Fermi National Accelerator Laboratory

TM-1650

User's Guide to ESME v. 7.1

S. Stahl and J. MacLachlan
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Illinois 60510

February 26, 1990

Operated by Universities Research Association Inc. under contract with the United States Department of Energy

User's Guide to ESME v. 7.1

S. Stahl J. MacLachlan

February 26, 1990

Abstract

ESME is a computer program to calculate the evolution of a distribution of particles in energy and azimuth as it is acted upon by the radiofrequency system of a proton synchrotron. It provides for the modeling of multiple rf systems, feedback control, spacecharge, and many of the effects of longitudinal coupling impedance. The capabilities of the program are described, and the requirements for input data are specified in sufficient detail to permit significant calculations by an uninitiated user. The program is currently at version 7.1 and extensively modified since the previous user documentation. Fundamental enhancements make version 6 data unuseable, but nearly all facilities of the earlier version have been retained and input data is similar. Also described is a VAX-based code management convention which has been established with a view to maintaining functional equivalence in versions used on different computers.

Contents

Chapter 1

Introduction

The program ESME has been developed to model those aspects of beam behavior in a proton synchrotron that are governed by the radiofrequency systems. It follows the evolution of a distribution in energy-azimuth coordinates turn-by-turn by iterating a map corresponding to the single-particle equations of motion. The map parameters may be updated each turn to reflect the action of the beam current on the individual particles through feedback loops, spacecharge, coupling impedance, *etc.* The code was initially developed during the years 1981–82 for the design of the Tevatron I Antiproton Source[1] and documented for general use in 1984.[4] The documentation corresponds to the last major revision of the code developed on the Cyber machines which was called ESME v. N. In 1986 provisions were made for longitudinal coupling and spacecharge[5] to investigate the usefulness of a γ_t-jump in the Fermilab Booster.[2] This version was adapted from the Cyber machines to the FPS-164 by Peter Lucas who made in the process a number of the improvements which entered into what was designated as v. 6, the version in most common use at Fermilab. A somewhat reworked VAX version, v. 6.05, was sent to Brookhaven.

The improvements made through 1987 were more or less incremental changes to the 1981 code. However, for several years there has been interest in capabilities for simultaneous operation of more than two independent rf systems, for explicit control over the separation between the accelerator reference orbit and the synchronous trajectory, and for input using time units rather than beam turn number. Because of this interest and because the manner of defining and controlling the various rf phase angles has proved confusing to some users, a major revision of the code was started in 1987. Although nearly all functions have been retained and data requirements are similar, the new code will not work with data prepared for earlier versions. The conceptual basis for the new version is discussed at length elsewhere.[10] This user documentation is being prepared in parallel with cleanup and validation tests. Thus, the new version, dubbed v. 7.1, will probably be less stable and more subject to bugs than v. 6, at least for the time it takes to accumulate experience from a reasonable variety of applications. Nonetheless, the new version is fundamentally improved, and there is no intention to maintain both.

ESME has been used frequently to assess the efficiency of a given rf beam manipulation or to optimize system parameters. In this mode the user needs to specify fully technical details of various subsystems and derive various numerical measures of system performance from the particle distribution. Thus, many data are required, and the program must include numerical analysis features. Equally useful, however, are qualitative calculations designed to illustrate a concept or

[1] Instructive examples of the capabilities of ESME can be found in reports relating to TeV I. Some of these will not be cited directly in the text but are included in the references. See for example refs. [1], [2], and [3].

[2] There are examples of the calculation of collective effects resulting from the Booster studies. See refs. [6], [7], [8], and [9].

explore the feasibility of a novel approach. For such use the code should require a minimum of system-specific data and provide easy access to a variety of graphical output. When a qualitative investigation has been fruitful it is natural to proceed in steps of greater realism and specificity to a thorough modeling of the process. ESME is intended to serve effectively over a wide range of problem specificity by separating distinct functions so that each is invoked only as needed and by establishing reasonable defaults so that generic systems can be represented by a few data. Thus, a few lines of data may serve to get a first look at a system which can be studied in greater detail by overriding defaults with specific input and by invoking additional functions like, for example, those related to collective behavior or those related to numerical evaluation of the properties of the distribution.

1.1 Coordinate system

The basic coordinates internal to ESME are the azimuth of the particles and the difference of their energy from the synchronous energy at the time the radiofrequency is passing through its synchronous phase. The azimuth is measured positive and negative with respect to a location where the rf voltage is applied:

$$-180^\circ \leq \Theta_{i,n} \leq 180^\circ \ ,$$

where i is the particle index and n is the turn number. The sense of Θ is clockwise positive, but the sense of the beam circulation is in the $-\Theta$ direction, counterclockwise. Typically a well-behaved bunch will be centered near $\Theta = 0 \bmod 180/h$. When there are h equivalent bunches it is generally efficient to calculate for a single bunch with periodic boundaries set at $\Theta = \pm 180^\circ/h$. The unit for energy is MeV; the unit for azimuth is radian internally but degree for input and output. The basic time unit of the difference equations is the turn number, but all input specification of parameter time dependence is for time in seconds.

One of the more difficult matters in using a general program for a wide range of problems is being able to specify the proper phases for all of the rf systems and the desired dependence of them on time or bunch centroid position. In v. 7.1 of ESME, where there may be systems with several different harmonic numbers all running at the same time, the phases of all systems are defined as their absolute phase at the synchronous time, *i.e.*, the time when a particle acted upon by all of the rf systems receives the energy matched to the specified energy and/or radial position change for the turn. Options are selected to indicate whether phases are to be controlled by an input program, to follow $\dot{B}$, to follow a radial offset program, to maintain a system as Landau cavity, *etc.* Many common options have been provided; the program structure easily accommodates other schemes which one is willing to specify in a FORTRAN subroutine.

1.2 Difference equations

The basis of the program is the pair of single particle difference equations

$$\begin{aligned} \vartheta_{i,n} &= \left[\frac{\tau_{s,n-1}}{\tau_{s,n}} \vartheta_{i,n-1} + 2\pi \frac{\tau_{i,n}}{\tau_{s,n}} - \pi \right]_{mod(2\pi)} + \pi \\ E_{i,n} &= E_{i,n-1} + eV(\varphi_{s,n} + h\vartheta_{i,n}) - eV(\varphi_{s,n}) \end{aligned}$$

giving the change in the azimuth and energy of particle i during the n-th turn of the synchronous particle. The n-th energy increment comes at the end of the n-th turn. The relation between the synchronous beam circulation period $\tau_{s,n}$ and $\tau_{i,n}$, that of the i-th particle, is treated exactly. Thus,

the kinematic non-linearity is treated exactly; this feature can be very important if the synchronous energy is close to the transition energy. The lattice non-linearity is expressed as the dependence of γ_t on the momentum difference $\Delta p/p$ between the particle and the synchronous momentum. The rf potential is the sum of one or more sinusoidal terms so that the dynamic non-linearity of a simple waveform is treated exactly and other forms of potential are treated in a fourier expansion of ten or fewer terms.[3] The equation used in the program is generalized somewhat to permit multiple evenly spaced cavities per turn as an option and to allow the drift between cavities to be subdivided for more frequent application of the space charge kicks if the dynamics require it. The program ignores a slightly subtle distinction between $\vartheta_{i,n}$, the azimuthal variable in the mapping, and the periodic spatial variable $-\pi \leq \Theta_{i,n} \leq \pi$ which differs from it by an amount generally of no practical importance.[4] The conversion from the mapping variable to the true azimuth $\Theta_{i,n}$ of the particle is

$$\Theta_{i,n} = \frac{\tau_{s,n}}{\tau_{i,n}}\vartheta_{i,n} \approx \left(1 - \eta\frac{\Delta p}{p}\right)\vartheta_{i,n} \ .$$

For most applications to a high energy synchrotron the η-term is small compared to one. There might be some case for which the distinction between ϑ and Θ is important. There is no difficulty in modifying ESME to produce $\Theta_{i,n}$, but the change is not made in the general release because it adds to both cpu and memory requirements without practical benefit for any anticipated use.

The treatment starts from the specification of a reference orbit of average radius R_{eq} on which the mean normal magnetic field $<B_y>$ is known. A particle which would follow the reference orbit has the reference momentum $p_o = 2.997925 \cdot 10^2 <B_y> R_{eq}$ with p_o in MeV/c, R_{eq} in m, and $<B_y>$ in Tesla. The angular frequency of beam circulation on this orbit is $\Omega_o = \beta_o c/R_{eq}$ where β_o is the Lorentz β. The variation of the guide field away from the reference orbit is completely characterized for the purposes of describing longitudinal motion by the momentum dependence of the transition energy for fixed guide field. At a minimum one need only specify γ_t = const., a choice adequate in many instances. In many problems it is also correct to identify the radius of the synchronous trajectory R_s with the reference radius R_{eq}. However, in applications like stacking or displacement acceleration where the synchronous trajectory may be offset radially, or even out of the beampipe entirely, it is necessary to take explicit account of the difference between them. ESME calculates all motion relative to a hypothetical synchronous particle. The synchronous frequency is calculated from momentum and radius. The momentum compaction α_p and magnetic field at the synchronous radius are determined from the values provided on the reference orbit.

In the section on parameters for the tracking routine there will be mention of a parameter for choosing alternative versions of the difference equations. There are fundamentally two choices, one which calculates $\Delta\vartheta$ as indicated above and one that equates it to the familiar approximation $-2\pi\eta\Delta p/p$. Substantial time can be saved by using the simple difference equation if it is sufficient. Significant savings in execution time could be expected if the main tracking loop were rewritten to use only the simple form of the difference equation while eliminating branches to unused portions of the code.

[3] A little-used option mentioned in ref. [4] to provide a perfectly linear sawtooth waveform has been dropped because of severe conflict with the mechanism by which the new code finds the synchronous phase.

[4] The difference equations given in this note are not precisely those give in ref. [10] first, because the sense of particle motion is in the $-\vartheta$ sense here and in the $+\Theta$ sense in ref. [10] and, second, because the equations have been modified (slightly) to rigorously conserve longitudinal phasespace density for all beam energies. A detailed derivation of the difference equations will be given in a separate FN.

1.3 Multiparticle calculations

The particle distribution at each turn is calculated from that on the prior turn by a single turn map applied to each particle independently. The program provides three general types of optional calculation on the properties of the distribution as a whole. The most common calculations are those which serve to quantify the properties of the distribution so that one can plot them as functions of time. Examples of such properties are first and second moments, rms emittance, equivalent matched emittance, fourier spectrum of beam current, *etc.* Another type of calculation involves calculating feedback contributions to rf system parameters. ESME provides for phase feedback and feedback to rf amplitude. The third general class of collective calculation is the evaluation of beam induced voltages from space charge and longitudinal coupling impedance. The space charge calculation is based on a constant geometric factor relating average beam radius and average beampipe size. The longitudinal impedance can be characterized by an arbitrary table of real and imaginary part *vs.* frequency and/or a table of resonances defined by resonant frequency, Q, and real part at resonance. This last type of calculation was introduced to investigate effects of intensity on single bunch behavior at transition. Formerly, the harmonic response of a resonator was treated in the steady-state approximation only. However, since this treatment may not realistically represent the driving terms for certain types of collective bunch motion, the response of a high-Q resonator may now be modeled in the time domain as well (see ref. [10]). Reference [12] describes the use of this facility in modeling the coupled bunch instability.

Chapter 2

Program Organization and Data Requirements

Data for ESME are generally acquired by NAMELIST reads dispersed among subroutines which segregate different program functions as much as practical. The subroutine reading a particular class of data will be called only if there is in the input stream a single-character command requesting the related program function. The data relating to different program functions or accelerator subsystems are also stored in different FORTRAN COMMON blocks. These blocks are initialized with values or switches to allow the code to proceed on the basis of a few input data. The initial values, or those read in over them are retained unaltered[1] so that a command may be repeated without reentering data that remain *apropos*. There are default values provided even for some quantities which are nearly always problem-specific so that the program may continue execution far enough to expose more than a single data error in a test run.

The following subsection calls attention to requirements on command ordering arising from data dependencies. Next is a subsection listing all of the ESME command characters with a short description of their functions. The subsection following the listing of the command codes contains in the same ordering the NAMELIST's associated with the commands and a brief description of the function of each input datum.

2.1 Command ordering

The order in which the commands may appear in the data will generally be logically apparent. For example the **R** command which initiates the input of the basic lattice parameters and energy scale usually will appear before any other command. Several commands require this information to perform their own functions. The **A** command which brings in the rf parameters is generally the second command to appear. The command **P** which establishes the initial phasespace distribution generally needs to be preceded by both. The command **B** which sets up the machinery for longitudinal impedance calculations needs the initial distribution to get the initial beam current distribution. Although by detailed knowledge of the internal organization of the program or by exploiting some specialized facilities it is possible to create correct data sets that do not observe a typical command ordering, it is generally safer to adhere to the order **R**, **A**, **P**, **B** of first appearance for these commands. Other commands can usually be ordered arbitrarily.

[1] There are two minor exceptions; see **O** and **T** commands.

2.2 Command data

The principal functions of the program are activated by single-character commands appearing in the *first position* of a data record. Starting in the fifth position is an optional character field which serves to annotate the dataset itself and appears in the program output as a useful indication of the motivation for each step. The commands initiate a problem data read and/or an action upon available data. If data is requested by a subroutine invoked by a data command, data in NAMELIST form will follow in the next data record(s). Exceptions to this general rule can occur if the command is to call one of the optional user-coded subroutines and that subroutine uses another input technique. Also the title for graphical output which may be read as the result of an **O** command follows the $GRAPH namelist input data as a formatted character string data item.

The data requested or process initiated by a data command is related to a separable subsystem of the accelerator or a distinct phase of the calculation. The course of program execution is governed by the order of the command data in the data stream; that is, the program is what is sometimes called "data-driven". Commands are listed below in four groups. The first group consist of the commands that will appear in virtually every problem data set. They are listed in the order in which they are generally used. Interesting calculations may be made using these commands only. The second group contains commands to select output options. Those in the third group invoke special calculations, including if desired ones written by the user. The fourth pair is a save/restore pair for the entire state of the calculation so that a long calculation can be protected by check-pointing or a group of calculations sharing a common initial point can proceed from the conditions at that point. Commands with strong order dependence are underlined. In the description of a command the first letter of a keyword is indicated in bold face to call attention to the association between the command character and its function. Perhaps a more mnemonic correspondence could be found, but there is reason to retain as many of the features of earlier versions as practical.

List of Data Commands

R read in the lattice (**R**ing) parameters, magnetic field ramp, problem energies, *etc.* according to NAMELIST /RING/.

A read in the radiofrequency (**A**cceleration) parameters according to NAMELIST /RF/.

P Populate the phasespace with the initial distribution described by parameters read according to NAMELIST /POPUL8/.

O Select graphical **O**utput options using parameters in NAMELIST /GRAPH/.

T Track distribution according to parameters read in NAMELIST /CYCLE/.

Q **Q**uit the program.

D **D**isplay graphical output.

H Select quantities to be plotted from **H**istory records according to NAMELIST /HISTRY/.

M Save azimuthal histograms of distribution for composition of **M**ountain-range plot.

N Plot mountain-range data.

W Write comment into printed output.

L Set Low-level parameters (to control feedback, γ_t-jump, *etc.*) in NAMELIST /LLRF/.

B Setup space charge calculation; read in **B**eam parameters in NAMELIST /SCHG/.

F Setup the fast **F**ourier transform of the theta distribution, according to parameters in NAMELIST /FFT/.

0-9 Enter subroutine SHAZAM at SHAZAM, SHAZAM1, SHAZAM2, ... to manipulate any quantity in COMMON storage in the program.

S Save tracking and control data.

G Get tracking and control data from a previous run.

2.2.1 R Command - Lattice Parameters

The members of NAMELIST /RING/ are stored in COMMON /RINGP/. This read is in SUBROUTINE RINGPAR, which also derives quantities like η, γ_t, *etc.* which depend only on lattice parameters. It is possible to distinguish the reference trajectory, of energy E_0, from the synchronous trajectory, of energy E_s, by specifying an offset from the reference trajectory.

