FINAL REPORT

APPLICATIONS OF MICROCOMPUTERS IN BRIDGE DESIGN

by

R. A. Love
Graduate Assistant

F. W. Barton
Faculty Research Scientist

and

W. T. McKeel, Jr.
Research Scientist

(The opinions, findings, and conclusions expressed in this report are those of the authors and not necessarily those of the sponsoring agencies.)

Virginia Highway & Transportation Research Council
(A Cooperative Organization Sponsored Jointly by the Virginia
Department of Highways & Transportation and
the University of Virginia)

In Cooperation with the U. S. Department of Transportation
Federal Highway Administration

Charlottesville, Virginia

March 1986
VHTRC 86-R28

BRIDGE RESEARCH ADVISORY COMMITTEE

L. L. MISENHEIMER, Chairman, District Bridge Engineer, VDH&T

J. E. ANDREWS, Bridge Design Engineer Supervisor, VDH&T

G. W. BOYKIN, District Materials Engineer, VDH&T

C. L. CHAMBERS, Division Bridge Engineer, FHWA

C. D. GARVER, JR., Division Administrator -- Construction Div., VDH&T

M. H. HILTON, Senior Research Scientist, VH&TRC

J. G. G. MCGEE, Assistant Construction Engineer, VDH&T

M. F. MENEFEE, JR., Structural Steel Engineer, VDH&T

R. H. MORECOCK, District Bridge Engineer, VDH&T

C. A. NASH, JR., District Engineer, VDH&T

F. L. PREWOZNIK, District Bridge Engineer, VDH&T

W. L. SELLARS, District Bridge Engineer, VDH&T

F. G. SUTHERLAND, Bridge Engineer, VDH&T

L. R. L. WANG, Prof. of Civil Engineering, Old Dominion University

ABSTRACT

The utilization of microcomputers in bridge design activities in state transportation departments was evaluated through contacts with 32 state agencies. While the present utilization of microcomputers was found to be limited, subsequent research showed the current generation of 16-bit machines to offer significant advantages in complementing existing computing facilities in a manner that fully utilizes the power of both mainframe and microcomputer.

The ability of microcomputers to run large bridge design applications in a stand-alone mode was demonstrated by successfully downloading and converting 4 mainframe programs. Running design and analysis programs in a stand-alone mode frees the mainframe CPU and increases access to software which can be run repetitively without mainframe cost considerations. When access to larger applications on the mainframe are required, the microcomputer used as an intelligent terminal can process input data locally and send it to the mainframe for processing. Output data, in return, can be downloaded to the microcomputer and reviewed off-line or input into microcomputer applications such as spreadsheets or graphics packages for further processing.

APPLICATIONS OF MICROCOMPUTERS IN BRIDGE DESIGN

by

R. A. Love
Graduate Assistant

F. W. Barton
Faculty Research Scientist

and

W. T. McKeel, Jr.
Research Scientist

PROBLEM STATEMENT

Computer applications in engineering design have had a dramatic effect on the analysis and design process in general. Automating analysis and design procedures has relegated much of the computational burden to machines, thus allowing the engineer more time to evaluate alternatives and assume a more creative role in design and decision making. Although the role that computers play may vary from one organization to another, their effect has been revolutionary.

The manner in which design computations are carried out in state departments of transportation is not standardized and varies greatly. Most software developed for design and analysis calculations within bridge divisions has been designed for implementation on large mainframe computers. Bridge designers, in large measure, have access to these programs via terminals, and this has created little demand for other computer configurations such as microcomputers. However, recent developments in microcomputer design have resulted in microcomputers that have stand-alone capabilities rivaling those of minicomputers and mainframes and that also possess versatile communications capability.

Still, there seems to be considerable difference of opinion regarding the role of microcomputers in bridge design. Many bridge divisions having their own large computers and access through terminals find their present configuration satisfactory and see no reason to incur the additional expense of microcomputers. Other bridge engineers, however, are required to use centralized state computer facilities sometimes shared by other state agencies. The inconvenience in gaining access, the high cost of computing and other charges, and excessive turnaround time may not be acceptable. These engineers see the new generation of microcomputers as a cost-effective and preferred alternative for using much of the bridge design software available. The many advantages of microcomputers, such as powerful computing capability, stand-alone capability, communications capability, and cost-effectiveness make them a powerful element in engineering computation.

In Virginia, much of the bridge design activities has been decentralized to offices in eight districts across the state. The present generation of microcomputers would appear to meet most of the computational needs of these offices. These smaller computers could supplement the mainframe, possibly using downloaded, smaller programs in a more efficient mode of operation.

OBJECTIVES

The primary objective of this project was to examine the current and future role of microcomputers in bridge design applications within state departments of transportation. The focus was on the use of microcomputers as a complement to present computing configurations to increase productivity and enhance cost-effectiveness. To achieve this objective, the tasks described below were undertaken.

First, the current manner in which bridge engineers utilize computers for design and analysis was evaluated. This was accomplished by contacting the Federal Highway Administration (FHWA), the American Association of State Highway Officials, and the Highway Engineer Exchange Program to solicit available information. Subsequently, several states were surveyed by phone to determine the computer configurations they presently use for bridge design applications and their current and projected uses of microcomputers.

Second, the capabilities of the present generation of 16-bit microcomputers in bridge design applications were evaluated . This evaluation consisted of compiling data on microcomputer capacities, operating systems, costs, and available languages. Several models of microcomputers currently available were used to run typical bridge design software and their performances were compared.

Third, the feasibility of converting current bridge design software from mainframes to microcomputers was evaluated through conversions of existing software. This conversion looked at how to download programs from a mainframe to a microcomputer, and the effect of downloading on programs in terms of compile/execution time, memory restrictions, and portability of the language.

Finally, after examination and study of the information collected and the tests performed, the potential for increased usage of microcomputers in bridge design activities was evaluated.

MICROCOMPUTER USE IN STATE BRIDGE DIVISIONS

To determine the trends in microcomputer use in the bridge divisions of various state departments of transportation, an informal telephone survey was undertaken. Through such an informal discussion format it was possible to gain a better insight into the subject than would have been possible using a formal written questionnaire. A total of 32 states were contacted (see Table 1). Initially states were contacted based on a prior knowledge of their use of microcomputers and in the process other states involved in microcomputer usage were identified. Additionally, information on states making early progress in the use of microcomputers for bridge-design-related activities was obtained during a visit to the FHWA.

TABLE 1

STATES CONTACTED

Alabama	Massachusetts	Pennsylvania
California	Michigan	South Carolina
Colorado	Minnesota	South Dakota
Connecticut	Mississippi	Tennessee
Delaware	Montana	Texas
Florida	Nebraska	Vermont
Georgia	New Jersey	Virginia
Illinois	New York	Washington
Iowa	North Carolina	West Virginia
Kentucky	Ohio	Wisconsin
Louisiana	Oklahoma	

The survey consisted of contacting a person within a state bridge division or computer division and asking questions from a prepared list (see Table 2). The questions were designed to determine the current mainframe computing environment, to assess the level of satisfaction with this environment, to identify the current utilization of microcomputers in bridge design applications, and to determine the attitudes and perceptions of engineers regarding the usefulness of microcomputers in design. Finally, any plans for future implementation of microcomputers were discussed. A summary of the responses to the survey is presented in Table 3.

TABLE 2

Phone Survey Questions

1. What kind of computer system is used for bridge design and analysis?

2. Do your engineers and designers have computer access through

   a. Direct access via a terminal? or

   b. Submitting data using data entry forms? (Data actually entered and program run by others)

3. Do you use microcomputers in bridge design?

4. If not, do you plan to purchase microcomputers in the near future for use in bridge design activities?

5. Do you use your microcomputers as

   a. A stand-alone unit? or

   b. As an intelligent terminal linked to a larger computer?

6. What kind of bridge design programs are run on your microcomputer?

7. Can a list of these programs be made available?

8. Are your design programs

   a. Written in-house? or

   b. Purchased from vendors?

9. What programming languages are used for programs written in-house?

10. Have you converted any programs currently running on a larger computer to run on your microcomputer?

11. If so, how was the program converted?

    a. Method of downloading used.

    b. Type of compiler or interpreter used.

12. Is increased use of microcomputers planned for the future?

    a. If so, what are your plans? (i.e. upgrade to more powerful machines, micro CAD systems, etc.)

TABLE 3

Summary of Responses to Questions 1-4 of TABLE 2
(MF: mainframe; MN: minicomputer)

State	Principal Computer Used for Bridge Design (not incl. CAD)	Direct Access to Mainframe/ via Terminals (Y/N)	Micro-computer Used for Bridge Design (Y/N)	Plan to Use Micros in Future for Bridge Design Applications (Y/N)
DE	MF	Y	N (1)	N
CA	MF	Y	N	Y (4)
CO	MF	Y	N	N
FL	MF	Y	N	Y (4)
KY	MF	Y	N (2)	Y (4)
IL	MF	Y	N	Y (4)
IO	MF	Y	N	Y (4)
LA	MF	Y	N (3)	N
MN	MF	Y	N	Y (4)
MS	MF	Y	N	Y
NY	MF	Y	Y (1)	Y (5)
OH	MF	Y	Y	
OK	MF	Y	N	N
PA	MF	Y	N (1)	N
TX	MF	Y	N	N
VT	MF	N	Y	
WA	MF	Y	Y	Y (4)
WI	MF	Y	N	Y (6)
GA	MN	Y	N	Y (4)
NE	MF	Y	N	N
TN	MF	Y	N	N
MT	MN	Y	Y	
NJ	MF	Y	Y	
NC	MF	Y	N	N
NE	MF	Y	Y (7)	Y
SC	MF	Y	N	N
AL	MF	Y	N	Y
SD	MF	Y	Y	
CT	MF	Y	N	N
WV	MF	Y	N	N
VA	MF	N	Y	

(1) Microcomputers are used for spreadsheets, word processing, data base management, etc.
(2) Microcomputers are used for planning.
(3) Microcomputers are used in roadway design.
(4) Plans not defined at present.
(5) Microcomputers would be used more for construction management, overload permit and splice design type work; number crunching will still be done on mainframe.
(6) Could possibly get involved with microcomputers if they demonstrate the ability to run large-scale programs in an efficient manner.
(7) Microcomputers currently used for field data collection.

From the information obtained in the survey, several conclusions were drawn. First, it was found that, as would be expected, the large majority of states use mainframe computers in their bridge design and analysis work. Thirty of 32 states contacted, or 94%, use mainframes as their primary computers. The two remaining states utilize minicomputers. However, in almost all instances the bridge divisions using mainframes share them with other state agencies on some type of time-sharing arrangement.

Almost all bridge design groups (94%) have direct access to the computer through terminals located within the group. Additionally, some states with remote design locations, such as Pennsylvania, have terminal access at the district office level. Through terminal access, the engineers are able to run mainframe applications in either an interactive or batch mode, review the results, modify input if desired, and rerun the application. Some states, such as Michigan and Delaware, use screen form packages which simplify data entry at the terminal by creating the actual input form for a given program on the terminal screen. Most states with computer configurations of this type expressed satisfaction with them. In fact, 11 out of the 30 states with terminal access to a mainframe or minicomputer indicated that it served their computing needs completely, and they thus expressed little or no interest in utilizing microcomputers.

However, the majority of the respondents did see some need for improving their computing environment. Reasons cited included slow turnaround time on time-share systems, a desire for better access to software, and insufficient access to terminals connected to the mainframe. Of the 21 states indicating a need for improvement in computer access, 9 states, including Virginia, have begun using microcomputers in some capacity for bridge design. Another 9 states indicated intentions to become involved in using microcomputers in bridge design activities, although, for the most part, no specific plans were reported. (See Table 3.)

The manner in which microcomputers are used for bridge design purposes varies widely from state to state. For example, in Montana, microcomputers are used almost exclusively for bridge design and analysis. Design and analysis programs previously run on IBM minicomputers have been converted from their original FORTRAN coding to BASIC and adapted to an IBM-PC. In South Dakota, and as part of this project in Virginia, FORTRAN bridge design and analysis programs have been downloaded from a mainframe computer and adapted to run on IBM-PC (or compatible) microcomputers using available microcomputer FORTRAN compilers. Ohio uses an IBM-PC 3270 networked to a mainframe and is developing some specialized bridge-design-related applications. New Jersey, taking an approach similar to that of Ohio, has recently purchased several IBM-PC's which will have communications capability with their mainframe via modems. These microcomputers were purchased to satisfy the need of remote design locations for access to the mainframe along with stand-alone computing capability.

Other states are using microcomputers in bridge-related areas but to a lesser extent. New York uses microcomputers for project management functions and for field data collection and review. Future uses may include overload permit and splice design applications. Massachusetts uses an IBM-PC for field data collection and expressed intentions of utilizing it for additional bridge design applications in the future. In Vermont, an IBM-PC AT to be delivered in the near future will be the prime computer used for bridge design applications.

In addition to the states already using or beginning to use microcomputers, 9 other states have indicated a desire to begin using them in the near future. Common among the responses from these states is an uncertainty as to exactly what the capabilities of microcomputers are when used in bridge design and analysis applications. Some engineers expressed doubts as to the ability of these machines to handle large programs; doubts also were expressed about how the integrity of software would be maintained with it being distributed among several users.

Clearly, there is a need to better define the role that microcomputers can play in bridge design at the state level. Several instances have been cited in which private design firms have acquired microcomputers as a complement to their computer configurations to increase productivity and decrease overall computing costs. Such should also be true in bridge design applications.

MICROCOMPUTER HARDWARE CAPABILITIES

The first generation of microcomputers were based on 8-bit central processing units (CPU's) and initially became available in the late seventies. Their use in engineering applications, and for most other applications for that matter, was limited due to speed and memory restrictions. The internal memory was commonly 64 kilobytes (kb), which rendered them unable to run complex programs. Engineering applications written for these machines were essentially small design aids written in BASIC, and the costs for some of the early 8-bit microcomputers were relatively high. The next generation of microcomputers were those with 16-bit CPU's and are the focus of this study.

Capabilities of 16-Bit Microcomputers

The current generation of 16-bit microcomputers generally use one of three types of central processor. They are the Intel 8086 and 8088 CPU's and the Motorola 68000.(1) The 8086 is a true 16-bit processor in that it moves data through a 16-bit data bus and processes 16 bits at a time. The 8088 moves data through an 8-bit bus and processes 16 bits at a time. Thus the 8088 is a less powerful processor than the 8086. The Motorola 68000 CPU is the most powerful of the three. It handles data through a 16-bit data bus but processes 32 bits at a time. Even though the 68000 is significantly more powerful than the 8086 and 8088, it is the least commonly used, because the 86/88 processors were around first and there is more software written supporting them. (1)

The single most important advantage the 16-bit microcomputers have over the earlier 8-bit machines is their internal memory capacity. The internal memory is classified into two types: read-only-memory (ROM) and random access memory (RAM). The ROM is factory installed and is read when the computer is turned on and also when various functions it contains are required by the operating system. It is permanent and cannot be altered by the computer operator. The ROM usually varies between machines made by different manufacturers, even though the same central processor is used. This may be a source of software incompatibility between machines.

The (RAM) is a temporary memory and is accessible to the user. It gives computers their real power since it determines the size of applications that can be run on them. The IBM-PC (8088 CPU), for example, has a memory capacity of one megabyte (1024 kb) with the first 256kb reserved for ROM and the remaining 768 kb used for RAM functions. A system configured with this much RAM permits moderately sophisticated computing, and in fact, represents a more powerful computing capability than do many minicomputers. (1) Four different models of 16-bit microcomputers were available for use during this project and are listed in Table 4.

In addition to the internal memory capabilities of the 16-bit microcomputers, a mass storage memory capability gives them access to vast amounts of data outside the CPU of the machine. This memory is considered long-term memory, since unlike RAM, it remains intact when the machine is turned off. Mass storage memory usually refers to floppy diskettes or hard disks, but can also be in the form of bubble storage devices, tape, and disk emulation. Floppy disk storage is by far the most common form of mass storage, although hard disk drives are rapidly gaining ground. Floppy disks commonly come in 3 1/2-, 5 1/4- and 8-inch sizes. The 5 1/4-inch drives are the most common and were employed on all the microcomputers used in this project. The storage capacity on the 5 1/4-inch diskettes can range from 320 kb to over one megabyte. The machines used in this project all had a mass storage capacity of 360 kb using double-sided, double-density disk drives.

Another important consideration of the floppy disks is that they are formatted by a special program that comes with the microcomputer's operating system. Disk formats vary between different operating systems capable of running on the same machine, the MS-DOS and CP/M, for example. Disk formats can also vary between different versions of the same operating system, the MS-DOS Ver. 1.1 and MS-DOS Ver. 2.1, for example. In general, different formats are nearly all mutually incompatible. (1)

The problem of incompatibility has been lessened somewhat by the availability of conversion software, but this software does not totally solve the problem.

Although floppy disks are convenient, they are much slower and have much less capacity than hard disk drives. Instead of the thin, flexible disks of the floppys, hard disks are machined aluminum platters. These platters are covered with a magnetic medium and are usually sealed within a dust free environment, although removable hard disks are available. The heads of the hard disk sweep near the surface of the

TABLE 4

MICROCOMPUTERS USED IN THIS PROJECT

ZENITH Z-151 (Marketed by NBI)

Word Length:	16-bit
Processor:	Intel 8088
Operating System:	MS-DOS
Installed RAM:	384 kb
Mass Storage:	2-360 kb DS/DD disk drives

IBM PERSONAL COMPUTER

Word Length:	16-bit
Processor:	Intel 8088
Operating System:	PC-DOS; CP/M; UCSD P-System
Installed RAM:	596 kb
Mass Storage:	2-360 kb DS/DD disk drives

COMPAQ PORTABLE COMPUTER

Word Length:	16-bit
Processor:	Intel 8088
Operating System:	MS-DOS; CP/M86; UCSD P-System
Installed RAM:	256 kb
Mass Storage:	2-360 kb DS/DD disk drives

AT&T PERSONAL COMPUTER 6300

Word Length:	16-bit
Processor:	Intel 8086
Operating System:	MS-DOS
Installed RAM:	512 kb
Mass Storage:	2-360 kb DS/DD disk drives

disk without touching it. The storage capacity of the hard disks is very high relative to that of floppy disks. This capacity can range from 5 megabytes to 20 megabytes or more. Also, since the disks are generally not removable, the compatibility problems associated with the floppy disks are not encountered. With the price of hard disk drives continuing to decrease, along with superior performance and a trend toward removable disks, they could eventually replace the floppy drives.

Another type of mass storage worth mentioning is commonly known as disk emulation or RAM disk. This method of storage is superior to that of both the floppy disks and hard disks in terms of speed, but lacks the permanence of the two. A RAM disk can be as much as 50 times faster than a hard disk drive , as no mechanical parts are involved. A RAM disk is created by software that, in effect, partitions the unused RAM into what the computer thinks is an additional disk drive. RAM disks are very useful when running a program which requires reading data frequently from a disk. The mechanical reading and writing of data from a floppy or hard disk can slow the execution time of the program, but this will not be the case if a RAM disk is used. The only major disadvantage of RAM disks is that their contents are lost at power-off. Transferring the desired data to a permanent media before turning the power off will alleviate this problem.

A large number of peripherals, including printers, plotters, expansion cards, and storage devices, are available for use with the 16-bit microcomputers. The peripherals usually interface with the microcomputer in either a serial or parallel mode. Connection of a serial device to the microcomputer will require installation of an RS-232 interface card in one of the microcomputer's expansion slots. Most of the numerous serial expansion cards available combine several system enhancements, such as additional RAM chips, on a single card. Similarly, connection of a parallel device will necessitate installation of a parallel expansion card if one is not already present.

For a comprehensive source of available peripherals, publications such as Computerworld's Microcomputer Hardware Buyer's Guide (2) and PC World's Annual Hardware Review (3) are recommended. Additionally, professional magazines such as Civil Engineering and Engineering News Record periodically include announcements of new equipment, which makes them good sources for current and new hardware items.

A complete discussion of microcomputer applications in bridge design, or any other engineering field, should include the so-called supermicrocomputers, which began to appear around 1980 and in recent years have become increasingly common. There is some disagreement as to what criteria qualifies a machine as a supermicro. Several supermicros employ both 16- and 32-bit architecture, but what really sets these machines apart from those already mentioned is performance.

Supermicros compete in performance directly with mid-ranged minicomputers. Some even have enough power to compete with higher level minicomputers. For example, a supermicro based on a Motorola 68000 CPU can address as much as 16 megabytes (mb) of internal memory, as opposed to 1 mb for the 8088 CPU's discussed earlier. They also can possess a storage memory capacity of over 100 mb. Perhaps most important, some of these machines have virtual memory capacity, which allows the execution of programs too large to fit into internal memory by allowing the functioning part of the program to remain in the main memory at all times. Virtual memory automatically loads the active part of a program into the main memory in time to respond to revelant instructions.(4) Another way in which supermicros outperform earlier machines is in their ability to support multiple users and multiple application tasks. Both hardware design and operating system software play a role in achieving such performance.(4) About one hundred companies manufacture supermicros at present. Machines used more specifically for scientific and engineering applications come from Sun Microsystems, Apollo, and Cadmus.(4) These supermicros can also be used in computer-aided-design and computer-aided-engineering functions.

Although hardware and software capabilities weigh heavily in decisions concerning the use of microcomputers in bridge divisions, the bottom line will probably be the costs associated with the equipment. Hardware costs include the CPU, monitor, printer, and other peripherals. Table 5 gives typical ranges for these costs. The estimated hardware costs for supermicro systems range from $5,000 to $100,000 and up. Included in this range are the new microcomputer based computer-aided-design systems.(4) The cost of equipment will vary greatly depending on the manufacturer, the vendor, and the status of the highly competitive and volatile microcomputer market.

TABLE 5

Typical Hardware Costs (Dollars)

1.	CPU unit	2,000 to 5,000
2.	Floppy disks	300 to 1,000
3.	Hard disks	2,000 to 6,000
4.	Printers	350 to 5,000
5.	Communications hardware	100 and up
6.	Memory expansion chips	200 and up
7.	Memory expansion boards	200 and up

Technology is changing so rapidly that there may be a feeling that a system may become obsolete before it can be installed. However, waiting to purchase the latest technology may result in no purchase at all. Research of the available equipment versus current and anticipated needs is highly important.

SOFTWARE CAPABILITIES

General Considerations

In terms of design applications, it is far more important to consider software than hardware capabilities. In this section, software capabilities of the 16-bit generation of microcomputers are reviewed. Fundamental software is the operating system which ties the main processor and memory to the display, keyboard, and disks.

Some of the operating systems available for the 16-bit microcomputers are the MS-DOS, CP/M-86 and the UCSD p-System, which are used for single-tasking operations on stand-alone machines. Other operating systems, used primarily for multi-tasking operations, are the Unix from Bell Labs, MP/M (an advanced version of CP/M), Pick, and Oasis. The leader among these multi-tasking operating systems is the Unix, of which there are several versions for a wide variety of microcomputers.

The MS-DOS Version 2.11 is the operating system on the four microcomputers used in this project and listed in Table 4. The operating system used on the IBM-PC is called the PC-DOS, which is essentially the same as the MS-DOS. The DOS operating systems are a collection of utilities which manage the operations and data within the computer. There are more than 40 commands which control various computer functions, some of which are essential for running the bridge application programs of this project. All applications run under the DOS, which provides the flexibility of input and output of data and file manipulation capabilities. Therefore, the capability of these microcomputers to perform large-scale design and analysis applications lies not only in the hardware and applications software, but also in the operating system. For more detailed information on the DOS refer to reference (5).

Two capabilities of the MS-DOS which served well when running the large FORTRAN bridge design programs encountered in this project were (1) output files could be spooled to the printer while program execution continued, and (2) batch capabilities allowed several program runs without an operator present. Since the execution time of some programs on microcomputers is slow relative to that on larger machines, the batch capability is a distinct benefit.

The ability of the 16-bit microcomputers to handle a wide variety of programming languages is a further indication of their computing power and versatility. Most of these machines come with a Basic interpreter, but there are also several dozen compilers available for a variety of languages. A fairly complete listing of these compilers and languages is given in Table 6.

When selecting a language for a particular application, factors to be considered include ease of use, portability, and selection of the best language for the task at hand. In general, however, the languages that are easiest to use are the least flexible. Portability is probably the most important consideration, since it allows programs to be run on different machines with little or no modification. It was found that FORTRAN was by far the most popular language, and all of the application programs developed were written in FORTRAN. Also, for this study, the Microsoft FORTRAN compiler was the most convenient to use, especially with large programs and on micros with no hard disk. The MS-FORTRAN compiler used in this project conforms to subset FORTRAN as described in ANSI X3.9-1978, but also contains extensions to this standard. These extensions are listed in the MS-FORTRAN User's Guide in the Appendix (6). Minimizing use of these extensions increased portability, which allowed the bridge design programs to be run easily on other microcomputers and the University of Virginia's Cyber mainframe.

Development of Design Software

With the tremendous growth in microcomputer hardware has come a corresponding growth in software and software vendors. In the bridge engineering field, many of the vendors are engineers who have moved into software development and brought several years of engineering applications experience into the market. The number of applications programs for civil engineering and construction alone has become so large that Hunt's Directory has made a business of keeping track of them and is a good source of information on software for potential bridge applications. (7)

Currently, the majority of vendor-supplied programs are analysis packages rather than design applications. Analysis programs require less upkeep since design programs usually include codes which are subject to change. A review of several software sources found that few bridge design applications were available. The design packages that were found included three systems for small bridges, a pier design program, a pile design program, an influence line generation program, and several coordinate geometry programs. However, almost every conceivable type of structural analysis program is available for all makes of microcomputers. These analysis packages range from simple beam analysis to full-feature, integrated finite element packages.

TABLE 6

Major Programming Languages for 16-Bit Microcomputers

PASCAL COMPILERS

1. Turbo Pascal (Borland International)
2. Pascal/MT+ (Digital Research)
3. Micro Concurrent Pascal (Enertec, Inc.)
4. UCSD Pascal Compiler (IBM)
5. IBM PC Pascal Compiler 2.0
6. MS Pascal (Microsoft)
7. Pascal 86/88 (Real Time Computer Science Corporation)
8. UCSD Pascal Compiler (Softech Microsystems)
9. Concurrent Pascal 8086 (Soft Machines, Inc.)
10. SBB Pascal (Software Building Blocks)

BASIC COMPILERS

1. CBASIC Compiler 2.0 (Digital Research)
2. BASIC Compiler (IBM)
3. ATV/BASIC (LanTech Systems, Inc.)
4. BASIC Compiler (Microsoft)
5. Business BASIC
6. BASIC Compiler (Quantum Software Systems)
7. BASIC Compiler (Softech Systems)
8. BASIC (Supersoft)
9. Squish (Versaterm Systems, LTD.)

BASIC INTERPRETERS

1. B1-286 1.4 (Control-C)
2. BASIC Interpreter (Microsoft)

COMBINED BASIC COMPILERS AND INTERPRETERS

1. APC BASIC (American Planning Corporation)
2. MegaBASIC
3. HAI*BAS (Holland Automation USA, Inc.)
4. Professional Basic (Morgan Computing Company, Inc.
5. Better BASIC (Summit Software Technology, Inc.)

MODULA-2 COMPILERS

1. Logitech Modula-2/86 (Logitech, Inc.)
2. Modula-2 for the IBM-PC (Modula Corporation)
3. M2M-PC (Modula Research Institute)
4. Volition Systems Modula-2 (Volition Systems)

APL INTERPRETERS

1. IBM-PC APL (IBM)
2. Sharp APL/PC (I. P. Sharp Associates LTD.)
3. APL* PLUS/PC (STSC, Inc.)
4. WATCOM APL (WATCOM Products, Inc.)

TABLE 6 (continued)

FORTRAN COMPILERS

1. FORTRAN 77 (Digital Research)
2. FORTRAN 77 Compiler (IBM)
3. FORTRAN Compiler 2.0
4. FORTRAN Compiler (Microsoft)
5. 87 FORTRAN/RTOS (MicroWare, Inc.)
6. FORTRAN 86/88 (Real-Time Computer Science Corporation)
7. FORTRAN 77 (Quantum Software Systems, Inc.)
8. FORTRAN 77 (Softech Microsystems)
9. FORTRAN Compiler (Supersoft)
10. Professional FORTRAN (IBM)
11. R/M FORTRAN (Ryan-McFarland)

FORTH COMPILERS AND INTERPRETERS

1. HSFORTH 2.01 (Harvard Softworks)
2. PC/FORTH 3.0 (Laboratory Microsystems, Inc.)
3. PC/FORTH+ 3.0
4. MMSFORTH (Miller Microcomputer Services)
5. MVP-FORTH PAD (Mountain View Press)
6. FORTH-32 (Quest Research)

C COMPILERS

1. C Compiler (C-Systems)
2. C Compiler (C Ware)
3. CC 86 (Control-C Software)
4. C86 (Computer Innovations, Inc.)
5. Small-C:PC (Custom software)
6. Digital Research C3 (Digital Research)
7. Lattice C Compiler (Lifeboat Associates)
8. Aztec C 86 1.06D (Manx Software Systems)
9. MWC-85 (Mark Williams Company)
10. C Compiler (Microsoft)
11. C Compiler (Quantum Software Systems, Inc.)
12. Inatant C (Rational Systems)
13. C 86/88 (Real-Time Computer Science Corporation)
14. C Compiler (Supersoft, Inc.)
15. C Compiler (Telecon Systems)
16. C Compiler (Whitesmith's LTD.)

COBOL COMPILERS

1. COBOL Compiler (Digital Research)
2. MBP COBOL Compiler (MBP Software Systems Technology)
3. Level II COBOL Compiler 2.6 (Micro Focus, Inc.)
4. Personal COBOL
5. COBOL Compiler (Microsoft)
6. RM/COBOL (Ryan-McFarland)

Source: Reference 8

Many states develop software in-house for their mainframe applications. However, since the use of microcomputers in state bridge divisions is on a relatively small scale at present, similar software development for micros is also limited. For states that already develop software for mainframe bridge design applications, the development of microcomputer applications would seem to be a logical extension.

Of the state bridge divisions currently utilizing microcomputers in bridge design, a few, such as Montana, Ohio, and Virginia, develop some software in-house. These programs are written primarily in BASIC, although Montana has converted several bridge design applications from a FORTRAN code running on an IBM 5100 minicomputer to BASIC for use on an IBM-PC. Table 7 is a list of typical bridge design applications developed in this manner. Most of these programs are small and designed to perform rather specialized functions. While a useful first step, they do not fully meet the need of bridge divisions for general application programs to run on micros.

Potentially one of the most attractive schemes for the development of microcomputer software for bridge design applications is the downloading and conversion of existing mainframe programs.

TABLE 7

Bridge Design Applications Written In-House Using BASIC

1. Bridge Centerline Grade (Va.)
2. Steel Beam or Girder Section Properties in Negative Moment Region (Va.)
3. Steel Beam or Girder Section Properties (Va.)
4. Critical Moments and Shears (Va.)
5. Concrete Section Analysis (Va.)
6. Live Load Reactions on Pier or Abutment (Va.)
7. Bolted Beam/Girder Splice Design and Analysis (Va.)
8. Concentric Curve Skewed Bridge Geometry (Va.)
9. Bearing Stiffener Design or Analysis (Va.)
10. Transverse Stiffener Design or Analysis (Va.)
11. Straight Roadway Skewed Bridge Geometry and Elevations Along Lines (Va.)
12. Various programs to determine bridge geometry and elevations (Mont.)
13. Various programs to determine bent and girder reactions due to various standard and nonstandard loadings (Mont.)
14. Slab Analysis by WSD or USD (Mont.)
15. Prestressed Beam Analysis (Mont.)
16. Prestressed Bulb T Beam Analysis (Mont.)
17. Welded Plate Girder Analysis (Mont.)
18. Two Column Bent Programs (Mn)
19. Coordinate Geometry Program
20. Beam Splice Design (Ohio)
21. Crane Loading Program (Ohio)
22. Analysis of Composite Rolled Beam (Ohio)

Conversion of Mainframe Programs

There are several advantages in having the ability to run large-scale converted mainframe bridge design software on a microcomputer. First, it allows greater flexibility to the engineer, as applications can be run at any time without the need for access to a mainframe. State bridge divisions may be only one type of several state agencies who share time on a mainframe; thus, depending on demand, computer access may not be possible due to a low priority. Also, microcomputers can insulate bridge designers from the inconveniences of unscheduled mainframe downtimes. The converted programs will also be familiar to the users. Programs that were converted as part of this study utilized the same input and output format as those run on the mainframe. In states where design activities are carried out in remote locations, microcomputers can provide an efficient and relatively inexpensive means of distributing computer power. The high costs of communicating with mainframes over phone lines can be minimized. Converting mainframe bridge design software to microcomputer use will ease demand on the mainframe and thereby allow more processor time for other large agency applications.

As part of this study, several attempts at downloading and converting mainframe programs were made. These conversions provided a means of identifying problems encountered and the level of effort involved. With the assistance of the Bridge Division and the Information Systems Division of the Virginia Department of Highways and Transportation, copies of the bridge design programs listed in Table 8 were obtained.

TABLE 8

Bridge Design Programs Obtained by the Virginia Department of Highways and Transportation

1. Prestressed Concrete I-Beam Design and Analysis Program
2. Steel Girder Design and Analysis Program (Composite)
3. Deck Slab Design Program
4. Critical Moments and Shears on a Simple Span for Moving Loads
5. Bridge Geometry Program
6. Georgia Continuous Beam Program
7. Georgia Pier Program
8. SIMON (a complete design system for steel bridge girders)

Of the programs listed in Table 8, successful conversions were made of the first four, but a number of problems were encountered when attempting to convert the remainder. First, many of the programs currently run on mainframes have been around for a long time and are written in early versions of FORTRAN. Some programs, such as the bridge geometry program, were originally written in assembly language and then converted to FORTRAN. Still others were written such that they required machine dependent software. All of these problems require changes in coding and the effort can become quite extensive. In fact, for some of the programs which are not converted, major parts would have to be rewritten entirely. Another obstacle to program conversion can be the programming technique used by the original programmer. An example of this occurred in both the Georgia continuous beam and Georgia pier programs, which are fairly long programs containing few subroutines. This can cause problems because large programs usually must be broken up into groups of subroutines in order to be compiled on the microcomputer; programs without subroutines may require major alterations to existing codes in order to be successfully compiled. Similarly, some programs are simply too large to be converted for use on the current generation of 16-bit microcomputers.

The programs converted in this project included those for the design and analysis of a prestressed concrete I-beam and a steel girder; the design of a deck slab; and the analysis of critical moments and shears. These programs are currently used by the Bridge Division of the Virginia Department of Highways and Transportation on an IBM 3084 mainframe computer. All four are written in FORTRAN and were converted for use on microcomputers as part of this project. Details of the procedure used are included in the Appendix. In this section, two of the large programs (prestressed beam and steel girder) will be used as examples, of the type of bridge design applications that can be run on 16-bit microcomputers. It should be noted that in the following examples the executable run files for each program reside on the same disk drive as the input and output files.

Details of two test problems run on each program but on different microcomputers are given in Table 9. Table 9 also gives the program source file size and executable run file size for the programs. These test runs were made on each of the microcomputers mentioned earlier and on an additional machine equipped with an 8087 math coprocessor chip. The prestressed beam program is fairly long, with approximately 3000 FORTRAN statements in the source file and an executable run file size of 161,480 kb. This size program would certainly not run on the earlier 8-bit machines. Theoretically, an IBM-PC with a full RAM capacity of 640 kb could run an application program of comparable size. Potential limitations related to mass storage capability for programs of this size will be discussed later. Table 9 illustrates that not only does a program of significant size run on the 16-bit microcomputers, but also it executes in a reasonably short time.

TABLE 9

Bridge Design Program Characteristics Illustrating Memory Capacity and Execution Time

PRESTRESSED CONCRETE I-BEAM DESIGN AND ANALYSIS PROGRAM
FORTRAN STATEMENTS IN SOURCE FILE: 2961
EXECUTABLE FILE SIZE: 161480 bytes

	EXECUTION TIME (SECONDS)				
TEST PROBLEM	ZENITH Z-151	IBM-PC	COMPAQ PORTABLE	AT&T PC	COMPAQ W/8087
PB1	54	43	43	33	36
PB2	57	46	47	35	38

TEST PROBLEM DESCRIPTIONS:

PB1: Design of a standard AASHTO type 5 beam with standard HS-20 loading.

PB2: Design of a non AASHTO beam for HS-20 loading and additional concentrated dead loads.

