GUVI

Engineering POC

Patty Lew

Phone: (310) 336-0229

Fax: (310) 336-1636

e-mail: patricia_lew@qmail2.aero.org

The Aerospace Corporation

2350 E. El Segundo Blvd.

El Segundo, CA 90245

Software Preliminary Design Review, 16 May 1997

GUVI Engineering POC

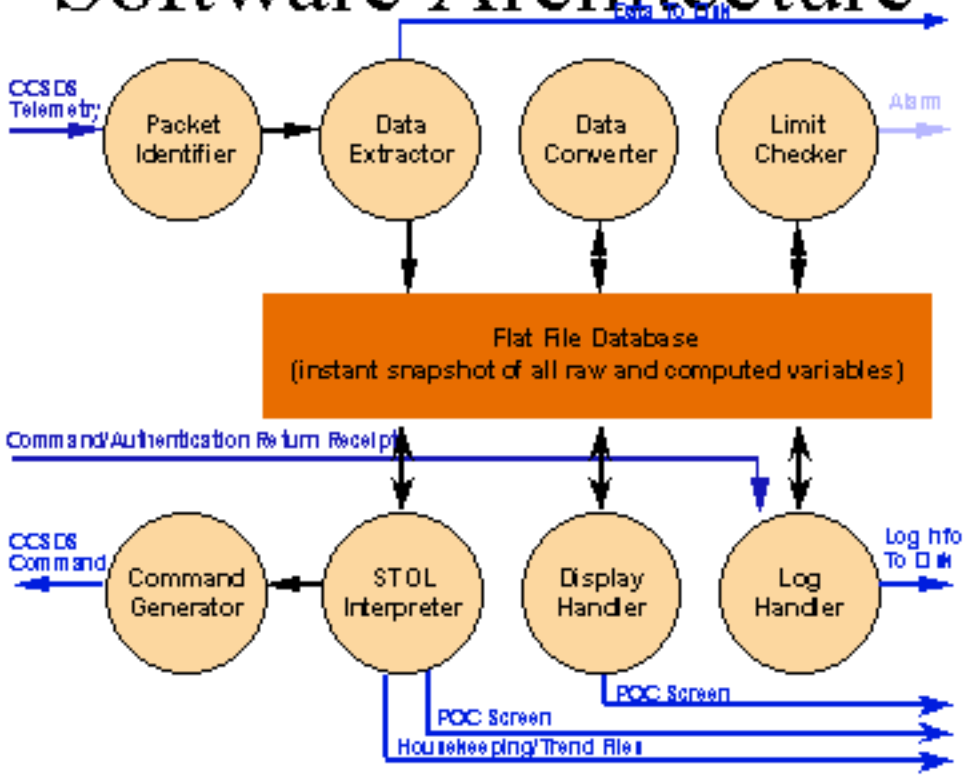
Purpose

- Fully tests and verifies every aspect of the GUVI ECU
- Monitors and archives data during system level calibration
- Ability to repeat the same functional test after each environmental test to determine liveness
- Capable of monitoring instrument life

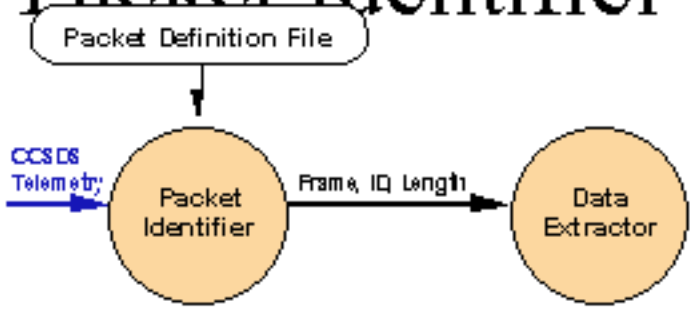
Software Requirements

- Receive and process telemetry
- Real time display capability
- Generate and transmit commands
- Script execution
- Log capability
- Alarm generation