R Command, Namelist /RING/			
Variable	Default Value	Default Unit	Description
REQ	None	m	The reference radius for the central orbit.
GAMMAT	None	-	Transition γ.
ALPHA1	None	-	Coefficient of $\Delta p/p$ in series expansion for α_p about reference orbit.
ALPHA2	None	-	Coefficient of $(\Delta p/p)^2$ in series expansion for α_p about reference orbit.
ALPHA3	None	-	Coefficient of $(\Delta p/p)^3$ in series expansion for α_p about reference orbit.
EK0I	None	MeV	Kinetic energy on the central orbit at T = TI.
EK0F	0.0	MeV	Kinetic energy on the central orbit at T = TF.
TI	0.0	s	Time corresponding to start of magnetic field change.
TF	0.0	s	Time corresponding to end of magnetic field change.
TSTART	0.0	s	Time at which tracking begins.
FRAC	1	-	Determines azimuthal periodicity, calculation restricted to $-180°/\text{FRAC} \leq \vartheta \leq 180°\text{FRAC}$.
PIPRAD	1.0	m	Radius of beam pipe.
EBDRY	F	-	Sets absorbing beam pipe "walls" at REQ $\pm$ PIPRAD.
DES	0.0	MeV	Specified energy offset of synchronous orbit relative to reference orbit.

R Command, Namelist /RING/, continued			
Variable	Default Value	Unit	Description
KURVEB	1	-	Magnetic field ramp from EK0I to EK0F: 1 – Linear 2 – Increasing parabolic 3 – Biased sinusoidal 4 – Decreasing parabolic 5 – Parabolic, from EKIDOT to EKFDOT.
EKIDOT	0.0	MeV/s	Slope of parabolic ramp at TI.
EKFDOT	0.0	MeV/s	Slope of parabolic ramp at TF.
JNRAMP	F	-	Establishes starting point of ramp as point at which program finds itself–for smoothly piecing ramp segments together.
GMAJMP	F	-	Set γ_t-jump on.
KINDG	1	-	Type of $\gamma_t$variation: [a] 1 – Linear (γ_t = GAMPAR(1) + GAMPAR(2) * T) 2 – Decreasing exponential (γ_t = GAMPAR(1) + GAMPAR(3) * (1 − $e^{-T/GAMPAR(2)}$)).
GAMPAR(1:3)	0.0	-	Coefficients for $\gamma_t$variation.

[a] T = 0 corresponding to time at which R command is invoked with GMAJMP = .TRUE.

An example of a "minimum" R command might be:

```
$RING REQ=1000., GAMMAT=18.75, EK0I=150000., FRAC=159. $END
```

in which the lattice is characterized simply by a radius, $\gamma_t$value, and a field which corresponds to a reference energy of 150 GeV. FRAC is useful for restricting the range of consideration of ESME to a suitable period of the ring. In this case, since EK0F is not specified, the default, EK0F = 0.0, indicates that the "field" [2] is static. In addition, the "start" time for this run is taken to be zero.

For TSTART ≤ TI the energy of the reference orbit is taken to be EK0I, while for TSTART ≥ TF the energy of the reference orbit is EK0F. Note that PIPRAD has no effect unless EBDRY = T. If the γ_t-jump option has been invoked, then γ_t is varied until the R command is issued with GMAJMP = F.

2.2.2 A Command - RF Parameters

The members of NAMELIST /RF/ are read in SUBROUTINE RFPROG, and stored in COMMON /RFP/. Up to 10 independent voltage sources may be specified.

A Command, Namelist /RF/			
Variable	Default Value	Unit	Description
NRF	1	-	Number of active RF sources.
H(1:10)	1	-	Harmonic numbers of sources.

[2] An explicit consideration of magnetic fields is not a part of the program; rather, an average guide field is assumed to exist.

A Command, Namelist /RF/, continued			
Variable	Default Value	Unit	Description
HW(1:10)	1	-	Voltage sources will be "active" for $-180°/HW \leq \theta \leq 180°/HW$.
ISYNC	0	-	Indicates synchronism condition to be imposed on RF: 0 – None, voltages and phases remain as programmed 1 – Phase of RF waveform shifted to synchronous, stable point 2 – Magnitude of RF waveform scaled to give correct synchronous energy gain 3 – Source 2 acts as Landau cavity to source 1; synchronism only assured for sources 1 and 2.
EXCHRF	T	-	Flag indicating that sources should be interchanged so that source 1 is always the greatest contributor to the bucket height.
VI(1:10)	0.0	MV	Voltage of source I at time TVBEG(I).
VF(1:10)	0.0	MV	Voltage of source I at time TVEND(I).
TVBEG(1:10)	0.0	s	Time corresponding to beginning of RF voltage change.
TVEND(1:10)	0.0	s	Time corresponding to end of RF voltage change.
KURVE(1:10)	0	-	Specifies type of RF voltage variation between times TVBEG and TVEND: 0 – None, voltage maintained at VI(I) 1 – Linear 2 – Isoadiabatic 3 – Sigmoid 4 – Cubic spline interpolation[a].
VKON	T	-	Indicates whether programmed voltage curves are to be active.
PSII(1:10)	0	deg	Phase of source I at time TPBEG(I).
PSIF(1:10)	0	deg	Phase of source I at time TPEND(I).
TPBEG(1:10)	0.0	s	Time corresponding to beginning of RF phase change.
TPEND(1:10)	0.0	s	Time corresponding to end of RF phase change.
KURVP(1:10)	0	-	Specifies type of RF phase variation between times TPBEG and TPEND: 0 - None, phase maintained at PSII(I) 1 - Linear 2 - Quadratic. 4 - Cubic spline interpolation[a].
PHKON	F	-	Indicates whether or not phase curves are to be active.
FRI(1:10)	0.0	MHz	Frequency of source I at time TFBEG(I).
FRF(1:10)	0.0	MHz	Frequency of source I at time TFEND(I).
TFBEG(1:10)	0.0	s	Time corresponding to beginning of frequency change.
TFEND(1:10)	0.0	s	Time corresponding to end of frequency change.

[a]Fit to values read from file. See Section3.2.2

A Command, Namelist /RF/, continued			
Variable	Default Value	Unit	Description
KURVF(1:10)	0	-	Specifies type of frequency variation between times TFBEG and TFEND: 0 - None, frequency maintained at FRI(I). 1 - Linear 2 - Quadratic. 4 - Cubic polynomial interpolation[a].
FRKON	F	-	Indicates whether frequency curves are to be active.
CNTINU	F	-	Sets the starting voltage, phase and/or frequency for any active sources as the current values–for smoothly piecing curve segments together.
VMATCHI(1:10)	F	-	For source I, VMATCH(I) = T results in the VI(I) being set so that source I is matched to the current distribution emittance. [b]
VMATCHF(1:10)	F	-	Sets VF(I) so that source I is matched to the current distribution emittance.
HOLDBH	F	-	If true, then voltage is to be varied so that bucket height due to **source 1**[c] is multiplied by HDECR on successive turns.
HDECR	1.0	-	Factor by which bucket height (for source 1) is to be adjusted on successive turns if HOLDBH = T.
HOLDBA	F	-	If true, then voltage is to be varied so that bucket area due to **source 1**[c] is multiplied by SDECR on successive turns.
SDECR	1.0	-	Factor by which bucket area (for source 1) is to be adjusted on successive turns if HOLDBA = T.
PHISLIM	.95	-	Voltage may not be reduced such that $\sin\phi_s >$ PHISLIM using options HOLDBH and HOLDBA.
PHSLIP	F	-	Flag indicating that the phase of at least one source is to be varied to correspond to a momentum offset from the synchronous value (see DELTRF).
DELTRF(1:10)	0.0	-	Momentum offset ($\Delta p/p$) at which source I is to be operated.

[a] Coefficients read from file. See Section 3.2.2
[b] Which means, in this instance, that the P command, or its equivalent, should precede the A command.
[c] The algorithms used to maintain the bucket height and area consider only a single source.

An example of a "minimum" A command might be:

```
$RF H(1)=1113, VI(1)=.100, PSII(1)=235.0 $END
```

Here, one voltage is specified by a minimum set of parameters. Note that H(I) is an integer (as is HW(I)). Since RF manipulations are at the heart of ESME, this command can become rather lengthy and involved; the example given above is exceptionally brief. Just as in the case of the magnetic field ramp – specified in the R command – at times prior to the start of a programmed variation the relevant quantity is maintained at the initial value, while at times after the indicated end of a programmed curve, the quantity is maintained at the final value. This makes it easier to specify multiple curves in unequal time steps.

The values of the programmed phase curves are to be distinguished from the phase of a given harmonic at the synchronous particle, though they may be the same. For example, a user could represent any periodic waveform (in a Fourier expansion of up to 10 terms) simply by specifying the correct relative phases and amplitudes of the voltages. The waveform could be modified over time by specifying the variations of the voltages, phases, and/or frequencies[3]. Finally, the user could specify that the program search for a "synchronous" point on the resultant waveform using ISYNC. Option ISYNC = 1 will result in the program searching for a stable value of the phase, stored internally in the program as PHIS, in units of 2π around the ring, and output as degrees of "shift" of the sum voltage waveform. Option ISYNC = 2 will cause the resultant waveform to be scaled to give only the correct magnitude of voltage at the synchronous particle, not necessarily at a stable slope, since the phases of the voltage sources at the synchronous particle remain at their programmed values. The Landau cavity option, ISYNC = 3, will vary the phases of voltage sources 1 and 2, and the magnitude of voltage source 2, so that the first and second derivatives of the voltage vanish at the synchronous particle. The phases and magnitudes of other voltages are not altered, though they will be included in the iteration of the difference equation if NRF > 2. As noted in the table, the options activated by HOLDBH and HOLDBA only apply to source 1.

2.2.3 P Command - Initial Distribution Parameters

The members of NAMELIST /POPL8/ are read in SUBROUTINE POPUL8, and stored in COMMON /POPLATE/.

P Command, Namelist /POPL8/

Variable	Default Value	Default Unit	Description
KIND	1	-	Chooses the type of distribution to be generated: 1–Rectangular outline, NTH by NE points, limited by THMIN, THMAX, REMIN, REMAX 2–Uniform rectangular grid NTH by NE, limits as in KIND = 1 3–Random uniform distribution of NPOINT points within rectangular limits as in KIND = 1 4–Random uniform in θ, limits THMIN, THMAX; Gaussian in E, limits at REMIN, REMAX = $\pm 2\sigma$, NPOINT points 5–Gaussian in θ, limits at THMIN,THMAX = $\pm 2\sigma$; random uniform in E, limits REMIN, REMAX, NPOINT points 6–Rectangular grid, regular in θ, Gaussian in E, NTH by NE points The remaining distribution types, except for 11, are *matched* , in that the distribution is limited by a contour of SBNCH eVs. 7–Bunch outline of NPOINT particles 8–Regular grid of approximately NTH by NE particles 9–Random uniform bunch of NPOINT particles within contour 10–Bi-Gaussian distribution of NPOINT particles, 95% within contour

[3]In cases in which a frequency and a phase variation are specified for the same source, the phase variation takes precedence.

P Command, Namelist /POPL8/, continued

Variable	Default Value	Default Unit	Description
			11–NPOINT uniformly spaced particles on flow lines just above and below bucket boundary 12–Random uniform in E, parabolic in θ 13–Parabolic distribution of NPOINT particles.
THMIN	-90.0	deg	Lower θ limit on rectangular distribution.
THMAX	90.0	deg	Upper θ limit on rectangular distribution.
REMIN	None	MeV	Lower energy limit on rectangular distribution; relative to the synchronous energy, ES.
REMAX	None	MeV	Upper energy limit on rectangular distribution.
NTH	2	-	Number of grid points in θ direction.
NE	2	-	Number of grid points in E direction.
SBNCH	0.1	eVs	Area within matching contour.
IPOP	1	-	Specifies which RF source to be used in matching: 0–All active (NRF) sources I–Source I ($1 \leq I \leq NRF$).
THOFF	0.0	deg	Amount to displace distribution generated in current call to POPUL8 in θ direction.
EOFF	0.0	MeV	Amount to displace currently generated distribution in E direction.
THTRAN	0.0	deg	Amount to displace all particles (generated in this and previous calls to POPUL8) in θ direction.
ETRAN	0.0	MeV	Amount to displace all particles in E direction.
NPOINT	1	-	Number of particles generated for all distributions except KIND = 1,2,6,8, in which NTH and NE are used.
PARTION	F	-	"Partition" distribution into separate classes[a]; each separate use of the P command with PARTION = T introduces a new partition.
ISEED	314159	-	Seed for random distributions.

[a]Different classes of particles may be plotted with distinct symbols.

If one wanted to populate a bi-Gaussian distribution of 100 particles, with an emittance (95%) matched to RF source 1 of .02 eV-s, then the P command would be:

```
$POPL8 KIND=10, NPOINT=100, IPOP=1, SBNCH=.02 $END
```

For multi-bunch simulations, in which several distributions are needed with the same parameters but in different positions, it is sufficient to simply re-issue the P command with only the desired position offset. For example:

```
$POPL8 THTRAN=45.0, ETRAN=0.0 $END
```

Since NAMELIST members which do not appear in the input remain unchanged, one may exploit

this property to abbreviate the amount of input. In particular, expeditious use of THTRAN and ETRAN can make it unnecessary to explicitly declare the position of each group of particles.

2.2.4 O Command - Graphical Output Options

The members of NAMELIST /GRAPH/ are read in SUBROUTINE GRAFSET, and stored in COMMON /GRAFIX/. The actual graphical output can be generated either during or after processing. The plotting routines written specifically for this version of ESME use GRAFMAKER under DI-3000[4]. The post-processor is described later in Appendix A. Users wishing to process ESME data independently or with different graphics routines may find it of some use.

O Command, Namelist /GRAPH/			
	Default		
Variable	Value	Unit	Description
MPLOT	1000	turn	Frequency of output; every MPLOT turns.
IDEV	1	-	Virtual device number for graphical output[a].
POSTP	F	-	Write all data in COMMON blocks to unit 18; do not call plotting routine.
TITLE	F	-	Indicates that line immediately following NAMELIST input is to be used as a plot title[b].
PLTSW			Select plot options:
(1)	T	-	Draw phase space plot.
(2)	T	-	Plot phase space points (different symbol for each class).
(3)	F	-	Interconnect points within each class.
(4)	F	-	Draw lines at centroid and $\pm\sigma$.
(5)	F	-	Draw voltage waveform.
(6)	F	-	Set plot boundaries to turning points of contour.
(7)	F	-	Suppress captions, axis labels etc...
(8)	T	-	Plot θ histogram.
(9)	F	-	Set θ histogram limits to turning points of contour.
(10)	T	-	Plot E histogram.
(11)	F	-	Set E histogram limits to turning points of contour.
(12)	F	-	Plot fourier amplitudes.
(13)	F	-	Include phases in plot of fourier spectrum.
(14)	F	-	Plot space charge energy loss (per turn) *vs.* θ.
(15)	F	-	Include distribution histogram on space charge plot.
(16)	F	-	Plot high-Q resonator voltage.
NPJMP	1	-	In phase space plot, plot only every NPJMPth point.

[a] DI-3000 specific; see 3.1

[b] This is an exception to the maintenance of NAMELIST input; TITLE is set to .FALSE. after every execution of the O command.

[4] GRAFMAKER and DI-3000 are trademarks of Precision Visuals, Inc.

O Command, Namelist /GRAPH/, continued			
Variable	Default Value	Unit	Description
KLPLOT	0	-	Controls plotting of classes in phase space plot and projections (see Section 2.2.3) 0–All classes plotted 1 ≤ KLPLOT ≤ KLASSES–Plot class KLPLOT only
IOPT	-1	-	Selects voltage source for contour plotting: < 0–None; no contour plotted 0–All active (NRF) sources 1-10-Source IOPT (1 ≤ IOPT ≤ NRF).
MTCHSI	F	-	Generate contour of area equal to initial distribution emittance.
MTCH95	F	-	Generate contour containing 95% of the present distribution emittance.
THPMIN	0.0[a]	deg	Lower θ limit for phase space plot.
THPMAX	0.0	deg	Upper θ limit for phase space plot.
DEPMIN	0.0[b]	MeV	Lower E limit for phase space plot.
DEPMAX	0.0	MeV	Upper E limit for phase space plot.
IEREF	1	-	Determines energy origin for phase space: 1–E0, the reference energy (often = ES) 2–ES, the synchronous energy 3–EBAR, the average particle energy 4–EREF, the "reference" particle energy[c].
NBINTH	50	-	The number of bins for the θ histogram.
THBMIN	0.0[d]	deg	Lower limit for θ histogram.
THBMAX	0.0	deg	Upper limit for θ histogram.
NBINE	50	-	The number of bins for the E histogram.
EBMIN	0.0[e]	MeV	Lower limit for E histogram.
EBMAX	0.0	MeV	Upper limit for E histogram.
IFBMIN	1	-	Lower limit for FFT plot.
IFBMAX	0[f]	-	Upper limit for FFT plot.
SCBMIN	0.0[g]	-	Lower θ limit for space-charge plot.
SCBMAX	0.0	-	Upper limit for space-charge plot.
RBMIN	0.0[h]	-	Lower θ limit for resonator voltage plot.
RBMAX	0.0	-	Upper limit for resonator voltage plot.