STEEL GIRDER DESIGN AND ANALYSIS PROGRAM
FORTRAN STATEMENTS IN SOURCE FILE: 896
EXECUTABLE FILE SIZE: 90360 bytes
EXECUTION TIME (SECONDS)

TEST PROBLEM	ZENITH Z-151	IBM-PC	COMPAQ PORTABLE	AT&T PC	COMPAQ W/8087
SG1	30	24	24	20	20
SG2	88	74	75	50	36

TEST PROGRAM DESCRIPTIONS:

SG1: Complete analysis of an interior bridge girder of composite construction.

SG2: Three separate complete designs of an interior composite bridge girder at web depths of 48 in, 51 in, and 54 in.

One factor that will affect execution time is the type of CPU employed by the microcomputer. The IBM-PC, the Compaq Portable, and the Zenith Z-151 all use the Intel 8088 CPU. A comparison of the test results for the program run on the IBM and Compaq machines showed virtually identical execution times. However, the run on the Zenith Z-151 was about 20% slower than those on the IBM and Compaq. The probable cause of this is differences in the basic input/output system (BIOS) and other system architecture of the machines. The BIOS is a collection of programs that control the handling of characters between the microprocessor and other devices of the computer system, (8) and is contained in a ROM chip on the microcomputer's system board.

The execution times for the test problems using the AT&T PC with the 8086 CPU turned out to be faster than those for the 8088 machines. This is not surprising since, as was mentioned earlier, the 8086 moves data to and from the CPU through a 16-bit data bus while the 8088 machines use an 8-bit bus.

Another hardware feature which may have a dramatic effect on program execution time is the 8087 math coprocessor. For the test problems in Table 9 there was an increase in execution time of up to 60% using an 8087. The extent to which the 8087 math coprocessor will increase execution time depends largely on the math processing requirements of the program at hand. Programs which perform several iterations will benefit most from an 8087. All of the bridge design software of this project, and most of that available commercially, will be able to take advantage of an 8087. There are disadvantages, however, to using the 8087. It draws a significant percentage of the power supplied to the system board of the microcomputer; in an IBM-PC it can use as much as one-eighth of the system board power. The 8087 also generates a significant amount of heat, so if the microcomputer has several expansion cards installed, problems may arise. Excessive power consumption and heat generation can cause erratic operation of the disk drives, memory malfunctions, periodic lockup of the computer, unsafe heat buildup inside the computer cabinet and, possibly burnout of the power supply. It has been found that most combinations of the expansion cards with an 8087 will allow safe operation of the microcomputer, but due to the possible detrimental effects, each individual microcomputer system should be properly evaluated before adding the 8087 coprocessor. (8)

Earlier, it was noted that a RAM disk may offer increased efficiencies for running certain programs. To illustrate the performance of a RAM disk in this study, the same test problems from Table 9 were run using a RAM disk. The results of the new runs are tabulated in Table 10. The amount of storage in the RAM disk drive varied between machines depending on available RAM. Enough storage was allocated for the RAM disks to allow the executable run files, the input files, and the output files to be stored. This permits direct comparison of the results summarized in Tables 9 and 10.

A comparison of the results in Table 10 with those of Table 9 shows that disk emulation significantly decreases execution times in all cases. The decreased execution times exhibited here can be attributed wholly to decreased input/output time and the decreased time required for the programs to be loaded into memory.

It is evident that the present generation of 16-bit microcomputers exhibit considerable computing power in terms of internal and mass storage memory. The information and examples given to this point indicate that the current generation of microcomputers possess sufficient computing power to be seriously considered as an alternative to the mainframe for bridge design applications.

TABLE 10

Comparison of Bridge Design Program Execution Times using a RAM Disk For Input/Output

PRESTRESSED CONCRETE I-BEAM DESIGN AND ANALYSIS PROGRAM

	EXECUTION TIME (SECONDS)				
TEST PROBLEM	ZENITH Z-151	IBM-PC	COMPAQ PORTABLE	AT&T PC	COMPAQ W/8087
PB1	Note 1	24	Note 1	12	Note 1
PB2	Note 1	27	Note 1	14	Note 1

STEEL GIRDER DESIGN AND ANALYSIS PROGRAM

	EXECUTION TIME (SECONDS)				
TEST PROBLEM	ZENITH Z-151	IBM-PC	COMPAQ PORTABLE	AT&T PC	COMPAQ W/8087
SG1	16	12	12	7	9
SG2	83	59	59	28	22

NOTE: Insufficient memory exists to simultaneously create an emulated disk drive and run the program.

All software must be properly maintained to ensure accurate, reliable results. Inevitably, most software, especially new programs, will have some bugs which will have to be worked out. Detection of these errors and their removal from the program as fast as possible are essential. And, as mentioned earlier, the ability to implement code changes and changes in design procedure is essential. Many of the mainframe programs used for bridge design applications are usually shared among states. The state that develops a given program usually assumes responsibility for maintaining the program and implementing major changes. If one of these programs has been converted for microcomputer use, subsequent changes must be transferred to the converted version. This may prove difficult if changes are not well documented and the conversion process requires extensive source code modifications.

Software development and purchases represent a sizable long-term investment. Changes in computing technology and outgrowing present computing facilities may necessitate a future changeover to more powerful and sophisticated microcomputers. This can have a drastic effect on currently used software, if software portability has not been sufficiently considered early on. Software planning must consider the potential for future migration of programs to other computers. One way to maximize portability is to utilize standard features of standard programming languages and minimize the use of proprietary languages. Where individual users continue to write programs, portability can be maximized by imposing guidelines for program development. These guidelines should specify the languages and operating systems that can be used. Also, complete program documentation should be required.

Finally, a consideration which has become intrinsically associated with microcomputers is the control over the distribution of software. Microcomputers have ushered in the age of truly distributed computing power. Associated with this distribution is the distribution of software, and some form of control must be implemented to properly manage and maintain the integrity of common software. This can be accomplished by centralizing the distribution of programs within the user division and, where possible, distributing executable modules only. Suggested changes to programs should be directed to a centralized location where changes can be made to the source code and the programs then can be redistributed.

There appears to be considerable interest in the utilization of microcomputers in bridge design. As this interest translates into the use of microcomputers, more and more microcomputer bridge design and analysis software will become available. It has already been noted that considerable programming of small design aids and some conversion of mainframe software are taking place, at least in Virginia and South Dakota. As with mainframe software, these microcomputer programs should be available for sharing among the state bridge divisions. Converted mainframe programs currently in use are shown in Table 11 and can be obtained by contacting the bridge division in the states.

TABLE 11

Currently Available Converted Mainframe Bridge Design Programs

1. Prestressed Concrete I-Beam Design and Analysis for Standard AASHTO and Nonstandard, Simple Span Bridge Girders (Virginia)
2. Steel Bridge Girder Design and Analysis (Virginia)
3. Deck Slab Design (Virginia)
4. Critical Moments and Shears on a Simple Span (Virginia)
5. Georgia Bent Program (South Dakota)
6. Continuous Span Prestressed Concrete Bridge Girder Design (South Dakota)
7. PCA Reinforced Concrete Column Design (South Dakota)

PLANS FOR MICROCOMPUTER IMPLEMENTATION

How and when a state DOT bridge design unit should start using microcomputers depend on several factors. Basically, microcomputer use should be considered anytime present computing capabilities require enhancement, such as additional computing power, distribution of computing power, and addition of communications capabilities.

One of the major obstacles to large-scale microcomputer implementation by bridge design groups is their divergence from traditional computerization norms. Much computing in typical bridge design groups is done through a mainframe controlled by a computer systems group, which, at least initially, may be reluctant to accept changes necessitated by the most efficient microcomputer implementation. The support required for microcomputer implementation includes some level of involvement of a computer systems group. The expertise these groups possess in computer hardware systems and in software development and maintenance will be necessary for proper implementation and support. However, changes in traditional attitudes toward computing will be necessary.

The basic computing configurations for 16-bit microcomputers are either in stand-alone operation or as intelligent terminals linked to mainframes.

In a stand-alone mode the microcomputer operates independently and self-sufficiently and provides a significant computing resource without the disadvantages of a time-shared mainframe system. The advantages of also using a microcomputer as an intelligent terminal are numerous. In fact, the ability to use a microcomputer in this mode is an example of how microcomputers can complement existing computer configurations in an efficient and cost-effective manner; the key is the ability of the microcomputer to communicate with a mainframe. Communication enables the engineer both to complement mainframe operations with the microcomputer capabilities and also to use mainframe resources to expand microcomputer power. A number of communications packages are available that allow engineers to communicate with virtually any mainframe system. In this mode, the microcomputer can be used to run applications that are, at present, too large for microcomputer implementation. Also, off-line preparation of data represents a potential for considerable cost savings.

When both personnel and machine costs are considered, the costs of communicating between terminals and the mainframe have become a relatively large portion of the total computational cost, since the costs of computing are decreasing while those of communication and personnel continue to rise.(9) Applications that use microcomputers to assist in the preparation of data and to speed communications to the mainframe show great potential. However, there are a number of costs inherent in microcomputer implementation that go beyond the initial purchase price. These include costs for service and maintenance training and additional hardware and software.

Microcomputers, while not overly fragile, are subject to breakdowns due to mechanical failure and extremes in environment. This will be particularly important if the machines are used in the field. Costs of this type are difficult to foresee and just as hard to plan against, but can be minimized by properly instructing users in machine operation.

A major cost consideration is that related to training. For example, it may be necessary to form and staff internal user support groups. Other training-associated costs may include the value of the time it takes individual users to learn how to operate the computer, the value of productivity lost while the engineers become computer proficient, the cost of time lost attempting to train persons who never become computer proficient, and even the cost of time lost when skilled users interrupt their own work to assist less skilled users with a problem.(10)

Another unexpected cost may be the cost of acquiring additional hardware and software. After initial installation, additional equipment such as peripheral devices, expansion cards, memory, or software utilities frequently are needed. Many costs of this nature can be expected over the life of the machine, but can be held to a minimum by proper planning for current and future uses of the microcomputer installation.

Particular schemes for microcomputer implementation will vary among states based on present computing configurations and the level of satisfaction with these systems. Future computing needs will also play a major role. In states such as Pennsylvania, where mainframe access is good all the way down to the district level and the level of satisfaction is high, microcomputers may play a minor role at best. However, in states where personnel are hampered in their access to a mainframe or dissatisfied with the service they receive, microcomputers can be a distinct benefit. Their usefulness is bound only by the imagination of the engineer and his ability to modify problem solving techniques and office procedures to effectively harness the computer's power.(9)

SUMMARY AND CONCLUSIONS

In this study, an effort was made to assess the present overall computer configurations used in state DOT bridge design groups, determine the utilization of microcomputers in these groups, illustrate applications of microcomputers in bridge design activities, and, finally, to discuss various plans for the application of microcomputers in bridge design.

To determine present overall computing configurations and microcomputer utilization in bridge design, discussions were held with a knowledgeable person in each of 32 state DOT bridge or computer service groups. These discussions also sought to determine levels of satisfaction with present computing systems. These discussions yielded the following findings.

1. Ninety-four percent of state DOT bridge design groups use mainframes as their primary computing resource.

2. Ninety-four percent of these groups have direct access to the mainframe through terminals located within the group.

3. Eleven of the 30 states with terminal access indicated that their computing needs are completely satisfied in this manner.

4. The 21 remaining states indicated a need for improvement in their computing configuration. The most common reasons given were --

    a. slow turnaround time,

    b. a desire for better hands-on access to software, and

    c. insufficient access to mainframe terminals.

5. Of the 21 states indicating a need for improvement --

    a. nine have begun using microcomputers for bridge design in some capacity, and

    b. nine have indicated intentions to become involved in using microcomputers in bridge design.

The results of the survey showed that, in general, the utilization of microcomputers in bridge design at DOTs is at present very limited.

Based on these results, other conclusions were drawn. They are:

1. There is a need to better define the role that microcomputers can play in bridge design applications at DOTs.

2. There is an uncertainty as to what the capabilities of these microcomputers are for bridge design and analysis applications.

3. There is doubt as to the ability of these machines to handle large programs.

4. There is doubt about how the integrity of software would be maintained when distributed among several users.

Subsequently, the ability of these microcomputers to run large bridge design applications efficiently in a stand-alone mode was demonstrated, and several considerations related to software were discussed. These included sources of software, portability, and maintenance.

The modes of operation such as stand-alone and as an intelligent terminal were viewed in the context of how they can best meet the computing needs of bridge designers. A plan for using the microcomputer as an intelligent terminal evolved which exhibited several beneficial features of both modes. In this plan, many large bridge design and analysis applications can be run in a stand-alone mode, thus freeing the mainframe CPU and allowing greater access to software which can be run repetitively without mainframe cost considerations. When access to larger applications on the mainframe are required, the microcomputer used as an intelligent terminal can process input data locally and send it to the mainframe for processing. Output data, in return, can be downloaded to the microcomputer and reviewed off-line. The output data could then be input into microcomputer applications such as spreadsheets or graphics packages for further processing. This plan shows how microcomputers can complement existing computing facilities in a manner which fully utilizes the power of the mainframe and the microcomputer in an economical way.

The development of microcomputers signals a new era in computer use. The significant computing power they possess, along with being relatively inexpensive when compared to traditional large computers, has assured their success. Their use is constantly being explored in many business and engineering applications. Many state DOT bridge design groups are in a position to make full use of microcomputer capabilities, and some states have already begun to do so. Although many serious organizational and financial considerations must be taken into account, a well-planned computing system with microcomputers complementing existing mainframes or minicomputers can significantly enhance present computing capabilities.

REFERENCES

1. Lu, Cary, "Microcomputers: The Second Wave," High Technology, September-October, 1982, pp. 36-52.

2. "Microcomputer Hardware Buyer's Guide," Computerworld, September 26, 1984, Vol. 18, pp. 43-46.

3. "Annual Hardware Review," PC World, Special Edition, 1984.

4. Davis, Dwight B., "Super Micros Into Mini Markets," High Technology, December 1983, pp. 36-46.

5. MS-DOS VERSION 2 REFERENCE GUIDE, Compaq Computer Corporation and Microsoft Corporation, 1984.

6. Microsoft FORTRAN Compiler for the MS-DOS Operating System, Users Guide and Reference Manual, Microsoft Corporation, 1984.

7. Hunt, Alfred J., ed., Hunt's Directory of Microcomputer Software and Services, Pleasant Hill, California, 1984.

8. DeVoney, Chris, IBM'S Personal Computer, Indianapolis: Que Corporation, 1983.

9. Davies, Gary W., "Microcomputers in Civil Engineering," Transportation Research Record 932.

10. "Cutting Through the Hidden Costs of Computing," Personal Computing, October 1984, pp. 122-129.

11. I.B.M. Personal Editor Uses Manual, I.B.M. Corporation, Boca Raton, Florida, 1982.

12. Huizenga, Charlie, and Barnaby, Chip, "But Is It Really FORTRAN?", PC World, February 1984, pp. 172-179.

13. Buttalla, Martin W., "Microcomputers: Do They Have a Place in Large Engineering Firms," Civil Engineering, June 1982, pp. 80-81.

14. "Prestressed Concrete I-Beam Design and Analysis -- Users Guide 18.00," Virginia Department of Highways and Transportation, rev. September 13, 1983.

15. "Plate Girder Design and Analysis -- Users Guide 22.00," Virginia Department of Highways and Transportation, rev. September 1, 1974.

16. "Deck Slab Design -- Users Guide 10.00," Virginia Department of Highways and Transportation, rev. May 13, 1970.

17. "Critical Moments and Shears for Moving Loads -- Users Guide 11.00," Virginia Department of Highways and Transportation, rev. June 2, 1970.

APPENDIX

MAINFRAME TO MICROCOMPUTER SOFTWARE CONVERSIONS

A.1 General

One of the tasks to be performed as part of this study was the investigation of the feasibility of converting mainframe software to run on a microcomputer. In the course of the project four programs were converted. This Appendix will give an overview of the methods used to accomplish this, including the method of downloading the programs from the mainframe to the microcomputer and those for editing and compiling the programs on the microcomputer. A detailed explanation of each individual conversion follows in sections A.5 through A.8.

A.2 Downloading

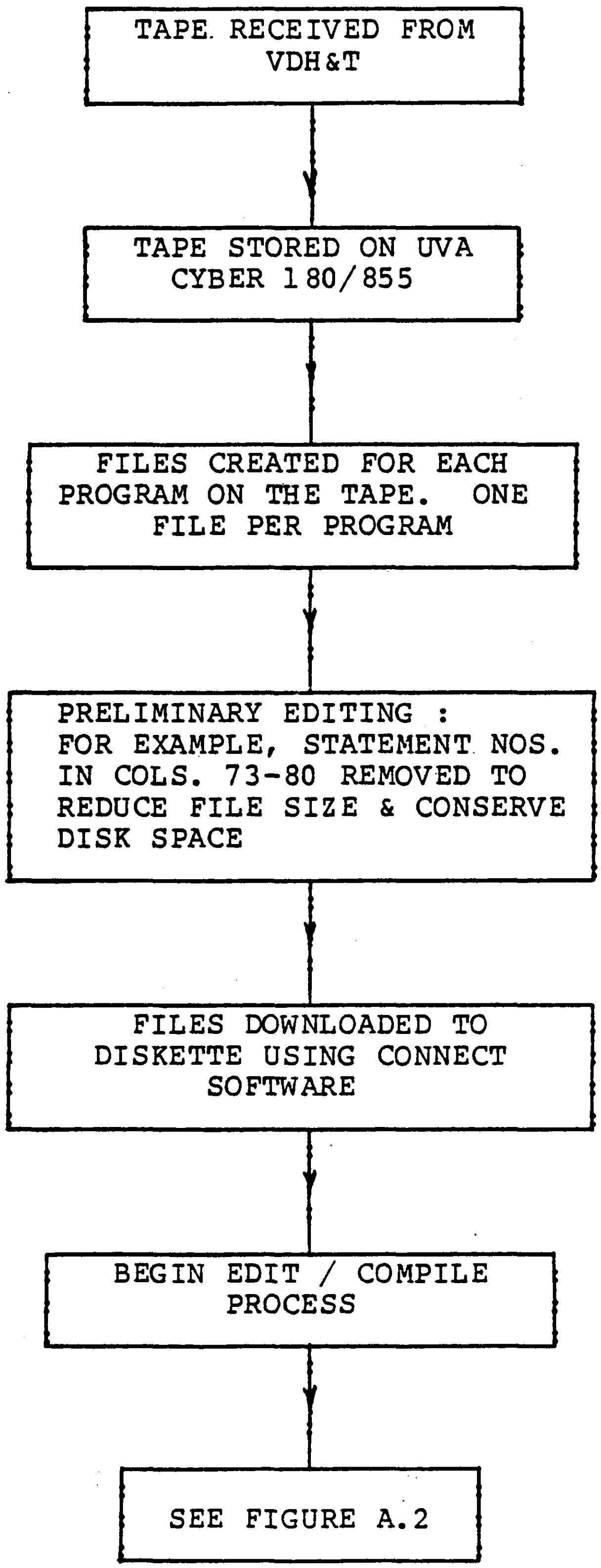
Downloading is the process of transferring data or files from a mainframe computer to a microcomputer. The exact method of doing this will depend on the computer configuration. In general, a micro-mainframe link and file transfer protocol are required. In this project, copies of the four bridge design programs were obtained on tape from the Bridge and Information Systems Divisions of the Virginia Department of Highways and Transportation. This tape, in turn, was stored on the University of Virginia's Cyber 180-855 mainframe computer and the programs transferred to direct access files. After preliminary editing on the Cyber, the program source files were downloaded onto floppy diskettes via microcomputers with communications capability with the Cyber. Two micro-mainframe communications methods were used depending on the microcomputer location used. One method was over a local area network and the other used a modem. See Figure A.1 for a graphic representation of the downloading procedure.

The communications software used to transfer the program source files was obtained from the Academic Computing Center at the University of Virginia. This software, called CONNECT, comes in various versions which can run on both the local area network or over a modem utilizing several makes of microcomputers. CONNECT utilizes the Kermit file transfer protocol developed at Columbia University to transmit files between the microcomputer and mainframe, or vice versa. CONNECT was a key link in the process of downloading and compiling the programs on the microcomputer. It performed the file transfer flawlessly and relatively quickly. The time to transfer a file was primarily determined by the baud rate of the local area network or modem.

A.3 Editing and Compiling

After the program source files were downloaded to the micro, the editing and compiling began. All of the programs required some modification to get them into a microcomputer compatible form. The incompatibilities which had to be corrected before the program could be compiled and executed on the microcomputer fell into two general categories: incompatibilities between the received FORTRAN IV format and the FORTRAN-77 compiler used on the computer, and incompatibilities between input/output methods.

FIGURE A.1
DOWNLOADING PROCEDURE



Editing of the bridge design source listings was done using the IBM Personal Editor, a limited-feature word processor, which is well suited for this type of editing. It was able to handle the largest files encountered with relative ease, could handle several files at once, and had a search and replace capability and a text movement capability within and between files. Refer to reference (11) for a detailed description of the editor.

The edited programs were compiled on the microcomputer using the Microsoft FORTRAN Compiler version 3.2 for the MS-DOS operating system.(6) This compiler conforms to Subset FORTRAN as described in ANSI X3.9-1978. It contains some extensions to the subset language and some features of the full ANSI standard known as FORTRAN-77. Essentially the compiler is a FORTRAN-77 compiler without some of the features of the full FORTRAN-77 standard. No instances were encountered in this project which required compiler capabilities beyond those of MS-FORTRAN.

The set of files that comprise the MS-FORTRAN compiler, along with brief descriptions, are listed in Table A.1.

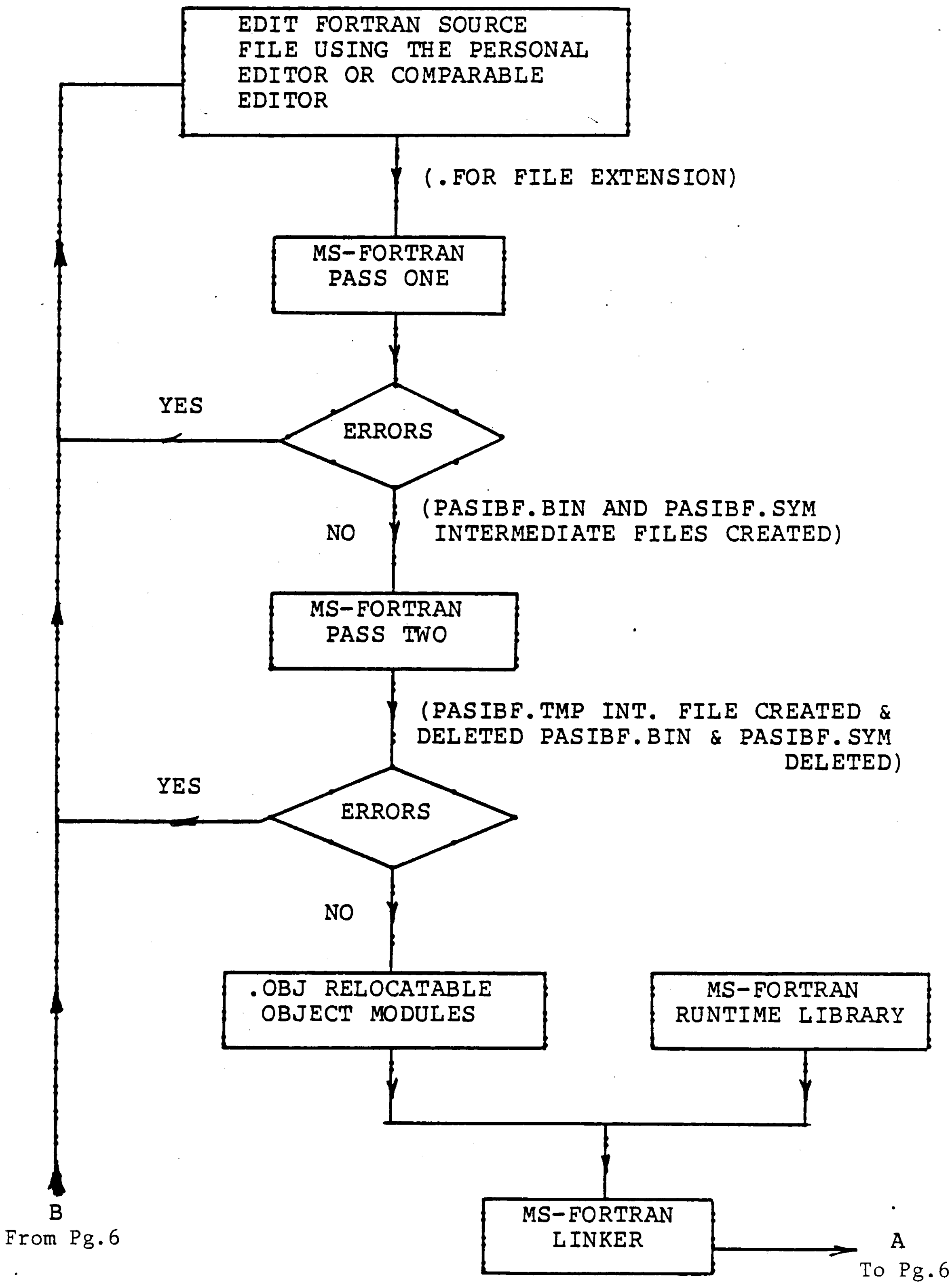
The process of compiling a FORTRAN program to run on the microcomputer using MS-FORTRAN involves compiling the source code into object code modules and linking these modules with any external libraries which the program may require. Compiling requires two passes of the compiler (a third optional pass was not required). The first pass checks the source code for any syntax errors that may be present and creates two intermediate files. The second pass reads the two intermediate files created by pass one and creates the relocatable object files which are written to disk. The relocatable object files, or modules, are called relocatable because they have relative rather than absolute addresses. The final step in creating an executable module is linking. The linker takes all the object modules and links them with the MS-FORTRAN runtime library. The result is an executable module with absolute addresses. A schematic of compiling and linking is presented in Figure A.2. This general procedure is common to most microcomputer FORTRAN compilers, not just MS-FORTRAN. Detailed explanations of the compiling and linking procedure used for the four programs mentioned earlier are given in sections A.5 through A.8. For a detailed explanation of the MS-FORTRAN compiler see reference (6). In the remainder of this appendix pertinent features of the compiler and linker will be explained as needed.

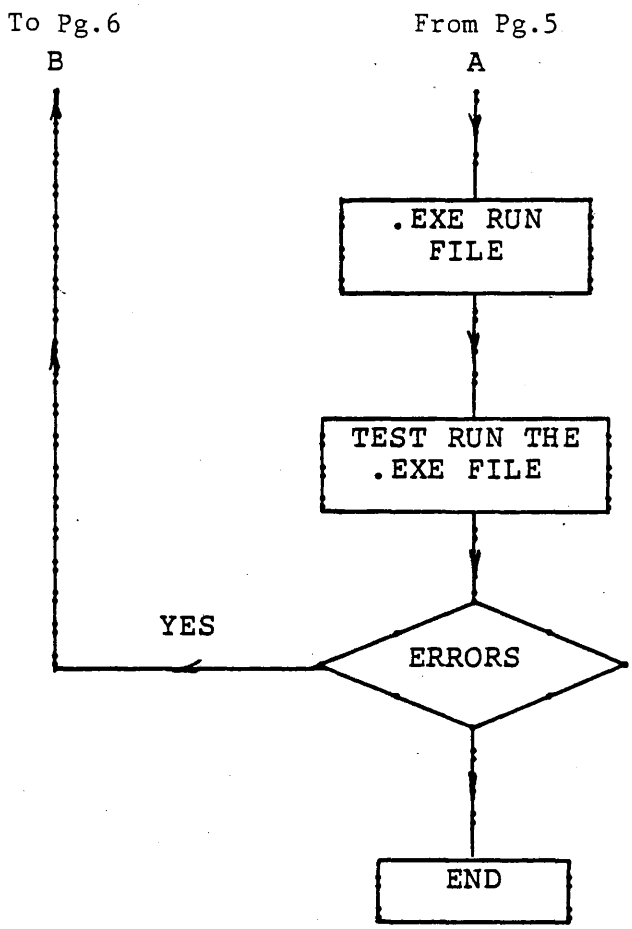
TABLE A.1

MS-FORTRAN SOFTWARE SYSTEM (6)

FILE	DESCRIPTION
FOR1.EXE	Pass one of the compiler
PAS2.EXE	Pass two of the compiler
PAS3.EXE	Pass three of the compiler
FORTRAN.LIB	The default MS-FORTRAN runtime library
FORTRAN.MAP	The link map for FORTRAN.LIB
MATH.LIB	The default floating point package library
MATH.MAP	The link map of MATH.LIB
8087.LIB	An auxiliary library for use with programs that are to run only on machines with the 8087 coprocessor installed and whose size you wish to reduce
8087.MAP	The link map of 8087.LIB
DOS2FOR.LIB	An auxiliary library containing an MS-DOS version 2.0 file system
DOS2FOR.MAP	A map of DOS2FOR.LIB
ALTMATH.LIB	An auxiliary library containing high-speed floating-point supported routines
ALTMATH.MAP	A map of ALTHMATH.LIB
DECMATH.LIB	An auxiliary library containing decimal floating point routines
DECMATH.MAP	The map of DECMATH.LIB
LINK.EXE	The default Microsoft Linker
LINK.V2	Optional version of Microsoft LINK (MS-DOS 2.0)
NULF.OBJ	The dummy file system
NULE6.OBJ	The dummy error system
ENTX6L.ASM	The assembler source of the execution control module that initializes and terminates every program

Figure A.2
THE COMPILE AND LINK PROCESS
USING MS-FORTRAN





To Pg.6

MS-FORTRAN FILE EXTENSIONS

.FOR	FORTRAN source file
.OBJ	Relocatable object file
.LST	Source listing file
.LIB	Library file
.EXE	Executable run file
.MAP	Linker map file
.BIN	Binary file
.TMP	Temporary file

The size of the source code to be compiled is limited by three factors imposed by the design of the microcomputers, not the compiler. First, the executable code must fit onto a single disk. For a double size/double density disk formatted in DOS 2.0 this will amount to 360 kb. The second size limitation is determined by the amount of internal memory available in the machine. This factor determines how large a program can be loaded into the machine. The third limitation on the source code involves the number and size of variables. These microcomputers organize data into 64-kb segments of memory. All local variables, constants and blank common blocks will reside in one of these segments. The total space taken up by all the local variables, constants, and blank common blocks cannot be larger than 64 kb minus the stack and heap. The stack and heap tell the processor where portions of code are located and how large they are, and rarely take up more than 4 kb. This leaves about 60 kb. For example, a single REAL*4 array could contain 15000 elements (say array VAR(15000), then 4x15000=60000 bytes). If there are other variables, constants, or blank common blocks, the array VAR must be smaller. In the event named common blocks are used, they will all reside in their own segment, so they can be as large as 64 kb. (12)

A.4 Incompatabilities

This section documents the incompatibilities encountered in converting the mainframe version of four bridge design programs to run on an IBM-PC or compatible microcomputer. These four programs are (1) Prestressed Concrete I Beam Design and Analysis, (2) Steel Girder Design and Analysis, (3) Deck Slab Design and Analysis, and (4) Critical Moments and Shears on a Simple Span. Upon initial inspection of these programs it was obvious that they were written some time ago in earlier versions of FORTRAN IV. Additionally, the Prestressed Beam Program utilized a data input/output routine which was dependent on the state's mainframe computer. Converting the programs mostly involved getting them in a FORTRAN-77 format and, in the case of the Prestressed Beam Program, changing the input/output method. The following is a detailed list of changes made to the programs to make them compatible not only FORTRAN-77 compatible, but also compatible for microcomputer use. The reference numbers for each item are also used to show the location of each item in the source listings which follow.

TABLE A.2

LIST OF CHANGES MADE TO BRIDGE DESIGN PROGRAMS

REF. NO.	DESCRIPTION
(1)	Program statement added to designate beginning of main program segment. Addition of this statement is optional under MS-FORTRAN.
(2)	The original FORTRAN IV versions of the programs used integer variable names to designate character data. All character variables have now been declared type CHARACTER per FORTRAN-77 standard.
(3)	All character type variables which were previously scattered throughout other common blocks are now all listed in common block ELL. This satisfies the FORTRAN-77 requirement that mixed character/non character variable types are not allowed in the same common block. Common block ELL now has only character variables. (Note: this item refers only to the Prestressed Beam Program.)
(4)	Integer variable ICARD has been removed. It has been replaced by character variable CARD. (Prestressed Beam Program only.)
(5)	Screen header and prompts have been added to make the programs more user friendly.
(6)	In the Prestressed Beam Program, subroutine INITIAL has been added as a replacement for the BLOCK DATA subroutine of the original version. The MS-FORTRAN compiler would not process the BLOCK DATA subroutine properly. Subroutine INITIAL accomplishes the same purpose by initializing variables using assignment statements rather than DATA statements.
(7)	The OPEN statement assigns I/O unit 5 to the external input file. The input file name is supplied by the user at runtime. The OPEN statement is optional in MS-FORTRAN.
(8)	The format for reading the header card from the external input file (character variable WORDS) has been changed in order to be compatible with the CHARACTER type declaration of WORDS. (Prestressed Beam Program only.)
(9)	REREAD is an assembly language routine on the VDH&T IBM 3084 mainframe that is used by the Prestressed Beam Program during data input operations. CALL REREAD has been removed from the microcomputer version and replaced by a functionally similar input method. This method is outlined in reference (13).

TABLE A.2 continued

(10) In lieu of using the REREAD routine, the method of inputing data from the external input file has been changed as follows:

1. All data read after the header card (WORDS) is read as an integer (NTYPE) and as a character variable (CARD), one line at a time. The format used is I1,A79, which causes all 80 columns of each input file line to be read. The character variable CARD can now be thought of as an internal file consisting of 79 alphanumeric characters, including blanks. This internal file can now be read again within the program in the desired format (i.e. F, E, A, I, etc.). For a detailed description see the MS-FORTRAN manual. (6)

2. Depending on the value of NTYPE, control is transferred to other READ statements. These READ statements cause the data in the internal file to be read in the format required for processing by the program.

3. In essence, all data read from the external input file unit 5 is read as an internal file (CARD) which is later reread in the proper numeric or character format required by the program. This method allows data to be input into the program in a similar manner as was done using the REREAD routine on the mainframe. Thus, changes to the program were kept at a minimum.

(11) Changed the statements BTYPE=ABLANK to BTYPE=BLANKA, BLANKB ... etc. The program would not run without these changes being made due to improper initialization of BTYPE.

(12) An apparent error in the original source code was fixed here.

(13) Array VMMS had to be initialized to zero in order for the program to run properly.

(14) A change or update was made in the program here per directions from the VDH&T Bridge Division.

(15) Removed call to IBM mainframe data routine.

(16) Removed variABLE IDATE from Write statement. See (15).

(17) Revised input format.

(18) Removed arrays IDESC(12) and IPROJ(4).

TABLE A.2 continued

(19) Original program used 10 digit integers to represent character data types Statement was: IF(NUM(8)-1077952576)604,605,604

(20) Call to subroutine FSTB removed in a revision made around 1975 per Information Systems Group.

(21) Declared new variable ZM type CHARACTER. ZM was NM in original program:

```
ZM='RL' was NM=-640466880
ZM='RR' was NM=-640073664
```

(22) CALL EXIT replaced by GO TO 500 and 500 STOP.

(23) Four lines were removed. They were:

```
IF(BOJNO-S)11,13,11
11 WRITE(6,12)
12 FORMAT(28HJOB NUMBER DOES NOT COMPARE)
CALL EXIT
```

(24) Removed line IF(BOJNO-S)11,19,11

(25) Removed variable S from READ statement.

A.5 Prestressed Concrete Design and Analysis Program

This section will describe in detail the procedure used to compile and link the Prestressed Concrete I-Beam Design and Analysis Program. After the procedure is described, directions will be given illustrating how to create data input files and run the program on the microcomputer.

A.5.1 Program Description (Described in Reference 14)

The program was written to design or review a simply supported, prestressed, pretensioned concrete composite I-beam. The program was originally obtained from the Florida State Department of Transportation and has been revised in accordance with VDH&T modifications to AASHTO bridge specifications. Every attempt has been made to minimize the necessary input required to design or analyze an AASHTO standard type II through type VI beam. For I-beams other than AASHTO standard type, dimensions must be input.

Two types of strands are used by the program: stress relieved and low relaxation. Three strand sizes are used: 7/16", 1/2", and 9/16" diameters.

