COMPUTER SCIENCES DEPARTMENT

University of Wisconsin-Madison

SAC-1 IMPLEMENTATION GUIDE

by

George E. Collins
and
Stuart C. Schaller

Computer Sciences Technical Report #268

January 1976

Computer Sciences Department
The University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

Received: January 22, 1976

SAC-1 IMPLEMENTATION GUIDE*

by

George E. Collins

and

Stuart C. Schaller

*This work has been supported by National Science Foundation Grant DCR74-13278

SAC-1 Implementation Guide

I. Introduction 1

II. Implementation Outline 2

III. SAC-1 Primitives 3

A. Required Primitives. 3

B. Fortran Primitives 4

C. Recommended Machine Language Routines. 4

IV. SAC-1 Subsystem Notes. 5

V. Test Programs. 8

Appendices. 10

A. SAC-1 Source Tape Format 10

B. SAC-1 Source Tape Contents 11

C. Specifications for SAC-1 Fortran Primitives. . . 12

D. SAC-1 Technical Reports. 16

I. Introduction

This report is meant to serve as a guide for the implementation of the SAC-1 System for Symbolic and Algebraic Calculation. It does not provide detailed implementation information, but rather it discusses the main tasks and points out where further information can be found.

It is recommended that the List Processing System Technical Report (LP, see Appendix D) and at least the introductions to the other SAC-1 Technical Reports be read in conjunction with this Guide.

Section II of this report outlines the tasks involved in the implementation. Section III discusses the primitives required. Section IV covers information on the interrelations of the sub-systems and Section V summarizes the test programs that are available. A series of appendices covers the SAC-1 source tape format and contents, and specifications for the SAC-1 Fortran primitives.

An index of SAC-1 algorithms, a summary of global variable specifications, and other information on using the SAC-1 System can be found in the SAC-1 Users' Guide.

II. Implementation Outline

SAC-1 is composed of a hierarchically structured set of subsystems. Each subsystem consists of subprograms written in standard (ANSI) Fortran and possibly specifications for several primitive routines which, for efficiency, are intended to be written in machine language.

The Fortran portions of each subsystem are supplied on the SAC-1 source tape whose format and detailed contents are listed in Appendices A and B. In addition the tape contains a Fortran implementation of the SAC-1 primitives and several test programs with their associated data.

The goal of the implementation procedure is to construct a library of Fortran-callable subprograms which will be loaded only as needed for user programs. This goal may be attained by proceeding as follows:

(1) Write and debug the primitive routines required by the desired SAC-1 subsystems and save the relocatable version of each routine in a subroutine library.

(2) For each subsystem compile all the Fortran subprograms and save the relocatable versions.

(3) As each subsystem is completed run the associated test program if it exists.

III. SAC-1 Primitives

A. Required Primitives

The heart of the SAC-1 system is a set of 20 primitive routines which are to be written in machine language.

The SAC-1 subsystems requiring machine language primitives are the List Processing System (LP), the Integer Arithmetic System (IA), and the Polynomial Factorization System (PF).

The List Processing System contains 15 primitives which can be divided into four groups. There are:

(1) Primitives used to insert and extract information from cells (see LP, pp. 8-9): STYPE, SCOUNT, SSUCC, ALTER, TYPE, COUNT, TAIL, and FIRST.

(2) Primitives used by BEGIN (LP, p. 9): NWPC and LOC.

(3) Primitives used for basic input and output (LP, p. 22): READ and WRITE.

(4) Primitives used for list input and output (LP, pp. 25-26): LSHIFT, RSHIFT and NBPW.

Note that (1) and (2) above require the implementer to specify the cell formats (LP, pp. 4-6).

The Integer Arithmetic System requires three primitives (see IA, pp. 7-11): ADD3, MPY and QR. The design of these primitives depends on the choice of system parameters BETA and THETA (IA, pp. 1,3,7-11). It is strongly recommended that BETA be made a power of two. The Polynomial Factorization System assumes this (PF, p. 47) as do several new subsystems currently under development.

The Polynomial Factorization System requires two primitives (see PF, pp. 13-15): AND and OR.

B. Fortran Primitives

The SAC-1 Fortran primitives are a collection of 20 routines programmed in ANSI Fortran which supplant the primitives described above. The primary intention of the Fortran primitives is to allow the implementer to test subsystems at the same time he is debugging the machine language primitives.