[a] THPMIN and THPMAX both 0.0 results in the phase space being plotted for $-180°/\text{FRAC} \leq \theta \leq 180°/\text{FRAC}$.

[b] DEPMIN and DEPMAX both 0.0 results in the phase space being plotted over approximately the entire range of particle energies

[c] A particle which ESME tracks from the origin (0,ES) as a reference.

[d] Limits of 0.0 for both THBMIN and THBMAX result in the plot range being the same as for the phase space plot.

[e] Limits of 0.0 for both EBMIN and EBMAX result in the plot range being the same as for the phase space plot.

[f] IFBMAX = 0 results in the upper limit being the greatest Fourier harmonic computed.

[g] Limits of 0.0 for both SCBMIN and SCBMAX result in the range for the plot being $\pm 180°/\text{FRAC}$.

[h] Limits of 0.0 for both RBMIN and RBMAX result in the range being $\pm 180°/\text{FRAC}$.

O Command, Namelist /GRAPH/, continued			
Variable	Default Value	Unit	Description
DTHCURV	0.0	deg	Amount by which contour will be moved in θ direction.
DECURV	0.0	MeV	Amount by which contour will be moved in E direction.
DELCON	.01	-	Determine RF bucket to within DELCON*360 degrees of RF waveform.
KNTLIM	500000	-	Number of iterations of difference equation which will be attempted to close contour.

The options enabled by the O command are largely self-explanatory; the default PLTSW settings result in the output of a phase space plot of the distribution and the bucket due to RF source 1, as well as plots of the projections of the distribution along the θ and E directions. An example of an O command requesting such output every 100 turns might be:

```
$GRAPH  MPLOT=100, THPMIN=-10.0, THPMAX=10.0,
        DEPMIN=-25.0, DEPMAX=25.0 $END
```

in which the limits are appropriate ones chosen by the user. In addition to the graphical output generated as a result of the O command, the distribution moments are computed and output, as well as a number of other system parameters. The moments included on the plots are derived from the particles which are in the class(es) being plotted and within the plot limits. The moments printed in the standard output are those for the entire distribution. The "default" limits for many of the plots serve as flags to the plotting routine to choose reasonable limits. Note that the phase space projection plots' limits (THBMIN, EBMIN, *etc.*) are necessary only if they are to be different than the phase space plot limits. The parameter DELCON is included to allow the user to either determine the separatrix arbitrarily closely or to save processing time, since in certain situations the routine which determines the "bucket" in ESME is required to perform many iterations of the difference equations in order to determine the separatrix to the specified accuracy. In those instances in which the contour-drawing routine is unacceptably slow (or unable) to find a contour, it may be useful to set KNTLIM to some lower number.

2.2.5 T Command - Track Distribution

The members of NAMELIST /CYCLE/ are read in SUBROUTINE CYCPROG and stored in COMMON /CYCLP/.

T Command, Namelist /CYCLE/			
Variable	Default Value	Unit	Description
TSTOP[a]	0.0	s	Time at which to stop tracking.
TTRACK[b]	0.0	s	Duration of time to track.
MSTEP	100	-	Number of tracking steps (minimum) per synchrotron period.
ACCEL0	1.0	-	Number of beam turns per tracking step (maximum) [c].

[a] TSTOP is set to 0.0 when tracking is completed, or interrupted by an ITRAP option.
[b] TSTOP takes precedence; if TSTOP=0.0, then TTRACK determines duration of tracking.
[c] Tracking will proceed at the nearest integer to ACCEL0 (not < 1), limited by MSTEP.

T Command, Namelist /CYCLE/, continued			
Variable	Default Value	Unit	Description
NCAV	1	-	Number of "RF cavities" in the ring, i.e., the number of iterations of the difference equations per turn.
LGRTHM	1	-	Select difference equations used in tracking 1–Complete kinematics, expand α_p to maximum order using input coefficients ALPHAN[a]. 2–Use the simplified difference equation $\vartheta_{i,n} = \frac{\tau_{s,n}}{\tau_{s,n-1}}\vartheta_{i,n-1} + 2\pi\eta\frac{\Delta p}{p}$
ITRAP(1:4)	0	-	Indicates a condition for which tracking should be interrupted before time indicated by TTRACK or TSTOP: 0–No trap 1–Trap on minimum bunch width 2–Trap on minimum bunch height 3–Trap for η = ETATRP (tolerance $\Delta\eta/\eta = \pm.01$) 4–Trap for $\lvert\sin\phi_s\rvert$ = PHISTRP (tolerance $\Delta\phi_s = \pm.005$) 5–Trap for $\eta > 0$ (transition crossing). 10-19–Call SUBROUTINE SHAZAM, enter at SHAZAM, SHAZAM1, SHAZAM2, ... following every iteration of the difference equations.
ETATRP	.001	-	For ITRAP = 3; tracking stopped when η = ETATRP.
PHISTRP	.95	-	For ITRAP = 4; tracking stopped when $\lvert\sin\phi_s\rvert$ = PHISTRP.
MGRACE	0	-	Allow a "grace period" of MGRACE turns before trapping conditions are checked.
HISTRY	F	-	Write a history record to unit 9 following every iteration of the difference equations.[b]
MOMNTS	F	-	Compute the distribution moments following every iteration of the difference equations.
BBDRY	F	-	Remove particles tracked outside of region $-180°/\text{FRAC} \leq \theta \leq 180°/\text{FRAC}$.

[a] See R command
[b] See Section 3.2

The options in the T command direct the flow of processing in SUBROUTINE CYCPROG, which iterates the difference equations for the particles. A typical T command line might be:

```
$CYCLE TTRACK=.001, LGRTHM=3, HISTRY=T, MOMNTS=T $END
```

in which tracking is specified to take place for .001 seconds of simulated time, with a calculation of distribution moments and a write to the history file at every iteration of the difference equations. Note that *either* TSTOP or TTRACK may be used to specify the duration of tracking, though TSTOP takes precedence. Also, since an interruption in tracking by an ITRAP option (for ITRAP < 6) results in TSTOP being set equal to 0.0, the stop time must be respecified in a subsequent T command. Any condition which halts or interrupts tracking is checked at most

once per turn, so tracking duration may be as much as one beam circulation period longer than specified by TSTOP or TTRACK. Four ITRAP variables are provided to allow for multiple traps and/or calls to SHAZAM routines during tracking. If $10 \leq$ ITRAP ≤ 19, then a call is made to the appropriate SHAZAM entry point following every turn (or ACCEL turns, if ACCEL0 > 1). Tracking is not stopped.

2.2.6 Q Command - Quit

The Q command directs ESME to cease processing. No more commands are read.

2.2.7 D Command - Display

The D command directs ESME to generate graphical output at the point at which the command is issued. The form of the output is specified by the most recent O command. The D command is useful for generating output at a particular point in a calculation, since the O command itself only provides output every MPLOT turns.

2.2.8 H Command - History Output

The members of NAMELIST /HISTRY/ are read in subroutine HISTORY.

H Command, Namelist /HISTRY/			
Variable	Default Value	Unit	Description
IDEV	1	-	The virtual device number for output[a].
NPLT(1:2,1:50)	0[b]	-	"Index" of element in history records; NPLT(1,I) is independent variable, NPLT(2,I) is dependent variable. Real records: 1–Time 2–PHIS, the "synchronous phase"[c] 3–PDOT, dp_s/dt 4–THBAR, the mean value of θ for the distribution 5–EBAR, the average energy of the distribution 6–THRMS, the rms spread in θ of the particles 7–ERMS, the rms energy spread 8–ES, the synchronous energy 9–E0, the energy on the reference orbit 10–ES-E0 11–THREF, the azimuth of a particle tracked from (0,ES) 12–EREF, the energy of a particle tracked from (0,ES)

[a] DI-3000 specific; see Section 3.1.
[b] The default value of 0 indicates to SUBROUTINE HISTORY that all of the desired history plots have been generated; so only the first set of consecutive nonzero entries to array NPLT will generate plots.
[c] See **A** command description.

H Command, Namelist /HISTRY/, continued			
Variable	Default Value	Unit	Description
			13–EPSILON, the "emittance"[a]
			14–NUS, the synchrotron frequency
			15–SBCKT, the RF "bucket" area
			16–HBCKT, the RF "bucket" height
			17–ETA, $1/\gamma_t^2 - 1/\gamma^2$
			18–ACCEL; see T command
			19–TAU, synchronous revolution period
			20–PSIADD, phase feedback; see L command
			21–DAMPL, voltage feedback factor; see L command
			22–DELR, synchronous orbit radius - reference orbit radius
			23–RFFREQ, frequency of RF source 1
			Integer records:
			31–TURN NUMBER
			32–KOUNT, number of particles
			Array records:
			51-60–SPARE(1-10)
			101-110–EV(1-10)
			111-120–PSI(1-10)
			121-130–FREQ(I)-FRI(I), Change in frequency of source I
			201-250–FAMPL(1-50), Fourier amplitudes; see F command
			251-300–FAZE(1-50), Fourier phases; see F command

[a] EPSILON $= 10^6 \tau(\sqrt{\sum \theta_i^2 \sum E_i^2 - (\sum \theta_i E_i)^2})/N$ eV-s

The output produced by the H command, as implied by the simplicity of the /HISTRY/ NAMELIST, is not presently as flexible as the output produced by the O command. The user simply specifies pairs of values to be plotted using the indices above in array NPLT, and optionally the device number, and the program produces a history plot for each pair of inputs assembled sequentially from the entire history record (*i.e.*, there is no choice of a range for any axis). For example, suppose that a simulation has run for 10,000 turns, and the desired output is a record of the distribution moments *vs.* time over that period. The H command might appear as:

```
$HISTRY NPLT=1,4,1,5,1,6,1,7,4,5 $END
```

This command will generate plots of THBAR, EBAR, THRMS, and ERMS *vs.* time, as well as a plot of EBAR *vs.* THBAR. (The manner in which FORTRAN array indices are cycled is taken advantage of here to avoid explicit reference to the indices; this is the recommended manner of input.) Due to the fact that the frequency of write operations to the history tape may be modified by the program[5], and to a culling procedure applied to the data points by the plotting routine, for simulations of more than several thousand turns not every turn will be included in the plot.

[5] See Appendix A

2.2.9 M Command - Save Mountain Range Data

The members of NAMELIST /MRANGE/ are read in SUBROUTINE MRINIT, and stored in COMMON /MRANGE/.

M Command, Namelist /MRANGE/			
Variable	Default Value	Default Unit	Description
TMBEGIN	0.0	s	Time at which to start saving mountain range data.
TMEND	0.0	s	Time after which to stop saving mountain range data.
MRMPLOT	1	turn	Turn interval at which to record mountain range data.
MRNBIN	50	-	Number of bins for mountain range histogram.
MRTHBMIN	$-180°$/FRAC	deg	Minimum value of θ for mountain range.
MRTHBMAX	$180°$/FRAC	deg	Maximum value of θ for mountain range.

The M and N commands are intended to provide plots similar to those provided by an oscilloscope displaying successive traces from a longitudinal monitor, each trace being vertically displaced from the previous one. The resultant display depicts the time evolution of the azimuthal projection of a distribution in a manner which somewhat resembles a "mountain range," hence the name. The M command merely directs the program to save the data, while the N command directs the program to process the data (saved in a file) and produce mountain range plots.

2.2.10 N Command - Plot Mountain Range Data

The members of NAMELIST /MRPLOT/ are read in SUBROUTINE MRPLT, and stored in COMMON /MRANGE/.

N Command, Namelist /MRPLOT/			
Variable	Default Value	Default Unit	Description
MRTHPMIN	0.0[a]	deg	Minumum θ value for mountain range plot
MRTHPMAX	0.0	deg	Maximum θ value for mountain range plot.
NTRACE	100	-	Number of traces on a page.
NSKIP	0	-	Number of records to be "skipped" between each trace.
TOPTOB	0.7	-	The fraction of the vertical range over which NTRACE traces are to be plotted (approximate if TBASE=T).
SCALE	0.3	-	The height of the first trace, in units in which the entire vertical range of the plot is 1.0.
MSTART	0[b]	-	Turn number at which to start plots.
MSTOP	0	-	Turn number at which to stop plots.

[a]Defaults of 0.0 for MRTHPMIN and MRTHPMAX imply that data is to be plotted over its entire range.
[b]The defaults of 0 for MSTART and MSTOP, or 0.0 for TMSTART and TMSTOP, imply that all mountain range records are to be plotted

N Command, Namelist /MRPLOT/, continued			
Variable	Default Value	Unit	Description
TMSTART	0.0	s	Time at which to start plots.
TMSTOP	0.0	s	Time at which to stop plots.
TBASE	F	-	Switch causing plot trace separation to be proportional to time.
IDEV	1	-	Output device.
LIM	F	-	Switch which causes dotted lines to be plotted connecting consecutive traces' leftmost and rightmost non-zero points.
SMOOTH	F	-	Switch which causes a smoothing algorithm to be applied to the data for plotting.

A typical pair of M and N commands producing mountain range plots might be

```
$MRANGE TMBEGIN=0.0, TMEND=1.0, MRMPLOT=10 $END
```

in which mountain range records are recorded every 10 turns of tracking from 0.0 to 1.0 seconds, and

```
$MRPLOT SMOOTH=T $END
```

which directs that all of the data accumulated thus far be plotted in mountain range format with smoothing. The default for TBASE (F) results in a fixed vertical separation between consecutive traces. The TBASE option causes the vertical separation between consecutive traces to be proportional to the time separating their records, which better simulates the mountain ranges normally depicted on an oscilloscope[6]. The values for TOPTOB and/or SCALE may have to be adjusted to achieve a satisfactory effect.

2.2.11 W Command - Write Comment

The W command simply directs the program to echo the characters following W (on the same line and after four spaces) to the standard output. It is intended to provide the user with the ability to insert comments into an input dataset.

2.2.12 L Command - Low Level Feedback Parameters

The members of NAMELIST /LLRF/ are read in subroutine LOWLVL and stored in COMMON /FEEDS/.

L Command, Namelist /LLRF/			
Variable	Default Value	Unit	Description
PHFBON	F	-	Activates phase feedback.
VFBON	F	-	Activates voltage feedback.

[6] In relativistic situations, no difference will be discerned between the plots generated with TBASE either T or F.

L Command, Namelist /LLRF/, continued			
Variable	Default Value	Default Unit	Description
NTUAVG	1	-	The number of past turns to average in computing the feedback; the default NTUAVG = 1 represents infinite-bandwidth feedback.
NTURES	1	-	The number of turns for the feedback to respond; the present signal is compared to the signal of NTURES turns ago.
IFTB	0	-	The form of phase feedback: 0–Critical damping 1–Fixed.
FBFACT	1.0	-	The gain applied to the phase feedback.
USEWT	F	-	Applies "weight function" W to phase signal over NTUAVG turns.
W(1:NTUAVG)	0.0	-	"Weight function" multiplying phase signal.
DLIMIT	5.7296	deg	The upper limit on the magnitude of the phase feedback on a given turn.
VFBFCTR	1.0	-	The gain applied to the voltage feedback.
VLIMIT	.1	MV	The limit on the voltage feedback applied on a given turn.
ETAJMP	0.0	-	The value of η[a] at which to "flip" the phase of the RF.

[a] $\eta = 1/\gamma_t^2 - 1/\gamma^2$

The phase feedback, intended primarily to damp dipole bunch oscillations, is computed according to the following formula for ITFB = 0:

$$\text{PSIADD} = \frac{\text{FBFACT}\sum_{i=1}^{\text{NTUAVG}}(W_i(\bar{\theta}_{n-i} - \bar{\theta}_{n-\text{NRESP}-i})}{\pi\nu_s \text{NTUAVG}^2 \sum_{i=1}^{NTUAVG} W_i}$$

where n is the current turn number, and ν_s is the synchrotron frequency. For ITFB = 1, ν_s is replaced by $4.0X10^{-3}$. A simple invocation of phase feedback would appear as

```
$LLRF PHFBON=T $END
```

The defaults imply critical damping.

