2 ✓
MASTER

UCID- 17050

Lawrence Livermore Laboratory

AN 880 NANOSECOND PARTICLE IN CELL MOVER FOR THE CDC 7600

Kent Estabrook and J. E. Tull

October 4, 1977

NOTICE

PORTIONS OF THIS REPORT ARE ILLEGIBLE. It has been reproduced from the best available copy to permit the broadest possible availability.

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

\$ COMPASS I=XME,L=LISTEL,B=BINELS,BOX=V24
IDENT MOVELES

*CC
*CC AN 880 NSEC 1-D ELECTROSTATIC PARTICLE MOVER FOR THE CDC 7600
*CC BY KENT ESTABROOK AND J. E. TULL
*CC LAWRENCE LIVERMORE LABORATORY JANUARY 20, 1976
*CC
*CC 1. ABSTRACT
*CC
*CC WE DESCRIBE A VERY FAST COMPUTATIONAL METHOD OF MOVING PARTICLES
*CC FOR ONE DIMENSIONAL ELECTROSTATIC PLASMA SIMULATIONS USING INTEGER
*CC ARITHMETIC. THE "CLOUD IN CELL" METHOD FORMS THE BASIS OF THIS TECHNIQUE.
*CC THIS PAPER IS IN THE FORM OF A COMPASS COMPILABLE SUBROUTINE WITH
*CC COMMENTS AND EXAMPLES DESCRIBING METHODS OF USING AN ENTIRELY INTEGER
*CC REPRESENTATION TO GAIN UP TO AN ORDER OF MAGNITUDE INCREASE
*CC IN SPEED OVER EQUIVALENT FLOATING POINT FORTRAN CODING. INTEGER
*CC ARITHMETIC HAS SEVERAL ADVANTAGES OVER FLOATING POINT ARITHMETIC
*CC FOR ONE DIMENSIONAL PARTICLE MOVERS. THE ADDS ARE MUCH FASTER, AND
*CC THE BINARY NUMERICAL DESCRIPTION ALLOWS THE IMPLEMENTATION OF VERY
*CC SIMPLE BOUNDARY CONDITIONS IF THE SIMULATION REGION EXTENDS FROM
*CC ZERO TO A POWER OF TWO. FURTHERMORE, INTEGER ARITHMETIC MAKES VERY
*CC EFFICIENT USE OF EACH MEMORY BIT SINCE THERE IS NO FLOATING
*CC POINT EXPONENT. CONSEQUENTLY, IT BECOMES FEASIBLE AND FAST IN LONG
*CC WORD LENGTH MACHINES TO PACK THE VELOCITY AND POSITION OF A
*CC PARTICULAR PARTICLE INTO THE SAME WORD, THUS SAVING A FACTOR OF TWO
*CC IN COMPUTER STORAGE AND/OR IO. THESE ADVANTAGES ARE COMBINED WITH THE
*CC FACT THAT INTEGER ADDS AND LOGICALS COMPLETE IN ONLY TWO COMPUTER
*CC CYCLES ALLOWING NEARLY COMPLETE OPTIMIZATION OF REGISTER AND
*CC INSTRUCTION INTERLEAVING. THE MEASURED TIMINGS ON A CDC 7600 ARE
*CC 880 NANOSECONDS AND 935 NANOSECONDS PER PARTICLE FOR THE PERIODIC
*CC AND GENERAL BOUNDARY CONDITION SECTIONS RESPECTIVELY. METHODS FOR ADAPTING
*CC THE TECHNIQUE TO OTHER COMPUTERS ARE DISCUSSED.
*CC
*CC THIS LISTING IS AVAILABLE FROM THE AUTHORS IN THE FORM OF A FILE
*CC ON THE MFE COMPUTER OR IN CARDS.
*CC
*CC 2. STANDARD FORTRAN PARTICLE MOVER ALGORITHM (CLOUD IN CELL.)
*CC
*CC 2.1 A SIMPLE FORTRAN PARTICLE PUSHING LOOP IS SHOWN BELOW. THE GRID
*CC SPACING HAS BEEN NORMALIZED TO UNITY; OTHERWISE THE LOOP IS
*CC UNOPTIMIZED FOR CLARITY:
*CC QOVRM=Q/EM
*CC DO LOOP K=1,NP
*CC IX=IFIX(X(K))
*CC DX=X(K)-FLOAT(IX)
*CC ONEMDX=1.-DX
*CC V(K)=V(K)+QOVRM*DT*(E(IX)*ONEMDX+E(IX+1)*DX)
*CC X(K)=X(K)+V(K)*DT
*CC C APPROPRIATE BOUNDARY CONDITIONS PERFORMED HERE
*CC IX=IFIX(X(K))
*CC DX=X(K)-FLOAT(IX)
*CC ONEMDX=1.-DX
*CC RHO(IX)=RHO(IX)+Q*ONEMDX
*CC RHO(IX+1)=RHO(IX+1)+Q*DX
*CC LOOP CONTINUE
*CC WHERE NP IS THE NUMBER OF PARTICLES OF A GIVEN SPECIES. X AND V
*CC ARE THE POSITION AND VELOCITY ARRAYS FOR THAT SPECIES AND
*CC Q AND EM ARE THE CHARGE AND MASS OF A MACRO PARTICLE OF THAT
*CC SPECIES. E AND RHO ARE THE ELECTRIC FIELD AND CHARGE DENSITY ARRAYS
*CC ON A GRID WITHIN THE SIMULATION REGION.

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

NOTICE

PORTIONS OF THIS REPORT ARE ILLEGIBLE. It has been reproduced from the best available copy to permit the broadest possible availability.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

```

*CC
*CC 2.2 IF THE VELOCITY IS NORMALIZED TO  $\sqrt{TE/EME} \cdot DT$  AND THE ELECTRIC
*CC FIELD TO  $(QE \cdot DLE/TE) \cdot RMASS \cdot DT \cdot \cdot 2$  WHERE TE, EME, QE AND
*CC DLE ARE THE ELECTRON TEMPERATURE (IN ENERGY UNITS), MASS, CHARGE AND
*CC DEBYE LENGTH RESPECTIVELY AND RMASS IS THE MASS RATIO OF A GIVEN
*CC SPECIES TO THE ELECTRON MASS AND FURTHERMORE IF THE FOLLOWING ARRAY:
*CC      DO LP10 I=0, NC-1    $$$ NC IS UPPER CELL BOUND
*CC      LP10 DE(I)=E(I+1)-E(I)
*CC IS FORMED PRIOR TO THE PARTICLE MOVING, THE LOOP BECOMES:
*CC      DO LOOP K=1, NP
*CC      IX=IFIX(X(K))
*CC      DX=X(K)-FLOAT(IX)
*CC      V(K)=V(K)+E(IX)+DX*DE(IX)
*CC      X(K)=X(K)+V(K)
*CC      C APPROPRIATE BOUNDARY CONDITIONS PERFORMED HERE
*CC      IX=IFIX(X(K))
*CC      SUMX(IX)=SUMX(IX)+X(K)
*CC      SUM1(IX)=SUM1(IX)+1.
*CC      LOOP CONTINUE
*CC RHO CAN BE OBTAINED FROM SUMX AND SUM1 AFTER ALL PARTICLES OF ONE
*CC SPECIES HAVE BEEN MOVED AS SHOWN BELOW:
*CC      DO LP10 I=0, NC-1
*CC      SUMDX=SUMX(I)-FLOAT(I)*SUM1(I)
*CC      RHO(I)=RHO(I)+(SUM1(I)-SUMDX)*Q
*CC      LP10 RHO(I+1)=RHO(I+1)+SUMDX*Q
*CC
*CC 2.3 THERE IS ONE FURTHER SIMPLIFICATION. IF THE FOLLOWING ARRAY:
*CC      DO LP20 I=0, NC
*CC      LP20 CE(I)=E(I)-FLOAT(I)*DE(I)
*CC IS FORMED BEFORE MOVING, THE THIRD AND FOURTH LINES MAY BE REPLACED BY:
*CC      V(K)=V(K)+CE(IX)+X(K)*DE(IX)
*CC THIS WAS NOT USED IN THE FOLLOWING INTEGER MOVER SINCE UNDER SOME
*CC CONDITIONS THERE MIGHT BE A PRECISION PROBLEM, AND ON THE CDC 7600
*CC ONLY A FEW PERCENT INCREASE IN SPEED WOULD BE GAINED. HOWEVER,
*CC AS EXPLAINED BELOW, THIS PRECISION LOSS IS GENERALLY NEGLIGIBLE.
*CC
*CC 3. ADVANTAGES OF INTEGER ARITHMETIC
*CC
*CC 3.1 THE LOOP HAS BEEN REDUCED TO ONE MULTIPLY AND A NUMBER OF
*CC ADDITIONS. INTEGER OR LONG ADDITIONS ARE CONSIDERABLY FASTER THAN
*CC NORMALIZED FLOATING POINT ADDITIONS, ALTHOUGH THE INTEGER MULTIPLY
*CC IS SOMEWHAT SLOWER THAN THE FLOATING MULTIPLY.
*CC
*CC 3.2 STANDARD SCALINGS AND NORMALIZATIONS AS DISCUSSED IN SECTION 2.2
*CC MEAN THAT THE FLOATING POINT EXPONENT IS NOT BEING WELL UTILIZED.
*CC INTEGER ARITHMETIC ON THE OTHER HAND, MAKES VERY EFFICIENT USE OF
*CC ALL MEMORY BITS UNDER THESE CONDITIONS. IT THEREFORE BECOMES FEASIBLE
*CC IN LONG WORD MACHINES TO  $\square$ PACK $\square$  A PARTICULAR PARTICLE'S POSITION AND
*CC VELOCITY INTO ONE MACHINE WORD WITH NO SIGNIFICANT LOSS OF
*CC ACCURACY. THIS PACKING CAN CAUSE A SIGNIFICANT REDUCTION IN REQUIRED
*CC COMPUTER MEMORY AND INPUT/OUTPUT CHARGES.
*CC
*CC 3.3 THERE ARE ADDITIONAL ADVANTAGES TO INTEGER ARITHMETIC. PERIODIC
*CC BOUNDARY CONDITIONS ARE REDUCED FROM TWO IF STATEMENTS PLUS APPROPRIATE
*CC COMPUTATIONS TO A SIMPLE MASK OF THE PARTICLE'S POSITION.
*CC OTHER BOUNDARY CONDITIONS CAN BE HANDLED IN A SIMILARLY
*CC CONVENIENT AND FAST METHOD AS DISCUSSED IN SECTION 4.3.
*CC THE ABOVE ASSUMES THAT THE SIMULATION REGION LIES BETWEEN
*CC ZERO AND AN INTEGER POWER OF TWO.
*CC

```

3.

```

*CC 3.4 ANOTHER ADVANTAGE OF INTEGER ARITHMETIC IS THAT
*CC THE TABLE LOOKUP INDEX FOR THE FIELDS AND CHARGE DENSITY ARE REDUCED
*CC FROM AN IFIX TO A SIMPLE SHIFT.
*CC
*CC 3.5 A SIMPLE EXAMPLE OF INTEGER ARITHMETIC IS SHOWN BELOW.
*CC DT DOES NOT NEED TO BE A POWER OF 2 BUT TO SIMPLIFY THE
*CC FOLLOWING EXAMPLES, DT=0.25. SUPPOSE THAT THE VELOCITY OF A PARTICLE
*CC IS 2 THERMAL VELOCITIES AND THAT THE FRACTIONAL PART OF THE
*CC VELOCITY IS CONTAINED IN 6 BITS OF THE WORD. V THEN HAS THE BINARY
*CC (NOT OCTAL) REPRESENTATION 10.000000. THE . IS BY USER DEFINITION AND
*CC IS ADJUSTABLE. (V DT) IS THEN 00.100000. SUPPOSE THAT THE ELECTRIC
*CC FIELD IS 2 TE/(QE*DLE) = 10.000000 AND (E DT**2) = 00.001000.
*CC SO (V(N+1/2) DT)=(V(N-1/2) DT) + (E(N) DT**2) IS (N REPRESENTS TIME)
*CC 00.100000 + 00.001000 = 00.101000. DIVIDING BY DT (SHIFT LEFT 2)
*CC GIVES 10.1 BINARY OR 2.5 THERMAL VELOCITIES WHICH IS CORRECT.
*CC THE TECHNIQUE TO OTHER MACHINES ARE ALSO DISCUSSED.
*CC
*CC 3.6 THERE ARE TWO PRECAUTIONS ONE SHOULD CONSIDER USING INTEGER MULTIPLIES
*CC 1. ALTHOUGH THE INTEGER ADD IS GOOD FOR 60 BITS, THE MULTIPLY IS ONLY
*CC GOOD FOR A FINAL PRODUCT OF 48 BITS ON CDC MACHINES. CARE MUST BE
*CC TAKEN TO INSURE THAT DX*DE NEVER EXCEEDS 48 BITS.
*CC 2. THE CDC MACHINES AT LLL CORRECTLY EXTEND THE SIGN BIT FOR
*CC FOR A NEGATIVE PRODUCT. IF THIS MOVER IS USED ON ANOTHER MACHINE
*CC MAKE SURE THAT THE MACHINE DOES NEGATIVE INTEGER PRODUCTS CORRECTLY
*CC
*CC 4. MACHINE CODED INTEGER MOVER
*CC
*CC 4.1 THE ACTUAL MACHINE LANGUAGE SOURCE IS CONTAINED IN
*CC SECTION 9. TWO ENTRY POINTS ARE PROVIDED.
*CC
*CC 4.2 THE FIRST ENTRY, MOVELESP, IS SET UP FOR PERIODIC BOUNDARY
*CC CONDITIONS (BX7 -X6*X7). NEGATIVE X IS CONSIDERED BELOW BOUNDS
*CC AND IS MASKED SUCH AS TO COME DOWN FROM THE TOP. SIMILARLY,
*CC PARTICLES GOING ABOVE THE TOP BOUNDARY (GIVEN BY A POWER OF TWO) ARE
*CC MASKED SO AS TO COME UP FROM THE BOTTOM. FOR EXAMPLE, SUPPOSE FOR
*CC SIMPLICITY THAT THE POSITION OF A PARTICLE IS DESCRIBED BY EIGHT BITS
*CC IN A TEN BIT WORD AND IS TYPICALLY 0010101011 WHERE THE RIGHTMOST
*CC SIX BITS REPRESENT THE FRACTIONAL PART OF THE POSITION AND THE NEXT
*CC TWO BITS REPRESENT THE CELL NUMBER. THE SIMULATION REGION THEN EXTENDS
*CC FROM ZERO TO FOUR. THE LEFTMOST TWO BITS ARE NOT PART OF THE POSITION.
*CC IF AFTER BEING MOVED A PARTICLE GOES OUT THE RIGHT BOUNDARY IT
*CC MIGHT HAVE A POSITION LIKE 0100100110 PUTTING IN BETWEEN THE FOURTH
*CC AND FIFTH CELLS. IF THIS PARTICLE'S NEW POSITION IS THEN MASKED
*CC SO AS TO ZERO OUT THE HIGH ORDER TWO BITS THE POSITION BECOMES
*CC 0000100110 (I.E., A LOGICAL PRODUCT OF 0100100110 AND 0011111111) PUTTING
*CC IT IN THE CORRECT LOCATION IN A PERIODIC SYSTEM. SIMILARLY, IF A PARTICLE
*CC GOES OUT THE LEFT BOUNDARY DURING A MOVE IT WILL HAVE A NEGATIVE POSITION
*CC SUCH AS 1111100101. AN IDENTICAL MASKING OF THE HIGH ORDER TWO BITS
*CC GIVES 0011100101 WHICH IS AGAIN THE CORRECT PERIODIC POSITION.
*CC
*CC 4.3 A SEPARATE ENTRY POINT, MOVELEEX, IS PROVIDED SO THAT THE USER CAN
*CC EMPLOY ANY DESIRED BOUNDARY CONDITION. IF A PARTICULAR PARTICLE GOES
*CC OUT OF THE SIMULATION REGION WHILE BEING MOVED, THE ROUTINE
*CC RETURNS TO ITS CALLING POINT AFTER STORING THE INDEX OF
*CC THE ERRANT PARTICLE IN THE VARIABLE KXVNOW. THE USER IS THEN FREE TO
*CC HANDLE THAT PARTICLE IN ANY DESIRED MANNER. ONE WORD OF CAUTION--
*CC THE PARTICLE'S VELOCITY/POSITION IS NOT UPDATED IN MEMORY PRIOR
*CC TO EXITING AND MUST BE HANDLED SEPARATELY OUTSIDE THE MOVER. THE
*CC VARIABLE KXVNOW SHOULD THEN BE INCREMENTED BY ONE AND THE MOVER
*CC REINITIALIZED. KXVNOW MUST BE SET TO THE INDEX OF THE FIRST PARTICLE