Software Architecture



Packet Identifier

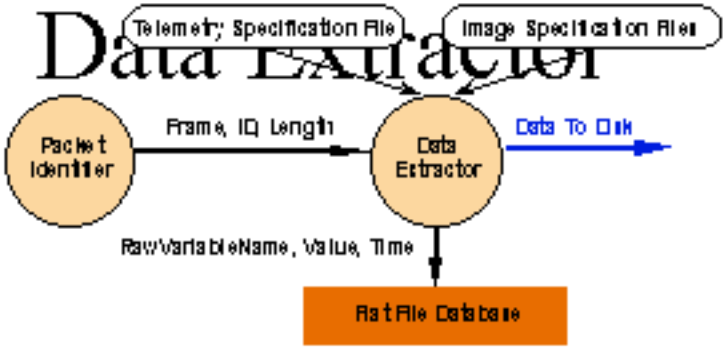


- Retrieve packet from memory buffer
- Validate packet given Packet Definition File specifications
- Pass valid packets to Data Extractor
- If SourceSeqCount not sequential, indicate sync loss in Status Window

Packet Definition File

- Tab delimited text file (i.e. spreadsheet)
- Each row defines an ApID to process

- Column Headers
  - Frame Name
  - Sub-Frame Size
  - Number of Sub-Frames
  - ID Sub-Frame
  - ID Start Byte



- Accept validated packets from Packet Identifier
- Store frame to disk
- Extract raw variables given Telemetry Specification File
  - Subcommed data allowed
  - Send "GUVI verified command" and "GUVI/Spacecraft ECU error" messages to Log

Telemetry Specification File

- Tab delimited text file
- Each row defines a new raw variable

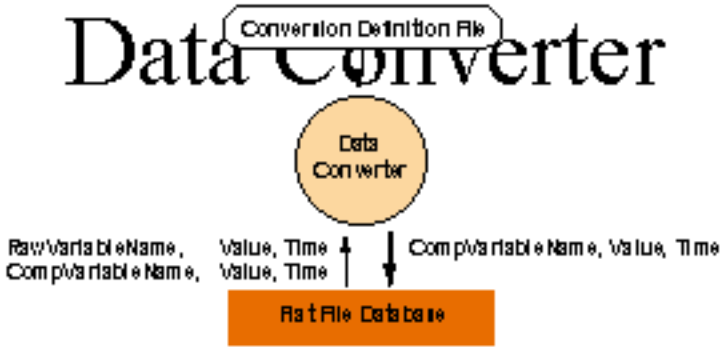
- Column Headers
  - Raw Variable Name
  - Description
  - Frame Name
  - Sub-Frame Number
  - Start Byte, Start Bit, Number of Bits
  - Time Offset (from start of frame)
  - Subcom 1 Name, Subcom 1 Low, Subcom

Image Specification File

- Tab delimited text file
- Given variables defined in Telemetry Specification File build an image variable

- Row 1: Image variable name
- Row 2: Image width and height
- Each remaining row defines an image pixel
  - Column headers
    - Horizontal Pixel

Data Converter

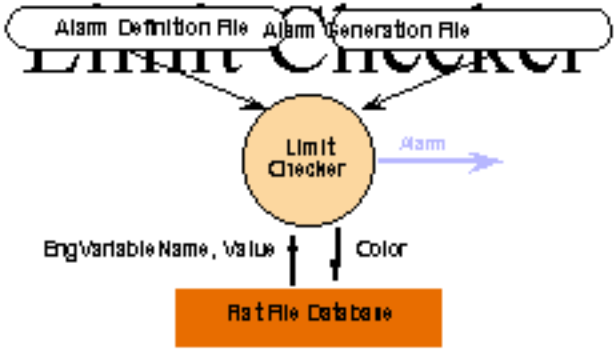


- Convert variables to engineering units given Conversion Definition File
  - RPN formulas with [+, -, *, /, LOG, LN, ^, &, |]
  - Piece-wise functions
  - Lookup tables
- Store computed variables into flat file database