The program will compute moment and shear for HS-20, H-20, HS-15, and H-15 highway loadings. Railroad loadings include Cooper E-10, E-20, etc. Concentrated dead load and concentrated live load can be input separately.

The program determines the number of strands required by the bottom fiber stress at midspan due to all loads. A preliminary design is made by assuming that the midspan eccentricity is equal to the distance from the centroid of the beam to the bottom fiber. However, AASHTO 1.6.10(B) will be the controlling factor in all occasions.

Strands are placed beginning from the lowest row, then proceeding upward. Each row is started in the center position and progresses outward in both directions. The preliminary strand pattern will be modified when the top fiber tension at midspan exceeds the allowable, although bottom fiber stress is satisfactory. The modification is made by moving strands from a lower row to a higher row, thus reducing the midspan eccentricity and top tensile stress as well. The required end eccentricity is determined from the top and bottom fiber stresses in the end of the beam at the time of release. This eccentricity is obtained by draping all the strands in the central position in each horizontal row to a level that will furnish the required end eccentricity.

The predicted loss of prestress will be computed according to AASHTO 1.6.7(B)(1) in the Interim Specifications unless the designer has entered his own prediction.

The hold-down (draped) point will be located at one-tenth the span length rounded off to the nearest one-quarter foot on each side of midspan of the beam, or, optionally, any location point.

Additional information concerning input data preparation can be found in reference 14. This program is currently run on a mainframe using punched cards for data input. However, the input format remains the same on the microcomputer version with punched cards being replaced by an input file. As on the mainframe, the program runs in batch mode on the microcomputer.

A.5.2 Comments on Compiling and Linking

One thing that becomes apparent early when compiling FORTRAN, and other languages, on a 16-bit microcomputer is that it can be a very slow process. Therefore, it is important to develop a plan of action before actually beginning to compile, because for every effort detected the procedure must be repeated.

One way to help reduce overall compile time is to break the source file up into smaller files and compile each one separately. This is especially important, and usually a must, when compiling large programs such as the Prestressed Beam program. A schematic of how the Prestressed Beam program was broken up is shown in Figure A.5. The source files must be broken into groups of subroutines. Programs not containing subroutines cannot be broken up in this manner. The smaller files can be debugged and compiled individually and the resulting object modules linked with the runtime libraries during the linking phase. It is a good idea to save the object files. If the program requires modification later, only the affected source file needs to be recompiled. The new object module can then be linked with the unaffected ones to create the revised executable file.

Care should be taken in handling common blocks when breaking large source files into smaller files as described above. The MS-FORTRAN compiler will indicate an effort if named common blocks within a compiland are of different size. However, if two common blocks with the same name are in different compilands and are not the same size, no error will be indicated. This can cause the resulting executable file to develop hard-to-detect runtime errors or give erroneous results.

Although the run file created in this compilation will take advantage of an 8087 coprocessor if present, no special effort to accommodate the 8087 was made during the compilation. The MS-FORTRAN compiler contains special commands which will produce optimized code for use with an 8087.

A.5.3 Compile and Link Procedure

As can be seen in Figure A.5, before compiling began on the Prestressed Concrete I-Beam Program, the source file was broken down into ten smaller files. This was done primarily out of necessity since a single source file would be too large for the compiler to handle. (For a detailed explanation of the limitations on source code size see reference 6.) Also, as was mentioned before, breaking the program into smaller files makes it much more manageable. Additionally, a special MS-FORTRAN metacommand was inserted as the first line of each file. This metacommand is called $DEBUG and its use directs the compiler to produce code which will pinpoint runtime errors in the source file. Without using $DEBUG, detecting the causes of runtime errors would be extremely difficult. After the debugging process was completed, $DEBUG was removed and the compilation process repeated. This was done because with $DEBUG, the compiler generates about 40% more code, which slows execution and occupies additional RAM.

Six floppy diskettes were used in the compile-link process. The six diskettes and their contents are as follows:

DISK 1	DISK 2	DISK 3	DISK 4	DISK 5	DISK 6
FOR1.EXE	FORTRAN.LIB	PBEAM1.FOR	A.OBJ	PBEAM.EXE	Blank
PAS2.EXE	MATH.LIB	PBEAM2.FOR	B.OBJ		(used
PE.EXE (IBM	LINK.EXE	PBEAM3.FOR	C.OBJ		to hold
PE.HLP Personal		PBEAM4.FOR	D.OBJ		inter-
PE.PRO Editor)		PBEAM5.FOR	E.OBJ		mediate
		PBEAM6.FOR	F.OBJ		files
		PBEAM7.FOR	G.OBJ		created
		PBEAM8.FOR	H.OBJ		by
		PBEAM9.FOR	I.OBJ		Pass 1)
		PBEAM10.FOR	J.OBJ		

Disk one contains compiler passes one and two and the page editor. Disk two contains the FORTRAN runtime libraries and the linker. Disk three contains the program FORTRAN source files. Disk four contains the relocatable object files created by pass two of the compiler. Disk five contains the executable run file, which is the end product of the compile-link process. Disk six holds the temporary intermediate files created during pass one of the compiler.

The steps in the compile and link procedures are as follows:

COMPILING

1. Boot the operating system.

2. Log onto drive B.

3. Place DISK 1 in drive A and DISK 6 in drive B.

4. Invoke Pass one of the compiler by typing A:FOR1 and hitting RETURN. The following prompts will appear on the screen:

```
Source file [.FOR]:
Object file [.OBJ]:
Source listing [NUL.LST]:
Object listing [NUL.COD]:
```

5. Replace DISK 1 in drive A with DISK 3.

6. In response to the Source file prompt type A:PBEAM1 and hit RETURN. (The .FOR extension is automatically added.)

7. In response to the Object file prompt type A and hit RETURN. (The .OBJ extension is automatically added.) This will cause the object file corresponding to PBEAM1.FOR to be named A.OBJ.

8. If a source listing file is desired (it is optional) type any valid file name in response to the Source listing prompt and hit RETURN. Otherwise, just hit the RETURN key.

9. If an object listing file is desired (also optional) type any valid file name in response to the Object listing prompt and hit RETURN. Otherwise, just hit the RETURN key. The compiler will begin Pass one after the last prompt is responded to.

10. After Pass one is complete, log onto drive A and replace DISK 3 with DISK 1.

11. Invoke Pass two of the compiler by typing PAS2 B/PAUSE (<u>DO NOT HIT RETURN YET</u>.) The following message will appear:

    Press enter key to begin pass two.

12. Now replace DISK 1 with DISK 4 and hit RETURN. Pass two will now begin. When the disk drives stop moving, check drive A to verify that the object file has been written to disk.

13. Repeat steps 2 thru 12 for source files PBEAM1.FOR, PBEAM3.FOR,........PBEAM10.FOR. Note that object files B.OBJ, C.OBJ, etc. correspond to those respective source files (See Figure A.5)

LINKING

1. With compiling complete, all the object files will now be on Disk 4. Log onto drive A and place DISK 4 in this drive.

2. Place DISK 2 containing the linker and runtime libraries in drive B. Copy the runtime library files FORTRAN.LIB and MATH.LIB onto DISK 4. Type B:LINK and hit RETURN. The following prompts will appear on the screen:

   Object modules [.OBJ]:
   Run file [.EXE]:
   List map [NUL.MAP]:
   Libraries [.LIB]:

3. Replace DISK 2 in drive B with DISK 5.

4. In response to the Object files prompt type:

   A+B+C+D+E+F+G+H+I+J and hit RETURN.

5. In response to the Run file prompt type B:PBEAM and hit RETURN. (The .EXE extension will be added automatically.)

6. In response to the List map prompt just hit RETURN.

7. In response to the Libraries prompt type A; and hit RETURN. After this last RETURN has been hit the linker will begin processing.

8. When the disk drives have stopped, check DISK 5 in drive B to see if the run file PBEAM.EXE is there.

<u>Note 1</u>:
The list map will in most instances not be required. It is useful, however, for determining the size of the loaded program. This helps determine internal memory requirements for programs.

This completes the compile-link procedure for the Prestressed Beam Design and Analysis Program.

A.5.4 Prestressed Beam Program Source Listing

This section contains the FORTRAN source listing for the microcomputer converted Prestressed Beam Design and Analysis Program. The changes that have been made are indicated in bold type. These changes are listed in Table A.2 and are cross referenced using the numbers to the left of each affected line.

