

## Software: Description

The software design of the DASS is centered around the **Data Capture** process. When the software begins execution, the main program forks into two processes, the *Socket Connect* (parent) and the *Data Capture* (child).

**Data Capture** polls the serial port for data, stores that data into memory, and then periodically saves the data onto Compact Flash for medium term storage. The determining factor for how often the data is saved to Compact Flash is the number of bytes to be stored. This was done in the interest of prolonging the life of the Compact Flash by minimizing the number of read/write cycles used by the software.

In order to download the data or upload a CR10X program onto the CR10X, the Data Capture process must be stopped. In the case of Data Download, Data Capture must merely save the data to the Compact Flash, allowing the user to have access to the most recent data available. The Program Upload on the other hand, requires full control of the serial port, meaning the Data Capture process must surrender the serial port since only one device may control it at a time. This means that outside processes must interact with the Data Capture process. This is where the Socket Connect comes in.

**Socket Connect** binds a character string address to itself.

Due to the fact that Socket Connect and Data Capture were spawned from the same process, this means that Socket Connect can communicate with Data Capture via its PID. This allows for a clean way to have processes such as Data Download and Program Upload to asynchronously communicate with Data Capture.

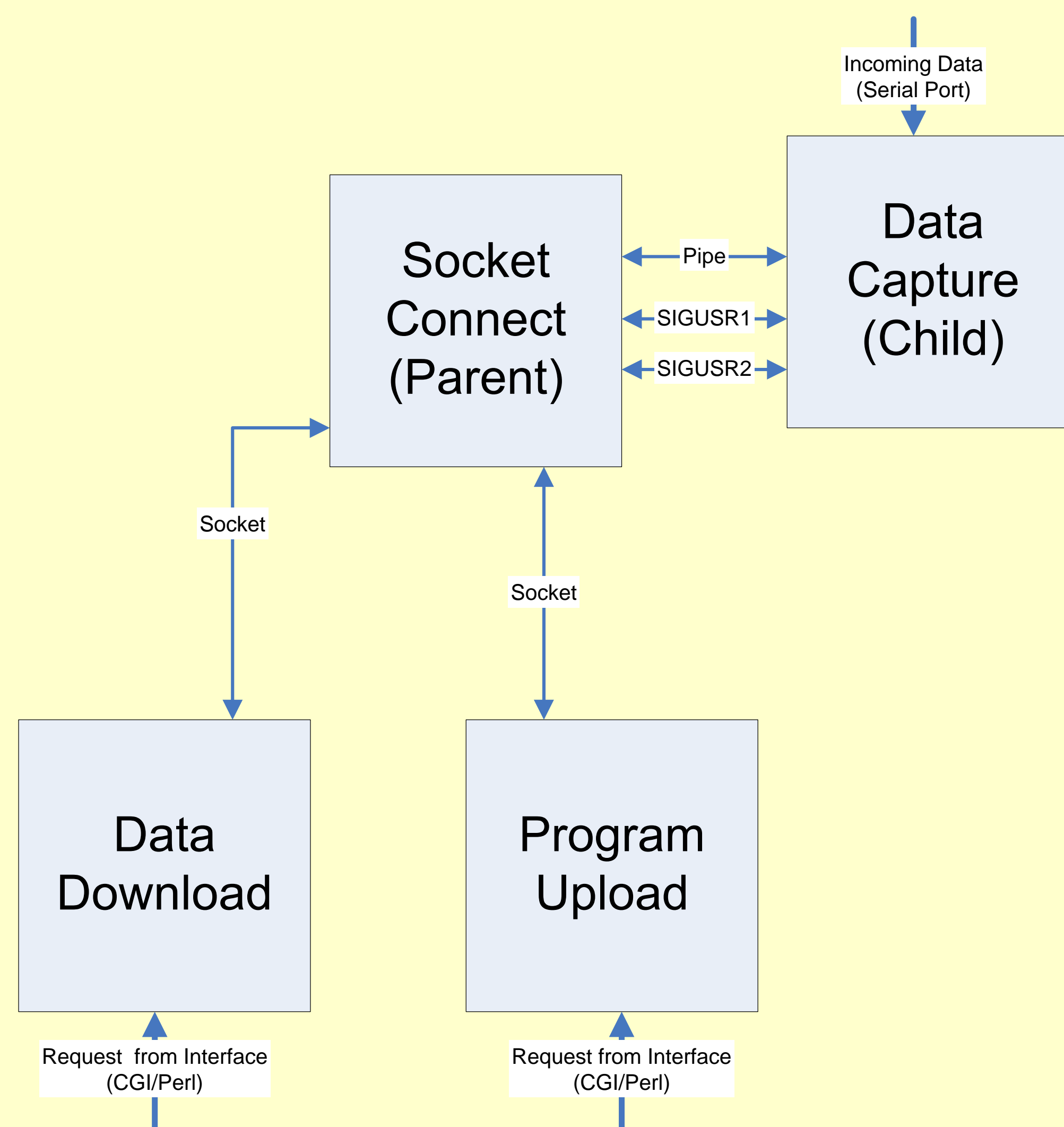
### **Software and GUI Interaction:**

The *Graphical User Interface (GUI)* was designed to allow the user to utilize the product seamlessly. By providing a clear set of point and click options, the DASS's GUI allows the user to utilize the powerful code that is implemented on the board to capture data, download data, and upload programs.

Since the GUI is based on *CGI* and *Perl*, the point of interaction between the C code and the CGI/Perl is the *syscall* command in Perl. By using *syscall*, the C code that is resident on the SBC can be called as an ordinary executable. *Syscall* also allows for the executable to return information, making it easy enough to make the Perl and C code interactive.

For example, when the user chooses to download data, before being taken to the listing screen on the GUI, the SBC executes the Data Download executable which causes the SBC to take all of the data saved into its buffer and save it out to the CF flash, into the appropriate file. This ensures that the .dat file that the user is downloading contains the most recent data possible.

## Software: Flow Chart



**STEVENS**  
Institute of Technology

### Sponsors



- **Professor Bruce McNair** — Senior Design Advisor
- The Electrical and Computer Engineering (ECE) Department
- The Following Individuals from Davidson Laboratories:
  - **Dr. Michael Bruno**
  - **Jeremy Turner**
  - **Dov Kruger**
  - **Donald Chelsey**