A BATCH SYSTEM WITH
RAPID OVERLAY CAPABILITIES

by

David Henry Mueller

MASTER

December 1973

49035

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

312 1500

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

UIUCDCS-R-73-614

A BATCH SYSTEM WITH
RAPID OVERLAY CAPABILITIES

by

David Henry Mueller

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

December 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

* Supported in part by the Atomic Energy Commission under contract
US AEC AT(11-1)2383 and submitted in partial fulfillment of the
requirements of the Graduate College for the degree of Master of
Science in Computer Science.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

GG

Acknowledgments

The author would like to express his thanks to Dr. C. W. Gear and Dr. M. H. Pleck for their assistance in the work reported in this thesis, to Mrs. Barbara Armstrong and Mrs. Debra Sweet for their help in typing the manuscript, to Mr. Wayne Walker for doing the illustrations, and to Miss Renee Krasno for her help in proofreading the manuscript.

List of Figures	Page
2.1 System as Viewed by DOS:.....	5
2.2 System as Viewed by the Supervisor,.....	5
2.3 System as Viewed by a Monitor.....	7
3.1 Schematic of Overlay Table Structures.....	11

Table of Contents	Page
Acknowledgments.....	i
List of Figures.....	ii
1. Introduction.....	1
2. System Structure.....	4
3. Data Structures for the System.....	10
4. System Routines.....	15
5. An Example of System Execution.....	19
6. Conclusion.....	21
Bibliography.....	24
Appendix A The Supervisor	

1. INTRODUCTION

The Illinois Computing Graphics System (ICGS) Supervisor was written as a part of a project whose main purpose was to develop a graphics and simulation system on a minicomputer. The machine chosen for this was a PDP-11/20, a minicomputer with a very flexible addressing scheme. Also attached to the mainframe were 16K of core storage and a KELL-A Extended Arithmetic Element, which provides hardware fixed-point multiply and divide functions as well as fast logical and arithmetic multiple shift capabilities.

Direct access storage was provided by a RF11/RS11 disk unit and controller, a fixed-head disk with a storage capacity of 256K words. Mass storage was provided by a TC11/TU56 DECTape unit, a bidirectional magnetic tape unit. Also attached to the system was a Documentation M200 card reader for card entry and an ASR Teletype for system control and limited data entry.

The unique element of the hardware system was the hard copy output device, a Gould 4800 electrostatic printer/plotter. This is a raster unit with a resolution of 80 dots per inch, which provides either fast hard copy plotting or printing capabilities. This was intended to provide graphics drawings as a part of the drafting system, graphs as a part of the simulation system, and printed output.

The operating system used was the standard DEC Disk Operating System. This software provides job management capabilities and I/O support for program communication with peripheral equipment.

It provides file structuring and managing capabilities for both the disk and DECTape units as well.

Programming was to be done by a staff of graduate students, under the supervision of faculty members of the University. The graduate students involved had had little experience in working with operating systems. Since this was true and since the major thrust of the project was not the development of the Supervisor itself, the capabilities of DOS were used without modification or addition, and the Supervisor was written as an executing program under DOS.

The Supervisor itself is written to provide sequencing and overlay support for the various system modules. The relationship of the Supervisor to the system modules is the same, then, as that between an operating system and a compiler or linkage editor. In turn, the system modules operate on user programs and bear the same relationship to them that a compiler does to the program it is compiling.

The system modules are of two basic types. The first type consists of a set of program steps which are performed in a pre-determined order, independent of the user programs on which they are working. An example of this type is the Grafix package. The second type consists of one or more program steps which may themselves require several overlays. These overlays may be called in an arbitrary order, depending on the data and the user program which is executing. An example of this is the DIFSUB step of the Illism package. Further, the sets of steps in the modules are not mutually exclusive; that is,

the same step may be used by two different modules. For instance, the Basic Plotting Package step is executed by the Grafix package to produce the final drawings on the Gould, and is intended to be used by the Illism package to produce graphs on the same device. In addition, most of the system's modules require all available core in order to reduce the amount of overlaying required. Finally, the entire system has to operate in batch mode with respect to the user programs.

These considerations led to a set of requirements which the Supervisor had to satisfy. First, it had to provide a rapid overlay capability to accommodate Illism. Second, it had to allow a large number of overlays for a module--approximately 15 for one module. Third, it had to provide for rapid sequencing between system modules and between steps within a module, and a great deal of flexibility in the order in which the steps were executed. Finally, it had to be as small as possible to allow as much core space as possible to be given over to the system modules.

2. SYSTEM STRUCTURE

With these goals in mind, the following structure emerged. The entire system runs under control of and with the services of the Disk Operating System (DOS). From the point of view of DOS, the system looks like a single program, a portion of which is permanently core resident and the remainder of which resides on storage devices in the form of overlays. The core resident portion is the Supervisor and is responsible for providing the overlaying facilities. DOS, then, views the system as in Figure 2.1.

The Supervisor views the remainder of the system as a set of monitors, each of which may or may not possess a set of overlays. Each of the monitors and overlays is identified to the Supervisor by an ordered pair of numbers. In the case of an overlay, the pair is (monitor number, overlay number). In the case of a monitor, the pair is (0, monitor number), since the Supervisor is monitor 0. The sets of overlays are structurally mutually exclusive, but the structural information for two overlays contained in the Supervisor may, in fact, refer to the same overlay file.

The monitors are responsible for notifying the Supervisor of changes in the status of the overlays, calling the Supervisor to read in overlays, and directing the Supervisor what to do when the monitor is done. Options for a monitor upon completion include running another monitor, turning to the batch input device for the next job, or returning to a previous monitor. The latter option allows a monitor to temporarily

Figure 2.1
System as viewed by DOS

Figure 2.2
System as viewed by the Supervisor

relinquish control to another monitor, then regain control later. It is the responsibility of the monitor to keep track of which overlays are in core and at what points another overlay is required. Since the Supervisor cannot monitor I/O operations, the monitor itself must also notify the Supervisor when the location of an overlay file has changed. A monitor knows the location of its file by the file name; the Supervisor, upon request, looks this file up in the directory of one of the mass storage devices and records the physical location of the file in its tables. The Supervisor can then use this information to read the file in rapidly when requested to do so.

All the facilities of DOS are also available to a monitor. In effect, the Supervisor and DOS combined appear to a monitor as a single operating system. Further, a monitor knows only of its own overlays; it may not have access to another monitor's overlays. Thus, to a monitor the system appears as in Figure 2.3.

Because the Gould was not defined as an I/O device under DOS and since we did not wish to extend DOS itself, a monitor was written to allow the Gould to be used as a line printer. A monitor may request that the line printer monitor be run before the next monitor is executed.

For purposes of batch execution, the monitors are grouped into sequences known as subsystems. The Grafix subsystem, for instance, consists of the Grafix compiler monitor and the Basic Plotting Package monitor. The structure of these subsystems is not contained in the Supervisor or its tables--it is defined by the monitors as each requests

Figure 2.3
System as viewed by a monitor

the next in the subsystem. The Supervisor maintains the identity of the first monitor in each subsystem and the subsystem name by which the user communicates with the system.

This overall structure gives the maximum amount of flexibility while requiring the least amount of information to be stored in the Supervisor. Further, by identifying the Supervisor itself as a monitor with the other monitors as its overlays, the same routines may be used for reading in both monitors and overlays. Likewise, the same table may be used to store the information common to both. By leaving the responsibility for calling for overlays up to the monitor and by requiring the monitor to transfer control to that overlay, the entire responsibility for determining the structure of the system below the level of the monitor is left to the monitors themselves--The Supervisor neither knows nor has to remember this structure. Finally, by requiring the monitors to tell the Supervisor what to do next, by allowing a monitor to be in more than one subsystem, and by allowing a group of monitors to be re-executed an arbitrary number of times, a great deal of flexibility in the structure of the subsystems is gained, and at the same time, the overhead on the Supervisor is reduced.

