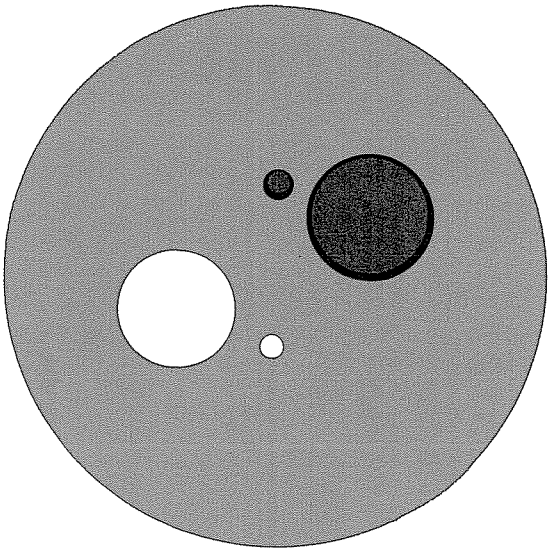
COMPUTER SCIENCES
DEPARTMENT

University of Wisconsin-
Madison



SAC-1 USER'S GUIDE

by

George E. Collins
and
Stuart C. Schaller

Computer Sciences Technical Report #269

February, 1976

The University of Wisconsin
Computer Sciences Department
1210 West Dayton Street

Received: February 9, 1976

SAC-1 USER'S GUIDE*†

by

George E. Collins
and
Stuart C. Schaller

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

†This report is the same as MACC Technical Report #43.

TABLE OF CONTENTS

I. Introduction

This document is a guide to the use of the SAC-1 system for symbolic algebraic calculation, supplementing the various technical reports which describe the individual subsystems of the SAC-1 system. The SAC-1 system is a portable, computer-independent system which is operational on many different computers, and the present document is restricted to presenting a brief overview of the current state of the entire system and providing a convenient reference guide for users of SAC-1.

Section II describes the capabilities of the various SAC-1 subsystems. Section III provides a summary of information necessary to use the system. Section IV describes the characteristics of a Univac 1110 implementation of SAC-1 at the University of Wisconsin. Several features unique to this implementation are also discussed. The appendices contain a list of error messages, an index of the SAC-1 algorithms, and a list of technical reports.

The two letter subsystem codes appearing throughout this report refer to the technical reports listed in Appendix D.

It is recommended that the user read the List Processing System technical report (LP) and at least the introductions to the reports concerning the Integer Arithmetic (IA), Polynomial (PO), and Polynomial GCD & Resultant (GR) Systems before attempting to use SAC-1.

II. SAC-1 Subsystem Summary

SAC-1 consists of a series of subsystems, each subsystem providing the system with some additional or enhanced capability. At present, 13 such subsystems have been completed, documented and released. Following are brief paragraphs summarizing the facilities and capabilities of each of the currently available subsystems.

(1) The SAC-1 List Processing System (LP). A list processing system on which all subsequent subsystems are dependent, since most of the arithmetic and algebraic objects processed by them are represented internally as lists.

(2) The SAC-1 Integer Arithmetic System - Version III (IA). This subsystem performs operations on infinite-precision integers, i.e. arbitrarily large integers represented as lists. Operations provided include the arithmetic operations, greatest common divisor calculation, and input/output.

(3) The SAC-1 Polynomial System (PO). Provides operations on polynomials in any number of variables with infinite-precision integer coefficients. Operations provided include addition, subtraction, multiplication, division, substitution and evaluation, differentiation and input/output. Greatest common divisor calculations are provided by the SAC-1 Polynomial GCD and Resultant System (see (5) below).

(4) The SAC-1 Modular Arithmetic System (MA). Provides the arithmetic operations in the finite field GF(p), where p is any odd single-precision prime number, and numerous operations on univariate or multivariate polynomials with coefficients in GF(p). Also included are programs for the Chinese remainder theorem and interpolation, for generating a list of large single-precision prime numbers, and for factorization of univariate polynomials over GF(p).

(5) The SAC-1 Polynomial GCD and Resultant System (GR). Provides programs based on very fast modular algorithms for computing the greatest common divisor or resultant of multivariate polynomials with infinite-precision integer coefficients. Also provides fast modular-algorithm programs for polynomial multiplication and division, and some other miscellaneous operations on polynomials.

(6) The SAC-1 Rational Function System (RF). Provides operations on rational functions whose numerators and denominators are multivariate polynomials with infinite-precision integer coefficients. Operations provided include addition, subtraction, multiplication and division, differentiation, substitution and input/output. Infinite-precision rational number arithmetic is included as a special case of rational function arithmetic.

(7) The SAC-1 Partial Fraction Decomposition and Rational Function Integration System (RI). Provides the partial fraction decomposition, relative to a square-free factorization, of any univariate rational function. Also, provides the indefinite integral of any univariate rational function in the form of its rational part plus the integrand of its transcendental part.

(8) The SAC-1 Polynomial Real Zero System (RZ). Computes, with guaranteed accuracy, to any specified precision, all the real zeros of any univariate polynomial with infinite-precision integer coefficients, and provides some other related capabilities.

(9) The SAC-1 Polynomial Linear Algebra System (LA). Performs operations on matrices whose elements are multivariate polynomials with infinite-precision integer coefficients. Operations include addition and multiplication, determinant calculation, inversion, null-space basis calculation, and solution of systems of linear equations. Fast modular algorithms are used.

(10) The SAC-1 Polynomial Factorization System (PF). Computes the complete factorization into irreducible polynomials (over the ring of the integers) of any univariate polynomial with infinite-precision integer coefficients. On the UNIVAC 1110, this system can

factor most polynomials with single-precision coefficients and degrees up to about 25 in less than a minute.

(11) The SAC-1 Gaussian Integer and Gaussian Polynomial System (GP). Provides operations on infinite-precision Gaussian integers (i.e. complex numbers with integer real and imaginary parts) and multivariate polynomials over the Gaussian integers. Operations provided include addition, subtraction, multiplication, division, substitution and evaluation, differentiation and input/output. This system also includes a modular algorithm for calculating greatest common divisors.

(12) The SAC-1 Complex Zero System (CZ). Computes, with guaranteed accuracy, to any specified precision, all the real and complex zeros of any univariate polynomial with infinite-precision Gaussian integer or Gaussian rational coefficients.

(13) The SAC-1 Real Algebraic Number System (RA). Provides routines to perform operations in $Q(\alpha)$ and $Q(\alpha)[x]$, where Q is the field of rational numbers, and α is any real algebraic number. Operations provided include addition, subtraction, multiplication, division, greatest common divisor calculation, and comparison with respect to the order relation of $Q(\alpha)$.

III. Using the SAC-1 System

To use the SAC-1 system the user must write a main program and possibly several subprograms which apply the required subprograms of the SAC-1 system. The local operating system is then responsible for linking together the SAC-1 routines which are needed by the user's program.

Part A of this section summarizes the mnemonic conventions used to identify SAC-1 algorithms. Part B outlines the steps necessary to write a main program. Parts C and D discuss subsystem dependencies and global variable specifications, and part E deals with common SAC-1 programming errors and how to avoid them.

III.A. Mnemonic conventions for SAC-1

The SAC-1 algorithms available to the user are listed in Appendix B by subsystem and in Appendix C alphabetically. Many of the SAC-1 algorithms have names which clearly describe what they do. For example, IPROD, PPROD, CPPROD, RPROD, GIPROD, GPPROD compute the product of two infinite-precision integers, two polynomials, two modular polynomials, two rational functions, two Gaussian integers, and two Gaussian polynomials respectively. For information on data structures and calling sequences it will be necessary to refer to the appropriate SAC-1 technical report.

