subroutine MAPL_CFIOCreateFromState
```

The result is is shown in Figure (2).

5 Spec calls for Import/Export/Internal states

The spec calls for Import/Export/Internal states need to be enclosed between !BOS/!EOS for documentation. The call to MAPL_AddXXXXSpec should have the form

```
call MAPL_AddExportSpec(GC,                                                  &
                        SHORT_NAME = 'KE',                                   &
```

```
                                LONG_NAME  = 'vertically_integrated_kinetic_energy', &
                                UNITS      = 'J m-2',                                &
                                DIMS       = MAPL_DimsHorzOnly,                      &
                                VLOCATION  = MAPL_VLocationNone,                     &
                                RC         = STATUS)
```

Note:

- The state variables are listed in a table (LaTeX longtable spanning multiple pages).
- **Multiple arguments** (e.g. `UNITS` and `DIMS`) in a single line would lead to errors.
- **UNITS** should be *LaTeX ready.*
  1. m/s^2 is represented as ‘`m s-2`’. ‘`a+2 b-4`’ stands for a^2b^{-4}.
  2. p^κ is represented as ‘`p$^\kappa$`’.
- **LONG NAME** should be such that after splitting with respect to ‘_’, it is a valid LaTeX statement and is printed as is. For example, the long name ‘`mid_layer_$p ^\kappa$`’ is printed as ‘`mid layer` p^κ’.

6 Resources

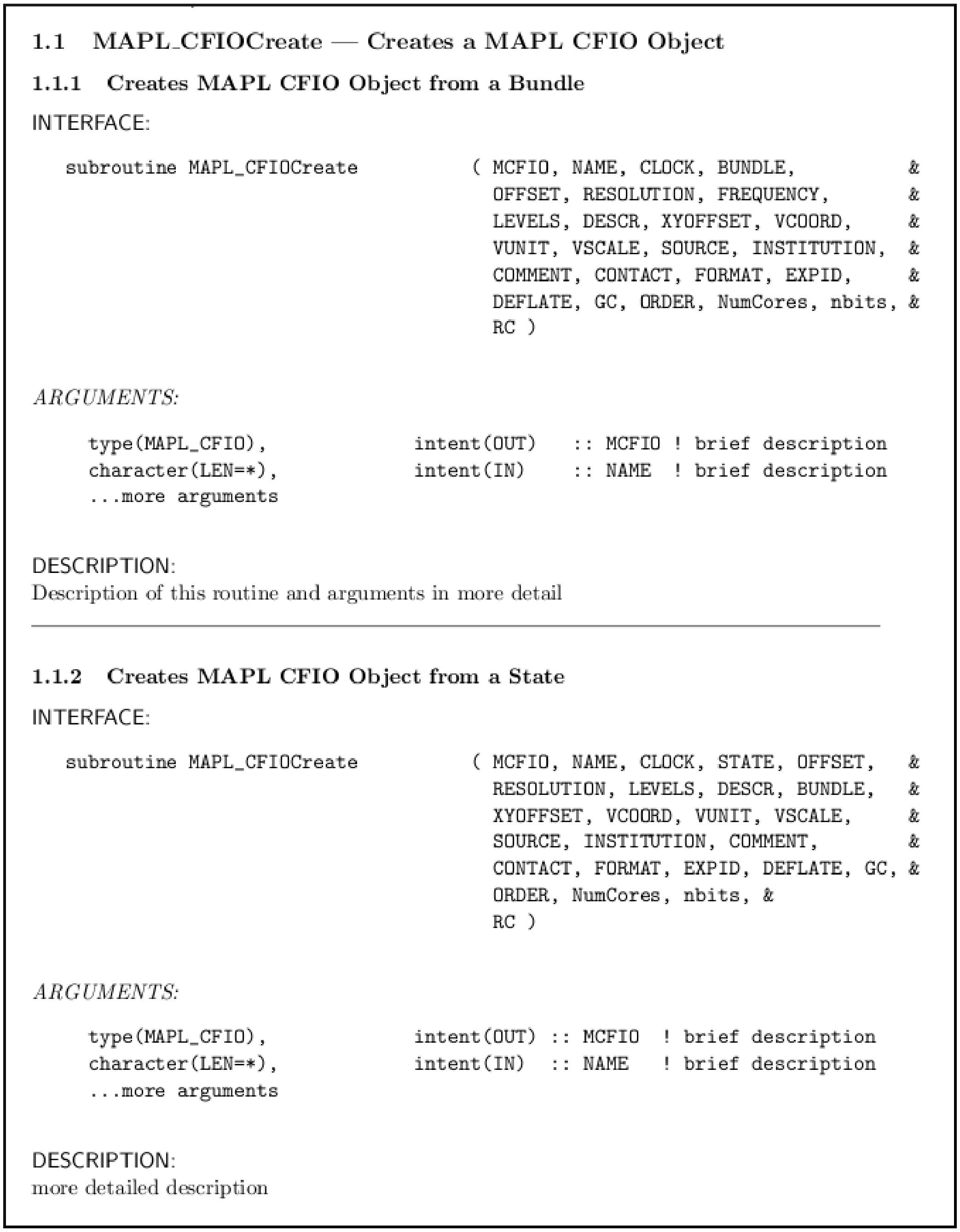
To document the resources, calls to `MAPL_GetResource` needs to be enclosed between `!BOR`/`!EOR`. For correct documentation, protex needs two lines: (1) comment line starting with the keyword `!RESOURCE_ITEM:` followed by the actual call to `MAPL_GetResource`.