The system itself consists of five parts: the core-resident Supervisor, the supporting system monitors, the user-supplied monitors and overlays, the table initializer, and the debugging support. The Supervisor consists of the Sequencer, the Overlayer, and the Table

Manager. There are two system monitors--one for using the Gould as a line printer, and one for handling system errors. The Initializer is a bootstrap routine which sets up the system tables and locates the monitors and overlays on the storage devices. Finally, there is a program to build overlay files and a version of the Initializer specifically designed for debugging.

3. DATA STRUCTURES FOR THE SYSTEM

The Supervisor routines are essentially table driven. The tables were set up to allow maximum flexibility in the structure and interrelationships of the monitors and overlays. The tree-like structure of the lines of control (see Figure 2.2) is the only structural constraint imposed; the remainder of the structure is defined by the tables.

The information contained in the tables includes, for all overlays and monitors, the location and length of the overlay on the mass storage devices and the starting address for the overlay in core. In addition, monitors require tables of their transfer addresses, the number of overlays per monitor, and the region of core occupied by each monitor and its overlays. The latter is required in order to initialize the stack pointer before a monitor is read in. This gives as much stack space as possible to each monitor while preventing a monitor from overwriting the stack with an overlay. Subsystem structure is defined by a table giving subsystem names and a corresponding table giving the first monitor in each subsystem.

All tables except the overlay table are simply vectors containing the information. The overlay table is constructed in a hierarchical manner to provide as much protection for the overlays as possible (see Figure 3.1). Each monitor is allocated a subtable containing the information for its overlays. In addition, the Supervisor,

Figure 3.1
Schematic of Overlay Table Structure

in its role as a monitor, is allocated a subtable containing the information for the monitors. The Supervisor remembers the identity of the currently executing monitor and allows it--using the overlay routine--to access only its own overlays. This provides some protection for the table and corresponding overlays.

The size of each subtable is determined at load time when the initializer is run. The subtables are allocated core space at that time and may be initialized then as well. The information in the subtables may also be changed, however, using the table managing routine. This would allow, for example, a monitor to move an overlay from tape to disk, then ask the Supervisor to change its tables to reflect the new situation. The length of the overlay and its starting address in core may also be changed. As is the case with the overlay routine, the table manager allows a monitor to access only the information on its own overlays.

The overlay table functions as follows. The Supervisor uses the number of current monitor to index the Monitor Table. This yields a pointer to the beginning of the monitor's subtable. The overlay number is then used to access the overlay information in the subtable. If the request is to read in an overlay, the location of the overlay, its length, and the core location to which it is to be read are used to read in the overlay. If the request is to change an entry in the table, the appropriate changes are made.

The Supervisor protects itself by not allowing any overlay to overwrite its code or its tables. However, the memory containing

the Supervisor, DOS, and the stack is not hardware-protected, and the Supervisor does not provide software protection for either DOS or the stack. Protection of the Supervisor's code and tables depends entirely on the monitors using only the Supervisor routines to access that area of core. Full software protection would have required changes to DOS, and hardware protection would have required hardware modification step which was not even contemplated.

Initialization of the tables is accomplished by a routine executed as a bootstrap to the system. The routine exists in two versions, differing only in the source of the data to be used in the initialization. In the version to be used in production runs, the data is contained in the routine itself and is presumed to be in the correct form; little checking is done for consistency. The version used for debugging uses data obtained from an input file such as a card reader or a disk file. This data is not presumed to be correct and extensive format and consistency checking is done.

The reasoning behind the use of two versions is as follows: In the case of production runs, the bootstrap should be fast, reliable, and easy to run, so that an operator with a minimum amount of training can execute the system with the greatest ease. Flexibility is not a great consideration in this case since the structure of the system as a whole will not change between production runs.

However, in the case of debugging runs where new portions of monitors are being tested, the situation is exactly reversed. The operator is an experienced programmer, and the speed of the bootstrap

is not as important as its flexibility. Since the structure of the system, particularly the size, location, and starting addresses of monitors and overlays, will almost certainly change between runs, the operator must be able to easily alter the initial data put in the Supervisor tables. This is best accomplished if the data is taken from an input file rather than stored within the Supervisor's load module.

By this table organization, then, the Supervisor gains flexibility. Changing the structure of the system requires changing only the information in the Supervisor tables, not altering the Supervisor code itself. Debugging is made somewhat easier since the tables may be reinitialized from an external file at the beginning of each run. Finally, the Supervisor provides as much protection as possible without making hardware changes or changes to DOS.

4. SYSTEM ROUTINES

The system is divided into five major sets of routines. The first set is a core-resident Supervisor which is supported by the second set of two systems monitors. The third set consists of the initializers; the fourth, the debugging support, the fifth is the remaining monitors and overlays.

The core-resident Supervisor is in turn made up of three routines: the Scheduler, the Overlayer, and the Table Manager. The Scheduler's main responsibility is to sequence from one monitor to another and to insure that the stack is properly relocated. The Scheduler is also responsible for executing the line printer emulator and for maintaining a structure known as the nest.

The nest is a pushdown stack used to provide a primitive looping facility among monitors. A monitor, before relinquishing control, may request that the Scheduler save its identification number so that it may be re-entered at a later point. These numbers are pushed onto the nest and popped off it at the request of a subsequent monitor. The monitor whose number was just popped off the nest is then given control. Thus, by making use of this facility and by passing information in core, a sequence of monitors may be re-executed an arbitrary number of times until some condition is satisfied.

When it is called, the Scheduler identifies the Supervisor as the monitor in control. It determines if the line printer emulator has been requested and executes it if it has. It then

determines what action has been requested by the previous monitor. Options include executing another specific monitor, either with or without stacking the previous monitor's number on the nest, or running the monitor whose number is on top of the nest. If the latter option is requested and the nest is empty, the Scheduler reads the next control card from the card reader and determines the next monitor from the subsystem name. In any case, the stack is reset, and the next monitor is read in and given control.

The Overlayer is responsible for reading all monitors and overlays into core. It is also responsible for providing most of the system protection, such as preventing a monitor from accessing overlays which are not its own and from overwriting the Supervisor. When it is called, the Overlayer first checks the overlay number to see that it is valid and that all the overlay's table information has been initialized. It then sets up the I/O blocks from the table, checks for Supervisor overwrite, and transfers the overlay to core. Finally, it returns control to the calling program.

The third routine in the core-resident Supervisor is the Table Manager. Its primary purpose is to allow monitors to change information in the Supervisor tables to correspond to changes in the status of the monitor's overlays. For example, this allows a monitor to shift the physical location of an overlay or even change the correspondence between overlay numbers and overlay files. The routine can alter any of the information in the overlay tables, but it does

restrict a monitor to changing only its own subtable. Further, this routine serves to insulate the monitor from the precise form of the data in the table while allowing it to change the data if necessary.

In addition to the core-resident Supervisor, there are two monitors used by the Supervisor--a line printer emulator and an error handler. These are identified within the system structure as overlays 0 and 1 to monitor 0, the Supervisor. The line printer emulator is responsible for taking an alphanumeric file from the disk and printing it on the Gould. Output had to be buffered in this fashion because the Gould produces smears on the paper if it is halted for too long. Further, in order to make the Gould available directly as an output device, it would have been necessary to define it as such in DOS, requiring changes to DOS.