LISTING

```
[1]         PROGRAM MAIN
[2]         CHARACTER*80 WORDS
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
            REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
            COMMON/ILL/ REQULT,ULTMOM,FPC,FPCI,NSTATE,MSTATE,K
            COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
           *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
           *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
           *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
            COMMON/TJH/ B,WD,C,E,A
            COMMON/LI/AR(11),YB1(11),YT1(11),D1(11),IB1(11),WTF1(11),
           *BPRIME(11),HH(11),GG(11),DIAGD(11),DIAGW(11)
            COMMON/HD/ H,G
            COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
            COMMON/BNS/ BNSTD
            COMMON/MM/ROW(30),NROW,SROW(18),IW,DROW(18),NSROW(30)
[3]         COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
            COMMON/CONC/ CNCP(20),CNCD(20),CCP(20),CCD(20),SCNCP(10),SCNCD(10)
            COMMON/FYB/KGRID,NSTRNS,ENDECC,IWCH
            COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
           *DNCDL2,DNCDL1
            COMMON/MMM/ FO,HDPT,P,COPE
            COMMON/AJH/C1,C2,C3,C4,C5,C6,LOLAX
            COMMON/ALL/ FBII,ACOMPR,TTEN,FTP,PLOSS,PPERST,RLOSS,ITT
            COMMON/DEF/ SUMSTR,ECALE,SHIELD,DIST,CMAX
            DIMENSION CHRCTR(6)
[4]   C     DIMENSION ICARD(26)
            DIMENSION DIA1(10),SAREA1(9),SAREA2(9)
            DIMENSION BTYPEA(3),SPANLA(3),BSPACA(3),TSA(3),SMBOLA(3),
           * SMBOLB(3), IWA(3),RROADA(3),DFACTA(3),VFACTA(3),
           * HDPTA(3),OCDLA(3),CDLA(3),EFWA(3),PLOSSA(3),COPEA(3),ITTA(3)
            DATA DIA1/' 3/8','3/8 ','7/16',' 1/2','1/2 ','9/16','.6  ','0.6 ',
           *'00.6','    '/
```

```
          DATA SAREA1/0.085,0.085,0.115,0.153,0.153,0.192,0.217,0.217,0.217/
          DATA SAREA2/0.080,0.080,0.108,0.144,0.144,0.192,0.215,0.215,0.215/
          DATA BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,DBLANK/' 1',
         *'1 ',' 2','2 ',' 3','3 ',' 4','4 '/
          DATA BLANKE,EBLANK,BLANKF,FBLANK/' 5','5 ',' 6','6 '/
          DATA CHRCTR/'HS-2',' H-2','H -2',' H-1','H -1','HS-1'/
    C
    C
    C     THIS IS THE MAIN DRIVER WHICH HANDLES ALL INPUT
    C
    C
    C
[5]       WRITE(*,1)
[5]     1 FORMAT(15X,'VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION'//
[5]      *32X,'BRIDGE DIVISION'//21X,'PRESTRESSED BEAM DESIGN AND ANALYSIS'
[5]      */////1X,'NOTE: '/1X,'ENTER INPUT FILE NAME FOR UNIT 5 PROMPT'/
[5]      *1X,'ENTER OUTPUT FILE NAME FOR UNIT 6 PROMPT'//)
[6]       CALL INITIAL
[7]       OPEN(5,FILE=' ')
        3 IBEG=1
[8]     5 FORMAT(A80)
[8]       READ (5,5) WORDS
[9] C     CALL REREAD
          IERR = 0
          SHIELD = 0.0
          ISTOP = 0
          K1=0
[10]   50 READ (5,60,END=111)NTYPE,CARD
    C      IF(NTYPE.EQ.9)  GO TO 111
[10]   60 FORMAT (I1,A79)
          GO TO (10,20,70,1240,90,110,180,230),NTYPE
          STOP 1
       10 K1=K1+1
[10]      READ(CARD,30)BTYPEA(K1),SPANLA(K1),BSPACA(K1),TSA(K1),
         *SMBOLA(K1),SMBOLB(K1),IWA(K1),RROADA(K1),DFACTA(K1),VFACTA(K1),
         *HDPTA(K1),OCDLA(K1),CDLA(K1),EFWA(K1),PLOSSA(K1),COPEA(K1),
         *ITTA(K1)
          GO TO 50
[10] 20   READ(CARD,80)IB,AREA,D,YB,YT,B,WD,C,E,A,H,G
     80   FORMAT(F8.2,F6.2,F5.2,9F4.2)
          GO TO 50
[10] 70   READ(CARD,100)DIA,LOLAX,FPS,UWB,UWS,SFPC,FPC,EC,ECSL,ES,
         *STBCL,STSCL,SPACE
     100  FORMAT(A4,I1,3F5.2,2F6.2,3F4.2,3F4.3)
          IF(FPS.EQ.0.0) FPS = 270.
          FPS = FPS * 1000.
          IF(UWB.EQ.0.0) UWB = 150.
          IF(UWS.EQ.0.0) UWS = 150.
          IF(SPACE.EQ.0.0) SPACE = 2.
          IF(STBCL.EQ.0.0) STBCL=2.0
          IF(STSCL.EQ.0.0) STSCL=2.
          IF (SFPC .EQ. 0.0) SFPC=4500.0
    C  REVISED 3-28-84 REQUEST NUMBER  3190 FRANK CHEN
           IF(ECSL.EQ.0.0)ECSL=4.07
```

```
              IF(EC.EQ.0.0)  EC=4.29
              IF(ES.EQ.0.0) ES=29.
              IF(FPC.EQ.0.0) FPC = 5000.
              FPCI=0.8*FPC
              IF(EC.EQ.0.0) EC=UWB**1.5 * 33.0 *  FPC**0.5/ 1000000.0
        290   CONTINUE
              LTYPE=0
              DO 300  IN = 1,10
              IF(DIA.EQ.DIA1(IN)) GO TO 310
        300   CONTINUE
        310   CONTINUE
              IF(IN.GE.10) IN = 4
              DIA = DIA1(IN)
              IF(FPS.EQ.270000.) GO TO 320
              ASTRN = SAREA2(IN)
              GO TO 50
        320   ASTRN = SAREA1(IN)
              GO TO 50
[10]    90    READ(CARD,120)(CNCP(I),I=1,10)
[10]    99    FORMAT(A79)
[10]          READ(5,99)CARD
        121   FORMAT(1X,10F5.2)
        120   FORMAT(10F5.2)
              DO 130  I=1,10
              IF(CNCP(I).EQ.0.0) GO TO 140
        130   CONTINUE
              I=11
        140   NCL=I-1
              DO 150  I=1,10
              IF(CNCD(I).EQ.0.0) GO TO 160
        150   CONTINUE
              I=11
        160   NCD=I-1
              IF(NCL.EQ.NCD) GO TO 50
              WRITE(6,170)
        170   FORMAT(T30,'ERROR IN CONCENTRATED LOAD INPUT')
              GO TO 420
[10]    110   READ(CARD,120)(SCNCP(I),I=1,10)
[10]          READ(5,99)CARD
[10]          READ(CARD,121)(SCNCD(I),I=1,10)
              DO 190  I=1,10
              IF(SCNCP(I).EQ.0.0) GO TO 200
        190   CONTINUE
              I=11
        200   NSP = I-1
              DO 210  I=1,10
              IF(SCNCD(I).EQ.0.0) GO TO 220
        210   CONTINUE
              I=11
        220   NSD = I-1
              IF(NSP.EQ.NSD) GO TO 50
              WRITE(6,170)
              GO TO 420
```

```
[10]  150   READ(CARD,120)(CCP(I),I=1,10)
[10]        READ(5,99)CARD
[10]        READ(CARD,121)(CCD(I),I=1,10)
            DO 240  I=1,10
            IF(CCP(I).EQ.0.0) GO TO 250
      240   CONTINUE
            I=11
      250   NL=I-1
            DO 260  I=1,10
            IF(CCD(I).EQ.0.0) GO TO 270
      260   CONTINUE
        270 ND=I
            IF(NL.EQ.ND) GO TO 50
            WRITE(6,170)
            GO TO 420
[10]  230   READ(CARD,120)(CCP(I),I=11,20)
[10]        READ(5,99)CARD
[10]        READ(CARD,121)(CCD(I),I=11,20)
            DO 2401 I=11,20
            IF(CCP(I).EQ.0.0) GO TO 2501
      2401  CONTINUE
            I=21
      2501  NL=I-1
            DO 2601 I=11,20
            IF(CCD(I).EQ.0.0) GO TO 2701
      2601  CONTINUE
       2701 ND=I
            IF(NL.EQ.ND) GO TO 50
            WRITE(6,170)
            GO TO 420
[10]  1240  READ(CARD,241)(SROW(I),I=1,18),KGRID
[10]        READ(5,99)CARD
[10]        READ(CARD,242)(NSROW(I),I=1,18),SHIELD
[10]        READ(5,99)CARD
[10]        READ(CARD,243)(DROW(I),I=1,18)
[10]  243   FORMAT(1X,18F2.0,F4.2)
[10]  241   FORMAT(18F2.0,F4.2)
[10]  242   FORMAT(1X,18I2,F4.2)
            IVIEW = 1
            GO TO 50
      111   SFPC1 = FPC
            SFPCI = FPCI
        280 DO 400 KN=1,K1
            IERR=0
      30    FORMAT(A2,F5.2,2(F4.2),A4,A1,I1,F3.0,2F4.3,F4.2,2F5.3,F4.2,
           *F3.1,F4.3,I1)
            BTYPE=BTYPEA(KN)
            SPANL=SPANLA(KN)
            BSPAC=BSPACA(KN)
             TSS = TSA(KN)
             TS=TSS-.5
            SMBOL1=SMBOLA(KN)
            SMBOL2=SMBOLB(KN)
            IW=IWA(KN)
```

```
          RROAD=RROADA(KN)
          DFACT=DFACTA(KN)
          VFACT=VFACTA(KN)
          HDPT=HDPTA(KN)
          NCDL=OCDLA(KN)
          CDL=CDLA(KN)
          EFW=EFWA(KN)
          PLOSS=PLOSSA(KN)
          ITT=ITTA(KN)
          COPE=COPEA(KN)
       40 IF(IW.EQ.0) IW=2
          FPC = SFPC1
          FPCI = SFPCI
          C6=-6.
[11]      IF(BTYPE.EQ.BLANKA.OR.BTYPE.EQ.ABLANK)BTYPE=BLANKA
[11]      IF(BTYPE.EQ.BLANKB.OR.BTYPE.EQ.BBLANK)BTYPE=BLANKB
[11]      IF(BTYPE.EQ.BLANKC.OR.BTYPE.EQ.CBLANK)BTYPE=BLANKC
[11]      IF(BTYPE.EQ.BLANKD.OR.BTYPE.EQ.DBLANK)BTYPE=BLANKD
[11]      IF(BTYPE.EQ.BLANKE.OR.BTYPE.EQ.EBLANK)BTYPE=BLANKE
[11]      IF(BTYPE.EQ.BLANKF.OR.BTYPE.EQ.FBLANK)BTYPE=BLANKF
          IF(BTYPE.EQ.BNSTD) GO TO 42
          DO 41 I = 1,11
          IF(BTYPE.EQ.BEAM(I)) GO TO 42
    41    CONTINUE
          IERR = 1
    42    IF(SPANL.LE.0.OR.SPANL.GT.170.) IERR = 2
          IF(BSPAC.EQ.0.) IERR = 3
           IF( TSS.EQ.0.0)IERR=4
          IF(IERR.NE.0) GO TO 410
          IF (DFACT.NE.0.0) GO TO 341
          DFACT = BSPAC/11.
    341   IF(VFACT.EQ.0.0) VFACT = 1.0
          LTYPE = 0
          IF(SMBOL1.EQ.CHRCTR(1)) LTYPE = 1
          IF(SMBOL1.EQ.CHRCTR(2).OR.SMBOL1.EQ.CHRCTR(3)) LTYPE = 2
          IF(SMBOL1.EQ.CHRCTR(4).OR.SMBOL1.EQ.CHRCTR(5)) LTYPE = 3
          IF(SMBOL1.EQ.CHRCTR(6)) LTYPE = 5
          IF(RROAD.NE.0.0) LTYPE = 4
          IF(SPANL.GT.0.00.AND.SPANL.LE.50.0) KODE = 1
          IF(SPANL.GT.50.0.AND.SPANL.LE.90.0) KODE = 2
          IF(SPANL.GT.90.0.AND.SPANL.LE.130.) KODE = 3
          IF(SPANL.GT.130..AND.SPANL.LE.170.) KODE = 4
          CALL PROPTY
          CALL HELP
    380   CALL PSTRES
          CALL OUTPUT
          CALL ZERO
          GO TO 400
    410   WRITE(6,415) IERR,NTYPE,CARD
    415   FORMAT('  ERROR NUMBER',I4,' IN INPUT CARD ',I1,A79)
      400 CONTINUE
           IF(NTYPE.EQ.9)  GO TO 3
    420   STOP
          END
```

```
      C   DATA SET PRSTRSBM   AT LEVEL 002 AS OF 06/16/78
            SUBROUTINE ALLOW
      C
[2]         CHARACTER*80 WORDS
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
            COMMON/ALL/ FBII,ACOMPR,TTEN,FTP,PLOSS,PPERST,RLOSS,ITT
            COMMON/ILL/ REQULT,ULTMOM,FPC,FPCI,NSTATE,MSTATE,K
            COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
           *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
           *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
           *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]         COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
            COMMON/AJH/C1,C2,C3,C4,C5,C6,LOLAX
      C
      C
      C     DETERMINE ALLOWABLE STRESSES
      C
      C
            FBII = 0.6*FPCI
            ACOMPR = 0.4*FPC
            IF(RROAD.NE.0.0) GO TO 1
            TTEN=-3.*SQRT(FPCI)
            IF(ITT.EQ.0) FTP=0
            IF(ITT.EQ.1)  FTP=-3.*FPC**0.5
            IF(ITT.EQ.2)  FTP=-6.*FPC**0.5
            GO TO 2
          1 TTEN=-3.*SQRT(FPCI)
            FTP = 0.0
      2     TENIN = C1*FPS*ASTRN
            PPERST = TENIN*(1.-PLOSS)
            RETURN
            END
            SUBROUTINE CAMBER
      C
[2]         CHARACTER*80 WORDS
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
            REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
            COMMON/FYB/KGRID,NSTRNS,ENDECC,IWCH
            COMMON/DEF/ SUMSTR,ECALE,SHIELD,DIST,CMAX
            COMMON/MM/ROW(30),NROW,SROW(18),IW,DROW(18),NSROW(30)
            COMMON/ALL/ FBII,ACOMPR,TTEN,FTP,PLOSS,PPERST,RLOSS,ITT
            COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
            COMMON/MMM/ FO,HDPT,P,COPE
            COMMON/ILL/ REQULT,ULTMOM,FPC,FPCI,NSTATE,MSTATE,K
            COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
           *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
           *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
           *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
```

```
[3]        COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      C
      C
      C   CAMBER CALCULATIONS
      C
      C
           F0 = TENIN * STRNS
           P = F0 * (1. - RLOSS)
           M1 = P * ENDECC
           M2 = P * ECCL
           IF(SHIELD.EQ.0.0) GO TO 1
      C  SHIELD IS ASSUMED EQUAL TO 0 , IF ASSUMPTION IS CHANGED THAN KDIST
      C AND SUMME FORMULARS HAVE TO BE CHANGED FOR CHANGE IN METHOD OF
      C EVALUATING STEEL DISTRIBUTION
           M3 = SUMSTR * ECALE * TENIN * (1. - PLOSS)
           M4 = P * (ENDECC + ((ECCL - ENDECC) * SHIELD / DIST))
           KDIST = (KGRID - ((NROW - 1) * SPACE + STBCL)) *(DIST -SHIELD )/
          * DIST
           KDIST = KDIST +(NROW -1)* SPACE +STBCL
           SUMME = NSROW(1) * STBCL + DROW(1) * (KDIST - (NROW - 1) * SPACE)
           DO 2 JR = 2,NROW
           SUMME = SUMME + NSROW(JR) * (STBCL + (JR - 1) * SPACE) + DROW(JR)
          * *  (KDIST - (NROW - 1) * SPACE + (JR - 1) * SPACE)
       2   CONTINUE
           ECALS = YB - SUMME / SUMSTR
           M5 = SUMSTR * ECALS * TENIN * (1. - PLOSS)
           DELS = SHIELD ** 2 / 6. * (M1 - M3 + 2 * M4 - 2 * M5) * 144.
       1   W = UWB * AREA / 144.
           DELB = 5. / 384. * (W * SPANL ** 4 * 1728.)
           DELPE=M1*SPANL**2/8.*144.
           DELPM=(M2-M1)*12.*(SPANL**2+2.*SPANL*HDPT-2.*HDPT**2)
           ECCD=1800000.+460.*FPCI
           CMAX=(DELPE+DELPM-DELB)/(ECCD*IB)
           RETURN
           END
           SUBROUTINE CONLD
      C
[2]        CHARACTER*80 WORDS
[2]        CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]        CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]       *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]        CHARACTER*1 SMBOLB,SMBOL2
           COMMON/CONC/ CNCP(20),CNCD(20),CCP(20),CCD(20),SCNCP(10),SCNCD(10)
           COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
          *DNCDL2,DNCDL1
           COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
          *0),BMSL(20),BMCDL(20)
           COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
          *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
          *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
          *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]        COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
           COMMON/MMM/ F0,HDPT,P,COPE
           DIMENSION V(15), BMM(15)
```

```
      DIMENSION VW(20),BMW(20)
C
C
C     CALCULATE SHEARS AND MOMENTS  INSPECTION POINTS DUE TO
C     CONCENTRATED LIVE LOADS
C
C
      DO 36 I = 1,20
      CCP(I) = CCP(I) * 1000.
36    CONTINUE
      DO 101 K = 1,20
      IF(CCP(K).LE.0.0) GO TO 102
      LW = K
101   CONTINUE
102   LW1 = LW + 1
      LW2 = LW + 2
      DO 103 K = 2,LW
103   CCD(LW2 - K) = CCD(LW1 - K)
      CCD(1) = 0.0
      DO 10 L = 1,15
      DO 9 M = 1,LW
      DIST = XDIST(L)
      CM = CCD(M)
      N = 0
4     N = N + 1
      CDIST = CM - CCD(N)
      IF(CDIST.GT.DIST) GO TO 4
      N1 = N
      SPMD = SPANL - DIST
      N = M - 1
5     N = N + 1
      IF(N.GT.LW) GO TO 6
      CCDIST = CCD(N) - DIST
      IF(CCDIST.LE.SPMD) GO TO 5
6     NN = N - 1
      CN = CCD(N1)
      SUMWC = 0.0
      SUMLD = 0.0
      DO 7 N = N1,NN
      SUMWC = SUMWC + (CCD(N) - CN) * CCP(N)
7     SUMLD = SUMLD + CCP(N)
      CBAR = SUMWC / SUMLD + DIST - CDIST
      SUBM = 0.0
      SUBW = 0.0
      REACTN = (1.0 - CBAR/SPANL) * SUMLD
      DO 8 N = N1,M
      SUBM = (CM - CCD(N)) * CCP(N) + SUBM
8     SUBW = SUBW + CCP(N)
      SUMM = 0
      IF(M.EQ.N1) GO TO 3
      MM1 = M - 1
      DO 2 N = N1,MM1
2     SUMM = SUMM + CCP(N)
3     V1 = ABS(REACTN - SUMM)
```

```
          V2 = ABS(REACTN - SUMM - CCP(M))
          VW(M) = AMAX1(V1,V2)
      9   BMW(M) = DIST * REACTN * 12.0 - SUBM * 12.0
          VMAX = 0
          BMAX = 0
          DO 11 M = 1,LW
          BMAX = AMAX1(BMAX,BMW(M))
          VMAX = AMAX1(VMAX,VW(M))
     11   CONTINUE
          V(L) = VMAX
          BMM (L) = BMAX
     10   CONTINUE
          DO 35 I = 1,11
          BMMA(I) = AMAX1(BMM(I),BMM(12-I))
          VMA(I) = AMAX1(V(I),V(12-I))
     35   CONTINUE
          BMMA(12) = AMAX1(BMM(12),BMM(13))
          BMMA(13) = BMMA(12)
          BMMA(14) = AMAX1(BMM(14),BMM(15))
          BMMA(15) = BMMA(14)
          VMA(12) = AMAX1(V(12),V(13))
          VMA(13) = VMA(12)
          VMA(14) = AMAX1(V(14),V(15))
          VMA(15) = VMA(14)
          RETURN
          END
          SUBROUTINE ECCEND
      C
[2]       CHARACTER*80 WORDS
[2]       CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]       CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]      *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]       CHARACTER*1 SMBOLB,SMBOL2
          REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
          COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
         *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
         *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
         *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]       COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
          COMMON/MM/ROW(30),NROW,SROW(10),IW,DROW(10),NSROW(30)
          COMMON/FYB/KGRID,NSTRNS,ENDECC,IWCH
          COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
          COMMON/DEF/ SUMSTR,ECALE,SHIELD,DIST,CMAX
      C
      C
      C   CALCULATE END ECCENTRICTY AND POSITION OF THE TOPMOST
      C   DRAPED STRANDS
      C
      C
          IF(IVIEW.NE.0) GO TO 20
          KGRID = 0
          IWCH = 1
          KS = 0
          IF(IW.EQ.1) WBK = 1.0
```

```
      IF(IW.EQ.2) WBK = 2.0
      IF(IW.EQ.3) WBK = 3.0
      TDS=0.0
      DO 1 JR=1,NROW
      SROW(JR)=ROW(JR)-WBK
      IF(SROW(JR).LE.0.0) GO TO 12
      DROW(JR)=WBK
      GO TO 11
   12 DROW(JR)=ROW(JR)
      SROW(JR)=0.0
   11 TDS=TDS+DROW(JR)
    1 CONTINUE
      X=STBCL
      SUMD1=SROW(1)
      SUMD2=DROW(1)
      SUMDW1=SROW(1)*X
      SUMDW2=DROW(1)*X
      DO 2 I=2,NROW
      X=X+SPACE
      SUMD1=SUMD1+SROW(I)
      SUMDW1=SUMDW1+SROW(I)*X
      SUMD2=SUMD2+DROW(I)
    2 SUMDW2=SUMDW2+DROW(I)*X
      XBAR1=SUMDW1/SUMD1
      XBAR2=SUMDW2/SUMD2
      XBAR22=XBAR2
    3 CGT=(SUMDW1+SUMD2*XBAR22)/STRNS
      CGTLMT=YB-ENDMAX
      IF(CGT.GE.CGTLMT) GO TO 4
      XBAR22=XBAR22+SPACE
      KS=KS+1
      X1=STBCL+(NROW-1+KS)*SPACE
      IF(X1.GT.(D-2.0)) GO TO 5
      GO TO 3
    5 IWCH=2
      X1=D-2.
      N=(X1-STBCL)/2
      X1=STBCL+2*N
    4 CONTINUE
      IF (KS .EQ. 0) X1=STBCL+(NROW-1)*SPACE
      NSTRNS=TDS
      KGRID=X1
      ENDECC=YB-CGT
      RETURN
20    SUMD1 = SROW(1)
      SUMD2 = DROW(1)
      SUMD3=NSROW(I)
      SUMDW1 = SROW(1) * STBCL
      SUMDW2=DROW(1)*KGRID
      SUMDW3=NSROW(I)*STBCL
      DO 22 I=2,18
      SUMD1=SUMD1+SROW(I)
      SUMD3=SUMD3+NSROW(I)
      SUMD2=SUMD2+DROW(I)
```

```
            SUMDW1=SUMDW1+SROW(I)*(STBCL+(I-1)*SPACE)
         22 SUMDW2=SUMDW2+DROW(I)*(KGRID-(I-1)*SPACE)
            SUMDW3=SUMDW3+NSROW(I)*(STBCL+(I-1)*SPACE)
            CGT = (SUMDW1 + SUMDW2) / (SUMD1 + SUMD2)
            ENDECC = YB - CGT
            ECALE = YB-(SUMDW2+SUMDW3)/(SUMD2+SUMD3)
            RETURN
            END
            SUBROUTINE HELP
      C
[2]         CHARACTER*80 WORDS
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
            REAL NCDL,IB,INA,LESV
            COMMON/CONC/ CNCP(20),CNCD(20),CCP(20),CCD(20),SCNCP(10),SCNCD(10)
            COMMON /ILL/ REQULT,ULTMOM,FPC,FPCI,NULL(3)
            COMMON/MMM/ FO,HDPT,P,COPE
            COMMON/BNS/ BNSTD
            COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
           *0),BMSL(20),BMCDL(20)
            COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
           *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
           *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
           *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[2]         COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
            COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
           *DNCDL2,DNCDL1
            COMMON/KAP/ W,WCP
            COMMON/MSC/ VNCDL(15),VCDL(15)
            COMMON / CHEN / VMMS(20),VSPC(20)
            COMMON / CHEN2 / DCDL1,DCDL2
            DIMENSION CONST(2,4),POINT(4)
            DIMENSION VMM(15),BMM(15) ,BMMS(15),VMMMS(20)
            COMMON/JJJ/ BB(11),WDD(11),CC(11),EE(11)
            COMMON/LI/AR(11),YB1(11),YT1(11),D1(11),IB1(11),WTF1(11),
           *BPRIME(11),HH(11),GG(11),DIAGD(11),DIAGW(11)
            DATA VMM,BMM,BMMS/45*0.0/
            DATA CONST/1.4322,2.0833,2.5174,3.5494,3.5156,5.4598,3.9333,6.300/
      C
      C     CALCULATE INSPECTION POINTS, AND THEIR RESPECTIVE SHEARS,
      C     MOMENTS AND DEFLECTIONS
      C
      C
[13]        DO 999 III=1,20
[13]    999 VMMS(III)=0.0
            CDL = CDL * 1000.
            NCDL = NCDL*1000.
[14]  C THE FOLLOWING 1 CARD REMOVED PER FRANK CHEN 3-21-1985.
[14]  C     NCDL=NCDL+COPE*WTF*UWS/144.
      C
      C     DETERMINE INSPECTION POINTS
      C
```

```
      TSPAN = SPANL * 0.1
      DO 5 I = 1,11
5     XDIST(I) = (I - 1) * TSPAN
      XDIST(12) = SPANL * 0.25
      XDIST(13) = SPANL * 0.75
      XDIST(14) = SPANL / 2.0 - HDPT
      XDIST(15) = SPANL / 2.0 + HDPT
C
C     COMPUTE DEFLECTIONS DUE TO THE WEIGHT OF THE BEAM,
C     SLAB AND DIAPHRAGMS
C
      DEFL14 = 0.0
      DEFL12 = 0.0
      WCP = 0.0
       TSS = TSS / 12.0
      ECI=(1800000.+460.*0.8*FPC)*IB
      ECSI=(1800000.+460.*FPC)*INA
[14]  C THE FOLLOWING 1 CARD CHANGED ON 3-21-1985 PER F. CHEN (WS=UWS*TSS*BSPA
[14]   WS = UWS * TSS * BSPAC + COPE * WTF * UWS / 144.
      WB = UWB * AREA / 144.0
      W = WS + WB
      REACTN = .5*W*SPANL
      WNCDL = NCDL/12.0
      WCDL =  CDL / 12.0
      BMREAC = 0.5*WB*SPANL
      RNCDL = 0.5*NCDL*SPANL
      RCDL = 0.5 * CDL * SPANL
      SPANN = SPANL * 12.0
      WI = WS/12.
      DEFK2 = 5.0* WI* SPANN ** 4/( 384.0*ECI)
      DEFK1 = 57.0* WI* SPANN ** 4/(6144.0*ECI)
      DNCDL1 = 57.0*WNCDL*SPANN**4/(6144.0*ECI)
      DCDL1 = 57.0 * WCDL * SPANN ** 4 / (6144.0 * ECSI)
      DNCDL2 = 5.0*WNCDL*SPANN**4/(384.0*ECI)
      DCDL2 = 5.0 * WCDL * SPANN ** 4 / (384.0 * ECSI)
      IF(BTYPE.EQ.BNSTD) GO TO 50
      DIAA = DIAGD(KASE) * DIAGW(KASE)
      DIAV = DIAA * (BSPAC - WDD(KASE)/12.)
      LESV = ((2*HH(KASE)+GG(KASE))*GG(KASE))*DIAGW(KASE)/144.0
      DIAV = DIAV - LESV
      IF(KASE.LE.4) GO TO 10
      DIAV=DIAV-(.83333+.4167)*1.0833*DIAGW(KASE)
   10 CONTINUE
      CP = DIAV * UWS
      REACTN = REACTN + (.5*KODE*CP)
      CONST1 = CP * SPANN ** 3 / (ECI * 100.0)
      WCP = (KODE*CP)/SPANL
      DEFL14 = CONST(1,KODE)*CONST1
      DEFL12 = CONST(2,KODE)*CONST1
C
C     COMPUTE DEFLECTIONS, BENDING MOMENTS AND SHEARS DUE TO
C     CONCENTRATED LOADS
C
50    DO 55 I = 1,10
```

```
      IF(CNCP(I).EQ.0.0) GO TO 56
55    CNCP(I) = CNCP(I) * 1000.
56    CONTINUE
      X12 = SPANN * 0.5
      X14 = SPANN * 0.25
      X122 = X12 * X12
      X142 = X14 * X14
      ECSI6 = 6.0*ECSI
      ECI6 = 6. * ECI
      X12L = X12 * SPANN
      X14L = X14 * SPANN
      DO 30 N = 1,10
      NN = N
      IF(CNCD(N).LE.0.0) GO TO 31
30    CONTINUE
31    N2 = NN/2
      N22 = N2 * 2
      IF(N2.LE.0) GO TO 35
      DO 34 N = 1,N2
      PDL = CNCP(N)
      P12 = PDL * X12
      P14 = PDL * X14
      DX = CNCD(N) *12.
      PDX = PDL * DX
      DX2 = DX * DX
      IF (DX.GT.X14) GO TO 33
      DEFL14 = DEFL14 + PDX * (3.0*(X14L-X142)-DX2)/ECI6
      GO TO 34
33    DEFL14 = DEFL14 + P14 * (3.0*(SPANN*DX-DX2) - X142)/ECI6
34    DEFL12 = DEFL12 + P12 * (3.0*X12L-X122-DX2)/ECI6
35    IF(NN.EQ.N22)  GO TO 36
      PNN1 = CNCP(N2+1)
      DEFL14 = DEFL14 + PNN1*X14*(3.0*SPANN**2-4.*X142)/(48.*ECI)
      DEFL12 = DEFL12 + PNN1*SPANN**3/(48.*ECI)
36    CONTINUE
      SUMW = 0.
      SUMWC=0
      DO 37 I = 1,NN
      SUMW=SUMW+CNCP(I)
   37 SUMWC=SUMWC+CNCP(I)*CNCD(I)
      CBAR=SUMWC
      IF(SUMW.EQ.0.0) GO TO 137
      CBAR=SUMWC/SUMW
  137 REACT1=SUMW*(1.0-CBAR/SPANL)
      DO 39 I = 1,15
      X = XDIST(I)
      VM = 0
      BM = 0
      DO 38 L = 1,NN
      IF(CNCD(L).GE.X) GO TO 87
      VM = VM + CNCP(L)
38    BM = BM + CNCP(L) * (X-CNCD(L))
   87 BMM(I)=(REACT1*X-BM)*12.
      VMM(I)=REACT1-VM
```

```
39    CONTINUE
      SDEF12 = 0
      SDEF14 = 0
C
C     SCNCP(N) = WEIGHT OF STATIC CONC DEADLOAD N
C     SCNCD(N) = DISTANCE FROMLEFT REACTION TO THE NTH LOAD
C
      IF(SCNCP(1).EQ.0) GO TO 86
      DO 120 I=1,10
120   SCNCP(I) = SCNCP(I)*1000.
      DO 80 N = 1,10
      NN = N
      IF(SCNCP(N).LE.0.0) GO TO 81
80    CONTINUE
81    SUMW = 0
      SUMWC = 0
      DO 82 I = 1,NN
      SUMW = SUMW + SCNCP(I)
82    SUMWC = SUMWC + SCNCP(I) * SCNCD(I)
      CBAR = SUMWC / SUMW
      REACT1 = (1.0 - CBAR/SPANL) * SUMW
      DO 85 I = 1,15
      X = XDIST(I)
      BM = 0
      VM = 0
      DO 83 M = 1,NN
      IF(X.LE.SCNCD(M)) GO TO 84
      VM=VM+SCNCP(M)
83    BM = SCNCP(M) * (X-SCNCD(M)) + BM
84    VMMS(I) = REACT1 - VM
   85 BMMS(I)=(REACT1*X-BM)*12
86    CONTINUE
      DO 40 I = 1,11
      BMMS(I) = AMAX1(BMMS(I),BMMS(12-I))
      VMMS(I) = AMAX1(VMMS(I),VMMS(12-I))
      BMM(I) = AMAX1(BMM(I),BMM(12-I))
40    VMM(I) = AMAX1(VMM(I),VMM(12-I))
      BMM(12) = AMAX1(BMM(12),BMM(13))
      BMM(13) = BMM(12)
      BMM(14) = AMAX1(BMM(14),BMM(15))
      BMM(15) = BMM(14)
      VMM(12) = AMAX1(VMM(12),VMM(13))
      VMM(13) = VMM(12)
      VMM(14) = AMAX1(VMM(14),VMM(15))
      VMM(15) = VMM(14)
      VMMS(12) = AMAX1(VMMS(12),VMMS(13))
      BMMS(12) = AMAX1(BMMS(12),BMMS(13))
      VMMS(14) = AMAX1(VMMS(14),VMMS(15))
      BMMS(14) = AMAX1(BMMS(14),BMMS(15))
      BMMS(15) = BMMS(14)
      VMMS(15) = VMMS(14)
      BMMS(13) = BMMS(12)
      VMMS(13) = VMMS(12)
C
```

```
C     COMPUTE MOMENTS AND SHEARS DUE TO DEAD LOAD
C
51    SLREAC = 0.5 * (WS + WCP ) * SPANL
      KK = 0
      N1 = 1
      N2 = 6
      N3 =1
3     DO 15 I = N1,N2,N3
      X = XDIST(I)
      VDL(I) = REACTN -( W + WCP ) * X
      VNCDL(I) = RNCDL - NCDL*X + VMM(I)
      VCDL(I)=RCDL-CDL*X
      BMDL(I) = (REACTN - (W + WCP)*X/2.)*X*12.
      BMBM(I) = (BMREAC-WB*X/2.)*X*12.
      BMSL(I) = (SLREAC -(WS +WCP ) * X/2.) * X * 12.
      BMNCDL(I) = (RNCDL - NCDL*X/2.)*X*12. + BMM(I)
      BMCDL(I) = (RCDL - CDL * X / 2.) * X * 12. + BMMS(I)
15    CONTINUE
      IF (KK.GT.0) GO TO 17
      KK = KK+1
      N1 = 12
      N2 = 15
      N3 = 2
      GO TO 3
17    CONTINUE
      DO 16 I =7,11
      BMDL(I) = BMDL(12-I)
      BMBM(I) = BMBM(12-I)
      BMSL(I) = BMSL(12-I)
      VNCDL(I) = VNCDL(12-I)
      VCDL(I) = VCDL(12 - I)
      BMNCDL(I) = BMNCDL(12-I)
      BMCDL(I) = BMCDL(12 - I)
16    VDL(I) = VDL(12-I)
C
C     CALCULATE MAXIMUM SHEARS AND MOMENTS INCLUDING LIVE LOAD
C     AT THE INSPECTION POINTS
C
      IF(LTYPE.EQ.0) GO TO 24
      CALL TYPELD
24    DO 21 I =1,11
      VSUM(I) = VMA(I) + VDL(I) +VNCDL(I) + VMMS(I) + VCDL(I)
      BMSUM(I)=BMDL(I)+BMMA(I)+BMNCDL(I)+BMCDL(I)
21    CONTINUE
      DO 23 I = 12,15,2
      VSUM(I) = VMA(I) + VDL(I) + VNCDL(I) + VMMS(I) + VCDL(I)
      BMSUM(I)=BMDL(I)+BMMA(I)+BMNCDL(I)+BMCDL(I)
23    CONTINUE
      VSUM(13) = VSUM(12)
      VSUM(15) = VSUM(14)
      BMSUM(13) = BMSUM(12)
      BMSUM(15) = BMSUM(14)
      BMSL(13) = BMSL(12)
      BMSL(15) = BMSL(14)
```

```
      BMBM(13) = BMBM(12)
      BMBM(15) = BMBM(14)
      BMDL(13) = BMDL(12)
      BMDL(15) =BMDL(14)
      BMNCDL(13) = BMNCDL(12)
      BMNCDL(15)  = BMNCDL(14)
      BMCDL(13) =BMCDL(12)
      BMCDL(15) = BMCDL(14)
       TS = TSS * 12.0 - 0.5
C
C  CST/DPG 8/81
C
      DO 223 I=1,15
  223 VMMMS(I)=VMMS(I)
      RETURN
      END
      SUBROUTINE JMLOAD(TOTTLD)
C
[2]   CHARACTER*80 WORDS
[2]   CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]   CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]  *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]   CHARACTER*1 SMBOLB,SMBOL2
      COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
     *DNCDL2,DNCDL1
      COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
     *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
     *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
     *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]   COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
     *0),BMSL(20),BMCDL(20)
      DIMENSION       C(3)
      DIMENSION BMPRIM(6)
      DIMENSION BMW(3)
      DIMENSION V(20)
      COMMON/J/ BMHS(20),BMSP(20),BMLL(20),VHS(20),VSP(20),VLL(20)
      DATA BMPRIM/2.8,0.0,2.8,0.0,16.8,11.2/
      DATA C/0.0,14.0,28.0/
      DATA V/20*0.0/
C
C
C     CALCULATE SHEARS AND MOMENTS  INSPECTION POINTS DUE TO HS-20
C     LIVE LOADS
C
C
      CL = SPANL * 0.5
      IF(SPANL.GT.24) GO TO 300
      PT = CL
      REACTN = 0.4
      BM = REACTN * PT * 480000
      GO TO 305
300   IF(SPANL.GT.28) GO TO 301
      PT = CL - 3.5
```

```
      CBAR = CL + 3.5
      REACTN = (1-CBAR/SPANL)*1.6
      BM = REACTN * PT * 480000
      GO TO 305
301   IF(SPANL.GT.32.67) GO TO 302
      PT = 14.0
      CBAR = 18.667
      GO TO 303
302   PT = CL - 2.33
      CBAR = CL + 2.33
303   REACTN = (1-CBAR/SPANL) * 1.8
304   BM = (PT * REACTN - 2.8) * 40000
305   CONTINUE
      ULTM = BM
      DO 18 LD = 1,15
      DIST = XDIST(LD)
      DO 17 M = 2,3
      BMW(M) = 0.0
      SUMWC = 0.0
      SUMLD = 0.0
      GO TO (1,1,8),M
1     IF(DIST.LE.C(2)) GO TO 3
      DFSTW = DIST - C(2)
      DLSTW = DFSTW + C(3)
      IF(DLSTW.GE.SPANL) GO TO 5
      KACE = 1
2     SUMLD = 1.8
      SUMWC = 33.6
      GO TO 10
3     DFSTW = DIST
      DLSTW = DFSTW + 14.0
      IF(DLSTW.GT.SPANL) GO TO 7
      KACE = 2
4     SUMLD = 1.6
      SUMWC = 11.2
      GO TO 10
5     KACE = 3
6     SUMLD = 1.0
      SUMWC = 11.2
      GO TO 10
7     SUMLD = 0.8
      SUMWC = 0.0
      GO TO 10
8     IF(DIST.LE.28.0) GO TO 9
      KACE = 5
      DFSTW = DIST - 28.0
      GO TO 2
9     DFSTW = DIST - C(2)
      IF(DFSTW.LE.0.0) GO TO 16
      KACE = 6
      GO TO 4
10    CBAR = SUMWC/SUMLD + DFSTW
      REACTN = (1.0 - CBAR/SPANL) * SUMLD
      BMW(M)=(DIST * REACTN - BMPRIM(KACE)) * TOTTLD
```

```
16      CONTINUE
17      CONTINUE
        BM    = AMAX1(BMW(2),BMW(3))
        BMHS(LD) = BM*12.
        DFSTW = DIST
        DLSTW = DFSTW + C(3)
        IF(DLSTW.GT.SPANL) GO TO 11
        CBAR = DIST + 9.33
        REACTN = (1.0 - CBAR/SPANL) * 1.6
        GO TO 13
11      DLSTW = DFSTW + C(2)
        IF(DLSTW.GT.SPANL) GO TO 12
        CBAR = DIST + 7.0
        REACTN = (1.0 - CBAR/SPANL) * 1.6
        GO TO 13
12      REACTN = (1.0 - DIST/SPANL) * 0.8
13      V(LD) = REACTN * TOTTLD
18      CONTINUE
        DO 20 LD = 1,11
        VHS(LD) = AMAX1(V(LD),V(12-LD))
20      BMHS(LD) = AMAX1(BMHS(LD),BMHS(12-LD))
19      CONTINUE
        VHS(12) = AMAX1(V(12),V(13))
        VHS(13) = VHS(12)
        VHS(14) = AMAX1(V(14),V(15))
        VHS(15) = VHS(14)
        ULTMI=ULTM*12
        IF(BMHS(5).LT.BMHS(4)) BMHS(5) = BMHS(4)
        IF(BMHS(6).LT.BMHS(5)) BMHS(6) = BMHS(5)
        IF(BMHS(6).LT.ULTMI) BMHS(6)=ULTMI
        DO 21 L = 4,8
21      BMHS(L) = AMAX1(BMHS(L),BMHS(12-L))
        RETURN
        END
        SUBROUTINE LANELD(TOTTLD)
      C
[2]     CHARACTER*80 WORDS
[2]     CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]     CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]    *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]     CHARACTER*1 SMBOLB,SMBOL2
        COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
       *DNCDL2,DNCDL1
        COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
       *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
       *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BF,AV
       *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]     COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
        COMMON/J/ BMHS(20),BMSP(20),BMLL(20),VHS(20),VSP(20),VLL(20)
        DIMENSION V(20),BM(20)
        DATA BM,V/40*0.0/
      C
      C
      C CALCULATE CONCENTRATED AND UNIFORMLY DISTRIBUTED LOADS
```

```
C      CALCULATE SHEARS AND MOMENTS  INSPECTION POINTS