Conversion Definition File

- Tab delimited text file
- Each row defines a new computed variable

- Column Headers
  - Computed Variable Name
  - Description
  - Formula
  - DependOn (when to recompute)
  - Subcom Name, Subcom Low, Subcom High



- Check computed variable against red/yellow limits defined in Alarm Definition file
  - Limits may depend on another variable (e.g. separate on/off limits)
  - Check performed when variable is computed
- Alarm Generation defined in respective file
  - Generate limit violation message to Log Handler Module

Alarm Definition File

- Tab delimited text file
- Each row defines a new set of limits

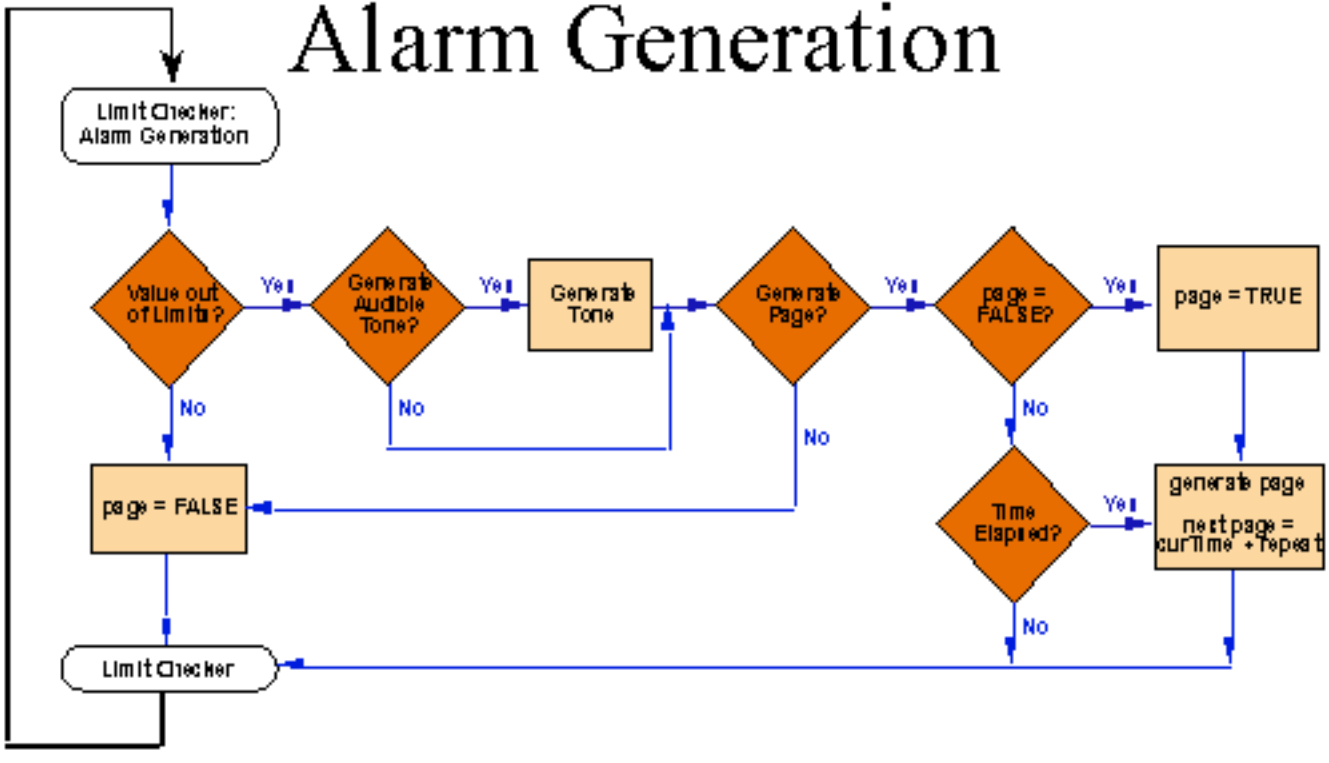
- Column headers
  - Computed Variable Name
  - Description
  - Red Low, Yellow Low, Yellow High, Red High
  - Subcom Name, Subcom Low, Subcom High
  - Alarm Type
    - L (Log limit violation)