The error routine handles errors which arise as a part of the Supervisor operation. These are exclusive of the errors handled by DOS, such as "device not ready" or "illegal opcode", and include such errors as attempting to overwrite the Supervisor or to use illegal overlays. In the current version, all errors are considered fatal. An error message is printed giving the type of error and all pertinent information, and the system is terminated.

The bootstrap routines are responsible for filling in the initial values of all Supervisor tables. There are two versions: one gets the values from tables in the initializer itself, and the other gets its values from an input device. In either case, the

Table Manager is used to locate the overlay files and fill in the initial values of the overlay table. The other tables are filled in directly.

Finally, a separate program is provided to create overlay files. This is not a part of the executing system, but a stand-alone program. A load module is built, using DOS routines, which reflects the core configuration at the time the overlay is in core. This is read into core, and the Overlay Builder is placed where the Supervisor would normally be. The user then specifies the name of the overlay file and the area of core containing the overlay. The program then creates a contiguous file of the required size and writes the overlay into it.

5. AN EXAMPLE OF SYSTEM EXECUTION

A typical run of the System might proceed as follows. The Supervisor and the Initializer are read into core, and the Initializer is executed. This fills in all the Supervisor tables, allocates the overlay table, and uses the Table Manager to locate monitors and overlays and fill their locations into the overlay table. Control is then transferred to the Scheduler.

The Scheduler reads a card to find the name of the first subsystem to be run. The subsystem table is accessed to give the first monitor in the subsystem and that monitor is read into core and given control.

When the monitor wishes to read in its first overlay, it calls the Overlayer, passing as a parameter the number of the overlay to be read in. The Overlayer uses the monitor number to gain access to the monitor's subtable, then the overlay number to get the information on the overlay. The Overlayer fills in the I/O control blocks from the overlay table and calls a DOS routine to do the transfer. It then returns control to the calling monitor.

Suppose that, at some point, the monitor moves one of its overlays from one device to another. The monitor must then call the Table Manager to make the necessary changes to the Supervisor tables. The Table Manager locates the file on the new device and changes the tables accordingly. Of course, specifications other than the physical location of the file could have been changed by this routine, had the

monitor so ordered.

Suppose the monitor then exits, requesting that the line printer be run, followed by a second monitor. The Scheduler is given control and it overlays and executes the line printer. It then overlays the second monitor and gives it control. This monitor may exit, requesting the execution of a third monitor, then its own re-execution. The second monitor's number is pushed onto the nest and the third monitor is read into core and executed. It returns, say, with no further requests. The nest is popped and the second monitor read in and executed again. This monitor then exits, with no further requests, and, since the nest is empty, another card is read from the card reader to determine the next subsystem.

6. CONCLUSION

In summary, a system has been written which satisfies the stated objectives. The system utilizes the fastest possible DOS routines for overlaying so that the overlaying takes place at almost the hardware speed. Little motion is wasted in getting the overlay into core.

The number of overlays per monitor is arbitrary, subject only to overall space in the Supervisor tables. The current restrictions are for 10 monitors with the total number of monitors and overlays set at 24. There is also a restriction on the number of subsystems to 5, but again, this is a restriction imposed by the size of the tables, not by the system code. Since monitor and overlay numbers are contained in a single byte, there would appear to be a restriction to 256 monitors each with 256 overlays; but by extending the monitor and overlay to a fullword and suitable adjusting the table accesses, the restriction can be raised to 65,536 monitors each with 65,536 overlays. If the former is not, for all practical purposes, sufficient, the latter certainly is.

The resident Supervisor currently takes up 600 words. This is the size of the DEC overlaying facility and seems to be a reasonable size. However, only about half of this is taken up by code; the remainder is occupied by the tables, primarily the overlay table. By

adjusting the size of these tables and consequently increasing or decreasing the number of monitors and overlays, the size of the Supervisor itself can be made to vary.

Rapid sequencing has been achieved by considering the Supervisor as a monitor with the actual monitors as its overlays. Thus, all of the speed achieved for the overlays is made available to the Scheduler. Finally, the system makes as few assumptions about the nature and interrelationships of the monitors and overlays as possible, thus giving them as much freedom as possible.

However, all this has not been accomplished without a price. Primarily in order to gain speed, DOS routines were used for the Overlayer which required the overlay files to be contiguous. Further, in order to give the programmers developing monitors and overlays as much control as possible over what went into the overlay file, the overlay builder requires a load module which is essentially a snapshot of core at the time that the overlay is in core. It then provides the programmer word-by-word control over what block of core is put into the file.

The resulting procedure for building overlays is rather lengthy, requiring the use of the DEC linkage editor to build the load module, then the overlay builder to make the actual overlay. Further, the debugging version of the Initializer requires the input file to be rather highly formatted, and requires the programmer to make some statement about all monitors and overlays in the system. The whole

debugging procedure thus becomes rather clumsy. The awkwardness of debugging is more than justified, however, by the fact that the system is not designed to facilitate debugging monitors and overlays, but to execute them efficiently in a production situation.

Bibliography

Brown, R. Leonard, Jr., Numerical Systems on A Minicomputer, Report No. UIUDCS-R-73-555 AEC-COO-1469-0215, Department of Computer Science, University of Illinois, Urbana, February, 1973.

Digital Equipment Corporation, PDP 11-20 Processor Handbook, Maynard, Massachusetts, 1971.

Mueller, David, The ICGS System: System Tables Manual, Report No. UIUCDCS-R-73-609 AEC COO-2383-003, Department of Computer Science, University of Illinois, Urbana, January, 1974.

Mueller, David, The ICGS System: Users Manual, Report No. UICUDCS-R-73-608 AEC COO-2383-002, Department of Computer Science, University of Illinois, Urbana, January, 1974.

Appendix A

The Supervisor

MAIN. MACRO V004A PAGE 1

```

1
2      ; PROGRAM NAME          SYSTEM NUCLEUS
3
4      ; AUTHOR/DATE          DAVID MUELLER  MAY, 1972
5
6      ; MACHINE/LANGUAGE     PDP11, 16K      PAL-11R, V005A
7
8      ; PURPOSE              TO SUPERVISE THE RUNNING OF ICGS
9                          PROGRAMS
10
11
12      ; DECLARATIONS
13
14      ; TITLE SYSPUB
15                          ; GLOBAL STATEMENT FOR PUBLIC USE
16      ; GLOBL CRLINK, DFLINK, KBLINK, LP, NEXT, PARAM
17      ; GLOBL SVC, OVLAY, UPDATE, LPFILE, CRHEAD
18
19      ; TITLE SYSINT
20                          ; GLOBAL STATEMENT FOR INIT
21      ; GLOBL NSYS, SYSTAB, SYSNAM, NTABLE, MODTBL, CORTOP, START
22      ; GLOBL SYSMOD, SVC, UPDATE, DFLINK, MODEND, REGION
23
24      ; TITLE SYSERR
25                          ; GLOBAL STATEMENT FOR ERROR
26      ; GLOBL KBLINK, PARAM, MODHLD, NFIRST, CORTOP, NEXT
27      ; GLOBL TRA
28      000000      ; CSECT
29      000000      R0=%0
30      000001      R1=%1
31      000002      R2=%2
32      000003      R3=%3
33      000004      R4=%4
34      000005      R5=%5
35      000006      SP=%6
36      000007      PC=%7
37
38      ; ENTRY POINTS
39
40 000000 000167 SVC:      JMP  SVC1          ; SCHEDULER
41                                000742
42 000004 000167 OVLAY:   JMP  OVRLY1        ; OVERLAY ROUTINE
43                                001310
44 000010 000167 UPDATE:  JMP  UPDT1         ; UPDATE ROUTINE
45                                001672
46
47      ; DATA ALLOCATIONS
48
49      ; TABLES
50
51      ; TABLES USED BY SYSTEM ALONE
52      ; SUBSYSTEM NAMES - 1/SUBSYSTEM
53 00014 000000 SYSNAM:  .WORD 0,0,0,0,0
54 00016 000000
55 00020 000000
56 00022 000000
57 00024 000000