The voltage feedback, intended to damp quadrupole bunch oscillations, operates according to

$$V_n' = (1 + \frac{2\pi\text{VFBFCTR}\sum_{j=0}^{n-k}(< E^2 >_{n-j} - < E^2 >_{n-j-1})}{100\nu_s H_k}) * V_n$$

where k is the turn index when feedback starts (i.e., the turn number when the L command is issued with VFBON = T), n is the current turn number, E is the energy, V is the voltage before feedback, V' is the voltage after feedback, and H_k is the height of the bucket on the kth turn. The factor of 100 in the denominator is somewhat arbitrary, so the user may have to adjust VFBFCTR to obtain satisfactory results.

2.2.13 B Command - Space Charge

The members of NAMELIST /SCHG/ are read at entry BEAMSC in subroutine FOURFIT and stored in COMMON /SPCHG/.

B Command, Namelist /SCHG/			
Variable	Default Value	Unit	Description
SCON	F	-	Activate space charge calculation.
A	$2.E-3$	m	Effective beam "radius".
B	$5.E-2$	m	Effective beam pipe "radius".
ENQ	$2.E+10$	-	Number of protons to be represented by the distribution.
NZ	0	-	Number of impedance values to be read from a file. [a]
NR	0	-	Number of resonance values to be read from a file. [b]
QREZON	F	-	Activate high-Q resonance voltage calculation.
MSC	1	-	Collective effects are to be calculated MSC times between rf cavities[c].
NNF [d]	0	-	Number of fourier harmonics to be stored in history.
NF(1:NNF)	0	-	Harmonic numbers of fourier spectrum components to be stored.
MAXFFTB	1024	-	Maximum number of bins to be used in Fourier transform.
MFFT	1	-	Frequency of Fourier transform calculation.

[a] Cubic polynomial coefficient table read from FORTRAN logical unit 11; see Section 3.2.3 for format specification.
[b] Resonance parameters read from FORTRAN logical unit unit 12; see Section 3.2.4 for format specification.
[c] The number of such calculations per turn will be MSC*NCAV.
[d] For instruction in the use of NF(1:NNF), see the description following the F command.

The B command controls facilities in ESME for modeling the interactions of the beam particles with each other through both the direct particle-particle force and through wakefields excited as a consequence of the interaction of the beam with its environment (vacuum chamber, rf cavities, *etc...*). The routines activated by SCON calculate these effects using an equivalent impedance. This may be adequate for many problems. However, implicit in this approach is the assumption of a steady-state solution for the response of the resonator[10]. This is reflected in the absence of any frequencies other than harmonics of the revolution frequency in this calculation, which utilizes the Fourier transform.

Transient effects are taken into account using the routines activated by QREZON. In cases where the space-charge force is to be calculated and resonant sources are present, QREZON will take precedence over SCON in the calculation of the voltage due to the resonator. Implicit in the code is the assumption that the distribution being tracked is periodic, and that the resonator voltage is applied in time steps equal to the time duration of the entire distribution. The simplest way to satisfy both of these constraints is to track a distribution which spans 360 degrees of azimuth (i.e., the entire ring), and calculate the resonator voltage once per turn. It may be satisfactory in other cases to track a distribution spanning $\frac{1}{\mathrm{FRAC}}$ of 360 degrees and calculating the voltage at FRAC equispaced intervals around the ring (i.e., $\mathrm{MSC} * \mathrm{NCAV} = \mathrm{FRAC}$).

A space charge command which utilizes ESME's capabilities to model the self-force, resistive wall impedance, and certain parasitic high-Q resonances might be:

```
 $SCHG SCON=T, A=.01, B=.03, ENQ=2.4E10,
       NR=6, NZ=29, MAXFFTB=512 $END
IMPEDANCE.DAT
RESONANCE.DAT
```

in which the cubic polynomial coefficients for the wall impedance are read from file impedance.dat, specified after the namelist, while the resonance values are obtained from the file resonance.dat.

The impedance file read is performed only if NZ $\neq$ 0, while the resonance file read is performed only if NR $\neq$ 0. No files need be specified if the values are 0. Note that the number of protons specified is for 1/FRAC of the machine circumference. The number of bins used in the FFT will be the minimum of MAXFFTB and an upper limit determined by the cutoff frequency of the beampipe.

2.2.14 F Command - Fourier Transform

The members of NAMELIST /FFT/ are read in subroutine FOURFIT and stored in COMMON /FOURIR/.

F Command, Namelist /FFT/

Variable	Default Value	Default Unit	Description
NBINFFT	256	-	Number of bins to be used in FFT.
FFTON	F	-	Activate Fourier transform calculation.
NNF	0	-	Number of fourier harmonics to be stored in history.
NF(1:NNF)	0	-	Harmonic numbers of fourier spectrum components to be stored.
MFFT	1	-	Frequency of Fourier transform calculation.
FFTOUT	F	-	If TRUE, Fourier transform is printed.

Many of the variables in NAMELIST /FFT/ are shared by NAMELIST /SCHG/, due to the fact that the Fourier transform is utilized in much of the space-charge calculation. The F command alone is intended to allow the user to examine the Fourier transform of the distribution, and to follow the development of selected harmonics of the distribution. If, for example, a user had set up a distribution encompassing 1/7th (FRAC = 7) of the machine circumference, and wanted to track the turn-by-turn development of the first five odd Fourier harmonics of the distribution, then the appropriate F command would be:

```
$FFT  FFTON=T, NBINFFT=32, NNF=5, NF=1,3,5,7,9 $END
```

If at some point the user should wish to plot the record for the amplitude of the third harmonic, then the history command should be issued to retrieve FAMPL(2)[7]. The harmonics of the distribution should be multiplied by the periodicity in determining harmonics of the revolution frequency. If, for example, FRAC = 7, the fifth harmonic of the distribution corresponds to harmonic thirty-five for the entire circumference.

2.2.15 0-9 Commands - Call User-Written SHAZAM Routines

The SHAZAM facility is intended to allow users to integrate their own routines into the code. All COMMON blocks in ESME are made available to SHAZAM. As indicated in section 2.2.5, SHAZAM entry points may be called following every iteration of the difference equations using the TRAP option provided through NAMELIST /CYCLP/. In addition, SHAZAM routines may be called explicitly using commands 0-9, just as with other commands. Of course, any subsequent input is up to the author of the code following the appropriate SHAZAM entry point. It may be NAMELIST-directed, as ESME largely is, or there may be no input at all. The user is cautioned, however, to ensure that any subsequent input data in the "command" file for ESME is read, as

[7]See Section 2.2.8, History command.

ESME will read the line in that file where it finds itself after returning from any SHAZAM routine and attempt to interpret it as a command.

2.2.16 S Command - Save Tracking Parameters

The S command directs subroutine SAVE to write all data in COMMON blocks to an external file. It allows the user to suspend tracking at any point in the program. Once all the tracking data is SAVE'd, it may be restored simply by issuing the G command (see next section). Subroutine SAVE is also useful for those who might want to analyze ESME tracking data independently of the program[8]. The format of the S command is as follows:

```
S
FILENAME.EXT
```

where FILENAME.EXT is the name of the external file to which the data is to be written.

2.2.17 G Command - Get Tracking Parameters

The G command directs subroutine GET to read data from an external file into ESME's common block data. It is intended to be used to retrieve data written using the S command. The G command format is very similar to that for the S command:

```
G
FILENAME.EXT
```

where FILENAME.EXT is the name of the external file to from which the data is to be read.

[8] See, for example, Appendix A.

Chapter 3

Using The Program

3.1 Running the Program

As configured, ESME may be run from a terminal or in batch mode[1]. Basically, the process of running ESME consists of associating file names with FORTRAN logical units and then issuing the proper directive to execute the program. FORTRAN logical unit 5 is expected to supply the input. An appropriate DI-3000 device driver[2] must be specified in order to use the graphical output routines available for ESME at Fermilab. By default, ESME chooses the device driver[3] associated with node 1 to direct output, though others may be explicitly specified[4]. In addition, graphical output may be written to a Metafile, and that Metafile may be used in conjunction with a translator to view the output on various devices.

The following DCL commands might typically be used in running the program:

```
$ ASSIGN TESTCASE.DAT FOR005
$ ASSIGN ESMEPOSTP.DAT FOR018
$ ASSIGN ESMEHIST.DAT FOR009
$ SETDRV QMP
$ RUN ESME
```

They specify that ESME's input (the commands) will be read from file TESTCASE.DAT, that any post-processor output[5] will be written to file ESMEPOSTP.DAT, while history records will be written to the file ESMEHIST.DAT. In addition, the device driver QMP is assigned to node 1. Of course, it is possible to run ESME interactively, in which case the commands and namelists would be issued by the user directly. However, the batch mode of running the program is probably the preferred one. A user might decide to associate input from his terminal with some FORTRAN unit addressed through a SHAZAM routine, in which case the program can be run interactively through that channel.

Once a run is completed, ESME may have produced a number of files. If a printer driver were used for graphical output, then a graphics file will be generated which may be printed. If IDEV[6] is set to 0, then a Metafile is produced, which may be used in conjunction with a translator to

[1] Some of the material that follows is specific to the graphics routines described in this document, and may not be applicable to every user.

[2] DI-3000 is a trademark of Precision Visuals, Inc.

[3] SETDRV command at Fermilab.

[4] See IDEV in chart in Section 2.2.4, Graphical Output.

[5] See Appendix A.

[6] See Section 2.2.4, Graphical Output.

generate output on physical devices. Finally, if the post-processor option[7] is in effect, then ESME writes its data to the file associated with FORTRAN logical unit 18 (every MPLOT turns, or upon issuance of the D command). History records are written the file associated with unit logical unit 9. These files may be processed later according to methods of the user's own choosing. A simple post-processor "shell" has been written which processes these files using routines from ESME. It is described in Appendix A.

For most applications, command procedures have been prepared which automate the process of running the program and obtaining output in a VAX-based environment. A command procedure (USESME.COM) has been prepared which defines a number of logicals and symbols which make it possible to use ESME's programs and procedures without explicit knowledge of where they are located. On node ALMOND at Fermilab, for example, this command procedure can be invoked by entering

```
@USR$DISK4:[ESME_FILES]USESME
```

Now the user has merely to prepare an input data file, and then enter the command:

```
RESMEVAX
```

to run the current generic version of ESME on the local VAX. The command procedure accomplishes its task by

1. Asking the user for an input data file.
2. Creating a subdirectory into which all of ESME's output will go.
3. Copying the command input file and any other specified files into this subdirectory.
4. Asking the user whether graphical output is to be printed and where.
5. Submitting the job to a batch queue.

Procedures have also been written which automate the process of running ESME on other computers with the VAX serving as a front-end and post-processor. See Appendix B for a description of their use.

3.2 Input Structure

In this section the format required of various input files is described. First and foremost is FORTRAN logical unit 5, which is the main input file for ESME (and often the only input file). In addition, ESME may read tables of values for the RF voltage, frequency, or phase as functions of time, and the wall impedance as a function of frequency. Finally, ESME may read a file containing the parameters for various high-Q resonances.

3.2.1 The Command File

The commands which ESME accepts have already been described in Chapter Two. Each command is indicated by a single character in the first column of a line. The following four places in the line are discarded, and the rest of the line is read[8] and echoed in the output along with the command as a comment. Following many of the commands is a namelist read. The namelist input is specified by entering a dollar sign in the *second* column of the input file, followed immediately

[7] POSTP=T in the O command.

[8] FORTRAN format (A1,4X,A74).

by the namelist identifier, after which any variable which is a member of the namelist may be entered by specifying its name followed by an equal sign followed by the value to be assigned. The assignment of logical variables will depend on the particular machine (e.g. .TRUE. or T). The namelist is terminated by a $END. All of the command examples given previously are valid VAX namelist entries.

3.2.2 The Voltage Table

In certain circumstances in which the time development of the RF voltage is rather involved, it may be advisable to provide ESME with a table describing the time-dependence of the voltage, from which the values of the voltage at a particular time can be interpolated. Such a table is read from the file specified on the line immediately following the /RF/ namelist if any of the NRF rf variations (voltage, frequency or phase) is declared to be of type 4[9]. The file is opened, assigned to FORTRAN unit 10, and read until a line specifying "Source i" is read, where i corresponds to the index of the source for which the tables are specified. The format of the input file will appear as:

Source i
Voltage
n_i
t_1^i, v_1^i
t_2^i, v_2^i
...
Frequency
n_i
t_1^i, f_1^i
t_2^i, f_2^i
...
Phase
n_i
t_1^i, ϕ_1^i
t_2^i, ϕ_2^i
...Source j
Voltage
n_j
t_1^j, v_1^j
...

The order of sources (for more than one source) is arbitrary, however the input must be ordered so that $t_2^i > t_1^i$. The entry n_i following the "Source i" line indicates how many sets (i.e., lines) of values are to follow. It is not necessary to specify all three tables for a given source. It is also not necessary to specify voltage, frequency and phase tables in one place under a single "Source" heading. It is necessary, however, that if they are specified in one place, that they are ordered in the sequence: voltage, frequency, phase. Only one rf table file may be read in a given run of the program, and the entire file is read and its entries stored at one time. Therefore, even tables which are not to be used until later in the run must be provided at the time the file is read.

The table entries then determine the voltage, if an rf curve type of "4" is specified, using cubic spline interpolation between the provided values, and the supplied end points. For example, at

[9]See Section 2.2.2, the A Command.

time t, the voltage v for source i will be computed according to

$$\begin{aligned}
&\text{for } t < \text{TVBEG(I)}: \\
&\qquad v = vi \\
&\text{for TVBEG(I)} \le t \le \text{TVEND(I)}: \\
&\qquad t_k^i = \max t_j^i \ni t_j^i \le t \qquad 1 \le j \le n_i \\
&\qquad v = vi + (vf - vi) * spline \\
&\text{for } t > \text{TVEND(I)}: \\
&\qquad v = vf
\end{aligned}$$

where vf and vi are the specified voltages for source i at times TVBEG(I) and TVEND(I), and *spline* is the value interpolated from the table read for voltage source i. Note that the values supplied in the table(s) are *not* voltages, frequencies, or phases, but rather the fraction of those quantities between the specified initial and final points. Thus, in general, the table will run from 0.0 to 1.0. Note also that the interpolation procedure will only be carried out for TVBEG(I) $\le t \le$ TVEND(I); this is consistent with the convention for all methods of varying the voltage.

3.2.3 The Impedance Table

The space-charge option in ESME allows the user to enter a cubic polynomial coefficient table for the wall impedance. This file, if required[10], must be listed on the line immediately following the space charge command namelist input. The file will be opened, assigned to FORTRAN unit 11, and NZ values of frequency (in MHz) will be read, each accompanied by eight polynomial coefficients. The format of the input is as follows:

f^1
$x_1^1, x_2^1, x_3^1, x_4^1$
$y_1^1, y_2^1, y_3^1, y_4^1$
...
f^2
$x_1^2, x_2^2, \ldots$

The first value on the top line is the frequency in MHz. The four values following on the next line are the coefficients for the real part of the impedance. The four values on the third line are the coefficients for the imaginary part of the impedance. The FORTRAN format for each line is list-directed, so it is sufficient to simply list the values separated by commas or spaces in the file. The entries should be made in order of increasing frequency ($f^1 < f^2 < \ldots < f^{NZ}$). The coefficients are used to compute the value of the impedance at a given frequency f, according to:

$$\begin{aligned}
&\text{if } f^1 \le f < f^{NZ}: \\
&\qquad f^n = \max f^i \ni f > f^i \qquad 1 < i < \text{NZ} \\
&\qquad \Delta f = f - f^n \\
&\qquad Z_r + iZ_i = x_1^n + \Delta f x_2^n + \Delta f^2 x_3^n + \Delta f^3 x_4^n + i(y_1^n + \Delta f y_2^n + \Delta f^2 y_3^n + \Delta f^3 y_4^n) \\
&\text{if } f < f^1: \\
&\qquad Z_r + iZ_i = x_1^1 + iy_1^1
\end{aligned}$$

[10] For NZ > 0, see Section 2.2.13.

if $f \geq f^{NZ}$:

$$Z_r + iZ_i = x^N Z_1 + iy^N Z_1$$

3.2.4 The Resonance Table

High-Q resonances may also be included in the impedance. The resonance values are read from the file whose name is specified following the /SCHG/ namelist[11], and the name of the impedance file, if any. This file is assigned to input unit 12. Each of the NR entries in the resonance file occupies a single line in the input file. The format for each line is list directed, where the values to be read in are:

frequency(MHz), magnitude(Ω), multiplier, Q

where the multiplier value simply multiplies the impedance which is calculated from the frequency, magnitude and Q values for the resonance.