The first one or two letters of an algorithm name usually indicate the algebraic structure required. For the most part of the following mnemonic conventions have been used:

Mnemonic	Meaning
A	Algebraic number
AP	Polynomial over an algebraic number field
C	Modular integer
CP	Modular (congruence) polynomial, dense representation
GI	Infinite precision Gaussian integer

Mnemonic	Meaning
GP	Gaussian polynomial
I	Infinite precision integer
M	Matrix
P	Polynomial (integer coefficients)
PM	Polynomial modulus (see RA)
R	Rational function
SGP	Modular Gaussian polynomial, sparse representation
SP	Modular polynomial, sparse representation

The remainder of an algorithm name usually indicates the operation performed. The following mnemonics have been used:

Mnemonic	Meaning
ABS	Absolute value
CONT	Polynomial content
CPP	Polynomial content and primitive part
CRA	Chinese remainder algorithm
DEG,DG	Polynomial degree
DET	Determinant
DERIV,DERV,DRV	Derivative
DIF	Difference
DIV	Divide
ERASE,ERAS,ERS	Erasure
EVAL,EVL,EV	Evaluation
GCD	Greatest common divisor

Mnemonic	Meaning
INV	Inverse
LCM	Least common multiple
LCF,LDCF,LNCF,LUCF	Leading coefficient, numeric, univariate
MOD	Modular homomorphism
MONIC,MON	Monic polynomial
MPY	Multiply
NEG	Negative
NORM,NM	Norm
NVBL,NV	Number of variables
POWER	Exponentiation
PROD	Product
PRS	Polynomial remainder sequence
Q,QR,QREM	Quotient, quotient and remainder
RECIP,RECP	Reciprocal
SIGN	Algebraic sign
SUBST,SUBS	Substitution
SUM	Sum
WRITE,WRIT,WRT	Write

III.B. Writing a Main Program

This section gives a checklist of things to include when writing a main program which uses the SAC-1 system. Examples are drawn from the SAC-1 implementation on a Univac 1110 at the University of Wisconsin.

(1) The user must declare as integers all variable names, array names, and function subprogram names. Alternatively, on the Univac 1110 with Fortran V it is possible to include the statement

```
IMPLICIT INTEGER (A-Z)
```

to accomplish the same purpose. Violations of this rule are a common cause of errors. This rule is subject to exception only if the user's program also does some floating point calculations (outside the SAC-1 system).

(2) A dimension statement must be included for the array from which the available space list will be created (see LP, pp. 4-8). The number of words in this array must be an integer multiple of the number of words in one SAC-1 cell as implemented at the user's installation.

The available space array must be initialized by calling the SAC-1 subprogram BEGIN (LP, p. 7,10) with two arguments. The first argument is the name of the available space array, the second is the number of words in the array.

A 10,000 cell available space list can be created on the Univac 1110, which has two words per cell, by the following:

```
DIMENSION SPACE(20000)
CALL BEGIN(SPACE,20000)
```

Many SAC-1 applications can be run with available space lists of 5,000 to 10,000 cells. The maximum size possible depends on the SAC-1 subprograms the user's program requires and on the central memory limitations at the user's installation. A summary of subsystem sizes on the Univac 1110 is given in section IV.F.

(3) All necessary common blocks for SAC-1 global variables must be declared and initialized in the main program. SAC-1 global

variables are contained in labeled common blocks named TR1 through TR9. The relations between the various subsystems and these common blocks are indicated schematically in Figure 1 and described in section III.C below. Detailed specifications for each common block are given in section III.D.

(4) Other arrays must be dimensioned as needed for the initialization of common blocks TR4, TR5, and TR9. See section III.D for details.

(5) It is often convenient to assign to mnemonic variables the logical unit numbers for input and printed or punched output. For example, on the Univac 1110 one might include the following:

```
INPUT = 5
PRINT = 6
PUNCH = 1
```

(6) Ensure that the program erases all lists which it generates as soon as these lists are no longer needed. This minimizes the length of the available space list required for any execution of the program.

III.C. SAC-1 Subsystem Dependencies and Common Block Usage

The hierarchical structure of the SAC-1 subsystems is shown schematically in Figure 1. The most basic subsystems are at the top, and the directions of dependency are shown by the arrows. The subsystems where common blocks containing SAC-1 global variables are first needed are indicated by the inclusion of the notation /TRn/ in the box labeled by the subsystem. There are 9 common blocks labeled TR1 through TR9.

To determine which common blocks must be declared and initialized in the main program the user must first decide which SAC-1 subsystems will be used. For each of these subsystems he

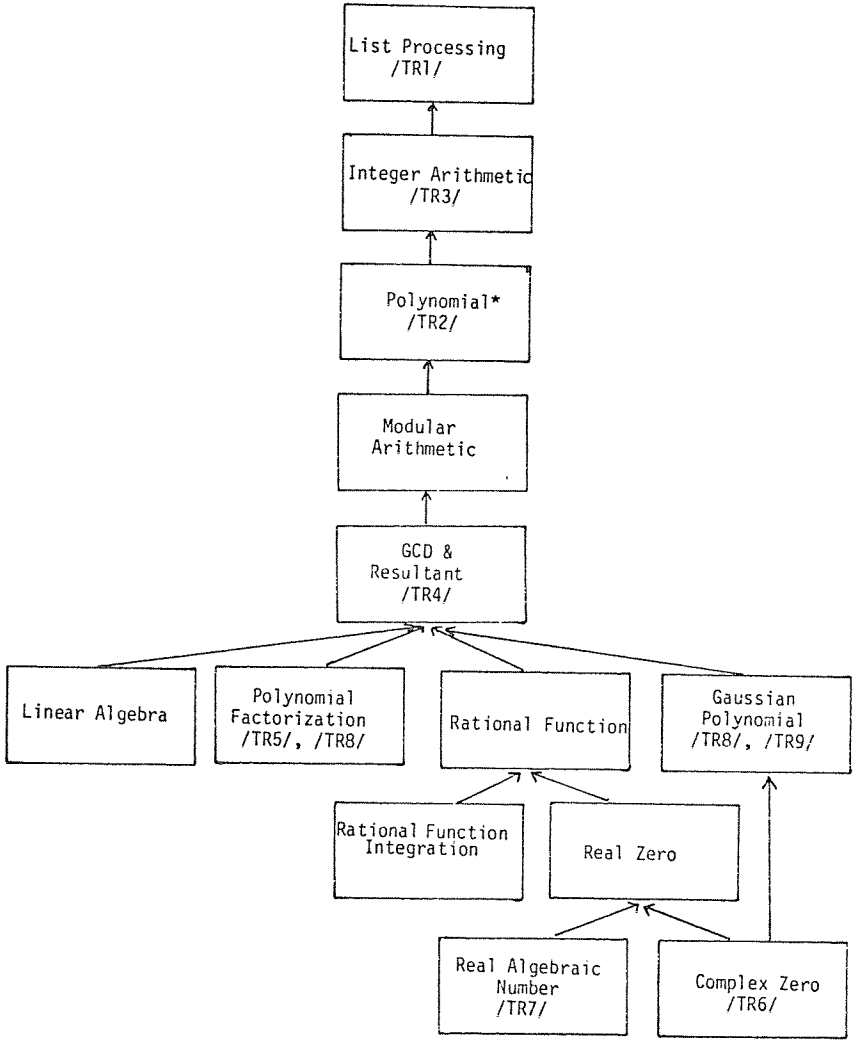


Figure 1: SAC-1 Subsystem Dependencies and Common Block Usage

*This is the Polynomial System with GCD routines omitted. See Section III.C.

follows the lines of dependency up until the List Processing system is reached. Every common block encountered must be included in his main program.

