

```

EEEEEEEEEE XX XX AAAAAA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSSS
EEEEEEEEEE XX XX AAAAAA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSSS
EEEEEEEEEE XX XX AAAAAA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSSS
EE XX XX AA AA MMMM MMMM PP PP LL EE SS
EE XX XX AA AA MMMM MMMM PP PP LL EE SS
EE XX XX AA AA MMMM MMMM PP PP LL EE SS
EE XX XX AA AA MM MM MM PP PP LL EE SS
EE XX XX AA AA MM MM MM PP PP LL EE SS
EEEEEEEE XX XX AA AA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSS
EEEEEEEE XX XX AA AA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSS
EEEEEEEE XX XX AA AA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSS
EE XX XX AAAAAAAAAA MM MM PP LL EE SS
EE XX XX AAAAAAAAAA MM MM PP LL EE SS
EE XX XX AAAAAAAAAA MM MM PP LL EE SS
EE XX XX AA AA MM MM PP LL EE SS
EE XX XX AA AA MM MM PP LL EE SS
EE XX XX AA AA MM MM PP LL EE SS
EEEEEEEEEE XX XX AA AA MM MM PP LLLLLLLLLL EEEEEEEEE SSSSSSSS
EEEEEEEEEE XX XX AA AA MM MM PP LLLLLLLLLL EEEEEEEEE SSSSSSSS
EEEEEEEEEE XX XX AA AA MM MM PP LLLLLLLLLL EEEEEEEEE SSSSSSSS

```



```

UU      UU      SSSSSSSS  SSSSSSSS  DDDDDDDD  IIIIII  SSSSSSSS  PPPPPPPP
UU      UU      SSSSSSSS  SSSSSSSS  DDDDDDDD  IIIIII  SSSSSSSS  PPPPPPPP
UU      UU      SS        SS        DD        DD        SS        PP        PP
UU      UU      SS        SS        DD        DD        SS        PP        PP
UU      UU      SS        SS        DD        DD        SS        PP        PP
UU      UU      SS        SS        DD        DD        SS        PP        PP
UU      UU      SSSSSS    SSSSSS    DD        DD        SS        PPPPPPPP
UU      UU      SSSSSS    SSSSSS    DD        DD        SS        PPPPPPPP
UU      UU      SS        SS        DD        DD        SS        PP
UU      UU      SS        SS        DD        DD        SS        PP
UU      UU      SS        SS        DD        DD        SS        PP
UUUUUUUUUU  SSSSSSSS  SSSSSSSS  DDDDDDDD  IIIIII  SSSSSSSS  PP
UUUUUUUUUU  SSSSSSSS  SSSSSSSS  DDDDDDDD  IIIIII  SSSSSSSS  PP

```

....
....
....
....

```

MM      MM      AAAAAA  RRRRRRRR
MM      MM      AAAAAA  RRRRRRRR
MMMM    MMMM  AA        AA  RR        RR
MMMM    MMMM  AA        AA  RR        RR
MM      MM      AA        AA  RR        RR
MM      MM      AA        AA  RR        RR
MM      MM      AA        AA  RRRRRRRR
MM      MM      AA        AA  RRRRRRRR
MM      MM      AAAAAAAAAA  RR  RR
MM      MM      AAAAAAAAAA  RR  RR
MM      MM      AA        AA  RR  RR

```

.TITLE USER_SYS_DISP - Example of user system service dispatcher
 .IDENT 'V04=001'

```

*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****
    
```

Facility: Example of User Written System Services

++ Abstract:

This module contains an example dispatcher for user written system services along with several sample services and a user rundown example. It is a template intend to serve as the starting point for implementing a privileged shareable image containing your own services. When used as a template, the definitions and code for the sample services should be removed.

Overview:

User written system services are contained in privileged shareable images that are linked into user program images in exactly the same fashion as any shareable image. The creation and installation of a privileged, shareable image is slightly different from that of an ordinary shareable image. These differences are:

1. A vector defining the entry points and providing other control information to the image activator. This vector is a the lowest address in an image section with the VEC attribute.
2. The shareable image is linked with the /PROTECT option that marks all of the image sections so that they will be protected and given EXEC mode ownership by the image activator.
3. The shareable image MUST be installed /SHARE /PROTECT with the INSTALL utility in order for the image activator to connect the privileged shareable image to the change mode

dispatchers.