The Fortran primitives can be separated into three groups.

(1) Those primitives whose names correspond exactly with the names of the associated machine language primitives: ADD3, ALTER, AND, COUNT, FIRST, MPY, OR, QR, READ, SCOUNT, SSUCC, STYPE, TAIL, TYPE, and WRITE.

(2) Those SAC-1 routines which are not considered to be primitives, but which have been rewritten to be used with the Fortran primitives: BEGIN and BORROW.

(3) Routines needed to support the Fortran primitives in (1) and (2): DLREAD, DLWRIT, and INITIO.

The Fortran primitives use a fixed-size available space list and require special initialization before doing input or output. Complete specifications for using these primitives are contained in Appendix C.

Note that the SAC-1 Users' Guide assumes that the machine language versions of the SAC-1 primitives have been implemented.

C. Recommended Machine Language Routines

To improve performance once the SAC-1 System with machine language primitives is running, it is recommended that the following subprograms also be written in machine language:

Subsystem	Subprograms
List Processing	ADV, BEGIN, BORROW, DECAP, ERLA, INV, PFA, and PFL (LP, pp. 10-12)
Integer Arithmetic	COMPAT (IA, p. 11)
Modular Arithmetic	CDIF, CPROD, CRECIP, CSUM (MA, p. 5)

IV. SAC-1 Subsystem Notes

This section summarizes information on subsystem requirements and interdependency. It also draws attention to differences between the subsystem documentation in the technical reports and the subsystems as distributed on the SAC-1 source tape.

Each SAC-1 subsystem is in a separate file on the distributed tape. If the entire system is to be implemented, it is recommended that the subsystems be implemented and tested in the order in which they occur on the tape (see Appendix B).

The entire SAC-1 System need not be implemented. Figure 1 gives a schematic representation of how the subsystems are related. This diagram applies only to the subsystems as distributed on tape.

The most significant difference between the distributed and documented subsystems is that the Polynomial System (PO) as distributed does not include the subprograms PGCD, PCONT, PCGCD, and PGCDA. These routines are used to calculate polynomial greatest common divisors and are documented in the associated technical report. These routines have been superceded, however, by new algorithms implemented in the GCD and Resultant System (see GR, pp. 1-2). This means that routines PPP and PCPP of the Polynomial System cannot be used until the GCD and Resultant System has been implemented. Note that the Polynomial System test program FANDG requires neither of these routines.

In the Polynomial Factorization System (PF), subprogram PFB1 has been deleted from the distributed SAC-1 system because a change of PFZ1 has eliminated its need.

The Integer Arithmetic System (IA) subprograms IDTOH, IHTOD, IREAD, and IWRITE as distributed do not correspond to the algorithm descriptions in the associated technical report. The input and output specifications for IREAD and IWRITE are unchanged.

The only other subsystem changes were to move several subprograms from the systems in which they are documented. This was done to avoid duplication and to reduce subsystem dependencies. These modifications are summarized in Table 1.