For example, if one uses the Real Algebraic Number System, one must include common blocks TR1, TR2, TR3, TR4, and TR7.

The following comments should be noted when using Figure 1.

(1) The Polynomial System indicated in Figure 1 corresponds to the system as documented in the SAC-1 Polynomial System technical report (PO) with the exception that the routines for polynomial gcd calculation, PCGCD and PGCDA, have been deleted. The routines PGCD, PCONT, PCPP, and PPP now depend on the Polynomial GCD and Resultant System (GR). Hence when using these routines, common block TR4 must be included.

(2) Common block TR4 is not explicitly mentioned in the GCD and Resultant System technical report (GR), but is required there.

(3) Common block TR8 is now required with the Polynomial Factorization System (PF) although it is not mentioned in the associated technical report.

III.D. Common Block Specifications

This section gives complete specifications for common blocks currently required in SAC-1. A summary of initializations appropriate to the Univac 1110 is given in section IV.B.

(1) Common Block TR1 (see LP, pp. 7,15).
Declaration: COMMON /TR1/ AVAIL, STACK, RECORD(72)
Definitions:

AVAIL-Location of the first cell on the available space list. It is initialized in BEGIN. Note that the function LENGTH(AVAIL) returns the current number of cells available.

STACK - Location of the system pushdown stack. It also is initialized in BEGIN.

RECORD - A 72 word I/O buffer used by the SAC-1 input and output routines.

Initialization: None required.

(2) Common Block TR2 (PO, pp. 7,12).
Declaration: COMMON /TR2/ SYMLST
Definition:

SYMLST - Location of the Polynomial System symbol list. This is a list of infinite precision integers representing the variables occurring in polynomials which have been read in.

Initialization: SYMLST = 0

(3) Common Block TR3 (IA, pp. 1,3)
Declaration: COMMON /TR3/ BETA,THETA
Definitions:

BETA - Radix for the representation of infinite-precision integers. If γ is the largest possible Fortran integer relative to a given compiler, then BETA is any power of two such that $64 \leq \text{BETA} < \gamma$. For efficient integer arithmetic, it is recommended that BETA be as large as possible. If the Integer Arithmetic primitives ADD3, MPY, and QR have not been implemented in machine language, there will be additional restrictions on BETA (IA, pp. 7-10). (The requirement that BETA be a power of two is not explicit in the SAC-1 technical reports. The Polynomial Factorization System, the Gaussian Polynomial System, and several new systems being developed, however, assume this.)

THETA - An intermediate number base used for input and output conversion of infinite-precision integers. THETA should be the largest power of 10 that is less than BETA. (THETA was introduced with Version III of the Integer Arithmetic System in July 1972. The example programs in SAC-1 technical reports issued prior to this date are incorrect because THETA, and, in some cases, the entire TR3 common block have been omitted.)

Initialization:

BETA = 2^{ζ} where $\zeta < \log_2 \gamma$,

THETA = 10^{η} where $\eta \leq \log_{10}$BETA.

On the Univac 1110, $\gamma = 2^{35}$, BETA = 2^{33}, and THETA = 10^9.

(4) Common Block TR4 (RI, pp. 1-2; RZ, pp. 1-2).
Declaration: COMMON /TR4/ PRIME, PEXP
Definitions:

PRIME - Location of a non-null list of single precision odd primes $(p_1, p_2, \ldots, p_r)$. This list can be generated by using subprogram GENPR of the Modular Arithmetic System (MA, p. 4). GENPR is a Fortran function called with three arguments: GENPR(A,k,m). A is a one dimensional array of length k and m is an odd Fortran integer, $m \geq 3$. The value of the function is the location of the ordered list of all prime numbers in the closed interval [m,m+2k-2]. Note that the primes p_i in the list PRIME must satisfy $p_i \leq$ BETA/2 (MA, pp. 2-3) and the maximal efficiency in most applications is attained with $p_i >$ BETA/4. A list of about 50 primes is adequate for most computations.

PEXP - The largest Fortran integer h such that 2^h is less than the smallest prime in the list PRIME. PEXP may be calculated by using subprogram ELPOF2 (RI, p. 16; RZ, p. 34).

Initialization:

```
DIMENSION A(k)
PRIME = GENPR(A,k,m)
MINPR = FIRST(PRIME)
CALL ELPOF2(MINPR,PEXP,DUMMY)
```

For definitions of k and m see above. On the Univac 1110, k = 500, $m = 2^{31} + 1$, and PEXP = 31. In this case GENPR generates 53 primes.

(5) Common Block TR5 (PF, p. 32).
Declaration: COMMON /TR5/ NU, SPRIME
Definitions:

NU - A positive Fortran integer which specifies the maximum number of primes for which mod p factorizations will be used in factoring a univariate polynomial over the integers.

SPRIME - The location of a list of small odd prime Fortran integers which are used for the mod p factorizations. This list should be of length $\geq$ min(2*NU,20) and can be generated with subprogram GENPR (see discussion of common block TR4).

Initialization:

For most cases the following will be sufficient:

```
DIMENSION B(20)
SPRIME = GENPR(B,20,3)
NU = 5
```

(6) Common Block TR6 (CZ, p. 299)
Declaration: COMMON /TR6/ METHOD, PRINV
Definitions:

METHOD - A Fortran integer determining the algorithm used to compute polynomial remainder sequences when isolating and refining the complex roots of polynomials. If METHOD = 0, an integer algorithm is used, and if METHOD = 1, a modular algorithm. See (CZ, p. 260-265 and 316-317) for comparisons.

PRINV - The location of a list of primes and inverses required by the modular method in the Complex Zero System. This list can be generated by using the subprogram GLPINV (CZ, p. 70) and the list of primes, PRIME, in common block TR4 (see above).

Initialization:

Either METHOD = 0
or
METHOD = 1.

If METHOD = 1 then use also

PRINV = GLPINV(PRIME).

(7) Common Block TR7 (RA, pp. 174-177).
Declaration: COMMON /TR7/ PSI, OMEGA
Definitions:

PSI - For calculations in $Q(\alpha)$ or $Q(\alpha)[x]$ (see RA, pp. 81-83) this must be the location of a primitive, square-free univariate integral polynomial with no linear factors. The real algebraic number α must be a root of PSI. Given any univariate polynomial $P(x)$ such that $P(\alpha) = 0$, PSI can be determined by using subprogram PGRRFD (RA, pp. 47-51).

OMEGA - The location of a rational interval containing exactly one root of PSI, namely α. This may also be calculated by using PGRRFD and selecting the appropriate interval from the set of rational intervals returned.

Initialization:

For example:

```
P = PREAD(5)
CALL PGRRFD(P,PSI,R,I)
OMEGA = BORROW(FIRST(I))
CALL PERASE(P)
CALL ERASE(I)
```

(8) Common Block TR8 (CZ, pp. 42, 302).
Declaration: COMMON /TR8/ ZETA, KAPPA
Definitions:

ZETA - A Fortran integer equal to the base 2 logarithm of BETA (see common block TR3 above).

KAPPA - A positive FORTRAN integer used in calculating quotients of Gaussian integers. If KAPPA is large, it is more likely that the nearest quotient will be found. Large values of KAPPA, however, increase the computing time of the quotient algorithms. KAPPA = 9 is recommended as a good compromise.

Initialization:

ZETA = $\log_2$ BETA
KAPPA = 9

On the Univac 1110 BETA = 2^{33}, hence ZETA = 33.

(9) Common Block TR9 (GP).