A privileged shareable image implementing user written system services is comprised of the following major components:

1. A transfer vector containing all of the entry points and collecting them at the lowest virtual address in the shareable image. This formalism enables revision of the shareable image without necessitating the relinking of images that use it.
2. A Privileged Library Vector in a PSECT with the VEC attribute that describes the entry points for dispatching EXEC and KERNEL mode services along with validation information.
3. A dispatcher for kernel mode services. This code will be called by the VMS change mode dispatcher when it fails to recognize a kernel mode service request.
4. A dispatcher for executive mode services. This code will be called by the VMS change mode dispatcher when it fails to recognize an executive mode service request.
5. Service routines to perform the various services.

The first four components are contained in this template and are most easily implemented in MACRO, while the service routines can be implemented in BLISS or MACRO. Other languages may be usable but are not recommended -- particularly if they require runtime support routines or are extravagant in their use of stack or are unable to generate PIC code.

This example is position-independent (PIC) and it is good practice to implement shareable images this way whenever possible.

Revision History:

V04-001	WMC0001	Wayne Cardoza	06-Sep-1984
		Make system version a weak reference.	
V03-002	KDM0074	Kathleen D. Morse	23-Aug-1983
		Use cpu-dependent routine to get the TODR value. Add \$\$\$DEF and remove \$PRDEF.	
V03-001	ACG0001	Andrew C. Goldstein	23-May-1983
		Fix change-mode dispatcher to clean two extra longwords off stack before calling user system service.	

Link Command File Example:

```

$!
$! Command file to link User System Service example.
$!
$ LINK/PROTECT/NOSYSSHR/SHARE=USS/MAP=USS/FULL SYS$INPUT/OPTIONS
$!