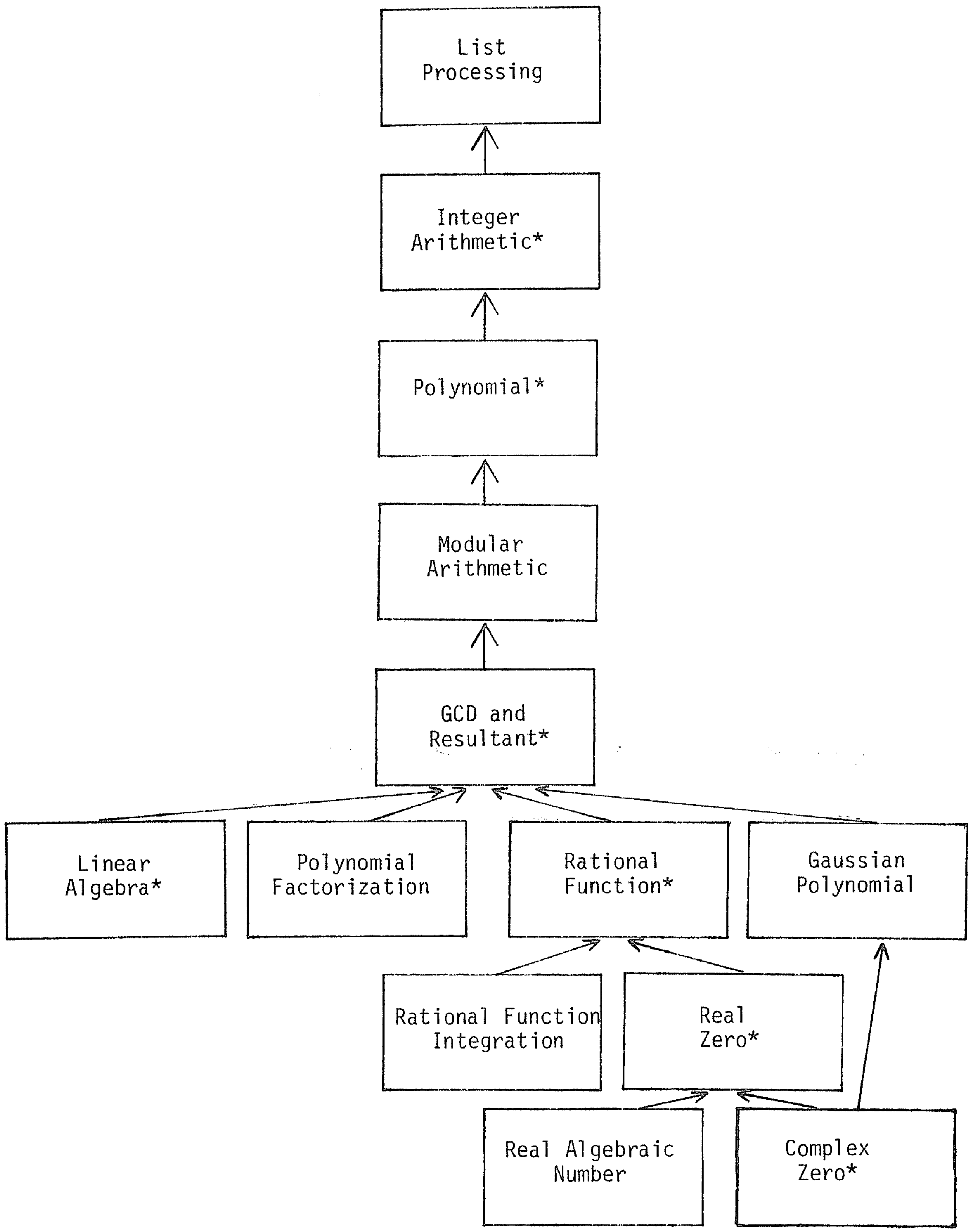


FIGURE 1: SAC-1 Subsystem Dependencies
*Test programs exist for these subsystems.

Sub-program	Subsystem on tape	Documentation
ADV2	List Processing	(CZ, p. 28), (RA, p. 31)
DECAP2	List Processing	(CZ, p. 28), (RA, p. 31)
FIRST2	List Processing	(CZ, p. 29), (RA, p. 32)
SECOND	List Processing	(CZ, p. 30)
ELPOF2	Integer Arithmetic	(RI, p. 16), (RZ, p. 34)
IGCDCF	Integer Arithmetic	(CZ, p. 31), (RA, p. 34)
PNORMF	Polynomial	(RZ, p. 35)
PPOWER	Polynomial	(RZ, p. 36)
LEXORD	GCD and Resultant	(RA, p. 32)
RNINT1	Rational Function	(CZ, p. 30), (RA, p. 38)
RNINT2	Rational Function	(CZ, p. 32), (RA, p. 38)

Table 1: Changes in Distributed SAC-1 Subsystems

The only use of non-standard Fortran occurs in subprograms which print error messages. These routines use a statement of the form PRINT n where n is a FORMAT statement number. If the local Fortran compiler will not accept the PRINT statement, these statements must be changed. A list of SAC-1 routines which produce error messages is given in Appendix A of the SAC-1 Users' Guide.

V. Test Programs

The subsystems for which test programs are available are indicated by an asterisk in Figure 1. A complete list of the test programs and the related subsystem references are given in Table 2.

The test programs are on the SAC-1 source tape as subsystem TP. The test data follow as subsystem TD.

The test programs supplied on tape do not correspond exactly to those referenced in the technical reports. The following changes to the test programs have been made where needed.

(1) The variable THETA has been added to common block TR3 in several test programs to bring them into conformance with version III of the Integer Arithmetic System.