Alarm Generation File

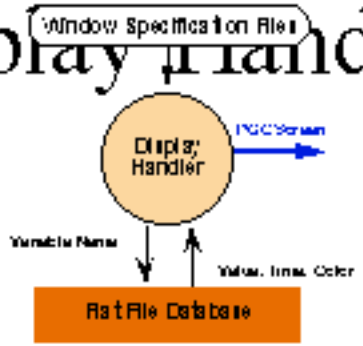
- Tab delimited text file
- Each row defines a new PageCode (from Alarm Definition File)

- Column Headers
  - PageCode
  - Phone Number
  - Error Code
  - Repeat (in minutes)

Alarm Generation

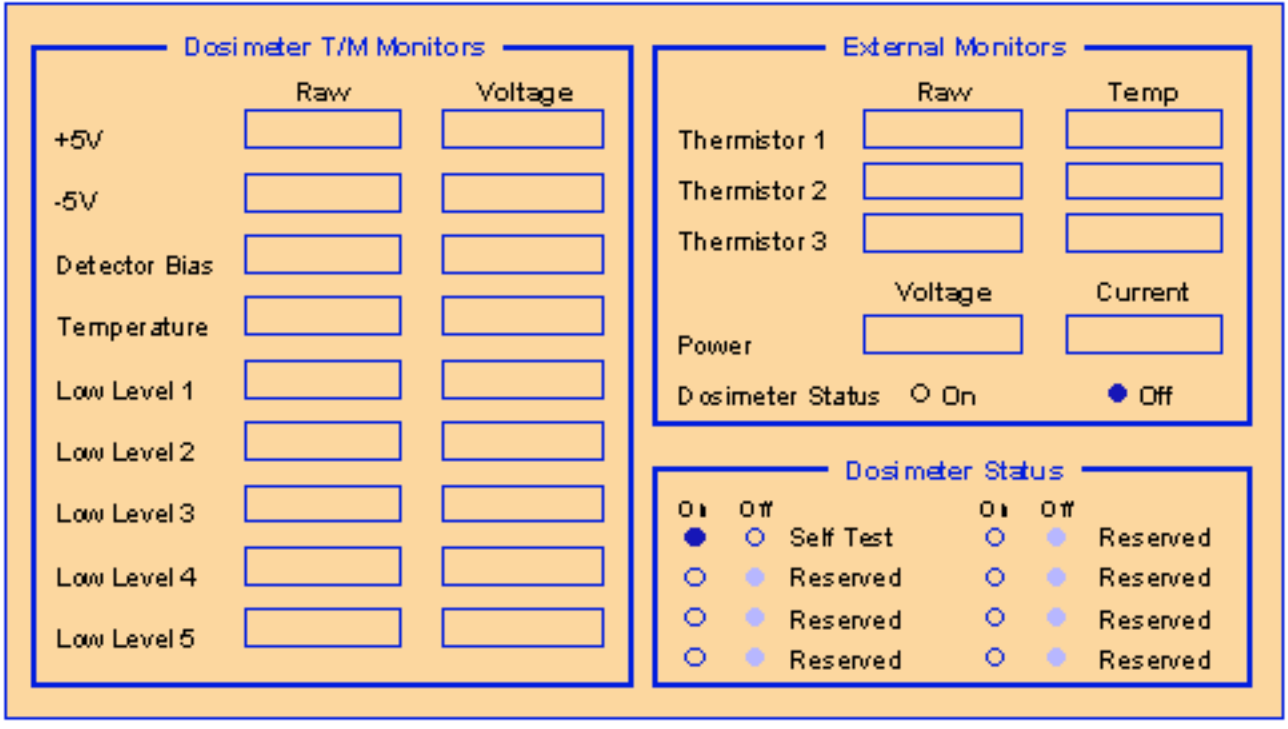


Display Handler



- Window Specification File defines how to draw a user window
- Window Types
  - Lists: 1D and 2D
  - Histograms: 1D (Y value indicates count) and 2D (intensity indicates count)
  - Complex:

