1981 Winter Simulation Conference Proceedings
T.I. Ören, C.M. Delfosse, C.M. Shub (Eds.)

A GPSS AUTOFLOW PROGRAMME

Robert Greer Lavery P. Eng.
Ryerson Polytechnical Institute
Toronto Ontario Canada

and

Edward R. Kungla
Programming Division
Ontario Hydro
Toronto Ontario Canada

ABSTRACT

This paper describes a programme which accepts a GPSS source deck as input data and produces as output a block diagram of the contained GPSS model on a Calcomp plotter.

INTRODUCTION

The purpose of the project described in this paper was to design and code a programme which would produce a plot of the block diagram of all segments of a GPSS model submitted to the programme as data. The general structure of the programme is illustrated in Figure 1. The programme consists of two modules, one written in COBOL and the other in FORTRAN.

PROGRAMME STRUCTURE AND DESIGN

The COBOL module reads a GPSS source card, checks to see if the card contains a GPSS block instruction and then either creates an output record (if a valid GPSS block instruction was recognized) or prints an appropriate message (for non-instruction statements).
An input record is considered a valid instruction if it is not a JCL statement (// in columns 1 and 2) or a comment statement (* in column 1), if it contains an accepted operation code in a given range on the card (only the first four characters of all operation codes are tested) and if at least one operand exists. Free format coding of instruction statements is allowed for in checking each card, i.e. all 72 columns are checked. The programme is designed to check up to 16 label characters per instruction although only the first five characters of each label are reproduced in the eventual plotted block diagram.
Auxiliary operators are only checked for the GATE block since a different block symbol can result from different auxiliary operands with this instruction.
For each valid block instruction found, an output record is created and passed to the FORTRAN module. When the end of the GPSS source module data set is determined, an end-of-file record containing dummy data (99999) is created and execution of the COBOL module is terminated.
The output records created and transferred as data to the FORTRAN module are rigid in format in spite of the free form coding anticipated on the source records. The first five characters of the record are always the instruction label, the next four are the block command, etc.
A flowchart giving an overview of the COBOL module logic is shown in Figure 2.
At Ryerson Polytechnical Institute, Toronto where the programme was developed, a Calcomp plotter was available which could be accessed through the "high-speed" FORTRAN compiler routine known as WATFIVP. Thus the portion of the programme which defines the plotter data and issues the appropriate calls to the plotter routines was written in FORTRAN.

81CH1709-5/81/0000-0079$00.75 © 1981 IEEE

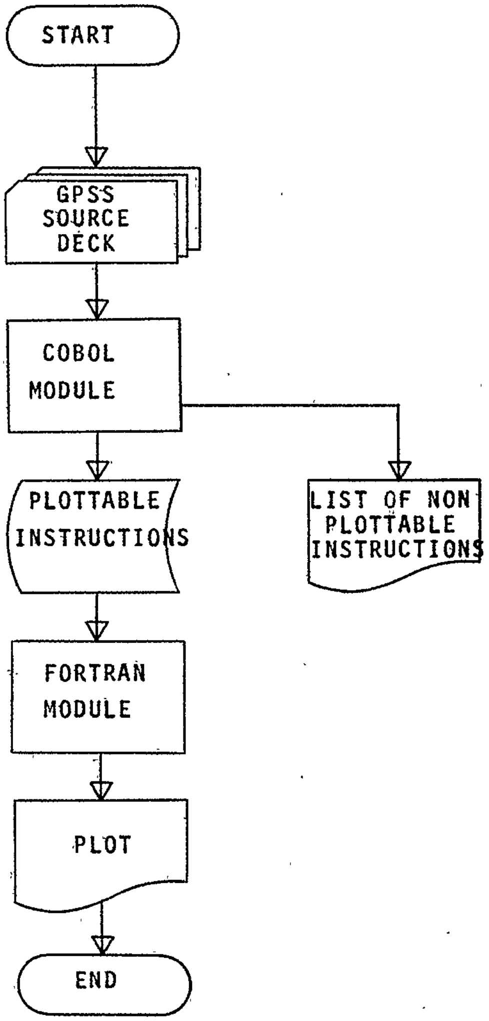

Figure 1: Flowchart Illustrating Programme General Structure.

