SLAC-PUB-3050
February 1983
(N)

CMS Nucleus Extensions in a DCSS *

R.S. Miller

R.H. Johnson

Stanford Linear Accelerator Center
P.O. Box 4349, Bin 97
Stanford, California 94305

February 25, 1983

One of the many useful commands that IBM provided in VM/SP2 was the NUCXLOAD command. This command can be used to extend the CMS nucleus by loading code into virtual memory and retaining it there. This allows commands to be executed in virtual memory without first having CMS load them into the user or transient area.
Several SP2 functions, such as EXECIO, come as nucleus extensions. There is one serious problem with the NUCX function however.Each user has a copy of the NUCX loaded code in his own virtual memory.

In our installation each user typically has about 48K of NUCX loaded code in virtual memory. Thus, for each user, there are 6 pages that must be managed by CP for that user. Our system usually has about 300 virtual machines during first shift operation. This means 6*300=1800 pages of NUCX loaded code! That is almost enough to fill up one 2305 drum. Wouldn't it be nice if one could change CMS so that NUCX loaded code could be shared in a DCSS?

Roy Miller at SLAC has recently changed our version of CMS so this is possible.The NUCX code that comes from IBM already has part of the desired capability. If NUCX loaded code has a zero length, then CMS will ignore it when cleaning up storage at ABEND time. IBM uses this technique to make EXECIO run as a nucleus extension. This means that if one can make NUCX think that code it

* Work supported by the Dept. of Energy, contract DE-AC03-76F00515

(Presented at the Annual SHARE Conference 1983, San Francisco, CA February 20-25, 1983)

knows about is in a DCSS, then CMS will be quite happy to use it and keep it around across ABENDs.

In order to load and execute NUCX code a new command had to be written. This command is called "SHRDCODE". SHRDCODE can load and attach a DCSS containing NUCX code. Currently SHRDCODE can load from either a MODULE file or a LOADLIB. At CMS IPL time, SHRDCODE tries to ATTACH the segment containing the NUCX code. If it gets the segment, NUCXLOAD entries are made for all modules in the segment. If the segment cannot be obtained, conventional NUCXLOADs are done for all modules that are in the segment. SHRDCODE produces a LOAD MAP identical to that produced by the LOAD command.

The format of the SHRDCODE command is:

```
 SHRDCODE   ATTACH     <segment> <source>
            GENERATE   <segment> <source>
```

SHRDCODE EXEC and SHRDCODE MODULE are used to generate and attach named segments containing re-entrant programs.

Where:

<segment>
: is the name of the named segment to be generated or attached. This segment must load outside of the defined storage of the virtual machine.

<source>
: is the filename of the input file to be used. This file, which has the name '<source> SEGMENT A' contains records describing the source and attributes of each program which is to be contained in the named named segment.

The source file which has the filename '<source> SEGMENT *', contains a list describing which programs are to be included in the segment and their origin and attributes. i.e.

Our current source file looks like this:

```
**********************************************************
*                                                        *
* * Format     Cmdname Fn     Ft      Fm Member  Options  *
*  LOADMODULE CMSINFO SCSSYS LOADLIB *  CMSINFO ( SYSTEM *
*  LOADMODULE EXEC    SCSSYS LOADLIB *  REXSYS  ( SYSTEM *
*  MODULE     ABEND   ABEND  MODULE  *          ( SYSTEM *
*                                                        *
**********************************************************
```

Format	program type (LOADMODULE or MODULE)
Cmdname	name by which program is to be invoked
Fn	filename of program
Ft	filetype of program. This should be LOADLIB for a member contained within a LOADLIB and MODULE for a CMS disk resident module.
Fm	filemode of program
Member	member name of program if LOADMODULE
Options	SERVICE or SYSTEM
SERVICE	this attribute indicates that the program is to be called during CMS ABEND processing.
SYSTEM	this attribute indicates that the program is to be called called disabled, protect key 0. It also prevents CMS from purging the entry during CMS ABEND processing. This attribute is required for shared code.

As the code for SHRDCODE was being written, several shortcomings in the NUCX support became apparent. Some of the problems we ran into were:

1) The key for the DCSS storage must be 'F' rather than 'D', otherwise code in the DCSS will be NUCXDROPped at each HX.

2) The code in the DCSS must all be defined with the 'SYSTEM' option. This means that when the code gets control, it will be running in key 0, disabled. If any program checks occur in such code, CMS will crash. So any code you put in a DCSS had better be stable.

The above is symptomatic of the problem that permanent resident status is needlessly tied to the SYSTEM option. It would be much nicer if there were four sets of options that allowed one to specify whether the code was to:

a. Run in user or system key.
b. Remain NUCXLOADed after an HX.
c. Run enabled or disabled.
d. Run in user or supervisor state.

3) The NUCXLOAD support for modules is primitive. NUCXLOAD will only read the first record of a module, and the entry point for the module is assumed to be the start of the module. In addition, the module must have no internal ADCONs. It can be a sizeable task to take a typical transient area module and turn it into a reentrant module with no ADCONs! You have to use LINEDIT instead of WRTERM, you must use the BUFFER=(n) form for FSREAD/WRITE, you must watch for AL4(*+4), and so on. It can be done, but it is tedious work.