```
!BOR
!RESOURCE_ITEM: K :: Value of isothermal temperature on coldstart
call MAPL_GetResource ( MAPL, T0, 'T0:', default=300., RC=STATUS )
!EOR
```

From the `!RESOURCE_ITEM` line, protex reads the unit (K) and the description (Value of isothermal temperature on coldstart). From the call line, protex reads the name (label) and its default value. The call line should NOT be continued to the next line(s).

7 Code

Any code fragment in addition to the ones already mentioned can be documented (as LaTeX verbatim) using a `!BOC`/`!EOC` block.



1.1 MAPL_CFIOCreate — Creates a MAPL CFIO Object

1.1.1 Creates MAPL CFIO Object from a Bundle

INTERFACE:

```
   subroutine MAPL_CFIOCreate          ( MCFIO, NAME, CLOCK, BUNDLE,          &
                                         OFFSET, RESOLUTION, FREQUENCY,       &
                                         LEVELS, DESCR, XYOFFSET, VCOORD,     &
                                         VUNIT, VSCALE, SOURCE, INSTITUTION,  &
                                         COMMENT, CONTACT, FORMAT, EXPID,     &
                                         DEFLATE, GC, ORDER, NumCores, nbits, &
                                         RC )
```

ARGUMENTS:

```
     type(MAPL_CFIO),             intent(OUT)   :: MCFIO ! brief description
     character(LEN=*),            intent(IN)    :: NAME  ! brief description
     ...more arguments
```

DESCRIPTION:
Description of this routine and arguments in more detail

1.1.2 Creates MAPL CFIO Object from a State

INTERFACE:

```
   subroutine MAPL_CFIOCreate          ( MCFIO, NAME, CLOCK, STATE, OFFSET,   &
                                         RESOLUTION, LEVELS, DESCR, BUNDLE,   &
                                         XYOFFSET, VCOORD, VUNIT, VSCALE,     &
                                         SOURCE, INSTITUTION, COMMENT,        &
                                         CONTACT, FORMAT, EXPID, DEFLATE, GC, &
                                         ORDER, NumCores, nbits, &
                                         RC )
```

ARGUMENTS:

```
     type(MAPL_CFIO),             intent(OUT) :: MCFIO  ! brief description
     character(LEN=*),            intent(IN)  :: NAME   ! brief description
     ...more arguments
```

DESCRIPTION:
more detailed description

Figure 2: ProTEX-ed version of MAPL_CFIOCreate