Declaration: COMMON /TR9/ GPRIME

Definition:

GPRIME - The location of a non-null list of single precision odd primes $(p_1,\ldots,p_r)$ such that each p_i is a prime Gaussian integer, i.e. $p_i \equiv 3 \pmod 4$. This list can be generated with the subprogram GIGNPR of the Gaussian Polynomial System (GP). GIGNPR is a Fortran function called with three arguments: GIGNPR (C,k,m). C is a one dimensional array of length k, and m is a Fortran integer, $m \geq 3$ and $m \equiv 3 \pmod 4$. The value of the function is the location of an ordered list of Gaussian prime numbers in the closed interval [m,m+4k-4]. A list of about 50 primes is adequate for most applications. The recommended magnitude of the primes, p_i, is discussed under common block TR4 above. If both common blocks TR4 and TR9 are needed, the prime list generated by GIGNPR can be used for both the variables PRIME and GPRIME.

Initialization:

```
DIMENSION C(k)
GPRIME = GIGNPR(C,k,m)
```

For definitions of k and m, see above. On the Univac 1110, k = 500 and $m = 2^{31} + 3$ are recommended. This generates a list of 43 Gaussian primes.

III.E. Common SAC-1 Programming Errors

When writing programs to use the SAC-1 system, one should beware of several common errors. These are detailed below.

A list of SAC-1 error messages is given in Appendix A.

(1) As mentioned previously, all variable names, arrays, and function subprogram names must be declared integer. If this is not done, various unexpected and unfortunate type conversions may occur, making garbage of the computations.

(2) If the global variables BETA and THETA in common block TR3 are not initialized properly, the SAC-1 input and output routines will terminate. This will often happen with no indication of what is wrong.

(3) Many of the SAC-1 input routines return a negative value if a syntax error or an input exception (end of file, etc.) is encountered. Attempts to use this value as a list location will often result in very strange errors. Note that the syntax for SAC-1 input is extremely restrictive.

(4) Many errors result from improper use of reference counts. Errors occur when one attempts to erase a list that has not been borrowed enough or when one attempts to use a list that has already been erased. Note that the list erasing routines do not set their arguments to zero. These errors usually show up a long time after they occur. The List Processing System (LP, pp. 7-12) describes the correct use of the required routines.

(5) If cells are missing from the available space list after all lists have been erased, the cause is often the use of nested functions. For example, if the statement A = PPROD(PSUM(X,Y),B) is executed, the polynomial returned by PSUM can never be erased. In this case it is necessary to assign an explicit intermediate variable:

```
T = PSUM(X,Y)
A = PPROD(T,B)
CALL PERASE(T)
```

(6) Errors often result if needed global variables are not initialized. For example, neglecting to initialize the global variable PRIME in common block TR4 will cause polynomial gcd calculations to terminate with the message 'ALGORITHM PGCDCF FAILS'. See Figure 1 to determine the necessary common blocks.

IV. SAC-1 Features at the University of Wisconsin

The SAC-1 System has been implemented at the University of Wisconsin on a Univac 1110 computer. The Univac 1110 has a word length of 36 bits. The University of Wisconsin machine has 512K words ($K=2^{10}=1024$) of core memory with a read/write cycle time of 1.5 μsec and 64K words of plated wire memory with a read cycle of .32 μsec and a write cycle of .52 μsec.

This section discusses this SAC-1 implementation and provides information on running programs in this environment. Parts A and B summarize implementation and global variable specifications. Part C presents a debugging aid while part D discusses control cards necessary to use SAC-1. Parts E, F, and G deal with memory and timing considerations on the Univac 1110.

IV.A. Univac 1110 Implementation Specifications

Each cell in the SAC-1 List Processing System on the Univac 1110 consists of two 36-bit words, or memory locations, with consecutive addresses. The location of a cell is the address of the first word.

The first word in the cell contains, from left to right, the reference count field (17 bits), the type field (1 bit), and the successor field (18 bits). The element field of a cell is the entire second word.

For the Univac 1110 implementation of the SAC-1 Integer Arithmetic System BETA is 2^{33} and THETA is 10^9. Thus ZETA is 33.

The logical unit numbers for SAC-1 input and output on the Univac 1110 are:

Card input:	5
Printed output:	6
Punched output:	1

IV.B. Global Variable Declaration and Initialization on the Univac 1110

```
C
C
C
C     INTEGER DECLARATIONS
C
      INTEGER SPACE
C
      INTEGER AVAIL,STACK,RECORD
      INTEGER SYMLST
      INTEGER BETA,THETA
      INTEGER PRIME,PEXP
      INTEGER NU,SPRIME
      INTEGER METHOD,PRINV
      INTEGER PSI,OMEGA
      INTEGER ZETA,KAPPA
      INTEGER GPRIME
C
      INTEGER PR
      INTEGER SPR
      INTEGER GPR
C
      INTEGER INPUT,PRINT,PUNCH
      INTEGER GENPR
      INTEGER GLPINV
      INTEGER PREAD,BORROW,FIRST,P,R,I
      INTEGER GIGNPR
C
C
C     AVAILABLE SPACE DECLARATION
C
      DIMENSION SPACE(20000)
C
C     COMMON BLOCK DECLARATIONS
C
      COMMON /TR1/ AVAIL,STACK,RECORD(72)
      COMMON /TR2/ SYMLST
      COMMON /TR3/ BETA,THETA
      COMMON /TR4/ PRIME,PEXP
      COMMON /TR5/ NU,SPRIME
      COMMON /TR6/ METHOD,PRINV
      COMMON /TR7/ PSI,OMEGA
      COMMON /TR8/ ZETA,KAPPA
      COMMON /TR9/ GPRIME
C
C     ARRAYS NEEDED FOR COMMON BLOCK INITIALIZATION
```

```
C
      DIMENSION PR(500)
      DIMENSION SPR(20)
      DIMENSION GPR(500)
C
C     INPUT/OUTPUT UNIT INITIALIZATION
C
      INPUT = 5
      PRINT = 6
      PUNCH = 1
C
C
C
C
C     AVAILABLE SPACE INITIALIZATION (10000 CELLS)
C
      CALL BEGIN(SPACE,20000)
C
C
C     COMMON BLOCK INITIALIZATION
C
C     TR2
      SYMLST = 0
C
C     TR3
      BETA = 2**33
      THETA = 10**9
C
C     TR4
C     (53 LARGE PRIMES GENERATED)
      PRIME = GENPR(PR,500,2**31+1)
      PEXP = 31
C
C     TR5
C     (12 SMALL PRIMES GENERATED)
      NU = 5
      SPRIME = GENPR(SPR,20,3)
C
C     TR6
      METHOD = 1
      PRINV = GLPINV(PRIME)
C
C     TR7
C     POLYNOMIAL CHARACTERIZING ALGEBRAIC NUMBER IS ON CARDS
      P = PREAD(INPUT)
      CALL PGRRFD(P,PSI,R,I)
      OMEGA = BORROW(FIRST(I))
      CALL PERASE(P)
      CALL ERASE(I)
      CALL ERASE(R)
```



```
C
C     TR8
      ZETA = 33
      KAPPA = 9
C
C     TR9
C     (43 LARGE GAUSSIAN PRIMES GENERATED)
      GPRIME = GIGNPR(GPR,500,2**31+3)
C
C
C
C     BODY OF SAC-1 PROGRAM
C
C
      END
```

IV.C. The SAC-1 Debugging System

The SAC-1 Debugging System consists of a set of machine language primitives which perform run-time checks on their arguments. When an error is encountered, such as an illegal value for a list pointer, a walk-back trace is printed and the run is terminated.

The walk-back trace consists of a message identifying the error condition followed by several lines tracing the subprogram calling sequence back to the main program.

The arguments to each subprogram in the walk-back are printed immediately after the subprogram name. The arguments are printed as integers, six arguments per line. If an argument is a list location, the current argument line is terminated and the list is printed using LWRITE. The system then prints the extent (i.e. the total number of cells) of the list and a count of the number of cells in the list with a reference count of one.

