

## Code Management for SSC Detector Development

**Frank E. Paige**

Physics Department  
Brookhaven National Laboratory  
Upton, NY 11973

Code management is crucial for maintaining large programs. Unfortunately, there is no good machine-independent system for doing it. To maintain ISAJET, therefore, we have adopted a hybrid system: the code is distributed to users in CERN's Patchy format, but the actual maintainance is done with DEC's Code Management System (CMS) on a VAX.

Patchy is a very old-fashioned system, but it is maintained by CERN for virtually all computers, and it is available free. It provides facilities for handling common blocks and other common material and for inserting machine dependent code. Patchy normally stores all of the source material in a single file, although more than one file can be used. The binary version of this file is called a PAM, and the card image version is called a CARDPAM. The two contain the same information, and utility programs are available to convert one into the other. Within the PAM lines of code are organized into decks, which in turn are grouped into patches. The lines of code are interspersed with Patchy control lines, which always start with a + in column one. The program YPATCHY reads the PAM and creates a Fortran or other file from it. For ISAJET we make each subroutine a deck, and we use patches for groups of subroutines which are logically related. We also include the documentation in the PAM, allowing it to refer to the common blocks.

Since Patchy is universally available and is used to maintain GEANT, it seems a natural choice for maintaining other physics and detector simulation programs.

We do not use Patchy for making changes to ISAJET. As a result, only a few Patchy commands appear in ISAJET.CAR:

+PATCH, (name)	Defines a group of decks. In ISAJET the main logical groups such as the event generation code, the analysis code, and the documentation are patches. Small patches are used for controlling the selection.
+DECK, (name)	Defines a deck. In ISAJET each subroutine is in a separate deck with the same name. Decks are also used for parts of the documentation and for special groups of common blocks.
+EOD	Ends a deck or patch.
+KEEP, (name)	Defines a common block or similar material.
+CDE, (name)	Inserts a previously defined KEEP of the same name.
+SELF, IF=(name)	Begins conditional material, which can be selected by +USE, (name). This is terminated by another Patchy command, including a +SELF.

The following simple CARDPAM illustrates all the major features of Patchy used in ISAJET.PAM:

```

+PATCH,EXCOM.  A COMMENT CAN FOLLOW THE .
+KEEP,ACON.
      COMMON/ACON/C1,C2
+KEEP,LUNS.
      LUN1=39
+PATCH,EXAMPLE.
+DECK,EXAMPLE.
C      Calculate black body flux per unit wavelength interval, for
C      wavelength in cm, temperature in degrees Kelvin.
+CDE,ACON.
+CDE,LUNS.
+SELF,IF=VAX
      OPEN(UNIT=LUN1,FILE='ACON.DAT',STATUS='OLD')
+SELF,IF=IBM
      OPEN(UNIT=1,ACCESS='DIRECT',RECL=80,STATUS='OLD')
+SELF
      READ(LUN1,'(1X,2F15.7)')C1,C2
      TEMP = 6000.0
      DO 1 J=1,10
          ALAM = 5.0*E-7 + 1.0E-7*(J-1)
          F = BB(TEMP,ALAM)

```

```

        WRITE(6,'('' Temp,Lambda,F(Lambda)='',1P3E12.5)')
    $  TEMP,ALAM,F
1    CONTINUE
    STOP
    END
+EOD.
+DECK,BB.
    FUNCTION BB(T,AL)
+CDE,ACON.
    BB = EXP(C2/(AL*T)) - 1.0
    BB = C1/BB
    RETURN
    END
+EOD

```

Note that common blocks and code are kept in separate patches, that each subroutine is in a separate deck, and that each common block is in a separate +KEEP deck and is inserted with +CDE as needed. This organization is not required but it seems desirable. For example, common block ACON can be changed in one place, and Patchy will propagate the change throughout the program.

YPATCHY is the Patchy program which reads the PAM and produces the desired output file, a Fortran source file for the above example. It is controlled by commands read from an input file called a CRADLE. A confusing but useful feature is that the CRADLE contains two types of selection. +USE, (name) causes Patchy to consider patch (name) in creating the output file, while +EXE, (name) actually causes (name) to be written out. One should normally +USE select all of the Fortran and common blocks even if one wants to write out only a single deck. +USE is also used to select IF+ flags. Thus the CRADLE to write out all the routines for the above example for a VAX is

```

+USE,EXCOM,EXAMPLE,VAX.
+EXE.
+PAM.
+QUIT.

```

To write out just function BB, you must still +USE select everything, but you +EXE select only one deck:

```

+USE,EXCOM,EXAMPLE.
+USE,IBM.

```

```
+EXE,PATCH=EXAMPLE,DECK=BB.
+PAM.
+QUIT.
```

Note that there can be several +USE or +EXE commands.

For complicated programs it is convenient to define "pilot patches" to select all the pieces needed for a given purpose, for example all the source or all the documentation. These traditionally have names beginning with \*. A suitable pilot patch for the above example is

```
+PATCH,*EXAMPLE.
+USE,EXCOM.
+USE,EXAMPLE.
+EOD.
```

If the above patch is added at the beginning of the CARDPAM, then one can use select all the Fortran with

```
+USE,*EXAMPLE
```

While this is a trivial saving here, it can be quite helpful for complex programs.

Similar patches can be used to select machine dependent features. For example in ISAJET we need double precision for some calculations on 32-bit computers. The CARDPAM contains code like

```
+SELF,IF=SINGLE.
    REAL A,B,C
+SELF,IF=DOUBLE.
    DOUBLE PRECISION A,B,C
+SELF.
```

Then we have

```
+PATCH,CRAY.
+USE,SINGLE.
...
+EOD
+PATCH,VAX.
+USE,DOUBLE.
...
+EOD.
```

Saying `+USE,VAX` selects the `IF=DOUBLE` material and also any special VAX material such as

```
+SELF,IF=VAX.
      ITMLST(1)=ISHFT(%LOC(JPI$_CPUTIM),16)+4
+SELF.
```

Again this organization is not mandatory, but it greatly simplifies making a version for a new computer.

While Patchy is quite convenient for distributing versions of a program, it is exceedingly clumsy for actually making changes. The commands to do this are line oriented, and the syntax for conditional material becomes awkward in correction patches. Nor is it practical to make changes by editing the entire CARDPAM as a single file. Therefore we now do the actual code maintenance using CMS on a VAX. This CMS system is internal and is never seen by outside users. In CMS the CARDPAM is broken up into decks, each of which is a CMS element and corresponds to a single subroutine. These are kept in a CMS library with an access control list. To modify a subroutine we first reserve it, preventing anyone else from modifying it:

```
CMS RESERVE <name> "<comment>"
```

This creates a file `<name>.CAR` in the current directory with the form

```
+DECK,<name>
      SUBROUTINE <name>
      ...
      END
+EOD
```

After editing this file, we can generate a Fortran file for testing with a procedure `MAKFOR`, which is written in DCL and uses Patchy to expand the `+CDE` and other Patchy control statements. Finally, when we are satisfied with the new version, we return it to the CMS with

```
CMS REPLACE <name> "<comment>"
```

When a new version is to be released, another procedure fetches all the individual files from the CMS library and combines them into a new CARDPAM. It also assembles the

changed files into a cradle which can be used with YEDIT, another Patchy utility program, to make a new PAM file from the old one. At present this procedure is not fully automatic, but we hope to make it so in the near future.

The combination of CMS and Patchy is admittedly not perfect, but we feel it is the best system available for supporting a large program which is run by a diverse group of users on many computers.