```

MAIN. MACRO V004A PAGE 1+

```

51 00026      000 NSYS:      BYTE 0          ; NUMBER OF SUBSYSTEMS
52                                     ; SUBSYSTEM TABLE
53 00027      377 SYSTAB:    BYTE -1,-1,-1,-1,-1 ; # OF MONITOR TO BE CALLED
    00030      377
    00031      377
    00032      377
    00033      377

54                                     ; TABLES USED BY SYSTEM AND INIT
55                                     ; # OF OVERLAYS / MONITOR
56 00034      377 NTABLE:    BYTE -1          ; SYSTEM
57 00035      377           BYTE -1          ; DUMMY MONITOR
58 00036      377           BYTE -1,-1,-1,-1,-1 ; SPARES
    00037      377
    00040      377
    00041      377
    00042      377
59 00043      377           BYTE -1,-1,-1,-1,-1 ;
    00044      377
    00045      377
    00046      377
    00047      377

60                                     ; EVEN          ; REGION ADDRESSES
61 00050 000000 REGION:     WORD 0          ; LP
62 00052 000000           WORD 0          ; ERROR
63 00054 000000           WORD 0,0,0,0,0   ; SPARES
    00056 000000
    00060 000000
    00062 000000
    00064 000000
64 00066 000000           WORD 0,0,0,0,0   ;
    00070 000000
    00072 000000
    00074 000000
    00076 000000

65                                     ; TABLES USED BY SYSTEM AND OVERLAY
66                                     ; EVEN          ; STARTING ADDRESSES
67 00100 000000 START:     WORD 0          ; LP
68 00102 000000           WORD 0          ; ERROR
69 00104 000000           WORD 0,0,0,0,0   ; SPARES
    00106 000000
    00110 000000
    00112 000000
    00114 000000
70 00116 000000           WORD 0,0,0,0,0   ;
    00120 000000
    00122 000000
    00124 000000
    00126 000000

71                                     ; TABLES USED BY OVERLAY AND LOOK
72                                     ; EVEN          ; BASES OF MONITOR OVERLAY TABLES
73 00130 000160 MODTAB:    WORD MON0       ; SYSTEM
74 00132 177777           WORD -1          ; DUMMY MONITOR
75 00134 177777           WORD -1,-1,-1,-1,-1 ; SPARES
    00136 177777
    00140 177777
    00142 177777
    00144 177777

```

MAIN. MACRO V004A . PAGE 1+

```

76 00146 177777 . WORD -1, -1, -1, -1, -1 ;
    00150 177777
    00152 177777
    00154 177777
    00156 177777
77 . EVEN ; SYSTEM OVERLAY TABLE
78 . LINE PRINTER
79 00160 000000 MON0: . WORD 0 ; ABSOLUTE BLOCK # ON DT/DF
80 00162 000000 . WORD 0 ; MEMORY ADDRESS
81 00164 000000 . WORD 0 ; # OF WORDS & UNIT SPECS
82 00166 000000 . WORD 0, 0, 0 ; ERROR
    00170 000000
    00172 000000
83 000420 . = +148.
84 00420 000000 MODEND: . WORD 0 ; END OF SYSTEM OVERLAY TABLE
85
86 . VARIABLES
87
88 . EVEN ; VARIABLES USED BY ALL ROUTINES
89 00422 000000 PARAM: . WORD 0, 0, 0, 0, 0 ; PARAMETER FIELD
    00424 000000
    00426 000000
    00430 000000
    00432 000000
90 00434 000000 . WORD 0, 0, 0, 0, 0
    00436 000000
    00440 000000
    00442 000000
    00444 000000
91 00446 000 SYSMOD: . BYTE 0 ; MONITOR NOW IN CONTROL
92 00447 000 MODHLD: . BYTE 0 ; MONITOR PREVIOUSLY IN CONTROL
93 00450 000 LP: . BYTE 0 ; LINE PRINTER INDICATOR=1 TO RUN
    ; =0 IF NOT
94
95 00451 000 NEXT: . BYTE 0 ; NEXT MONITOR TO BE EXECUTED
96 00452 000000 LFFILE: . WORD 0, 0, 0 ; FILE TO BE PRINTED
    00454 000000
    00456 000000
97 . VARIABLES USED BY SYSTEM ALONE
98 00460 044 SYSCHK: . ASCII /##/ ; SYSTEM CONTROL CARD ID
    00461 044
99 00462 000 NEST: . BYTE 0, 0, 0, 0, 0 ; NEST
    00463 000
    00464 000
    00465 000
    00466 000
    00467 000
100 . EVEN ; ADDRESS OF BEGINNING OF NEST
101 0470 000462/NFIRST: . WORD NEST ; (NOTE - THIS VARIABLE ALSO
102 . REFERENCED BY ERROR ROUTINE)
103 0472 000470/NLAST: . WORD NEST+6 ; ADDRESS OF END OF NEST + 1
104 0474 000462/SR1: . WORD NEST ; SAVE AREA FOR NEST POINTER
105 . VARIABLES USED BY LOOK ALONE
106 0476 000000: USET: . WORD 0 ; UNIT NUMBERS TO SET IN NWORDS
107 0500 040000 . WORD 40000
108 0502 100000 . WORD 100000
109 0504 140000 . WORD 140000
110 000000 STATUS=3 ; INDEX TO STATUS BYTE FOR UPDATE

```

```

111
112 0506 000000 CORTOP: WORD 0 ADDRESS OF TOP WORD IN CORE
113
114
115
116 021411 EXIT=021411 EXIT CODE
117 0510 014760 UNIT: RAD50 /DF / UNIT NAMES
118 0512 016040 RAD50 /DT /
119 000510 /DF=UNIT CONSTANT /DF / (RADIX 50)
120 000512 /DT=UNIT+2 CONSTANT /DT / (RADIX 50)
121 0514 000 ZERO: BYTE 0 CONSTANT 0
122 0515 001 ONE: BYTE 1 CONSTANT 1
123
124 I/O BUFFERS
125
126 EVEN BUFFER USED BY SYSTEM ALONE
127 0516 000122 CRHEAD: WORD 82 MAXIMUM BYTE COUNT
128 0520 000 BYTE 0 MODE
129 0521 000 BYTE 0 STATUS
130 0522 000122 WORD 82 ACTUAL BYTE COUNT
131 CARD READER BUFFER
132 0524 000 SYSCRD: BYTE 0,0,0 CONTROL CARD ID
133 0525 000
134 0526 000
135 0527 000 NEXTIP: BYTE 0,0,0,0,0 NEXT INPUT
136 0530 000
137 0531 000
138 0532 000
139 0533 000
140 000646 = +74 REMAINDER OF BUFFER
141
142 I/O CONTROL BLOCKS
143
144 EVEN LINK BLOCKS FOR PUBLIC USE
145 0646 000000 WORD 0 CARD READER LINK BLOCK
146 0650 000000 CRLINK: WORD 0 LINK POINTER
147 0652 012624 RAD50 /CRD/ LOGICAL NAME
148 0654 001 BYTE 1 # OF WORDS TO FOLLOW
149 0655 000 BYTE 0 UNIT NUMBER
150 0656 012620 RAD50 /CR / PHYSICAL DEVICE NAME
151 0660 000000 WORD 0 DISK LINK BLOCK
152 0662 000000 DFLINK: WORD 0 LINK POINTER
153 0664 016003 RAD50 /DSK/ LOGICAL NAME OF DATA SET
154 0666 001 BYTE 1 # OF WORDS TO FOLLOW
155 0667 000 BYTE 0 UNIT #
156 0670 014760 RAD50 /DF / PHYSICAL DEVICE NAME
157 0672 000000 WORD 0 TTY LINK BLOCK
158 0674 000000 KBLINK: WORD 0 LINK POINTER
159 0676 100071 RAD50 /TTY/ LOGICAL NAME OF DATA SET
160 0700 001 BYTE 1 NUMBER OF WORDS TO FOLLOW
161 0701 000 BYTE 0 UNIT #
162 0702 042420 RAD50 /KB / UNIT NAME
163
164 CONTROL BLOCK FOR OVLAY AND LOOK
165 LINK BLOCK
166 0706 000000 LINK: WORD 0 LINK POINTER
167 0710 000000 WORD 0 LOGICAL NAME OF DATA SET
168 0712 001 BYTE 1 NUMBER OF WORDS TO FOLLOW