```



```

*CC TO BE MOVED WHEN INITIALLY CALLING THE MOVER EACH TIMESTEP. IF
*CC KXVNOW=KXVEND+1, WHERE KXVEND IS DESCRIBED IN SECTION 5.5, THE
*CC MOVER HAS ENDED NORMALLY FOR THAT CALL.
*CC
*CC 4.4 THE CODE ASSUMES THAT THE POSITION AND VELOCITY OF A PARTICLE
*CC ARE PACKED IN INTEGER FORM INTO ONE (60 BIT ON A CDC 7600) MACHINE WORD.
*CC TWO PARAMETERS, MODIFIABLE AT COMPILE TIME, DESCRIBE THE SUBDIVISION
*CC OF EACH COMPUTER WORD.  $\text{NBITSCEL}$  IS THE NUMBER OF BITS ALLOCATED
*CC FOR THE INTEGER CELL POSITION. THE SIMULATION REGION THEN EXTENDS
*CC FROM ZERO TO  $2^{**}\text{NBITSCEL}$ .  $\text{NFRACT}$  IS THE NUMBER OF BITS ALLOCATED
*CC TO THE FRACTIONAL PART OF THE POSITION AND ALSO TO THE FRACTIONAL
*CC PART OF THE VELOCITY. THE UPPERMOST (LEFT) BIT OF THE WORD IS
*CC THE SIGN BIT OF THE VELOCITY. THIS LEAVES  $59-\text{NBITSCEL}-2^{*}\text{NFRACT}$  BITS
*CC FOR THE INTEGER PORTION OF THE VELOCITY. TYPICALLY  $\text{NFRACT}$  IS 21 AND
*CC  $\text{NBITSCEL}$  IS 12. FOR THESE VALUES THE LOW ORDER (RIGHTMOST) 21 BITS
*CC FORM THE DX PART OF X, THE NEXT 12 BITS ARE THE CELL NUMBER, THE NEXT 21
*CC BITS ARE THE FRACTIONAL PART OF THE VELOCITY, THE NEXT 5 BITS ARE THE
*CC INTEGER PART OF THE VELOCITY AND THE UPPERMOST BIT IS THE VELOCITY
*CC SIGN. IF A SMALLER NUMBER OF CELLS ARE USED  $\text{NFRACT}$  MAY BE INCREASED
*CC ACCORDINGLY. CARE MUST BE TAKEN TO INSURE ENOUGH BITS FOR THE INTEGER
*CC PORTION OF THE VELOCITY.
*CC
*CC 4.5 THE ACTUAL LOOP IS NOW 7.5 WORDS IN LENGTH WHICH FITS INTO THE
*CC CDC 7600 STACK. THE MOVER UNPACKS, MOVES, COLLECTS CHARGE AND
*CC PACKS IN 880 NANOSECONDS PER PARTICLE ON THE CDC 7600 IN THE
*CC PERIODIC VERSION AND 935 NANOSECONDS IN THE GENERAL BOUNDARY CONDITION
*CC VERSION. BOTH VERSIONS HAVE BEEN RUNNING IN PRODUCTION CODES
*CC SINCE OCTOBER, 1975.
*CC
*CC 5. AUXILIARY CODE CHANGES
*CC
*CC 5.1 THE E AND DE ARRAYS MUST HAVE THE APPROPRIATE BIASING IN THE MOVER
*CC FOR THE INTEGER ADDITIONS TO BE PERFORMED PROPERLY. TO CONVERT
*CC FROM FLOATING POINT TO INTEGER PRIOR TO THE MOVER:
*CC      TWONFRACT=2.**NFRACT
*CC      DO LP30 I=0,NC
*CC          IE(I)=IFIX(E(I)*TWONFRACT)
*CC      LP30 IDE(I)=IFIX(DE(I)*TWONFRACT)
*CC TO REMOVE THIS BIASING FOR OTHER COMPUTATIONS:
*CC      OTWONFRACT=1./TWONFRACT
*CC      DO LP40 I=0,NC
*CC          LP40 E(I)=OTWONFRACT*FLOAT(IE(I))
*CC THE CONVENTION HERE IS THAT THE BIASED OR INTEGER EQUIVALENTS OF THE
*CC X, V, E, DE, AND SUMIX ARRAYS HAVE AN  $\text{I}$  PREFIXED TO
*CC THEIR NAMES.
*CC
*CC 5.2 TO CONVERT THE FLOATING POINT X AND V ARRAYS TO ONE PACKED
*CC INTEGER ARRAY, IXV:
*CC      NBITSX=NBITSCEL+NFRACT
*CC      NBITSV=60-NBITSX
*CC      DO LP50 K=1,NP
*CC          LP50 IXV(K)=((IFIX(V(K)*TWONFRACT).SHL.NBITSX).INT.Q8MOM(NBITSV))
*CC              .UN.(IFIX(X(K)*TWONFRACT))
*CC HERE .INT. IS A LOGICAL PRODUCT (SOMETIMES CALLED .AND.)
*CC AND .UN. IS A LOGICAL SUM (SOMETIMES CALLED .OR.).
*CC CHAT HAS A DIFFERENT MEANING FOR .INT., .AND., .UN. AND .OR.
*CC THAN CDC AND IBM COMPILERS.  $\text{Q8MOM}(J)$  IS AN INLINE INSTRUCTION THAT
*CC FORMS A BIT MASK WITH ONES IN THE UPPERMOST J BITS.
*CC
*CC 5.3 TO CONVERT THE SINGLE PACKED INTEGER ARRAY TO SEPARATE

```

```

*CC FLOATING POINT ARRAYS:
*CC      DO LP60 K=1,NP
*CC      X(K)=FLOAT(IXV(K).INT.-QBMMOM(NBITSV))*OTWONFRACT
*CC      LP60 V(K)=FLOAT(IXV(K).SHR.NBITSX)*OTWONFRACT
*CC THE .SHR. OPERATION CORRECTLY EXTENDS THE SIGN OF THE VELOCITY ON
*CC LLL'S 7600S. THIS MAY NOT BE GENERALLY TRUE ON OTHER MACHINES.
*CC
*CC 5.4 THE EQUIVALENT OF THE SUMX AND SUMI ARRAYS ARE RETURNED FROM THE
*CC MOVER AS ONE INTEGER PACKED ARRAY, ISUMIX. THE SUMI PORTION IS IN THE UPPER
*CC 18 BITS AND THE SUMX PORTION IN THE REMAINDER. FROM THIS THE
*CC CHARGE DENSITY MAY BE FORMED AS FOLLOWS:
*CC      I45MFRACT=45-NFRACT
*CC      DO LP70 I=0,NC-1
*CC      ICELLTIMESONES=(1*(ISUMIX(I).SHR.45)).SHL.NFRACT
*CC      ISUMDXCELL=(ISUMIX(I).INT.-QBMMOM(18))-ICELLTIMESONES
*CC      ISUMONES=(ISUMIX(I).INT.QBMMOM(18)).SHR.I45MNFRACT
*CC      IRHO(I)=IRHO(I)+ISUMONES-ISUMDXCELL
*CC      IRHO(I+1)=IRHO(I+1)+ISUMDXCELL
*CC      LP70 CONTINUE
*CC A UNIT CHARGE HAS BEEN ASSUMED IN THE ABOVE.
*CC IT IS CONSIDERABLY FASTER AND SLIGHTLY MORE ACCURATE TO CALCULATE THE
*CC ELECTRIC FIELD IE AND IDE IN INTEGER FORM DIRECTLY FROM IRHO
*CC BY GAUSS'S LAW WITHOUT GOING THROUGH FLOATING POINT.
*CC
*CC 5.5 THE CALLING SEQUENCES FOR THE TWO ENTRY POINTS ARE IDENTICAL:
*CC      CALL MOVELESP(IXV,IE,IDE,ISUMIX)
*CC      CALL MOVELEEX(IXV,IE,IDE,ISUMIX)
*CC IN ADDITION THERE IS ONE REQUIRED COMMON BLOCK:
*CC      COMMON/CMOVELES/KXVEND,KXVNOW,DT,NPOCELL,DENSCAL
*CC WHERE KXVNOW AND ISUMIX HAVE BEEN DESCRIBED ABOVE. KXVEND IS
*CC AN INPUT TO THE MOVER AND IS THE INDEX OF THE LAST PARTICLE TO BE
*CC MOVED DURING THE CURRENT CALL.
*CC
*CC 5.6 AN EXAMPLE OF HOW TO SET UP AN INITIAL LOADING IS SHOWN BELOW:
*CC      K=1
*CC      DO LP76 I=1,NC
*CC      IX=1.SHL.NFRACT
*CC      DO LP76 N=1,NE      $$$ NE IS NUMBER OF MACRO ELEC SHEETS PER CELL
*CC      Q=0.                $$$ SEE HANDBOOK MATH FUNCTIONS P953
*CC      DO LP73 L=1,12
*CC      LP73 Q=Q+RNFL(0)      $$$ RNFL(0) A RANDON NUMBER FROM 0 TO 1
*CC      VEL=Q-6.              $$$ MAX VEL=6.; EXP(-6*6/2)=1.5E-8
*CC      IV=(IFIX(VEL*TWONFRACT).SHL.NBITSX).INT.QBMMOM(NBITSV)
*CC      IXV(K)=IX.UN.IV
*CC      LP76 K=K+1
*CC TO LOAD A DENSITY GRADIENT, VARY THE NUMBER OF PARTICLES
*CC PER CELL AND THEIR POSITION IN THE CELL.
*CC
*CC 6. POSSIBLE MODIFICATIONS
*CC
*CC 6.1 IF ONE DID NOT WANT TO BE CONSTRAINED TO A POWER OF 2
*CC IN THE SYSTEM LENGTH, AND IF ONE WANTED STICKY BOUNDARY CONDITIONS, ONE
*CC COULD DELETE THE MASK LOGICAL PRODUCT (BX7 -X6*X1)
*CC AND EVERY TEN TIMESTEPS GO THROUGH A SEPARATE FILTER, SWAPPING
*CC PARTICLES BEYOND THE BOUNDS WITH THE END PARTICLES OF THE ARRAYS AND
*CC CEASE PROCESSING THEM BY RESETTNG KXVEND. ONE SHOULD THEN
*CC SET UP END ADDITIONS TO RHO SO THAT TOTAL CHARGE WILL BE CONSERVED.
*CC ALTERNATIVELY, ONE COULD USE THE ENTRY MOVELEEX.
*CC
*CC 6.2 CODE IS NOW SET UP SO THAT CELL SIZE IS UNITY. TO CHANGE THIS,