The version of LWRITE used by the Debugging System prints atoms as decimal integers instead of octal as indicated in (LP, p. 25). In addition, polynomial variable names are printed as lists of base 64 integers. These changes make the lists printed by the Debugging System fairly easy to read.

The Debugging System can be accessed by including two extra control cards with a Univac 1110 run as described in the next section. As with all error checking, one pays a penalty both in time and space. Runs using the Debugging System will take 10 to 20 percent longer to execute than those without it. Approximately 5000 extra words of random-access memory are required.

IV.D. Deck Structure and Control Cards for SAC-1 on the UNIVAC 1110

The control cards necessary to compile programs using the SAC-1 system are given below. The relocatable elements for SAC-1 system subprograms are on a read-only shared file named SAC1*LIB. The 1110 Collector will load the required SAC-1 routines if a "LIB SAC1*LIB." card is inserted in the run stream following the call to the MAP processor. A typical deck follows:

```
@RUN,/R [name],[proj#],[user#],[cost or time],[pages]/[cards]
@FOR,SIZ MAIN
     [main program]
@FOR,SIZ [subprogram name]
     [subprogram provided by user]
@MAP
 LIB SAC1*LIB.
@XQT
     [input data]
@FIN
```

The SAC-1 Debugging System is on a read-only file name SAC1*DEBUG. The debugging versions of the SAC-1 primitives will be loaded if the MAP control cards are changed to:


```
@MAP
 IN  TPF$.
 IN  SAC1*DEBUG.
 LIB SAC1*LIB.
```

The user should be aware that the cost of a complicated SAC-1 collection can easily exceed $1.00. Hence, the cost of a SAC-1 run frequently is greater than the $1.50 default limit imposed on the 1110.

IV.E. Univac 1110 Memory Requirements

The UW Univac1110 has 576K words of addressable core memory. However, an individual user is normally restricted to the use of a maximum of 64K words of core memory. Furthermore, the cost of running a job is an increasing function of the number of words used. Therefore, the SAC-1 user will ordinarily need or desire to make some estimate of the number of words to be assigned to the array from which the available space list is created.

Besides the available space list, the user's memory requirements are for programs of three types:

(a) The main program
(b) Fortran utility routines
(c) SAC-1 subprograms

The Fortran Utility routines and SAC-1 subprograms which are loaded are a function of the user's main program, but their memory requirements often can not be accurately determined except by a trial execution of the program. However, an upper bound for the SAC-1 subprograms can be obtained from Table 2, which shows, for each SAC-1 subsystem, the approximate total memory requirement for all subprograms of the subsystem.

System	Memory Required (36 bit words)
List Processing	1,300
Integer Arithmetic	2,900
Polynomial	4,100
Modular Arithmetic	4,700
GCD & Resultant	6,600
Rational Function	1,800
Rational Function Integration	3,000
Real Zero	4,300
Linear Algebra	5,000
Polynomial Factorization	3,400
Gaussian Polynomial	16,300
Complex Zero	12,600
Real Algebraic Number	6,700

Table 2: SAC-1 Subsystem Memory Requirements

IV.F. Using More than 64K words on the 1110

Although programs are normally restricted to using fewer than 64K words on the Univac 1110, it is possible to run programs with a larger memory size. This facility is of importance to SAC-1 users who wish to use extremely large available space lists.

This is accomplished as follows:

(1) Divide the available space array into separate arrays each using up to 64K words.

(2) Put each of these arrays in a separate labeled common block.

(3) For each common block make up a card in the format:

```
Col 2
    ↓
    IN name
```

where name is the common block name.

(4) These cards must then be inserted in the @MAP control cards (see IV.D) following the card 'LIB SAC1*LIB.'

(5) The common blocks so declared will be loaded contiguously immediately following the program. Thus the main program must initialize the available space list by executing CALL BEGIN(array, n) where array is the name of the array in the first common block to be declared following the 'LIB SAC1*LIB.' card, and n is the combined length of the common blocks used for the available space list.

For example, to declare an available space array of 128,000 words (64,000 cells) the following is required:

(1) In the main program

```
COMMON /DUMMY1/ SPACE1(64000)
COMMON /DUMMY2/ SPACE2(64000)
.
.
.
CALL BEGIN (SPACE1, 128000)
```

(2) The @MAP control cards must be

```
@MAP
  IN  TPF$.
  LIB SAC1*LIB.
  IN  DUMMY1
  IN  DUMMY2
```

For more information on this facility see section 7.3.5 of the Univac Fortran V Reference Manual.

IV.G. Univac 1110 Timing Information

Following are the approximate execution times of all the SAC-1 subprograms which have been programmed in 1110 assembly language. These include 16 primitive subprograms and 13 other subprograms which have been programmed in assembly language to increase the efficiency of the system. All computing times are given in microseconds and are derived from information on the execution times of individual instructions as given in the Univac 1108 System Description (1968, Appendix C) rather than from empirical observations. The Univac 1110 timings will be somewhat faster, but they are less predictable since execution times for programs residing in 1110 core memory will be slower than programs residing in plated wire memory. The times given do not include execution of the calling sequence, (a single jump instruction) which requires an additional .75 microseconds. Note also that an additional .75 microseconds will be required to store the value returned by any function subprogram.

ADD3	13.25
ADV	10.5
ALTER	6.0
BEGIN	3.875n + 11.375(n=# of cells created)
BORROW	10.5
CDIF	6.0
CPROD	17.62
CRECIP	$12.6\text{Log}_2 p$ + 9.0 average (p = modulus)
CSUM	8.5
COMPAT	12.75
COUNT	4.5
DECAP	12.75
ERLA	10.5n + 10.5 (n=# of cells returned)
FIRST	3.75
INV	8.25n + 5.25 (n = length of list)
LOC	2.25

LSHIFT	14.25
MPY	13.625
NBPW	1.5
NWPC	1.5
PFA	11.25
PFL	11.25
QR	24.375
RSHIFT	14.25
SCOUNT	9.75
SSUCC	6.0
STYPE	9.75
TAIL	3.75
TYPE	5.25

Appendix A. SAC-1 Error Messages

When unrecoverable errors are encountered within the SAC-1 system, an error message is printed and the computation is halted. The error messages are grouped here according to their common symptoms.

(1) When the available space initialized by subprogram BEGIN has been completely exhausted the following results:

Message	Source
OUT OF AVAILABLE SPACE	NOAVLS (LP)

(2) The following messages occur when the list of primes, PRIME, in common block TR4 is exhausted. (These messages may also appear if this common block was incorrectly initialized.)

Message	Source	
ALGORITHM PGCDCF FAILS	PGCDCF	(GR)
ALGORITHM PDIV FAILS	PDIV	(GR)
ALGORITHM PMPY FAILS	PMPY	(GR)
ALGORITHM PRES FAILS	PRES	(GR)
ALGORITHM PTDIV FAILS	PTDIV	(GR)
INSUFFICIENT PRIMES, CREFIN	CREFIN	(RZ)
INSUFFICIENT PRIMES, RROOTS	RROOTS	(RZ)
INSUFFICIENT PRIMES, STURM	STURM	(RZ)
LIST OF PRIMES EXHAUSTED ALGORITHM MMPY FAILS.	MMPY	(LA)
LIST OF PRIMES EXHAUSTED. ALGORITHM PDET FAILS.	PDET	(LA)
LIST OF PRIMES EXHAUSTED. ALGORITHM PLES FAILS.	PLES	(LA)
OUT OF PRIMES--CZSPRS	CZSPRS	(CZ)
ALGORITHM ANRECP FAILS	ANRECP	(RA)
ALGORITHM PSMRS1 FAILS	PSMRS2	(RA)
RSQDEC FAILS, X = -2	RSQDEC	(RI)

(3) The following messages occur when the primes on the list PRIME in common block TR4 are not large enough.