```

MAIN. MACRO V004A PAGE 1+

```

162 0713      000 UNITNO:  .BYTE 0      ; UNIT #
163 0714 000000 UNAME:  .WORD 0      ; PHYSICAL DEVICE NAME
164                                     ; CONTROL BLOCK FOR OVERLAY ALONE
165                                     ; TRAN BLOCK
166 0716 000000 TRAN:  .WORD 0      ; ABSOLUTE BLOCK NUMBER
167 0720 000000 MEMADD: .WORD 0      ; MEMORY START ADDRESS
168 0722 000000 NWORDS: .WORD 0      ; # OF WORDS
169 0724 000004                                     .WORD 4      ; FUNCTION STATUS
170 0726 000000                                     .WORD 0      ; # OF WORDS NOT TRANSFERRED
171      000716'BLKNO=TRAN
172                                     ; CONTROL BLOCK FOR LOOK ALONE
173 0730 000000                                     .WORD 0,0    ; FILE BLOCK
      0732 000000
174 0734 000000 FILE:  .WORD 0,0      ; FILE NAME
      0736 000000
175 0740 000000 EXT:  .WORD 0      ; EXTENSION
176 0742 000000                                     .WORD 0      ; UIC
177 0744 000000                                     .WORD 0      ; PROTECT CODE
178
179      NAME          SCHEDULER
180
181      AUTHOR/DATE    DAVID MUELLER      JULY, 1972
182
183      MACHINE/LANGUAGE PDP-11, 16K      PAL-11R, V005A
184
185      PURPOSE        TO SUPERVISE THE SEQUENCING OF MONITORS
186
187      CALL SEQUENCE   JMP SVC
188
189      GLOBAL PARAMETERS NEXT NEXT MONITOR TO BE RUN
190      LP LINE PRINTER INDICATOR = 0 IF NO RUN
191      = 1 IF RUN
192      LPFILE LINE PRINTER FILE NAME
193      PARAM PARAMETER FIELD FOR COMMUNICATION
194      BETWEEN MONITORS
195
196      ENTRY POINT FOR THE INITIALIZER - RETURN POINT FOR A MODULE
197
198 0746 116767 SVC1:  MOVB SYSMOD,MODHLD  ;SAVE NUMBER OF MODULE RELEASING
      177474
      177473
199 0754 105067      CLRB SYSMOD          ;CONTROL, IDENTIFY SYSTEM IN
      177466
200 0760 016706      MOV REGION,SP        ;CONTROL
      177064
201 0764 016701      MOV SR1,R1           ;RESTORE NEST POINTER
      177504
202
203      RUN LINE PRINTER IF NECESSARY
204
205 0770 105767      TSTB LP              ;CHECK FOR LINE PRINTER REQUEST
      177454
206 0774 001410      BEQ SYS010           ;BRANCH IF NOT
207 0776 004567      JSR R5,OVRLAY        ;ELSE OVERLAY LINE PRINTER
      177002
208 1002 000401      BR .+4
209 1004 000514      .WORD ZERO

```

MAIN. MACRO V004A PAGE 1+

```

210 1006 004777      JSR  PC,@START      ; RUN LINE PRINTER
      177066
211 1012 105067      CLRB LP
      177432
212
213      ; DECIDE WHAT MODULE IS TO BE RUN NEXT AND OVERLAY IT
214
215 1016 105767 SYS010: TSTB NEXT      ; CHECK NEXT
      177427
216 1022 001452      BEQ  SYS050      ; BRANCH IF NEXT IS TO BE ACCESSED
217 1024 116700      MOVB NEXT,R0      ; SET UP TO TRANSFER CONTROL
      177421
218 1030 002424      BLT  SYS030      ; BRANCH IF NEXT IS TO BE ADDED TO
219 1032 116700 SYS020: MOVB NEXT,R0      ; SET UP TO INDEX TABLES
      177413
220 1036 042700      BIC  #177400,R0
      177400
221 1042 006300      ASL  R0
222 1044 016006      MOV  REGION(R0),SP
      000050
223 1050 004567      JSR  R5,OVRLAY      ; OVERLAY NEXT MODULE
      176730
224 1054 000401      BR   +4
225 1056 000451      .WORD NEXT
226
227      ; TRANSFER CONTROL TO NEXT MODULE
228
229 1060 010167      MOV  R1,SR1      ; SAVE NEXT POINTER
      177410
230 1064 116767      MOVB NEXT,SYSMOD      ; IDENTIFY NEXT MODULE IN CONTROL
      177361
      177354
231 1072 105067      CLRB NEXT
      177353
232 1076 000170      JMP  @START(R0)      ; TRANSFER CONTROL
      000100
233
234      ; NEST ADDITION REQUESTED
235
236 1102 105467 SYS030: NEGB NEXT      ; NEXT = -NEXT
      177343
237 1106 020167      CMP  R1,NLAST      ; CHECK FOR NEST OVERFLOW
      177360
238 1112 001403      BEQ  SYS040      ; BRANCH TO ERROR ROUTINE IF SO
239 1114 116721      MOVB NEXT,(R1)+      ; ELSE RETURN TO OVERLAY MODULE
      177331
240 1120 000744      BR   SYS020
241 1122 004567 SYS040: JSR  R5,OVRLAY      ; NEST OVERFLOW - GET ERROR
      176656
242 1126 000401      BR   +4      ; ROUTINE
243 1130 000515      .WORD ONE
244 1132 105067      CLRB PARAM      ; SET ERROR CODE
      177264
245 1136 010167      MOV  R1,PARAM+2
      177262
246 1142 004577      JSR  R5,@START+2      ; GO TO ERROR ROUTINE
      176734