C      SELECT MAXIMUM SHEAR AND MOMENT COMBINATION
C
C
       CONCLV = 0.65 * TOTTLD
       CONCLM = 0.45 * TOTTLD
       W = 0.016 * TOTTLD
       DO 1 LD = 1,15
       DIST = XDIST(LD)
       REACTV = 0.5*W*SPANL+ (1.0-DIST/SPANL) *CONCLV
       REACTM = 0.5*W*SPANL+ (1.0-DIST/SPANL) *CONCLM
       V(LD) =  REACTV - W*DIST
       BM(LD) = DIST * (REACTM-W*DIST*0.5 )*12.
       BMLL(LD) = BM(LD)
       VLL(LD) = V(LD)
1      CONTINUE
       DO 2 LD = 1,11
       VLL(LD) = AMAX1(V(LD),V(12-LD))
       BMLL(LD) = AMAX1(BM(LD),BM(12-LD))
2      CONTINUE
       RETURN
       END
       SUBROUTINE MACKS(BMMAX,VMAX,M,V,BMLL,REACTN,MBM,MVM)
C
C
C      DETERMINE MAXIMUM SHEARS AND BENDING MOMENTS FOR RR LOADING
C
C
       BM = ABS(BMLL)
       VEE = ABS(V)
       IF(BMMAX.GT.BM) GO TO 1
       BMMAX = BM
1      IF(VMAX.GT.VEE) GO TO 2
       VMAX = VEE
2      CONTINUE
       RETURN
       END
       SUBROUTINE MILLER
C
[2]    CHARACTER*80 WORDS
[2]    CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]    CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]   *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]    CHARACTER*1 SMBOLB,SMBOL2
       REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
       COMMON/HLF/ X1,X2,Y1,Y2,Y12
       COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
      *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
      *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
      *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]    COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
       COMMON/MM/ROW(30),NROW,SROW(18),IW,DROW(18),NSROW(30)
       COMMON/LI/AR(11),YB1(11),YT1(11),D1(11),IB1(11),WTF1(11),
      *BPRIME(11),HH(11),GG(11),DIAGD(11),DIAGW(11)
```

```
      COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
      COMMON/FYB/KGRID,NSTRNS,ENDECC,IWCH
C
C
C     CALCULATE PLACEMENT OF STRANDS
C
C
      IF(IVIEW.NE.0) GO TO 100
      THETA = ATAN(Y2/X2)
      CL = X1/2.0
      X3 = STSCL / SIN(THETA)
      X4 = Y1 / TAN(THETA)
      X5 = X4 - X3 + STSCL
      Y3 = X5 * TAN(THETA)
      Y4 = X2 * TAN(THETA)
      Y34 = Y3 + Y4
      HSPACE = SPACE/2.0
      FL = X1 - 2.0* X2
      FLCL = FL/2.0
      IF(STRNS.LE.0.0) GO TO 999
      DIST = CL + X4 - X3 - X5
      NROW = 0
      NSTRNS = STRNS/2.0
      IF(IW.EQ.2) GO TO 11
      A = DIST
      NS = STRNS
      K=1
      NS=NS-1
21    A=A-SPACE
      IF(A.LT.-0.001.OR.NS.LT.2) GO TO 22
      K=K+2
      NS = NS - 2
      GO TO 21
22    IF(A.LT.-0.001) GO TO 23
      K=K+NS
      NS=0
23    NROW=NROW+1
      ROW(NROW)=K
      IF(NS.EQ.0)GO TO 99
      H=STBCL
24    H=H+SPACE
      IF(NS.LE.0) GO TO 99
      IF(H.GT.Y3) GO TO 25
      IF(NS.LT.K) GO TO 33
      NROW=NROW+1
      ROW(NROW)=K
      NS=NS-K
      GO TO 24
25    IF(NS.LE.0) GO TO 99
      IF(H.GT.Y34) GO TO 29
      IF(H.GE.Y1) GO TO 50
      CL2=CL-(H-Y3)/TAN(THETA)
      GO TO 51
50    CL2=CL-(H-Y1)/TAN(THETA)
```

```
51    DIST=CL2-STBCL/SIN(THETA)
      A = DIST
      K=1
      NS = NS - 1
26    A=A-SPACE
      IF(A.LT.-0.001.OR.NS.LT.2) GO TO 27
      K=K+2
      NS=NS-2
      GO TO 26
27    IF(A.LT.-0.001) GO TO 28
      IF(NS.EQ.1) GO TO 28
      K=K+NS
      NS=0
28    NROW=NROW+1
      ROW(NROW)=K
      IF(NS.LE.0) GO TO 99
      GO TO 24
29    IF(IW.EQ.1) FLCL = 0.
      A=FLCL
      K=1
30    A=A-SPACE
      IF(A.LT.2.0.OR.NS.LT.2) GO TO 31
      NS=NS-2
      K=K+1
      GO TO 30
31    IF(A.LT.2.0) GO TO 32
      K=K+NS
      NS=0
32    NROW=NROW+1
      IF(NROW*SPACE+STBCL.GT.YB) GO TO 998
      ROW(NROW)=K
      IF(NS.LE.0) GO TO 99
      IF(NS.LT.K) GO TO 29
      NS=NS-K
      GO TO 32
33    IF(NS.EQ.4) GO TO 34
      IF(NS.EQ.6) GO TO 34
      IF(NS.EQ.8) GO TO 34
      IF(NS.EQ.10) GO TO 34
      NROW = NROW + 1
      ROW(NROW) = NS
      GO TO 99
34    K = NS - 1
      NROW = NROW + 1
      ROW(NROW) = K
      NS = NS - K
      IF(NROW*SPACE+STBCL.GT.YB) GO TO 998
      GO TO 24
11    A = DIST + HSPACE
      K = 0
      H = STBCL
1     A = A-SPACE
      IF(A.LT.-0.001.OR.NSTRNS.LE.0) GO TO 2
      K = K+1
```

```
      NSTRNS = NSTRNS - 1
      GO TO 1
2     LPW = 2 * K
      NROW = NROW + 1
      ROW(NROW) = LPW
      IF(ROW(NROW).LE.0.0) NROW = NROW - 1
      IF(NSTRNS.LE.0) GO TO 99
3     H = H + SPACE
      IF(H.GT.Y3) GO TO 4
      IF(NSTRNS.LT.K) GO TO 11
      NROW = NROW + 1
      ROW(NROW) = LPW
      NSTRNS = NSTRNS -K
      GO TO 3
4     CONTINUE
      IF(NSTRNS.LE.0) GO TO 99
      IF(H.GT.Y34) GO TO 7
      IF(H.GE.Y1) GO TO 60
      CL2=CL-STSCL-(H-Y3)/TAN(THETA)
      GO TO 61
60    CL2=CL-(H-Y1)/TAN(THETA)
61    DIST=CL2-STSCL/SIN(THETA)
      A = DIST + HSPACE
      K = 0
5     A = A-SPACE
      IF(A.LT.-0.001.OR.NSTRNS.LE.0) GO TO 6
      K = K+1
      NSTRNS = NSTRNS - 1
      GO TO 5
6     LPW = 2 * K
      NROW = NROW + 1
      ROW(NROW) = LPW
      IF(NSTRNS.LE.0) GO TO 99
      GO TO 3
7     A = FLCL+HSPACE
      K = 0
8     A = A-SPACE
      IF(A.LT.2.0.OR.NSTRNS.LE.0) GO TO 9
      NSTRNS = NSTRNS - 1
      K = K + 1
      GO TO 8
9     LPW = 2 * K
      NROW = NROW + 1
12    CONTINUE
      ROW(NROW) = LPW
      IF(NSTRNS.LE.0) GO TO 99
      IF(NSTRNS.LT.K) GO TO 7
      NSTRNS = NSTRNS - K
      NROW = NROW + 1
      IF(NROW*SPACE+STSCL.GT.YB) GO TO 998
      GO TO 12
99    CONTINUE
102   SUMSTR = 0
      SUMMST = 0
```

```
      SUMSL = 0
      IF(IVIEW.NE.0) NROW=18
      DO 13 JR = 1,NROW
      SUMMST = ROW(JR) * (STBCL + ((JR - 1) *SPACE)) + SUMMST
13    SUMSTR = ROW(JR) + SUMSTR
      ECAL= SUMMST / SUMSTR
      IF(IVIEW.NE.0) STRNS = SUMSTR
999   CONTINUE
      RETURN
  100 NROW=0
      NSTRNS = 0
      DO 101 I=1,18
      NROW=NROW+1
      NSTRNS=DROW(I)+NSTRNS
  101 ROW(I)=SROW(I)+DROW(I)
      GO TO 102
998   ISTOP = 1
      RETURN
      END
      SUBROUTINE MOMENT
C
[2]   CHARACTER*80 WORDS
[2]   CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]   CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]  *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]   CHARACTER*1 SMBOLB,SMBOL2
      COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
     *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
     *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
     *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]   COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
     *0),BMSL(20),BMCDL(20)
      COMMON/ILL/ REQULT,ULTMOM,FPC,FPCI,NSTATE,MSTATE,K
      COMMON/LI/AR(11),YB1(11),YT1(11),D1(11),IB1(11),WTF1(11),
     *BPRIME(11),HH(11),GG(11),DIAGD(11),DIAGW(11)
      COMMON/HD/ H,G
C
C
C     CALCULATE REQUIRED ULTIMATE AND RESISTING MOMENT CAPACITY
C
C
      EFD = D + TS - ECAL
      AST = STRNS * ASTRN
      P =  AST / (EFD * EFW)
      IF(FPS.EQ.0.OR.FPS.EQ.270000.) GO TO 5
      FSU = FPS * (1.0 - 0.5 * P * FPS/SFPC)
      GO TO 6
    5 FSU=240000
    6 FLCK=(AST*FSU)/(0.85*SFPC*EFW)
      RCK = P * FSU / SFPC
      IF (RROAD .NE. 0.0) GO TO 10
      REQULT=1.5*(BMDL(6)+BMNCDL(6)+BMCDL(6))+2.5*BMMA(6)
      GO TO 20
```

```
10    XM = 2.0 - SPANL * 0.004
      IF (SPANL .GT. 100.) XM = 1.6
      REQULT = XM * (BMDL(6) + BMNCDL (6) + BMCDL(6)) + 2.3 * BMMA(6)
20    CONTINUE
      IF (FLCK .GT. TS) GO TO 40
      IF (RCK .GT. 0.3) GO TO 30
C
C     BEGIN CALCULATIONS FOR RECTANGULAR SECTIONS
C
      ULTMOM=AST*FSU*EFD*(1.-0.59*P*FSU/SFPC)
      NSTATE = 3
      GO TO 500
30    ULTMOM = 0.25 * SFPC * EFW * EFD * EFD
      NSTATE = 4
      GO TO 500
C
C     BEGIN CALCULATIONS FOR FLANGED SECTION----MAXIMUM DEPTH OF
C     COMPRESSION BLOCK(YCMX) IS BASED ON MAXIMUM REINFORCEMENT
C     INDEX OF 0.3
C
40    CONTINUE
      IF (BTYPE .EQ. BEAM(5) .OR. BTYPE .EQ. BEAM(6)) GO TO 200
      YCMX = (0.354 * EFD) - TS
      CS = 0.85 * SFPC * EFW * TS
      YCS = EFD - TS/2.
      Z = 0.5 * (WTF - BP)
      THETA = ATAN (Z/G)
      ABC = ((AST * FSU) - CS)/ (0.85 * FPC)
      ABC1MX = WTF * H
      ABC2MX = (WTF * G) - (Z * G)
      IF (ABC .GT. ABC1MX) GO TO 50
      YC1 = ABC / WTF
      ABC1 = WTF * YC1
      NSTATE = 1
      IF (YCMX .GE. YC1) GO TO 100
      YC1 = YCMX
      ABC1 = WTF * YC1
      NSTATE = 2
      GO TO 100
50    IF (ABC .GT. (ABC1MX + ABC2MX)) GO TO 60
      RQABC2 = ABC - ABC1MX
      YC2 = 0.0
      ABC2 = 0.0
      XV = 0.5 * TAN(THETA)
      WTF2 = WTF
60    CONTINUE
      ABC2 = ABC2 + ((0.5 * WTF2) - (0.5 * XV))
      IF (ABC2 .GT. RQABC2) GO TO 70
      YC2 = YC2 + 0.5
      WTF2 = WTF2 - 2. * XV
      GO TO 60
70    ABC2 = ABC2 - ((0.5 * WTF2) - (0.5 * XV))
      ABC1 = ABC1MX
      NSTATE = 1
```

```
      IF(YCMX.GE.(H+YC2)) GO TO 110
      IF (YCMX .GT. H .AND. YCMX .LT. (H + YC2)) GO TO 71
      YC2 = 0.0
      ABC2 = 0.0
      YC1 = YCMX
      ABC1 = YC1 * WTF
      NSTATE = 2
      GO TO 100
71    YC2 = YCMX - H
      ABC2 = (WTF2 * YC2) - (YC2 * TAN(THETA) * YC2)
      ABC1 = ABC1MX
      NSTATE = 2
      GO TO 110
80    RQABC3 = ABC - (ABC1MX + ABC2MX)
      YC3 = RQABC3 / BP
      ABC3 = YC3 * BP
      ABC2 = ABC2MX
      NSTATE = 1
      IF (YCMX .GE. (H + G + YC3)) GO TO 120
      IF (YCMX .GT. H .AND. YCMX .LE. (H+G)) GO TO 85
      YC1 = YCMX
      ABC1 = YC1 * WTF
      ABC2 = 0.0
      ABC3 = 0.0
      NSTATE = 2
      GO TO 100
85    YC2 = YCMX - H
      ABC2 = (WTF * YC2) - YC2 * TAN(THETA)* YC2
      ABC1 = ABC1MX
      ABC3 = 0.0
      NSTATE = 2
      GO TO 110
C
C     CALCULATE MOMENT ARM BETWEEN TENSION STEEL AND COMPRESSION CONC.
C
100   YCB = EFD - (TS + YC1/2.)
      CB = ABC1 * 0.85 * FPC
      GO TO 150
110   AM1 = H * WTF * H/2.
      AM2 = ((WTF-2.*YC2*TAN(THETA)) * YC2) * (H+YC2/2.)
      AM3 = ((YC2 * TAN(THETA) * YC2) / 2. * (H + YC2 / 3.)) * 2.
      YCG = (AM1 + AM2 + AM3) / (ABC1 + ABC2)
      YCB = EFD - TS - YCG
      CB = (ABC1 + ABC2) * 0.85 * FPC
      GO TO 150
120   AM1 = H * WTF * H/2.
      AM2 = ((WTF - 2. * G * TAN(HETA)) * G) * (H + G/2.)
      AM3 =((G * TAN(THETA) * G) /2. * (H + G/3.)) * 2.
      AM4 = (YC3 * BP) * (H + G + YC3/2.)
      YCG = (AM1 + AM2 + AM3 + AM4) / (ABC1 + ABC2 + ABC3)
      YCB = EFD - TS - YCG
      CB = (ABC1 + ABC2 + ABC3) * 0.85 * FPC
150   ULTMOM = CS * YCS + CB * YCB
      GO TO 500
```

```
200   CONTINUE
      H1 = H
      G1 = G
      H = 5.
      G = 3.
      AM1MX = (WTF * H) * H/2.
      AM2MX = (WTF - 26.) * G * (H + G/2.) + (13. * G) * (H + G/3.)
      AM3MX = (BP * 4.) * (H + G + 2.) + 16. * (H + G + 4./3.)
      YCMX = (0.354 * EFD) - TS
      YCS = EFD - TS/2.
      CS = 0.85 * SFPC * EFW * TS
      PHI = ATAN (13./G)
      RHO = ATAN (1.)
      ABC = ((AST*FSU) - CS) / (0.85*FPC)
      ABC1MX = WTF * H
      ABC2MX = (WTF * G) - (13. * G)
      ABC3MX = (BP + 8.) * 4. - (4. * 4.)
      IF (ABC .GT. ABC1MX) GO TO 210
      YC1 = ABC / WTF
      ABC1 = WTF * YC1
      NSTATE = 1
      IF (YCMX .GE. YC1) GO TO 300
      YC1 = YCMX
      ABC1 = WTF * YC1
      NSTATE = 2
      GO TO 300
210   IF (ABC .GT. (ABC1MX + ABC2MX)) GO TO 240
      RQABC2 = ABC - ABC1MX
      YC2 = 0.0
      ABC2 = 0.0
      XV = 0.5 * TAN(PHI)
      WTF2 = WTF
220   CONTINUE
      ABC2 = ABC2 + ((0.5 * WTF2) - (0.5 * XV))
      IF (ABC2 .GT. RQABC2) GO TO 230
      YC2 = YC2 + 0.5
      WTF2 = WTF2 - 2. * XV
      GO TO 220
230   ABC2 = ABC2 - ((0.5 * WTF2) - (0.5 * XV))
      NSTATE = 1
      IF (YCMX .GE. (YC2 + H))GO TO 310
      IF (YCMX .GT. H .AND. YCMX .LT. (H + G)) GO TO 235
      YC1 = YCMX
      ABC1 = YC1 * WTF
      NSTATE = 2
      GO TO 300
235   YC2 = YCMX - H
      ABC2 = (WTF * YC2) - (YC2 * TAN(PHI) * YC2)
      NSTATE = 2
      GO TO 310
240   IF (ABC .GT. (ABC1MX + ABC2MX + ABC3MX)) GO TO 270
      RQABC3 = ABC - (ABC1MX + ABC2MX)
      YC3 = 0.0
      ABC3 = 0.0
```

```
      XV2 = 0.5 * TAN(RHO)
      WTF3 = WTF - 26.
250   CONTINUE
      ABC3 = ABC3 + ((0.5 * WTF3) - (0.5 * XV2))
      IF (ABC3 .GT. RQABC3) GO TO 260
      YC3 = YC3 + 0.5
      WTF3 = WTF3 - 2. * XV2
      GO TO 250
260   ABC3 = ABC3 - ((0.5 * WTF3) - (0.5 * XV2))
      NSTATE = 1
      IF (YCMX .GE. (H + G + 4.)) GO TO 320
      IF (YCMX .LE. H) GO TO 261
      IF (YCMX .GT. H .AND. YCMX .LE. (H + G)) GO TO 262
      IF (YCMX .GT. (H + G) .AND. YCMX .LE. (H + G + 4.)) GO TO 263
      GO TO 500
261   YC1 = YCMX
      ABC1 = YC1 * WTF
      NSTATE = 2
      GO TO 300
262   YC2 = YCMX - H
      ABC2 = (WTF * YC2) - (YC2 * TAN(PHI) * YC2)
      NSTATE = 2
      GO TO 310
263   YC3 = YCMX - (H + G)
      ABC3 = ((WTF - 26.) * YC3) - (YC3 * TAN(RHO) * YC3)
      NSTATE = 2
      GO TO 320
270   RQABC4 = ABC - (ABC1MX + ABC2MX + ABC3MX)
      YC4 = RQABC4 / BP
      ABC4 = YC4 * BP
      NSTATE = 1
      IF (YCMX .GE. (H + G + 4. + YC4)) GO TO 330
      IF (YCMX .LE. H) GO TO 271
      IF (YCMX .GT. H .AND. YCMX .LE. (H + G)) GO TO 272
      IF (YCMX .GT. (H+G) .AND. YCMX .LE. (H+G+4.)) GO TO 273
      IF (YCMX .GT. (H+G+4.) .AND. YCMX .LE. (H+G+4.+YC4)) GO TO 274
      GO TO 500
271   YC1 = YCMX
      ABC1 = YC1 * WTF
      NSTATE = 2
      GO TO 300
272   YC2 = YCMX - H
      ABC2 = (WTF * YC2) - (YC2 * TAN(PHI) * YC2)
      NSTATE = 2
      GO TO 310
273   YC3 = YCMX - (H + G)
      ABC3 = ((WTF - 26.) * YC3) - (YC3 * TAN(RHO) * YC3)
      NSTATE = 2
      GO TO 320
274   YC4 = YCMX - (H + G + 4.)
      ABC4 = YC4 * BP
      NSTATE = 2
      GO TO 330
300   YCB = EFD - (TS + YC1/2.)
```

```
      CB = ABC1 * 0.85 * FPC
      GO TO 350
 310  AM2 = (WTF - 26.) * YC2/2. * (H + YC2/2.) + (13.*YC2)*(H+YC2/3.)
      YCG = (AM1MX + AM2) / (ABC1MX + ABC2)
      YCB = EFD - (TS + YCG)
      CB = (ABC1MX + ABC2) * 0.85 * FPC
      GO TO 350
 320  AM3 = (BP * YC3) * (H + G + YC3/2.) + (4. * YC3) * YC3/3.
      YCG = (AM1MX + AM2MX + AM3) / (ABC1MX + ABC2MX + ABC3)
      YCB = EFD - (TS + YCG)
      CB = (ABC1MX + ABC2MX + ABC3) * 0.85 * FPC
      GO TO 350
 330  AM4 = (BP * YC4) * (H + G + 4. + YC4/2.)
      YCG = (AM1MX + AM2MX + AM3MX + AM4) / (ABC1MX+ABC2MX+ABC3MX+ABC4)
      YCB = EFD - (TS + YCG)
      CB = (ABC1MX + ABC2MX + ABC3MX + ABC4) * 0.85 * FPC
 350  ULTMOM = CS * YCS + CB * YCB
      G = G1
      H = H1
 500  CONTINUE
      RETURN
      END
      SUBROUTINE OUTPUT
      C
[2]   CHARACTER*80 WORDS
[2]   CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA,COMENT
[2]   CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]  *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]   CHARACTER*1 SMBOLB,SMBOL2
      REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
[15]  C  REAL*8 ADATE
      COMMON/BNS/BNSTD
      COMMON/IBM/ ACI(15),VS(20)
      COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
     *DNCDL2,DNCDL1
[3]   COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
     *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
     *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
     *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
      COMMON/MMM/ FG,HDPT,P,COPE
      COMMON/MM/ROW(30),NROW,SROW(18),IW,DROW(18),NSROW(30)
      COMMON/MSC/ VNCDL(15),VCDL(15)
      COMMON/ILL/ REQULT,ULTMOM,FPC,FPCI,NSTATE,MSTATE,K
      COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
     *0),BMSL(20),BMCDL(20)
      COMMON/JDF/ FTLL(20),FBLL(20),FTSL(20),FBSL(20),FTBM(20),
     *FBBM(20),FTDL(20),FBDL(20),FTNCDL(20),FBNCDL(20),ST(20),SB(20)
     *,FT(20),FB(20),FTI(20),FBI(20),FTIB(20),FBIB(20),FTIBSN(20),
     *FBIBSN(20),FTCDL(20),FBCDL(20)
      COMMON/FYB/KGRID,NSTRNS,ENDECC,IWCH
      COMMON/DEF/ SUMSTR,ECALE,SHIELD,DIST,CMAX
      COMMON/JRR/S(15),SQ
      COMMON/ALL/ FBII,ACOMPR,TTEN,FTP,PLOSS,PPERST,RLOSS,ITT
```

```
      COMMON/AJH/C1,C2,C3,C4,C5,C6,LOLAX
      COMMON/KAP/ W,WCP
      COMMON/TJH/ B,WD,C,E,A
      COMMON/HD/ H,G
      COMMON/CONC/ CNCP(20),CNCD(20),CCP(20),CCD(20),SCNCP(10),SCNCD(10)
      COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
      COMMON FCK
      COMMON / CHEN / VMMS(20),VSPC(20)
      COMMON / CHEN2 / DCDL1,DCDL2
      DIMENSION COMENT(24)
      DIMENSION BMCD(20),BMLL(20),BMTOT(20),VCD(20),VLL(20),VTOT(20)
      DATA COMENT/'UNDE','R RE','INF.',' FLG','D. S','ECT.','OVER',' REI
     *','NF. ','FLGD','. SE','CT. ','UNDE','R RE','INF.',' REC','T. S','
     *ECT.','OVER',' REI','NF. ','RECT','. SE','CT. '/
C
C
C     PRINT OUT DESIGN INFORMATION
C
C
[15] C     CALL DDATE (ADATE)
      NI = (NSTATE -1) * 6 + 1
      NF = NSTATE * 6
      PULL = C1 * FPS * ASTRN
      NPULL = PULL / 10
      PULL = (NPULL + 1) * 10.
      PULL=(NPULL)*10
C
C     ROUND OFF CONCRETE STRENGTHS TO NEAREST 10 PSI
C
      IF(FPC .LE. 5000.) GO TO 10
      NFPC = FPC / 10
      FPC = (NFPC + 1) * 10.
      FPC=(NFPC)*10
10    IF(FPCI .LE. 4000.) GO TO 20
      NFPCI = FPCI / 10
      FPCI = (NFPCI + 1) * 10.
      FPCI=(NFPCI)*10
20    CONTINUE
C
C     CALCULATE L.L. STRESS IN TOP FIBER OF SLAB AT MID-SPAN
C
      FTCSL = BMMA(6)/ZTSL
C
C     CONVERT UNITS TO KIPS AND FEET FOR OUTPUT BY DEFINING
C     NEW VARIABLES
C
      PWT = UWB*AREA/144000.
      CWT=PWT+(BSPAC*TSS*UWS/ 1000.)+(COPE*WTF*UWS/144000.)
      OHT = D + TS + COPE
      RLOSS = RLOSS * 100.
      PLOSS = PLOSS * 100.
      PNCDL = NCDL/1000.
      PCDL = CDL/1000.
      NFPS = FPS/1000.
```

```
      RQUM = REQULT/12000.
      UM   = ULTMOM/12000.
      DSL1=DEFK1+DNCDL1
      DSL2=DEFK2 + DNCDL2
      DDF1 = DEFL12
      DDF2 = DEFL14
      DCD1=DCDL1
      DCD2=DCDL2
      DO 25 I=1,10
      SCNCP(I) = SCNCP(I)/1000.
25    CONTINUE
      DO 30 I = 1,20
      CNCP(I) = CNCP(I)/1000.
      CCP(I) = CCP(I)/1000.
      BMCD(I) = (BMDL(I)+BMNCDL(I))/12000
      BMLL(I) = BMMA(I)/12000.
      BMTOT(I) = BMSUM(I)/12000.
      VCD(I) = VDL(I)/1000.
      VLL(I) = VMA(I)/1000.
      VTOT(I) = VSUM(I)/1000.
30    CONTINUE
C
45    FORMAT(5X,'NOTE----BEAM DESIGN BASED ON ULTIMATE MOMENT ',
     * 'REQUIREMENTS')
C
50    FORMAT(5X,'NOTE----STRANDS DRAPED TO TOP MOST POSITION MAY ',
     * 'NEED SHIELDING')
C
   52 FORMAT(5X,'NOTE----BEAM DESIGN BASED ON LOW-RELAXATION ',
     2'STRAND.')
C
55    FORMAT(5X,'NOTE----DESIGN DOES NOT SATISFY ULTIMATE MOMENT ',
     * 'CONDITIONS BUT SATISFIES STRESS REQUIREMENTS.')
C
60    FORMAT(////,5X,'NOTE----NO. OF STRANDS REQUIRED IS GREATER ',
     * 'THAN ALLOWED BY PROGRAM (90).',/,5X,'SUMMARY OF MOMENTS ',
     * 'SHEARS, AND STRESSES DUE TO EXTERNAL LOADS WILL FOLLOW')
C
62    FORMAT(  ///,5X,'NOTE----PROBLEM AS STATED REQUIRED PLACEMENT OF
     *STRANDS ABOVE THE NEUTRAL AXIS, THIS RUN ABORTED.')
C
   65 FORMAT(1H1,  'E01-1514-01',  T41,'VIRGINIA DEPARTMENT OF HIGHWAYS
     *AND TRANSPORTATION', T121, 'JOB NO.  ',/,T121,/, T42,
     *'PRESTRESSED CONCRETE BRIDGE GIRDER DESIGN PROGRAM',//)
   68 FORMAT(15X,'DATE',T27,'REQ.',T39,'PROJECT    CHARGE    NO.',
     *T78,  'D   E   S   C   R   I   P   T   I   O   N',
     *   /,  T14,'SUBMITTED',T28,'BY',T34,
     *'CO. ROUTE CITY/CO. SECT. JOB NO. ACT', /, 13X,A80///)
C
70    FORMAT(  /,'    *****  REVIEW ONLY  *****  REVIEW ONLY  *****  REVI
     *EW ONLY  *****  REVIEW ONLY  *****  REVIEW ONLY  *****  REVIEW ONL
     *Y  *****',/)
C
75    FORMAT(  /,5X,'*** INPUT DATA *** ',//,5X,'BEAM TYPE',9X,'=    ',
```

```
     * A2,13X,'UNIT WT. BEAM CONC.     =',F6.0,'    PCF',13X,
     * 'STRAND SIZE ',13X,'=',2X,A4,' IN',/,5X,'SPAN LENGTH',7X,'=',
     * F7.2,' FT',8X,'UNIT WT. SLAB CONC.     =',F6.0,'    PCF',13X,
     * 'STRAND ULT STRENGTH',6X,'=',I6,' K')
C
80    FORMAT(  5X,'BEAM SPACING',6X,'=',F7.2,' FT',8X,'28-DAY ST.(SLAB',
     * ' CONC.) =',F6.0,'    PSI',13X,'NO. OF WEB STRANDS         =',
     * I6,/,5X,'SLAB THICKNESS     =',F7.2,' IN',8X,
     * 'E(BM.CONC.)',12X,'=',F8.2,' E(06)PSI',8X,'GRID SIZE    ',13X,
     * '=',F6.2,' IN')
C
85    FORMAT(5X,'L.L. DIST. FACTOR =',F7.2,11X,
     *'E(SLAB CONC.)              =',F8.2,' E(06)PSI',8X,
     *'STRAND CL. BOTT. BEAM     =',F6.2,' IN',/,5X,
     *'COMP. SLAB WIDTH  =',F7.2,' IN',8X,
     *'E(PRESTRESS STEEL)       =',F8.2,' E(06)PSI',8X,
     *'STRAND CL. SIDE BEAM      =',F6.2,' IN')
C
90    FORMAT(5X,'UNIF. D.L. N-COMP =',F7.3,' KLF',7X,
     *'AASHTO L.L.               =',3X,A4,A1,17X,
     *'MAX COMP BM. CONC(ALLOW) =',F6.0,' PSI',/,5X,
     *'UNIF. D.L. COMP    =',F7.3,' KLF',7X,
     *'RAILROAD L.L.             =','     E-',F3.0,16X,
     *'MAX. TENSION BEAM CONC.  =',F6.0,' PSI')
C
92    FORMAT( //,5X,'*** SECTION PROPERTIES ***',/,
     *32X,'PRECAST',11X,'COMPOSITE',//,
     *5X,'AREA                       =',F10.2,10X,F10.2,' IN2',/,
     *5X,'WEIGHT                     =',F10.2,10X,F10.2,' KLF',/,
     *5X,'MOMENT OF INERTIA          =',F10.2,10X,F10.2,' IN4',/,
     *5X,'YB BEAM                    =',F10.2,10X,F10.2,' IN',/,
     *5X,'SECTION MODULUS BOTTOM =',F10.2,10X,F10.2,' IN3',/,
     *5X,'YT BEAM                    =',F10.2,10X,F10.2,' IN',/,
     *5X,'SECTION MODULUS TOP        =',F10.2,10X,F10.2,' IN3',/,
     *5X,'YTS SLAB                   =',20X,F10.2,' IN',/,
     *5X,'SECTION MODULUS SLAB       =',20X,F10.2,' IN3',/,
     *5X,'HEIGHT                     =',F10.2,10X,F10.2,' IN')
C
95    FORMAT(//,5X,'*** BEAM DIMENSIONS (INCHES) ***',/,5X,'B=',
     * F6.2,2X,'W=',F6.2,2X,'C=',F6.2,2X,'E=',F6.2,2X,'A=',F6.2,2X,
     * 'H=',F6.2,2X,'G=',F6.2)
C
100   FORMAT(//,5X,'*** CONCENTRATED LOADS APPLIED TO NON-COMPOSITE'
     * ,' SECTION ***',/,5X,'LOAD (KIPS)',15X,10(1X,F6.2),/,5X,
     * 'DIST. FROM LT. REACT. (FT)',10(1X,F6.2))
C
105   FORMAT(//,5X,'*** CONCENTRATED STATIC LOADS APPLIED TO ',
     * 'COMPOSITE SECTION ***',/,5X,'LOAD (KIPS)',15X,10(1X,F6.2),/,
     * 5X,'DIST. FROM LT. REACT. (FT)',10(1X,F6.2))
C
110   FORMAT(//,5X,'*** CONCENTRATED LIVE LOADS APPLIED TO ',
     * 'COMPOSITE SECTION ***',/,5X,'LOAD (KIPS)',15X,10(1X,F6.2),/,
     * 5X,'DIST FROM LT. LOAD (FT)',3X,10(1X,F6.2),
     * /,5X,'LOAD (KIPS)',15X,10(1X,F6.2),/,
```

```
     * 5X,'DIST. TO NEXT LOAD  (FT)',2X,10(1X,F6.2))
C
  115 FORMAT(  //,5X,'*** BEAM DESIGN ***')
C
120    FORMAT(/,5X,'TYPE OF BEAM',14X,'=',3X,A2,22X,
     * 'D.L. DEFL AT MID-SPAN =',F9.3,' IN ( NON-COMP )',F12.3,' IN ',
     * '( COMP )',/,5X,
     * 'NO. OF STRANDS',12X,'=',F6.0,21X,
     * 'D.L. DEFL AT 1/4 PT.  =',F9.3,' IN ( NON-COMP )',F12.3,' IN ',
     * '( COMP )',/,5X,
     * 'SIZE OF STRANDS & PULL',4X,'=   ',A4,' ',F8.0)
C
125    FORMAT(5X,'TYPE OF STRANDS',11X,'=',I6,'K',20X,'ULTIMATE ',
     * 'MOMENT REQUIRED =',F6.0,' FT-KIPS',/,5X,'ECCENTRICITY AT C.L.'
     * ,6X,'=',F8.2,' IN',16X,'ULTIMATE MOMENT PROVIDED =',F6.0,
     * ' FT-KIPS  ',6(A4),/,5X,'ECCENTRICITY AT END',7X,'=',
     * F8.2,' IN')
C
  130 FORMAT ( 5X, 'NO. OF DEPRESSED STRANDS  =', I5,
     *       22X,'CRACKING STRESS =',2X,F8.2,' PSI')
  132 FORMAT(' ',5X,'DEPRESSED TOP ',I1,' STRANDS TO POSITION A-',F5.2)
  133 FORMAT(' ',5X,'CONCRETE RELEASE STRENGTH =',F8.2,' PSI')
C
135    FORMAT(5X,'CONCRETE 28-DAY STRENGTH  =',F8.0,'   PSI',13X,
     * 'TOP FIBER DESIGN STRESS (C.L.)     =',F6.0,' PSI',/,5X,
     * 'HOLD DOWN FROM C. L.        =',F8.2,' FT',16X,
     * 'BOTTOM FIBER DESIGN STRESS (C.L.) =',F6.0,' PSI',//,5X,
     * 'DIST. TO TOP DRAPED STRDS =',F8.2,' IN'/,5X,
     * 'SHIELD LENGTH FROM END     =',F8.2,' FT',16X,
     * 'MAXIMUM CAMBER =',F6.2,' IN',/,59X,'PRESTRESS LOSS =',F6.2,
     * ' PERCENT',/,59X,'LOSS AT RELEASE=',F6.2,' PERCENT')
C
136    FORMAT(//,5X,'L.L. STRESS IN TOP FIBER OF SLAB AT MIDSPAN =',
     * F6.0,' PSI',//)
C
140    FORMAT( //,55X,'*** STRAND PATTERN ***',//,9X,'(C.L. OF BEAM)',
     *50X,'(END OF BEAM)')
C
145    FORMAT(5X,'ROW',I3,' HAS',F4.0,' STRANDS',5X,'ROW',I3,' HAS'
     *,F4.0,' STRANDS',' WITH',F4.0,' STRANDS SHIELDED',5X,
     *'ROW',F6.2,' INCHES FROM BOTTOM HAS ',F4.0,'STRANDS')
C
  150 FORMAT(///,28X,'*** MOMENT SUMMARY (FT-KIPS) ***',38X,'*** ',
     *       'SHEAR SUMMARY (KIPS) ***',//,5X,'SECTION',7X,'BEAM',5X,
     *       'SLAB',3X,'NON-COMP',4X,'COMP',3X,'L.L.+I',4X,'TOTAL',16X,
     *       'BEAM & SLAB',3X,'NON-COMP',4X,'COMP',3X,'L.L.+I',3X,
     *       'TOTAL')
C
  155 FORMAT(7X,I2,5X,2(3X,F6.1),1X,4(3X,F6.1),18X,F6.1,7X,F6.1,1X,3
     *       (2X,F6.1))
C
  160 FORMAT(3X,'HOLD-DOWN',2X,2(3X,F6.1),1X,4(3X,F6.1),18X,F6.1,7X,
     *       F6.1,1X,3(2X,F6.1))
C
```

```
165   FORMAT(////,30X,'**** STRESSES IN EXTREME FIBERS DUE TO ',
     * 'EXTERNAL LOADS (LBS PER SQ. IN.) ****',//,72X,'TOTAL D.L.',
     * 14X,'L.L. + IMPACT',5X,'TOTAL',/,5X,
     * 'SECTION',8X,'BEAM',12X,'SLAB',8X,
     * 'NON-COMP SEC. ','NON-COMP SEC.',6X,'COMP SECT.', 7X,
     * 'COMP. SEC.',/,
     * 17X,'TOP',4X,'BOT',6X,'TOP',4X,'BOT',6X,'TOP',4X,
     * 'BOT',6X,'TOP',4X,'BOT',6X,'TOP',4X,'BOT',6X,'TOP',4X,'BOT'
     * ,6X,'TOP',4X,'BOT')
C
170   FORMAT(7X,I2,4X,7(2(1X,F6.0),2X))
C
175   FORMAT(3X,'HOLD-DOWN',1X,7(2(1X,F6.0),2X))
C
180   FORMAT(//,10X,'**** STRESSES DUE TO EXTERNAL LOADS PLUS ',
     * 'PRESTRESS (LBS PER SQ. IN.) ****',//,40X,'BEAM PLUS',/,
     * 38X,'INITIAL PREST.',5X,
     * 'FINAL PREST. PLUS',6X,'ALL LOADS PLUS',/,17X,'INITIAL PREST.',
     * 7X,'- LOSSES  REL.',4X,'TOT. D.L.(N/C SEC.)',6X,'FINAL PREST.',
     * /,8X,4(10X,'TOP',5X,'BOT'))
C
185   FORMAT(7X,I2,4(7X,F6.0,2X,F6.0))
C
190   FORMAT(4X,'HOLD-DOWN',1X,2(2X,F6.0),3(7X,F6.0,2X,F6.0))
C
195   FORMAT(/////,6X,'**** STIRRUP SPACING ****',24X,'**** MAX. ',
     *'ULT. HORIZ. SHEAR (VQ/I) ****',/,9X,20X,
     * 31X,'(BETWEEN SL. AND GIR. FLANGE)',//,5X,'SECTION',42X,
     * 'SECTION',5X,'REG. SLAB',8X,'P.C. PANEL',/)
C
  200 FORMAT(7X,I2,6X,'NO. 4 (GR. 60) AT' ,F5.1,' IN',16X,I2,
     * F12.1,' PSI',F13.1,' PSI')
C
      WRITE (6,65)
[8]   WRITE(6,68)WORDS
      IF(IVIEW.NE.0) WRITE(6,70)
      WRITE(6,75) BTYPE,UWB,DIA,SPANL,UWS,NFPS
       TS = TS + 0.5
      WRITE(6,80) BSPAC,SFPC, IW,TS,EC,SPACE
      WRITE(6,85) DFACT,ECSL,STBCL,EFW,ES,STSCL
      WRITE(6,90) PNCDL,SMBOL1,SMBOL2,ACOMPR,PCDL,RROAD,FTP
      WRITE(6,92) AREA,AREAC,PWT,CWT,IB,INA,YB,YBC,ZBB,ZBBC,YT,YTC,
     *  ZTB,ZTBC,YTCSL,ZTSL,D,OHT
      WRITE(6,95) B,WD,C,E,A,H,G
      WRITE(6,100) (CNCP(I),I=1,10), (CNCD(I), I=1,10)
      WRITE(6,105) (SCNCP(I), I=1,10), (SCNCD(I), I=1,10)
      WRITE(6,110) (CCP(I), I=1,10), (CCD(I), I=2,10),
     *   (CCP(I), I=11,20),(CCD(I), I=11,20)
       TS = TS -0.5
      IF(ISTOP.EQ.1) GO TO 235
      IF(STRNS .GT. 90.) WRITE(6,60)
      WRITE (6,65)
      IF(IVIEW.EQ.1) WRITE(6,70)
      IF(STRNS .GT. 90.) GO TO 240
```

```
      WRITE(6,115)
      IF(MSTATE .EQ. 2) WRITE(6,45)
      IF(IWCH .EQ. 2) WRITE(6,50)
      IF(LOLAX.GE.1) WRITE(6,52)
      IF(K .EQ. 1) WRITE(6,55)
      WRITE(6,120) BTYPE,DSL2,DCD2,STRNS,DSL1,DCD1,DIA,PULL
      WRITE(6,125) NFPS,RQUM,ECCL,UM,(COMENT(I),I=NI,NF),ENDECC
      WRITE(6,130) NSTRNS,FCK
      IF(IVIEW.EQ.0) WRITE(6,132) IW,KGRID
      WRITE(6,133) FPCI
      FPC9=0.8001*FPC
      IF(FPCI.GT.FPC9) WRITE(6,131)
  131 FORMAT(9X,'NOTE *** CONCRETE RELEASE STRENGTH EXCEEDS',
     *' 80% OF COMPRESSIVE STRENGTH. ***')
      WRITE(6,135) FPC,ST(6),HDPT,SB(6),KGRID,SHIELD,CMAX,PLOSS,RLOSS
      WRITE(6,136) FTCSL
      WRITE(6,140)
      DO 210 I = 1,NROW
      XX = KGRID - (I-1)*SPACE
      SS=SROW(I)-NSROW(I)
      IF(SS.EQ.0) GO TO 205
C     NSS = SS
C     IF(NSS.NE.(2*(NSS/2))) SS = SS + 1.
      IF(SHIELD.EQ.0.) SS = 0.
  205 WRITE(6,145) I,ROW(I),I,SROW(I),SS,XX,DROW(I)
210   CONTINUE
      WRITE (6,65)
      IF(IVIEW.EQ.1) WRITE(6,70)
240   WRITE(6,150)
      DO 215 I =1,11
      II = I-1
      ZBM=BMBM(I)/12000.
      ZBL=BMSL(I)/12000.
      ZBNL=BMNCDL(I)/12000.
      ZBCL=BMCDL(I)/12000.
      ZVNL=VNCDL(I)/1000.
      ZVCL=VCDL(I)/1000.
  215 WRITE(6,155) II,ZBM,ZBL,ZBNL,ZBCL,BMLL(I),BMTOT(I),VCD(I),ZVNL,
     *      ZVCL,VLL(I),VTOT(I)
C
      YBM=BMBM(14)/12000.
      YBL=BMSL(14)/12000.
      YBNL=BMNCDL(14)/12000.
      YBCL=BMCDL(14)/12000.
      YVNL=VNCDL(14)/1000.
      YVCL=VCDL(14)/1000.
      WRITE(6,160) YBM,YBL,YBNL,YBCL,BMLL(14),BMTOT(14),VCD(14),YVNL,
     *      YVCL,VLL(14),VTOT(14)
      WRITE(6,165)
      DO 220 I=1,11
      II = I-1
      FTDL(I) = FTDL(I) + FTNCDL(I)
      FBDL(I) = FBDL(I) + FBNCDL(I)
220   WRITE(6,170) II,FTBM(I),FBBM(I),FTSL(I),FBSL(I),FTNCDL(I),
```

```
     *   FBNCDL(I),FTDL(I),FBDL(I),
     *   FTCDL(I), FBCDL(I),FTLL(I),FBLL(I),ST(I),SB(I)
      FTDL(14) = FTDL(14) + FTNCDL(14)
      FBDL(14) = FBDL(14) + FBNCDL(14)
      WRITE(6,175) FTBM(14),FBBM(14),FTSL(14),FBSL(14),FTNCDL(14),
     *   FBNCDL(14),FTDL(14),FBDL(14),
     *   FTCDL(14), FBCDL(14),FTLL(14),FBLL(14),ST(14),SB(14)
      WRITE (6,65)
      IF(IVIEW.EQ.1) WRITE(6,70)
      WRITE(6,180)
      DO 225 I = 1,11
      II = I-1
225   WRITE(6,185) II,FTI(I),FBI(I),FTIB(I),FBIB(I),FTIBSN(I),
     *   FBIBSN(I),FT(I),FB(I)
      WRITE(6,190) FTI(14),FBI(14),FTIB(14),FBIB(14),FTIBSN(14),
     *   FBIBSN(14),FT(14),FB(14)
      WRITE(6,195)
      DO 230 L = 1,11
      LL = L-1
C
C  STIRRUP SPACING PRINT CORRECTION 7-16-81
      L2=L
      IF(L2.GT.6) L2=11-LL
  230 WRITE(6,200) LL,S(L2),LL,VS(L),VSPC(L)
      IF(IVIEW.NE.1) SHIELD = 0.0
      RETURN
235   ISTOP = 0
      WRITE(6,62)
      RETURN
      END
      SUBROUTINE PROPTY
C
[2]         CHARACTER*80 WORDS
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
      REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
      COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
     *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
     *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
     *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]         COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      COMMON/HD/ H,G
      COMMON/TJH/ B,WD,C,E,A
      COMMON/HLF/ X1,X2,Y1,Y2,Y12
      COMMON/JJJ/ BB(11),WDD(11),CC(11),EE(11)
      COMMON/MMM/ FO,HDPT,P,COPE
      COMMON/BNS/ BNSTD
      COMMON/LI/AR(11),YB1(11),YT1(11),D1(11),IB1(11),WTF1(11),
     *BPRIME(11),HH(11),GG(11),DIAGD(11),DIAGW(11)
C
C
C     DETERMINE DESIGN SECTION PROPERTIES
```

```
C
C
      IF(BTYPE.EQ.BNSTD) GO TO 1
      DO 4 IPNT = 1,11
      IF(BTYPE.EQ.BEAM(IPNT)) GO TO 5
4     CONTINUE
5     CONTINUE
C
C     GET PROPERTIES FOR A STANDARD BEAM FROM TABLES
C
      KASE = IPNT
      AREA = AR(IPNT)
      YB = YB1(IPNT)
      IB = IB1(IPNT)
      YT = YT1(IPNT)
      D = D1(IPNT)
      WTF = WTF1(IPNT)
      BP=BPRIME(IPNT)
      X1 = BB(IPNT)
      X2 = (BB(IPNT) - WDD(IPNT))/2.0
      Y1 = CC(IPNT)
      Y2 = EE(IPNT)
      Y12 = Y1 + Y2
      H = HH(IPNT)
      G = GG(IPNT)
      B = BB(IPNT)
      W = WDD(IPNT)
      C = CC(IPNT)
      E = EE(IPNT)
      A = WTF1(IPNT)
      WD = BP
      GO TO 2
1     CONTINUE
C
C     DETERMINE PROPERTIES FOR 'NS' BEAM
C
      X1 = B
      X2 = (B - WD ) / 2.0
      Y1 = C
      Y2 = E
      Y12 = Y1 + Y2
      BP = WD
      WTF = A
C
C     DETERMINE EFFECTIVE FLANGE WIDTH
C
2     IF(EFW.NE.0.0) GO TO 10
      FW1 = SPANL/4.
      FW2 = BSPAC
      FW3 = (12.*TS + WTF ) / 12.
      EFW = FW1
      IF(EFW.GT.FW2) EFW = FW2
      IF(EFW.GT.FW3) EFW = FW3
10    CONTINUE
```

```
      C
      C     COMPUTE COMPOSITE SECTION PROPERTIES
      C
            EFW = EFW * 12.
            ASL = EFW * TS * ECSL / EC
            IBSL = ASL*TS*TS/12.
            YBC = (ASL*(TS/2+D+COPE)+COPE*WTF*(COPE/2+D)+AREA*YB) /
           *(ASL+AREA+COPE*WTF)
            INA = ASL*(TS/2.+D+COPE)**2+COPE*WTF*(COPE/2+D)**2+AREA*YB**2+
           *IBSL+IB+WTF*COPE**3/12-(ASL+COPE*WTF+AREA)*YBC**2
            YTC = D - YBC
            ZBB=IB/YB
            ZTB=IB/YT