```

```
Options file for the link of User System Service example.
```

```
SYSS$SYSTEM:SYS.STB/SELECTIVE
```

```
Create a separate cluster for the transfer vector.
```

```
CLUSTER=TRANSFER_VECTOR,,,SYSS$DISK:[]USSDISP
```

```
GSMATCH=LEQUAL,1,1
```

```
.PAGE
```

```
.SBTTL Declarations and Equates
```

```
Include Files
```

```
.LIBRARY "SYSS$LIBRARY:LIB.MLB" ; Macro library for system structure
; definitions
```

```
Macro Definitions
```

```
DEFINE_SERVICE - A macro to make the appropriate entries in several
different PSECTS required to define an EXEC or KERNEL
mode service. These include the transfer vector,
the case table for dispatching, and a table containing
the number of required arguments.
```

```
DEFINE_SERVICE Name,Number_of_Arguments,Mode
```

```
.MACRO DEFINE_SERVICE,NAME,NARG=0,MODE=KERNEL
```

```
.PSECT $$$TRANSFER_VECTOR,PAGE,NOWRT,EXE,PIC
```

```
.ALIGN QUAD ; Align entry points for speed and style
```

```
.TRANSFER NAME ; Define name as universal symbol for entry
```

```
.MASK NAME ; Use entry mask defined in main routine
```

```
.IF IDN MODE,KERNEL
```

```
CHMK #<KCODE_BASE+KERNEL_COUNTER> ; Change to kernel mode and execute
```

```
RET ; Return
```

```
KERNEL_COUNTER=KERNEL_COUNTER+1 ; Advance counter
```

```
.PSECT KERNEL_NARG,BYTE,NOWRT,EXE,PIC
```

```
.BYTE NARG ; Define number of required arguments
```

```
.PSECT USER_KERNEL_DISP1,BYTE,NOWRT,EXE,PIC
```

```
.WORD 2+NAME-KCASE_BASE ; Make entry in kernel mode CASE table
```

```
.IFF
```

```
CHME #<ECODE_BASE+EXEC_COUNTER> ; Change to executive mode and execute
```

```
RET ; Return
```

```
EXEC_COUNTER=EXEC_COUNTER+1 ; Advance counter
```

```
.PSECT EXEC_NARG,BYTE,NOWRT,EXE,PIC
```

```
.BYTE NARG ; Define number of required arguments
```

```
.PSECT USER_EXEC_DISP1,BYTE,NOWRT,EXE,PIC
```

```
.WORD 2+NAME-ECASE_BASE ; Make entry in exec mode CASE table
```

```

        .ENDC
        .ENDM  DEFINE_SERVICE      :
:
: Equated Symbols
:
$PHDDEF      : Define process header offsets
$PLVDEF      : Define PLV offsets and values
$SSDEF       : Define system status codes
:
Initialize counters for change mode dispatching codes
:
KERNEL_COUNTER=0      : Kernel code counter
EXEC_COUNTER=0        : Exec code counter
:
Own Storage
:
.PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
KERNEL_NARG:      : Base of byte table containing the
:                 : number of required arguments.
.PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
EXEC_NARG:        : Base of byte table containing the
:                 : number of required arguments.
.PAGE
.SBTTL  Transfer Vector and Service Definitions
:++
: The use of transfer vectors to effect entry to the user written system services
: enables some updating of the shareable image containing them without necessitating
: a re-link of all programs that call them. The PSECT containing the transfer
: vector will be positioned at the lowest virtual address in the shareable image
: and so long as the transfer vector is not re-ordered, programs linked with
: one version of the shareable image will continue to work with the next.
:
: Thus as additional services are added to a privileged shareable image, their
: definitions should be added to the end of the following list to ensure that
: programs using previous versions of it will not need to be re-linked.
: To completely avoid relinking existing programs the size of the privileged
: shareable image must not change so some padding will be required to provide
: the opportunity for future growth.
:--
DEFINE_SERVICE  USER_GET_TODR,1,KERNEL  : Service to get value of time
:                                     : of day register
DEFINE_SERVICE  USER_SET_PFC,2,KERNEL  : Service to set value of process
:                                     : default pagefault cluster
DEFINE_SERVICE  USER_NULL,0,EXEC       : Null exec service
:
: The base values used to generate the dispatching codes should be negative for
: user services and must be chosen to avoid overlap with any other privileged
: shareable images that will be used concurrently. Their definition is
: deferred to this point in the assembly to cause their use in the preceding
: macro calls to be forward references that guarantee the size of the change
: mode instructions to be four bytes. This satisfies an assumption that is
: made by for services that have to wait and be retried. The PC for retrying
: the change mode instruction that invokes the service is assumed to be 4 bytes

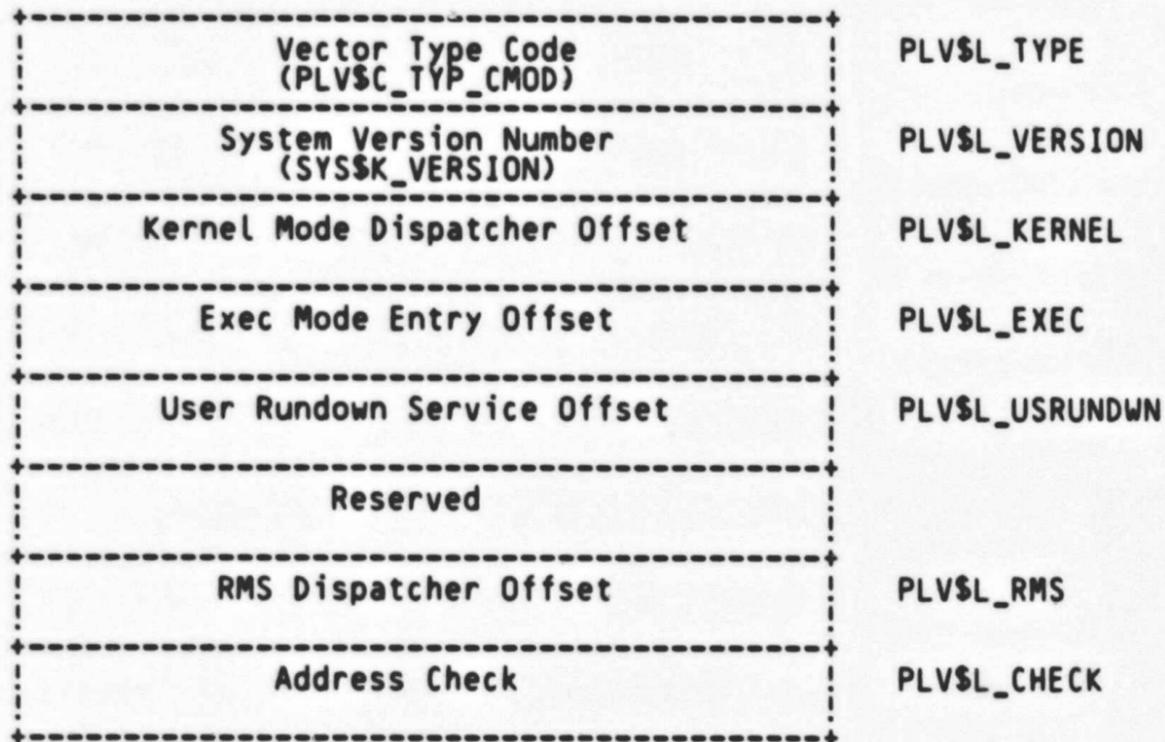
```