```


*CC CHANGE B6 WHICH SHIFTS X TO SET UP FOR INDEX FETCH OF IE, IDE AND ISUMIX.
 *CC
 *CC 6.3 THE LOOP FITS INTO A CDC 7600 STACK BUT IS HALF A WORD LONGER THAN
 *CC A CDC 6600 STACK. THERE ARE SEVERAL POSSIBLE MODIFICATIONS.
 *CC THREE 15 BIT INSTRUCTIONS MAY BE DELETED WITH LITTLE SACRIFICE IN
 *CC PRECISION. CHANGE E AS SHOWN BELOW:
 *CC DO LP80 I=1,NC
 *CC LP80 E(I)=E(I)-I.SHL.NFRACT
 *CC MAKE SURE THAN X*DE(ICELL) IS NOT GREATER THAN 48 BITS. A .SHR. OF DE
 *CC MAY BE NECESSARY WITH SOME SACRIFICE IN PRECISION AND OF COURSE THE
 *CC .SHR. NFRACT AFTER THE MULTIPLY MUST BE REPLACED BY AN APPROPRIATE ADJUST-
 *CC MENT FOR THE BINARY PLACE OF DE. REMEMBER THAT E AND DE HAVE DT**2 IN THEM.
 *CC THEN X6 SHOULD BE LOADED WITH MX6 NBITSV BEFORE THE LOOP AND
 *CC SHOULD NOT BE CHANGED DURING THE LOOP. THE MAKE MASK INSTRUCTIONS
 *CC AND MASK OF X TO DX CAN THEN BE DELETED, SAVING 45 BITS STACK SPACE.
 *CC IF ONE DOES NOT FOURIER SMOOTH, THE LOW ORDER BITS OF X*DE SURELY
 *CC DO NOT MATTER SINCE THEY ARE NOISE ON A FIRST ORDER CORRECTION. IF ONE
 *CC DOES FOURIER SMOOTH, THEN IT MIGHT MAKE SOME DIFFERENCE. THERE SHOULD
 *CC NOT BE ANY REGISTER CONFLICT PROBLEM, SO ONE MIGHT RUN IN LESS
 *CC CYCLES. ANOTHER WAY TO SAVE STACK SPACE IS TO THROW
 *CC OUT THE NOP AND REPLACE THE SX5 B1 AND LX5 45 BY SA5 A5
 *CC WHERE A5 IS THE ADDRESS OF A WORD WITH ONE BIT IN THE 45 PLACE.
 *CC
 *CC 6.4 WE HAVE NOT HAND-CODED THE NEW IBM MACHINES, BUT HAVE SOME SUGGESTIONS.
 *CC IF ONE WISHES TO CONVERT THIS CODE TO AN IBM MACHINE ONE WOULD NOT HAVE
 *CC TO PACK. DON'T TRY STORING FRACTIONAL PARTS INTO LESS THAN 20 BITS.
 *CC (SOME PEOPLE PACK TO 16 BITS WHICH SOMETIMES WORKS FOR ELECTRONS.)
 *CC YOU CAN USE A LOAD MULTIPLE AND GET X AND V TOGETHER IF THEY ARE INTERLACED.
 *CC YOU CAN TO GET USE LOAD MULTIPLES TO GET E AND DE AND TO GET THE TWO
 *CC PARTS OF THE CHARGE DENSITY IF THEY ARE EACH INTERLACED (I.E., THREE GROUPS
 *CC OF TWO SHORT WORDS APIECE). YOU CAN PROBABLY ARRANGE TO DO PERIODIC
 *CC BOUNDARY CONDITIONS WITHOUT A MASK BY USING IBM'S 32 BIT WORD BY
 *CC SIMPLY LETTING THE HIGH BITS FALL OFF THE END. IBM HAS A 64 BIT
 *CC MULTIPLY AND A DOUBLE REGISTER SHIFT WHICH EASILY ALLOWS THE SIMPLIFICATION
 *CC DISCUSSED IN SECTION 6.3. IBM HAS THE SO CALLED LOOK AHEAD HARDWARE
 *CC TO SCHEDULE OPERATIONS TO FUNCTIONAL UNITS WHICH LEADS ONE TO
 *CC BELIEVE THAT ONE CAN NOT REALLY GAIN BY SCHEDULING INSTRUCTIONS HIMSELF.
 *CC A LOT OF SPEED CAN BE REALIZED BY INTERLACING INSTRUCTIONS SO THAT THEY
 *CC ISSUE EVERY CYCLE AS CDC USERS HAVE KNOWN FOR SOME TIME SINCE THEY HAVE
 *CC EXPLICIT CONTROL. SCHEDULING IS NOT NEARLY AS CRITICAL WITH IBM AS
 *CC WITH CDC, BUT CAN MAKE A DIFFERENCE AND I WOULD SUGGEST THAT
 *CC IBM USERS LEAVE THIS CODING LOGIC IN THE SAME ORDER (WITHIN PRACTICALITY).
 *CC AN EXAMPLE OF POOR SCHEDULING AS A MULTIPLY FOLLOWED IMMEDIATELY
 *CC BY AN INSTRUCTION WHICH USES THAT RESULT OF THAT MULTIPLY.
 *CC IBM'S LOOK AHEAD SOMETIMES HELPS BUT OFTEN (ALWAYS WITH CDC)
 *CC THE MACHINE IS FORCED TO DO NOTHING UNTIL THE MULTIPLY IS FINISHED.
 *CC THE MACHINE COULD BEGIN FOUR MORE INDEPENDENT OPERATIONS SUCH AS
 *CC FETCHES, STORES, LOGICALS, ADDS, AND NORMALIZATIONS IN THOSE
 *CC FOUR LOST CYCLES IF THEY ARE EXPLICITLY ISSUED.
 *CC
 *CC 7. GENERAL COMMENTS ON NOISE AND ERRORS
 *CC
 *CC A EXPERIMENTALIST CAN TELL YOU THAT REAL PLASMAS
 *CC ARE NOT VERY REPRODUCIBLE AND THEIR LARGE UNCERTAINTY IS DUE
 *CC TO THEIR NOISE.
 *CC PLASMAS AND PARTICLE PUSHERS ARE VERY NOISY AND THE NOISE AMPLIFIES
 *CC THE MACHINE ROUND OFF ERROR. TO CONVINCE YOURSELF OF THIS
 *CC TAKE YOUR PRESENT PARTICLE PUSHER AND REVERSE THE ORDER OF ADDS
 *CC IN SUCH CASE AS:
 *CC IX=IFIX(X(K))

```

*CC      FI=FLOAT(IX)
*CC      V=V+E(IX)*(FI+1.-X(K))+E(IX+1)*(X(K)-FI)
*CC AND WATCH ALL THE ANSWERS CHANGE TO THE SECOND PLACE (EVEN WITH A LARGE
*CC NUMBER OF PARTICLES PER DEBYE LENGTH). IF YOU RUN ENOUGH PARTICLES
*CC PER DEBYE LENGTH, AND NO MORE THAN TWO OR AT MOST THREE DEBYE LENGTHS
*CC PER CELL, THEN THE ROUND OFF ERROR WILL BE MUCH LESS THAN THE ACTUAL
*CC ERROR. A MUCH MORE DRAMATIC CHANGE CAN BE FOUND BY LOADING THE SAME
*CC TYPE OF INITIAL DISTRIBUTION FUNCTION BUT CHANGING THE
*CC RANDOM NUMBER GENERATOR STARTING POINT. THIS WILL QUICKLY
*CC CONVINCE YOU HOW TRULY COMPLEX PLASMAS ARE AND HOW WELL THEY REMEMBER
*CC THEIR INITIAL CONDITION. THE TIME EVOLUTION OF COLLISIONLESS
*CC PLASMAS IS VERY SENSITIVE TO THEIR INITIAL CONDITION EVEN IN A STRONGLY
*CC UNSTABLE INITIAL CONDITION SUCH AS A TWO STREAM INSTABILITY WITH
*CC A STRONG GROWTH RATE. CHANGING THE RANDOM NUMBER GENERATOR WILL
*CC PROBABLY NOT AFFECT THE  $E^{*2}$  OF THE FIRST MAXIMUM, BUT WILL EFFECT
*CC ALL THE REST. AN OLD MAXIM IS TO DOUBLE THE NUMBER OF PARTICLES
*CC AND SEE HOW MUCH THE ANSWERS CHANGE. YOU MAY BE ASSURED, IF YOU
*CC HAVE REASONABLE STATISTICS FOR YOUR GROWTH RATES, THAT THE DIFFERENCES
*CC ARE DUE AT LEAST AS MUCH TO THE CHANGE IN INITIAL CONDITION AS TO THE
*CC NUMERICAL DIFFERENCE (LESS NOISE) OF THE INCREASED NUMBER OF PARTICLES.
*CC BROWN, GITOMER AND LEWIS (REFERENCE 16) HAVE SOME INTERESTING
*CC THINGS TO SAY ABOUT THIS.
*CC
*CC 8. REFERENCES
*CC
*CC REFERENCES ON INTEGER MOVERS:
*CC 1. OSCAR BUNEMAN, "COMPUTER SIMULATION OF THE SATELITE PHOTO
*CC SHEATH," ESRIN (EUROPEAN SPACE RESEARCH INSTITUTE) INTERNAL
*CC RESEARCH NOTE NUMBER 92, JUNE 1970 AT FRASCATI, ITALY
*CC 2. BRENDAN GODFREY, CLIFF RHODES, KEITH TAGGERT
*CC AFWL-TR-72-47 AP 72
*CC 3. J. BORIS AND K. V. ROBERTS, J. COMP. PHYS. 4, 4, (1969).
*CC
*CC PARTIAL LIST OF REFERENCES OF PARTICLE SIMULATION
*CC 4. OSCAR BUNEMAN, PHYS. REV. 115, 503, (59).
*CC 5. JOHN DAWSON PHYS. REV. 118, 381 (1960).
*CC 6. OSCAR BUNEMAN, J. COMP. PHYS. 1, 517 (1967).
*CC 7. J. M. DAWSON, AND R. SHANNY, PHYS. FLUIDS 11, 1506 (1968).
*CC 8. R. W. HOCKNEY, J. COMP. PHYS. 8, 19 (1971).
*CC 9. H. OKUDA AND C. K. BIRDSALL, PHYS. FLUIDS 13, 2123 (1970).
*CC 10. J. DENAVIT AND W. L. KRUER, PHYS. FLUIDS 14, 1782 (1971).
*CC 11. A. BRUCE LANGDON, J. COMP. PHYSICS 6, 247 (1970).
*CC 12. W. L. KRUER, J. M. DAWSON, AND B. ROSEN, J. COMP. PHYSICS
*CC 13, 114 (1973).
*CC 13. A. BRUCE LANGDON AND C. K. BIRDSALL, PHYS. FLUIDS 8, 2115 (1970).
*CC 14. METHODS IN COMPUTATIONAL PHYSICS, EDITED BY B. ALDER,
*CC S. FERNBACH, M. ROTHENBURG, (ACADEMIC, NEW YORK), VOLUME 9 (1970).
*CC 15. METHODS IN COMPUTATIONAL PHYSICS, EDITED BY B. ALDER, S.
*CC FERNBACH, VOLUME EDITOR: J. KILLEEN, (ACADEMIC, NEW YORK)
*CC VOL 16 (TO BE PUBLISHED).
*CC 16. BROWN, GITOMER AND LEWIS, J. COMP. PHYS. 14, 193 (1974).
*CC
*CC THERE ARE MANY OTHER VALUABLE PAPERS ON PARTICLE SIMULATION
*CC BY THESE AND OTHER AUTHORS.
*CC
*CC WE WISH TO ACKNOWLEDGE VALUABLE CONVERSATIONS WITH A. B. LANGDON,
*CC B. F. LASINSKI, AND W. L. KRUER.
*CC
*CC 9. SOURCE LISTING OF MOVER
*CC

```

```

*CC      X0=POSITION
*CC      X1=XV
*CC      X2=DE
*CC      X3=E
*CC      X4=SUMIX
*CC      X5=SUMIX INCREMENT
*CC      X6=MASKS
*CC      X7=STORAGE

*CC      B1=1
*CC      B2=XV ADDRESS+K
*CC      B3=XV ADDRESS END
*CC      B4=ADDRESS E
*CC      B5=ADDRESS DE
*CC      B6=NFRACT
*CC      B7=NBITSX

*CC      A0=ADDRESS RETURN
*CC      A5,A6 NOT USED
*CC
*CC TWIDDLE NEXT 4 NUMBERS FOR BIT ARRANGEMENT
NFRACT EQU 24
MASKDX EQU 60-NFRACT
NBITSCEL EQU 11
NBITSX EQU NFRACT+NBITSCEL
NBITSV EQU 60-NBITSX
DUMMY BSS 1 .DUMMY FOR A4 SET
USE /CMOVELZS/
KXVEND BSS 1
KXVNOW BSS 1
DT BSS 1
NPOCELL BSS 1
DENSCL BSS 1
USE

*CC
*CC ENTRY MOVCHK(NFRACT,NBITSCEL,IERRSW)
ENTRY MOVCHK
MOVCHK BSS 1
SA1 X1 .FETCH NFRACT IN MAIN
SA2 X2 .FETCH NBITSCEL IN MAIN
SX4 NFRACT
SX5 NBITSCEL
IX1 X1-X4
IX2 X2-X5
NZ X1,NOTOK
NZ X2,NOTOK
SX6 0
SB7 X7
SA6 X3 .STORE ERRSW=0
JP B7
NOTOK SX6 1
SB7 X7
SA6 X3 .STORE ERRSW=0
JP B7

*CC
*CC
*CC
*CC
*CC PERIODIC BOUNDARY CONDITION ENTRY POINT...
*CC

```

```

9.

*CC SUBROUTINE MOVELESP(IXV,IE,IDE,ISUMIX)
  ENTRY MOVELESP
MOVELESP  BSSZ  .1
          SA0   X7          .ADDRESS RETURN
          SA5   LOOP+4      .FETCH INSTRUCTION WORD TO LOAD ADR SUMIX
          BX0   X4          .SAVE ADDRESS SUMIX
          SA4   PART1+4     .FETCH FINAL INSTRUCTION WITH SUMIX
          SB2   X1          .ADR XV
          SA1   KXVEND      .DO LOOP LIMIT
          SB1   !
          SB6   NFRACT
          SB7   NBITSX
          SB3   X1+B2        .B3=ADR XV END
          SB3   B3-B1        .B3=ADR XV END-1
          SA1   B2          .FIRST XV FETCH
          MX6   42
          BX5   X6*X5        .ZERO ADDRESS BITS FOR SUMIX ADR
          BX7   X5*X0        .UNION ADR SUMIX INTO INSTRUCTION WORD
          LX6   30
          LX0   30          .SHIFT LEFT ADDRESS SUMIX FOR INSERTION
          BX4   X4*X6
          SA7   A5          .STORE INSTRUCTION WORD FOR SUMIX ADR INSERT
          BX7   X0+X4        .INSERT ADDRESS SUMIX
          SA7   A4          .STORE FINAL INSTRUCTION WITH ADDRESS SUMIX
          SA4   DUMMY        .FOR SA7 A4 IN LOOP
          SB4   X2          .ADDRESS E
          SB5   X3          .ADDRESS DE
          MX6   NBITSV      .MASK
          SX5   B1          .PREPARE FOR SUMIX
          LX5   45          .SHL. FOR SUMIX
          BX0   -X6*X1       .FIRST POSITION ISOLATED
          MX6   MASKDX
          AX3   B6,X0        .FIRST ICELL,B6 HAS NFRACT
          BX0   -X6*X0       .FIRST DX
          SA2   X3+B5        .FIRST DE(ICELL)
          SA3   X3+B4        .FIRST E(ICELL)
          AX4   B7,X1        .FIRST VELOCITY.SHR.NBITSX,X1 FREE
          EQ    B2,B3,PART1 .ONE PARTICLE?

*SS
LOOP      DX2   X2*X6        .K DE(ICELL)*DX
          SX5   B1          .PREPARE FOR SUMIX
          SA1   B2+B1        .K+1 XV FETCH
          SA7   A4          .K-1 SUMIX STORE
          LX5   45          .K .SHL. FOR SUMIX
          AX2   NFRACT      .K .SHR. MULTIPLY ADJUST
          NO
          IX2   X2*X3        .K E+(DE*DX)
          SB2   A1
          IX2   X4+X2        .K NEW VELOCITY
          MX6   NBITSV
          IX7   X0+X2        .K NEW POSITION
          BX0   -X6*X1       .K+1 POSITION ISOLATED
          BX7   -X6*X7       .K PERIODIC BC
          LX2   NBITSX      .K VELOCITY PACK
          IX5   X5+X7        .K SUMIX INCREMENT
          AX3   B6,X7        .K ICELL,B6 HAS NFRACT
          BX2   X6*X2        .K ZERO LOW NBITSX BITS
          SA4   X3+0         .K SUMIX FETCH, ADR LOADED IN INITIALIZATION
          BX7   X2+X7        .K XV PACK,X2 HAS V.SHL.NBITSX
          MX6   MASKDX

```

```

AX3    B6,X0          .K+1 ICELL,B6 HAS NFRACT
BX6    -X6*X0         .K+1 DX
SA2    X3+B5          .K+1 DE(ICELL) FETCH
SA3    X3+B4          .K+1 E(ICELL) FETCH
SA7    A1-B1          .K XV STORE
IX7    X4+X5          .K SUMIX UPDATE,X4,5 FREE
AX4    B7,X1          .K+1 VELOCITY.SHR.NBITSX, X1 FREE
NE     B2,B3,LOOP     .B2=PRES XV ADR,B3=MAX XV ADR

*DD
PART1  DX2    X2*X6          .LAST DE(ICELL)*DX
        SX5    B1            .PREPARE FOR SUMIX
        SA7    A4            .LAST-1 SUMIX STORE
        LX5    45            .LAST .SHL. FOR SUMIX
        AX2    NFRACT        .LAST .SHR. MULTIPLY ADJUST
        IX2    X2*X3          .LAST E+(DE*DX)
        SB2    A1
        IX2    X4+X2          .LAST NEW VELOCITY
        MX6    NBITSV
        IX7    X0+X2          .LAST NEW POSITION
        BX7    -X6*X7        .LAST PERIODIC BC
        LX2    NBITSX        .LAST VELOCITY PACK
        IX5    X5+X7          .LAST SUMIX INCREMENT
        AX3    B6,X7          .LAST ICELL,B6 HAS NFRACT
        BX2    X6*X2          .LAST ZERO LOW NBITSX BITS
        SA4    X3+0          .LAST SUMIX FETCH
        BX7    X2+X7          .LAST XV PACK,X2 HAS V.SHL.NBITSX
        MX6    MASKDX
        SA7    A1            .LAST XV STORE
        IX7    X4+X5          .LAST SUMIX UPDATE,X4,5 FREE
        SA7    A4            .LAST SUMIX STORE
        SB6    A0            .RETURN ADDRESS
        JP     B6

*CC
*CC GENERAL BOUNDARY CONDITION ENTRY POINT...
*CC THE XV ADDRESS SENT TO THE MOVER MUST BE XV(1) WITH THE STARTING
*CC AND ENDING MEMBERS OF THE ARRAY DEFINED BY KXVNOW AND KXVEND.
*CC
*CC      SUBROUTINE MOVELEEX(IVX,IE,IDE,ISUMIX)
          ENTRY MOVELEEX
XVADRST BSSZ    1          .SUBROUTINE ARG XV ADR
MOVELEEX BSSZ    1
        SA0    X7            .ADDRESS RETURN
        SA5    LOOPEX+5      .FETCH INSTRUCTION WORD FOR SUMIX
        BX0    X4            .SAVE ADR SUMIX
        SA4    PARTIEY+4     .FETCH FINAL INSTRUCTION SUMIX
        LX0    15            .SHIFT LEFT ADR SUMIX 15 BITS
        SB2    X1            .ADR XV
        BX7    X1            .SAVE START ADDRESS XV
        SA7    XVADRST
        SB4    X2            .ADDRESS E
        SB5    X3            .ADDRESS DE
        MX6    42
        LX6    15
        BX4    X4*X6
        BX5    X5*X6
        BX6    X0+X4          .UNION ADR SUMIX WITH INSTRUCTION
        BX7    X0+X5          .UNION ADR SUMIX WITH FINAL INSTRUCTION
        SA6    A4            .STORE INSTRUCTION WITH ADDRESS SUMIX
        SA7    A5            .STORE INSTRUCTION WITH FINAL ADDRESS SUMIX
        SA1    KXVEND        .DO LOOP LIMIT