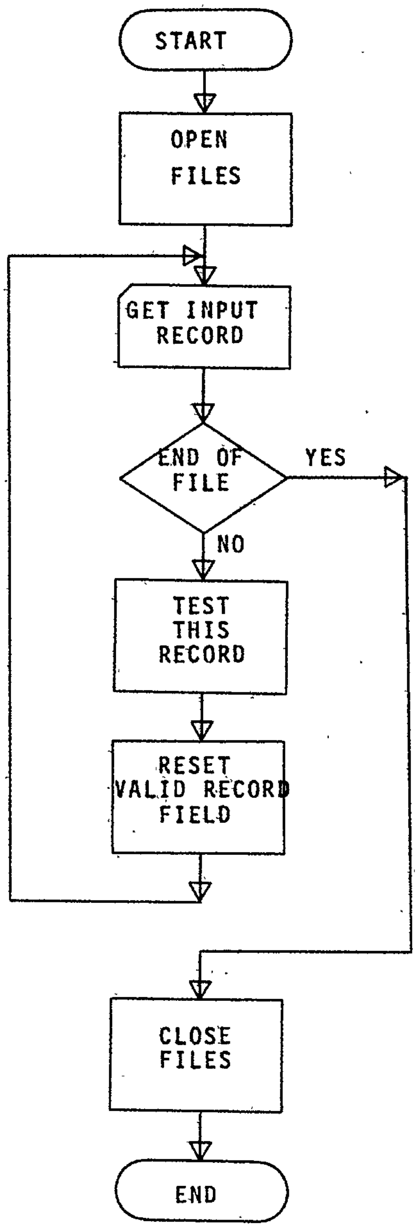

Figure 2: Flowchart Illustrating COBOL Module Logic.

The main logic of the FORTRAN module (illustrated in Figure 3) reads an input record, checks the GPSS block instruction on the input record and calls the appropriate subroutine to plot a symbol representing that block. The input record is then also printed. If the input record is the end-of-file record the programme will terminate. If not, a new input record will be read.

No plotting is done if the input record accessed does not contain a valid block instruction. This should only occur for the end-of-file record.

The programme, as designed, contains a separate subroutine for each symbol in the GPSS V instruction set. Users wishing to add additional block symbols need only add the appropriate plotting subroutine to the FORTRAN module and the equivalent block instruction to the table of acceptable operation codes.

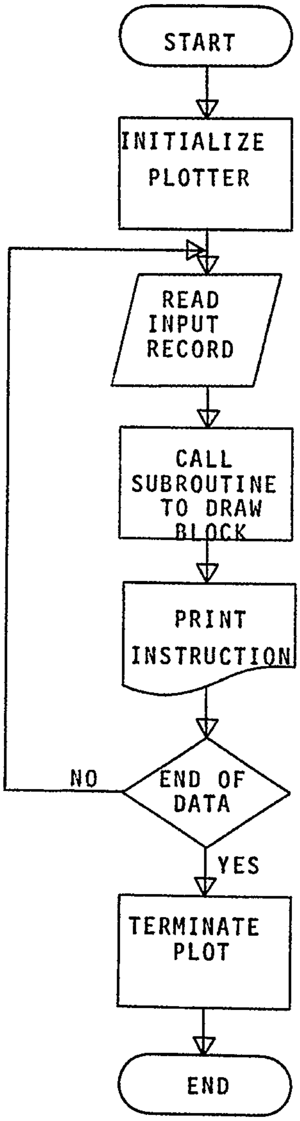

Figure 3: Flowchart Illustrating FORTRAN Module Logic.

The instructions for which separate symbols are provided are listed in Table 1. The symbols plotted for the blocks are those shown in the GPSS textbooks by Thomas J. Schriber (GPSS /360 instruction set) and Bobillier, Kahan and Probst(additional GPSS V instructions).
A copy of the source code for this programme is available to interested parties upon written request to Professor Greer Lavery at Ryerson Polytechnical Institute.
A sample of the output of the described programme is shown in Figure 4.