; less than that saved in the change mode exception frame. Of course, the particular
 ; service routine determines whether this is possible.

```
KCODE_BASE=-1024      ; Base CHMK code value for these services
ECODE_BASE=-1024     ; Base CHME code value for these services
.PAGE
.SBTTL Change Mode Dispatcher Vector Block
```

++ This vector is used by the image activator to connect the privileged shareable
 ; image to the VMS change mode dispatcher. The offsets in the vector are self-
 ; relative to enable the construction of position independent images. The system
 ; version number will be used by the image activator to verify that this shareable
 ; image was linked with the symbol table for the current system.

Change Mode Vector Format



The reference to SYSS\$K_VERSION will only be resolved if the image is
 linked against SYS.STB. In other cases the version check is
 unnecessary and will not be done.

```
.WEAK SYSS$K_VERSION
.PSECT USER_SERVICES,PAGE,VEC,PIC,NOWRT,EXE
.LONG PLV$C_TYP_CM0D      ; Set type of vector to change mode dispatcher
.LONG SYSS$K_VERSION     ; Identify system version
.LONG KERNEC_DISPATCH-.  ; Offset to kernel mode dispatcher
.LONG EXEC_DISPATCH-.    ; Offset to executive mode dispatcher
.LONG USER_RUNDOWN-.     ; Offset to user rundown service
```

```
.LONG 0      : Reserved.
.LONG 0      : No RMS dispatcher
.LONG 0      : Address check - PIC image
.PAGE
.SBTTL Kernel Mode Dispatcher
```

++
: Input Parameters:

```
(SP) - Return address if bad change mode value
R0  - Change mode argument value.
R4  - Current PCB Address. (Therefore R4 must be specified in all
      register save masks for kernel routines.)
AP  - Argument pointer existing when the change
      mode instruction was executed.
FP  - Address of minimal call frame to exit
      the change mode dispatcher and return to
      the original mode.
```

```
--
.PSECT USER_KERNEL_DISP0, BYTE, NOWRT, EXE, PIC
KACCVIO: MOVZWL #SS$_ACCVIO, R0      : Kernel access violation
          RET                               : Set access violation status code
          : and return
KINSFARG: MOVZWL #SS$_INSFARG, R0    : Kernel insufficient arguments.
          RET                               : Set status code and
          : return
KNOTME:  RSB                               : RSB to forward request

KERNEL_DISPATCH::                          : Entry to dispatcher
MOVAB W^=KCODE_BASE(R0), R1                : Normalize dispatch code value
BLSS KNOTME                                 : Branch if code value too low
CMPW R1, #KERNEL_COUNTER                   : Check high limit
BGEQU KNOTME                                : Branch if out of range
```

```
: The dispatch code has now been verified as being handled by this dispatcher,
: now the argument list will be probed and the required number of arguments
: verified.
```

```
MOVZBL W^KERNEL_NARG[R1], R1               : Get required argument count
MOVAL @#4[R1], R1                           : Compute byte count including arg count
IFNORD R1, (AP), KACCVIO                     : Branch if arglist not readable
CMPB (AP), W^<KERNEL_NARG-KCODE_BASE>[R0] : Check for required number
BLSSU KINSFARG                               : of arguments
MOVL FP, SP                                  : Reset stack for service routine
CASEW R0, -                                  : Case on change mode
          -                                  : argument value
          #KCODE_BASE, -                     : Base value
          #<KERNEL_COUNTER-1>               : Limit value (number of entries)
KCASE_BASE:                                 : Case table base address for DEFINE_SERVICE
```

```
: Case table entries are made in the PSECT USER_KERNEL_DISP1 by
: invocations of the DEFINE_SERVICE macro. The three PSECTS,
: USER_KERNEL_DISP0, 1, 2 will be abutted in lexical order at link-time.
```

