9.0 What is a User Program File?

- A user program file is an ASCII file with the .PRG filename extension that contains the user program segments for use with a potential array (.PA or .PA0).
- It shares the name of the potential array it supports. TEST.PRG would automatically be used to support the TEST.PA array.
- User programs are associated with arrays not instances.
How SIMION Utilizes User Programs

- Each time the Fly’、“m button is clicked SIMION automatically compiles all user program files for any instances that have user programs attached to their associated potential array.

- If Adjustable variables are found, the user will be given a screen to view and change their initial values.

- User program segments are called only when ions are flying within the array that the user program segments support.
User Program Segments

- The user program file contains user program segments.
- User program segments act like subroutines.
- SIMION calls each type of program segment at specific points in a trajectory calculation.
- Thus you must use a specific user program segment to control a specific aspect of the trajectory calculation.
The 9 Program Segments

- Initialize
- Init_P_Values
- Tstep_Adjust
- Fast_Adjust

- Efield_Adjust
- Mfield_Adjust
- Accel_Adjust
- Terminate
- Other_Actions
Integration into SIMION

Block Diagram of Trajectory Calculations and Where User Program Segments are Called

- **Start Fly'm**
- Compile All User Programs, RESET ADJUSTABLE VARIABLES and ARRAYS and Let User ACCESS ADJUSTABLE VARIABLES
- While Single Run Not Made or in Rerun Mode
  - Create all Ions - Then SEG INITIALIZE each Ion
  - Before Flying any Ions Call SEG INIT_P_VALUES
- While Ions Remain to be Flown
  - Before Flying Next Ion or Group - RESET STATIC VARIABLES AND ARRAYS
  - Select Next Ion or Group and Start to Fly it
- While Selected Ion or Group Still Flying
  - Determine Next Time Step to Use - Then SEG TSTEP_ADJUST
  - Single Time Step Integration Controller
  - SEG OTHER_ACTIONS
  - Compute Ion Replsions - If Active
  - Find Electrostatic Instance (If Any) - Then SEG FAST_ADJUST
  - Compute E Fields - Then SEG EFIELD_ADJUST
  - Compute E Acceleration - Then SEG ACCEL_ADJUST
  - Find Magnetic Instance (If Any) - Then SEG FAST_ADJUST
  - Compute M Fields - Then SEG MFIELD_ADJUST
  - Compute M Acceleration - Then SEG ACCEL_ADJUST
  - After ALL Ions Splat Call SEG TERMINATE
- If PE Pot. Update Call SEG FAST_ADJUST
- Restore any SEG INIT_P_VALUES and/or PE Updated Potentials
- Apply Relativistic Total Acceleration Corrections as Required
- End of Fly'm

The Idaho National Engineering and Environmental Laboratory
Define_Adjustable Viscous_Damping 0 ; adjustable variable Viscous_Damping
; set to 0 (no viscous damping by default)
; adjustable at the beginning of each

Fly’m

Begin_Segment Accel_Adjust ; start of Accel_Adjust program segment

Recall Ion_Ax_mm ; recall current x acceleration (mm/usec^2)
Recall Ion_Vx_mm ; recall current x velocity (mm/sec)
Recall Viscous_Damping ; recall the viscous damping term
Multiply ; multiply times x velocity
Subtract ; and subtract from x acceleration
Store Ion_Ax_mm ; return adjusted value to SIMION

Recall Ion_Ay_mm ; recall current y acceleration (mm/usec^2)
Recall Ion_Vy_mm ; recall current y velocity (mm/sec)
Recall Viscous_Damping ; recall the viscous

; exit to SIMION (optional statement)
Bad Style - no Advantage

Runs no faster than a fully commented style.

defa viscous_damping 0,seg accel_adjust rcl ion_ax_mm rcl ion_vx_mm rcl viscous_damping * - sto ion_ax_mm rcl ion_ay_mm rcl ionVy_mm rcl viscous_damping * - sto ion_ay_mm rcl ion_az_mm rcl ion_vz_mm rcl viscous_damping * - sto ion_az_mm
Language Rules

- Case is Ignored (upper and lower case are the same)
- Blank lines and indentation's are ignored
- The Semicolon ; Starts an In-Line Comment
- The language is based on the HP RPN calculator format with a 10 register rotary stack:

  \[
  25 \ 10 \ * \ 5 \ / \quad ; \quad \text{means} \ (25 * 10) / 5
  \]

- The Compiler looks for words and converts them into:
  - Commands
  - Numbers
  - Variable Names and Labels
Examples of Commands

+ or: Add

Adds contents of x and y registers, puts result in y-register, and renames it as x-register (e.g. 1 2 + becomes 3 in register where 1 was originally stored).

>KE or: Speed_to_Kinetic_Energy

Converts from speed (mm/usec) to kinetic energy (eV). On entry the x-register is assumed to contain the ion's speed and the y-register is assumed to contain the mass of the ion (amu). On exit the x-register contains the ion's KE and the y-register is unchanged. The >SPD command performs the reverse transformation.
Classes of Commands

- Calculations
  +  -  *  /  SIN, COS, LOG

- Transformations
  >DEG, >RAD, >KE, >P3D, >WBC

- Flow Controls
  LBL, GSB, RTN, EXIT

- Tests (do next command if true else skip next command)
  X=0, X<=0, X!=Y, X>Y

- Communication to User and Variables
  STO, RCL, R/S, KEY?, MESSAGE
Types of Variables

- **Adjustable Variables** (defined at top of program file)
  
  DEFA  MY_VARIABLE  30.0
  
  Global and lasts *throughout* a Fly’m

- **Static Variables** (defined at top of program file)
  
  DEFS  TIME_FLAG-2.0E4
  
  Global and Reset to initial value *before* each ion is flown

- **Temporary Variables**
  
  STO  LUCKY
  
  Local and Temporary: Created with STO command. Name must be *unique* (not Adjustable, Static, or Reserved variable name)

- **Reserved Variables**
  
  Ion_Color   Ion_Charge   Ion_Px_mm

  Used to communicate with SIMION. Each Program Segment has specific read and write access to various reserved variables.