[11] For NR > 0, see Section 2.2.13.

Chapter 4

Programming

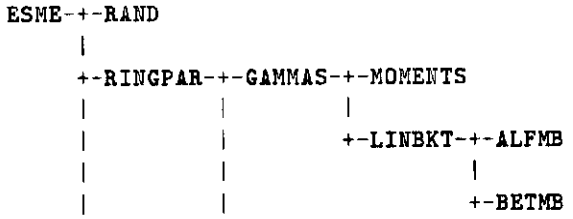
In this chapter some of the basic structure of the program is presented, and the use of certain VAX-based facilities in the development of the code is discussed. Though somewhat specialized, it is felt that the utility of these features to those who might be willing and able to use them is of great enough importance to warrant some description here. This chapter is primarily of interest to programmers. There is no need to make use of the material in this section in order to use the program. Furthermore, this section alone is not likely to make the reader an ESME expert. It is merely the hope of the authors that the material presented here will ease the process of incorporating changes and additions to the code.

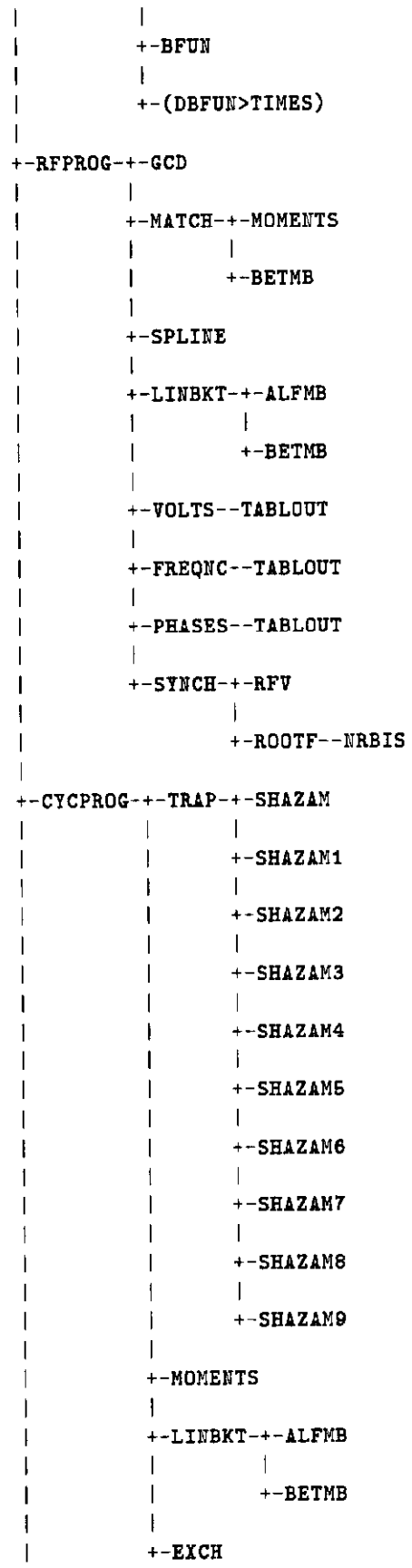
4.1 Program Basics

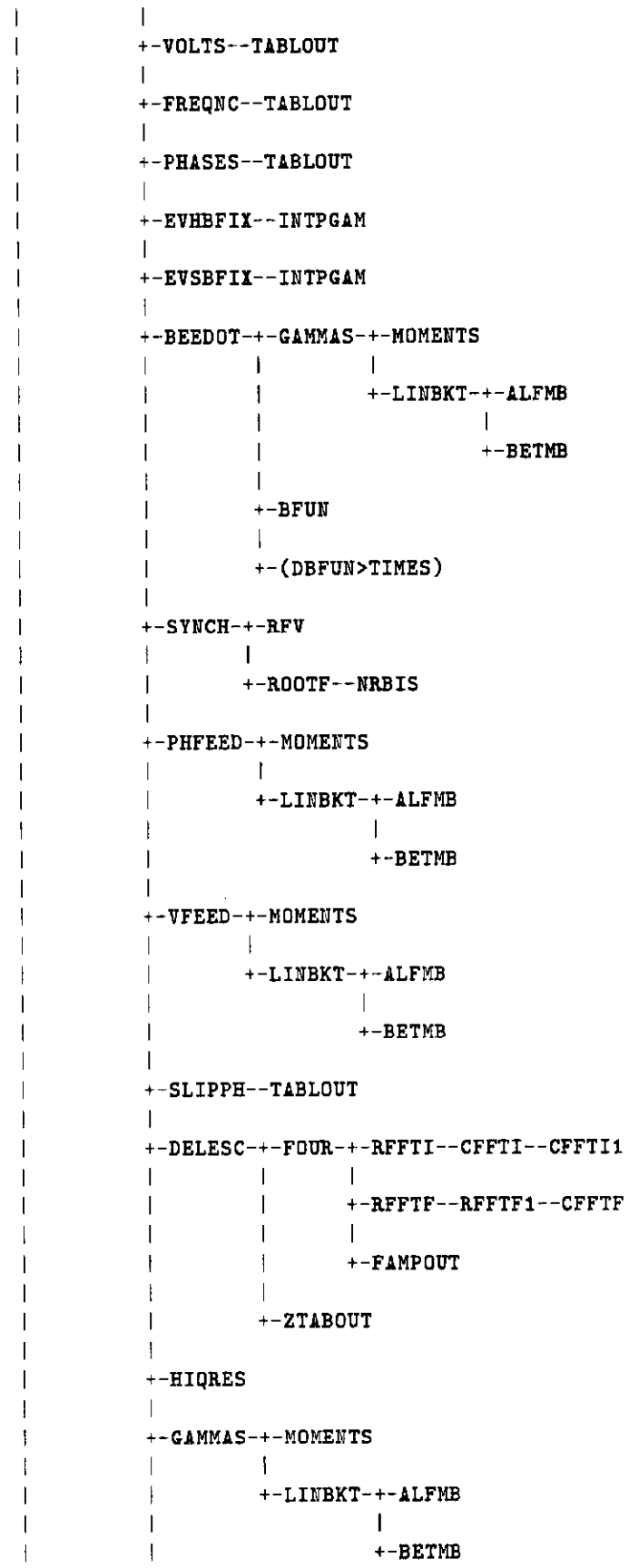
The first part of this section describes the overall structure of ESME. The second section details the main tracking loop. The third section lists the important variables in ESME, which are held in COMMON and thus available in SHAZAM.

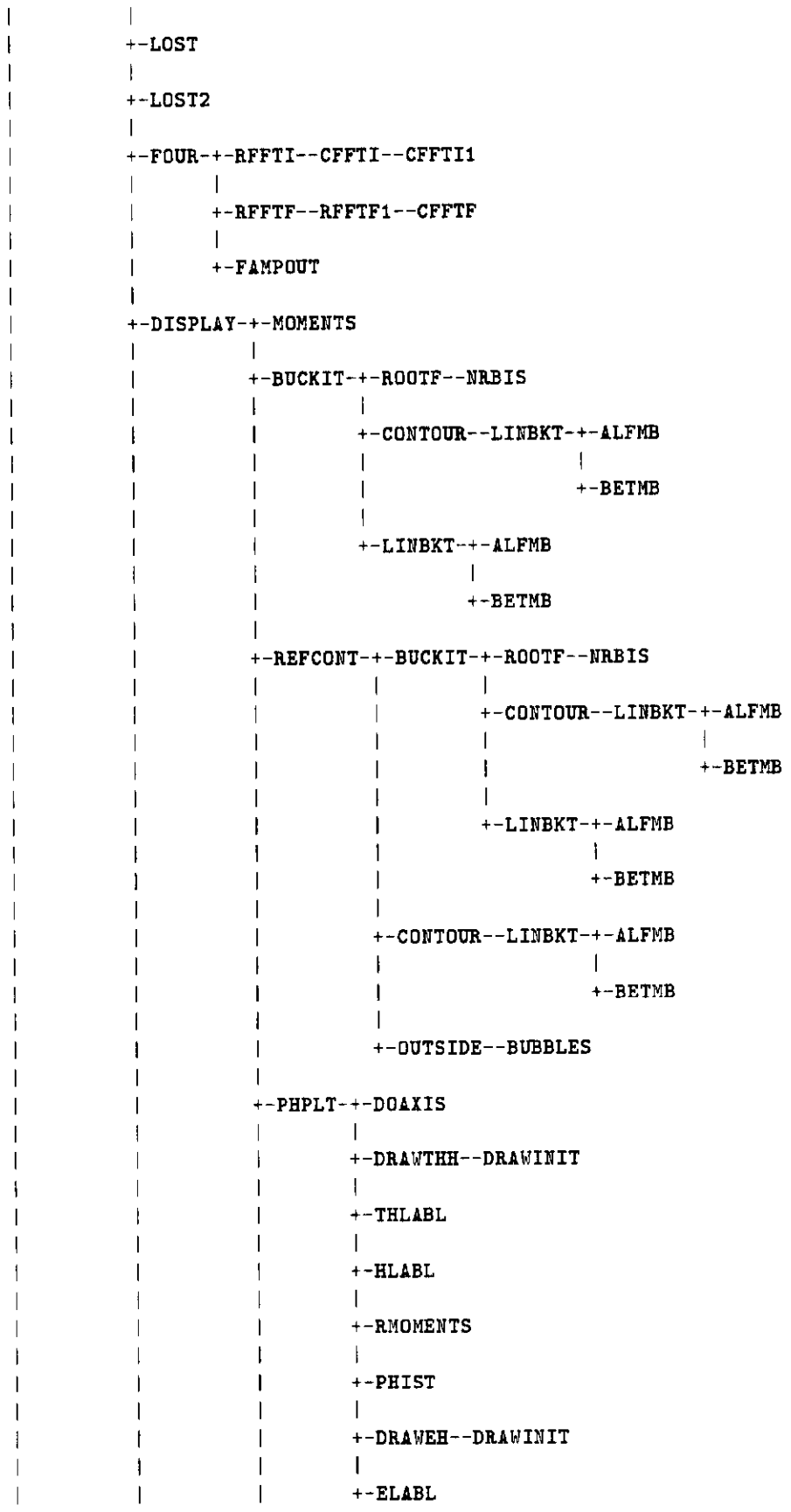
4.1.1 Program Structure

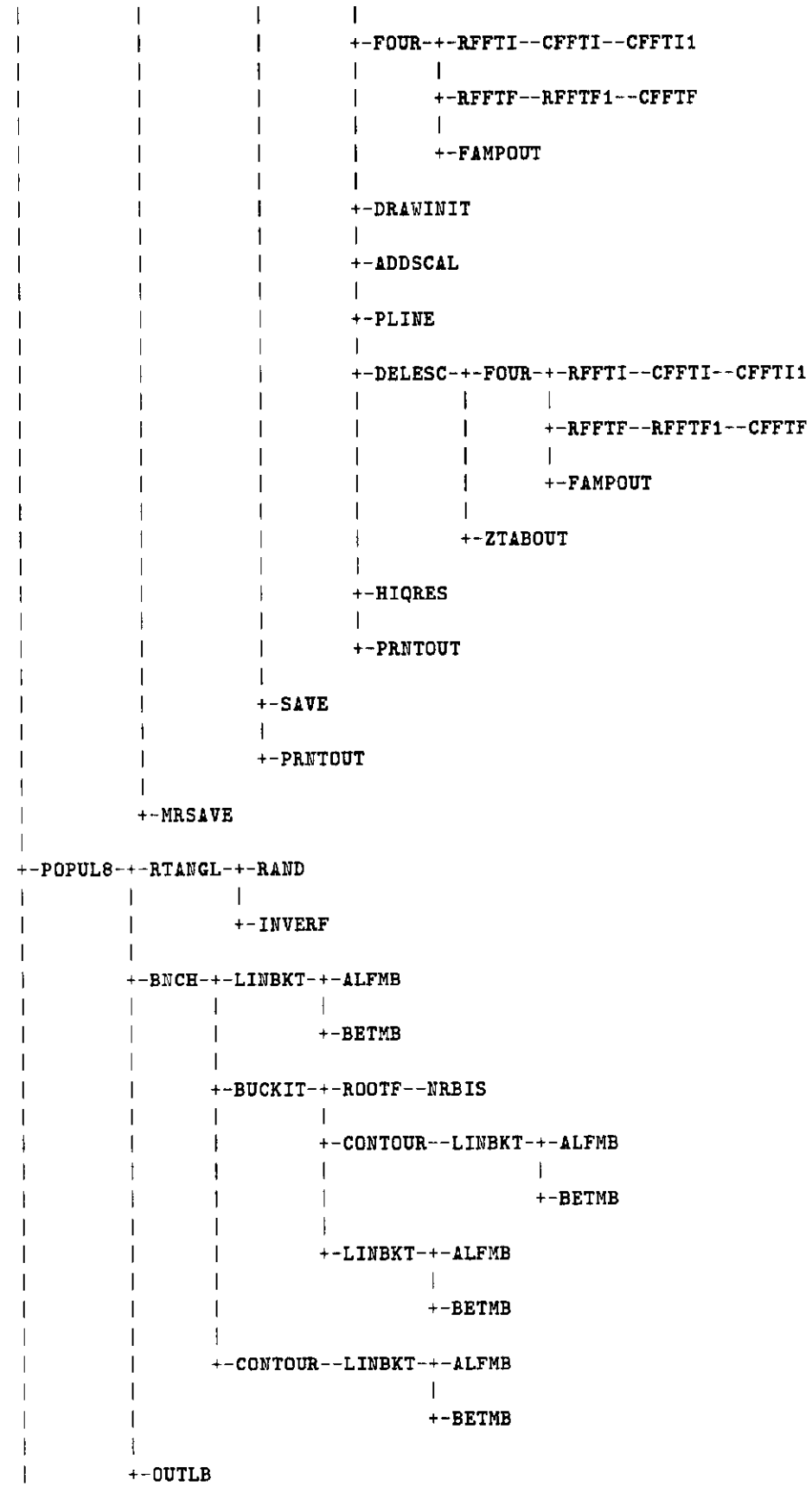
The basic pattern of ESME is a main program which calls subroutines selected by single letter commands in the input stream. The called routine reads in any needed parameters; it, and any dependent subroutines, carry out calculations for a distinct phase of the calculation or for a distinct accelerator subsystem. The program is integrated by putting particle coordinates and system variables into named commons each of which contain a group of closely related quantities. Higher level subroutines communicate through common. Certain lower level routines and some utility routines shared among different functional areas pass data through calling lists. Nearly all system and coordinate variables are stored in common blocks; only loop counters and a few intermediate results are local variables. Thus, the program is divided into numerous functional modules but important variables are global. The large number of parameters often required to specify a distribution and the rf systems which act upon it encourage this structure, as well as making it easier to incorporate changes into the code. A schematic tree diagram indicating the program flow follows.