```

	SA4	DUMMY	.FOR SA7 A4 IN LOOP
	SA2	KXVNOW	.NOW INDEX FOR XV. B2 HAS BEGIN ADR
	SB1	1	
	SB6	NFRACT	
	SB7	NBITSX	
	SB3	X1+B2	.B3=ADR XV END
	SB3	B3-B1	.B3=ADR XV END-1
	SX2	X2-1	.REMEMBER XV ARRAY DIMENSIONED FROM 1
	SB2	X2+B2	.BEGIN ADDRESS XV
	SA1	B2	.FIRST XV FETCH
	MX6	NBITSV	.MASK
	SX5	B1	.PREPARE FOR SUMIX
	LX5	45	.FIRST .SHL. FOR SUMIX
	BX0	-X6*X1	.FIRST POSITION ISOLATED
	MX6	MASKDX	
	AX3	B6,X0	.FIRST ICELL.B6 HAS NFRACT
	BX6	-X6*X0	.FIRST DX
	SA2	X3+B5	.FIRST DE(ICELL)
	SA3	X3+B4	.FIRST E(ICELL)
	AX4	B7,X1	.FIRST VELOCITY.SHR.NBITSX, X1 FREE
	EQ	B2,B3,PARTIEX	.ONE PARTICLE?
*SS			
LOOPEX	DX2	X2*X6	.K DE(ICELL)*DX
	SX5	B1	.PREPARE FOR SUMIX
	SA1	B2+B1	.K+1 XV FETCH
	SA7	A4	.K-1 SUMIX STORE
	LX5	45	.K .SHL. FOR SUMIX
	AX2	NFRACT	.K .SHR. MULTIPLY ADJUST
	NO		
	IX2	X2*X3	.K E+(DE*DX).X3 FREE
	SB2	A1+0	
	IX2	X4*X2	.K NEW VELOCITY
	MX6	NBITSV	
	IX7	X0+X2	.K NEW POSITION
	BX0	-X6*X1	.K+1 POSITION ISOLATED
	BX3	X6*X7	.MASK OUT X TO LOOK FOR OVERFLOW
	LX2	NBITSX	.K VELOCITY PACK
	IX5	X5+X7	.K SUMIX INCREMENT
	NZ	X3,BOUNDS	.EXIT IF X.LT.0 OR X.GT.2**NBITSCEL
	AX3	B6,X7	.K ICELL.B6 HAS NFRACT
	BX2	X6*X2	.K ZERO LOW NBITSX BITS
	SA4	X3+0	.K SUMIX FETCH
	BX7	X2+X7	.K XV PACK,X2 HAS V.SHL.NBITSX
	MX6	MASKDX	
	AX3	B6,X0	.K+1 ICELL.B6 HAS NFRACT
	BX6	-X6*X0	.K+1 DX
	SA2	X3+B5	.K+1 DE(ICELL) FETCH
	SA3	X3+B4	.K+1 E(ICELL) FETCH
	SA7	A1-B1	.K XV STORE
	IX7	X4+X5	.K SUMIX UPDATE,X4,5 FREE
	AX4	B7,X1	.K+1 VELOCITY.SHR.NBITSX, X1 FREE
	NE	B2,B3,LOOPEX	.R2=PRE5 XV ADR,B3=MAX XV ADR
*DD			
PARTIEX	DX2	X2*X6	.LAST DE(ICELL)*DX
	SX5	B1	.PREPARE FOR SUMIX
	SA7	A4	.LAST-1 SUMIX STORE
	LX5	45	.LAST .SHL. FOR SUMIX
	AX2	NFRACT	.LAST .SHR. MULTIPLY ADJUST
	IX2	X2*X3	.LAST E+(DE*DX)
	SB2	A1+B1	.A1 HAS ADDRESS K+1 PARTICLE

```

IX2      X4+X2      .LAST NEW VELOCITY
MX6      NBITSV
IX7      X0+X2      .LAST NEW POSITION
BX3      X6*X7      .MASK OUT X TO LOOK FOR OVERFLOW
LX2      NBITSX      .LAST VELOCITY PACK
NZ       X3,BOUNDS  .EXIT IF X.LT.0 OR X.GT.2*NBITSCEL
IX5      X5+X7      .LAST SUMIX INCREMENT
AX3      B6,X7      .LAST ICELL,B6 HAS NFRACT
EX2      X6*X2      .LAST ZERO LOW NBITSX BITS
SA4      X3+0       .LAST SUMIX FETCH
EX7      X2+X7      .LAST XV PACK,X2 HAS V.SHL.NBITSX
MX6      MASKOX
SA7      A1         .LAST XV STORE
IX7      X4+X5      .LAST SUMIX UPDATE,X4,5 FREE
SA7      A4         .LAST SUMIX STORE
SX7      B2+B1      .KXVEND+1

```

*CC X7 HAS ADDRESS XV + KXVEND + 2 SO THAT SUBTRACT MAKES

*CC KXVEND+1 SINCE DIMENSIONED FROM ONE

```

EXIT     SA1      XVAORST      .START ADDRESS
          IX7      X7-X1      .INDEX EXIT PARTICLE
          SA7      KXVNOW
          SB6      A0
          JP       B6

```

```

BOUNDS   SX7      B2         .B2 HAS ADDRESS PARTICLE K+1
          JP       EXIT      .PARTICLE K HAS BOUNDS EXIT
*CC BOUNDS PARTICLE HAS NOT BEEN MOVED FROM PREVIOUS POSITION

```

\$CHAT (KENTLIB,ORDERLIB,STACKLIB)%ME,EX,BIN,EF,V24,S,S..L
PROGRAM TT(OUTPUT,TAPE3=OUTPUT)

\$\$\$ FORTRAN ELECTROSTATIC MOVER

```

PARAMETER(NCELLS=4)
PARAMETER(NPCELL=2)
PARAMETER(NPTCLES=NPCELL*NCELLS)
PARAMETER(P1=3.141592654,TWOP1=6.283185308)
PARAMETER(OT=.25)

```

```

COMMON/CX/X(NPTCLES)
COMMON/CV/V(NPTCLES)
COMMON/CE/E((0,NCELLS))
COMMON/CRHO/RHO((0,NCELLS))

```

```

FMTE     FORMAT(10E12.4)
FMTE     FORMAT(10F12.6)
FMTO     FORMAT(/5022)
FMTI     FORMAT(IX,10I10)

```

```

          CHANGE(0+EXP)
          ZDS(10,6ROUTPUT,ISS,ERRZDS00)
          ZDS(10,4RWMX,ISS,ERRZDS00)
          ZDS(10,4RWFMY,ISS,ERRZDS00)
          ZDS(10,4RWFMO,ISS,ERRZDS00)
ERRZDS00 CONTINUE
          CRTID(0RWFMO,1)
          CALL KEEP00(4RWFMO)
          ONCELLS=1./FLOAT(NCELLS)
          ONPTCLES=1./FLOAT(NPTCLES)
          ADDVEL=1.414213562
          ADDVEL=0. $$$ FIRST LOOK FOR FLASMA OSCILLATIONS
          SYLEN=4.

```



```

      K=1
      DO LD000 I=0,NCELLS-1
      RHO(I)=FLOAT(NPCELL)
      DO LD001 N=1,NPCELL
      Q=0.
      DO LD002 L=1,12
LD002  Q=Q*RNFL(I)
      ADDVEL=-ADDVEL
      V(K)=Q-6.*ADDVEL
      X(K)=FLOAT(I)
      K=K+1
LD001  CONTINUE
LD000  CONTINUE

      DO TIMELOOP00 ITIMESTEP=0,10

$$$ GAUSS'S LAW
      P5ONPCCELL=.5/FLOAT(NPCELL)
      RHO(0)=RHO(0)+RHO(NCELLS)
      RHO(NCELLS)=0.
      E(0)=0.
      DO GAUSS00 IX=0,NCELLS-2
      E(IX+1)=E(IX)+1.*P5ONPCCELL*(RHO(IX+1)+RHO(IX))
GAUSS00 CONTINUE
      EZ=0
      DO AVE00 IX=0,NCELLS-1
      EZ=EZ+E(IX)
AVE00  CONTINUE
      EZ=EZ*ONCELLS
      DO AVE01 IX=0,NCELLS-1
      E(IX)=-.(E(IX)-EZ)
AVE01  CONTINUE
$$$ CHECK TO SEE IF AVERAGE IE IS ZERO
      C=CR=0
      DO ZCHECK00 IX=0,NCELLS-1
      C=C+E(IX)
      CR=CR+RHO(IX)
ZCHECK00 CONTINUE
      IF (ABS(CR-FLOAT(NPTCLES)).GT.1.E-7) GO TO CRASH00
      IF (ABS(C).GT.1.E-7) GO TO CRASH00

      TESQR=P=0
      IKE=0.
      DO TKEP00 K=1,NPTCLES
      P=P+V(K)
      TKE=TKE+V(K)**2
TKEP00 CONTINUE
      TKE=TKE*.5*ONPTCLES
      TP=P*ONPTCLES
      DO TESQR00 IX=0,NCELLS-1
      TESQR=TESQR+E(IX)**2
TESQR00 CONTINUE
      TESQR=TESQR*ONCELLS
      WRITE(3,FMT) TIME, TESQR, TKE, TP, F(0)
      IF (ITIMESTEP.INT.-QBMM(55)), PLOTS,
BACKPLOTS00 CONTINUE
      WRITE(3,FMT) X
      WRITE(3,FMT) V
      WRITE(3,FMT) E

```

```

WRITE(3,FMTF)RHO
TIME=TIME+DT
OVSET(0,RHO(0),NCELLS+1)

E(NCELLS)=E(0)
DO PTICLELOOP00 K=1,NPTCLES
  IXK=FIX(X(K))
  DXK=X(K)-FLOAT(IXK)
  V(K)=V(K)+DT*(E(IXK)*(1.-DXK)+E(IXK+1)*DXK)
  X(K)=X(K)+V(K)*DT
  IF(X(K).LT.0.)X(K)=X(K)+SYSLEN
  IF(X(K).GT.SYSLEN)X(K)=X(K)-SYSLEN
  IXK=FIX(X(K))
  DXK=X(K)-FLOAT(IXK)
  RHO(IXK)=RHO(IXK)+(1.-DXK)
  RHO(IXK+1)=RHO(IXK+1)+DXK
PTICLELOOP00 CONTINUE

TIMELOOP00 CONTINUE
  CALL EXIT(1)
PLOTS CONTINUE
  LINEARPLOT(0.,1.,RHO(0),NCELLS)
  WRITE(100,FMTRHO)TIME
FMTRHO FORMAT(=ELECTRON CHARGE DENSITY AT TIME=,F10.3)
  LINEARPLOT(0.,1.,E(0),NCELLS)
  WRITE(100,FMTEPLOT0)TIME
FMTEPLOT0 FORMAT(=ELECTRIC FIELD AT TIME=,F10.3)
  CALL FPAMEID
  MAPS(0.,SYSLEN,-3.,3.,1.,999.,15.,999)
  DO VXPLOTOO K=1,NPTCIF5
    POINT(X(K),V(K))
VXPLOTOO CONTINUE
  WRITE(100,FMTXVO)TIME
FMXVO FORMAT(=ELECTRON PHASE SPACE AT TIME=,F10.3)
  GO TO BACKPLOTS00

CRASH00 CONTINUE
  WRITE(3,FMTF)C,CR,ONCELLS,ONPTCLES
  WRITE(3,FMTF)F
  WRITE(3,FMTF)RHO
  CALL EXIT(1)
END

```

```

$$$ ELECTRON ONLY TEST FOR LANDAU DAMPING
%CHAT (BINEL5,KENTLIB,ORDERLIB,STACKLIB)%ME,EX,BIN,EF,V24,S,S,,L
  PROGRAM TT(OUTPUT,TAPE3=OUTPUT)
  PARAMETER(NCELLS=32)
  PARAMETER(NPCELL=1024)
  PARAMETER(NPTCLES=NPCELL*NCELLS)
  PARAMETER(NFRACT=25)
  PARAMETER(PI=3.141592654,TWOPI=6.283185308)
  PARAMETER(NCELLSM1=NCELLS-1)
  PARAMETER(MAXTIMESTEPS=100)
  PARAMETER(LENSCMX/=512)

  LCM (LCMIVX)
  COMMON/LCMIVY,IVXLCM(NPTCIF5)
  COMMON/CIVX/IVX(LENSCMXV+2)

```

```

COMMON/CIE/IE((0,NCELLS+1))
COMMON/CIDE/IDE((0,NCELLS+1))
COMMON/CISUMIX/ISUMIX((0,NCELLS+1))
COMMON/CIRHO/IRHO((0,NCELLS+1))
COMMON/CRHO/RHO((0,NCELLS+1))
COMMON/CFV/FV((0,129))
COMMON/CMOVELES/KXVEND,KXVNOW,TWONFRACT,OTWONFRACT
$,LOG2NPCELL,LOG2DTSQR,NBITSCELL,NSHRRHO,NBITSX,NBITSV
$,EPSILON,DT
$,ADVEL,SYSLEN,VEOVD,EPSILONV,XSCALE,OXSCALE,ESCALE,OESCALE
$,DX,VELSCALE,OVELSCALE,ONPTCLES,ONCELLS
$,I45MNFRACT,NBUFS0,NBUFS,NBUFEXTRA,LENBLOCK,ILCMBUF
$,I,ITIMESTEP,IPLLOTINTVAL,IXVPLLOTINTVAL
$,REALKLOAD,DXL,IX,IV,X,V,ENDCOMR
COMMON/COMOCT/ICELECTRIC,ICRHO,ICRHOC
$,IBITSHL45,IRHOI,ICELL
$,TIMESONESISUMDXCELL,ISUMONES
$,IEZ,MASKCHK,ENDCOMU
COMMON/COMPRINT/IWRITE,FMTPRINT,FMTprt((0,11)),LIMPRT
LCM (COMHIST)
COMMON/COMHIST/PHASEHIST((0,MAXTIMESTEPS))
$,ESQRHIST((0,MAXTIMESTEPS))
$,AMPHIST((0,MAXTIMESTEPS))

DATA DT/.125/,EPSILON/.04/,SYSLEN/12./
DATA ADVEL/0./
DATA IPLLOTINTVAL/32/,IXVPLLOTINTVAL/0/

FMTE FORMAT(10E12.4)
FMTF FORMAT(10F12.6)
FMTO FORMAT(/5022)
FMTI FORMAT(1X,10I10)

CHANGE(4+EX0)
ZDS(10,6ROUTPUT,ISS,ERRZDS00)
ZDS(10,5RDWFMX,ISS,ERRZDS00)
ZDS(10,5RDWFMY,ISS,ERRZDS00)
ZDS(10,5RDWFM0,ISS,ERRZDS00)
ERRZDS00 CONTINUE
CR1ID(0WFM0,1)
CALL KEEP80(5RDWFM0)
LOG2NPCELL=FIX(ALOG(FLOAT(NPCELL))*1.442695+.1)
LOG2DTSQR=FIX(ALOG(DT*DT)*1.442695+.1)
NBITSCELL=FIX(ALOG(FLOAT(NCELLS))*1.442695+.1)
NSHRRHO=1+LOG2NPCELL-LOG2DTSQR
NBITSX=NFRACT+NBITSCELL
NBITSV=60-NBITSX
TWONFRACT=2.*NFRACT
OTWONFRACT=1./TWONFRACT
XSCALE=SYSLEN*OTWONFRACT/FLOAT(NCELLS) $$$ IN DEBYELENGTHS
OXSCALE=1./XSCALE
DX=SYSLEN/FLOAT(NCELLS)
$$$ CELL SIZE ACTUALLY USED IS 1. SO MUST SCALE V.E APPROPRIATELY
VELSCALE=XSCALE/DT
OVELSCALE=1./VELSCALE
ESCALE=XSCALE/(DT**2)
OESCALE=1./ESCALE
ONPTCLES=1./FLOAT(NPTCLES)
ONCELLS=1./NCELLS
NBUFS0=NPTCLES/LENSCMXV