(2) The common block TR4 has been introduced and initialized where needed to allow the use of the modular polynomial GCD algorithm implemented in the GCD and Resultant (GR) System.

(3) The test programs have been modified to use the SAC-1 Fortran Primitives. The required changes are listed in Appendix C.

(4) Minor changes include changing PRINT n to WRITE(OUT,n), removing extraneous output, and removing calls to timing routines.

Each deck of test data begins with a character code card as required by the routine INITIO of the SAC-1 Fortran Primitives. The test program output should agree with the SAC-1 technical reports with the exception of available space list lengths and the contents of the list PRIME.

To use these test programs with machine language primitives, the following changes must be made to each main program:

(1) Declare the available space array;

(2) Change the call to BEGIN to CALL BEGIN(SPACE,n) where SPACE is the name of the available space array and n is its size;

(3) Change the global variable initialization to reflect the local implementation requirements (see the SAC-1 Users' Guide);

(4) Remove the call to INITIO;

(5) Remove the character code card from each deck of test data.

Subsystem	Test Program (Subroutines)	Reference	Test Data Deck
Integer Arithmetic	IATEST	(IA, p. 55)	IATEST
Polynomial	FANDG	(PO, p. 28)	FANDG
GCD and Resultant	GRTEST	(GR, p. 86)	GRTEST
Rational Function	HILB (INVERT)	(RF, p. 7)	HILB
Rational Function	GHILB (INVERT)	(RF, p. 7)	GHILB
Real Zero	RZTEST	(RZ, p. 45)	RZTEST
Linear Algebra	LATEST (MREAD, MWRITE)	(LA, p. 67)	LATEST
Complex Zero	CZTEST	(CZ, p. 299)	CZTEST

Table 2: SAC-1 Test Programs

APPENDICES

A. SAC-1 Source Tape Format

The SAC-1 source tape is written with one of the three following sets of physical characteristics:

	1	2	3
Character Code:	BCD	EBCDIC	ASCII
Characters/Record:	84	84	81
Tracks:	7	9	9
Density (BPI):	800	800	800
Parity:	even	odd	odd

The tape is written as a series of files, each followed by a file mark. The last file is followed by two file marks.

Each file is a sequence of unblocked card images. Each card image contains a two-letter subsystem code in columns 73 and 74 and a sequence number in columns 75 through 80. The sequence numbers begin for each file with 10 and are incremented by 10.

Within files containing Fortran source images, each subprogram is proceded by a header card with the following information in columns 1 through 24:

```
Col 1       Col 13
↓           ↓
C*-*-*-*-*-*XXXXXXbbbbbb
```

where XXXXXX is the left justified subprogram name and ƀ represents a blank character.

B. SAC-1 Source Tape Contents

A complete SAC-1 source tape contains the following 18 files.

File #	Subsystem Code	Approx. # Card Images	Approx. # Sub-programs	File Contents
1	AA	1	0	One card image containing the SAC-1 version number and date.
2	BB	1	0	Character code file. One card image containing in columns 1 through 47 the SAC-1 character set in SAC-1 collating sequence.
3	TP	734	11	Test program file.
4	TD	115	8	Test data file. The various data decks are preceeded by header cards as described in Appendix A. These header cards carry the name of the associated test program.
5	FP	399	20	Fortran primitives.
6	LP	388	23	List Processing System.
7	IA	834	26	Integer Arithmetic System.
8	PO	1147	35	Polynomial System.
9	MA	1144	35	Modular Arithmetic System.
10	GR	1652	42	Polynomial GCD and Resultant System.
11	RF	495	17	Rational Function System.
12	RI	649	15	Partial Fraction Decomposition and Rational Function Integration System.
13	RZ	915	24	Polynomial Real Zero System.
14	LA	1238	33	Polynomial Linear Algebra System.
15	PF	813	21	Univariate Polynomial Factorization System.
16	GP	4337	123	Gaussian Integer and Gaussian Polynomial System.
17	CZ	2779	98	Gaussian Polynomial Complex Zero System.
18	RA	1397	53	Real Algebraic Number System.