Sample Complex Window



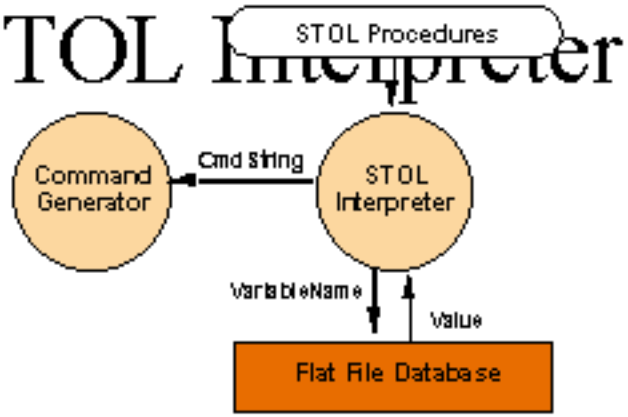
Window Definition Files

- 1 Dimensional List
  - Window Name, Window Type (1dList), Update Rate, Sort
  - Variable Name, Format (C format string)
- 2 Dimensional List
  - Window Name, Window Type (2dList), Update Rate
  - Variable Name, Format, Cell.h, Cell.v
- 1 Dimensional Histogram

Window Definition Files (continued)

- Complex Window
  - Window Name, Window Type (Complex), Update Rate, Width, Height
  - Rect, left, top, width, height, color
  - Text, left, bottom, string, font, size, bold, color
  - Frame, left, top, width, height, title, font, size, bold, color
  - Var, left, bottom, width, variable name, format string, font, size, bold, color, justify, box, instant update

STOL Interpreter



- Implement subset of GODDARD TPOCC STOL directives
- Execution initiated by selecting Run STOL from the Control menu
  - Read ASCII STOL file
  - Perform Syntax check of directives
  - Perform Lexical/Semantic analysis of directives

SSTOL Directives

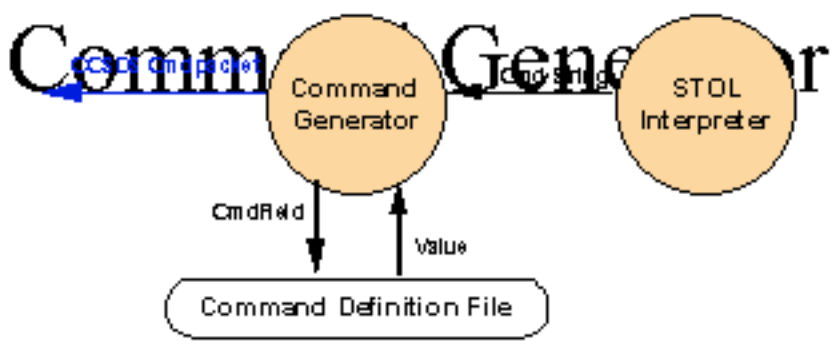
- Identifiers
  - Raw/Engineering variable
  - Statement Labels (e.g. STEP1:)
  - Integers (e.g. I:vvv, X:vvv)
  - Floating point numbers
  - Strings (e.g. "...")
- Local variables and simple math
  - LOCAL count
  - LET count=0
  - count = count + 0.5
- Operators (EQ, NE, LE, LT, GE, GT,

STOL Directives (continued)