```

MAIN. MACRO V004A PAGE 1+

```

247 1146 104060      EMT 60          ) SYSTEM EXIT
248
249          ) GET NEXT MODULE FROM NEST IF NEST IS NOT EMPTY, ELSE GET
250          ) NEXT MODULE FROM READER
251
252 1150 020167 SYS050: CMP  R1,NFIRST      ) CHECK TO SEE IF NEST IS EMPTY
                177314
253 1154 001403      BEQ  SYS060          ) BRANCH IF SO
254 1156 114167      MOVB -(R1),NEXT      ) ELSE TAKE NEXT MODULE FROM NEST
                177267
255 1162 000723      BR   SYS020          ) AND BRANCH TO OVERLAY IT
256 1164 012746 SYS060: MOV  #CRLINK,-(SP) ) NEST EMPTY - INIT CARD READER
                000650
257 1170 104006      EMT 6
258 1172 012746 SYS070: MOV  #CRHEAD,-(SP) ) READ A CARD
                000516
259 1176 012746      MOV  #CRLINK,-(SP)
                000650
260 1202 104004      EMT 4
261 1204 012746      MOV  #CRLINK,-(SP) ) WAIT
                000650
262 1210 104001      EMT 1
263 1212 026767      CMP  SYSCRD,SYSCHK    ) CHECK FOR SYSTEM CONTROL CARD
                177306
                177240
264 1220 001364      BNE  SYS070          ) BRANCH TO READ ANOTHER IF NOT
265 1222 012746      MOV  #NEXTIP,-(SP)    ) ELSE CONVERT MODULE NAME TO RAD50
                000527
266 1226 005046      CLR  -(SP)
267 1230 104042      EMT 42
268 1232 103757      BCS  SYS070          ) BRANCH IF INVALID
269 1234 011603      MOV  (SP),R3          ) STORE SUBSYSTEM NAME
270 1236 012716      MOV  #CRLINK,(SP)     ) RELEASE CARD READER
                000650
271 1242 104007      EMT 7
272 1244 020327      CMP  R3,#EXIT        ) CHECK FOR EXIT
                021411
273 1250 001417      BEQ  SYS100          ) BRANCH IF SO
274 1252 005000      CLR  R0              ) ELSE SET UP TO SEARCH SUBSYSTEM
275 1254 012702      MOV  #SYSNAM,R2      ) TABLE
                000014
276 1260 022203 SYS080: CMP  (R2)+,R3      ) CHECK SUBSYSTEM NAME
277 1262 001405      BEQ  SYS090          ) BRANCH IF EQUAL
278 1264 005200      INC  R0              ) ELSE INCREMENT SUBSYSTEM #
279 1266 120067      CMPB R0,NSYS         ) CHECK TO SEE IF DONE
                176534
280 1272 003772      BLE  SYS080          ) BRANCH IF NOT
281 1274 000733      BR   SYS060          ) IF DONE SKIP CARD
282 1276 116067 SYS090: MOVB SYSTAB(R0),NEXT ) SUBSYSTEM FOUND - SET NEXT
                000027
                177145
283 1304 002727      BLT  SYS060          ) BRANCH IF INVALID
284 1306 000651      BR   SYS020
285 1310 012746 SYS100: MOV  #DFLINK,-(SP) ) EXIT - RELEASE DISK
                000662
286 1314 104007      EMT 7
287 1316 104060      EMT 60          ) AND EXIT

```

MAIN. MACRO V004A PAGE 1+

```

288
289
290      ; PROGRAM NAME      OVRLAY
291
292      ; AUTHOR/DATE      DAVID MUELLER      JULY, 1972
293
294      ; MACHINE/LANGUAGE  PDP-11, 16K      PAL-11R, V005A
295
296      ; PURPOSE          TO READ IN OVERLAYS
297
298      ; CALL SEQUENCE     JSR  R5,OVRLAY
299                          BR   +4
300                          ; WORD OVR
301                          ; WHERE OVR IS THE ADDRESS OF A BYTE
302                          ; CONTAINING THE NUMBER OF THE OVERLAY TO BE
303                          ; READ IN. A MONITOR MAY READ IN ONLY ITS
304                          ; OWN OVERLAYS
305
306
307 1320 010046 OVRLY1: MOV  R0,-(SP)      ; SAVE REGISTERS
308 1322 010146      MOV  R1,-(SP)
309 1324 010246      MOV  R2,-(SP)
310
311      ; GET PARAMETERS AND VALIDATE OVERLAY NUMBER
312
313 1326 116700      MOVB SYSMOD,R0      ; GET NUMBER OF MODULE CURRENTLY
314      177114
315 1332 042700      BIC  #177400,R0    ; IN CONTROL
316      177400
317 1336 016501      MOV  2(R5),R1      ; GET OVERLAY NUMBER ADDRESS
318      0000002
319 1342 126011      CMPB NTABLE(R0),(R1)
320      0000034
321 1346 002507      BLT  OVR050
322
323      ; GET PARAMETERS FOR TRANSFER
324
325 1350 006300      ASL  R0
326 1352 016000      MOV  MODTBL(R0),R0 ; GET BASE OF OVERLAY TABLE
327      000130
328 1356 003503      BLE  OVR050        ; BRANCH IF THERE ARE NO OVERLAYS
329 1360 111101      MOVB (R1),R1      ; GET OVERLAY NUMBER
330 1362 042701      BIC  #177400,R1
331      177400
332 1366 110102      MOVB R1,R2
333 1370 006301      ASL  R1
334 1372 060100      ADD  R1,R0
335 1374 006301      ASL  R1
336 1376 060100      ADD  R1,R0
337 1400 012067      MOV  (R0)+,BLKNO   ; PICK UP ABSOLUTE BLOCK NUMBER
338      177312
339 1404 003501      BLE  OVR060        ; BRANCH IF NOT INITED
340 1406 012067      MOV  (R0)+,MEMADD   ; PICK UP MEMORY ADDRESS
341      177306
342 1412 003476      BLE  OVR060        ; BRANCH IF NOT INITED
343 1414 012001      MOV  (R0)+,R1      ; PICK UP NUMBER OF WORDS
344 1416 032701      BIT  #100000,R1   ; TEST FOR UNIT NUMBER

```


MAIN. MACRO V004A PAGE 1+

```

100000
337 1422 001003      BNE  OVR010      ; BRANCH IF ONE
338 1424 105067      CLRB UNITNO      ; ELSE SET UNIT NUMBER TO 0
177263
339 1430 000403      BR   OVR020      ;
340 1432 112767 OVR010: MOVB #1,UNITNO ; SET UNIT NUMBER TO 1
000001
177253
341 1440 032701 OVR020: BIT   #40000,R1 ; CHECK FOR UNIT NAME
040000
342 1444 001404      BEQ  OVR030      ; BRANCH IF DISK
343 1446 016767      MOV  DT,UNAME     ; ELSE SET TO DECTAPE
177040
177240
344 1454 000403      BR   OVR040      ;
345 1456 016767 OVR030: MOV  DF,UNAME   ; SET TO DISK
177026
177230
346 1464 042701 OVR040: BIC   #140000,R1 ; CLEAR UNIT BITS
140000
347 1470 022701      CMP   #3777,R1    ; CHECK TO SEE IF NOT INITED
003777
348 1474 001445      BEQ  OVR060      ; BRANCH IF SO
349 1476 010167      MOV  R1,NWORDS    ;
177220
350
351      ; CHECK FOR POSSIBLE SYSTEM PROTECTION VIOLATION
352
353 1502 006301      ASL   R1            ; FIND TOP ADDRESS OF OVERLAY
354 1504 066701      ADD  MEMADD,R1    ;
177210
355 1510 162701      SUB   #2,R1       ;
000002
356 1514 020167      CMP   R1,CORTOP   ; TEST AGAINST MAXIMUM TOP
176766
357 1520 101041      BHI   OVR070      ; BRANCH IF GREATER THAN
358 1522 012746      MOV  #LINK,-(SP)  ; ELSE INIT DATA SET
000706
359 1526 104006      ENT   6           ;
360 1530 012746      MOV  #TRAN,-(SP)  ; TRANSFER OVERLAY
000716
361 1534 012746      MOV  #LINK,-(SP)  ;
000706
362 1540 104010      ENT   10          ;
363 1542 012746      MOV  #LINK,-(SP)  ; WAIT
000706
364 1546 104001      ENT   1           ;
365 1550 012746      MOV  #LINK,-(SP)  ; RELEASE DATA SET
000706
366 1554 104007      ENT   7           ;
367 1556 012602 OVR999: MOV  (SP)+,R2   ; RESTORE REGISTERS
368 1560 012601      MOV  (SP)+,R1    ;
369 1562 012600      MOV  (SP)+,R0    ;
370 1564 000205      RTS   R5         ;
371
372      ; INVALID OVERLAY
373