```

```

NBUFEXTRA=NPTCLES-NBUFS0*LENSCMXV
NBUFS=NBUFS0
IF(NBUFEXTRA.NE.0)NBUFS=NBUFS0+1
DATA LOCEPSILON/.LOC.EPSILON/
WRITE(59,(06)0)LOCEPSILON
  MOVCHK(NFRACT,NBITSCELL,IERSW)
  IF(IERSW)CALL EXIT(10)

IBITSHL45=1.SHL.45
DXL=SYLEN/FLOAT(NPTCLES)
X=-DXL
VPOVE=SQRT((SYLEN/TWOP)**2+3.)
EPSILONV=EPSILON*VPOVE
WRITE(59,FMT)VPOVE
REALKLOAD=TWOP/FLOAT(NPTCLES)
DO LD000 K=1,NPTCLES
  ARG=REALKLOAD*FLOAT(K)
  PERT=EPSILON*COS(ARG)
  PERTV=PERT*VPOVE
  X=X+DXL*(1./PERT)
  IX=(IFIX(X*OXSCALE)).INT.-QBMOM(NBITSV)
  ICELL=IX.SHR.NFRACT
  ISUMIX(ICELL)=ISUMIX(ICELL)
  $+(IBITSHL45.UN.IX)
  Q=0.
DO LD002 L=1,12
LD002  Q=Q+RNFL(0)
  ADDVEL=-ADDVEL
  VEL=Q-6.+ADDVEL+PERTV
  IV=(IFIX(VEL*OVELSCALE)).SHL.NBITSX).INT.QBMOM(NBITSV)
  IVXLCM(K)=IX.UN.IV
LD000  CONTINUE

  SETCH(0.,42.)
  DO WRITE00 !WRITE=3,100,97
  WRITE(IWRITE,FMT)NFRACT,NPCELL,NCELLS
WRITE00  PRINTCOM(KXVEND,ENDCOMR)

DO TIMELOOP00 ITIMESTEP=0,MAXTIMESTEPS

$$$ GAUSS'S LAW
  I45MNFRACT=45-NFRACT
  IRHO(0)=0
  MASKCHK=1.SHL.44
  DO LP70 I=0,NCELLSM1
    IF(ISUMIX(I))CRASH00.,$$$ CHECK FOR OVERFLOW
    IF(ISUMIX.INT.MASKCHK)CRASH00.,CRASH00
    ICELLTIMESONES=(1*(ISUMIX(I).SHR.45)).SHL.NFRACT
    ISUMDXCELL=(ISUMIX(I).INT.-QBMOM(18))-ICELLTIMESONES
    ISUMONES=(ISUMIX(I).INT.QBMOM(18)).SHR.I45MNFRACT
    IRHO(I)=IRHO(I)+ISUMONES-ISUMDXCELL
    IRHO(I+1)=ISUMDXCELL
LP70  CONTINUE
    IRHO(0)=IRHO(0)+IRHO(NCELLS)
    IRHO(NCELLS)=0
    IRHO1=NPCELL.SHL.(NFRACT+1)
    IE(0)=IRHO1-IRHO(0)-IRHO(NCELLSM1)
    DO GAUSS00 IX=0,NCELLS-2
    IE(IX+1)=IE(IX)+IRHO1-IRHO(IX)-IRHO(IX+1)
GAUSS00  CONTINUE

```

```

      IEZ=0
      DO AVE00 IX=0,NCELLSM1
      IEZ=IEZ+IE(IX)
AVE00  CONTINUE
      IEZ=IEZ/NCELLS
      DO AVE01 IX=0,NCELLSM1
      IE(IX)=-(IE(IX)-IEZ)
AVE01  CONTINUE
      IE(NCELLS)=IE(0) $$$ BE SURE TO MAKE DE(NCELLS-1) OK
      DO AVE02 IX=0,NCELLSM1
      IDE(IX)=(IE(IX+1)-IF(IX)).SHR.NSHRRHO
AVE02  CONTINUE
      DO AVE03 IX=0,NCELLSM1
AVE03  IE(IX)=IE(IX).SHR.NSHRRHO
$$$ CHECK TO SEE IF AVERAGE IE IS ZERO
      ICELECTRIC=ICRHO=0
      DO ZCHECK00 IX=0,NCELLSM1
      ICELECTRIC=ICELECTRIC+IE(IX)
      ICRHO=ICRHO+IRHO(IX)
ZCHECK00 CONTINUE
      ICRHOC=NPTCLES.SHL.NFRACT
      IF(IABS(ICRHO-ICRHOC).GT.77B)GO TO CRASH00
      IF(IABS(ICELECTRIC/NCELLS).GT.77B)GO TO CRASH00

      TESQR=0
      DO TESQR00 IX=0,NCELLSM1
      TESQR=TESQR+FLOAT(IE(IX))*2
TESQR00 CONTINUE
      TESQR=TESQR*ESCALE*ESCALE*.5*ONCELLS
      ASIN=ACOS=0
      REALKDX=6.283185/NCELLS
      DO FT000 I=0,NCELLSM1
      ARG=REALKDX*FLOAT(I)
      ELECTRIC=FLOAT(IE(I))*ESCALE
      ASIN=ASIN+SIN(ARG)*ELECTRIC
      ACOS=ACOS+COS(ARG)*ELECTRIC
FT000  CONTINUE
      PHASEHIST(ITIMESTEP)=ATAN2(ASIN,ACOS)
      AMPHIST(ITIMESTEP)=SQRT(ASIN**2+ACOS**2)
      ESQRHIST(ITIMESTEP)=TESQR
      IF(MOD(ITIMESTEP,IPLLOTINTVAL).EQ.0)CALL PLOTS
      IF(IXVPLOTINTVAL).BACKPLOTXV00.
      IF(MOD(ITIMESTEP,IXVPLOTINTVAL)).PLOTXV00.
BACKPLOTXV00 CONTINUE
      TIME=TIME+DT
      QVSET(0,ISUMIX(0),NCELLS+1)

      LCMINDEX=1
      DO LCMLP00 ILCMBUF=1,NBUFS
      LENBLOCK=LENSCMXV
      IF(ILCMBUF.EQ.(NBUFS0+1))LENBLOCK=NBUFXTRA
      READ LCM(IVX(1),IVXLCM(LCMINDEX),LENBLOCK)
      KXVEND=LENBLOCK
      CALL MOVELESP(IVX,IE,IDE,ISUMIX)
      WRITE LCM(IVX(1),IVXLCM(LCMINDEX),LENBLOCK)
      LCMINDEX=LCMINDEX+LENBLOCK
LCMLP00 CONTINUE

TIMELOOP00 CONTINUE
      LINEARPLOT(0.,DT,PHASEHIST(0),MAXTIMESTEPS+1)

```

```

      CRTBCD(=HIST PHASE FUNDAMENTAL=)
      SEMILOGPLT(0.,DT,AMPHIST(0),MAXTimesteps+1)
      CRTBCD(=HIST AMP FUNDAMENTAL=)
      LINEARPLOT(0.,DT,ESQRHIST(0),MAXTimesteps+1)
      CRTBCD(=HIST ESQR=)
      CALL EXIT(1)

      ENTRY PLOTS
      IP=0
      TKE=0.
      NSHRIV2=NFRACT+LOG2DTSQR/2-5

      DO RH000 IX=0,NCELLSM1
      RHO(IX)=FLOAT(IRHO(IX)/NPCELL)*OTWONFRACT
RH000  CONTINUE
      LINEARPLOT(0.,DX,RHO(0),NCELLS)
      WRITE(100,FMTRHO)TIME
FMTRHO  FORMAT(=ELECTRON CHARGE DENSITY AT TIME=,F10.3)

      QVSET(0,FV(0),130)
      QVSET(0,IRHO(0),NCELLS+1)
      DVP=5./64.
      ODVP=1./DVP
      DO TKEP00 K=1,NPTCLES
      IV=IVXLCM(K).SHR.NBITSX
      ICELL=(IVXLCM(K).INT.-Q8MOM(NBITSV)).SHR.NFRACT
      IRHO(ICELL)=IRHO(ICELL)+IV $$$ VDRIFT CALC
      IP=IP+IV
      V=FLOAT(IV)*VELSCALE
      TKE=TKE+V**2
      IVFV=IFIX(V*ODVP+65.)
      IF(IVFV)TKEP00,TKEP00,
      IF(IVFV-129).TKEP00,TKEP00
      FV(IVFV)=FV(IVFV)+1.
TKEP00  CONTINUE
      TKE=TKE*.5*ONPTCLES
      TP=FLOAT(IP)*VELSCALE*ONPTCLES
      WRITE(3,FMTF)TIME,TESQR,TKE,TP

      DO VDRIFT00 ICELL=0,NCELLS
VDRIFT00  RHO(ICELL)=FLOAT(IRHO(ICELL)/NPCELL)*VELSCALE
      LINEARPLOT(0.,DX,RHO(0),NCELLS)
      WRITE(100,FMTVDRI)TIME
FMTVDRI  FORMAT(=DRIFT VELOCITY AT TIME=,F10.3)

      DV=.125*FLOAT(NCELLS)/SYSLEN
      SEMILOGPLT(-64.*DVP,DVP,FV(1),129)
      WRITE(100,FMTFV)TIME
FMTFV  FORMAT(=F(V) AT TIME=,F10.3)

      DO EPL0T00 IX=0,NCELLS
      RHO(IX)=FLOAT(IE(IX))*ESCALE
EPL0T00  CONTINUE
      LINEARPLOT(0.,DX,RHO(0),NCELLS)
      WRITE(100,FMTEPLOT0)TIME
FMTEPLOT0  FORMAT(=ELECTRIC FIELD AT TIME=,F10.3)
      RETURN PLOTS

      PLOTXV00  CONTINUE
      CALL FRAMEID

```

```

      MAPS(0.,SYLEN,-4.5,4.5,.1,.999,.15,.999)
      DO VXPLOT00 K=1,NPTCLES
      X=FLOAT(IVXLCM(K).INT.-QBMMOM(NBITSV))*XSCALE
      V=FLOAT(IVXLCM(K).SHR.NBITSX)*VELSCALE
      CALL POINT(X,V)
VXPLOT00 CONTINUE
      WRITE(100,FMTXV0)TIME
FMTXV0  FORMAT('ELECTRON PHASE SPACE AT TIME',F10.3)
      GO TO BACKPLOTXV00

```

```

CRASH00 CONTINUE
      IWRITE=3
      FMTPRINT='FMTG'
      PRINTCOM(KXVEND,ENDCOMR)
      FMTPRINT='FMT0'
      PRINTCOM(ICELECTRIC,ENDCOM0)
      WRITE(3,FMT0)IDE
      WRITE(3,FMT0)IE
      WRITE(3,FMT0)IRHO
      WRITE(3,FMT0)ISUMIX
      CALL EXIT(1)
      END

```

\$\$\$ ELECTRON AND IONS TOGETHER TEST

```

$CHAT (BINELS,KENTLIB,ORDERLIB,STACKLIB)%ME,EX,BIN,EF,V24,S,S,,L
      PROGRAM TT(OUTPUT,TAPE3=OUTPUT)
      PARAMETER(NCELLS=16)
      PARAMETER(NPCELL=512)
      PARAMETER(NPTCLES=NPCELL*NCELLS)
      PARAMETER(NFRACT=25)
      PARAMETER(P1=3.141592654,TWOP1=6.28318530P)
      PARAMETER(NCELLSM1=NCELLS-1)
      PARAMETER(MAXTIMESTEPS=2000)
      PARAMETER(LENSCMXV=512)

      LCM (LCMIVX)
      COMMON/LCMIVX/IVXLCM(NPTCLES)
      LCM (LCMIVXIONS)
      COMMON/LCMIVXIONS/IVXIONSLCM(NPTCLES)
      COMMON/CIVX/IVX(LENSCMXV+2)
      COMMON/CIE/IE((0,NCELLS+1))
      COMMON/CIDE/IDE((0,NCELLS+1))
      COMMON/CISUMIX/ISUMIX((0,NCELLS+1))
      COMMON/CIRHO/IRHO((0,NCELLS+1))
      LCM (LCMIRHOELE)
      COMMON/LCMIRHOELE/IRHOLCMELE((0,NCELLS+1))
      COMMON/CRHO/RHO((0,NCELLS+1))
      COMMON/CFV/FV((0,129))
      COMMON/CMOVELES/KXVEND,KXVNOW,TWONFRACT,OTWONFRACT
      $,LOG2NPCELL,LOG2DTSQR,NBITSCELL,NSHRRHO,NBITSX,NBITSV
      $,EPSILON,DT
      $,NBITSMIOME,NBITSRTMIOME,TEOTI,OVILOAD
      $,VISCALE,OVISCALE,NBITSSHIONS,RMIOME
      $,ADDVEL,SYLEN,VEOV,EPSONV,XSCALE,OXSCALE,ESCALE,OESCALE
      $,DX,VSCALE,OVSCALE,ONPTCLES,ONCELLS
      $,I45MNFRACT,NBUFS0,NBUFS,NBUFXTRA,LENBLOCK,ILCMBUF
      $,I,ITIMESTEP,IPL0TINTVAL,IXVPL0TINTVAL
      $,REALKIOAD,DXL,X,V,ENDCOMR
      COMMON/COMOCT/ICELECTRIC,ICRHO,ICRHOC

```



```

$.IBITSHL45,IRHO1,ICELL
$,TIMESONESISUMDXCELL,ISUMONES
$,IX,IV
$,IEZ,MASKCHK,ENDCOMO
COMMON/COMPRINT/IWRITE,FMTPRINT,FMTprt((0,11)),LIMPRT
LCM (COMHIST)
COMMON/COMHIST/OSCILLOSCOPE((0,MAXTimesteps))
$      ,ESQRHIST((0,MAXTimesteps))
$      ,AMPHIST((0,MAXTimesteps))

DATA DT/.25/,EPSILON/.08/
DATA SYLEN/16./
DATA NBITSMIOME/6/ $$$ 6>64, 8>256
DATA TEOTI/30./
DATA ADDVEL/0./
DATA IPLOTINTVAL/128/,IXVPLTINTVAL/0/

FMTE FORMAT(10E12.4)
FMTF FORMAT(10F12.6)
FMTO FORMAT(/5022)
FMTI FORMAT(1X,10I10)

CHANGE(□+EX□)
ZDS(10,6ROUTPUT,ISS,ERRZDS00)
ZDS(10,5RDWFMX,ISS,ERRZDS00)
ZDS(10,5RDWFMY,ISS,ERRZDS00)
ZDS(10,5RDWFM0,ISS,ERRZDS00)
ERRZDS00 CONTINUE
CRTID(□WFM□,1)
CALL KEEP80(5RDWFM0)
LOG2NPCELL=IFIX(ALOG(FLOAT(NPCELL))*1.442695+.1)
LOG2OTSQR=IFIX(ALOG(DT*DT))*1.442695-.1)
NBITSCELL=IFIX(ALOG(FLOAT(NCELLS))*1.442695+.1)
NSHRRHO=1+LOG2NPCELL-LOG2OTSQR
NBITSX=NFRACT+NBITSCELL
NBITSV=60-NBITSX
IRHO1=NPCELL.SHL.(NFRACT+1)
I45MNFRACT=45-NFRACT
TWONFRACT=2.*NFRACT
OTWONFRACT=1./TWONFRACT
XSCALE=SYLEN*OTWONFRACT/FLOAT(NCELLS) $$$ IN DEBYELENGTHS
OYSCALE=1./XSCALE
DX=SYLEN/FLOAT(NCELLS)
$$$ CELL SIZE ACTUALLY USED IS 1, SO MUST SCALE V.E APPROPRIATELY
VSCALE=XSCALE/DT
OVSCALE=1./VSCALE
ESCALE=XSCALE/(DT**2)
OESCALE=1./ESCALE
ONPTCLES=1./FLOAT(NPTCLES)
ONCELLS=1./NCELLS

NBITSRTMIOME=NBITSMIOME.SHR.1
RMIOME=2.*NBITSMIOME
VSCALE=VSCALE
OVSCALE=OVSCALE
OVILOAD=OVSCALE/SQRT(RMIOME*TEOTI)

NBUFS0=NPTCLES/LENSCMXV
NBUFXTRA=NPTCLES-NBUFS0*LENSCMXV
NBUFS=NBUFS0