- Looping
  - GOTO StatementLabel
- Commanding (/CMD and /SEND)
  - Commands to GUVI instrument
  - /CMDs are buffered until a /SEND
- WAIT [X] (X is in seconds, optional)
- START X (X is another STOL procedure name)
- ASK "prompt" localVariable

zDirectives

- Data Acquisition
  - .zDataAcquisition ON|OFF
  - .zSaveData ON|OFF
  - .zMDC Playback|RealTime yyyydoyhhmmss
  - .zIncrement_Data_File
- Commanding
  - Supplemental header info
    - .zCmdDeltaTimeOut xSeconds
    - .zCmdAbsTimeOut yyyydoyhhmmss
  - H/W GSE commands TBD (.zGUVI ...)
- Save window



- Receive command string from STOL interpreter and translate to binary
  - **Discrete command**
    - Generate binary bytes given Command Definition file
    - Variable data field allowed (STOL generated)
  - **Upload command - segment binary file (max 4K) into 256 byte blocks**
- Add CCSDS header, supplementary information (authentication, time out, command

Command Definition File

- Tab delimited text file
- Each row defines a command or a command field

- Column headers
  - Command Name
  - Description (C format string for command description block)
  - Hazardous (Y/N)

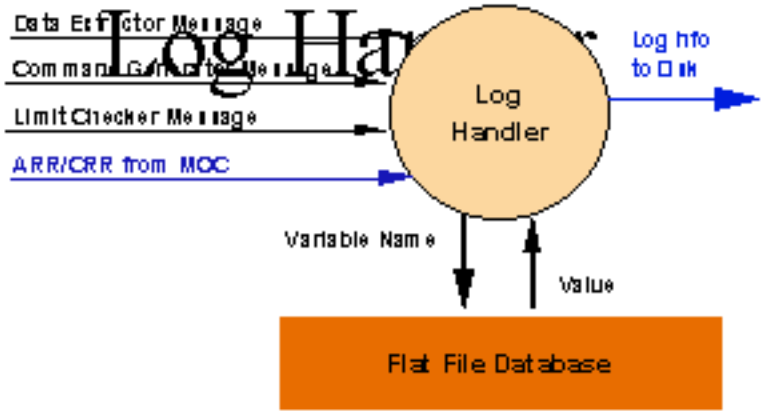
Sample STOL Procedures

Telemetry Processing

```
LOCAL nImage, n
ASK "Number of images to collect"
nImage
; Process realtime data
.sMDC realtime
.sDataAcquistion on
; Throw away first partial frame
WAIT UNTIL (subpacket .EQ. 57)

; Start collecting nImage frames
.sClearVariable imageSum
LET n = 0
LoopImage:
   IF (n .GE. nImage) THEN
      GOTO Done
   ENDIF
   WAIT UNTIL (subpacket .EQ. 57)
   n = n + 1
   GOTO LoopImage
Done:
   .sDataAcquisition off
   .sSaveImageBinary "call.dat" imageSum
```

Commanding

```
LOCAL sPos


ASK "Input scan mirror position" sPos


; Send command to set scan mirror pos
.sCmdDeltaTimeOut 30
/CMD SCAN_MOTOR_POSITION I:sPos
/SEND
```



- Receive and log the following:
  - GUVI verified command message from Data Extractor
  - GUVI/Spacecraft ECU error messages from Data Extractor
  - Misc. messages from Data Extractor
  - Command sent message from Command Generator

Execution Sequence

- Load database definition files
- Sync to UT
- Initialize TCP/IP interrupt routine
- Establish MDC connection
- Establish H/W GSE connection
- Execute startup STOL procedure (optional)
- Initialize to previous settings (window locations, data acquisition parameters, etc.)