            ZTBC = INA / YTC
            ZBBC = INA/YBC
            YTCSL = D + TS +COPE - YBC
             ZTSL = INA / YTCSL * EC / ECSL
            AREAC = AREA+ASL+COPE*WTF
      C
      C     DETERMINE HOLD-DOWN POINTS AND ROUND TO NEAREST 3 INCHES
      C
            IF(HDPT.NE.0.0) GO TO 11
            HDPT = 0.10 * SPANL
            IHDPT = HDPT
            FRHDPT = HDPT - IHDPT
            XFR = FRHDPT / 0.25
            IXFR = XFR
            FRHDPT = IXFR * 0.25
            HDPT = IHDPT + FRHDPT
         11 CONTINUE
            RETURN
            END
            SUBROUTINE PSTRES
      C
[2]         CHARACTER*80 WORDS
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
            REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
            COMMON/ILL/ REQULT,ULTMOM,FPC,FPCI,NSTATE,MSTATE,K
            COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
           *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
           *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB.ZTBC,WTF,BP,AV
           *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]         COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
            COMMON/MMM/ FO,HDPT,P,COPE
            COMMON/ALL/ FBII,ACOMPR,TTEN,FTP,PLOSS,PPERST,RLOSS,ITT
            COMMON/FYB/KGRID,NSTRNS,ENDECC,IWCH
            COMMON/DEF/ SUMSTR.ECALE,SHIELD,DIST.CMAX
            COMMON/MM/ROW(30),NROW,SROW(10),IW,DROW(10),NSROW(30)
            COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
           *0),BMSL(20),BMCDL(20)
            COMMON/JDF/ FTLL(20),FBLL(20).FTSL(20),FBSL(20),FTBM(20),
```

```
     *FBBM(20),FTDL(20),FBDL(20),FTNCDL(20),FBNCDL(20),ST(20),SB(20)
     *,FT(20),FB(20),FTI(20),FBI(20),FTIB(20),FBIB(20),FTIBSN(20),
     *FBIBSN(20),FTCDL(20),FBCDL(20)
      COMMON/AJH/C1,C2,C3,C4,C5,C6,LOLAX
      COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
      COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
     *DNCDL2,DNCDL1
      COMMON FCK
C
C
C     CALCULATE PRESTRESSING REQUIRED
C
C
      K = 0
   88 IF(LOLAX.NE.0) GO TO 4
C
C     COEFFICIENTS FOR STRESS RELIEVED STRANDS
C
      C1 = 0.7
      C2 = 0.85
      C3 = 20000
      C4 = 0.4
      C5 = 0.2
      TC = 10.
      GO TO 5
C
C     COEFICIENTS FOR STABILIZED STRANDS
C
4     C1 = 0.75
      C2 = 0.90
      C3 = 4860
      C4 = 0.1
      C5 = 0.05
      TC = 45.
      IF(K.GT.0) GO TO 300
5     MSTATE = 1
      CALL ALLOW
C
C    CACULATE STRESSESINSPECTION POINTS
C
      DO 47 I=1,15
      FTLL(I)=BMMA(I)/ZTBC
      FBLL(I)=BMMA(I)/ZBBC
      FTSL(I)=BMSL(I)/ZTB
      FBSL(I)=BMSL(I)/ZBB
      FTBM(I)=BMBM(I)/ZTB
      FBBM(I)=BMBM(I)/ZBB
      FTDL(I)=BMDL(I)/ZTB
      FBDL(I)=BMDL(I)/ZBB
      FTNCDL(I)=BMNCDL(I)/ZTB
      FBNCDL(I)=BMNCDL(I)/ZBB
      FTCDL(I) = BMCDL(I) / ZTBC
      FBCDL(I) = BMCDL(I) / ZBBC
      ST(I) = FTDL(I) + FTNCDL(I) + FTLL(I) + FTCDL(I)
```

```
      SB(I) = FBDL(I) + FBNCDL(I) + FBLL(I) + FBCDL(I)
47    CONTINUE
      FCK=.2*(FBDL(6)+FBNCDL(6)+FBCDL(6))+.5*FBLL(6)
      FCK1=FCK-7.5*SQRT(FPC)
      IF (FCK1 .LE. 0) FCK1=0
      IF((FCK-FTP).LE.(7.5* FPC**0.5)) GO TO 8
      STRESB=FBDL(6)+FBNCDL(6)+FBLL(6)+FBCDL(6)+FCK-7.5*FPC**0.5
      GO TO 9
    8 STRESB=FBDL(6)+FBNCDL(6)+FBLL(6)+FBCDL(6)+FTP
C
C     DETERMINE STRESS TO BE OVERCOME BY PRESTRESS
C
    9 TSTRES =STRESB/   0.8
      PLOSX = PLOSS / 100.
      IF(IVIEW.NE.0) GO TO 300
C
C     OBTAIN INITIAL NUMBER OF STRANDS AND ECCENTRICITY OF THE PATTERN
C     AT MIDSPAN
C
      ECC  = YB
      TEMPP = TSTRES/(1./AREA + ECC /ZBB)
      FBP = 0.
401   STRNS = TEMPP/TENIN
      NSTN = STRNS
      NNSTN = NSTN/2
      NMSTN = NNSTN*2
      STRNS = NMSTN
      RSTRNS = 1.2*7.5*FPC**0.5/((1./AREA+ YB/ZBB)*TENIN)
      NSTNR = RSTRNS
      NNSTNR = NSTNR/2 + 1
      NMSTNR = NNSTNR * 2
      RSTRNS = NMSTNR
      IF(RSTRNS.GT.STRNS) STRNS = RSTRNS
      IF(STRNS.GT.90.) GO TO 505
300   CALL MILLER
      IF(ISTOP.EQ.1) GO TO 505
319   ECCL  = YB - ECAL
      W   = UWB * AREA / 144.
      IF(PLOSX.NE.0.0) GO TO 318
C
C     DETERMINE PRESTRESS LOSSES
C
      AST = ASTRN * STRNS
      RN=ES*1000000/(33.*UWB**1.5*FPCI**.5)
      DLM = W * SPANL * SPANL * 1.5
      TEMP= 1. + (RN*AST) * (1/AREA + ECCL * ECCL /IB)
      T1  = 18
      CRST= (ALOG10(T1)/TC)*(C1 *FPS/(C2 *FPS)-0.55)*C1 *FPS
      F18 = (C1 *FPS - CRST + RN*DLM*ECCL / IB) / TEMP
      FCIR= F18*AST*(1/AREA + ECCL * ECCL / IB) - (DLM *ECCL / IB)
      SE=RN*FCIR
       WS=UWS*TSS*BSPAC + NCDL
      MNS = WS * SPANL * SPANL * 1.5
      MCDL=  CDL * SPANL * SPANL * 1.5
```

```
      FCDS= MNS*ECCL/IB +  MCDL *(ECCL+(YBC-YB))/INA
      CRC = 12 * FCIR - 7 * FCDS
      RH=70
      SH  = 17000 - 150 * RH
      CRS = C3  - C4* SE - C5  * (SH + CRC)
      DFS = SH + SE + CRC + CRS
      PLOSS= DFS / (C1* FPS)
      RLOSS = (C1 * FPS - F18)/(C1 * FPS)
      GO TO 321
C
C     DETERMINE PRESTRESSING FORCES, INITIAL AND EFFECTIVE
C
318   PLOSS = PLOSX
      RLOSS = 0.5 * PLOSS
321   IF(LOLAX.EQ.0) GO TO 317
      IF(F18.LE.(0.7 * FPS)) GO TO 317
      C1 = C1 - 0.01
      C3 = C3 - 232.
      C4 = C4 - .052
      C5 = C5 - .0026
      CALL ALLOW
      GO TO 319
317   FO = TENIN * STRNS
      P = FO*(1. - PLOSS)
      FTPR = FO/AREA  -(FO *ECCL )/ZTB
      FBP = FO/AREA +(FO *ECCL )/ZBB
103   TSTRES =STRESB/ ( 1. - PLOSS )
      IF(IVIEW.NE.0) GO TO 700
407   IF(FBP.LT.TSTRES) GO TO 301
      GO TO 304
301   CONTINUE
29    STRNS = STRNS + 2.
      IF(STRNS.GT.90.) GO TO 505
      PLOSS = 0.
      GO TO 300
304   CONTINUE
      FTFOG = FTPR * (1 - RLOSS) + FTBM(6)
      FBFOG = FBP * (1 - RLOSS) - FBBM(6)
      FTFIN=P/AREA-P*ECCL/ZTB+FTDL(6)+FTNCDL(6)+FTLL(6)+FTCDL(6)
      FBFIN=P/AREA+P*ECCL/ZBB-FBDL(6)-FBNCDL(6)-FBLL(6)-FBCDL(6)
      RFPCI = FBFOG/0.6
      IF(RFPCI.LE.FPCI) GO TO 1
      FPCI = RFPCI
      IF(FPC.LT.FPCI) FPC = FPCI
      GO TO 101
1     CONTINUE
      IF(FTFOG.GT.TTEN) GO TO 320
      CALL STRMOD(ECAL)
      GO TO 319
101   CALL ALLOW
      GO TO 103
320   IF(FTFIN.GT.ACOMPR) GO TO 30
      IF(FBFIN.GT.FTP) GO TO 408
30    FPC =(FTFIN/0.4) + 1.
```

```
      IF(FPC.LT.FPCI) FPC = FPCI
      CALL ALLOW
      GO TO 103
408   CONTINUE
700   CALL MOMENT
      IF(IVIEW.NE.0) GO TO 701
      IF(ULTMOM.GT.REQULT) GO TO 470
      K = K+ 1
      IF(K .EQ. 1) GO TO 470
468   CONTINUE
      MSTATE = MSTATE + 1
      IF(MSTATE.GE.2) MSTATE = 2
      GO TO 29
  470 END1=(FO/AREA+TTEN)*ZTB/FO
      END2 = (FBII - FO/AREA)*ZBB/FO
      ENDMAX = END1
      IF(END2.LT.ENDMAX) ENDMAX = END2
701   CALL ECCEND
      CALL ALLOW
      IF (IVIEW.NE.0) GO TO 602
      IF (IWCH.LE.1) GO TO 702
C
C     COMPUTE LENGTH OF SHIELDING
C
      DO 600 I = 1.30
600   NSROW(I) = SROW(I)
      FPCI1 =FO * ENDECC/ZBB  + FO/AREA
      TCK=(FO/AREA-FO*ENDECC/ZTB)
602   NSTRN = 0.
      I = 0
10    I = I+ 1
      SUMSTR = 0.
      SUMMST = 0.
      IF(IVIEW.NE.0) GO TO 601
      NSROW(I) = SROW(I)/2
      IF(NSROW(I).NE.(2*(NSROW(I)/2))) NSROW(I) = NSROW(I) - 1
601   NSTRN  = NSTRN + NSROW(I)
      SUMSTR=NSROW(1)+DROW(1)
      SUMMST=NSROW(1)*STBCL+DROW(1)*(KGRID-(NROW-1)*SPACE)
      DO 472 JR=2,NROW
      SUMSTR=SUMSTR+NSROW(JR)+DROW(JR)
      SUMMST=SUMMST+NSROW(JR)*(STBCL+((JR-1)*SPACE))+DROW(JR)*(KGRID-
     * (JR-1)*SPACE)
  472 CONTINUE
      ECALE = YB - SUMMST / SUMSTR
      IF(IVIEW.NE.0) GO TO 702
      FPCI2 = SUMSTR*TENIN*(1/AREA-ECALE/ZTB)*(1.0-RLOSS)
      IF(FPCI2.LT.TTEN)GO TO 10
      FPCI3 = SUMSTR*TENIN*(1/AREA+ECALE/ZBB)*(1.0-RLOSS)
      FPCI3 = FPCI3 / .6
      IF(FPCI3.GT.FPCI) GO TO 10
      SHIELD = 0.0
C  SHIELD IS ASSUMED EQUAL TO 0 , IF ASSUMPTION IS CHANGED THAN KDIST
C AND SUMME FORMULARS HAVE TO BE CHANGED FOR CHANGE IN METHOD OF
```

```
C EVALUATING STEEL DISTRIBUTION
      CFACT = (FPCI1 * (1. - RLOSS) - FBII) * ZBB / (6. * W)
      IF(CFACT.LE.0.0) GO TO 702
      SHIELD=(SPANL-SQRT(SPANL**2-4.*CFACT))/2.0
702   X = 0.
      DD =0.
C
C     COMPUTE STRESSESINSPECTION POINTS
C
      DIST = SPANL* .5-HDPT
      E = ENDECC
      DO 501 I = 1,14
      FO = TENIN * STRNS
      IF (IVIEW.NE.0) GO TO 603
      IF (IWCH.LE.1) GO TO 703
603   FSHLD = SPANL - SHIELD
      IF(SHIELD.EQ.0.) GO TO 703
      IF(X.GE.SHIELD.AND.X.LE.FSHLD) GO TO 703
      KDIST = (KGRID - ((NROW - 1) * SPACE + STBCL)) *(DIST -DD)/
     * DIST
      KDIST = KDIST +(NROW -1)* SPACE +STBCL
      SUMME = NSROW(1) * STBCL + DROW(1) * (KDIST - (NROW - 1) * SPACE)
      DO 2 JR = 2,NROW
      SUMME = SUMME + NSROW(JR) * (STBCL + (JR - 1) * SPACE) + DROW(JR)
     * *  (KDIST - (NROW - 1) * SPACE + (JR - 1) * SPACE)
 2    CONTINUE
      FO = SUMSTR * TENIN
      E = YB - SUMME / SUMSTR
703   P = FO *(1. - PLOSS)
      IF(I.EQ.12.OR.I.EQ.13) E = ENDECC+(ECCL-ENDECC)*(0.25*SPANL/DIST)
      IF(I.EQ.14) E = ECCL
      FTI(I)=(FO/AREA-FO*E/ZTB)
      FBI(I)=(FO/AREA+FO*E/ZBB)
      FTIB(I)=FTI(I)*(1.-RLOSS)+FTBM(I)
      FBIB(I)=FBI(I)*(1.-RLOSS)-FBBM(I)
      RFPCI = FBIB(I)/.6
      IF(RFPCI.GT.FPCI) FPCI = RFPCI
      FTIBSN(I) = (P/AREA-P*E/ZTB) + FTDL(I) + FTNCDL(I)
      FBIBSN(I) = (P/AREA+P*E/ZBB) - FBDL(I) - FBNCDL(I)
      FT(I)=(P/AREA-P*E/ZTB)+FTDL(I)+FTLL(I)+FTNCDL(I)+FTCDL(I)
      FB(I)=(P/AREA+P*E/ZBB)-FBDL(I)-FBLL(I)-FBNCDL(I)-FBCDL(I)
      X = X + SPANL*0.1
      IF(I.EQ.12.OR.I.EQ.13) X = 0.25*SPANL
      IF(I.EQ.14) X = 0.5*SPANL - HDPT
      DECC = (ECCL-ENDECC)*X/DIST
      IF(X.LE.SHIELD) DD = X
      IF(X.GE.FSHLD) DD = SPANL - X
      E = ENDECC + DECC
      IF(X.GT.DIST) E = ECCL
      IF(X.GT.(SPANL/2. + HDPT)) E = (ECCL - ENDECC ) * (SPANL-X)/(SPANL
     **0.5-HDPT) +ENDECC
501   CONTINUE
      FCK=ABS(FB(6)-FCK)
      CALL CAMBER
```

```
          CALL SHEAR(FPC)
          IF(K .EQ. 1) GO TO 502
          GO TO 505
    502   CALL OUTPUT
          K = K+1
          GO TO 88
    505   CONTINUE
          RETURN
          END
          SUBROUTINE RRLOAD
    C
[2]       CHARACTER*80 WORDS
[2]       CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]       CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]      *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]       CHARACTER*1 SMBOLB,SMBOL2
          COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
         *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
         *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
         *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]       COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
          COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
         *DNCDL2,DNCDL1
          COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
         *0),BMSL(20),BMCDL(20)
          DIMENSION W(18),C(18),POINT(9)
          DATA W/0.5,4*1.0,4*0.65,0.5,4*1.0,4*0.65/
          DATA C/0.0,8.0,13.0,18.0,23.0,32.0,37.0,43.0,48.0,56.0,64.0,69.0,7
         *4.0,79.0,88.0,93.0,99.0,104.0/
          DATA WU,CU/0.1,109.0/
    C
    C
    C     CALCULATE SHEARS AND MOMENTS  TENTH POINTS DUE TO
    C     COOPER'S E-LOADING
    C
    C
          SPAN=SPANL
          TOTLLD = RROAD
          FRACT = 1.0 + (35.0 - (SPANL*SPANL)/500.)/100.
          IF(FRACT.LT.1.20) FRACT = 1.20
          DO 13 LD = 1,15
          BMMAX = 0.0
          VMAX = 0.0
          DIST = XDIST(LD)
          DO 12 M = 1,18
          SUMWC = 0.0
          SUMLD = 0.0
          BM = 0.0
    C
    C     CHECK WHEEL POSITIONS FOR MAX. MOMENT
    C
          IF(C(M).GT.DIST) GO TO 6
          DFSTW = DIST - C(M)
          IC = 1
```

```
      CN = CU
    9 DSTRN = SPAN - DFSTW
      IF(CN.GT.DSTRN) GO TO 1
      DISTWU = DSTRN - CN
      SUMWC =(SPAN-DISTWU/2.0)*WU*DISTWU
      JLM = 18
      SUMLD = DISTWU * WU
      GO TO 4
    1 SUMWC = 0.0
      DO 2 JL = M , 18
      JLM = JL
      IF(C(JL).GE.DSTRN) GO TO 3
    2 CONTINUE
      GO TO 4
    3 JLM = JLM - 1
    4 DO 5 N = IC,JLM
      SUMLD = SUMLD + W(N)
    5 SUMWC = SUMWC + (DFSTW + C(N)- C(IC))*W(N)
      CBAR = SUMWC/SUMLD
      REACTN = (1.0 - CBAR/SPAN) * SUMLD
      GO TO 10
    6 DO 7 JL = 1,M
      JFW = JL
      CM = C(M) - C(JL)
      IF(CM.LT.DIST) GO TO 8
    7 CONTINUE
    8 IC = JFW
      DFSTW = DIST - CM
      CN = CU - C(IC)
      GO TO 9
   10 DO 11 JJ = IC,M
   11 BM = W(JJ)*(C(M)-C(JJ)) + BM
      BMLL = (BM-DIST*REACTN)*TOTLLD*12.
      VSUB = 0.0
      IF(M.EQ.IC) GO TO 15
      MM1 = M - 1
      DO 14 MMM = IC,MM1
   14 VSUB = VSUB + W(MMM)
   15 V = (REACTN - VSUB) * TOTLLD
      REACTN = REACTN * TOTLLD
      CALL MACKS(BMMAX,VMAX,M,V,BMLL,REACTN,MBM,MVM)
   12 CONTINUE
      VMA(LD) = VMAX * 1000. * FRACT * DFACT
      BMMA(LD) = BMMAX * 1000. * FRACT * DFACT
   13 CONTINUE
      RETURN
      END
      SUBROUTINE SHEAR(FPC)
C
[2]   CHARACTER*80 WORDS
[2]   CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]   CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]  *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]   CHARACTER*1 SMBOLB,SMBOL2
```

```
      REAL MU(20),VU(20),MCR(20),MUVU(20)
      REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
      COMMON/FYB/KGRID,NSTRNS,ENDECC,IWCH
[3]   COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
     *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
     *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
     *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
      COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
     *DNCDL2,DNCDL1
      COMMON/MMM/ FO,HDPT,P,COPE
      COMMON/IBM/ ACI(15),VS(20)
      COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
     *0),BMSL(20),BMCDL(20)
      COMMON/JRR/S(15),SQ
      COMMON/MSC/ VNCDL(15),VCDL(15)
      COMMON / CHEN / VMMS(20),VSPC(20)
      COMMON / ALL / FBII,ACOMPR,TTEN,FTP,PLOSS,PPERST,RLOSS,ITT
      DIMENSION VC(15),VCG(15),RJ(15),DD(15),VPR(15),VULT(15)
      DIMENSION E(20),XD(20),FPE(20),FD(20),VP(20),VPU(20),VUVPU(20),PHI
     *VC(20), PHIVPU(20),VCI(20),PHIVCI(20),VCIM(20),VCW(20),PHIVCW(20)
      DATA MUVU,E,XD,FPE,FD,MCR,VP,VPU,PHIVPU,VCI,PHIVCI,VCIM,VCW,
     *PHIVCW,VUVPU,PHIVC,VULT/335*0.0/
      DATA VC,VCG,RJ,DD,VPR,VU/95*0.0/
C
C
C     CALCULATE COMBINED SHEAR DESIGN POINTS
C
C
      AV=.20
      SPANL=12.0*SPANL
      HDPT=12.0*HDPT
      CLL=2.5
      DO 996 I = 1,11
      VU(I)=1.5*(VDL(I)+VNCDL(I)+VCDL(I)+VMMS(I))+2.5*VMA(I)
996   CONTINUE
      Q = EFW*TS*(TS*0.5 + YTC)
      DM=D+TS
      X1=((SPANL/2.0)-HDPT)/(SPANL/10.0)
      X2=((SPANL/2.0)+HDPT)/(SPANL/10.)+1.0
      N1=X1+1
      N2=X2+1
      TTHETA=(ECCL-ENDECC)/(SPANL/2.0-HDPT)
      DD(1) = YT + ENDECC + TS
      DO 88 I=2,11
      IF(I.LE.N1)DD(I)=DD(I-1)+(SPANL/10.)*TTHETA
      IF(I.GT.N1.AND.I.LT.N2) DD(I) = YT+ECCL + TS
      IF(I.GE.N2)DD(I)=DD(I-1)-(SPANL/10.0)*TTHETA
88    CONTINUE
      DO 99 I=1,11
      IF(I.LE.N1) CTHETA=((SPANL/2.0)-HDPT)/SQRT((ECCL-ENDECC)**2+(SPANL
     */2.0-HDPT)**2)
      IF(I.GT.N1.AND.I.LT.N2)CTHETA=1.0
      IF(I.GE.N2) CTHETA=((SPANL/2.0)-HDPT)/SQRT((ECCL-ENDECC)**2+(SPANL
```

```
     */2.0-HDPT)**2)
      VDEC=(VU(1)-2.5*VMA(6))/5.0
      STHETA=SQRT(1-CTHETA**2)
      VPR(I)=FO*(1.-PLOSS)*STHETA
99    CONTINUE
      DO 101 J=1,11
      S(J)=AV*2.0*FPY*DM/(VU(1)-VPR(J)-(J-1)*VDEC)
101   CONTINUE
      DO 244 J=1,11
      IF(AV   .LE.0.11)GO TO 200
      IF (4.*TS .GT. 15.0) GO TO 201
      IF(S(J).GT.4.*TS)S(J)=4.*TS
      GO TO 202
  201 IF (S(J) .GT. 15.0) S(J)=15.0
      GO TO 202
200   IF(S(J).GT.12.) S(J)=12.0
202   CONTINUE
70    AVM=(S(J)*100.*BP)/(2.*FPY)
      IF(AV   .LT.AVM)GO TO 1200
      GO TO 1201
1200  S(J)=S(J)-1.0
      GO TO 70
1201  CONTINUE
244   CONTINUE
      DO 500 I=1,11
      VS(I)=(1.5*(VCDL(I)+VMMS(I))+2.5*VMA(I))*Q/(INA*WTF)
      VSPC(I)=(1.5*(VCDL(I)+VMMS(I))+2.5*VMA(I))*Q/(INA*(WTF-4.))
500   CONTINUE
C
C     CALCULATE SPACING OF THE WEB REINFORCEMENT AT THE
C     QUARTER POINTS
C
      QMU=1.5*BMDL(12)+2.5*BMMA(12)
      VUQ=(1.5*VDL(12)+2.5+VMA(12))-VPR(2)
      IF(X1.GT.2.5)DDQ=DD(1)+(SPANL*.25)*TTHETA
      IF(X1.GT.2.5) CTHETA=((SPANL/2.0)-HDPT)/SQRT((ECCL-ENDECC)**2+(SPA
     *NL/2.0-HDPT)**2)
      IF(X1.LE.2.5)  DDQ = YT +ECCL + TS
      IF(X1.LE.2.5)CTHETA=1.0
      RJQ=QMU/(P*CTHETA*DDQ)
      VCQ = 0.06*FPC*BP*RJQ*DDQ
      VCGQ = 180.0*BP*RJQ*DDQ
      IF(VCQ.GT.VCGQ) VCQ=VCGQ
      SQ=AV*2.0*FPY*KJQ*DDQ/VUQ
      SGQ = 0.75*DM
      IF(SQ.GT.SGQ) SQ=SGQ
      IF(AV.LE.0.11) GO TO 208
      IF (4.*TS .GT. 15.0) GO TO 209
      IF(SQ.GT.4.*TS) SQ=4.*TS
      GO TO 210
  209 IF (SQ .GT. 15.0) SQ=15.0
      GO TO 210
208   IF(SQ.GT.12.0) SQ=12.0
210   CONTINUE
```

```
71    AVMQ = SQ*100.*BP/(2.*FPY)
      IF(AV.LT.AVMQ) GO TO 1202
      GO TO 1203
1202  SQ = SQ-1.0
      GO TO 71
1203  CONTINUE
      X=0.
      XDD = YT + ECCL + TS
      AS = ASTRN * STRNS
      FPCC = P/AREA
      HSPAN = SPANL/2.
C
C     APPLY 1971 ACI LOAD FACTORS  TENTH POINTS
C
      DO 5  I = 1,11
      VULT(I)=1.5*(VDL(I)+VNCDL(I))+2.5*VMA(I)
      MUVU(I) = BMSUM(I)/VSUM(I)
      E(I) = (ECCL-ENDECC)*X/(HSPAN-HDPT)
      IF(X.GT.(HSPAN-HDPT)) E(I) = ECCL-ENDECC
      IF(X.GT.(HSPAN+HDPT)) E(I) = (ECCL-ENDECC)*((SPANL-X)/
     *(HSPAN-HDPT))
      XD(I) = YT + E(I) + ENDECC + TS
      FPE(I) = P/AREA + P * YB * E(I)/IB
      FD(I) = BMBM(I) * YB/IB
      MCR(I) = (IB/YB) * ((7.5 * SQRT(FPC)) + FPE(I) - FD(I))
      VP(I) = P * STHETA
      IF( X .GT. (HSPAN - HDPT)) VP(I) = 0.
      IF(X.GT.(HSPAN+HDPT)) VP(I) = P*STHETA
      VPU(I)=(AS/80.)*FPS*SQRT(XD(I)/BP)
      PHIVPU(I) = 0.85 * VPU(I)
      VCI(I) = 0.6*BP*XD(I)*SQRT(FPC)+MCR(I)/(MUVU(I)-XD(I)/2.)+VDL(I)
      IF(X .EQ. 0.0) VCI(I) = 0.0
      PHIVCI(I) = 0.85 * VCI(I)
      VCIM(I) = 0.85 * 1.7 * BP * XD(I) * SQRT(FPC)
      IF(XD(I).LT.(0.8*D)) XD(I) = 0.8*D
      PHIVCW(I)=VP(I)
      PHIVC(I)=PHIVCW(I)
      ACI(I)=2.*AV*XDD*FPY/(VULT(I)-PHIVC(I))
C     VCW(I)=BP*XD(I)*(3.5*SQRT(FPC)+.3*FPCC)+VP(I)
C     PHIVCW(I)=.85*VCW(I)
C     VUVPU(I)=VULT(I)-PHIVPU(I)
C     IF (PHIVCW(I) - PHIVCI(I)) 6,6,7
C   6 PHIVC(I)=PHIVCW(I)
C     GO TO 8
C   7 PHIVC(I)=PHIVCI(I)
C    8 CONTINUE
C     IF (PHIVC(I)-VCIM(I)) .LE. 0.0) PHIVC(I)=VCIM(I)
C     IF (PHIVC(I) .GT. VUVPU(I)) GO TO 11
C     ACI(I)=2.*AV*.85*XDD*FPY/(VULT(I)-PHIVC(I))
      IF (0.75*D .GT. 15.) GO TO 300
      SMAX = 0.75 * D
      GO TO 301
  300 SMAX=15.
301   IF(ACI(I)   .LT. SMAX) GO TO 14
```

```
      ACI(I)    = SMAX
14    CONTINUE
      GO TO 4
11    CONTINUE
      ACI(I)    = 2.*AV *80.*FPY*XDD  /(AS*FPS*SQRT(XDD  /BP))
      IF (0.75*D .GT. 15.) GO TO 400
      SMAX =0.75 * D
      GO TO 401
  400 SMAX=15.
401   IF(ACI(I)    .LT. SMAX) GO TO 18
      ACI(I)    = SMAX
18    CONTINUE
4     X = X + SPANL/10.
5     CONTINUE
      SPANL=SPANL/12.0
      HDPT=HDPT/12.0
      RETURN
      END
      SUBROUTINE SPCL
C
[2]         CHARACTER*80 WORDS
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
      COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
     *DNCDL2,DNCDL1
      COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
     *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
     *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
     *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]         COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      COMMON/J/ BMHS(20),BMSP(20),BMLL(20),VHS(20),VSP(20),VLL(20)
      DIMENSION V(20)
      DIMENSION CG(3),WB(3),WT(3),HW(3)
      DATA CG,WB,WT,HW/2.0,2*2.8,4.0,2*14.0,48.0,40.0,30.0,24.0,32.0,24.
     *0/
      DATA V/20*0.0/
C
C
C     CALCULATE SHEAR AND MOMENTS  INSPECTION POINTS DUE TO
C    -H-15 OR H-20 LIVE LOADING
C
C
      HSPAN = SPANL * 0.5
      DO 4 LD = 1,15
      DIST = XDIST(LD)
      DLSTW = DIST + WB(LTYPE)
      IF(DLSTW.GT.SPANL) GO TO 1
      CBAR = DIST + CG(LTYPE)
      REACTN = (1.0 - CBAR/SPANL) * WT(LTYPE)
      BMSP(LD)= DIST * REACTN*12000.
      GO TO 3
1     IF(DIST.GT.HSPAN) GO TO 2
```

```
            REACTN = (1.0-DIST/SPANL) * HW(LTYPE)
            BMSP(LD)= REACTN * DIST*12000.
            GO TO 3
      2     IF(LD.GT.11) GO TO 9
            BMSP(LD) = BMSP(12-LD)
            GO TO 3
      9     IF(LD.EQ.13) BMSP(LD) = BMSP(12)
            IF(LD.EQ.15) BMSP(LD) = BMSP(14)
      3     CONTINUE
            DLSTW = DIST + WB(LTYPE)
            IF(DLSTW.GT.SPANL) GO TO 7
            CBAR = DIST + CG(LTYPE)
      6     REACTN = (1.0 - CBAR/SPANL) * WT(LTYPE)
            GO TO 8
      7     REACTN = (1.0 - DIST/SPANL) * HW(LTYPE)
      8     V(LD) = REACTN*1000.
      4     CONTINUE
            DO 5 LD = 1,11
            VSP(LD) = AMAX1(V(LD),ABS(V(12-LD)))
      5     CONTINUE
            VSP(12) = V(12)
            VSP(13) = VSP(12)
            VSP(14) = V(14)
            VSP(15) = VSP(14)
            RETURN
            END
            SUBROUTINE STRMOD(ECAL)
      C
[2]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,WORDS,DIA
[2]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]         CHARACTER*1 SMBOLB,SMBOL2
            COMMON/MM/ROW(30),NROW,SROW(18),IW,DROW(18),NSROW(30)
            COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
      C
      C
      C     CALCULATE ROW LOCATION FOR STRAND PLACEMENT
      C
      C
            ROWNR = ROW(NROW)
            ROW1 = ROW(1)
            IF(ROW(NROW).NE.ROW1) GO TO 1
            ROW(NROW) = ROWNR -2
            NROW = NROW + 1
            ROW(NROW) = 2
            GO TO 6
      1     IF(ROW1.GT.ROW(2).AND.NROW.EQ.2) GO TO 4
            N = NROW
      2     IF(ROW(N).GT.ROWNR) GO TO 3
            N = N - 1
            GO TO 2
      3     ROW(N) = ROW(N) - 2
            NROW = NROW + 1
            ROW(NROW) = 2
```

```
      GO TO 6
4     IF(ROW1-ROW(2).GT.2) GO TO 5
      NROW = 3
      ROW(1) = ROW(1) - 2
      ROW(3) = 2
      GO TO 6
5     ROW(1) = ROW(1) - 2
      ROW(2) = ROW(2) + 2
6     S1 = 0.
      S2 = 0.
      DO 8 JR = 1,NROW
      S2 = ROW(JR) * (STBCL + ((JR-1)*SPACE)) + S2
8     S1 = ROW(JR) + S1
      ECAL = S2/S1
      RETURN
      END
      SUBROUTINE TYPELD
C
[2]   CHARACTER*80 WORDS
[2]   CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[2]   CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]  *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]   CHARACTER*1 SMBOLB,SMBOL2
      COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
     *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
     *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
     *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[3]   COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
      COMMON/CONC/ CNCP(20),CNCD(20),CCP(20),CCD(20),SCNCP(10),SCNCD(10)
      COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
     *0),BMSL(20),BMCDL(20)
      COMMON/J/ BMHS(20),BMSP(20),BMLL(20),VHS(20),VSP(20),VLL(20)
      COMMON/MMM/ FO,HDPT,P,COPE
      COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
     *DNCDL2,DNCDL1
      DIMENSION WEIGHT(5)
      DATA WEIGHT/2*40.0,30.0,0.0,30.0/
C
C
C     CALCULATE AND APPLY LIVE LOAD IMPACT FACTOR
C
C
      FRACT = (50./(SPANL + 125.)) + 1.
      IF(FRACT.GT.1.3) FRACT = 1.3
      DO 5 I = 1,10
      IF(CCP(I).NE.0.0.OR.CCD(I).NE.0.0) GO TO 6
5     CONTINUE
      TOTTLD = WEIGHT(LTYPE)*1000.
      GO TO(1,2,2,3,1),LTYPE
1     CALL JMLOAD(TOTTLD)
      IF(LTYPE.EQ.1) CALL SPCL
      CALL LANELD(TOTTLD)
      DO 100 L = 1,15
      BMMA(L) = AMAX1(BMHS(L),BMLL(L),BMSP(L))*FRACT*DFACT
```

```
      VMA(L) = AMAX1(VHS(L),VLL(L),VSP(L))*FRACT*DFACT
100   CONTINUE
      GO TO 102
2     CALL SPCL
      CALL LANELD(TOTTLD)
      DO 101 L = 1,15
      BMMA(L) = AMAX1(BMSP(L),BMLL(L))*FRACT*DFACT
      VMA(L) = AMAX1(VSP(L),VLL(L))*FRACT*DFACT
101   CONTINUE
      GO TO 102
3     CALL RRLOAD
      GO TO 102
6     CALL CONLD
      DO 7 I = 1,15
      BMMA(I) = BMMA(I)*FRACT*DFACT
7     VMA(I) = VMA(I)*FRACT*DFACT
102   RETURN
      END
      SUBROUTINE ZERO
C
[2]   CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,WORDS,DIA
[2]   CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[2]  *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[2]   CHARACTER*1 SMBOLB,SMBOL2
      COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
     *0),BMSL(20),BMCDL(20)
      COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
     *DNCDL2,DNCDL1
      COMMON/MSC/ VNCDL(15),VCDL(15)
      COMMON/MM/ROW(30),NROW,SROW(18),IW,DROW(18),NSROW(30)
      COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
C
C
C     THIS ROUTINE INITIALIZES ARRAYS
C
C
      DO 2 I = 1,20
      VSUM(I) = 0.0
      VMA(I) = 0.0
      VDL(I) = 0.0
      BMDL(I) = 0.0
      BMMA(I) = 0.0
      BMNCDL(I) = 0.0
      BMCDL(I) = 0.0
      BMSUM(I) = 0.0
2     CONTINUE
      DO 3 I = 1,15
      VNCDL(I) = 0.0
      VCDL(I) = 0.0
3     CONTINUE
      IF(IVIEW.EQ.1) GO TO 6
      DO 4 I = 1,30
      ROW(I) = 0.
4     NSROW(I) = 0.
```

```
            DO 5 I = 1,18
            SROW(I) = 0.
      5     DROW(I) = 0.
      6     RETURN
            END
[6]         SUBROUTINE INITIAL
[6]   C REPLACES BLOCK DATA SUBROUTINE FROM ORIGINAL PROGRAM
[6]         CHARACTER*80 WORDS
[6]         CHARACTER*4 DIA1,CHRCTR,SMBOLA,SMBOL1,DIA
[6]         CHARACTER*2 BLANKA,ABLANK,BLANKB,BBLANK,BLANKC,CBLANK,BLANKD,
[6]        *DBLANK,BLANKE,EBLANK,BLANKF,FBLANK,BTYPEA,BTYPE,BNSTD,BEAM
[6]         CHARACTER*1 SMBOLB,SMBOL2
[6]         REAL IB,IB1,INA,NCDL,MNCDL,IBSL,MNS,MCDL,KDIST,KGRID
[6]         COMMON/ILL/ REQULT,ULTMOM,FPC,FPCI,NSTATE,MSTATE,K
[6]         COMMON/KI/ ASL,IBSL,INA,YTC,YBC,YTCSL,ZTSL,AREAC,ECCL,
[6]        *ENDMAX,TENIN,SPANL,BSPAC,TS,EFW,UWB,UWS,EC,ECSL,ES,ASTRN,
[6]        *FPS,NCDL,ZTB,ZBB,YT,AREA,D,IB,ZBBC,STRNS,ECAL,YB,ZTBC,WTF,BP,AV
[6]        *,FPY,LTYPE,KASE,KODE,RROAD,SFPC,DFACT,CDL,TSS
[6]         COMMON/JJJ/ BB(11),WDD(11),CC(11),EE(11)
[6]         COMMON/BNS/ BNSTD
[6]         COMMON/CONC/ CNCP(20),CNCD(20),CCP(20),CCD(20),SCNCP(10),SCNCD(10)
[6]         COMMON/LI/AR(11),YB1(11),YT1(11),D1(11),IB1(11),WTF1(11),
[6]        *BPRIME(11),HH(11),GG(11),DIAGD(11),DIAGW(11)
[6]         COMMON/J/ BMHS(20),BMSP(20),BMLL(20),VHS(20),VSP(20),VLL(20)
[6]         COMMON/IBM/ ACI(15),VS(20)
[6]         COMMON/JWM/ VMA(20),VDL(20),XDIST(15),DEFK2,DEFL12,DEFK1,DEFL14,
[6]        *DNCDL2,DNCDL1
[6]         COMMON/ELL/WORDS,SMBOL1,SMBOL2,BTYPE,DIA,BEAM(11)
[6]         COMMON/MM/ROW(30),NROW,SROW(18),IW,DROW(18),NSROW(30)
[6]         COMMON/LLI/ BMMA(20),BMDL(20),BMSUM(20),BMBM(20),BMNCDL(20),VSUM(2
[6]        *0),BMSL(20),BMCDL(20)
[6]         COMMON/STD/STBCL,STSCL,SPACE,IVIEW,ISTOP
[6]         COMMON/JDF/ FTLL(20),FBLL(20),FTSL(20),FBSL(20),FTBM(20),
[6]        *FBBM(20),FTDL(20),FBDL(20),FTNCDL(20),FBNCDL(20),ST(20),SB(20)
[6]        *,FT(20),FB(20),FTI(20),FBI(20),FTIB(20),FBIB(20),FTIBSN(20),
[6]        *FBIBSN(20),FTCDL(20),FBCDL(20)
[6]         COMMON/JRR/ S(15),SQ
[6]         COMMON/MSC/ VNCDL(15),VCDL(15)
[6]         COMMON/ALL/ FBII,ACOMPR,TTEN,FTP,PLOSS,PPERST,RLOSS,ITT
[6]         DO 10 IX=1,20
[6]         CNCP(IX)=0.0
[6]         CNCD(IX)=0.0
[6]         CCP(IX)=0.0
[6]         CCD(IX)=0.0
[6]         FTCDL(IX)=0.0
[6]         FBCDL(IX)=0.0
[6]         BMHS(IX)=0.0
[6]         BMSP(IX)=0.0
[6]         BMLL(IX)=0.0
[6]         VHS(IX)=0.0
[6]         VSP(IX)=0.0
[6]         VLL(IX)=0.0
[6]         VS(IX)=0.0
[6]         VMA(IX)=0.0
```

```
[6]          VDL(IX)=0.0
[6]          BMMA(IX)=0.0
 .           BMDL(IX)=0.0
 .           BMSUM(IX)=0.0
 .           BMBM(IX)=0.0
 .           BMNCDL(IX)=0.0
 .           VSUM(IX)=0.0
 .           BMSL(IX)=0.0
 .           FTLL(IX)=0.0
 .           FBLL(IX)=0.0
 .           FTSL(IX)=0.0
 .           FBSL(IX)=0.0
 .           FTBM(IX)=0.0
 .           FBBM(IX)=0.0
 .           FTDL(IX)=0.0
 .           FBDL(IX)=0.0
 .           FTNCDL(IX)=0.0
 .           FBNCDL(IX)=0.0
 .           ST(IX)=0.0
 .           SB(20)=0.0
 .           FT(IX)=0.0
 .           FB(IX)=0.0
 .           FTI(IX)=0.0
 .           FBI(IX)=0.0
 .           FTIB(IX)=0.0
 .           FBIB(IX)=0.0
 .           FTIBSN(IX)=0.0
 .           FBIBSN(IX)=0.0
 .        10 BMCDL(IX)=0.0
 .           DO 15 IC=1,15
 .           S(IC)=0.0
 .           VNCDL(IC)=0.0
 .           ACI(IC)=0.0
 .           XDIST(IC)=0.0
 .        15 VCDL(IC)=0.0
 .           DO 20 ID=1,18
 .           SROW(ID)=0.0
 .        20 DROW(ID)=0.0
 .           DO 25 IE=1,30
 .           ROW(IE)=0.0
 .        25 NSROW(30)=0.0
 .           BNSTD='NS'
 .           IVIEW=0.0
 .           DIA='1/2 '
 .           BEAM(1)=' 1'
 .           BEAM(2)=' 2'
 .           BEAM(3)=' 3'
 .           BEAM(4)=' 4'
 .           BEAM(5)=' 5'
 .           BEAM(6)=' 6'
 .           BEAM(7)=' 7'
 .           BEAM(8)=' 8'
 .           BEAM(9)=' 9'
[6]          BEAM(10)='10'
```

```
[6]        BEAM(11)='11'
[6]        AR(1)=0.0
 .         AR(2)=369.0
 .         AR(3)=559.0
 .         AR(4)=789.0
 .         AR(5)=1013.0
 .         AR(6)=1085.0
 .         DO 30 IF=7,11
 .         YB1(IF)=0.0
 .         YT1(IF)=0.0
 .         D1(IF)=0.0
 .         IB1(IF)=0.0
 .         WTF1(IF)=0.0
 .         BPRIME(IF)=0.0
 .         WDD(IF)=0.0
 .         BB(IF)=0.0
 .         CC(IF)=0.0
 .         EE(IF)=0.0
 .         HH(IF)=0.0
 .         GG(IF)=0.0
 .      30 AR(IF)=0.0
 .         YB1(1)=0.0
 .         YB1(2)=15.83
 .         YB1(3)=20.27
 .         YB1(4)=24.73
 .         YB1(5)=31.96
 .         YB1(6)=36.38
 .         YT1(1)=0.0
 .         YT1(2)=20.17
 .         YT1(3)=24.73
 .         YT1(4)=29.27
 .         YT1(5)=31.04
 .         YT1(6)=35.62
 .         D1(1)=0.0
 .         D1(2)=36.0
 .         D1(3)=45.0
 .         D1(4)=54.0
 .         D1(5)=63.0
 .         D1(6)=72.0
 .         IB1(1)=0.0
 .         IB1(2)=50979.0
 .         IB1(3)=125390.0
 .         IB1(4)=260741.0
 .         IB1(5)=521163.0
 .         IB1(6)=733320.0
 .         WTF1(1)=0.0
 .         WTF1(2)=12.0
 .         WTF1(3)=16.0
 .         WTF1(4)=20.0
 .         WTF1(5)=42.0
 .         WTF1(6)=42.0
 .         BPRIME(1)=0.0
 .         BPRIME(2)=6.0
[6]        BPRIME(3)=6.0
```

```
[6]        BPRIME(4)=8.0
[6]        BPRIME(5)=8.0
 .         BPRIME(6)=8.0
 .         DO 35 IG=1,10
 .         SCNCP(IG)=0.0
 .      35 SCNCD(IG)=0.0
 .         WDD(1)=0.0
 .         WDD(2)=6.0
 .         WDD(3)=7.0
 .         WDD(4)=8.0
 .         WDD(5)=8.0
 .         WDD(6)=8.0
 .         BB(1)=0.0
 .         BB(2)=18.0
 .         BB(3)=22.0
 .         BB(4)=26.0
 .         BB(5)=28.0
 .         BB(6)=28.0
 .         CC(1)=0.0
 .         CC(2)=6.0
 .         CC(3)=7.0
 .         CC(4)=8.0
 .         CC(5)=8.0
 .         CC(6)=8.0
 .         EE(1)=0.0
 .         EE(2)=6.0
 .         EE(3)=7.5
 .         EE(4)=9.0
 .         EE(5)=10.0
 .         EE(6)=10.0
 .         HH(1)=0.0
 .         HH(2)=6.0
 .         HH(3)=7.0
 .         HH(4)=8.0
 .         HH(5)=8.0
 .         HH(6)=8.0
 .         GG(1)=0.0
 .         GG(2)=3.0
 .         GG(3)=4.5
 .         GG(4)=6.0
 .         GG(5)=4.0
 .         GG(6)=4.0
 .         DO 40 IH=1,11
 .         DIAGD(IH)=0.0
 .      40 DIAGW(IH)=0.0
 .         ASTRN=0.153
 .         SPACE=2.0
 .         STBCL=2.0
 .         STSCL=2.0
 .         UWB=150.
 .         UWS=150.
 .         SFPC=4500.
 .         ES=29.0
[6]        FPS=270000.0
```

```
[6]          FPY=60000.0
[6]          FPC=5000.
 .           FPCI=4000.
 .           PLOSS=0.0
 .           EC=4.29
 .           ECSL=4.07
 .           RETURN
[6]          END
```

