**SimEscape Embedded Software Flowchart**

**Setupsensors() (Run At Startup)**
This function initializes the accelerometer and gyroscope for operation.

**loop() Data Send Loop (Run Infinitely)**

**getFullPacket()**
This function reads the current values from the accelerometer and gyroscope registers. The values are 16-bit integers read in 8 bits at a time. This function concatenates the upper 8 bits onto the lower 8 bits into a 16 bit integer variable. Six values are read: xAccl, yAccl, zAccl, xGyro, yGyro, and zGyro. For transmission, they are printed into a string in a tab delimited format beginning with an ASCII 'S' byte for start and an ASCII 'E' byte for end: sprintf(fullPacket,"%5.1f,%5.1f,%5.1f,%5.1f,%5.1f,%5.1f",xAcc, yAcc, zAcc, xGyro, yGyro, zGyro);

**printLine(char* inputString)**
This function accepts an input C-style string, and prints it to the RS-232 output buffer one character at a time, terminating with a newline and return. Therefore, data values are tab delimited and data sets are line delimited.

**RS-232 Core**
(Avalon Memory-Mapped Interface)

---

**SimEscape PC Input Mapping Software Flowchart**

**main() User Interface thread**
This function initializes the PC software user interface, which allows the user to activate, deactivate, and visualize the data acquisition thread. It also runs RS-232 port initialization.

**DAQThread() State Machine**

**getSensorData(IMUPACKET* packet) function**
This function flushes the RS-232 input buffer, then waits until at least 100 bytes are in the buffer. At this point, the thread will grab 100 bytes from the input buffer and parse those bytes for a complete data packet (start byte, xAccl, yAccl, zAccl, pitch, roll, yaw, and byte) which it will load into an IMUPACKET struct.

**calculateMovement(IMUPACKET* packet, long* xChange, long* yChange) function**
This function accepts an input IMUPACKET struct, and using the SENSITIVITY variable and sensor data, calculates pixel displacements for the current screen. These are saved in xChange and yChange.

**updateMousePosition(long x, long y) function**
This function accepts x and y displacements for the screen cursor. It will then offset the Windows cursor position from its old state to its new state in screen space utilizing the Win32 GetCursorPos() and SetCursorPos() functions.

**windows.h** (Allows access to the Win32 API, for reading and writing to the current Windows cursor position)