XAD

SDE

: E

SDE

SDE

```

.PSECT USER_KERNEL_DISP2,BYTE,NOWRT,EXE,PIC
BUG_CHECK IVSSRVRQST,FATAL      ; Since the change mode code is validated
                                ; above, we should never get here

```

```

.PAGE
.SBTTL  Executive Mode Dispatcher

```

```

++
: Input Parameters:

```

- (SP) - Return address if bad change mode value
- RO - Change mode argument value.
- AP - Argument pointer existing when the change mode instruction was executed.
- FP - Address of minimal call frame to exit the change mode dispatcher and return to the original mode.

```

--
EACCVIO: .PSECT USER_EXEC_DISPO,BYTE,NOWRT,EXE,PIC
          MOVZWL #SS$_ACCVIO,RO      ; Exec access violation
          RET      ; Set access violation status code
          ; and return
EINSFARG: MOVZWL #SS$_INSFARG,RO     ; Exec insufficient arguments.
          RET      ; Set status code and
          ; return
ENOTME:  RSB      ; RSB to forward request

EXEC_DISPATCH::
          MOVAB W^ECODE_BASE(RO),R1  ; Entry to dispatcher
          BLSS ENOTME                ; Normalize dispatch code value
          CMPW R1,#EXEC_COUNTER      ; Branch if code value too low
          BGEQU ENOTME              ; Check high limit
          ; Branch if out of range

```

```

: The dispatch code has now been verified as being handled by this dispatcher,
: now the argument list will be probed and the required number of arguments
: verified.

```

```

          MOVZBL W^EXEC_NARG[R1],R1  ; Get required argument count
          MOVAL @#4[R1],R1           ; Compute byte count including arg count
          IFNORD R1,(AP),EACCVIO     ; Branch if arglist not readable
          CMPB (AP),W^<EXEC_NARG-ECODE_BASE>[RO] ; Check for required number
          BLSSU EINSFARG              ; of arguments
          MOVL FP,SP                 ; Reset stack for service routine
          CASEW RO,-                 ; Case on change mode
          -                          ; argument value
          #ECODE_BASE,-              ; Base value
          #<EXEC_COUNTER-1>          ; Limit value (number of entries)
ECASE_BASE: ; Case table base address for DEFINE_SERVICE

```

```

: Case table entries are made in the PSECT USER_EXEC_DISP1 by
: invocations of the DEFINE_SERVICE macro. The three PSECTS,
: USER_EXEC_DISPO,1,2 will be abutted in lexical order at link-time.

```

```

.PSECT USER_EXEC_DISP2,BYTE,NOWRT,EXE,PIC

```

XAD

: D

: D

: F

: XA_

```

BUG_CHECK IVSSRVRQST,FATAL      : Since the change mode code is validated
                                : above, we should never get here
.PAGE
.SBTTL  User Rundown Service

```

```

:++
: Functional description:
: This service is invoked from within the kernel mode system service
: that performs image rundown. It is invoked before any system
: rundown functions (i.e. deassign channels, release memory) are
: performed. User code should not invoked any RMS services or RTL
: routines, must not signal any exceptions. User code can invoke
: most system services except those that use RMS (e.g. $PUTMSG).

```

Calling sequence:

```

JSB  USER_RUNDOWN
Entered at IPL=0 and must leave at IPL=0.

```

Input Parameters:

```

R4 - Current PCB Address. (Therefore R4 must be specified in all
    register save masks for kernel routines.)

R7 - Access mode parameter to $RUNDWN maximized with previous mode

AP - Argument pointer existing when the $RUNDWN system
    service was invoked.

4(AP) - Access mode parameter to $RUNDWN

```

```

.PSECT  USER_CODE,BYTE,NOWRT,EXE,PIC

```

```

USER_RUNDOWN::
PUSHL  R2                : Entry point for service
PUSHAB B^SYSOUT          : Save a register
PUSHL  S^#SYS_LEN       : Set up address of descriptor
MOVAL  -(SP), R2        : Set up length
$ASSIGN_S 4(R2), (R2)    : Grab some temporary storage
BLBC   RO, 10$          : Assign a channel to operator console
$OUTPUT (R2), S^#MSG_LEN, B^MSG : Error
$DASSGN_S (R2)          : Print the message on operator console
10$:  ADDL2 #12, SP      : Get rid of the channel
      MOVL (SP)+, R2    : Clean up
      RSB              : Restore register

```

```

:SYSOUT: .ASCII /_OPA0:/
:SYS_LEN=-SYSOUT
:MSG: .ASCII /*** Image exiting ***/
:MSG_LEN=-MSG
.PAGE
.SBTTL  Get Time of Day Register Value

```

```

:++
: Functional Description:
: This routine reads the content of the hardware time of day
: processor register and stores the resulting value at the
: specified address.