```

```

21
IF (NBUFEXTRA.NE.0) NBUFS=NBUFS0+1

DATA LOCEPSILON/.LOC.EPSILON/
WRITE(59,0(06)0)LOCEPSILON
MOVCHK(NFRACT,NBITSCELL,IERSW)
IF(IERSW)CALL EXIT(10)

IBITSHL45=1.SHL.45
DXL=SYLEN/FLOAT(NPTCLES)
X=-DXL
VPOVE=SQRT((SYLEN/TWOPI)**2+3.)
EPSILONV=EPSILON*VPOVE
WRITE(59,FMTF)VPOVE
REALKLOAD=TWOPI/FLOAT(NPTCLES)
DO LD000 K=1,NPTCLES
ARG=REALKLOAD*FLOAT(K)
PERT=EPSILON*COS(ARG)
PERTV=PERT*VPOVE
X=X+DXL*(1.+PERT)
IX=(IFIX(X*OXSCALE)).INT.-Q8MOM(NBITSV)
ICELL=IX.SHR.NFRACT
ISUMIX(ICELL)=ISUMIX(ICELL)
$+(IBITSHL45.UN.IX)
Q=0.
DO LD002 L=1,12
LD002 Q=Q+RNFL(0)
VEL=Q-6.
IV=(IFIX(VEL*OVSCALE).SHL.NBITSX).INT.Q8MOM(NBITSV)
IVXLCM(K)=IX.UN.IV
VEL=Q-6.+PERTV
IV=(IFIX(VEL*OVILOAD).SHL.NBITSX).INT.Q8MOM(NBITSV)
IVXIONSLCM(K)=IX.UN.IV
LD000 CONTINUE

SETCH(0.,42.)
DO WRITE00 IWRITE=3,100,97
WRITE(IWRITE,FMTI)NFRACT,NPCELL,NCELLS
WRITE00 PRINTCOM(KXVEND,ENDCOMR)

DO TIMELOOP00 ITIMESTEP=0,MAXTIMESTEPS

CALL MOVFROMLCM(IVXLCM) $$$ ELECTRONS
WRITELCM(IRHO(0),IRHOLCMELE(0),NCELLS+1)

DO IONIE00 I=0,NCELLS
IE(I)=-(IE(I).SHR.NBITSMIOME)
IDE(I)=-(IDE(I).SHR.NBITSMIOME)
IONIE00 CONTINUE
CALL MOVFROMLCM(IVXIONSLCM) $$$ MOVE IONS
OSCILLOSCOPE(ITIMESTEP)=FLOAT(IRHO(0)/NPCELL)

READLCM(ISUMIX(0),IRHOLCMELE(0),NCELLS+1)
DO RHOSUM00 I=0,NCELLS
RHOSUM00 IRHO(I)=IRHO(I)-ISUMIX(I) $$$ ION RHO - ELECTRON RHO

$$$ GAUSS'S LAW
IE(0)=0
DO GAUSS00 IX=0,NCELLS-2
IE(IX+1)=IE(IX)+IRHO(IX)+IRHO(IX+1)
GAUSS00 CONTINUE

```

```

      IEZ=0
      DO AVE00 IX=0,NCELLSM1
      IEZ=IEZ+IE(IX)
AVE00  CONTINUE
      IEZ=IEZ/NCELLS
      DO AVE01 IX=0,NCELLSM1
      IE(IX)=IEZ-IE(IX)   $$$ MAKE ELECTRIC FIELD NEGATIVE FOR ELECTRONS
AVE01  CONTINUE
      IE(NCELLS)=IE(0) $$$ BE SURE TO MAKE DE(NCELLS-. ) OK
      DO AVE02 IX=0,NCELLSM1
      IDE(IX)=(IE(IX+1)-IE(IX)).SHR.NSHRRHO
AVE02  CONTINUE
      DO AVE03 IX=0,NCELLSM1
AVE03  IE(IX)=IE(IX).SHR.NSHRRHO
$$$ CHECK TO SEE IF AVERAGE IE IS ZERO
      ICELECTRIC=ICRHO=0
      DO ZCHECK00 IX=0,NCELLSM1
      ICELECTRIC=ICELECTRIC+IE(IX)
      ICRHO=ICRHO+IRHO(IX)
ZCHECK00 CONTINUE
      IF(IABS(ICRHO).GT.7B)GO TO CRASH00
      IF(IABS(ICELECTRIC/NCELLS).GT.7B)GO TO CRASH00

      TESQR=0
      DO TESQR00 IX=0,NCELLSM1
      TESQR=TESQR+FLOAT(IE(IX))*2
TESQR00 CONTINUE
      TESQR=TESQR*ESCALE*ESCALE*.5*ONCELLS
      ASIN=ACOS=0
      REALKDX=6.283185/NCELLS
      DO FT000 I=0,NCELLSM1
      ARG=REALKDX*FLOAT(I)
      ELECTRIC=FLOAT(IE(I))*ESCALE
      ASIN=ASIN+SIN(ARG)*ELECTRIC
      ACOS=ACOS+COS(ARG)*ELECTRIC
FT000  CONTINUE
      AMPHIST(ITIMESTEP)=SQRT(ASIN**2+ACOS**2)
      ESQRHIST(ITIMESTEP)=TESQR
      IF(MOD(ITIMESTEP,IPL0TINTVAL).EQ.0)CALL PLOTS
      IF(IXVPL0TINTVAL).BACKPLOTXV00,
      IF(MOD(ITIMESTEP,IXVPL0TINTVAL)).PLOTXV00,
BACKPLOTXV00 CONTINUE
      TIME=TIME+DT

TIMELOOP00 CONTINUE
      LINEARPLOT(0.,DT,OSCILLOSCOPE(0),MAXTIMESTEPS+1)
      CRTBCD(□HIST PHASE FUNDAMENTAL□)
      SEMILOGPLT(0.,DT,AMPHIST(0),MAXTIMESTEPS+1)
      CRTBCD(□HIST AMP FUNDAMENTAL□)
      LINEARPLOT(0.,DT,ESQRHIST(0),MAXTIMESTEPS+1)
      CRTBCD(□HIST FSQR□)
      CALL EXIT(1)

      ENTRY MOVFROMLCM(LCMARRAY)
      QVSET(0,ISUMIX(0),NCELLS+1)
      INTEGER LCMARRAY(1)
      LCM (LCMARRAY)
      LCMINDEX=1
      DO LCMLP00 ILCMBUF=1,NBUFS
      LENBLOCK=LENSCMXV

```

```

      IF (LCMBUF.EQ.(NBUF50+1))LENBLOCK=NBUFEXTRA
      READ LCM(IVX(1),LCMARRAY(LCMINDEX),LENBLOCK)
      KXVEND=LENBLOCK
      CALL MOVELESP(IVX,1E,1DE,ISUMIX)
      WRITE LCM(IVX(1),LCMARRAY(LCMINDEX),LENBLOCK)
      LCMINDEX=LCMINDEX+LENBLOCK
LCMLP00  CONTINUE
      IRHO(0)=0
      MASKCHK=1.SHL.44
      DO LP70 I=0,NCELLSM1
        IF (ISUMIX(I))CRASH00,,$$$ CHECK FOR OVERFLOW
        IF (ISUMIX.INT.MASKCHK)CRASH00,,$$
        ICELLTIMESONES=(1*(ISUMIX(1).SHR.45)).SHL.NFRACT
        ISUMDXCELL=(ISUMIX(1).INT.-Q8MOM(18))-ICELLTIMESONES
        ISUMONES=(ISUMIX(1).INT.Q8MOM(18)).SHR.145MNFRACT
        IRHO(1)=IRHO(1)+ISUMONES-ISUMDXCELL
        IRHO(I+1)=ISUMDXCELL
LP70  CONTINUE
      IRHO(0)=IRHO(0)+IRHO(NCELLS)
      IRHO(NCELLS)=0
      RETURN MOVFROMLCM

      ENTRY PLOTS

      SPECPLOTS(IVXIONSLCM,.5/64.,$ IONS$)
      READLCM(IRHO(0),IRHOLCMELE(0),NCELLS+1)
      SPECPLOTS(IVXLCM,.5/64.,$ELECTRONS$)
      GO TO AROUND00
      ENTRY SPECPLOIS(IVXARRAYLCM,DVP,LABELSPECIES)
      LCM(IVXARRAYLCM)
      INTEGER IVXARRAYLCM(1)
      DO RH000 IX=0,NCELLSM1
        RHO(IX)=FLOAT(IRHO(IX)/NPCELL)*OTWONFRACT
RH000  CONTINUE
      LINEARPLOT(0,DX,RHO(0),NCELLS)
      WRITE(100,FMTRHO)LABELSPECIES,TIME
FMTRHO  FORMAT(A10,$ CHARGE DENSITY AT TIME$,F10.3)

      QVSET(0,FV(0),130)
      QVSET(0,IRHO(0),NCELLS+1)
      ODVP=1./DVP
      IP=0
      TKE=0.
      DO TKEP00 K=1,NPTCLES
        IVXSCM=IVXARRAYLCM(K)
        IV=IVXSCM.SHR.NBITSX
        ICELL=(IVXSCM.INT.-Q8MOM(NBITSV)).SHR.NFRACT
        IRHO(ICELL)=IRHO(ICELL)+IV $$$ VDRIFT CALC
        IP=IP+IV
        V=FLOAT(IV)*VSCALE
        TKE=TKE+V**2
        IVFV=FIX(V*ODVP+65.)
        IF (IVFV)TKEP00,TKEP00,
        IF (IVFV-129),TKEP00,TKEP00
        FV(IVFV)=FV(IVFV)+1.
TKEP00  CONTINUE
      TKE=TKE*.5*ONPTCLES
      TP=FLOAT(IP)*VSCALE*ONPTCLES
      IF (LABELSPECIES.EQ.$ELECTRONS$)GO TO ELE00
      TKEIONS=TKE*RMIONE

```

```

      TPIONS=TP*RM1OME
ELE00  CONTINUE

      DO VDRIFT00 ICELL=0,NCELLS
VDRIFT00  RHO(ICELL)=FLOAT(IRHO(ICELL)/NPCELL)*VSCALE
          LINEARPLOT(0.,DX,RHO(0),NCELLS)
          WRITE(100,FMTVDRIFT)LABELSPECIES,TIME
FMTVDRIFT  FORMAT(A10,DRIFT VELOCITY AT TIME,F10.3)

          SEMILOGPLT(-63.*DVP,DVP,FV(1),129)
          WRITE(100,FMTFV)LABELSPECIES,TIME
FMTFV  FORMAT(A10,EF(V) AT TIME,F10.3)
      RETURN SPECPLOTS
AROUND00  CONTINUE

      TOTENERGY=TKEIONS+TKE+TESQR
      TOTP=TPIONS+TP
      WRITE(3,FMTF)TIME,TESQR,TKE,TP
      $,TKEIONS,TPIONS,TOTENERGY,TOTP

      DO EPLOT00 IX=0,NCELLS
      RHO(IX)=FLOAT(IE(IX))*ESCALE
EPLOT00  CONTINUE
          LINEARPLOT(0.,DX,RHO(0),NCELLS)
          WRITE(100,FMTEPLOT0)TIME
FMTEPLOT0  FORMAT(ELECTRIC FIELD AT TIME,F10.3)
      RETURN PLOTS

PLOTXV00  CONTINUE
      PSPLIT(4.5,1VXLCM,ELECTRON)
      PSPLIT(.5,1VXIONSLCM,ION)
      GO TO BACKPLOTXV00
      ENTRY PSPLIT(VMAXPS,LCMARRAYPS,LABELPS)
      LCM(LCMARRAYPS)
      INTEGER LCMARRAYPS(1)
      CALL FRAMEID
      MAPS(0.,SYLEN,-VMAXPS,VMAXPS,.1,.999,.15,.999)
      DO VXPL0T00 K=1,NPTCLES
      X=FLOAT(LCMARRAYPS(K).INT.-QBMM(NBITSV))*XSCALE
      V=FLOAT(LCMARRAYPS(K).SHR.NBITSX)*VSCALE
      CALL POINT(X,V)
VXPL0T00  CONTINUE
          WRITE(100,FMTXV0)LABELPS,TIME
FMTXV0  FORMAT(A9,PHASE SPACE AT TIME,F10.3)
      RETURN PSPLIT

CRASH00  CONTINUE
      IWRITE=3
      FMTPRINT=FMTG
      PRINTCOM(KXVEND,ENDCOMR)
      FMTPRINT=FMT0
      PRINTCOM(ICELECTRIC,ENDCOM0)
      WRITE(3,FMT0)IDE
      WRITE(3,FMT0)IE
      WRITE(3,FMT0)IRHO
      WRITE(3,FMT0)ISUMIX
      CALL FXI(1)
      END

```

\$\$\$ ELECTRONS AND IONS, FREE EXPANSION

\$CHAT XME,,BIN,EF,V24,M,M,,L

PROGRAM WFM(DWFMPRT,TAPE3=DWFMPRT

\$,DECODE=20,TAPE63=DECODE)

OPTIMIZE

PARAMETER(NCELLS=2048) \$\$\$ MUST BE POWER OF 2

PARAMETER(NPCELL=256) \$\$\$ MUST BE POWER OF 2

PARAMETER(NCELLBEGIN=NCELLS-100)

PARAMETER(NCELLEND=NCELLS)

PARAMETER(NCELLSLOAD=NCELLEND-NCELLBEGIN)

PARAMETER(NPARTICLES=NCELLSLOAD*NPCELL)

PARAMETER(NFRACT=23)

PARAMETER(PI=3.141592654,TWOP1=6.283185308)

PARAMETER(NCELLSM1=NCELLS-1)

PARAMETER(MAXTimesteps=131072)

PARAMETER(1TimestoreIntval=256

\$,LENHIST=MAXTimesteps/1TimestoreIntval)

PARAMETER(LENHEADER=1,KVXBEGIN=LENHEADER+1)

PARAMETER(LENSCMVXBLOCK=512+LENHEADER)

PARAMETER(NBLOCKS=128) \$\$\$ SET BY HAND

PARAMETER(LCMLENGTHS=NPCELL*NCELLSLOAD+NBLOCKS*LENHEADER)

LCM (LCMIVX)

COMMON/LCMIVX/1VXLCM(LCMLENGTHS)

LCM (LCMIVXION)

COMMON/LCMIVXION/1VXIONLCM(LCMLENGTHS)

COMMON/CIVX/1VX(LENSCMVXBLOCK+2)

COMMON/CIE/IE((0,NCELLS+1))

COMMON/CIDE/IDE((0,NCELLS+1))

COMMON/CISUMIX/ISUMIX((0,NCELLS+1))

COMMON/CIRHO/IRHO((0,NCELLS+1))

LCM (LCMIRHOELE)

COMMON/LCMIRHOELE/IRHOLCMELE((0,NCELLS+1))

COMMON/CRHO/RHO((0,NCELLS+1))

COMMON/CFV/FV((0,129))

INTEGER IFV((0,129))

EQUIVALENCE(1FV(0),FV(0))