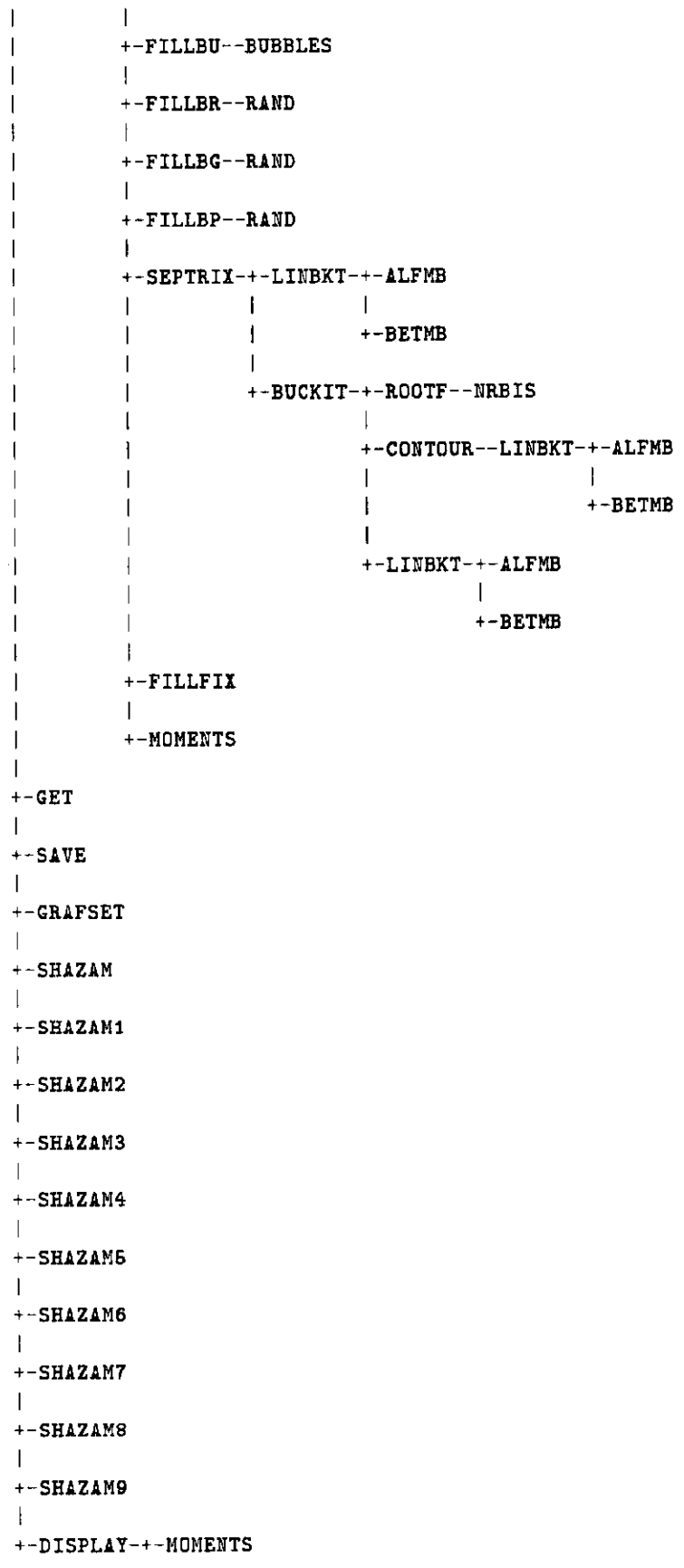


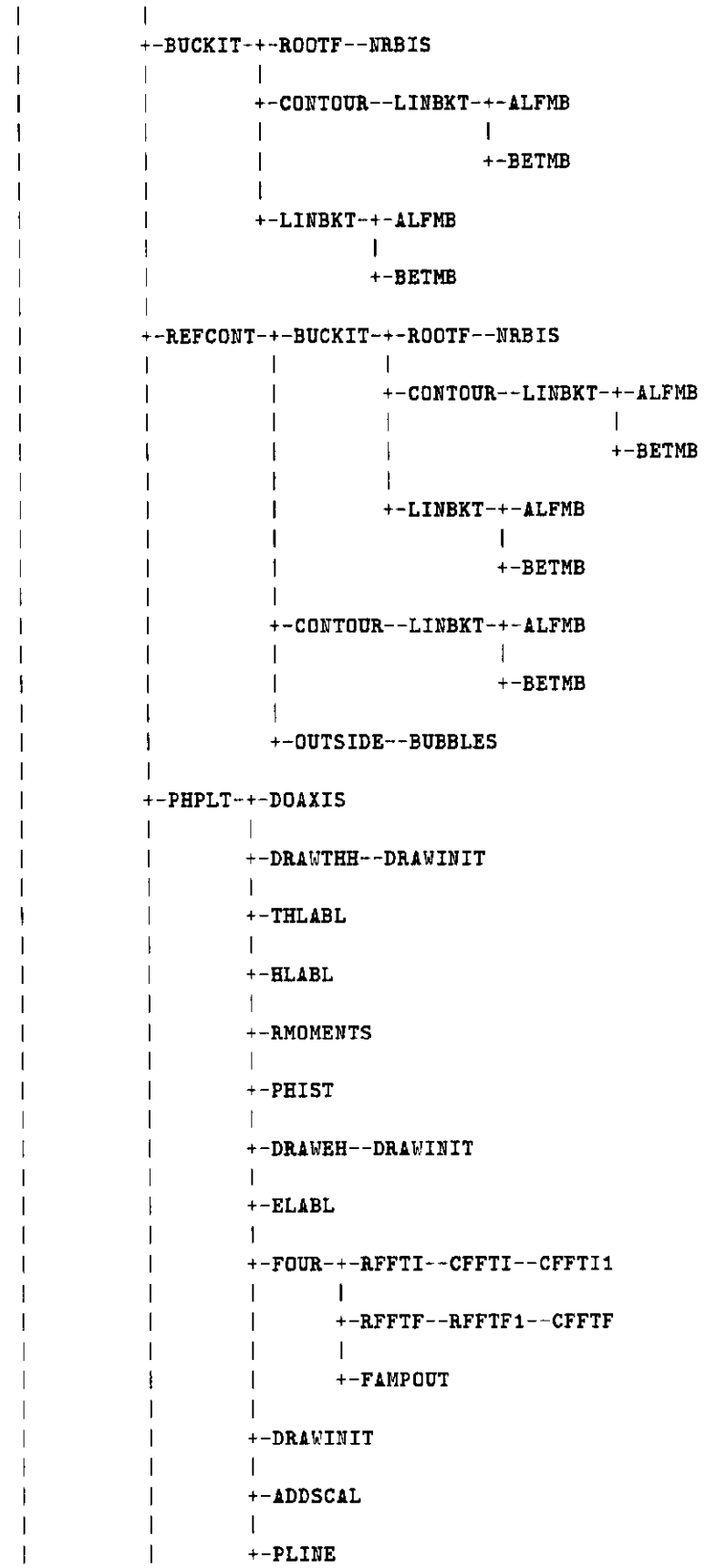

+-BFUN
+-(DBFUN>TIMES)
+-RFPROG-+-GCD
+-MATCH-+-MOMENTS
+-BETMB
+-SPLINE
+-LINBKT-+-ALFMB
+-BETMB
+-VOLTS--TABLOUT
+-FREQNC--TABLOUT
+-PHASES--TABLOUT
+-SYNCH-+-RFV
+-ROOTF--NRBIS
+-CYCPROG-+-TRAP-+-SHAZAM
+-SHAZAM1
+-SHAZAM2
+-SHAZAM3
+-SHAZAM4
+-SHAZAM5
+-SHAZAM6
+-SHAZAM7
+-SHAZAM8
+-SHAZAM9
+-MOMENTS
+-LINBKT-+-ALFMB
+-BETMB
+-EXCH


+-VOLTS--TABLOUT
+-FREQNC--TABLOUT
+-PHASES--TABLOUT
+-EVHBFIX--INTPGAM
+-EVSBFIX--INTPGAM
+-BEEDOT-+-GAMMAS-+-MOMENTS
+-LINBKT-+-ALFMB
+-BETMB
+-BFUN
+-(DBFUN>TIMES)
+-SYNCH-+-RFV
+-ROOTF--NRBIS
+-PHFEED-+-MOMENTS
+-LINBKT-+-ALFMB
+-BETMB
+-VFEED-+-MOMENTS
+-LINBKT-+-ALFMB
+-BETMB
+-SLIPPH--TABLOUT
+-DELESC-+-FOUR-+-RFFTI--CFFTI--CFFTI1
+-RFFTF--RFFTF1--CFFTF
+-FAMPOUT
+-ZTABOUT
+-HIQRES
+-GAMMAS-+-MOMENTS
+-LINBKT-+-ALFMB
+-BETMB


+-LOST
+-LOST2
+-FOUR-+-RFFTI--CFFTI--CFFTI1
+-RFFTF--RFFTF1--CFFTF
+-FAMPOUT
+-DISPLAY-+-MOMENTS
+-BUCKIT-+-ROOTF--NRBIS
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-LINBKT-+-ALFMB
+-BETMB
+-REFCONT-+-BUCKIT-+-ROOTF--NRBIS
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-LINBKT-+-ALFMB
+-BETMB
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-OUTSIDE--BUBBLES
+-PHPLT-+-DOAXIS
+-DRAWTHH--DRAWINIT
+-THLABL
+-HLABL
+-RMOMENTS
+-PHIST
+-DRAWEH--DRAWINIT
+-ELABL


+-FOUR-+-RFFTI--CFFTI--CFFTI1
+-RFFTF--RFFTF1--CFFTF
+-FAMPOUT
+-DRAWINIT
+-ADDSCAL
+-PLINE
+-DELESC-+-FOUR-+-RFFTI--CFFTI--CFFTI1
+-RFFTF--RFFTF1--CFFTF
+-FAMPOUT
+-ZTABOUT
+-HIQRES
+-PRNTOUT
+-SAVE
+-PRNTOUT
+-MRSAVE
+-POPUL8-+-RTANGL-+-RAND
+-INVERF
+-BNCH-+-LINBKT-+-ALFMB
+-BETMB
+-BUCKIT-+-ROOTF--NRBIS
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-LINBKT-+-ALFMB
+-BETMB
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-OUTLB


+-FILLBU--BUBBLES
+-FILLBR--RAND
+-FILLBG--RAND
+-FILLBP--RAND
+-SEPTRIX-+-LINBKT-+-ALFMB
+-BETMB
+-BUCKIT-+-ROOTF--NRBIS
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-LINBKT-+-ALFMB
+-BETMB
+-FILLFIX
+-MOMENTS
+-GET
+-SAVE
+-GRAFSET
+-SHAZAM
+-SHAZAM1
+-SHAZAM2
+-SHAZAM3
+-SHAZAM4
+-SHAZAM5
+-SHAZAM6
+-SHAZAM7
+-SHAZAM8
+-SHAZAM9
+-DISPLAY-+-MOMENTS


+-BUCKIT-+-ROOTF--NRBIS
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-LINBKT-+-ALFMB
+-BETMB
+-REFCONT-+-BUCKIT-+-ROOTF--NRBIS
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-LINBKT-+-ALFMB
+-BETMB
+-CONTOUR--LINBKT-+-ALFMB
+-BETMB
+-OUTSIDE--BUBBLES
+-PHPLT-+-DOAXIS
+-DRAWTHH--DRAWINIT
+-THLABL
+-HLABL
+-RMOMENTS
+-PHIST
+-DRAWEH--DRAWINIT
+-ELABL
+-FOUR-+-RFFTI--CFFTI--CFFTI1
+-RFFTF--RFFTF1--CFFTF
+-FAMPOUT
+-DRAWINIT
+-ADDSCAL
+-PLINE

```
|          |        |
|          |        +-DELESC-+-FOUR-+-RFFTI--CFFTI--CFFTI1
|          |        |        |      |
|          |        |        |      +-RFFTF--RFFTF1--CFFTF
|          |        |        |      |
|          |        |        |      +-FAMPOUT
|          |        |        |
|          |        |        +-ZTABOUT
|          |        |
|          |        +-HIQRES
|          |        |
|          |        +-PRNTOUT
|          |
|          +-SAVE
|          |
|          +-PRNTOUT
|
+-HISTORY-+-GLABL
|         |
|         +-GETDAT
|
+-FFTSET--FOUR-+-RFFTI--CFFTI--CFFTI1
|              |
|              +-RFFTF--RFFTF1--CFFTF
|              |
|              +-FAMPOUT
|
+-BEAMSC
|
+-LOWLVL-+-MOMENTS
|        |
|        +-LINBKT-+-ALFMB
|                 |
|                 +-BETMB
|
+-MRINIT
|
+-MRPLT--PLINE
```

This is a detached tree starting at the subprogram 'CFFTF'

```
CFFTF--CFFTF1-+-PASSF4
              |
              +-PASSF2
              |
              +-PASSF3
              |
              +-PASSF5
              |
              +-PASSF
```

Other routines (usually passed as arguments):

```
DRAWPH--DRAWINIT
```

```
BETPRMB
ALFPRMB
PREFIX
BVOLT
ENGFMT
DRFV
DBVOLT
```

4.1.2 Main Tracking Loop

The fundamental mapping or tracking algorithm is contained in the subroutine CYCPROG which is called when a **T** command is encountered in the data stream. The subroutine reads data for the number of turns to track *etc.* and then tracks successive turns applying the difference equations to each particle on each turn. At the end of each turn the tracking duration is checked, various properties of the distribution are calculated, system parameters are updated, and tests are applied to see if any selected parameter has reached a desired endpoint. The loop structure is schematized in Fig. 4.1.2. The boxed numbers in the figure are Fortran statement numbers relating the depicted structure to the Fortran listing. These markers should help the intensely curious or thoroughly sceptical find his way through the core code. An IF loop is depicted in the diagram by its head statement number in a box to the outside of a vertical line representing the scope of the loop; a DO loop is represented with its terminating number in a box at the end of the vertical line representing its scope. The names of loop counters are given within ovals. Steps of the calculation in italics are executed optionally according to logical switches and integer conditions set by the input data.