TCP/IP Interrupt Routine

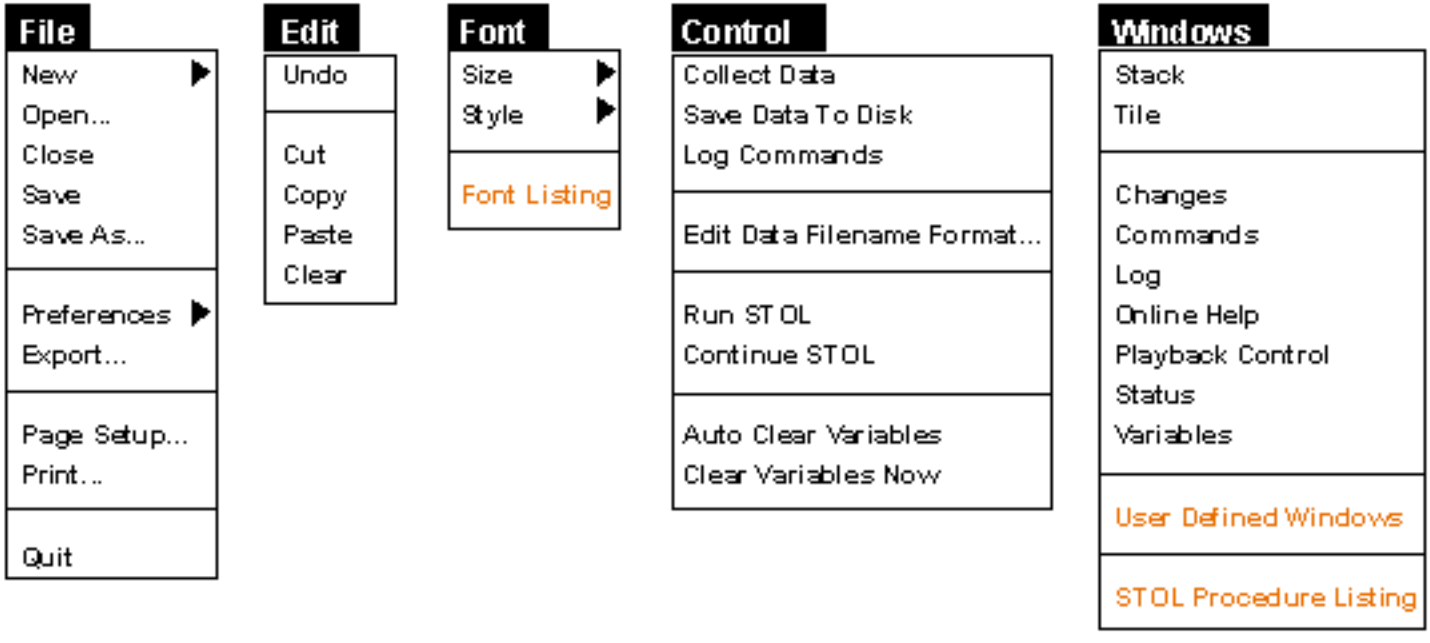
- Process event
  - Remote port closing, remote time-out, no connection, process error
  - Internet control message arrived, process error
  - Urgent data outstanding, process error
  - Data arrival

- Data arrival
  - Check semaphores, if locked return
  - If circular buffer full, sender will have to wait, return