COMMON/CMOVELES/KVXEND,KVXNOW,TWONFRACT,OTWONFRACT

\$,LOG2NPCELL,LOG2DTSQR,NBITSCELL,NSHRRHO,NBITSX,NBITSV

\$,NBITSMIOME,OVILOAD

\$,VISCALE,OVISCALE,NBITSSHLION

\$,XSCALE,OXSCALE,ESCALE,OESCALE

\$,DX,VSCALE,OVSCALE,ONPARTICLES,ONCELLS

\$,145MNFRACT,NBUFS0,NBUFS,LENBUFEXTRA,LENBLOCK

\$,DXL,ENDCOMR

COMMON/INPUTCOM/1Timestep,TIME

\$,SYLEN,SYLENLOAD

\$,DT,1PLOTINTVAL,1XVPL0TINTVAL,TEOT1,RMIOME

\$,TEHOTOTE,NRATIOOTEOTEHOT

\$,ENDCOMINPUT

COMMON/ENERGYCOM/TKE,TKEELECTRONLEFT,TKEELECTRONRIGHT

\$,TKEION,TKEIONLEFT,TKEIONRIGHT,TESQR,TOTENERGY

COMMON/MOMENTUMC/TP,TPELECTRONLEFT,TPELECTRONRIGHT

\$,TPION,TPIONLEFT,TPIONRIGHT,TOTP

COMMON/COMOCT/ICRHO,ICELL

\$,IRHOIONLEFT,IRHOELECTRONLEFT

\$,IRHOIONRIGHT,IRHOELECTRONRIGHT

\$,IX,IV,1TEMPORARY,IXLOADRT,IRHORIGHT

\$,ENDCOMO

COMMON/LRRCOM/KVXNOWPREV,NCOUNTPREV

```

$,IVINDEX,IVINDEXIONGT4,IVINDEXELECTRONGT4,I,K,ILCMBUF
$,X,V
$,ENDCOMERR
  LCM (COMHIST)
  COMMON/COMHIST/ISCOPEELE((0,3),(0,LENHIST))
$,      ISCOPEION((0,3),(0,LENHIST))
$,      ,ESQRHIST((0,LENHIST))
  COMMON/COMVBINS/IELEVBINS((0,20)),IONVBINS((0,20))
  COMMON/DISKIOCOM/IDISKAD,MINUSWORD,ITYPEIO,ERRADRIO
$,SPAREIO(10)
  COMMON/IBETACOM/IBETA((0,15))
  COMMON/GOBCOM/IGOBCOMA((0,40))
  COMMON/COMSTARTTM/LLLPLT(3),LSTIME,LSDATE,NTIME,NDATE
$,MESSAGEPLOT(15),SPAREPLOT(5)
  COMMON/COMCTRLE/IPRINTCTRLE,DUMCOMCTRLE(36)
  COMMON/COMPRINT/IWRITE,FMTPRINT,FMTPR((0,11)),LIMPRT

  COMMON/TALKFLOECOM/NAMECODE,NAMEDROP,NAMEPRT
$,TIMECODE,TIMESTOP,PLOTINTVAL,LUSERNO,USER9999
$,MICUSED,MICUSEDPP,MICUSEDSTRT,MAXMINUTES
$,NAMERESTART1,NAMERESTART2,NAMECHAT,NAMELOD,NAMEEX
$,TLEOJ,BANKEOJ,SPAREFLO(5)
  DATA NAMECODE/□WFM□/,NAMEDROP/□+WFM□/,NAMEPRT/7ROWFMPRT/
  DATA NAMECHAT/2REF/,NAMELOD/3REFL/,NAMEEX/6RLISTEL/

  COMMON/ELFCOM/IELFSW,MCTELF(3),IELFDIRC1,IELFDIRC2,MCTELFEND(7)
  DATA MCTELF(1)/□ SUE. CHECK ELF WRS L .WFM:□/
  DATA IELFDIRC1/□WFMSTORE1:□/
  DATA MCTELFEND(1)
$,/□FILES ALWITH +GIE END□/
  DATA MCTELFEND(7)/77B/
  COMMON/MCTTAPECOM/MCTTAPE(8),IOUTTAPE,IOUTTAPE2,MCTTAPEEND(3)
  DATA MCTTAPE(1)
$,/□SUI. CHECK FILES ALWITH +GIE END□/
  DATA MCTTAPE(8)/□TAPES □/
  COMMON/TAPEMESCOM/IIAPEMES(9)
  COMMON/FROGCOM/IFROGSW, IFROGLINE1(3),IFROGLINEEND
  DIF(NCELLS-200)1..
  DATA IFROGSW/1/
  DATA IFROGLINE1/□FAM. DWFM0□/

  DATA DT/.25/
  DATA SYLEN/3000./
  DATA RM10MF/256./ $$$ NEEDS TO BE POWER OF 2
  DATA TEOT1/3./
  DATA TEHOTOTE/10./
  DATA NRATIOTEOTEHOT/10/
  DATA IPLOTINTVAL/4096/ $$$ MAKE MULTIPLE ITIMESTOREINTVAL
  DATA IXVPLTINTVAL/4096/

FMTE  FORMAT(10E12.4)
FMTE  FORMAT(10F12.6)
FMTO  FORMAT(/5022)
FMti  FORMAT(1X,10110)
FMtA  FORMAT(12A10)

      NAMECHAT=2REF
      NAMELOD=3REFL
      NAMEEX=6RLISTEL

```



```

ZDS(10,6ROUTPUT,ISS,ERRZDS00)
ZDS(10,5RDWFMX,ISS,ERRZDS00)
ZDS(10,5RDWFMY,ISS,ERRZDS00)
ZDS(10,5RDWFM0,ISS,ERRZDS00)
ERRZDS00 CONTINUE
CRTID(0WFM0,1)
CALL KEEP80(5RDWFM0)
CLOCK(1ISCLTIME,1SDATE)
IGOBCOMA(448)=1ISCLTIME
IGOBCOMA(458)=1SDATE
GOB(2412B,1ER,0,MICUSEDSTRT) $$$ INITIALIZE TIMER

LOG2NPCELL=FIX(ALOG(FLOAT(NPCELL))*1.442695+.1)
LOG2DTSQR=FIX(ALOG(DT*DT))*1.442695-.1)
NBITSMIOME=FIX(ALOG(RMIOME))*1.442695+.1)
NBITSCELL=FIX(ALOG(FLOAT(NCELLS))*1.442695+.1)
NSHRRHO=1+LOG2NPCELL-LOG2DTSQR
NBITSX=NFRACT+NBITSCELL
NBITSV=60-NBITSX
IRHO1=NPCELL.SHL.(NFRACT+1)
145MNFRACT=45-NFRACT
TWNFRACT=2.*NFRACT
OTWONFRACT=1./TWNFRACT
XSCALE=SYSLEN*OTWONFRACT/FLOAT(NCELLS) $$$ IN DEBYELENGTHS
OXSCALE=1./XSCALE
DX=SYSLEN/FLOAT(NCELLS)
$$$ CELL SIZE ACTUALLY USED IS 1, SO MUST SCALE V.E APPROPRIATELY
VSCALE=XSCALE/DT
OVSCALE=1./VSCALE
ESCALE=XSCALE/(DT*.2)
OESCALE=1./ESCALE
ONPARTICLES=1./FLOAT(NPARTICLES)
ONCELLS=1./NCELLS
IXLOADRT=-Q8MOM(NBITSV).INT.Q8MOM(NBITSV+NBITSCELL+1) $$$ CENTER CELL
IRHORIGHT=(1.SHL.45)+IXLOADRT
QVSET(0,1SCOPEELE(0,0),(8*(LENHIST+1)))

VISCALE=VSCALE
OVISCALE=OVSCALE
OVILOAD=OVSCALE/SQRT(RMIOME*TEOT1)

NBUFS0=NPARTICLES/(LENSCMVXBLOCK-LENHEADER)
LENBUFEXTRA=NPARTICLES-NBUFS0*(LENSCMVXBLOCK-LENHEADER)
NBUFS=NBUFS0
IF(LENBUFEXTRA).EXACTBUF00,
NBUFS=NBUFS0+1
LENBUFEXTRA=LENBUFEXTRA+LENHEADER
EXACTBUF00 CONTINUE
IF(NBUFS.GT.NBLOCKS)CALL CRASH(0NBLOCKS.LT0)

DATA LOCDT/.LOC.DT/
WRITE(59,(06)0)LOCDT
MOVCHK(NFRACT,NBITSCELL,1ERSW)
IF(1ERSW)CALL CRAE(0MOVER0)

SYSLENLOAD=DX*FLOAT(NCELLEND-NCELLBEGIN)
DXL=SYSLENLOAD*ONPARTICLES
OVSCALELOAD=OVSCALE
VMAXPLOT=4.
IF(NRATIO10TEOTEHOT).EQ000,

```

```

      VMAXPLOT=4.*SQRT(TEHOTOTE)
EQ000  CONTINUE
      SETCH(0.,42.)
      DO WRITE00 IWRITE=3,100.97
      WRITE(IWRITE,FMT1)NFRACT,NPCELL,NCELLS,NCELLBEGIN,NCELLEND,NPARTICLES
      PRINTCOM(ITIMESTEP,ENDCOMINPUT)
WRITE00  PRINTCOM(KVXEND,ENDCOMR)
      REWIND 63
      WRITE(63,FMT630)TEHOTOTE,NRATIOOTEOTEHOT,RMIOME
      $.TEOTI,SYSELOAD
FMT630  FORMAT(TEHOT/TE=□,F4.0,□ TE/TEHOT=□,13.□ MI/ME=□
      $,F5.0,□ TE/TI=□,F4.0,□ PLASLEN=□,F6.0,□ DEBYELEN=□)
      READ(63,FMTA)(MESSAGEPLOT(IM),IM=1,9)
      DATA MESSAGEPLOT(11)/KENT ESTABROOK BOX V24 L 545 X4291□/

      FACENERGY=.5
      NRATIOOTEOTEHOTLOAD=NRATIOOTEOTEHOT
      OVSCALELOADHOT=OVSCALELOAD*SQRT(TEHOTOTE)
      CALL LOADPTCLES(IVXLCM)
      FACENERGY=.5*RMIOME
      OVSCALELOADHOT=OVSCALELOAD=OVLOAD
      NRATIOOTEOTEHOTLOAD=0
      CALL LOADPTCLES(IVXIONLCM)
      GO TO AROUNDLOAD00

      ENTRY LOADPTCLES(LCMLOAD)
      LCM(LCMLOAD)
      INTEGER LCMLOAD(1)
      QVSET(0,FV(0),'29)
      X=-DXL+DX*FLOAT(NCELLBEGIN)
      IFENERGY=0.
      NCOUNTHOT=NCOUNT=0
      LCMINDEX=1
      DO LCMLP01 ILCMBUF=1,NBUFS
      LCNBLOCK=LENSCHVXBLOCK
      IF(ILCMBUF.EQ.(NBUFS0+1))LCNBLOCK=LENBUFEXTRA
      IVX(1)=LCNBLOCK
      DO LD000 K=KVXBEGIN,IVX(1)
      X=X+DXL
      IX=(IFIX(X*OXSCALE))
      IF((IX.SHR.NFRACT).GT.NCELLSMI)CALL CRASH(LOAD□)
      Q=0.
      DO LD002 L=1,12
LD002  Q=Q+RNFL(0)
      VEL=Q-6.
      OVSCALELOAD0=OVSCALELOAD
      IF(NRATIOOTEOTEHOTLOAD),LD003,
      ICT=ICT+1
      IF(ICT.LT.NRATIOOTEOTEHOTLOAD)GO TO LD003
      ICT=0
      NCOUNTHOT=NCOUNTHOT+1
      OVSCALELOAD0=OVSCALELOADHOT
LD003  CONTINUE
      IV0=IFIX(VEL*OVSCALELOAD0)
      IF(((IV0.SHL.NBITSX).SHR.NBITSX).NE.IV0)CALL CRASH(VMAX.GT.AL□)
      IV=(IV0.SHL.NBITSX).INT.Q8MOM(NBITSV)
      IVX(K)=IX.UN.IV
      NCOUNT=NCOUNT+1
      V=FLOAT(IVX(K).SHR.NBITSX)*VSCALE
      ENERGY=V*V*FACENERGY

```

```

      TENERGY=TENERGY+ENERGY
      IENERGY=IFIX(10.*ENERGY)
      IF(IENERGY.GT.129)GO TO LD000
      IFV(IENERGY)=IFV(IENERGY)+1
LD000  CONTINUE
      WRITE LCM(IVX(1),LCMLOAD(LCMINDEX),LENBLOCK)
      LCMINDEX=LCMINDEX+LENBLOCK
LCMLP01 CONTINUE
      IWRITE=3
      PRINTCOM(IFV(0),IFV(60))
      KCT=0.
      DO PCT00 K=0,129
        KCT=KCT+IFV(K)
        FV(K)=FLOAT(IFV(K))
PCT00  CONTINUE
      IWRITE=3
      PRINTCOM(IFV(0),IFV(60))
      TENERGY=TENERGY*ONPARTICLES
      WRITE(3,FMT1)TENERGY
      WRITE(3,FMT1)KCT,NCOUNTHOT,NCOUNT
      SEMILOGPLT(.05,.1,FV(0),129)
      IF(NCOUNT.NE.NPARTICLES)CALL CRASH('NCOUNT')
      RETURN LOADPTCLES
AROUNDLOAD00 CONTINUE

      DO TIMELOOP00 ITIMESTEP=0,MAXTIMESTEPS+1

      IPLOTSW=0
      IF(MOD(ITIMESTEP,IPLOTINTVAL).EQ.0)IPLOTSW=1

      IONSW=0
      CALL MOVFROMLCM(IVXLCM) $$$ ELECTRONS
      WRITELCM(IRHO(0),IRHOLCMELF(0),NCELLS+1)

      IONSW=1.
      DO IONIE00 I=0,NCELLS
        IE(I)=-(IE(I).SHR.NBITSMIOME)
        IDE(I)=-(IDE(I).SHR.NBITSMIOME)
IONIE00 CONTINUE
        IDE(NCELLSMI)=IDE(NCELLS)=0 $$$ PREVENT LOOPING
        CALL MOVFROMLCM(IVXIONLCM) $$$ MOVE ION
        LABELSPECIES=' ION'
        IF(IPLOTSW)CALL RHOPL0T

      READLCM(ISUMIX(0),IRHOLCMELE(0),NCELLS+1)
      DO RHOSUM00 I=0,NCELLS
RHOSUM00 IRHO(I)=IRHO(I)-ISUMIX(I) $$$ ION RHO - ELECTRON RHO

      $$$ GAUSS'S LAW
      IRHO(0)=IRHO(0)+((IRHOIONLEFT-IRHOELECTRONLEFT).SHL.NFRACT)
      IE(0)=IRHO(0) $$$ SINCE ALL RHO'S COUNTED TWICE
      IRHO(NCELLS+1)=0
      DO GAUSS00 IX=0,NCELLS
        IE(IX+1)=IE(IX)+IRHO(IX)+IRHO(IX+1)
GAUSS00 CONTINUE
      ITIMESTOREPREV=ITIMESTORE
      ITIMESTORE=ITIMESTEP/ITIMESTOREINTVAL

      IF(ITIMESTOREPREV.EQ.ITIMESTORE)GO TO INTIMEST00

```

```

      TESQR=0
      DO TESQR00 IX=0,NCELLSM1
      TESQR=TESQR+FLOAT(IE(IX).SHR.NSHRRHO)**2
TESQR00  CONTINUE
      TESQR=TESQR*ESCALE*ESCALE*.5/FLOAT(NCELLSLOAD)
      ESQRHIST(ITIMESTORE)=TESQR
      WRITE(3,FMTA)ELECTRON VBINS
      IWRITE=3
      PRINTCOM(IELEVBINS(0),IELEVBINS(20))
      WRITE(3,FMTA)ION VBINS
      PRINTCOM(IONVBINS(0),IONVBINS(20))
      ITI=ITE=0
      DO IZVBINS K=0,20
      ITI=ITI+IONVBINS(K)
      ITE=ITE+IELEVBINS(K)
IZVBINS  IELEVBINS(K)=IONVBINS(K)=0
      WRITE(3,FMTI)ITE,ITI
INTIMEST00 CONTINUE

      IF(IPLOTSW)CALL PLOTS
      IF(IXVLOTINTVAL),BACKPLOTXV00,
      IF(ITIMESTEP),BACKPLOTXV00,
      IF(MOD(ITIMESTEP,IXVLOTINTVAL)),PLOTXV00,
BACKPLOTXV00 CONTINUE
      DO AVE02 IX=0,NCELLSM1
      IDE(IX)=(IE(IX)-IE(IX+1)).SHR.NSHRRHO
      IE(IX)=-IE(IX).SHR.NSHRRHO $$$ MAKE E NEGATIVE FOR ELECTRON MOVER
AVE02  CONTINUE
      DIF(0),5,
      ICRHO=0
      DO ZCHECK00 IX=0,NCELLS
      ICRHO=ICRHO+IRHO(IX)
ZCHECK00 CONTINUE
      IF(IABS(ICRHO).GT.7B)CALL CRASH(=ICRHO)
      IF(IABS(IE(NCELLS+1)).GT.7B)CALL CRASH(=IE(NCELLS+1))

      TIME=TIME+DT
      TIMECODE=TIME
      CALL TALKFLOETTY
      IXVLOTINTVAL-IFIX(PLOTINTVAL/DT)
TIMELOOP00 CONTINUE
      CALL EOJPRT(=TIMELOOP)

      ENTRY MOVFROMLCM(LCMARRAY)
      QVSET(0,ISUMIX(0),NCELLS+1)
      INTEGER LCMARRAY(1)
      LCM (LCMARRAY)
      LCMINDEX=1
      DO LCMLP00 ILCMBUF=1,NBUFS
      LENBLOCK=LENSCMVXBLOCK
      IF(ILCMBUF.EQ.(NBUFS0+1))LENBLOCK=LENBUFEXTRA
      READ LCM(IX(1),LCMARPAY(LCMINDEX),LENBLOCK)
      KVXNOW=KVXBEGIN
MOVER00  CONTINUE
      KVXEND=IX(1)
      IF(IX(1).LE.LENHEADER)GO TO ENDBLOCK00
      IF(KVXNOW-(KVXEND+1)),ENDBLOCK00,CRASH01
      CALL MOVELEEX(IX,IE,IDE,ISUMIX)
      IF(KVXNOW-(KVXEND+1)),ENDBLOCK00,CRASH01