Message	Source	
ALGORITHM CGCDCF FAILS	CGCDCF	(GR)
ALGORITHM CPRES FAILS	CPRES	(GR)
ALGORITHM CPTDIV FAILS	CPTDIV	(GR)
ELEMENTS OF GF(P) EXHAUSTED. ALGORITHM CPDET FAILS.	CPDET	(LA)
ELEMENTS OF GF(P) EXHAUSTED. ALGORITHM CPRRE FAILS.	CPRRE	(LA)
ELEMENTS OF GF(P) EXHAUSTED. ALGORITHM MCPMPY FAILS.	MCPMPY	(LA)
ALGORITHM CSVRS1 FAILS	CSVRS2	(RA)

(4) The following miscellaneous messages are each followed by their cause:

Message	Source	
RSQDEC FAILS, X = -1	RSQDEC	(RI)

A singular matrix was found during generation of the partial fraction decomposition of a rational function.

Message	Source	
SPRIME LIST EXHAUSTED IN PFZ1	PFZ1	(PF)

The list of small primes, SPRIME, in common block TR5 was exhausted during factorization of a univariate polynomial.

Message	Source	
LIST OF GAUSSIAN PRIMES EXHAUSTED IN GPGCDC	GPGCDC	(GP)

The modular Gaussian polynomial gcd algorithm failed because there were not enought Gaussian primes in the list GPRIME of common block TR9.

Message	Source	
ELEMENTS OF GF(P) EXHAUSTED IN SGPGCD	SGPGCD	(GP)

The Gaussian primes in GPRIME were not large enough.

***** LIST PROCESSING SYSTEM (LP) *****

***** INTEGER ARITHMETIC SYSTEM (IA) *****

***** POLYNOMIAL SYSTEM (PO) *****

***** MODULAR ARITHMETIC SYSTEM (MA) *****

***** POLYNOMIAL GCD AND *****
***** RESULTANT SYSTEM (GR) *****

***** RATIONAL FUNCTION SYSTEM (RF) *****

***** PARTIAL FRACTION DECOMPOSITION AND ****
***** RATIONAL FUNCTION INTEGRATION ****
***** SYSTEM (RI) ****

ELECF2	RI	16
LCP	RI	17
MATSED	RI	18
MATX	RI	19
MUSSLE	RI	20
PCDEC	RI	22
PINT6	RI	23
PGREM	RI	24
PSEREE	RI	25
PVECT	RI	26
RDEC	RI	27
RINT6	RI	28
RINTGS	RI	29
RSCDEC	RI	30

***** POLYNOMIAL REAL ZERO SYSTEM (RZ) *****

ALGORITHM	TECH REPT	PAGE
ANALR	RZ	11
IRIS	RZ	11
ISOLT	RZ	12
ISTURM	RZ	13
IVAR	RZ	13
NRTS	RZ	14
REFINE	RZ	14
REVPL	RZ	15
RRTS	RZ	16
SCALER	RZ	18
SNEND	RZ	19
SREFIN	RZ	20
SROOTS	RZ	22
ISOLAT	RZ	22
NROOTS	RZ	23
XROOTS	RZ	23
SEVAL	RZ	27
STURM	RZ	28
VAR	RZ	32
ASYMNS	RZ	33
ELECF2	RZ	34
IPRINT	RZ	34
PNSIMP	RZ	35
PPGALR	RZ	36
PSTURD	RZ	37
XCOPY	RZ	37
SCFREE	RZ	38

***** POLYNOMIAL LINEAR ALGEBRA SYSTEM (LA) ****

ALGORITHM	TECH REPT	PAGE
MSUB	LA	12
MDET	LA	13
MTRAN	LA	14
MPROD	LA	15
MZERO	LA	16

MVLIST	LA	17
MERASE	LA	18
MPOD	LA	19
MGARN	LA	20
MCENV	LA	22
MCPERS	LA	23
MCPEVL	LA	24
MCPINT	LA	25
MPPY	LA	26
MMCG	LA	28
MCPMPY	LA	30
MCPMDB	LA	32
MCMPY	LA	33
MCOPP	LA	34
MZCON	LA	35
ME=	LA	36
MCPEQ	LA	37
NULCON	LA	38
DBRRE	LA	39
PLES	LA	40
CPRRE	LA	43
CRRE	LA	46
PDET	LA	49
CBDET	LA	51
CPDET	LA	52
CDET	LA	54
MINV	LA	56
NULSP	LA	58

***** POLYNOMIAL FACTORIZATION SYSTEM (PF) ****

ALGORITHM	TECH REPT	PAGE
LAST	PF	6
PTECF	PF	6
MIPOD	PF	9
MPPOD	PF	9
MFLPR	PF	9
MRECIP	PF	10
PPGREM	PF	10
MPSPEQ	PF	10
AND	PF	13
OR	PF	13
IAND	PF	15
IOR	PF	15
ILS	PF	15
MEMBER	PF	16
SUBSET	PF	16
GCR	PF	17
SELECT	PF	19
PSFREE	PF	23
PFH1	PF	24
PFC1	PF	26
PFP1	PF	27
PFZ1	PF	30
PFACT1	PF	36

***** GAUSSIAN INTEGER AND *****
***** GAUSSIAN POLYNOMIAL SYSTEM (GP) *****

ALGORITHM	TECH REPT	PAGE
ADVZ	GP	
APAIR	GP	
CGIPGD	GP	
CGIPRD	GP	
CGINEC	GP	
CPLDCF	GP	
DECAP2	GP	
FIRST2	GP	
GIDITS	GP	
GICONJ	GP	
GILIF	GP	
GIPPAS	GP	
GIPWUP	GP	
GIPWW	GP	
GIQCD	GP	
GIQCDS	GP	
GIQAPR	GP	
GIRAKE	GP	
GIRCRM	GP	
GIRMRS	GP	
GIPPOD	GP	
GIW	GP	
GIWP	GP	
GIWLS	GP	
GIWEAD	GP	
GIWEM	GP	
GIWND	GP	
GISQM	GP	
GITRUN	GP	
GIUMK	GP	
GIWNIT	GP	
GIWNTS	GP	
GPCONT	GP	
GPDFP	GP	
GPDLG	GP	
GPDLRV	GP	
GPDIF	GP	
GPIPAS	GP	
GPMOV	GP	
GPQCD	GP	
GPQCDC	GP	
GPQID	GP	
GPQIDP	GP	
GPLDCF	GP	
GPLNCF	GP	
GPMDSQ	GP	
GPMVRM	GP	
GPMPNV	GP	
GPNCNT	GP	
GPNORM	GP	

GPNVBL	GP
GPONE	GP
GPORDR	GP
GPPF	GP
GPPROD	GP
GPW	GP
GPRED	GP
GPSPRD	GP
GPSQ	GP
GPSUBS	GP
GPSUM	GP
GPUM	GP
GPVEL	GP
GPVLST	GP
IPITS	GP
IRGCND	GP
ILRGD	GP
ITRUNC	GP
LEXGRD	GP
LPAIR	GP
PINFNM	GP
PNVEL	GP
SECOND	GP
SGPCPP	GP
SGPCRA	GP
SGPDEG	GP
SGPDG1	GP
SGPDGS	GP
SGPDIF	GP
SGPERS	GP
SGPLCD	GP
SGPQD1	GP
SGPINT	GP
SGPLC	GP
SGPLNC	GP
SGPLUC	GP
SGPMOD	GP
SGPMUN	GP
SGPONE	GP
SGPPRD	GP
SGPW	GP
SGPWNV	GP
SGPPED	GP
SGPREM	GP
SGPSPD	GP
SGPSPR	GP
SGPUCT	GP
SGPUEV	GP
SGPUP	GP
SGPUW	GP
SMONIC	GP
SPCRA	GP
SPDEG1	GP
SPDEG	GP
SPDEGS	GP
SPDIF	GP