4.1.3 Important Variables

The various named common blocks are listed below. Each entry includes a statement of purpose and a list of all the included variables. If the variables come primarily from one subroutine, that subroutine is named on the last line of the entry; likewise, if a significant number of the variables come directly from the input data, the controlling NAMELIST is identified also.

```
C  Block parameterizing isolated or barrier bucket generation.
      PARAMETER (ISRC3 = 10)
      COMMON /BKTSUP/ THL(ISRC3),THU(ISRC3)
Principal Source: RFPROG                          Controlling NAMELIST: /RF/

C  THE MAIN STORAGE BLOCK. CONTAINS NPHASE PHASE SPACE COORDINATES AND
C  THE POINTERS TO SEPARATE PARTITIONS OF THE DISTRIBUTION
      PARAMETER (NPHASE = 50000, NKLIM=50)
      COMMON /BLANK/ PHASE(0:NPHASE,2)
      COMMON /IBLANK/ KOUNT,KLASSES,KLIMIT(0:NKLIM)
      COMMON /LBLANK/ PARTION
      LOGICAL PARTION
Principal Source: POPUL8                       Controlling NAMELIST: /POPL8/

C  BUCKET PARAMETERS TURN-BY-TURN
      COMMON /BUCKET/ PHISL,PHIUSFP,GNUS,SBCKT,HBCKT
      COMMON /IBUCKET/ NFIRST,NLAST
Principal Source: CYCPROG
```

```
C  BUNCH PARAMETERS TURN-BY-TURN
      COMMON /BUNCH/ THBAR,EBAR,THRMS,ERMS,EPSILON,SBUNCH
Principal Source: CYCPROG

C  CONTAINS MATHEMATICAL AND PHYSICAL CONSTANTS, PROGRAM CONSTANTS, AND
C  CONVERSION FACTORS FROM EXTERNAL TO INTERNAL UNITS
      COMMON /CONST/ HALFPIE,PIE,TWOPIE,FOURPIE,RADDEG,DEGRAD,BIG,SMALL,
     1     C,EMCSQ,EMCSQS,RP,QE,ZNAUGHT
Principal Source: BLOCK DATA

C  MISCELANEOUS PARAMETERS AND INTERMEDIATE RESULTS TURN-BY-TURN
      PARAMETER (KSRC = 10)
      COMMON /CURRENT/ EV(KSRC),PSIO(KSRC),PSI(KSRC),FREQ(KSRC),
     1     ES,PS,RS,BETAS,BETASQ,ETA,EO,PO,BETAO,ACCEL,
     2     THREF,EREF,TAU,TIME,DEBFLD,PDOT,ALFAO,ALFA1,ALFA2,
     3     ELO,EHI,EAVG,TGCURR,GMSQINV,DEO,DEBAR,DELEREF,DELR,
     4     RFFREQ,PHIS,DELES
      COMMON /ICURRENT/ M,MM,NTURNS,MHIST
      COMMON /LCURRENT/ TRPSW,TRANSW
      LOGICAL TRPSW,TRANSW
Principal Source: CYCPROG

C  PHASE SPACE COORDIATES OF BUCKET OR OTHER CONTOUR OF INTEREST
      PARAMETER (ICURVE = 5001)
      COMMON /CURVES/ CURVE(ICURVE,2),TURNPT(4,3),NC
Principal Source: BUCKET & REFCONT

C  PARAMETERS GOVERNING DURATION & OPTIONAL FEATURES OF THE TRACKING CALCULATION
      PARAMETER (NTRP = 4)
      COMMON /CYCLP/ TBEGIN,TEND,TSTART,TSTOP,TTRACK,DES,ACCELO,
     1     ETATRP,PHISTRP
      COMMON /ICYCLP/ MSTEP,LGRTHM,ITRAP(NTRP),MGRACE(NTRP),NTRAP,NCAV
      COMMON /LCYCLP/ HISTRY,MOMNTS,BBDRY,EAPROX
      LOGICAL HISTRY,MOMNTS,BBDRY,EAPROX
Principal Source: CYCPROG                          Controlling NAMELIST: /CYCLE/

C  PARAMETERS DEFINING FEEDBACK LOOPS (CURRENTLY PHASE & VOLTAGE - NO RADIAL)
      PARAMETER (IFBPRS = 1001)
      COMMON /FEEDS/ DLIMIT,PLIMIT,FBFACT,VLIMIT,
     1     VFBFCTR,FBPRS(IFBPRS),PSIADD,DAMPL,W(IFBPRS)
      COMMON /IFEEDS/ NTUAVG,NTURES,IFTB
      COMMON /LFEEDS/ PHFBON,VFBON,USEWT
      LOGICAL PHFBON,VFBON,USEWT
Principal Source: LOWLVL                           Controlling NAMELIST: /LLRF/

C  PARAMETERS DEFINING AND RESULTS OF FOURIER TRANSFORM OF CHARGE DISTRIBUTION
      PARAMETER (IFFT = 1024, NFMAX = 50)
```

```
      COMMON /FOURIR/ BINWIDT,FAMPL(NFMAX),FAZE(NFMAX),SCFRAC,SCTHLO,
     +   WSAVE(3*IFFT+15),F(IFFT),G(IFFT)
      COMMON /IFOURIR/ NBINFFT,MFFT,MAXFFTB,NF(NFMAX),NNF,NCC
      COMMON /LFOURIR/ FFTON
      LOGICAL FFTON
Principal Source: FOURFIT                          Controlling NAMELIST: /FFT/

C  Parameters defining transition gamma time dependence and
C  transition phase switch timing.
      COMMON /GAMJMP/ GAMPAR(5),ETAJMP
      COMMON /IGAMJMP/ KINDG
      COMMON /LGMAJMP/ GMAJMP
      LOGICAL GMAJMP
Principal Source: RINGPAR                          Controlling NAMELIST: /RING/

C  PARAMETERS DEFINING DESIRED GRAPHICAL OUTPUT
      PARAMETER (ISIZGM = 10000, ISIZPH = 2000)
      COMMON /GRAFIX/ THPMIN,THPMAX,DEPMIN,DEPMAX,DTHCURV,DECURV,
     1     THBMIN,THBMAX,EBMIN,EBMAX,
     2     HBCKTN,SBCKTN,SREF,
     3     SCBMIN,SCBMAX,RBMIN,RBMAX,DELCON
      COMMON /IGRAFIX/ MPLOT,IOPT,IEREF,NBINTH,NBINE,KNTLIM,
     1     IGMARY(ISIZGM),IPHARY(ISIZPH),IFBMIN,IFBMAX,IDEV,NPJMP,
     2     KLPLOT
      COMMON /LGRAFIX/ PLTSW(20),TITLE,MTCHSI,MTCH95,DRWREF,NODRAW,
     1     POSTP
      LOGICAL PLTSW,TITLE,MTCHSI,MTCH95,DRWREF,NODRAW,POSTP
Principal Source: GRAFSET                          Controlling NAMELIST: /GRAPH/

C  DESCRIPTIVE HEADING FOR GRAPHICAL OUTPUT
      PARAMETER (MAXTTL=50)
      COMMON /HEADING/ TITL
      COMMON /IHEADING/ TITLEN
      CHARACTER TITL*50
      INTEGER TITLEN
Principal Source: GRAFSET

C  Include specifying mountain range parameters.
C  For saving data (plotting parameters are local to MRPLT).
      COMMON /MRANGE/ MRTHMINB,MRTHMAXB,TMBEGIN,TMEND
      REAL MRTHMINB,MRTHMAXB,TMBEGIN,TMEND
      COMMON /IMRANGE/ MRMPLOT,MRNBIN
      INTEGER MRMPLOT,MRNBIN
Principal Source: MRINIT                           Controlling NAMELIST: /MRANGE/

C  THE PARAMETERS DEFINING THE INITIAL PHASESPACE DISTRIBUTION(S)
      COMMON /POPLATE/ THMIN,THMAX,REMIN,REMAX,
     1     SBNCH,THOFF,EOFF,THTRAN,ETRAN
```

```
      COMMON /IPOPLATE/ KIND,IPOP,NTH,NE,NPOINT,ISEED
Principal Source: POPUL8                          Controlling NAMELIST: /POPL8/

C  PARAMETERS DEFINING THE RF SYSTEMS
      PARAMETER (NSRC = 10)
      COMMON /RFP/
     1     VI(NSRC),VF(NSRC),TVBEG(NSRC),TVEND(NSRC),VTABL(5,21,NSRC),
     2     FRI(NSRC),FRF(NSRC),TFBEG(NSRC),TFEND(NSRC),FTABL(5,21,NSRC),
     3     PSII(NSRC),PSIF(NSRC),TPBEG(NSRC),TPEND(NSRC),
     4     PTABL(5,21,NSRC),C1(NSRC),C2(NSRC),DELTRF(NSRC),
     5     HDECR,SDECR,PHISLIM
      COMMON /IRFP/ NRF,H(NSRC),HW(NSRC),HGCD,HMAX,ISYNC,
     1     KURVE(NSRC),NTV(NSRC),NTABV(NSRC),
     2     KURVF(NSRC),NTF(NSRC),NTABF(NSRC),
     3     KURVP(NSRC),NTP(NSRC),NTABP(NSRC)
      COMMON /LRFP/ VKON,FRKON,PHKON,PHSLIP,EXCHRF,HOLDBH,HOLDBA,CNTINU,
     1     VMATCHI(NSRC),VMATCHF(NSRC)
      INTEGER H,HW,HGCD,HMAX
      LOGICAL VKON,FRKON,PHKON,PHSLIP,VMATCHI,VMATCHF
      LOGICAL EXCHRF,HOLDBH,HOLDBA,CNTINU(NSRC)
Principal Source: RFPROG                          Controlling NAMELIST: /RFP/

C  THE LATTICE PARAMETERS INCLUDING TIME DEPENDENCES
      COMMON /RINGP/ REQ,GAMMAT,ALPHA0,ALPHA1,ALPHA2,ALPHA3,
     1     TAUINF,EKOI,EKOF,EKS,TI,TF,PI,PF,EKIDOT,EKFDOT,
     2     PIDOT,PFDOT,THL0,THHI,THRNG,FRAC,PIPRAD
      COMMON /IRINGP/ KURVEB
      COMMON /LRINGP/ JNRAMP,EBDRY
      LOGICAL JNRAMP,EBDRY
Principal Source: RINGPAR                         Controlling NAMELIST: /RING/

C  AREA TO BE USED WITHIN A SINGLE SUBROUTINE ON A GIVEN CALL (MOSTLY GRAPHICS)
      COMMON /SCRATCH/ SCRPAD(4422)

C  Contains "spare" variables which are written to history
C  along with everything else.  These can be whatever the user
C  wishes; these commons are available to CYCPROG and SHAZAM.
C  Also contains labels for spares used in history plots.
      PARAMETER (NSPARE = 10, TITLL=14, UNITL=7)
      COMMON /SPARES/ SPARE(NSPARE)
      COMMON /CSPARES/ SPLABL(NSPARE),SPUNIT(NSPARE)
      CHARACTER*14 SPLABL
      CHARACTER*7 SPUNIT
Principal Source: CYCPROG

C  PARAMETERS DEFINING THE SPACE CHARGE & WALL IMPEDANCE ENERGY/TURN
      PARAMETER (ITBOUT=40, IRLLEN=30)
      COMMON /SPCHG/ A,B,ENQ,ZTABL(ITBOUT,9),EVSC(1024),
```

```
     1     RESTBL(IRLLEN,4)
      COMMON /ISPCHG/ NZ,NR,MSC
      COMMON /LSPCHG/ SCON
      LOGICAL SCON
Principal Source: FOURFIT                          Controlling NAMELIST: /SCHG/

C  A BLOCK TO CONTAIN CPU TIME SINCE START
      COMMON/TIMES/ CPUBEG,CPUNOW
      COMMON/ITIMES/ ITIME,IRCOD1,IRCOD2
      DOUBLE PRECISION CPUBEG, CPUNOW

C  Record the version number for program documentation & debugging.
C  VERNUM is the version of ESME
C  PVER is the post-processor version number.
C  Version 7.10 (beta) 1 DEC 1989
      PARAMETER (VERNUM=7.10, PVER=1.00)

C  Commons for voltage due to resonator.
      PARAMETER (MAXRB=10000, MAXRES=30)
      COMMON /ZRES/ VRES(MAXRB),RVO(MAXRES),RVODOT(MAXRES),
     +              DQDTO(MAXRES)
      COMMON /IZRES/ NBRES
      COMMON /LZRES/ QREZON
      LOGICAL QREZON
Principal Source: FOURFIT
```

4.2 Code Management

The current version of ESME is largely an outgrowth of efforts to tailor the program to specific applications. The various efforts embarked upon to adapt the code for different purposes revealed that the code had diverged sufficiently enough from its original intent that a systematic overhaul was called for. The aims of this overhaul were to enhance the capabilities of the program somewhat, to clarify both the overall structure of the code and its command structure, and to make the process of adapting the code for specific problems slightly more appealing (less painful). The tools presented here were developed specifically to aid in the process of coding. The following two sections detail these tools and their use. It is not necessary to read the first section to make use of these tools, so the impatient reader may skip immediately to Section 4.2.2

4.2.1 Tools–An Overview

The development process was aided by VAX DEC/Module Management System (MMS) in conjunction with VAX DEC/Code Management System (CMS)[1], and Fermilab's CDF EXPAND[13] utility. CMS was used to maintain a source-code library for ESME. MMS was used to construct the executable version of the code.

MMS is patterned after the UNIX[2] *make* utility. At the heart of MMS is a description file, which MMS processes to determine what actions are necessary to produce an up-to-date executable

[1]VAX, DEC/MMS, and DEC/CMS are trademarks of Digital Equipment Corporation.

[2]UNIX is a registered trademark of American Telephone and Telegraph Company.

version of the "target", which in the case of ESME is an executable version of the program. The description file consists of dependency "rules" and action lines. Each dependency rule consists of a target and its sources (e.g., A.EXE and A.OBJ). MMS compares the dates of the targets and sources to determine if the target is older than any of the sources. If so, the target needs to be "updated", and in that case the action specified on the action line is executed. A simple example of a description file for ESME might be:

```
ESME.EXE : ESME.OBJ
           LINK ESME
ESME.OBJ : ESME.FOR
           FORTRAN ESME
```

in which all the source code for ESME is found in ESME.FOR. Of course, the description given here is incomplete. However, the underlying principle itself is very simple– that of updating only those system components which it is necessary to update to build a new target. Using this system, the process of code development is simplified, since MMS can "pick up" any changes to the common CMS library. In addition, individual users can incorporate their own changes into the code very easily.

As mentioned earlier, the Fermilab utility EXPAND was used with MMS to build the program. Had ESME been intended only for use on VAX systems, EXPAND could have been dispensed with. This utility allows the user to build the program for various machine architectures[3] from the same cource code, in addition to offering various pre-processor options. Of course, in cases where the compiler for the machine architecture does not reside in the evironment (i.e., VAX) in which the code is prepared, MMS cannot direct the VAX to compile and link the code. However, presumably, once the source code has been prepared for the destination machine, it can be exported for compilation and linking there.

4.2.2 Using the Tools

The relevant command procedures and MMS files are found in `USR$DISK4:[ESME_FILES.MAKE]`. Generally, the user should first SET the appropriate CMS library and FETCH any routines to be modified. The addition of any routines which are not part of ESME will require the user to modify some of the files in `USR$DISK4:[ESME_FILES.MAKE]`. Such operations will not be discussed here, though an examination of the relevant MMS (INC.MMS, ESME.MMS, ESMESRC.MMS) files should provide some illumination as to how to proceed in such a case.

Once the appropriate files have been modified, the VMS user may incorporate them into an executable version of ESME using the prepared command procedure MAKESME.COM. This procedure then prepares either an executable or source-code version of the program using the routines supplied by the user while supplying the remainder of the routines from ESME libraries. If the procedure USESME.COM has been invoked, it should be sufficient to simply enter the command `MAKESME`. The command procedure will then

1. Ask the user if he wishes to make ESME, ESMEPLOT, or both.
2. Ask whether the output is to be executable or source code.
3. Ask for a location in which to place the output; if one is not already available, then it will be created.

[3] At present, EXPAND accepts the arguments VAX, CYBER, IBM, ACP_NODE, FTN77, and FPS for the /ENVIRONMENT qualifier.

4. Query the user for source files (presumably modified).
5. Query the user for pre-processor and/or compile options.
6. Submit the job to a batch queue.

The resultant executable (assuming no compile time errors) or source code file will be constructed in the specified directory. Note that the prepared procedures for running ESME will not use this version by default. If you've gotten this far, though, you probably know what to do.