C. Specifications for SAC-1 Fortran Promitives

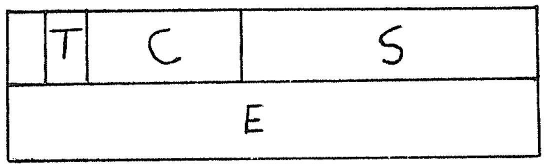
We describe in the following a SAC-1 implementation written entirely in standard (ANSI) Fortran. By changing a very few parameters, which appear as constants in the subprograms, the implementation is usable on any computer. In the following these parameters are chosen to be nearly optimal for a computer with a 32-bit word length (such as the IBM 360 and 370). This parameter choice is indeed permissible for any computer with $\gamma \geq 2^{31}$ (so every integer less than 2^{31} in absolute value is a Fortran integer), although it will not be optimal for longer word lengths. The changeable constants are associated with the widths of the count and successor fields and with the radix β in an obvious manner and will not be made explicit.

This implementation is intended to be used primarily for the initial installation of SAC-1 on a new computer and for its application to small problems. For serious use of SAC-1, more efficient assembly-language subprograms should be substituted for the eight list-processing field primitives, the three integer arithmetic primitives, and the two polynomial factorization primitives.

1. Cell Structure and List Processing Primitives

In this implementation, each cell consists of two consecutive elements of a large array called SPACE, which is in labelled common block S. In the Fortran subprograms, SPACE is declared to have 20,000 elements, but this number can be changed as desired. The location of a cell is its index in the array SPACE. Thus the various cell locations are 1,3,5,7,....

Following is a cell structure diagram:



T	=	type	(1 bit)
C	=	count	(10 bits)
S	=	successor	(20 bits)
E	=	element	(1 word)

In the first cell word, the sign bit (the leftmost bit) is unused, and will always be "+". Arithmetically, the Fortran integer stored in the first cell word is given by the mixed radix formula

(1) $N = T\cdot 2^{30} + C\cdot 2^{20} + S$

with $0 \le T < 2$, $0 \le C < 2^{10}$ and $0 \le S < 2^{20}$. Conversely, T,C and S in these ranges are uniquely determined by N, if $0 \le N < 2^{31}$. The Fortran subprograms for the eight list-processing field primitives are based directly on equation (1) and the constants

(2) $2^{10} = 1{,}024$, $2^{20} = 1{,}048{,}576$, $2^{30} = 1{,}073{,}741{,}824$.

Note that this implementation allows a maximum of $2^{19} = 524{,}288$ cells.

The primitives LOC and NWPC are used only by BEGIN. Instead of implementing these two primitives, BEGIN has been written as a primitive. Note that if the number of elements in SPACE is changed, then the two occurrences of the constant "19999" in BEGIN must also be changed accordingly.

Since the maximum possible reference count with this implementation is 1023, there is a possibility of "overflow". To prevent such overflow, a recursive algorithm for BORROW is provided. Whenever necessary to prevent count field overflow, this algorithm obtains a new cell from AVAIL, applies itself recursively to the successor and element (if a list) of the given cell, and returns as value the location of the new cell. Since count overflow will be rare, average computing times are not significantly affected.

2. Integer Arithmetic Primitives

For the integer arithmetic implementation, we choose $\beta = 2^{15} =$ 32,576, and accordingly $\theta = 10^4 = 10{,}000$. We then have $\beta^2 < \gamma$ so the Fortran programs are straightforward, using divisions and multiplications by β. In an assembly language implementation one might set $\beta = 2^{30}$, so these primitives are quite inefficient in time and space utilization, and should be the first to be re-implemented in assembly language.

3. Input-Output Primitives

The primitives READ and WRITE, written in standard Fortran, are computer-independent. They are also very efficient and could well be used permanently. However, certain restrictions and reservations need to be stated.

(a) The input deck for each SAC-1 program must begin with a special "character code card" which is punched in columns 1-47 with the 47 characters of the Fortran character set as follows:

```
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ+-*/(),.b=$
```

This card is read in by INITIO, a subroutine of READ and WRITE, to establish the correspondence between the SAC-1 internal codes and the computer internal codes. This feature can be eliminated however, if desired, by using a DATA statement in INITIO to achieve the effect of the READ statement between the statements labelled 2 and 3.

(b) In the DATA statements in READ, WRITE and INITIO, an appropriate unit number must be assigned to the variable PRINT, the desired unit number for all SAC-1 printed output.

(c) There is a very remote possibility that, on some computers, the internal machine codes for the 47 Fortran characters, considered as Fortran integers, are not all distinct. Should this occur, however, it will be detected by INITIO upon reading the character code card, and the message "incorrect code card" will be printed.