```

```

        ICELL=((IVX(KVXNOW).INT.-QBMM(NBITSV)).SHR.NFRACT)
        IV=IVX(KVXNOW).SHR.NBITSX
        V=FLOAT(IV)*VSCALE
        IF(ICELL.LT.3)GO TO LEFTSIDE00
        IF(ICELL.GE.(NCELLS-3))GO TO RIGHTSIDEBOTHSPEC00
        CALL CRASH(ICELL)
LEFTSIDE00 CONTINUE
        IF(IONSW)GO TO IONLEFT00
        TKEELECTRONLEFT=TKEELECTRONLEFT+V**2*.5*ONPARTICLES
        TPELECTRONLEFT=TPELECTRONLEFT+V
        IRHOELECTRONLEFT=IRHOELECTRONLEFT+1
        IVS=IFIX(-V*16.) $$$ 16 BUCKETS
        IPARCEL=MOD(IVS,4)*15 $$$ FOR PACKING
$$$ LOW ORDER BITS HAVE SMALLER V'S
        IF(IPARCEL.GT.45)CALL CRASH(IPAR.GT.45)
        IVINDEX=IVS/4
        IF(IVS.LT.0)CALL CRASH(IVS)
        IF(IVS.GT.20)IVS=20
        IELEVBINS(IVS)=IELEVBINS(IVS)+1
        IF(IVINDEX.LT.4)GO TO INELECTRONSLEFT00
        IVINDEXELECTRONGT4=IVINDEXELECTRONGT4+1
        IVINDEX=3
        IPARCEL=45
INELECTRONSLEFT00 CONTINUE
        ISCOPEELE(IVINDEX,ITIMESTORE)=ISCOPEELE(IVINDEX,ITIMESTORE)
        $+(1.SHL,IPARCEL)
SWAP00 CONTINUE
        ITEMPORARY=IVX(KVXNOW)
        IVX(KVXNOW)=IVX(KVXEND)
        IVX(KVXEND)=ITEMPORARY
        IVX(1)=IVX(1)-1 $$$ RESET NUMBER PARTICLES
        GO TO MOVER00

IONLEFT00 CONTINUE
        TKEIONLEFT=TKEIONLEFT+V**2*.5*RMIONE*ONPARTICLES
        TPIONLEFT=TPIONLEFT+V*RMIONE*ONPARTICLES
        IRHOIONLEFT=IRHOIONLEFT+1
        IVS=IFIX(-V*16.) $$$ 16 BUCKETS
        IPARCEL=MOD(IVS,4)*15 $$$ FOR PACKING
        IF(IPARCEL.GT.45)CALL CRASH(IPAR.GT.45)
        IF(IPARCEL.LT.0)CALL CRASH(IPAR.LT.0)
        IVINDEX=IVS/4
        IF(IVS.LT.0)CALL CRASH(IVS)
        IF(IVS.GT.20)IVS=20
        IONVBINS(IVS)=IONVBINS(IVS)+1
        IF(IVINDEX.LT.4)GO TO INIONLEFT00
        IVINDEXIONGT4=IVINDEXIONGT4+1
        IVINDEX=3
        IPARCEL=45
INIONLEFT00 CONTINUE
        ISCOPEION(IVINDEX,ITIMESTORE)=ISCOPEION(IVINDEX,ITIMESTORE)
        $+(1.SHL,IPARCEL)
        GO TO SWAP00

RIGHTSIDEBOTHSPEC00 CONTINUE
        IF(IONSW)GO TO IONRIGHT00
        IPELECTRONRIGHT=TPELECTRONRIGHT+2.*V*ONPARTICLES
        TKEELECTRONRIGHT=TKEELECTRONRIGHT+.5*V*V*ONPARTICLES
        IRHOELECTRONRIGHT=IRHOELECTRONRIGHT+1
        GO TO ONRIGHT00

```

```

IONRIGHT00 TPIONRIGHT=TPIONRIGHT+2.*V*RM10ME*ONPARTICLES
TKEIONRIGHT=TKEIONRIGHT+V*V*.5*RM10ME*ONPARTICLES
IRHOIONRIGHT=IRHOIONRIGHT+1
ONRIGHT00 CONTINUE
IX=IXLOADRT
ISUMIX(NCELLSM1)=ISUMIX(NCELLSM1)+IRHORIGHT
IV=(-IVX(KVXNOW)).INT.Q8MOM(NBITSV) $$$ REFLECT VELOCITY
IVX(KVXNOW)=IV.UN.IX
KVXNOW=KVXNOW+1
GO TO MOVER00

CRASH01 CALL CRASH(IVXNOW.GT)

ENDBLOCK00 CONTINUE
WRITE LCM(IVX(1),LCMARRAY(LCMINDEX),LENBLOCK)
LCMINDEX=LCMINDEX+LENBLOCK
LCMLP00 CONTINUE
IRHO(0)=0
DO LP70 I=0,NCELLSM1
ICELLTIMESONES=(1*(ISUMIX(1).SHR.45)).SHL.NFRACT
ISUMDXCELL=(ISUMIX(1).INT.-Q8MOM(15))-ICELLTIMESONES
ISUMONES=(ISUMIX(1).INT.Q8MOM(15)).SHR.145MNFRACT
IRHO(1)=IRHO(1)+ISUMONES-ISUMDXCELL
IRHO(I+1)=ISUMDXCELL
LP70 CONTINUE
RETURN MOVFROMLCM

ENTRY PLOTS

SPECPLOTS(IVXIONLCM,1.2/64.)
READLCM(IRHO(0),IRHOLCMELE(0),NCELLS+1)
LABELSPECIES=PELECTRONS
CALL RHOPLOT
SPECPLOTS(IVXLCM,VMAXPLOT/64.)
GO TO AROUND00

ENTRY RHOPLOT
DO RH000 IX=0,NCELLSM1
RHO(IX)=FLOAT(IRHO(IX)/NCELL)*OTWONFRACT
RH000 CONTINUE
SEMILOGPLT(0.,DX,RHO(0),NCELLS)
WRITE(100,FMTIRHO)LABELSPECIES,TIME
FMTIRHO FORMAT(A10,PE CHARGE DENSITY AT TIMEPE,F10.3)
RETURN RHOPLOT

ENTRY SPECPLOTS(IVXARRAYLCM,DVP)
LCM(IVXARRAYLCM)
INTEGER IVXARRAYLCM(1)
QVSET(0,FV(0),130)
QVSET(0,IRHO(0),NCELLS+1)
ODVP=1./DVP
IP=0
TKE=0.
LCMINDEX=1
DO LCMLP03 ILCMBUF=1,NBUFS
LENBLOCK=LENSCMVXBLOCK
IF(ILCMBUF.EQ.(NBUFS+1))LENBLOCK=LENBUFXTRA
READ LCM(IVX(1),IVXARRAYLCM(LCMINDEX),LENBLOCK)
DO TKEP00 K=KVXBEGIN,IVX(1)
IV=IVX(K).SHR.NBITSX

```

```

      DIF(0),2.
      ICELL=(IVX(K).INT.-Q8MOM(NBITSV)).SHR.NFRACT
      IRHO(ICELL)=IRHO(ICELL)+IV $$$ VDRIFT CALC
      IP=IP+IV
      V=FLOAT(IV)*VSCALE
      TKE=TKE+V**2
      IVFV=IFIX(V*ODVP+65.)
      IF(IVFV)TKEP00,TKEP00,
      IF(IVFV-129),TKEP00,TKEP00
      FV(IVFV)=FV(IVFV)+1.
TKEP00  CONTINUE
      LCMINDEX=LCMINDEX+LENBLOCK
LCMLP03  CONTINUE
      TKE=TKE*.5*ONPARTICLES
      TP=FLOAT(IP)*VSCALE*ONPARTICLES
      IF(LABELSPECIES.EQ.¤ELECTRONS¤)GO TO ELE00
      TKEION=TKE*RMIONE
      TPION=TP*RMIONE
ELE00  CONTINUE

      DIF(0),FMTVDRIFT,
      DO VDRIFT00 ICELL=0,NCELLS
VDRIFT00  RHO(ICELL)=FLOAT(IRHO(ICELL)/NPCELL)*VSCALE
      LINEARPLOT(0.,DX,RHO(0),NCELLS)
      WRITE(100,FMTVDRIFT)LABELSPECIES,TIME
FMTVDRIFT  FORMAT(A10,¤DRIFT VELOCITY AT TIME¤,F10.3)

      SEMILOGPLT(-63.*DVP,DVP,FV(1),129)
      WRITE(100,FMTFV)LABELSPECIES,TIME
FMTFV  FORMAT(A10,¤F(V) AT TIME¤,F10.3)
      RETURN SPECPLOTS
AROUND00  CONTINUE

      TOTENERGY=TKEION+TKE+TESQR
      $+TKCELECTRONLEFT+TKEIONLEFT
      TOTP=TPION+TP
      $+TPELECTRONRIGHT+TPIONRIGHT+TPELECTRONLEFT+TPIONLEFT
      CALL FRAME
      SETCH(0.,42.,0,0,0)
      WRITE(59,FMTF)TIME
      DO WRITE01 IWRITE=3,100,97
      PRINTCOM(ITIMESTEP,ENDCOMINPUT)
      PRINTCOM(TKE,TOTENERGY)
      PRINTCOM(TP,TOTP)
      PRINTCOM(KVXNOWPREV,ENDCOMERR)
WRITE01  CONTINUE

      DO EPLOT00 IX=0,NCELLS
      RHO(IX)=FLOAT(IE(IX).SHR.NSHRRHO)*ESCALE
EPLOT00  CONTINUE
      LINEARPLOT(0.,DX,RHO(0),NCELLS)
      WRITE(100,FMTEPLOT0)TIME
FMTEPLOT0  FORMAT(¤ELECTRIC FIELD AT TIME¤,F10.3)
      RETURN PLOTS

PLOTXV00  CONTINUE
      PSPLIT(VMAXPLOT,IVXICM,¤ELECTRON¤)
      PSPLIT(1.2,IVXIONLCM,¤ION ¤)
      GO TO BACKPLOTXV00
      ENTRY PSPLIT(VMAXPS,LCMARRAYPS,LABELPS)

```



```

      LCM (LCMARRAYPS)
      INTEGER LCMARRAYPS(1)
      CALL FRAMEID
      MAPS(0.,SYSLN,-VMAXPS,VMAXPS,.1,.999,.15,.999)
      LCMINDEX=1
      DO LCMLP04 !LCMBUF=1,NBUFS
      LENBLOCK=LENSCMVXBLOCK
      IF (LCMBUF.EQ.(NBUFS+1))LENBLOCK=LENBUFEXTRA
      READ LCM(IVX(1),LCMARRAYPS(LCMINDEX),LENBLOCK)
      DO VXPLOTOO K=KVXBEGIN,IVX(1)
      X=FLOAT(IVX(K).INT.-Q8MOM(NBITSV))*XSCALE
      V=FLOAT(IVX(K).SHR.NBITSX)*VSCALE
      CALL POINT(X,V)
VXPLOT00 CONTINUE
      LCMINDEX=LCMINDEX+LENBLOCK
LCMLP04 CONTINUE
      WRITE(100,FMTXV0)LABELPS,TIME
FMTXV0 FORMAT(1A9,1 PHASE SPACE AT TIME1,F10.3)
      RETURN PSLOT

      ENTRY EOJPRT(LABELEOJPRT)
      WRITE(3,FMTA)LABELEOJPRT
      WRITE(59,FMTA)LABELEOJPRT
      DTHIST=DT*FLOAT(ITIMESTOREINTVAL)
      LINEARPLOT(0.,DTHIST,ESQRHIST(0),ITIMESTORE+1)
      CRTBCD(1HIST ESQR1)
      HISTPKG(1SCOPEELE,1 ELECTRONS1)
      HISTPKG(1SCOPEION,1 IONS1)
      CALL CRASH(1EOJ1)
      ENTRY HISTPKG(1SCOPEPEC,1 LABELHIST)
      LCM (1SCOPEPEC)
      INTEGER 1SCOPEPEC(10,3),(0,LENHIST)
      DO HISTELEPLOT00 K=0,ITIMESTORE+1
      IQ=0.
      DO IQTOT00 I=0,3
      ISCM=1SCOPEPEC(I,K)
      DO IQTOT01 IPACK=0,3
      IQ=IQ+(ISCM.INT.-Q8MOM(45))
      ISCM=ISCM.SHR.15
IQTOT01 CONTINUE
IQTOT00 CONTINUE
      RHO(K)=FLOAT(IQ)
HISTELEPLOT00 CONTINUE
      LINEARPLOT(DTHIST*.5,DTHIST,RHO(0),ITIMESTORE+1)
      WRITE(100,FMTHIST0)LABELHIST
FMTHIST0 FORMAT(1NUMBER OF1,A10,1 COLLECTED VS TIME (ALL ENERGIES)1)
      VNOW=0.
      DO HISTPLOT00 I=0,3
      DO HISTPLOT01 IPACK=0,45,15
      DO HISTPLOT02 K=0,ITIMESTORE+1
      RHO(K)=FLOAT(1SCOPEPEC(I,K).SHR.IPACK).INT.-Q8MOM(45))
HISTPLOT02 CONTINUE
      VNOWP=VNOW+1./16.
      LINEARPLOT(DTHIST*.5,DTHIST,RHO(0),ITIMESTORE+1)
      WRITE(100,FMTHIST01)LABELHIST,VNOW,VNOWP
FMTHIST01 FORMAT(1NUMBER OF1,A10,1 COLLECTED VS TIME 1
      $/1FROM1, F6.3,1 TO 1,F6.3,1 ELECTRON THERMAL VELOCITIES1)
      VNOW=VNOWP
HISTPLOT01 CONTINUE
HISTPLOT00 CONTINUE

```

```

RETURN HISTPKG

ENTRY CRASH(LABELCRASH)
WRITE(3,FMTXX)LABELCRASH,CRASH,IVX(KVXNOW)
WRITE(59,FMTXX)LABELCRASH,CRASH,IVX(KVXNOW)
FMTXX FORMAT(=CRASH=,A10,06,022)
CALL FRAME
SETCH(0.,42.,0.0,0)
DO ICRASHWRITE00 IWRITE=3,100,97
FMTPRINT=FMTO
PRINTCOM(IT,IMESTEP,ENDCOMINPUT)
PRINTCOM(KVXNOWPREV,ENDCOMERR)
PRINTCOM(KVXEND,ENDCOMR)
PRINTCOM(TKE,TOTENERGY)
PRINTCOM(TP,TOTP)
FMTPRINT=FMTO
PRINTCOM(ICRHO,ENDCOMO)
ICRASHWRITE00 CONTINUE
IF(LABELCRASH.NE.=EOJ=)GO TO ONCRASH00
CALL TALKFLOEOJ(ICRHO)
GO TO TIMELOOP00
ONCRASH00 CONTINUE
PAUSE
WRITE(3,FMTO)IDE
WRITE(3,FMTO)IE
WRITE(3,FMTO)IRHO
WRITE(3,FMTO)ISUMIX
CALL EXIT(1)
END

```

NOTICE

"This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research & Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights."

"Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Energy Research & Development Administration to the exclusion of others that may be suitable."

Technical Information Department

LAWRENCE LIVERMORE LABORATORY

University of California | Livermore, California | 94550