SPERAS	GP
SPEVAL	GP
SPGCD1	GP
SPINTI	GP
SPLACF	GP
SPMOD	GP
SPPFYI	GP
SPSEG	GP
SPSV	GP
SPSXE	GP
SPFROD	GP
SP$DE	GP
SP$FE*	GP
SP$	GP
SPXEM	GP
SPSUM	GP
SPUCNT	GP
SPUEPP	GP
SPUDIV	GP
SPUEV	GP
SPUPR	GP
SSPROD	GP
WRITCH	GP

***** GAUSSIAN POLYNOMIAL *****
***** COMPLEX ZERO SYSTEM (CZ) *****

ALGORITHM	TECH REPT	PAGE
CZFLIP	CZ	25
CZINAM	CZ	26
ADDC	CZ	28
DECAP2	CZ	29
FIRST2	CZ	29
SECOND	CZ	30
INIT1	CZ	30
INTCCF	CZ	31
INIT2	CZ	32
MULC	CZ	33
MULR	CZ	33
RSAVE9	CZ	34
CZREAD	CZ	35
ELOG2	CZ	41
GPINCF	CZ	44
PALTLX	CZ	46
PISUM	CZ	47
PMLMP	CZ	48
PTCS	CZ	49
PSEF	CZ	50
PORD	CZ	51
PWRAZR	CZ	52
GPCONS	CZ	52
GPSYMP	CZ	54
PGSFD	CZ	57
GPGSFD	CZ	58
GLPINV	CZ	70

RTPR	CZ	72
SMRR	CZ	74
MRTI	CZ	76
CZAPRS	CZ	93
CZCPRS	CZ	98
CZSPRS	CZ	101
CZCLS	CZ	109
CZAMRS	CZ	112
CZRVSS	CZ	117
CZNRNA	CZ	119
CZANPA	CZ	120
CZAERA	CZ	123
CZNZHP	CZ	125
PSPLT	CZ	133
GPSPLT	CZ	134
PHALX	CZ	136
GPHALX	CZ	137
PSGR	CZ	138
GPSGR	CZ	140
PRSGR	CZ	142
GPRSGR	CZ	144
PHOML	CZ	147
GPHOME	CZ	149
PHOMC	CZ	150
GPHOMC	CZ	152
GPDNOM	CZ	153
GPROT	CZ	155
PTRAN	CZ	157
GPTRAN	CZ	159
GPRTR	CZ	161
GPLRTR	CZ	163
GPHPTC	CZ	166
CZNZVH	CZ	169
CZNZHH	CZ	172
PNRRL	CZ	173
PSRVL	CZ	174
GPRLHL	CZ	175
GPRRVL	CZ	176
GPRRLR	CZ	177
CZRLA	CZ	179
CZRAGA	CZ	183
GPRRMA	CZ	185
CZRSU	CZ	189
CZRASP	CZ	190
GPRRAR	CZ	191
GPRRIC	CZ	195
GPREND	CZ	198
GPRIR	CZ	210
PRIR	CZ	221
CZREF	CZ	235
CZREFA	CZ	241
CZRISA	CZ	248
CZMRSA	CZ	253
PSQFF	CZ	256
GPSQFF	CZ	258
PMRIR	CZ	261

***** REAL ALGEBRAIC NUMBER SYSTEM (RA) *****

	ALGORITHM	TECH REPT	PAGE
1	AAES	RA	96
2	ADD2	IA	7
3	ADIF	RA	94
4	ADJCOL	RI	14
5	ADV	LP	10
6	ADV2	CZ	28
7	ADV2	GP	0
8	ADV2	RA	31
9	ALTER	LP	9
10	ANALR	RZ	11
11	ANALRR	RZ	33
12	AND	PF	13
13	ANEG	RA	94
14	APRECP	RA	113
15	APAIR	GP	0
16	APGCD	RA	164
17	APMECD	RA	169
18	APMCNA	RA	162
19	APOREM	RA	117
20	APREM	RA	120
21	APROD	RA	97
22	AQ	RA	120
23	ARTF	RA	85
24	ASIGN	RA	88
25	ASPROD	RA	98
26	ASQ	RA	121
27	ASUM	RA	93
28	AZTEST	RA	84
29	BEGIN	LP	10
30	BOMROW	LP	10
31	CANALR	RZ	18
32	CPDET	LA	51
33	CDIA	GR	36
34	CDET	LA	54
35	CDIF	MA	5
36	CDMM	MA	16
37	CDLDCF	GR	12
38	CGIMOD	GP	0
39	CGIPRD	GP	0
40	CGINEC	GP	0
41	CIEVR	RA	159
42	CINV	LP	11
43	CMPLEL	MA	23
44	CMSB	MA	15
45	CMSNIC	MA	9
46	CMPROD	MA	25
47	CNERS	RZ	19
48	COMPAT	IA	11
49	CONC	LP	11
50	COUNT	LP	9
51	CPLERL	MA	20
52	CPLG	MA	21
53	CPLG	MA	22
54	CPCOMP	GR	38
55	CPCRA	GR	29

	ALGORITHM	TECH REPT	PAGE
56	CPDDF	MA	20
57	CPDEG	GR	35
58	CPDEGS	GR	34
59	CPDEG1	GR	36
60	CPDET	LA	52
61	CPDIF	MA	7
62	CPDIV	GR	23
63	CPDRV	MA	10
64	CPEGCD	MA	12
65	CPERAS	MA	13
66	CPEVAL	MA	17
67	CPLARN	MA	16
68	CPGCD1	MA	10
69	CPINS	MA	26
70	CPINT	MA	18
71	CPINTI	GR	31
72	CPLDCF	GP	0
73	CPLNCF	GR	36
74	CPLUCF	GR	36
75	CPMOD	MA	15
76	CPMON	GR	39
77	CPMPY	GR	22
78	CPMPYI	GR	27
79	CPNEG	MA	7
80	CPNV	GR	40
81	CPONE	GR	40
82	CPOWER	MA	8
83	CPPROD	MA	8
84	CPQ	GR	25
85	CPQDB	GR	26
86	CPQREM	MA	11
87	CPREAD	MA	14
88	CPREM	MA	10
89	CPRES	GR	16
90	CPRES1	GR	17
91	CPRGD	MA	5
92	CPRRL	LA	43
93	CPSUM	MA	6
94	CPTDIV	GR	24
95	CPTOM	MA	22
96	CPTRCF	CZ	44
97	CPUCNT	GR	13
98	CPUCPP	GR	14
99	CPUDIV	GR	26
100	CPUIV	GR	31
101	CPUPR	GR	27
102	CPWRIT	MA	14
103	CRECIP	MA	5
104	CREFIN	RZ	20
105	CRESCP	RA	108
106	CRRE	LA	46
107	CSPROD	MA	8
108	CSUBS2	RA	149
109	CSUM	MA	5
110	CSVRS2	RA	153