A.5.5 Guide for Using Microcomputer Version of Prestressed Beam Design and Analysis Program - Version 1.0

The following is a guide for using the microcomputer version of the Virginia Department of Highways and Transportation Prestressed Beam Design and Analysis Program.

The form of the data to be input into the program is identical to the format currently used on existing data input forms. These forms can be filled out as desired and an input file for the program created from them, with all data in the appropriate lines and columns. Input files are currently created using the IBM Personal Editor. See the section "Creating Program Input Files" for information on how to create these files.

PROCEDURE FOR RUNNING PRESTRESSED BEAM DESIGN AND ANALYSIS PROGRAM

This procedure assumes that the program will be on drive A and the input files and output files will be on drive B. Drive A will be the default drive. (See note 1 below.) All user responses are indicated with bold type.

1. With the program in drive A [A> prompt], type PBEAM and hit the RETURN key. After a few seconds the following header will appear.

VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION
BRIDGE DIVISION
PRESTRESSED BEAM DESIGN AND ANALYSIS PROGRAM

NOTE:
ENTER INPUT FILE NAME FOR UNIT 5 PROMPT
ENTER OUTPUT FILE NAE FOR UNIT 6 PROMPT

File name missing or blank - Please enter name

UNIT 5?

2. In response to the unit 5 prompt type B:infile and hit RETURN where infile is any legal MS-DOS file name used to designate the input file created on drive B:. After a few seconds the following prompt will appear:

UNIT 6:

3. In response to this prompt, type B:resfile and hit RETURN. Resfile is any legal MS-DOS file name you wish to call the output file. This will cause the output from the program to be written to the file resfile, which will reside on drive B:. (Alternatively, to direct all output to the printer, type lpt1 and RETURN in response to the unit 6? prompt.)

4. If all goes well, the program will run for a minute or two and display the following message when execution is completed: Stop - Program terminated.

   At this point the program results are ready for inspection.

5. If the program generates an error message the most probable cause will be an error in the input file. Inspect the input file to make sure all data are in proper position and rerun the program, repeating steps 1 thru 4.

Note 1:
The disk drive on which the program and input/output files reside is purely a matter of convenience. All three files can reside on one drive if desired. However, if the size of the problem is large, it may be advantageous to direct the output file to the other drive, since this file can become very large.

A.6 Steel Girder Design and Analysis Program

This section will describe in detail the procedure used to compile and link the Steel Girder Design and Analysis Program. After the procedure is described, directions will be given illustrating how to create data input files and run the program on the microcomputer.

A.6.1 Program Description

The purpose of this program is to design or analyze a simply supported, composite welded plate girder. The input is arranged to allow for a complete design, a complete analysis, or a combination of the two for a single girder entry. The input also allows for several different trial web depths in a design for a single girder entry. The program uses 1973 AASHTO specifications as interpreted and modified by the Virginia Department of Highways and Transportation's Bridge Division, along with current bridge office practices.

For complete details of all program features, including data input arrangement, see reference 15.

A.6.2 Comments on Compiling and Linking

Unlike the Prestressed Concrete I-Beam Program, this program could not be broken down into a set of smaller source files, because it contained no subroutines. The program was small enough (approx. 900 lines) that it could be compiled as a single source file. Also, the program required very few changes to enable it to run on the microcomputer.

Additionally, the comments made in section A.5.2 also hold true for this program.

A.6.3 Compile and Link Procedure

The compile and link procedure for the Steel Girder Design and Analysis Program is somewhat simpler since only one source file was involved. The method is straightforward, and required three diskettes in the compile and link process. The three diskettes and their contents are as follows:

DISK 1	DISK 2	DISK 3
FOR1.EXE	FORTRAN.LIB	STEEL.FOR
PAS2.EXE	MATH.LIB	STEEL.OBJ
PE.EXE (IBM Personal	LINK.EXE	STEEL.EXE
PE.HLP Editor)		
PE.PRO		

As for the Prestressed Beam Program, disk one contains compiler passes one and two and the page editor. Disk two contains the FORTRAN runtime libraries and the linker. Disk three contains the Steel Girder Program FORTRAN source file and the relocatable object file and executable run file created during the compile and link process.

The steps in the compile and link procedure are as follows:

COMPILING

1. Boot the operating system.

2. Log onto drive B.

3. Place DISK 1 in drive A and DISK 3 in drive B.

4. Invoke pass one of the compiler by typing A:FOR1 and hitting RETURN. The following prompts will appear on the screen:

   ```
   Source file [.FOR]:
   Object file [.OBJ]:
   Source listing [NUL.LST]:
   Object listing [NUL.COD]"
   ```

5. In response to the Source file prompt type STEEL and hit RETURN. (The .FOR extension is automatically added)

6. In response to the Object file prompt hit RETURN. This will cause the object file to be automatically named STEEL.OBJ.

7. If a source listing is desired (it is optional) enter any valid MS-DOS file name in response to the Source listing prompt and hit RETURN. Otherwise, just hit the RETURN key.

8. If an object listing is desired (it is optional) enter any valid MS-DOS file name in response to the Object listing prompt and hit RETURN. Otherwise, just hit the RETURN key. The compiler will begin Pass one after the last prompt is responded to.

9. After Pass one is complete, invoke Pass two of the compiler by typing A:PAS2 and hitting RETURN. When the disk drives stop moving (it will take several minutes) check the contents of DISK 3 in drive B. It should contain the files STEEL.FOR and STEEL.OBJ. This completes the compiling process.

LINKING

1. Replace DISK 1 in drive A with DISK 2. Leave DISK 3 in drive B and remain logged onto drive B.

2. Invoke the linker by typing A:LINK and hitting RETURN. The following prompts will appear:

   Object Modules [.OBJ]:
   Run file ([.EXE]:
   List map [NUL.MAP]:
   Libraries [.LIB]:

3. In response to the Object modules prompt type STEEL and hit RETURN.

4. In response to the Run file prompt type STEEL and hit RETURN. (The .EXE extension will be automatically added creating the file STEEL.EXE)

5. In response to the List map prompt hit RETURN.

6. In response to the Libraries prompt type A: and hit RETURN. After this last prompt is responded to the linker will begin processing.

7. When the linker stops, check DISK 3 in drive B to vertify that the run file STEEL.EXE has been created.

This completes the compile-link procedure for the Steel Girder Design and Analysis Program.

A.6.4 Steel Girder Program Source Listing

This section contains the FORTRAN source listing for the microcomputer converted version of the Steel Girder Design and Analysis Program. The changes that have been made are indicated in bold type. These changes are listed in Table A.2 and are cross-referenced using the numbers to the left of each affected line.

LISTING

```
[1]           PROGRAM MAIN
              DIMENSION RI(23) , W(6,23) , X(20) , XLM(20) ,  DM(20) , XD1M(20),
             1  XLMS(20) , TFT(6) , BFT(6), TFW(6) , BFW(6) , XLLM(6) , XLLS(6),
             2  DLM(20) , MARK(6) , MT(6) , DLS(20) ,  DS(20) , XD1S(20) , G(6),
             3  XSS(20) , XLSM(6) ,XMFT(6), TFW2(4), TFT2(4), BFT2(4), BFW2(4),
             4  XT(4), XB(4), ZSB(4), ZST(4), DSPA(6), DEF1(6), DEF2(6), DEF3(6)
             5   ,TAB(33),XT1(4),XB1(4)
              DIMENSION SLS(20),SLM(20)
              REAL A(6),VV(6),FFV1(6),FFV2(6),C(6),TTEM1(6),TTEM2(6)
              INTEGER IPAT(6)
              DATA TAB/0.75,0.8125,0.875,0.9375,1.0,1.0625,1.125,1.1875.
             1   1.25,1.3125,1.375,1.4375,       1.5,1.5625,1.625,1.6875.1.75,
             2   1.8125,1.875,1.9375,2.0,2.125,2.25,2.375,2.5,2.625,2.75,2.875,
             3   3.0,3.25,3.5,3.75,4.0/
[15]  C       CALL DDATE (IDATE)
[5]           WRITE(*,1)
[5]         1 FORMAT(15X,'VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION'//
[5]          *32X,'BRIDGE DIVISION'//24X,'PLATE GIRDER DESIGN AND ANALYSIS'/////
[5]          *1X,'NOTE: '/1X,'ENTER INPUT FILE NAME FOR UNIT 1 PROMPT'/
[5]          *1X,'ENTER OUTPUT FILE NAME FOR UNIT 8 PROMPT'//)
            3 IN    = 0
              JN    = 0
              K = 1
              READ(1,900,END=2000)KC,RI,AJOB
              IF (KC-1) 10,10,5
            5 WRITE(8,990)AJOB
              STOP 11111
           10 IPG = 1
[16]          WRITE (8,940) AJOB,IPG
              WRITE(8,941) (RI(N),N =6,11),KC,(RI(N),N=1,5),(RI(N),N=12,23)
              READ(1,902)KC,CULT,CALL,STD,STN,N,IA,DEFL,SC,NOB,NOC,CLW
              IF(CLW.EQ.0.0)CLW=150.0
```

```
      IF(KC - 2)5,20,5
   20 WRITE(8,942)
      WRITE(8,943)CULT,CALL,STD,STN,N,IA,DEFL,SC,NOB,NOC,CLW
      FN = N
      GO TO (25,25,25,25,25,25),NOB
      WRITE (8,997) NOB
      STOP 11111
   25 IN    = IN    + 1
      READ(1,904,END=5) KC,MARK(IN),MT(IN),(W(IN,J),J=1,23)
      IF(KC - 3)5,30,5
   30 IF (MT(IN)-2) 32,35,31
   31 WRITE (8,998) MT(IN)
      STOP 11111
   32 IF (MT(IN)-1) 31,35,35
   35 IF(NOB - IN)40,40,25
   40 WRITE(8,944)
      DO 50 I = 1,NOB
   50 WRITE(8,945) MARK(I), MT(I)  ,(W(I,J),J=1,15)
   55 JN    = JN    + 1
      READ(1,906,END=5) KC,MARK(JN), BFT(JN),BFW(JN),TFT(JN),TFW(JN),
     1 XMFT(JN) , XLLM(JN) , XLLS(JN) , XLSM(JN)
      IF(KC - 4)5,60,5
   60 IF(NOB - JN)65,65,55
   65 WRITE(8,946)
      DO 70 I= 1,NOB
   70 WRITE(8,947)MARK(I),(W(I,J),J=16,23),BFT(I),BFW(I),TFT(I),
     1TFW(I),XMFT(I),XLLM(I),XLLS(I),XLSM(I)
   75 WEBD2 = W(K,6)
   80 IPG = IPG + 1
      LO=0
[16]  WRITE (8,940) AJOB,IPG
      W1= W(K,1)
      W2= W(K,2)
      T = W(K,4)
      TB = W(K,5)
      W4 = W(K,3)
      WEBD1 = W(K,7)
      WEBIN = W(K,8)
      WEBT = W(K,9)
      SL = W(K,10)
      FLL = W(K,11)
      FLR = W(K,12)
      LSC = W(K,13)
      CTF = W(K,14)
      CBF = W(K,15)
      WCDL = W(K,16)
      W2DL = W(K,17)
      WD = W(K,18)
      CH = W(K,19)
      CH1 = W(K,20)
      SLL = W(K,21)
      W3 = W(K,22)
      SK = W(K,23)
      XLLM1 = XLLM(K)
```

```
      XLLS1 = XLLS(K)
      XLSM1 = XLSM(K)
      BFT1 = BFT(K)
      BFW1 = BFW(K)
      TFT1 = TFT(K)
      TFW1 = TFW(K)
      XMFT1=XMFT(K)
      IT = MT(K)
      KN1=0
      KN0=0
      KBW1 = 0
      KBT1 = 0
      KTW1 = 0
      KTT1 = 0
      DO 85 I=1,4
      TFW2(I) = 0.0
      TFT2(I) = 0.0
      BFW2(I) = 0.0
      BFT2(I) = 0.0
      XT(I) = 0.0
      XT1(I)=0.0
      XB1(I)=0.0
   85 XB(I) = 0.0
      SLH = SL/2.0
      IF(XMFT1)100,90,100
   90 XMFT1= 4.0
  100 IF(BFW1)110,105,110
  105 BFW1 = 12.0
      KBW1 = 1
  110 IF(BFT1)120,115,120
  115 BFT1 = 0.75
      KBT1 = 1
  120 IF(TFW1)130,125,130
  125 TFW1 = 12.0
      KTW1 = 1
  130 IF(TFT1)140,135,140
  135 TFT1 = 0.75
      KTT1 = 1
  140 FT=BFT1
      FW=BFW1
      PER=0.0
      IF(CH)155,145,155
  145 PER = 0.11
      IF(SL - 150.0)155,155,150
  150 PER = 0.18
  155 PAC = 50.0/ (125.0 + SL) + 1.0
      IF(PAC - 1.3)165,165,160
  160 PAC = 1.3
  165 IF(XLLM1)180,170,180
  170 XLLM1 = (36.0 - 168.00024/SL)*(SL/2.0 - 2.33333) - 112.0
      IF(SL - 144.82)180,175,175
  175 XLLM1 = 4.5*SL + 0.08*SL*SL
  180 IF(XLLS1)195,185,195
  185 XLLS1 = 72.0 - 672.0/SL
```

```
      IF(SL - 127.25)195,195,190
190   XLLS1 = 0.32*SL + 26.0
195   IF(XLSM1)210,200,210
200   XLSM1 = 36.0 - 672.0/SL
      IF(XLSM1 - 13.0)205,210,210
205   XLSM1 = 13.0
210   GO TO (212,211),IT
211   S81= (W1 + W2)/2.0
      S82 = 3.0*SL
      S83 = 12.0
      TEM = S81
      SX = S81/24.0/W3
      GO TO 214
212   S81 = W1
      S82 = SL
      S83 = 6.0
      TEM = S81/2.0
      SX = W3/2.0
214   IF( T )220,215,220
215   T = 8.5
      IF(S81 - 84.0)216,216,220
216   T = 8.0
      IF(S81 - 72.0)218,218,220
218   T = 7.5
220   DT = T - 0.5
      S83 =DT*S83
      B = AMIN1(TEM,S82,S83)
      SDL=(T*S81+TFW1*TB)*CLW/144.0
      GO TO (221,222),IT
221   SDL = SDL/2.0 + W2DL
      B = B + W(K,3)
222   DLLM1 = XLLM1*PAC*SX*12000.
      DLLS1 = XLLS1*PAC*SX*1000.
      DLSM1 = XLSM1*PAC*SX*1000.
      SLLM1 = SLL*1.5*SL*SL
      SDLMC = 1.5*SL*SL*WCDL
      SLLS1=SLL*SLH
      XIS=(DLLS1-DLSM1)/20.
      TEM = SL - 4.66667
      SDL = SDL + WD
      DO 226 I = 1,20
      XI = I
      X(I) = XI*SL/40.0
      TEM1 =     SLH - X(I)
      TEM2 = SL - 2.0*X(I) - 4.66667
      TEM3 = X(I)/2.0*(SL - X(I))
      DLS(I)=DLLS1-XI*XIS
      SLS(I)=SLL*TEM1
      DLM(I)=DLLM1*(1.0-TEM2*TEM2/(TEM*TEM))
      SLM(I)=SLL*TEM3
      IF (TEM2) 223,224,224
223   DLM(I)=DLLM1
224   XDIS(I) = WCDL*TEM1
      SLS(I)=SLL*TEM1
```

```
226 XDIM(I)=WCDL*TEM3
    SLM(I)=SLL*TEM3
    KN = 0
228 J=0
    IF(LO) 229,229,232
229 BFT1=FT
    BFW1=FW
    TEM3=BFT1
    GO TO 230
232 TFT1=FT
    TFW1=FW
    TEM3=TFT1
230 J = J + 1
    TEM = 20000.
    TEM1= 36000.
    TEM2= 58000.
    IF (SC-1) 248,235,233
233 TEM3=0.75
235 TEM = 27000.
    TEM1= 50000.
    TEM2= 70000.
    IF(TEM3 - 0.75)248,248,240
240 TEM = 25000.
    TEM1= 46000.
    TEM2= 67000.
    IF(TEM3 - 1.5)248,248,245
245 TEM = 23000.
    TEM1= 42000.
    TEM2= 63000.
248 IF(J - 1)250,250,255
250 BSA = TEM
    BSY = TEM1
    BSU = TEM2
    TEM3= TFT1
    GO TO 230
255 TSA = TEM
    TSY = TEM1
    TSU = TEM2
    FSAV = 12000.
    IF (SC-1) 258,256,259
256 FSAV = 17000.
    IF(WEBT - 0.75)258,258,257
257 FSAV = 15000.
    GO TO 258
259 FSAV=17000.
258 DTT = DT + TB
    DTH = DT/2.0
    TFTH = TFT1/2.0
    TFWH = TFW1/2.0
    BFTH = BFT1/2.0
    WEBDH = WEBD2/2.0
    TD = TFT1 + WEBD2 + BFT1
    TFA1 = TFT1*TFW1
    Q1 = TFA1*TFTH
```

```
      A2 = WEBD2*WEBT
      Q2 = A2*(TFT1 + WEBDH)
      BFA1 = BFT1*BFW1
      Q3 = BFA1*(TD - BFTH)
      SI1 = Q1*TFTH
      SI2 = Q2*(TFT1 + WEBDH)
      SI3 = Q3*(TD - BFTH)
      AS=TFA1+A2+BFA1
      QT = Q1 + Q2 + Q3
      SIT = SI1 + SI2 + SI3
      SI1 = TFW1*TFT1**3/12.0
      SI2=WEBT*WEBD2**3/12.0
      SI3 = BFW1*BFT1**3/12.0
      SIT = SIT + SI1 + SI2 + SI3
      YS = QT/AS
      SI = SIT - YS*QT
      ZS1T= SI/YS
      ZS1B=SI/(TD-YS)
      SI4 = SI
      YS1 = YS
      J = 0
  260 A1 = DT*B/FN
      Q1 = A1*DTH
      A2 = TFW1*TB/FN
      Q2 = A2*(DT + TB/2.0)
      Q3 = AS*(DTT      + YS)
      SI1 = Q1*DTH
      SI2 = Q2*(DT + TB/2.0)
      SI3 = Q3*(DTT      + YS)
      AT = A1 + A2 + AS
      QT = Q1 + Q2 + Q3
      SIT= SI1 + SI2 + SI3
      SI1 = B/FN*DT**3/12.0
      SI2 = TFW1/FN*TB**3/12.0
      SIT = SIT + SI1 + SI2 + SI
      YC = QT/AT
      CI = SIT - YC*QT
      IF(J)275,270,275
  270 J = J + 1
      FN = 3.0*FN
      CIL=CI
      YCL = YC
      GO TO 260
  275 FN = FN/3.0
      YCC = YC
      YCC1= YCC
      CIC = CI
      CIC1= CIC
      CIL1= CIL
      YCL1= YCL
      WTFT1 = AS*3.40278
      BDL = WTFT1 + PER*WTFT1 + CH
      SDLM = 1.5*SL*SL*(SDL + BDL)
      FSBD = SDLM*(TD - YS)/SI
```

```
      FSBDC = SDLMC*(TD + DTT       - YCC)/CIC
      FSBL =(DLLM1 + SLLM1)*(TD + DTT      - YCL)/CIL
      FSB = FSBD + FSBDC + FSBL
      FSTD=SDLM*YS/SI
      FSTDC = SDLMC*(YCC - DTT     )/CIC
      FSTL=(DLLM1+SLLM1)*(YCL-DTT)/CIL
      FST =(FSTD + FSTDC + FSTL)
      FCDC=SDLMC*YCL/CIL/FN
      FCL =(DLLM1 + SLLM1)*YCL/CIL/FN
      FC =(FCDC + FCL )
      DO 290 I=1,20
      DM(I) = (SDL + BDL)*(X(I)/2.0*(SL - X(I)))
  290 DS(I)=(SDL+BDL)*(SLH-X(I))
      IF(NOC.EQ.0) FRBMT=FSBD+FSBDC+27500.
      IF(NOC.EQ.1) FRBMT=FSBD+FSBDC+18000.
      IF(NOC.EQ.2) FRBMT=FSBD+FSBDC+16000.
  295 FRBWT=FRBMT
      BSA=AMIN1 (BSA,FRBMT)
      IF (LO-1) 300,320,320
  300 FA=BFA1
      FT = BFT1
      FW = BFW1
      FS = FSB
      SA = BSA
      KT = KBT1
      KW = KBW1
  305 IF(FS              -SA)310,310,330
  310 IF(FS + 0.03 *SA-SA)330,315,315
  315 LO = LO + 1
      IF(LO - 1)320,320,380
  320 FT = TFT1
      FW = TFW1
      FS = FST
      SA = TSA
      FA = TFA1
      KT = KTT1
      KW = KTW1
      GO TO 305
  330 IF(KT)335,340,335
  335 IF(FT-XMFT1) 350,340,338
  338 FT=XMFT1
      IF(FS-SA) 350,350,340
  340 IF(KW)345,315,345
  345 IF(FS-SA) 350,350,348
  348 FW=FW+2.0
      KN1=KN1+1
      GO TO 228
  350 TEM= FS/SA
      FA = FA*TEM
      FT = FA/FW
      IFT = FT
      XFT = IFT
      KN=KN+1
      IF(KN-15)355,355,1330
```

```
1330 KN=0
     GO TO 315
 355 XFT = XFT + 0.0625
     IF(FT - XFT)360,360,355
 360 FT = XFT
     DO 1300 II=1,33
     IF(FT-TAB(II))1310,1320,1300
1310 FT=TAB(II)
     GO TO 1320
1300 CONTINUE
1320 IF(FT-XMFT1)365,365,362
 362 FT=XMFT1
     GO TO 340
 365 IF (FT-0.75) 370,228,228
 370 FT=0.75
     IF(LO-1) 374,376,376
 374 BFT1=FT
     BFW1=FW
     GO TO 315
 376 TFT1=FT
     TFW1=FW
     GO TO 315
 380 IF (KNO) 382,382,384
 382 LO=0
     KNO=1
     IF(KN1)300,300,383
 383 BFW1=BFW1-2.0
     GO TO 300
 384 IF (FC-CALL) 386,386,385
 385 WRITE(8,993)MARK(K) ,FC
 386 J = 0
     JXT=0
     JXB=0
     TEM = BFT1
     TEM1 = BFW1
     TEM2 = TFT1
     TEM3 = TFW1
     XKL = CBF
 395 J1 = 0
     J = J + 1
     IF(J-1) 398,398,396
 396 TEM=BFT2(J-1)
     TEM1=BFW2(J-1)
     XKL=CBF
 398 IF(J-4) 397,397,600
 397 IF(BFT2(J)) 403,403,399
 399 TEM=TFT1
     TEM1=TFW1
     XKL=CTF
     J1=J1+1
 400 IF(J-1) 404,404,402
 402 TEM=TFT2(J-1)
     TEM1=TFW2(J-1)
     XKL=CTF
```

```
 404 IF(TFT2(J)) 403,403,401
 401 TEM=BFT1
     TEM1=BFW1
     XKL=CBF
     GO TO 395
 403 J1=J1+1
     CUT=0.25*TEM
     IF(CUT.LT.0.625)CUT=0.625
     TEM=TEM-CUT
     IF(TEM-0.75)405,420,1205
1205 DO 1210 II=1,33
     IF(TEM-TAB(II))1207,420,1210
1207 TEM=TAB(II-1)
     GO TO 420
1210 CONTINUE
 405 TEM=TEM+CUT
     DO 1230 II=1,33
     IF(TEM-TAB(II))1220,1240,1230
1220 TEM=TAB(II-1)
     GO TO 1240
1230 CONTINUE
1240 IF(XKL)406,408,406
 406 TEM1=TEM1-2.0
     IF (TEM1-12.0) 407,420,420
 407 TEM1=12.0
 408 GO TO (411),J1
     DO 409 N=J,4
     TFT2(N)=TEM
 409 TFW2(N)=TEM1
     TEM=BFT1
     TEM1=BFW1
     GO TO 395
 411 DO 412 N=J,4
     BFT2(N)=TEM
 412 BFW2(N)=TEM1
     TEM=TFT1
     TEM1=TFW1
     GO TO 400
 420 IF(J-1) 821,821,823
 821 IF(J1-1) 823,823,822
 822 TEM2=TEM
     TEM3=TEM1
 823 GO TO (439),J
     GO TO(425),J1
     TEM2=TEM
     TEM3=TEM1
     TEM  = BFT2(J)
     TEM1 = BFW2(J)
     GO TO 439
 425 TEM2=TFT2(J-1)
     TEM3 = TFW2(J-1)
 439 M = 0
     A1 = TEM2*TEM3
     Q1 = A1*TEM2/2.0
```

```
      A2 = WEBD2*WEBT
      TD1 = TEM2 + WEBD2 + TEM
      Q2 = A2*(WEBDH + TEM2)
      A3 = TEM*TEM1
      Q3 = A3*(TD1 - TEM/2.0)
      AS = A1 + A2 + A3
      QT = Q1 + Q2 + Q3
      SI1 = Q1*TEM2/2.0
      SI2 = Q2*(WEBDH + TEM2)
      SI3 = Q3*(TD1 - TEM/2.0)
      SIT = SI1 + SI2 + SI3
      SI1 = TEM3*TEM2**3/12.0
      SI2 = WEBT*WEBD2**3/12.0
      SI3 = TEM1*TEM**3/12.0
      SIT = SIT + SI1 + SI2 + SI3
      YS = QT/AS
      SI = SIT - QT*YS
  428 A1 = B*DT/FN
      Q1 = A1*DT/2.0
      A2 = TEM3*TB/FN
      Q2 = A2*(DT + TB/2.0)
      Q3 = AS*(DTT      + YS)
      SI1 = Q1*DT/2.0
      SI2 = Q2*(DT + TB/2.0)
      SI3 = Q3*(DTT      + YS)
      XR = TFT1 - TFT2(J)
      IF(TFT2(J)) 427,426,427
  426 XR=0.0
  427 IF(XR)422,422,421
  421 A2 = TEM3*(TB + XR)/FN
      Q2 = A2*(DT +(TB + XR)/2.0)
      Q3 = AS*(DTT      + XR + YS)
      SI2 = Q2*(DT +(TB + XR)/2.0)
      SI3 = Q3*(DTT      + XR + YS)
  422 SIT = SI1 + SI2 + SI3
      SI1 = B/FN*DT**3/12.0
      SI2 = TEM3/FN*TB**3/12.0
      IF(XR)424,424,423
  423 SI2 = TEM3/FN*(TB + XR)**3/12.0
  424 SIT = SIT + SI1 + SI2 + SI
      AT = A1 + A2 + AS
      QT = Q1 + Q2 + Q3
      YC = QT/AT
      CI = SIT - QT*YC
      IF(M)436,431,436
  431 M = M + 1
      FN = 3.0*FN
      CIL=CI
      YCL = YC
      GO TO 428
  436 FN = FN/3.0
      YCC = YC
      CIC = CI
      GO TO(440),J1
```

```
      TFT2(J) = TEM2
      TFW2(J) = TEM3
      TEM = BFT1
      TEM1 = BFW1
      XKL = CBF
      GO TO 450
  440 BFT2(J) = TEM
      BFW2(J) = TEM1
      TEM = TFT1
      TEM1 = TFW1
      XKL = CTF
  450 ZS3T= SI/YS
      ZS3B=SI/(TD1-YS)
      GO TO (455),J1
      ZS3=(SLH-2.33)*SQRT(1.0-ZS3T/ZS1T)+2.33
  454 ZS4=SLH-ZS3
      JXT=JXT+1
      XT(J)=ZS4
      XT1(J)=ZS3
      IF(XT(J) - 15.0)515,550,550
  515 XT(J)=0.0
      XT1(J)=0.0
      IF(J - 2)520,530,530
  520 DO 525 N=1,4
      TFT2(N) = TFT1
  525 TFW2(N) = TFW1
      GO TO 540
  530 DO 535 N=J,4
      TFT2(N) = TFT2(N-1)
  535 TFW2(N) = TFW2(N-1)
  540 CTF = 0.0
      XKL = CTF
      GO TO 395
  455 ZS3=(SLH-2.33)*SQRT(1.0-ZS3B/ZS1B)+2.33
  456 ZS4=SLH-ZS3
      JXB=JXB+1
      XB(J) = ZS4
      XB1(J)=ZS3
      IF (XB(J)-15.0) 475,550,550
  475 XB(J)=0.0
      XB1(J)=0.0
      IF(J - 2)480,490,490
  480 DO 485 N=1,4
      BFT2(N) = BFT1
  485 BFW2(N) = BFW1
      GO TO 500
  490 DO 495 N=J,4
      BFT2(N) = BFT2(N - 1)
  495 BFW2(N) = BFW2(N - 1)
  500 CBF = 0.0
      XKL = CBF
      GO TO 400
  550 GO TO (565),J1
      XM = (SDL + BDL)/2.0*ZS4*(SL - ZS4)*12.0
```

```
      XM1=(WCDL+SLL)*ZS4/2.0*(SL-ZS4)*12.0
      XM2=DLLM1*(1.0-(SL-2.0*ZS4-4.6667)**2 /(SL-4.6667)**2)
      XM = XM* YS/SI
      XM1 = XM1*(YCC - DTT      - XR)/CIC
      XM2 = XM2*(YCL - DTT      - XR)/CIL
      XM = XM + XM1 + XM2
      IF(XM - TSA)555,395,560
  555 IF(XM + 0.03 *TSA - TSA)560,395,395
  560 ZS3 = ZS3*XM/TSA
      IF(JXT-8) 454,454,395
  565 XM = (SDL + BDL)*ZS4/2.0*(SL - ZS4)*12.
      XM1=(WCDL+SLL)*ZS4/2.0*(SL-ZS4)*12.0
      XM2=DLLM1*(1.0-(SL-2.0*ZS4-4.6667)**2 /(SL-4.6667)**2)
      XM = XM*(TD1 - YS)/SI
      XM1 = XM1*(TD1 + DTT      + XR - YCC)/CIC
      XM2 = XM2*(TD1 + DTT      + XR - YCL)/CIL
      XM = XM + XM1 + XM2
      BSA = AMIN1(BSA,FRBWT)
      IF(XM - BSA)570,400,575
  570 IF(XM + 0.03 *BSA - BSA)575,400,400
  575 ZS3 = ZS3*XM/BSA
      IF(JXB-8) 456,456,400
  600 N = 0
      IC1=0
      Q = B/FN*DT*(YCL - DT/2.0)
      SR=(DLLS1+SLLS1)*Q/CIL
      IF(NOC.EQ.0) SZR=10600.0*STD*STD*STN
      IF(NOC.EQ.1) SZR= 7850.0*STD*STD*STN
      IF(NOC.EQ.2) SZR= 5500.0*STD*STD*STN
      ANG = SK*3.14159/180.0
      TEM = (TFW2(4)/2.0  - WEBT/2.0)*TAN(ANG)
      TEM1 = (1.0 + STD/2.0)*TAN(ANG/2.0)
      E = TEM + TEM1
      IF (E-3.0) 601,601,602
  601 E=3.0
  602 HBM = AMIN1(FLL,FLR)
  604 N=N+1
      TEM = N
      TEM1 = E + 6.0*TEM
      IF(HBM/TEM1 - 1.0)606,606,604
  606 Q = (4 + N)*6
      HBM = SLH*12.0+ HBM
      IF(SR.EQ.0.0)GO TO 610
      SPA = SZR/SR
      IF(SPA - 24.0)620,620,610
  610 SPA = 24.0
  620 HBM= HBM-E-Q
      NSPA=HBM/SPA+1.0
      TEM = TFT2(4)*TFW2(4) + WEBT*WEBD2 + BFT2(4)*BFW2(4)
      TEM1 = AMIN1(BSY,TSY)
      TEM = TEM*TEM1
      TEM1 = 0.85*CULT*B*DT
      NO = Q/6.0
      TEM = AMIN1(TEM,TEM1)
```

```
[14]  C THE FOLLOWING EQN CHANGED PER F. CHEN INSTRUCTIONS ON 4-8-1985
[14]        TEM1=.4*STD*STD*CULT**.5*(CLW**1.5*33.0*CULT*.5)**.5
            NR = TEM/(0.85*TEM1)
        625 SPAN = NSPA
            SPA=HBM/SPAN
            NA = STN*(1.0 + SPAN + Q)
            IF(NA - NR)630,650,650
        630 TEM = NR - NA
            N = 0
        635 N = N + 1
            TEM1=N
            IF(TEM-STN*TEM1)640,640,635
        640 NSPA = NSPA + N
            GO TO 625
        650 V = DLLS1 + SLL*SLH + (BDL + SDL)*SLH + WCDL*SLH
            XIS=(DLLS1+SLLS1-DLSM1)/5.0
             FV1 = V/(WEBT*WEBD2)
            IF (SC.EQ.0.0) FSVD=36000.0
            IF (SC.NE.0.0) FSVD=50000.0
             FV=56250000.0/(WEBD2/WEBT)**2
             IF(FV.GT.(FSVD/3.0))  FV=FSVD/3.0
             IF(WEBT-(WEBD2/150.0)) 1005,2005,2005
       2005 IF(FV-FV1) 1005,1004,1004
       1004 IPATH=1
            GO TO 671
       1005 IPATH=2
            FV2=350000000.0/(WEBD2/WEBT)**2
            IF(FV2-FV1) 1003,1002,1002
       1002 DSPA1 = 0.5* WEBD2
            GO TO 671
       1003 DSPA1 = 0.25* WEBD2
        671 ICSTGO=1
            DO 690 N=1,5
            X1 = N
            X1=X1*SL/10.0
            DEF1(N)= BDL*X1*(SL**3-2.0*SL*X1*X1+X1**3)*0.72/(290000.0*SI4)
            DEF2(N)= SDL*X1*(SL**3-2.0*SL*X1*X1+X1**3)*0.72/(290000.0*SI4)
        690 DEF3(N)=WCDL*X1*(SL**3-2.0*SL*X1*X1+X1**3)*0.72/(290000.0*CIC1)
            DELAL = SL*12.0/DEFL
            DELAC=(DLLM1+SLLM1)*12.0*SL*SL/(29000000.0*CIL1)
            WRITE(8,948)MARK(K)
            FST=FST*(-1.0)
            FC=FC*(-1.0)
             WRITE(8,949) TFT1,TFW1,BFT1,    BFW1    ,WEBT,WEBD2,SI4,YS1,CIC1,
          2  YCC1,CIL1,YCL1,FSB,FST,FC
            WRITE(8,950)
            WRITE(8,951)
            WRITE(8,952) (TFT2(N),TFW2(N),XT1(N),N=1,4)
            WRITE(8,953)
            WRITE(8,951)
            WRITE(8,952) (BFT2(N),BFW2(N),XB1(N),N=1,4)
            IF (IPATH.EQ.1) WRITE(8,1954) DELAC,DELAL
            IF (IPATH.EQ.2) WRITE(8,954)DELAC,DELAL,DSPA1
        660 DO 672 I=1,5
```

```
 752 WRITE (8,999) MARK(K),FV
 754 DOL1 = 23000.0/SQRT(ABS(FST))
     IF(LSC)755,760,755
 755 DOL1 = DOLI*2.0
 760 DOL = WEBD2/WEBT
     IF(DOL - DOL1)770,770,765
 765 WRITE(8,996) MARK(K), DOL
 770 J = -9
     M = 0.
     DO 772 N=1,20
     DM(N) = DM(N)/1000.0*12.0
     XD1M(N) = XD1M(N)/1000.0*12.0
     DLM(N) = DLM(N)/1000.0
     SLM(N)=SLM(N)/1000.0*12.0
     DS(N) = DS(N)/1000.0
     XD1S(N) = XD1S(N)/1000.0
     SLS(N)=SLS(N)/1000.0
 772 DLS(N) = DLS(N)/1000.0
     FNCDTF = DM(20) * YS1/SI4*1000.0
     IF(SC.NE.0.) GO TO 1270
     DIPSPA = SQRT((20000.0 - FNCDTF)*TFW1 *TFW1/7.5)/12.0
     GO TO 1280
1270 DIPSPA = SQRT((27000.0 - FNCDTF)*TFW1 *TFW1/14.4)/12.0
1280 IF(DIPSPA.GE.25.0) DIPSPA = 25.0
     WRITE(8,957) MARK(K)
 775 J = J + 10
     M = M + 10
     WRITE(8,958) (X(I),I=J,M), (DM(I),I=J,M), (XD1M(I),I=J,M), (DLM(I)
    1,I=J,M), (SLM(I),I=J,M)
     WRITE(8,959) (DS(I),I=J,M), (XD1S(I),I=J,M), (DLS(I),I=J,M), (SLS(
    1 I),I=J,M)
     IF(M - 20)775,780,780
 780 TEM1=S81-120.0
     IF (TEM1) 778,779,779
 778 TEM1=0.0
 779 TEM=1.0+((S81-72.)+(S81-48.)+2.0*TEM1)/S81
     GO TO(782,790),IT
 782 TEM = S81 + W4 -(W2 +96.0)
     IF(TEM)788,788,785
 785 TEM =  (TEM + TEM +72.0)/S81
     GO TO 790
 788 TEM = (72.0 + TEM)/S81
     IF(TEM.LT.0.0)TEM=0.0
 790 DLLR=XLLS1/2.0*TEM*PAC+SLL*SLH/1000.
     CH1=CH1*S81/COS(ANG)/12000.
     GO TO (794,796),IT
 794 CH1=CH1/2.0
 796 TEM = (BDL + SDL)*SLH/1000.0+CH1
     TEM1 = WCDL*SLH/1000.0
     SLLS1=SLL*SLH/1000.0
     DLLS1=DLLS1/1000.0
     WRITE(8,960) TEM, TEM1, SLLS1, DLLS1, DLLR, WTFT1
     IF(WEBD2 - WEBD1)800,820,820
 800 IF(WEBIN)805,805,810
```

```
805 WRITE(8,997) MARK(K), WEBIN
    GO TO 820
810 WEBD2 = WEBD2 + WEBIN
    GO TO 80
820 K = K + 1
    IF(K - NOB)75,75,3
993 FORMAT('0',5X,'ALLOWABLE CONCRETE STRESS HAS BEEN EXCEEDED, MARK =
   1 ',A2,'STRESS = ',F5.0)
990 FORMAT('0',5X,'INPUT CARDS ARE OUT OF ORDER OR MISSING, JOB =',A4)
991 FORMAT('0',5X,'SPAN LENGTH TO GIRDER DEPTH RATIO EXCEEDED, MARK =
   1',A2,' RATIO = ',F4.1)
992 FORMAT('0',5X,'SPAN LENGTH TO COMPOSITE DEPTH RATIO EXCEEDED, MARK
   1 = ',A2,' RATIO = ',F4.1)
994 FORMAT('0',5X,'TOP FLANGE B OVER T RATIO EXCEEDED AT MIDSPAN, MARK
   1 = ',A2,' B/T = ',F4.1)
995 FORMAT('0',5X,'TOP FLANGE B OVER T RATIO EXCEEDED AT TRANSITION PO
   1INT ',I1,', MARK = ',A2,' B/T = ',F4.1)
996 FORMAT('0',5X,'WEB DEPTH TO WEB THICKNESS RATIO EXCEEDED, MARK = '
   1,A2,' RATIO = ',F5.1)
900 FORMAT(I1,A1,2A2,A3,A4,A2,A4,4A3,11A4,A1,A4)
902 FORMAT(I1,2F4.0,F4.3,F1.0,I2,I4,F4.0,F1.0,2I1,F3.0)
904 FORMAT(I1,A2,I1,3F5.2,F4.2,F3.2,2F3.0,F2.0,F4.4,F5.2,2F4.2,3F1.0,
   12F4.0,4F3.0,F4.3,F2.0)
906 FORMAT (I1,A2,2(F5.4,F4.2),F5.4,F7.1,2F5.1)
940 FORMAT('1E01-0417-01',T40,
   *        'VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION',//,58X
   *       ,'BRIDGE DIVISION',27X,//,T10,'JOB NUMBER ',A4,25X,
   *        'STEEL GIRDER DESIGN AND ANALYSIS',19X,'PAGE ',I2,/)
941 FORMAT(46X,'PROJECT CHARGE NO. ',A2,A4,4A3///9X,'DATE SUBMITTED ','
   1I1,A1,2A2,'  REQUESTED BY ',A3,'  ROUTE ',A4,'  DESCRIPTION ',11A4
   2,A1//)
942 FORMAT(19X,'CONCRETE STRENGTH   STUD    NO. OF',10X,'AASHO  DEFLEC
   1TION       NUMBER  NUMBER   WT. OF'/ 21X,'ULT.     ALL.   DIAMETER
   2  STUDS      N    SPECS.  CONSTANT   SC     OF      OF    CONCRETE'
   3/       20X,'(PSI)     (PSI)      (IN)     PER ROW',10X,'USED',18X,
   4    'BEAMS   CYCLES  LBS./C.F.'/)
943 FORMAT(17X,2F9.0,3X,F7.4,F9.0,I7,I9,F9.0,F7.0,I5,7X,I1,F12.1//)
944 FORMAT(51X,'WEB DEPTH   INCRE-  WEB'/6X,'MARK T   S1,S3   S2,S4
   1 S5       T       TB    MIN    MAX     MENT  THICK     L       LL     LR
   2  LSC CTF CBF'/11X,'Y'/11X,'P  --------------------------- INCHES -
   3--------------------------        (FT)    -- INCHES --'/)
945 FORMAT (7X,A2,I3,3F8.2,F7.2,F6.2,1X,F6.0,F6.0,  2F7.4,   3F7.2,
   11X,3F4.0)
946 FORMAT('0'/7X,'MARK  WCDL  W2DL    WD    CH    CH1   SLL  DF,E   S
   1KEW    BFT    BFW   TFT    TFW   MFT     LLMOM   LL ENSHR  MSHR'//
   212X,'------ POUNDS PER LINEAR FOOT ------',10X,'DEG  ------------
   3 INCHES ------------    FT-KIPS   KIPS    KIPS'/)
947 FORMAT (8X,A2,1X,2F6.0,2X,F5.0,1X,F5.0,1X,F5.0,1X,F5.0,F7.3,1X,
   1F5.0,F9.4,1X,F4.0,F8.4,1X,F4.0,F9.4,1X,3F8.1)
948 FORMAT('0',61X,'MARK = ',A2//9X,'MIDSPAN GIRDER DIMENSIONS - IN.',
   111X,'--------------- MIDSPAN SECTION PROPERTIES --------------- MI
   2DSPAN STRESSES-PSI'//5X,'TOP FLANGE    BOTTOM FLGE',7X,'WEB',17X,
   3'STEEL',10X,'COMPOSITE DEAD',12X,'LIVE',8X,'BOTTOM    TOP    TOP'/
   4/4X,'THKNESS WIDTH  THKNESS WIDTH  THKNESS DEPTH    I(IN.4)    YS(
```

```
   5IN)    I(IN.4)     YC(IN)    I(IN.4)      YL(IN)    STEEL    STEEL   CONC.
   6')
 949 FORMAT('0',2X,2(F8.4,F6.1),1X,F8.4,1X,F6.1,1X,3(F10.0,F10.3),
   12F8.0,F7.0)
 950 FORMAT('0'/53X,'TOP FLANGE CUTOFF POINTS')
 951 FORMAT('0',18X,'1ST CUTOFF POINT',11X,'2ND CUTOFF POINT',11X,'3RD
   1CUTOFF POINT',11X,     '4TH CUTOFF POINT'/'0',14X,'THKNESS  WIDTH
   2DISTANCE   THKNESS  WIDTH  DISTANCE   THKNESS  WIDTH  DISTANCE   T
   3HKNESS  WIDTH  DISTANCE')
 952 FORMAT('0',10X,4(F10.4,F7.1,F10.2))
 953 FORMAT('0'/52X,'BOTTOM FLANGE CUTOFF POINTS')
 954 FORMAT('0'/29X,'DEFLECTIONS - IN.',32X,'TRANSVERSE STIFFENER SPACI
   1NG - IN.'// 9X,'LIVE + I AT MIDSPAN, ACTUAL =',F7.4,', ALLOW. =',
   2F7.4//42X,'STEEL      SLAB     C.DEAD',7X,'AT CENTER OF BEARING, REQ
   3UIRED SPACING =',F7.2)
1954 FORMAT('0'/29X,'DEFLECTIONS - IN.',32X,'TRANSVERSE STIFFENER SPACI
   1NG - IN.'// 9X,'LIVE + I AT MIDSPAN, ACTUAL =',F7.4,', ALLOW. =',
   2F7.4//42X,'STEEL      SLAB     C.DEAD',7X,'AT CENTER OF BEARING, STI
   3FFENERS NOT REQUIRED')
 955 FORMAT('0',9X,'AT SPAN',I2,'/10 POINT, DEFLECTION =',F7.4,2F9.4,
   *          7X,'AT SPAN',I2,'/10 POINT, REQUIRED SPACING =',F7.2)
 956 FORMAT('0'/60X,'STUD SPACING'//12X,'END DISTANCE, E =',F6.2,' IN.,
   1  END GROUP = ',I1,' AT 6 IN. =',F6.2,' IN.,  ',I3,' SPACES AT ',
   2F5.2,' IN. =',F7.2,' IN.')
 957 FORMAT('0'/61X,'MARK = ',A2/'0',38X,'MOMENTS(IN-KIPS) AND SHEARS(K
   1IPS) AT SPAN 40TH POINTS'//)
 958 FORMAT('0',12X,'POINTS',10F10.2//'0',9X,'DEAD MOMENT',F9.1,9F10.1/
   1'0',9X,
   2'C.D. MOMENT',F9.1,9F10.1/'0',9X,'LIVE MOMENT',F9.1,9F10.1/
   3'0',9X,'SWLL MOMENT',F9.1,9F10.1/)
 959 FORMAT('0',9X,'DEAD SHEAR',10F10.1/'0',9X,'C.D. SHEAR',10F10.1/'0'
   1,9X,'LIVE SHEAR',10F10.1/'0',9X,'SWLL SHEAR',10F10.1//)
 960 FORMAT('0',9X,'DEAD LOAD END SHEAR=',F7.1,' KIPS, COMPOSITE DEAD
   1LOAD END SHEAR=',F7.1,' KIPS, SWLL END SHEAR =',F7.1,' KIPS'
   2/'0',23X,'LIVE LOAD END SHEAR =',F7.1,' KIPS,      LIVE LOAD END RE
   3ATION =',F7.1,' KIPS'/'0',47X,'GIRDER WEIGHT PER FOOT = ',F7.1,' P
   4OUNDS')
C1290 FORMAT('0'//12X,'*** MIDSPAN NON-COMP COMPRESSIVE STRESS = ',F9.0,
C    *' PSI'/)
C1350 FORMAT('0'/12X,'*** ALLOWABLE DIAPHRAGM SPACING = ',F9.3,' FEET'/)
  997 FORMAT('0',5X,'THE NUMBER OF BEAMS INPUT VALUE IN CARD TWO IS LESS
    1 THAN ONE OR GREATER THAN SIX, NOB = ',I2)
  998 FORMAT('0',5X,'THE BEAM TYPE INPUT VALUE IS LESS THAN ONE OR GREAT
    1ER THAN TWO, TYPE =',I2)
  999 FORMAT('0',5X,'THE SHEAR STRESS IN THE GIRDER WEB EXCEEDS ALLOWABL
    1E, MARK = ',A2,', STRESS = ',F5.0)
 1101 FORMAT('0'/29X,'DEFLECTIONS - IN.',32X,'TRANSVERSE STIFFENER SPACI
    1NG - IN.'// 9X,'LIVE + I AT MIDSPAN, ACTUAL =',F7.4,', ALLOW. =',
    2F7.4//42X,'STEEL      SLAB     C.DEAD',7X,'AT CENTER OF BEARING, STI
    1FFENERS NOT REQUIRED')
 1102 FORMAT('0',9X,'AT SPAN',I2,'/10 POINT, DEFLECTION =',F7.4,2F9.4,
    *          7X,'AT SPAN',I2,'/10 POINT, STIFFENERS NOT REQUIRED')
 2000 STOP
      END
```