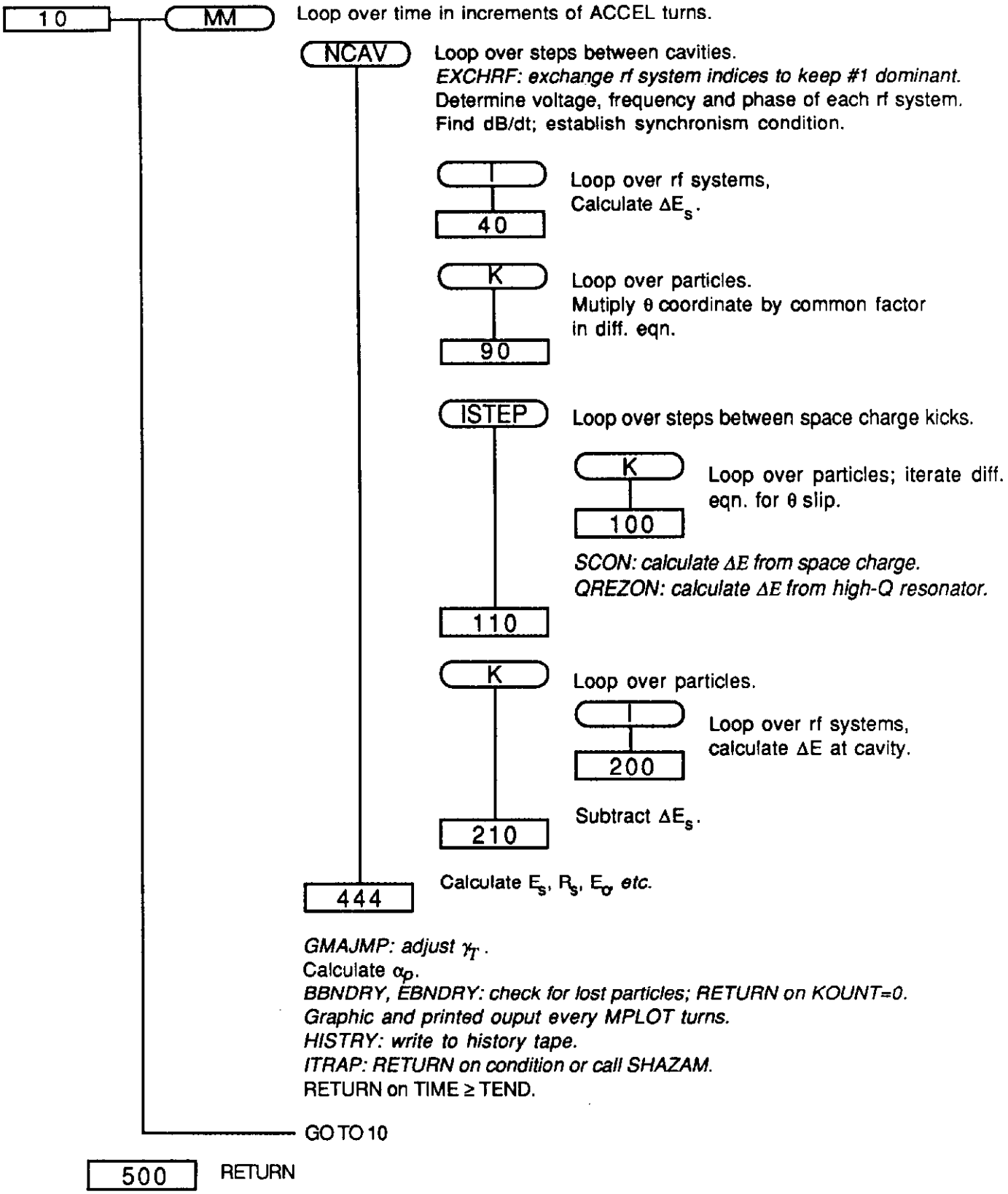


Figure 4.1: Schematic representation of the main tracking loop in ESME

Appendix A

Post-Processing

For those users who wish to process ESME data independently of the program, a post-processor option is provided[1] When this option is in effect, any plotting routine calls are substituted for by writes of ESME's common blocks (containing essentially all of the information about the current state of the simulation) to FORTRAN unit 18 using subroutine SAVE. Later, this file may be read using subroutine GET. The code for a graphics post-processor is appended here as an example. The plotting routines employed here are the same ones imbedded in ESME. This command "shell" is modelled after that of the main program. As in ESME, one-letter commands initiate various routine calls and namelist reads. Since the plotting routines are those of ESME, for which the graphical output options set in the O Command were specifically intended, the user can construct plots using those options set during the running of the program as retrieved from COMMON. Those employing other graphics routines may wish to implement another sort of interface entirely (e.g. menu-driven), with an entirely different set of output options.

```
      PROGRAM ESMEPLOT
C  This program "post-processes" ESME output data.  It may read
C  and plot history data or process the output file generated by
C  ESME during a run.
C
C  Possible enhancements include:
C   1) Input to GET command specifying
C      record to be read -- at present GET command only invokes GET
C      to read next record.
C   2) Command invoking CONTOUR interactively. (C is available)
C   3) "Menu" input option; input via menus rather than NAMELISTs.
C      Should be more robust and easier for the first-time user.
C   1 is easy; 2 and 3 are a little tougher.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
INCLUDE 'ESME$INC:E65_GRAFIX.INC'
INCLUDE 'ESME$INC:E65_CURVES.INC'
INCLUDE 'ESME$INC:E65_VERSION.INC'
      DATA NODRAW,POSTP /.TRUE.,.FALSE./
      DATA IDEV /1/
```

[1] POSTP=T, O Command, Section 2.2.4.

```
      DIMENSION ISVARY(ISIZPH)
      CHARACTER*1 CMND
      CHARACTER*74 WORDS
      LOGICAL EXFG, DONE
      CHARACTER*128 GRAFIL,HFIL,MRFIL,OHFIL,OGRAFIL,OMRFIL
      DATA GRAFIL,HFIL,MRFIL /3*' '/
C
C  READ SINGLE LETTER COMMANDS.
C
      PRINT 19,PVER
19    FORMAT(//1X,'ESME RF PLOTTING PROGRAM : VERSION NUMBER IS ',F5.2)
      ISTAT=LIB$INIT_TIMER()
C      IF(MOD(ISTAT,2).NE.0) CALL LIB$STOP(%VAL(ISTAT))
      IF(.NOT.ISTAT) CALL LIB$STOP(%VAL(ISTAT))
C
C  Return point after execution of a command.
C
   10 CONTINUE
      PRINT *,'ENTER COMMAND: '
      READ(5,2000) CMND,WORDS
 2000 FORMAT(A1,4X,A74)
      ISTAT=LIB$STAT_TIMER(2,ITIME)
      IF(.NOT.ISTAT) CALL LIB$STOP(%VAL(ISTAT))
      CPTIME=1.E-2*ITIME
      PRINT 2010, CPTIME
 2010 FORMAT('0',53('+'),2X,F9.2,' SEC.',2X,54('+'))
       PRINT 2020, CMND,WORDS
 2020 FORMAT(' CMND IS ',A1,'  ',A74)
      GO TO (100,200,300,400,500,600,700,800,900)
     +       INDEX('ODGHZXUNQ',CMND)
      PRINT*, 'THE IMPLEMENTED COMMANDS ARE'
      PRINT*, '     O: SET OUTPUT OPTIONS'
      PRINT*, '     D: DISPLAY PHASE SPACE WITH CURRENT PLOT PARAMETERS'
      PRINT*, '     G: GET A RECORD FROM TAPE 7'
      PRINT*, '     H: RETRIEVE AND PLOT HISTORY DATA FROM TAPE 9'
      PRINT*, '     Z: REWIND TAPE 7'
      PRINT*, '     X: PLOT REMAINDER OF TAPE 7'
      PRINT*, '     U: CHOOSE GRAPHICS AND/OR HISTORY FILES'
      PRINT*, '     N: COMPOSE AND DISPLAY MOUNTAIN RANGE PLOT'
      PRINT*, '     Q: STOP'
C  Try again if the command wasn't recognized.
      GO TO 10
C
C  O: OPTIONS FOR OUTPUT GRAPHICS
C
  100 CALL GRAFSET
      GO TO 10
C
```

```
C  D:  DISPLAY GRAPHICALLY PHASE POINTS AND (OPTIONALLY) BUCKET
C
  200 CONTINUE
      IF(IOPT.GE.0)THEN
        IF(MTCHSI .OR. MTCH95)THEN
C  Draw matched contour.
          CALL REFCONT
        ELSE
C  Draw bucket.
          THETAS=0.0
          CALL BUCKIT(IOPT,THETAS,DELCON)
        ENDIF
      ENDIF
      DO 210 I=1,NC
        CURVE(I,1)=CURVE(I,1)+DTHCURV
        CURVE(I,2)=CURVE(I,2)+DECURV
  210 CONTINUE
C  Plot distribution.
      CALL PHPLT
      GO TO 10
C
C  G: GET COORDINATES AND MACHINE PARAMETERS
C
  300 PRINT *, 'READING COORDINATES AND MACHINE PARAMETERS'
C  Save logical telling us whether we have initialized
C  DI-3000 or not and reset it after restoring COMMONs.
C  Also save phase space plot template.
      SAVND=NODRAW
C      DO 310 I=1,ISIZPH
C        ISVARY(I)=IPHARY(I)
C  310 CONTINUE
      CALL GET(.TRUE.,ITURN,DONE)
      NODRAW=SAVND
C      DO 320 I=1,ISIZPH
C        IPHARY(I)=ISVARY(I)
C  320 CONTINUE
      POSTP=.FALSE.
      PRINT *, 'DATA FOR TURN ',ITURN,' READ'
      GO TO 10
C
C  H: HISTORY OF IMPORTANT PARAMETERS COLLECTED ON TAPE9 EACH STEP
C
  400 CALL HISTORY(.TRUE.)
      GO TO 10
C
C
C  Z: REWIND GRAPHICAL OUTPUT TAPE TO BEGINNING (PRESENTLY
C     ONLY WAY TO GET AT A RECORD BEFORE THE CURRENT ONE)
```

```
C
  500 REWIND(7)
C  Save logical telling us whether we have initialized
C  DI-3000 or not and reset it after restoring COMMONs.
      SAVND=NODRAW
C      DO 510 I=1,ISIZPH
C        ISVARY(I)=IPHARY(I)
C  510 CONTINUE
      CALL GET(.TRUE.,ITURN,DONE)
      NODRAW=SAVND
C      DO 520 I=1,ISIZPH
C        IPHARY(I)=ISVARY(I)
C  520 CONTINUE
      POSTP=.FALSE.
      PRINT *, 'DATA FOR TURN ',ITURN,' READ'
      GO TO 10
C
C  X: JUST PLOT EVERYTHING FROM HERE ON OUT
C
  600 CONTINUE
C  Save logical telling us whether we have initialized
C  DI-3000 or not and reset it after restoring COMMONs.
      SAVND=NODRAW
      CALL GET(.TRUE.,ITURN,DONE)
      NODRAW=SAVND
      IF(DONE)THEN
        PRINT *,'ALL PLOTS COMPLETED'
        GO TO 10
      ENDIF
      PRINT *, 'DATA FOR TURN ',ITURN,' READ'
      IF(IOPT.GE.0)THEN
        IF(MTCHSI .OR. MTCH95)THEN
C  Draw matched contour.
          CALL REFCONT
        ELSE
C  Draw bucket.
          THETAS=0.0
          CALL BUCKIT(IOPT,THETAS,DELCON)
        ENDIF
      ENDIF
      DO 610 I=1,NC
        CURVE(I,1)=CURVE(I,1)+DTHCURV
        CURVE(I,2)=CURVE(I,2)+DECURV
  610 CONTINUE
      CALL PHPLT
      GO TO 600
C
C  Use specified graphics and history files.
```

```
C
  700 CONTINUE
      OGRAFIL=GRAFIL
      OHFIL=HFIL
      OMRFIL=MRFIL
      NAMELIST /USEFIL/ GRAFIL, HFIL, MRFIL
      READ(5,USEFIL)
      IF(GRAFIL .NE. OGRAFIL)THEN
C  First close old file if this is a new one
        IF(OGRAFIL.NE.' ')CLOSE(UNIT=7)
C  See if the requested file exists.
        INQUIRE(FILE=GRAFIL,EXIST=EXFG)
C  Exit if it doesn't
        IF(.NOT.EXFG)THEN
          PRINT *,'FILE NAMED ',GRAFIL,' CANNOT BE FOUND '
          GOTO 710
        ENDIF
C  Print reassurance if it does
        PRINT *,'GRAPHICAL DATA FROM FILE',GRAFIL
C  Open the requested file.  Note that we haven't read
C  anything yet.
        OPEN(UNIT=7,FILE=GRAFIL,ACCESS='SEQUENTIAL',
     1       STATUS='OLD')
      ENDIF
  710 CONTINUE
      IF(HFIL .NE. OHFIL)THEN
C  First close any old history file.
        IF(OHFIL.NE.' ')CLOSE(UNIT=9)
C  See if the requested file exists.
        INQUIRE(FILE=HFIL,EXIST=EXFG)
C  Exit if it doesn't
        IF(.NOT.EXFG)THEN
          PRINT *,'FILE NAMED ',HFIL,' CANNOT BE FOUND '
          GO TO 720
        ENDIF
C  Print reassurance if it does
      PRINT *,'HISTORY DATA FROM FILE',HFIL
C  Open the requested file.
      OPEN(UNIT=9,FILE=HFIL,ACCESS='SEQUENTIAL',
     1     STATUS='OLD')
      ENDIF
  720 CONTINUE
      IF(MRFIL .NE. OMRFIL)THEN
C  First close old file if this is a new one
        IF(OMRFIL.NE.' ')CLOSE(UNIT=20)
C  See if the requested file exists.
        INQUIRE(FILE=MRFIL,EXIST=EXFG)
C  Exit if it doesn't
```

```
        IF(.NOT.EXFG)THEN
          PRINT *,'FILE NAMED ',MRFIL,' CANNOT BE FOUND '
          GOTO 730
        ENDIF
C  Print reassurance if it does
        PRINT *,'GRAPHICAL DATA FROM FILE',MRFIL
C  Open the requested file.  Note that we haven't read
C  anything yet.
        OPEN(UNIT=20,FILE=MRFIL,ACCESS='SEQUENTIAL',
     1       STATUS='OLD')
       ENDIF
  730 CONTINUE
       GO TO 10
C
C  M: DISPLAY MOUNTAIN RANGE(s)
C
  800 CONTINUE
       CALL MRPLT
       GO TO 10
C
C  Q: QUIT PROGRAM ENTIRELY; NOTHING FURTHER TO DO
C
  900 CONTINUE
       PRINT *, 'QUIT COMMAND'
C
C  Terminate DI-3000 and GRAFMAKER; if necessary.
C
       IF(.NOT.NODRAW)CALL JCHTRM(.TRUE.)
       STOP
C
       END
```

Appendix B

Running ESME on other computers

ESME has also been installed on an FPS-164 with ADCALC as front-end, and on an Amdahl 5890 with FNAL as front-end. The command procedures used to run ESME on these computers function essentially as RESMEVAX, described in Section 3.1. However, the graphics processing for the output from these computers is performed on ADCALC. To run ESME on the FPS

1. Log on to node ADCALC.
2. Invoke `USR$DISK4:[ESME_FILES]USESME.COM`
3. Type "RESMEFPS"

and then respond to the queries. RESMEFPS functions very similarly to RESMEVAX, and requires no extra work on the part of the user. As of this writing, node ALMOND is not connected to the Amdahl at Fermilab, so running ESME on the Amdahl requires a little more initiative on the part of the user. Job submission to the Amdahl requires that the user invoke a prepared command procedure while logged on to FNAL. The user can obtain this procedure by invoking USESME on node ALMOND, and then copying ESME$COM:RESMEAMD.COM to the user's account on FNAL. One these preliminaries have been accomplished, ESME may be run on the Amdahl, and its ouput processed on the VAX, through the following

1. Log on to node FNAL.
2. Invoke RESMEAMD.COM; respond to the queries to submit a job.
3. Log on to ADCALC.
4. Invoke `USR$DISK4:[ESME_FILES]USESME.COM`.
5. Type "POSTESME"

Postesme will create a subdirectory on ADCALC into which all of the output files from the Amdahl will be copied (via FNAL). It also causes a batch job to be submitted to copy these files and process the ouput using ESMEPLOT. (See Section A for a description.)

Bibliography

[1] *Design Report Tevatron I Project*, Fermi National Accelerator Laboratory (October 1984), Chapters 4–6

[2] J. E. Griffin, J. A. MacLachlan, A. G. Ruggiero, K. Takayama, "Time and Momentum Exchange for the Production and Collection of Intense Antiproton Beams at Fermilab", IEEE Trans. Nucl. Sci. **30**#4(1983) 2630–2632

[3] J. E. Griffin, J. A. MacLachlan, Z. B. Qian," RF Exercises Associated with Acceleration of Intense Antiproton Bunches at Fermilab", IEEE Trans. Nucl. Sci. **30**#4(1983) 2627–2629

[4] J. A. MacLachlan, "ESME: Longitudinal Phasespace Particle Tracking — Program Documentation", Fermilab TM-1274(May 1984), unpublished (largely obsolete)

[5] J. A. MacLachlan, "Longitudinal Phasespace Tracking with Spacecharge and Wall Coupling Impedance", Fermilab FN-446 (February 1987), unpublished.

[6] P. Lucas and J. MacLachlan, "Simulation of Spacecharge Effects and Transition Crossing in the Fermilab Booster", Proc. 1987 IEEE Particle Accelerator Conference held at Washington, D.C. (16–19 March 1987) p1114.

[7] S. Stahl and C. Ankenbrandt, "Simulation of the Capture Process in the Fermilab Booster", Proc. 1987 IEEE Particle Accelerator Conference held at Washington, D.C. (16–19 March 1987)p1117

[8] P. Lucas and Q. Kerns, "Simulation of a Programmed Frequency Shift Near Extraction from the Fermilab Booster", Proc. IEEE Particle Accelerator Conference held at Washington, D.C. (16–19 March 1987)p1108

[9] S. Stahl and S. A. Bogacz, "Coupled Bunch Instability in a Circular Accelerator and Possible Cures: Longitudinal Phasespace Simulation", Phys. Rev. **D37**#5(1 March 1988)1300–1306

[10] J. A. MacLachlan, "Fundamentals of Particle Tracking for the Longitudinal Projection of Beam Phasespace in Synchrotrons", Fermilab FN-481(15 April 88), unpublished.

[11] J. A. MacLachlan, "Difference Equations for Longitudinal Motion in a Synchrotron", Fermilab FN-529(15 December 1989), unpublished.

[12] S. Stahl and S. A. Bogacz, "Simulation of Coupled Bunch Mode Growth Driven by a High-Q Resonator: A Transient Response Approach", Proc. 1989 IEEE Particle Accelerator Conference held at Chicago, Il. (20–23 March 1989)p1175.

[13] M. W. Eaton, "Expand and Xfort, Macro Expanders for CDF", Fermilab CDF-194(January 1986), unpublished.