	ALGORITHM	TECH REPT	PAGE
221	GPXPNV	GP	0
222	GPXPIP	CZ	263
223	GPXCAT	GP	0
224	GPXDRM	GP	0
225	GPXEAR	CZ	191
226	GPXPER	CZ	177
227	GPXAME	CZ	175
228	GPXPIC	CZ	195
229	GPXPGA	CZ	185
230	GPXNVL	CZ	176
231	GPXVEL	GP	0
232	GPXZE	GP	0
233	GPUPDP	GP	0
234	GPXP	GP	0
235	GPXROD	GP	0
236	GPX	GP	0
237	GPXZAD	CZ	196
238	GPXID	GP	0
239	GPXROM	CZ	153
240	GPXIG	CZ	210
241	GPXOT	CZ	155
242	GPXOUR	CZ	144
243	GPXPLT	CZ	134
244	GPZERO	GP	0
245	GPSX	GP	0
246	GPUOFF	CZ	256
247	GPUUR	CZ	140
248	GPSUMS	GP	0
249	GPSUM	GP	0
250	GPSYMP	CZ	54
251	GPTRAN	CZ	159
252	GPUX	GP	0
253	GPVEL	GP	0
254	GPVEST	GP	0
255	IARSE	IA	13
256	IASD	PF	15
257	IBITS	GP	0
258	IBTCD	IA	35
259	IBTCM	IA	34
260	ICOMP	IA	14
261	IDIF	IA	17
262	IDTCM	IA	32
263	IDTCM	IA	30
264	IFACT	GR	41
265	IGCD	IA	25
266	IGCDCF	CZ	31
267	IHCDCF	RA	34
268	IHTCD	IA	31
269	IHTCD	IA	35
270	ILCP	IA	30
271	ILHCD	GP	0
272	ILS	PF	15
273	IMADD	IA	18
274	IMFI	GR	41
275	INEG	IA	13

	ALGORITHM	TECH REPT	PAGE
276	INV	LP	12
277	IOR	PF	15
278	IPOWER	IA	20
279	IPRINT	RZ	34
280	IPROD	IA	19
281	IQ	IA	24
282	IQR	IA	21
283	IQRS	IA	21
284	IREAD	IA	32
285	IRLN	IA	25
286	IRNSPR	RA	35
287	IROOTS	RZ	22
288	IROUND	GP	0
289	IRTS	RZ	11
290	ISIGNL	IA	12
291	ISOLAT	RZ	22
292	ISULT	RZ	12
293	ISTURM	RZ	13
294	ISUM	IA	15
295	ITOS	PO	21
296	ITRUNC	GP	0
297	IVAF	RZ	13
298	IWRITE	IA	36
299	LAST	PF	0
300	LCM	RI	17
301	LENGTH	LP	12
302	LEXCRD	GP	0
303	LEXGRD	RA	32
304	LOC	LP	9
305	LPAIR	GP	0
306	LREAD	LP	24
307	LSHIFT	LP	25
308	LWRITE	LP	25
309	MATSED	RI	16
310	MATX	RI	19
311	MCMFY	LA	33
312	MCMEQ	LA	37
313	MCMERS	LA	23
314	MCMIVL	LA	24
315	MCMINT	LA	25
316	MCMMDP	LA	32
317	MCMMPY	LA	30
318	MCMNV	LA	22
319	MDIF	LA	13
320	MEMBER	PF	16
321	MEN	LA	36
322	MERASE	LA	16
323	MGAEN	LA	20
324	MIMOD	PF	9
325	MINV	LA	56
326	MMCB	LA	28
327	MMOD	LA	19
328	MMPY	LA	26
329	MPLPR	PF	9
330	MPMOD	PF	9

	ALGORITHM	TECH REPT	PAGE
441	PSHEMG	RA	53
442	PSUEST	PU	17
443	PSUM	PO	21
444	PTCS	CZ	49
445	PTDIV	GR	20
446	PTLCF	PF	6
447	PTMAN	CZ	157
448	PVEL	PO	19
449	PVECT	RI	26
450	PVLIST	PO	18
451	PWHAZR	CZ	52
452	PWRITE	PO	10
453	PWRITS	PO	11
454	QR	IA	10
455	RECEND	PZ	37
456	RCOMP	RZ	37
457	RDEC	RI	27
458	RDFFIV	RF	16
459	RDIF	RF	15
460	READ	LP	22
461	REFINE	RZ	14
462	RERASE	RF	6
463	REVAL	RZ	15
464	RINTG	RI	28
465	RINTGS	RI	29
466	RINV	RF	16
467	RNALS	RA	36
468	RNALSF	CZ	34
469	RNCOMP	RA	43
470	RNLIF	RA	45
471	RNLG	CZ	33
472	RNORAS	RA	37
473	RNFLOR	RA	37
474	RNINT1	CZ	30
475	RNINT1	RA	38
476	RNINT2	CZ	32
477	RNINT2	RA	38
478	RNISPR	RA	39
479	RNLG	RA	40
480	RNPROD	RA	40
481	RNS	RA	42
482	RNRECP	RA	41
483	RNSICR	RA	42
484	RNSUB	RA	44
485	RNUP	CZ	33
486	RORBLP	RF	6
487	RPOLY	RF	7
488	RPOLY2	RF	7
489	RPROD	RF	15
490	RPSUB	RF	17
491	RQ	RF	16
492	RREAD	RF	4
493	RROOTS	RZ	23
494	RRTS	RZ	16
495	RSHIFT	LP	26

	ALGORITHM	TECH REPT	PAGE
496	RSWDEC	RI	30
497	RSUEST	RF	17
498	RSUP	RF	15
499	RTMR	CZ	72
500	RVLIST	RF	6
501	RWRITE	RF	5
502	SCOUNT	LP	9
503	SECOND	CZ	30
504	SECOND	GP	0
505	SELECT	PF	19
506	SEVAL	RZ	27
507	SGFERR	GP	0
508	SGFCRA	GP	0
509	SGMDEG	GP	0
510	SGFBGS	GP	0
511	SGFBG1	GP	0
512	SGFDIF	GP	0
513	SGFERS	GP	0
514	SGFGCD	GP	0
515	SGFGD1	GP	0
516	SGFINT	GP	0
517	SGFLC	GP	0
518	SGFLNC	GP	0
519	SGFLOC	GP	0
520	SGFMOD	GP	0
521	SGFMON	GP	0
522	SGFONE	GP	0
523	SGFPRD	GP	0
524	SGPG	GP	0
525	SGPGNV	GP	0
526	SGPRLD	GP	0
527	SGPREM	GP	0
528	SGPSPD	GP	0
529	SGPSPR	GP	0
530	SGPUCT	GP	0
531	SGPULV	GP	0
532	SGPUP	GP	0
533	SGPUW	GP	0
534	SMCNIC	GP	0
535	SMRR	CZ	74
536	SPCRA	GP	0
537	SPEEG	GP	0
538	SPEEGS	GP	0
539	SPEEGT	GP	0
540	SPDIF	GP	0
541	SPERAS	GP	0
542	SPEVAL	GP	0
543	SPGCD1	GP	0
544	SPINTI	GP	0
545	SPLNCF	GP	0
546	SPMOD	GP	0
547	SPMPYI	GP	0
548	SPNEG	GP	0
549	SPNV	GP	0
550	SPONE	GP	0

	ALGORITHM	TECH REPT	PAGE
551	SPPROD	GP	0
552	SPS	GP	0
553	SPSDB	GP	0
554	SPSRLP	GP	0
555	SPSTM	GP	0
556	SPSUM	GP	0
557	SPUCNT	GP	0
558	SPUCPP	GP	0
559	SPUDIV	GP	0
560	SPUEV	GP	0
561	SPUPR	GP	0
562	SQFREE	RZ	38
563	SSPROD	GP	0
564	SSUEC	LP	9
565	STACK2	LP	13
566	STACK3	LP	13
567	STO1	PO	21
568	STURM	RZ	28
569	STYPE	LP	9
570	SUBSET	PF	16
571	TAIL	LP	9
572	TYPE	LP	9
573	UNSTK2	LP	13
574	UNSTK3	LP	13
575	VAR	RZ	32
576	VCOMP	LA	34
577	WRITCH	GP	0
578	WRITE	LP	22

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.