```

```

: Input Parameters:
: 04(AP) - Address to return time of day value
: R4 - Address of current PCB

```

```

: Output Parameters:
: R0 - Completion Status Code

```

```

--
: .ENTRY USER_GET_TODR,^M<R2,R3,R4>
: MOVL 4(AP),R1 : Get address to store time of day register
: IFNOWRT #4,(R1),10$ : Branch if not writable
: JSB G^EXESREAD_TODR : Call cpu-dependent routine
: MOVL R0,(R1) : Return current time of day register
: MOVL #SS$_NORMAL,R0 : Set normal completion status
: RET : and return

10$: MOVZWL #SS$_ACCVIO,R0 : Indicate access violation
: RET
: .PAGE
: .SBTTL Set Page Fault Cluster Factor

```

```

:++
: Functional Description:
: This routine sets the page fault cluster to the specified value
: and returns the previous value.

```

```

: Input Parameters:
: 04(AP) - New value for Page Fault Cluster factor
: 08(AP) - Address to return previous value
: (0 means none)
: R4 - PCB address of current process

```

```

: Output Parameters:
: R0 - Completion Status code

```

```

--
: .ENTRY USER_SET_PFC,^M<R4,R5>
: MOVL @#CT[SGL_PHD,R5 : Get address of process header
: MOVL 8(AP),R1 : Get address to store previous value
: BEQL 10$ : Branch if none
: IFNOWRT #4,(R1),30$ : Branch if not writable
: MOVZBL PHD$B_DFPFC(R5),(R1) : Return current value
10$: MOVB 4(AP),R0 : Get new value for PFC
: CMPB R0,#127 : Check for legal value
: BLEQU 20$ : Branch if legal
: MOVB #127,R0 : Set to maximum value
20$: MOVB R0,PHD$B_DFPFC(R5) : Set new value into PHD
: MOVL #SS$_NORMAL,R0 : Set normal completion status
: RET : and return

30$: MOVZWL #SS$_ACCVIO,R0 : Indicate access violation
: RET

: .PAGE
: .SBTTL Null Service

```

```

:++
: Functional Description:

```

```

: Input Parameters:

```

:
: Output Parameters:
:--

```
.ENTRY USER_NULL,^M<>      : Entry definition
MOVZWL #SS$_NORMAL,RO      : Set normal completion status
RET                          : and return

.END
```

XAC

++
X

XA

2\$:
5\$:
10\$:

15\$:

The image displays a grid of 100 small terminal window screenshots, arranged in a 10x10 grid. Each window shows a different VAX/VMS command and its output. The windows are arranged in a grid, with some larger windows in the center and smaller ones around the edges. The text in the windows is mostly black on a white background, with some red text indicating errors or warnings. The windows show various outputs, including file attributes, directory listings, and command results. Some windows show the output of the 'TYPE' command for various files, showing their attributes and contents. Some windows show error messages or specific data outputs. The windows are arranged in a grid, with some larger windows in the center and smaller ones around the edges. The text in the windows is mostly black on a white background, with some red text indicating errors or warnings. The windows show various outputs, including file attributes, directory listings, and command results. Some windows show the output of the 'TYPE' command for various files, showing their attributes and contents. Some windows show error messages or specific data outputs.

Examples of visible text in the windows include:

- DRMULT B32
- DRMAST MAR
- ADDRIVER MAR
- TDRIVER MAR
- USSTEST MAR
- GBLSECUR0 MAR
- USSDISP MAR
- DOD_ERAPAT MAR
- LBRMAC MAR
- XADDRIVER MAR
- LABLOCIN MAR
- DRSLV MAR
- DTE_DF03 MAR
- SECRET MAR
- WORKO LIS
- EXAMPLES