6. If all goes well, the program will run for a minute or two and display the following message when execution is complete.

   Stop - Program terminated.

   At this point program results are ready for inspection.

7. If the program generates an error message, the most probable cause will be an error in the input file. Inspect the input file to make sure all data are in proper position and rerun the program, repeating steps 3 thru 6.

Note 1.

The disk drive on which the program and input/output files reside is purely a matter of convenience. All three files can reside on one drive if desired. However, if the size of the problem is large, it may be advantageous to direct the output file to the other drive, since this file can become very large.

EXAMPLE: Run program STEEL with input data file named PROB.N01, and output data file named RESULT.N01.

User Input-

1. Boot the operating system (if not already done).

2. Put program in drive A and input data file in drive B.

3. Type STEEL (CR).

4. Type B:PROB.N01 (CR).

5. Type B:RESULT.N01 (CR).

6. Inspect results in file RESULT.N01 when execution stops.

A.7 Deck Slab Program

This section will describe in detail the procedures used to compile and link the Deck Slab Design Program. After the procedure is described, directions will be given illustrating how to create data input files and run the program on the microcomputer.

A.7.1 Program Description

This program is used to compute the design parameters of reinforced concrete slabs at the centerline of the span between stringers. Along with the design parameters, the program will calculate the maximum moment that the slab will resist and the stress in the concrete and steel at the maximum moment. See reference 16 for other details.

A.7.2 Compiling and Linking Considerations

The compiling and linking considerations are the same as those for the Steel Girder Design and Analysis Program. Refer back to sect. A.6.2.

A.7.3 Compile and Link Procedure

The compile and link procedure for this program is identical to that used for the Steel Girder Design and Analysis Program. The only difference is that DISK 3 will contain the following files: SLAB.FOR, SLAB.OBJ, SLAB.EXE

Refer to section A.6.3 for the compile and link procedure. In all places where STEEL is typed in, use SLAB.

A.7.4 Deck Slab Design Program Source Listing

This section contains the FORTRAN source listing for the microcomputer converted version of the Deck Slab Design Program. The changes that have been made are indicated in bold type. These changes are listed in Table A.2 and are cross-referenced using the numbers to the left of the affected line.

LISTING

```
[1]           PROGRAM SLAB
        C     PROGRAM  FOR DESIGN OF CONCRETE  SLABS
        C     USING VARIABLE ALLOWABLE WORKING STRESSES FOR CONCRETE AND STEEL
              DIMENSION  AST(10,10),ASB(10,10),SLABT(20),BARA(10),CC(20),D(20)
[18]          DIMENSION  BD(10),INUM(9)
              CHARACTER IDTE*6,IREQD*3,IRTE*4,IPROJ*17,IDESC*46,JOB*4
              COMMON AFS,AFC,RN,FS,FC,AKD
[5]           WRITE(*,2)
[5]         2 FORMAT(15X,'VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION'//
[5]          *32X,'BRIDGE DIVISION'//21X,'           DECK SLAB DESIGN
[5]          *//////1X,'NOTE:  '/1X,'ENTER INPUT FILE NAME FOR UNIT 8 PROMPT'/
[5]          *1X,'ENTER OUTPUT FILE NAME FOR UNIT 9 PROMPT'//)
         2000 IPG=1
              ICTR=0
         1000 READ(8,1,END=999) IDTE,IREQD,IRTE,IPROJ,IDESC,JOB
[17]        1 FORMAT(A6,A3,A4,A17,A46,A4)
              READ(8,50) B,RATA,UNUSED,TD,RN,JJ,AFC,AFS,CCF,CCL,DCC,SLABTF,
             1SLABTL,DSLABT
           50 FORMAT(F5.2,F4.3,2F5.3,F2.0,I2,F4.0,F5.0,6F5.3)
        —     LL = (CCL-CCF)/DCC +1.0
           52 DO 54  L=1,LL
              IF (L-1) 51,53,51
           53 CC(L) = CCF
              GO TO 54
           51 CC(L) = CC(L-1) +DCC
           54 CONTINUE
              READ(8,55)(INUM(I),BD(I),BARA(I),I=1,9)
[17]       55 FORMAT(I1,F4.3,F3.2,8(I2,F4.3,F3.2)) [I1 WAS A1]
              KK = (SLABTL -SLABTF)/DSLABT + 1.0
           58 DO 60  K=1,KK
              IF (K-1) 61,62,61
           62 SLABT(K) = SLABTF
              GO TO 60
           61 SLABT(K) =  SLABT(K-1) +DSLABT
```

```
 60 CONTINUE
    WRITE (9,700) JOB,IPG
700  FORMAT ('1',27X,'VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATI
   1ON, BRIDGE DIVISION'///' ',2X,'JOB NO. ',A4,41X,'DECK SLAB DESIGN',
   253X,'PAGE ',I2///)
    N=RN
    IFC=AFC
    IFS=AFS
    WRITE (9,600)IDTE,IREQD,IRTE,IPROJ,IDESC
    WRITE (9,601)B,RATA,UNUSED,TD,N,JJ,IFC,IFS,CCF,CCL,DCC,SLABTF,
   1SLABTL,DSLABT
600 FORMAT(' ',2X,'DATE SUBMITTED ',A6,' REQ BY ',A3,' RTE ',A4,' PROJ
   1. NO. ',A17,' DESCRIPTION ',A46)
601 FORMAT('0',10X,'B',7X,'A''S',4X,'WS AND     D'' ',6X,'N   JJ   ALLO
   1WABLE STRESSES',8X,'BAR SPACINGS',13X,'SLAB THICKNESS'/' ',18X,'/A
   2S',5X,'COVER',23X,'CONCRETE   STEEL',5X,'FROM',5X,'TO   INTERVAL',
   34X,'FROM',5X,'TO   INTERVAL'/' ',9X,'(IN)',13X,'(IN)',5X,'(IN)',
   416X,'(PSI)',5X,'(PSI)',5X,'(IN)',4X,'(IN)',4X,'(IN)',6X,'(IN)     (
   5IN)     (IN)'
   5   //'0',8X,F6.2,3X,F5.3,2F9.3,2I5,5X,I4,6X,I5,2X,3F8.3,2X,3F8.3/)
    WRITE (9,602)                   (INUM(I),BD(I),BARA(I),I=1,7)
602 FORMAT('0',2X,'------BAR------   ------BAR------   ------BAR------
   1   ------BAR------   ------BAR------   ------BAR------   ------BAR
   2------'/' ',2X,'NO   DIA   AREA   NO   DIA   AREA   NO   DIA   ARE
   3A   NO   DIA   AREA   NO   DIA   AREA   NO   DIA   AREA   NO   DIA
   4   AREA'/' ',6X,'(IN)    (IN)',7X,'(IN)    (IN)',7X,'(IN)    (IN)',7X
   5,'(IN)    (IN)',7X,'(IN)    (IN)',7X,'(IN)    (IN)',7X,'(IN)    (IN)'/
   6'0',3X,I1,F7.3,F6.2,6(3X,I2,F7.3,F6.2))
[19]    IF(INUM(8)-0)604,605,604
604 WRITE (9,606)(INUM(I),BD(I),BARA(I),I=8,9)
606 FORMAT ('0',2X,'------BAR------   ------BAR------'/' ',6X,'(IN)
   1(IN)       (IN)    (IN)'/'0',2X,I2,F5.3,2X,F4.2,3X,I2,F7.3,F6.2)
605 IPG=IPG+1
    WRITE(9,20) IDTE,IPG,AFC,AFS,RN,RATA
    WRITE (9,23)
    DO 501  K=1,KK
  6 D(K)  =  SLABT(K)-UNUSED
    DO 501  J=1,JJ
    DO 501  L=1,LL
    ASB(L,J)   =  BARA(J)*B/CC(L)
  4 AST(L,J)   =  ASB(L,J)*RATA
    SUBAA=(2.0*RN-1.0)*AST(L,J)+RN*ASB(L,J)
    SUBAC=2.0*B*((1.0-2.0*RN)*AST(L,J)*TD-RN*ASB(L,J)*D(K))
100 AKD   = (SQRT (SUBAA**2.0-SUBAC)-SUBAA)/B
  8 IF(AKD-TD-BD(J)/4.0)9,10,10
  9 IF(AKD-TD+BD(J)/4.0)11,11,12
 11 PARAA = RN*(ASB(L,J)+AST(L,J))
    PARAC = 2.0*RN*(ASB(L,J)*D(K)+AST(L,J)*TD)*B
101 AKD   = (SQRT (PARAA**2.0 + PARAC) -PARAA)/B
    CALL  STRESS (D(K))
    PROPA = AST(L,J)*(TD-AKD)/(D(K)-AKD)
    PROPB =  TD -AKD/3.0
    PROPC =  ASB(L,J)*(D(K)-AKD/3.0)
102 AJD   =  (PROPA * PROPB +PROPC)/(PROPA +ASB(L,J))
```

```
103 RMM   =  FC * B*AKD*AJD/24.0
104 RMCFC =  RMM*12.0/FC
    GO TO 500
 12 PARBA  =  RN*ASB(L,J)
    PARBC  =  2.0*RN*ASB(L,J)*D(K)*B
105 AKD    = (SQRT (PARBA**2.0 +PARBC) -PARBA)/B
    CALL  STRESS (D(K))
106 AJD  = D(K)-AKD/3.0
    GO TO 103
 10 CALL  STRESS (D(K))
    ARC = B*AKD/2.0
    ARS=(2.0*RN-1.0)*AST(L,J)*(1.0-TD/AKD)
    TAR =  ARC + ARS
    QARC =  ARC*(D(K)-AKD/3.0)
    QARS =  ARS*(D(K)-TD)
    TQAR =  QARC +QARS
107 AJD  =  TQAR/TAR
108 RMM  =  TQAR*FC/12.0
    GO TO 104
500 LNO  = 8.0 * BD(J)
    WRITE(9,40)SLABT(K),LNO,CC(L),D(K),ASB(L,J),AKD,AJD,RMCFC,RMM,FC,
   1FS
    IF(ICTR-29)502,503,503
503 IPG=IPG+1
    WRITE(9,20) IDTE,IPG,AFC,AFS,RN,RATA
    WRITE(9,23)
    ICTR=0
    GO TO 501
502 ICTR=ICTR+1
501 CONTINUE
    GO TO 2000
 20 FORMAT('1',10X,'DATE  ',A6,23X,'PROPERTIES OF SLABS AT CENTER LINE
   1 OF SPAN',20X,'PAGE',I4/
   1'0',53X,'ALLOWABLE FC =',F8.0,'  PSI'/'0',53X,'ALLOWABLE FS =',
   2F8.0,'  PSI'/'0',53X,'N =',F5.0,'    A''S/AS =',F6.3)
 23 FORMAT('0',28X,'T    BAR  BAR     D      AS     KD     JD   RES MOM
   1   RESIST.   FC      FS'/' ',32X,'SIZE  SPA.
   2      DIV BY FC  MOMENT'/' ',27X,'(IN)         (IN)   (IN)    (IN2)  (I
   3N)    (IN)    (IN3)    (FT LBS)  (PSI)  (PSI)'//)
 40 FORMAT ('0',25X,F6.2,I4,2F7.2,3F7.3,F7.2,F9.0,F8.0,F7.0)
999 STOP
    END
    SUBROUTINE  STRESS(D)
    COMMON AFS,AFC,RN,FS,FC,AKD
109 FC   = AFS*AKD/(RN*(D-AKD))
    IF(FC-AFC)13,13,14
 14 FC = AFC
110 FS = AFC*RN*(D-AKD)/AKD
    GO TO 15
 13 FS = AFS
 15 RETURN
    END
[20]  C     SUBROUTINE FSTB(T,M,F)
      C     REAL M,M1,M2
```

```
C       W=150.*T/12.+15.
C       A=.1*W
C       B=520.
C       C=1040.-M
C       S=(-B+SQRT(B**2-4.*A*C))/(2.*A)
C       M1=M-.1*W*S**2
C       M2=M-M1
C       F1=(M1/M)*F
C       IF (F1.LE.21000) RETURN
C       M=M1*(21000/F1)+M2
C       F=F-(F1-21000)
C       RETURN
C       END
```

4. If all goes well, the program will run for a minute or two and display the following message when execution is completed:

   Stop - Program terminated.

   At this point the program results are ready for inspection.

5. If the program generates an error message the most probable cause will be an error in the input file. Inspect the input file to make sure all data are in proper position and rerun the program, repeating steps 1 thru 4.

Note 1:
The disk drive on which the program and input/output files reside is purely a matter of convenience. All three files can reside on one drive if desired. However, if the size of the problem is large, it may be advantageous to direct the output file to the other drive, since this file can become very large.

A.8 Critical Moments and Shear for Moving Loads Program

This section will describe in detail the procedure used to compile and link the Critical Moments and Shears for Moving Loads Program. After the procedure is described, directions will be given illustrating how to create data input files and run the program on the microcomputer.

A.8.1 Program Description

This program computes the maximum moment and shear in a beam due to a series of concentrated loads, such as that imposed by a multiaxle vehicle. Simple beam analysis is used for load distribution, and vehicles with 25 axles can be accommodated. (17)

A.8.2 Compiling and Linking Consideration

The compiling and linking considerations are the same as those for the Steel Girder Design and Analysis Program. Refer back to section A.6.2.

A.8.3 Compile and Link Procedure

The compile and link procedure for this program is identical to that used for the Steel Girder Design and Analysis Program. The only difference is that DISK 3 will contain the following files: MOMENT.FOR, MOMENT.OBJ, MOMENT.EXE.

Refer to section A.6.3 for the compile and link procedure. In all places where STEEL is typed in use MOMENT.

A.8.4 Critical Moments and Shears for Moving Loads Program

This section contains the FORTRAN source listing for the microcomputer version of the Critical Moments and Shears for Moving Loads Program. The changes that have been made are indicated in bold type. These changes are listed in Table A.2 and are cross-referenced using the numbers to the left of each affected line.

LISTING

```
[1]          PROGRAM MOMENT
      C              CRITICAL MOMENTS AND SHEARS PROGRAM
[21]         CHARACTER*2 ZM
             DIMENSION WL(25), DB(25),  X(25), Y(25), Z(25), B(25,25)
[5]          WRITE(*,550)
[5]      550 FORMAT(15X,'VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION'//
[5]         *32X,'BRIDGE DIVISION'//21X,'CRITICAL MOMENTS AND SHEARS PROGRAM '
[5]         */////1X,'NOTE: '/1X,'ENTER INPUT FILE NAME FOR UNIT 5 PROMPT'/
[5]         *1X,'ENTER OUTPUT FILE NAME FOR UNIT 6 PROMPT'//)
           2 READ (5,1,END=500) IM, (X(I), I = 1,20), BOJNO
           1 FORMAT (I2, A4, A2, A3, 5A4, A1, 11A4, 1X, A3)
             IF (IM - 1)23, 5, 4
           4 WRITE (6,3)
           3 FORMAT (23H CARDS ARE OUT OF ORDER)
[22]      23 GO TO 500
           5 WRITE (6,6) BOJNO, (X(I), I = 1,20)
           6 FORMAT('1',32X,'VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION
            1, BRIDGE DIVISION'/' JOB NO. ',A3,41X,'CRITICAL MOMENTS AND SHEARS
            2'//'0DATE SUBMITTED ',A4,A2,' REQ.BY ',A3,' ROUTE ',A4,' PROJECT N
            3UMBER ',4A4,A1,' DESCRIPTION ',10A4,A3/)
[25]         READ (5,7)   IM, SF, ST, SB, NO, S
[17]       7 FORMAT (I2, 3F5.2, I2)
             IF (IM - 2) 4,13, 4
[23]
          13 WRITE (6,14) SF,ST, SB, NO
          14 FORMAT (1H0, 41X, 28H(SPAN LENGTH IN  FEET)   NO., /, 43X, 27HFROM
            1      TO       BY     AXLE, /, 1H0, 39X, 3F8.2, I5)
             IM = 0
          16 IM = IM + 1
             MM = 5 * IM
             LM = MM - 4
[25]         READ (5,17)     NM, (WL(I), DB(I), I = LM, MM)
[17]      17 FORMAT (I2, 5(F9.3, F6.3))
             IF (NM - IM - 2) 4, 18, 4
[24]
          18 IF (MM - NO) 16, 21, 21
```

```
   21 WRITE (6,22) (I, WL(I), DB(I), I = 1,NO)
   22 FORMAT (1H0, 43X, 24HAXLE   LOADING  DISTANCE, //,(44X,I5,2F10.3))
      WRITE (6,26)
   26 FORMAT (81H0  SPAN   MAXIMUM    MOMENT     MOMENT     MOMENT  LOCA
     1TED AT   SHEAR  LOCATED AT,/,81H0 LENGTH   MOMENT    X 1.25     X
     21.30    BY AASHO   IM JM KM (NO IMP.)  LM MM NM,//)
C     INPUT COMPLETED, INITIALIZATION BEGUN
      S = SF
      IP = NO + 19
    8 IM = 1
      JM = 1
      XM = 0.0
C     FIND MAXIMUM MOMENT FOR SINGLE LOADS (FIG. 1)
      DO 35 I = 1,NO
      IF(I - 1)10,20,10
   10 IF(DB(I-1) - S/2.0)35,35,15
   15 IF(I - NO)20,25,20
   20 IF(DB(I) - S/2.0)35,35,25
   25 XMAX = S * WL(I) / 4.0
      IF(XM - XMAX)30,35,35
   30 XM = XMAX
      KM = I
      JM = I
   35 CONTINUE
C     FIND MAXIMUM SHEAR FOR SINGLE LOADS
      LM = 1
      MM = 1
[21]  ZM = 'RL'
      SM =WL(1)
      IF(SM - WL(NO))40,45,45
[21] 40 ZM = 'RR'
      SM = WL(NO)
      MM = NO
C     INITIALIZE TO STEP THRU NO. OF LOADS
   45 DO 215 II = 2,NO
C     COMPUTE MAXIMUM SHEAR ON LEFT FOR II LOADS (FIG. 2)
      SMAX = WL(II)
      T = 0.0
      DO 50 I = 2,II
      J = II + 1 - I
      T = DB(J) + T
      IF(S - T)55,55,50
   50 SMAX = (1.0 - T/S) * WL(J) + SMAX
   55 IF(SM - SMAX)60,65,65
   60 SM = SMAX
      LM = I - 1
      MM = II
[21]  ZM = 'RL'
C     COMPUTE MAXIMUM SHEAR ON RIGHT FOR II LOADS (FIG. 4)
   65 JJ = NO + 2 - II
      SMAX = WL(JJ - 1)
      T = 0.0
```

```
      DO 70 I = JJ,NO
      T = DB(I-1) + T


      IF (S - T)75,75,70
   70 SMAX = (1.0 - T/S) * WL(I) + SMAX
   75 IF (SM - SMAX)80,85,85
   80 SM = SMAX
      MM =  NO  + 1 - II
      LM = I - MM
[21]  ZM = 'RR'
C     DO II LOADS AT (NO + 1 - II) POSITIONS (FIG. 3)
   85 JJ = JJ - 1
      DO 215 KK = 1,JJ
C     START AT LOAD WL(KK) AND USE II LOADS. (FIG. 3)
C     SET UP B MATRIX WITHOUT SIGNS (FIG. 5)
      B(1,1) = 0.0
      DO 90 I = 2,II
      B(I,I) = 0.0
      DO 90 J = I,II
      K = KK + J - 2
      B(I-1,J) = B(I-1,J-1) + DB(K)
   90 B(J,I-1) = B(I-1,J)
C     DO SIGNS OF B MATRIX  NW CORNER  (FIG. 6)
      IF(II - 3) 115,95,95
   95 K = (II -1)/ 2
      L = K + 1
      DO 100 J = 1,K
      M = J + 1
      DO 100 I = M,L
  100 B(I,J) = - B(I,J)
C     DO SIGNS OF MATRIX  SE CORNER (FIG. 7)
      IF (II -3)115,115,105
  105 M = II - 1
      L = M - (II - 4)/ 2
      DO 110 I = L,M
      K = I + 1
      DO 110 J = K,II
  110 B(I,J) = -B(I,J)
C     SET UP Y MATRIX (FIG. 8)
  115 T = 0.0
      H = 0.0
      DO 120 J = 1,II
      K = KK + J - 1
      H = H + WL(K)
  120 T = WL(K) * B(1,J) + T
      Y(1) = T / H
      DO 240 J = 2, II
  240 Y(J) = B(1,J) - Y(1)
C     SET UP SIGNS ON Y MATRIX (FIG. 8)
      IF(II - 2) 135,135,125
  125 J = (II + 1)/ 2
      DO 130 I = 2,J
  130 Y(I) = -Y(I)
```

```
C      SET UP X AND Z MATRICES (FIG. 9)
  135 DO 140 J = 1,II
      T = (S - Y(J)) / 2.0
      X(J) = T - B(1,J)
  140 Z(J) = T - Y(II)
      J = II / 2
      DO 145 I = 1,J
      K = II + 1 - I
      Z(K) = Z(K) + Y(K)
  145 X(K) = X(K) + Y(K)
C      TESTS FOR END OF SPAN (FIG. 10)
      DO 215 I = 1,II
      IF(KK - 1)150,155,150
  150 IF(DB(KK-1) - X(I))215,215,155
  155 IF(X(I))215,215,160
  160 IF(KK - JJ)165,175,165
  165 J = KK + II - 1
  170 IF(DB(J) - Z(I))215,215,175
  175 IF(Z(I))215,215,180
C      COMPUTE U AND V TERMS IN MAXIMUM MOMENT EQUATION (FIG. 11)
  180 U = 0.0
      V = 0.0
      K = KK + I - 1
      DO 190 J = 1,II
      L = KK + J - 1
      T = WL(L) * B(I,J)
      IF (T) 185,190,190
  185 V = V + T
  190 U = U + T
      XMAX =((Y(I)/S + 1.0)* H/2.0 - U/S) * (S - Y(I))/2.0 + V
  205 IF (XM - XMAX)210,215,215
  210 XM = XMAX
      IM = II
      JM = KK
      KM = K
  215 CONTINUE
      IF(XM)220,235,220
  220 XM3 = XM * 1.25
      XM4 = XM * 1.30
      T = 50.0 / (S + 125.0) + 1.0
      IF (T - 1.3) 230, 230, 225
  225 T = 1.3
  230 XM5 = XM * T
[21]  WRITE(6,200)S,XM,XM3,XM4,XM5,IM,JM,KM,SM,LM,MM,ZM
  200 FORMAT (F8.2,4F11.3,3I3,F11.3,2I3,1X,A2)
  235 S = S + SB
      IP = IP + 1
      IF (MOD(IP,54)) 255,245,255
  245 WRITE (6,250)
  250 FORMAT (1H1)
  255 IF(S - ST)8,8,2
[22]  500 STOP
```

A.8.5 Guide for Using Microcomputer Version of Critical Moments and Shears on a Simple Span Program - Version 1.0

The following is a guide for using the microcomputer version of the Virginia Department of Highways and Transportation Critical Moments and Shears on a Simple Span Program.

The form of the data to be input into the program is identical to the format currently used on existing data input forms. These forms can be filled out as desired and an input file for the program created from them, with all data in the appropriate lines and columns. Input files are currently created using the IBM Personal Editor. See the section "Creating Program Input Files" for information on how to create these files.

PROCEDURE FOR RUNNING CRITICAL MOMENTS AND SHEARS PROGRAM

This procedure assumes that the program will be on drive A and the input files and output files will be on drive B. Drive A will be the default drive. (See note 1 below.) All user responses are indicated with bold type.

1. With the program in drive A [A>prompt], type MOMENT and hit the RETURN key. After a few seconds the folllowing header will appear:

VIRGINIA DEPARTMENT OF HIGHWAYS AND TRANSPORTATION
BRIDGE DIVISION
CRITICAL MOMENTS AND SHEARS PROGRAM

NOTE:
ENTER INPUT FILE NAME FOR UNIT 5 PROMPT
ENTER OUTPUT FILE NAME FOR UNIT 6 PROMPT

File name missing or blank - Please enter name

UNIT 5?

2. In response to the unit 5 prompt type B:infile and hit RETURN, where infile is any legal MS-DOS file name used to designate the input file created on drive B:. After a few seconds the following prompt will appear:

UNIT 6?

3. In response to this prompt type B:resfile and hit RETURN. Resfile is any legal MS-DOS file name you wish to call the output file. This will cause the output from the program to be written to the file resfile, which will reside on drive B:. (Alternatively, to direct all output to the printer, type lpt1 and RETURN in response to the unit 6? prompt.)

4. If all goes well, the program will run for a minute or two and display the following message when execution is completed:

   Stop - Program terminated.

   At this point the program results are ready for inspection.

5. If the program generates an error message the most probable cause will be an error in the input file. Inspect the input file to make sure all data are in proper position and rerun the program, repeating steps 1 thru 4.

Note 1:
The disk drive on which the program and input/output files reside is purely a matter of convenience. All three files can reside on one drive if desired. However, if the size of the problem is large, if may be advantageous to direct the output file to the other drive, since this file can be become very large.