(d) READ has no provision for detecting an end-of-file since none is provided in standard Fortran. This causes no problem unless one writes a SAC-1 main program which expects an end-of-file indication from some SAC-1 input subprogram. If a computer is used whose Fortran compiler provides for end-of-file detection, READ can be easily modified to meet its end-of-file specifications.

The primitive subprograms LSHIFT, RSHIFT and NBPW are used only by LREAD and LWRITE. These three primitives should be deleted and DLREAD and DLWRIT should be substituted for LREAD and LWRITE. The specifications for DLREAD and DLWRIT are identical with those of LREAD and LWRITE except that, in the external canonical form of a list, atoms appear in decimal form (a sign followed by a sequence of decimal digits, where a plus sign or leading zeros may be omitted).

4. Using the Fortran Primitive

The following points summarize the changes required to a SAC-1 main program in order to use the SAC-1 Fortran Primitives.

(a) Remove the available space array declaration. This is not strictly necessary but will save a large amount of memory since the Fortran Primitives have a large available space array declared locally.

(b) Remove the parameters in the call to BEGIN. This is because the available space array is local to the Fortran Primitives.

(c) Insert CALL INITIO(IN) before any SAC-1 input or output is attempted. IN is the Fortran unit number of the i/o device containing the input data.

(d) Make sure that a character code card (see above) is the first card in the data deck.

(e) Initialize the global variables to values assumed by the Fortran Primitives. For the implementation discussed here $\beta = 2^{15}$ and $\theta = 10^4$. The affected global variables are as follows:

Common Block	Initialization

TR3	BETA = 2**15
	THETA = 10**4

TR4	PEXP = 13
	PRIME = GENPR(PR,500,2**13+1)
	where PR is a 500 word array.

TR8	ZETA = 15
TR9	GPRIME = GIGNPR(GPR,500,2**13+3) where GPR is a 500 word array.

D. SAC-1 Technical Reports

System	Report
(LP)	The SAC-1 List Processing System, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 129, July 1971, 34 pages.
(IA)	The SAC-1 Integer Arithmetic System - Version III, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 156, March 1973, 63 pages.
(PO)	The SAC-1 Polynomial System, by G. E. Collins, U.W. Comp. Sci. Dept. Report No. 115, March 1971, 66 pages.
(MA)	The SAC-1 Modular Arithmetic System, by G. E. Collins, L. E. Heindel, E. Horowitz, M. T. McClellan and D. R. Musser. U.W. Comp. Sci. Dept. Report No. 165, Nov. 1972, 50 pages.
(GR)	The SAC-1 Polynomial GCD and Resultant System, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 145, Feb. 1972, 94 pages.
(RF)	The SAC-1 Rational Function System, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 135, Sept. 1971, 31 pages.
(RI)	The SAC-1 Partial Fraction Decomposition and Rational Function Integration System, by G. E. Collins and E. Horowitz. U.W. Comp. Sci. Dept. Report No. 80, Feb. 1970, 47 pages.
(RZ)	The SAC-1 Polynomial Real Zero System, by G. E. Collins and L. E. Heindel. U.W. Comp. Sci. Dept. Report No. 93, Aug. 1970, 72 pages.
(LA)	The SAC-1 Polynomial Linear Algebra System, by G. E. Collins and M. T. McClellan. U.W. Comp. Sci. Dept. Report No. 154, April 1972, 107 pages.
(PF)	The SAC-1 Polynomial Factorization System, by G. E. Collins and D. R. Musser. U.W. Comp. Sci. Dept. Report No. 157, March 1972, 65 pages.
(GP)	The SAC-1 Gaussian Integer and Gaussian Polynomial System, by B. F. Caviness, G. E. Collins, H. I. Epstein, M. Rothstein, and S. C. Schaller. (In Preparation.)
(CZ)	Algebraic Algorithms for Computing the Complex Zeros of Guassian Polynomials, by J. R. Pinkert. U.W. Comp. Sci. Dept. Report No. 188, July 1973, 322 pages.
(RA)	Algorithms for Polynomials Over a Real Algebraic Number Field, by C. M. Rubald. U.W. Comp. Sci. Dept. Report No. 206, Jan. 1974, 224 pages.
(UG)	SAC-1 User's Guide, by G. E. Collins and S. C. Schaller. U.W. Comp. Sci. Dept. Report No. 269, Jan. 1976.