ADVANCE
ALTER
ASSEMBLE
ASSIGN
BUFFER
CHANGE
COUNT
DEPART
ENTER
EXAMINE
EXECUTE
FAVAIL
FUNAVAIL
GATE (FACILITY)
GATE (LOGIC)
GATE (MATCH)
GATE (STORAGE)
GATHER
GENERATE
HELP
INDEX
JOIN
LEAVE
LINK
LOGIC

LOOP
MARK
MATCH
PREEMPT
PRINT
PRIORITY
QUEUE
RELEASE
REMOVE
RETURN
SAVAIL
SAVEVALUE
SCAN
SEIZE
SELECT
SPLIT
SUNAVAIL
TABULATE
TERMINATE
TEST
TRACE
TRANSFER
UNLINK
UNTRACE
WRITE

Table 1: List of GPSS Instructions Whose Block Symbols Can be Plotted by the Described Programme.

REFERENCES

SIMULATION USING GPSS (1974) by Thomas J. Schriber, John Wiley and Sons, Publisher.

SIMULATION WITH GPSS AND GPSS V (1976) by P.A. Bobillier, B.C. Kahan and A.R. Probst, Prentice Hall Inc., Publisher.

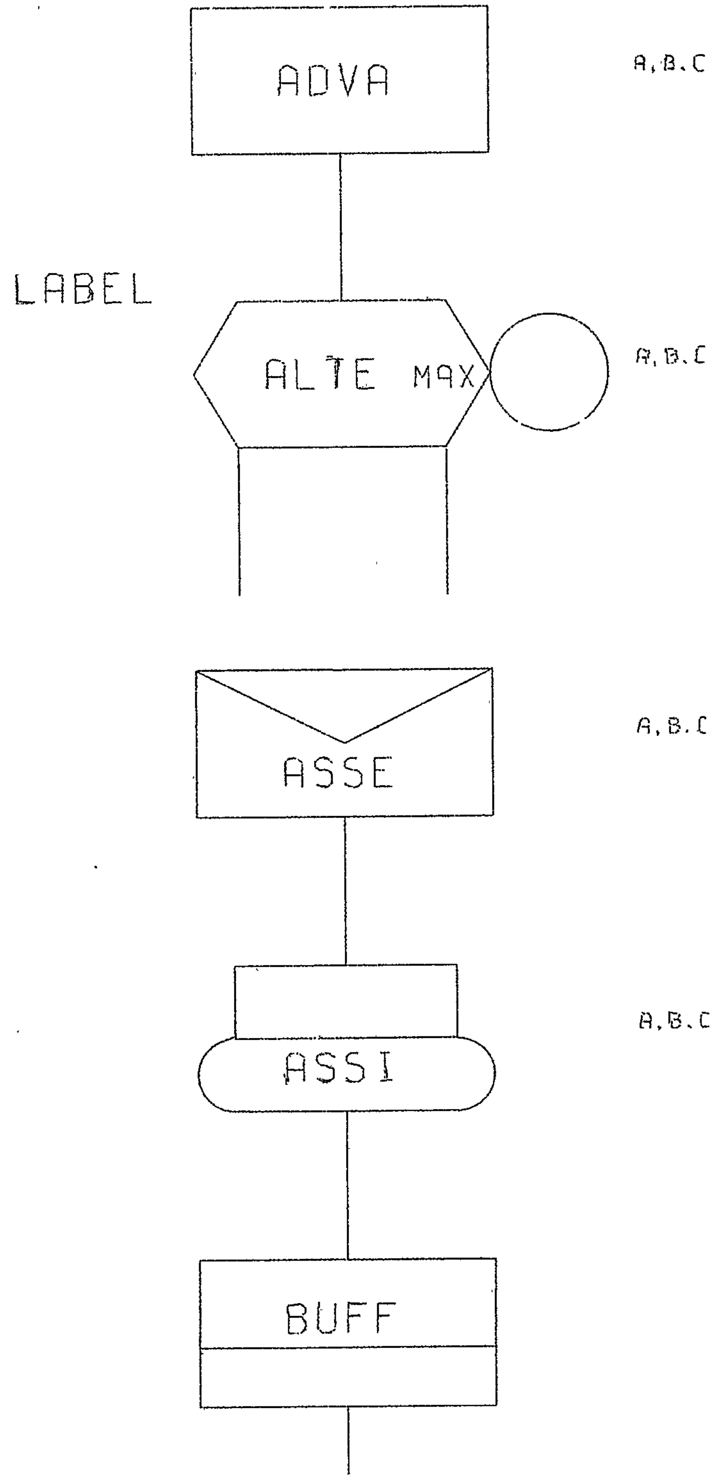

Figure 4: Sample of Plotted Output.