```

MAIN. MACRO V004A PAGE 1+

```

374 1566 112767 OVR050: MOVB #1,PARAM      ; SET ERROR CODE
      000001
      176626
375 1574 111167      MOVB (R1),PARAM+2    ; STORE OVERLAY # REQUESTED
      176624
376 1600 116067      MOVB NTABLE(R0),PARAM+3; STORE NUMBER OF OVERLAYS
      000034
      176617
377 1606 000423      BR    OVR080
378
379      ; UNITIATED OVERLAY
380
381 1610 112767 OVR060: MOVB #2,PARAM      ; SET ERROR CODE
      000002
      176604
382 1616 110267      MOVB R2,PARAM+2      ; STORE OVERLAY #
      176602
383 1622 000415      BR    OVR080
384
385      ; SYSTEM PROTECTION VIOLATION
386
387 1624 112767 OVR070: MOVB #3,PARAM      ; SET ERROR CODE
      000003
      176570
388 1632 110267      MOVB R2,PARAM+2      ; STORE OVERLAY #
      176566
389 1636 016767      MOV  MEMADD,PARAM+4   ; STORE MEMORY ADDRESS
      177056
      176562
390 1644 016767      MOV  NWORDS,PARAM+6   ; STORE NUMBER OF WORDS
      177052
      176556
391 1652 010167      MOV  R1,PARAM+10      ; STORE CALCULATED TOP ADDRESS
      176554
392
393      ; CALL ERROR ROUTINE
394
395 1656 116767 OVR080: MOVB SYSMOD,PARAM+1 ; STORE MONITOR #
      176564
      176537
396 1664 105067      CLR8 SYSMOD          ; PUT SYSTEM IN CONTROL
      176556
397 1670 004567      JSR  R5,OVRLAY        ; GET ERROR ROUTINE
      176110
398 1674 000401      BR    +4
399 1676 000515      ; WORD ONE
400 1700 004577      JSR  R5,@START+2      ; GO TO ERROR ROUTINE
      176176
401 1704 104060      EMT  60              ; TERMINATE SYSTEM
402
403      ; NAME                UPDATE
404
405      ; AUTHOR/DATE          DAVID MUELLER      JULY, 1972
406      ; MACHINE/LANGUAGE    PDP-11, 16K        PAL-11R, V005A
407
408      ; PURPOSE              TO MAKE CHANGES TO THE SYSTEM TABLES
409

```

. MAIN. MACRO V004A PAGE 1+

```

410      CALL SEQUENCE      JSR R5,UPDATE
411      BR CONT
412      BYTE OVERLAY
413      BYTE STATUS
414      WHERE OVERLAY IS THE OVERLAY NUMBER
415      STATUS IS AN INSTRUCTION BYTE AS FOLLOWS
416      BIT 0 = UNIT NUMBER - 0=0
417      1=1
418      BIT 1 = UNIT NAME - 0=DISK
419      1=DECTAPE
420      BIT 2 = BLANK - FOR ROUTINE USE ONLY
421      BIT 3 = 1 IF MEMORY ADDRESS TO BE
422      CHANGED
423      BIT 4 = 1 IF LENGTH TO BE CHANGED
424      BIT 5 = 1 IF ROUTINE TO LOOK ONLY AT
425      UNIT INDICATED
426      BIT 6 = 1 IF ROUTINE IS TO SCAN ALL
427      UNITS
428      BIT = RESULT - = 0 IF FILE FOUND
429      = 1 IF FILE NOT FOUND
430      THIS BYTE IS FOLLOWED BY WORDS CONTAINING
431      THE FOLLOWING PARAMETERS, IN ORDER
432      1) FILE NAME AND EXTENSION, IN RADIX 50.
433      IF BIT 5 OR 6 = 1
434      2) MEMORY ADDRESS, IF BIT 3 = 1
435      3) LENGTH, IF BIT 4 = 1
436      THE PARAMETERS ARE PRESENT IFF THE
437      CORRESPONDING BITS IN THE STATUS BYTE ARE
438      ON. THEY MUST BE IN THE ORDER GIVEN, BUT
439      ANY COMBINATION OF THEM MAY BE PRESENT.
440      THE LAST PARAMETER IS FOLLOWED BY AN
441      INSTRUCTION LABELED "CONT"
442
443      SAVE REGISTERS AND INITILIZE
444
445 1706 010046 UPDT1:  MOV R0,-(SP)
446 1710 010146      MOV R1,-(SP)
447 1712 010246      MOV R2,-(SP)
448 1714 010346      MOV R3,-(SP)
449 1716 010446      MOV R4,-(SP)
450 1720 012704      MOV #4,R4
451      0000004
452      SET R0 TO POINT TO TABLE ENTRY
453
454 1724 116700      MOVBS SYSMOD,R0      GET MONITOR #
455      176516
456 1730 006300      ASL R0      GET MONITOR TABLE
457 1732 016000      MOV MODTBL(R0),R0
458      000130
459 1736 116501      MOVBS 2(R5),R1      GET OVERLAY #
460      000002
461 1742 006301      ASL R1      GET OVERLAY ENTRY
462 1744 0060100     ADD R1,R0
463 1746 006301      ASL R1
464 1750 0060100     ADD R1,R0
465

```

MAIN. MACRO V004A PAGE 1+

```

463      ; GET ADDRESS OF PARAMETER LIST
464
465 1752 010501      MOV  R5,R1
466 1754 062701      ADD  #4,R1
      000004
467
468      ; CHECK FOR EITHER LOOK OR SEARCH AND LOOK IF NECESSARY
469
470 1760 132765      BITB #140,STATUS(R5) ; CHECK STATUS BYTE
      000140
      000003
471 1766 001464      BEQ  UPD020      ; BRANCH IF NEITHER
472 1770 012167      MOV  (R1)+,FILE      ; GET FILENAME
      176740
473 1774 012167      MOV  (R1)+,FILE+2
      176736
474 2000 012167      MOV  (R1)+,EXT      ; GET EXTENSION
      176734
475 2004 116502 UPD010: MOVB STATUS(R5),R2 ; GET UNIT CODE
      000003
476 2010 042702      BIC  #177774,R2
      177774
477 2014 020227      CMP  R2,#1      ; CHECK FOR DF1
      000001
478 2020 001503      BEQ  UPD050      ; BRANCH IF SO
479 2022 010203      MOV  R2,R3
480 2024 042702      BIC  #177775,R2      ; GET UNIT NAME
      177775
481 2030 016267      MOV  UNIT(R2),UNAME
      000510
      176656
482 2036 116567      MOVB STATUS(R5),UNITNO ; GET UNIT NUMBER
      000003
      176647
483 2044 142767      BICB #376,UNITNO
      000376
      176641
484 2052 012746      MOV  #LINK,-(SP)      ; INIT
      000706
485 2056 104006      EMT  6
486 2060 012746      MOV  #FILE,-(SP)      ; LOOK
      000734
487 2064 005046      CLR  -(SP)
488 2066 012746      MOV  #LINK,-(SP)
      000706
489 2072 104014      EMT  14
490 2074 012746      MOV  #LINK,-(SP)      ; RELEASE
      000706
491 2100 104007      EMT  7
492 2102 042766      BIC  #177477,4(SP) ; CHECK TO SEE IF FILE IS PRESENT
      177477
      000004
493 2110 022766      CMP  #300,4(SP)      ; AND CONTIGUOUS
      000300
      000004
494 2116 001042      BNE  UPD040      ; BRANCH IF NOT
495 2120 012610      MOV  (SP)+,(R0)      ; ELSE SET BLOCK NUMBER

```

MAIN. MACRO V004A PAGE 1+

```

496 2122 042760      BIC  #140000, 4(R0)      ; SET UNIT CODES
      140000
      000004
497 2130 056360      BIS  USET(R3), 4(R0)      ;
      000476
      000004
498 2136 022626      CMP  (SP)+, (SP)+      ; CLEAR STACK
499
500      ; CHECK FOR MEMORY ADDRESS AND LENGTH
501
502 2140 142765 URD020: BICB #200, STATUS(R5) ; SET RESULT
      000200
      000003
503 2146 132765      BITB #10, STATUS(R5) ; CHECK FOR PRESENCE OF MEM ADD
      000010
      000003
504 2154 001402      BEQ  UPD030      ; BRANCH IF NOT
505 2156 012160      MOV  (R1)+, 2(R0)      ; ELSE SET MEMADD
      000002
506 2162 132765 UPD030: BITB #20, STATUS(R5) ; CHECK FOR PRESENCE OF LENGTH
      000020
      000003
507 2170 001407      BEQ  UPD999      ; BRANCH IF NOT
508 2172 042711      BIC  #140000, (R1)      ; ELSE SET LENGTH
      140000
509 2176 042760      BIC  #37777, 4(R0)      ;
      037777
      000004
510 2204 052160      BIS  (R1)+, 4(R0)      ;
      000004
511
512      ; END OF SUBROUTINE
513
514 2210 012604 UPD999: MOV  (SP)+, R4      ; UNSTACK REGISTERS
515 2212 012603      MOV  (SP)+, R3
516 2214 012602      MOV  (SP)+, R2
517 2216 012601      MOV  (SP)+, R1
518 2220 012600      MOV  (SP)+, R0
519 2222 000205      RTS  R5      ; RETURN
520
521      ; LOOK UNSUCCESSFUL
522
523 2224 022626 UPD040: CMP  (SP)+, (SP)+      ; POP STACK
524 2226 005726      TST  (SP)+
525 2230 152765 UPD050: BICB #200, STATUS(R5) ; SET ERROR CODE
      000200
      000003
526 2236 132765      BITB #100, STATUS(R5) ; CHECK TO SEE IF SEARCH
      000100
      000003
527 2244 001761      BEQ  UPD999      ; BRANCH IF NOT
528 2246 005304      DEC  R4      ; ELSE DECREMENT SEARCH COUNTER
529 2250 001757      BEQ  UPD999      ; BRANCH IF DONE
530 2252 105265      INCB STATUS(R5)      ; ELSE CHANGE UNIT NUMBER
      000003
531 2256 000652      BR   UPD010      ; AND BRANCH TO TRY AGAIN
532      000001      END

```

MAIN. MACRO V004R PAGE 1+
SYMBOL TABLE

BLKNO = 000716R	CORTOP 000506R	CRHEAD 000516R
ORLINK 000650R	DF = 000510R	DFLINK 000662R
DT = 000512R	EXIT = 021411	EXT 000740R
FILE 000734R	KBLINK 000674R	LINK 000706R
LP 000450R	LPFILE 000452R	MEMADD 000720R
MODEND 000420R	MODHLD 000447R	MODTBL 000130R
MON0 000160R	NEST 000462R	NEXT 000451R
NEXTIP 000527R	NFIRST 000470R	NLAST 000472R
NSYS 000026R	NTABLE 000034R	NWORDS 000722R
ONE 000515R	OVRLAY 000004R	OVRLY1 001320R
OVR010 001432R	OVR020 001440R	OVR030 001456R
OVR040 001464R	OVR050 001566R	OVR060 001610R
OVR070 001624R	OVR080 001656R	OVR999 001556R
PARAM 000422R	PC =%000007	REGION 000050R
R0 =%000000	R1 =%000001	R2 =%000002
R3 =%000003	R4 =%000004	R5 =%000005
SP =%000006	SR1 000474R	START 000100R
STATUS= 000003	SVC 000000R	SVC1 000746R
SYSCHK 000460R	SYSORD 000524R	SYSMOD 000446R
SYSNAM 000014R	SYSTAB 000027R	SYS010 001016R
SYS020 001032R	SYS030 001102R	SYS040 001122R
SYS050 001150R	SYS060 001164R	SYS070 001172R
SYS080 001260R	SYS090 001276R	SYS100 001310R
TRAN 000716R	TRA. = ***** G	UNAME 000714R
UNIT 000510R	UNITNO 000713R	UPDATE 000010R
UPDT1 001706R	UPD010 002004R	UPD020 002140R
UPD030 002162R	UPD040 002224R	UPD050 002230R
UPD999 002210R	USET 000476R	ZERO 000514R
ABS. 000000 000		
002260 001		

ERRORS DETECTED: 0
FREE CORE. 16174. WORDS
/LP:CDT1:SYSTEM.SRC

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.

C00-2383-0005

2. TITLE

A BATCH SYSTEM WITH RAPID OVERLAY CAPABILITIES

3. TYPE OF DOCUMENT (Check one):

☒ a. Scientific and technical report

☐ b. Conference paper not to be published in a journal:

Title of conference _____

Date of conference _____

Exact location of conference _____

Sponsoring organization _____

☐ c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

☒ a. AEC's normal announcement and distribution procedures may be followed.

☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.

☐ c. Make no announcement or distribution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear

Professor and Principal Investigator

Organization

Department of Computer Science

University of Illinois

Urbana, Illinois 61801

Signature

Charles W. Gear

Date

December 1973

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

☐ a. AEC patent clearance has been granted by responsible AEC patent group.

☐ b. Report has been sent to responsible AEC patent group for clearance.

☐ c. Patent clearance not required.

BIBLIOGRAPHIC DATA SHEET		1. Report No. UIUCDCS-R-73-614	2.	3. Recipient's Accession No.	
4. Title and Subtitle A BATCH SYSTEM WITH RAPID OVERLAY CAPABILITIES				5. Report Date December 1973	
				6.	
7. Author(s) David Henry Mueller				8. Performing Organization Rept. No.	
9. Performing Organization Name and Address Department of Computer Science University of Illinois Urbana, Illinois 61801				10. Project/Task/Work Unit No.	
				11. Contract/Grant No. US AEC AT(11-1)2383	
12. Sponsoring Organization Name and Address US AEC Chicago Operations Office 9800 South Cass Avenue Argonne, Illinois				13. Type of Report & Period Covered Thesis Research	
				14.	
15. Supplementary Notes					
16. Abstracts A batch supervisor for a minicomputer has been written which provides rapid sequencing between jobs. The supervisor can handle a small number of system programs, and also allows each system program to have a large number of overlays. Overlays are read into core rapidly, and the sequencing of overlays and job steps is very flexible. Finally, the supervisor is small, leaving most of the minicomputer's core storage to the system programs.					
17. Key Words and Document Analysis. 17a. Descriptors Supervisory Systems Executive routines overlays minicomputers					
17b. Identifiers/Open-Ended Terms					
COSATI Field/Group					
18. Availability Statement unlimited			19. Security Class (This Report) UNCLASSIFIED		21. No. of Pages 43
			20. Security Class (This Page) UNCLASSIFIED		22. Price