Main Event Loop

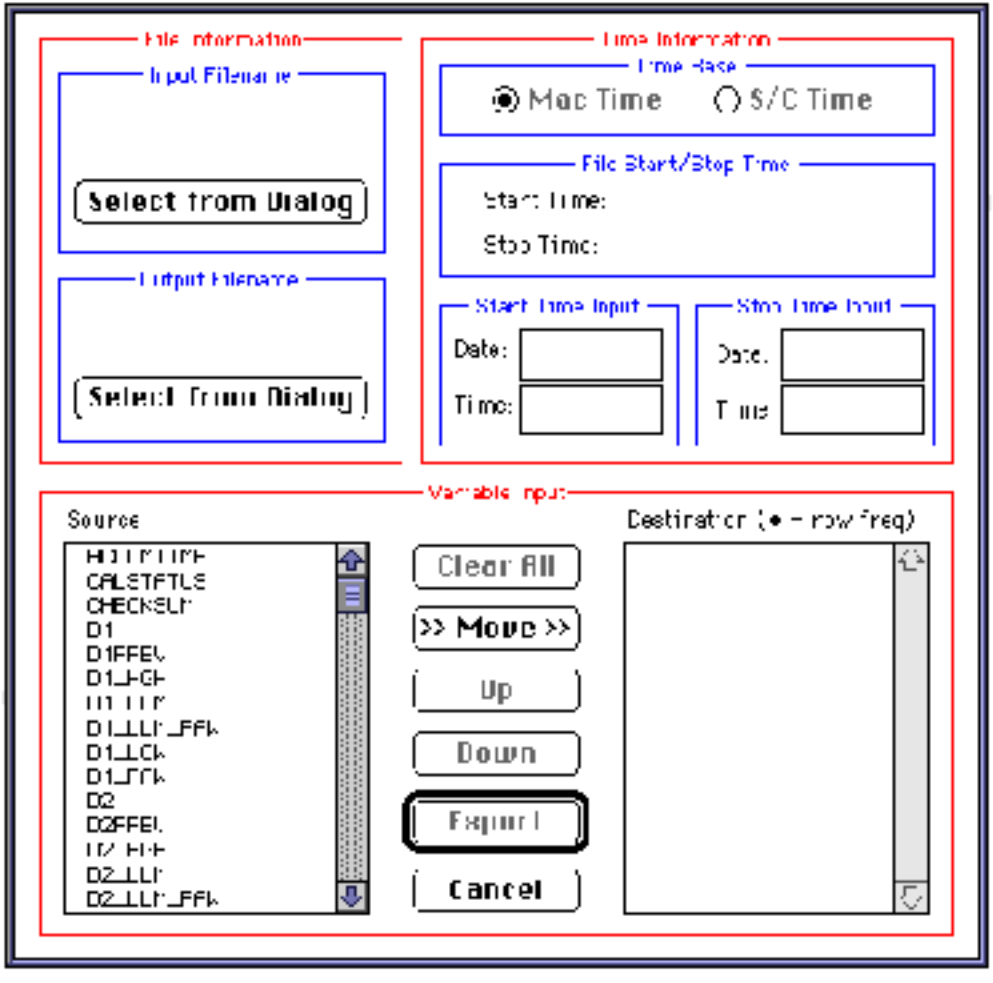
- Read packet from TCP/IP buffer (enforce semaphores)
- Validate packet
- Process packet
  - Extract raw variables (log various telemetry items)
  - Save packet to disk
  - Calculate engineering variables
  - Check limits, set color (red, yellow, green), generate page
  - Update windows

User Interface

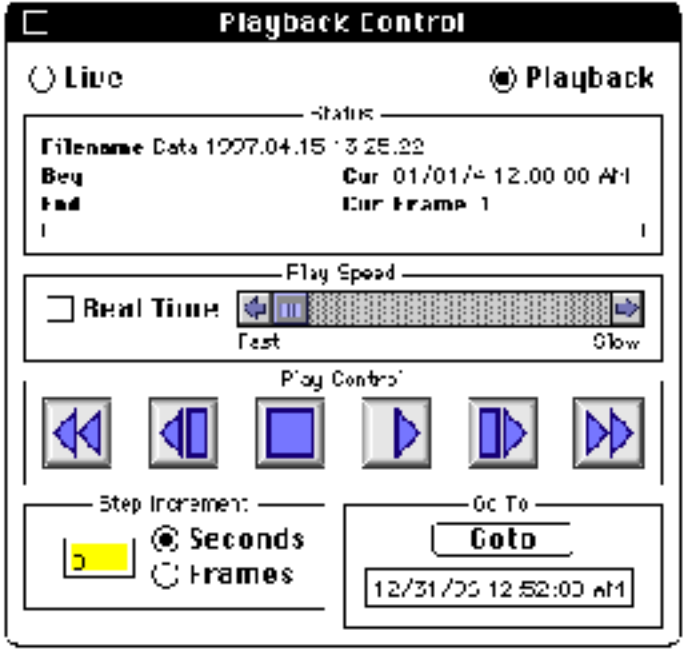


New: Data | Log | STOL
Preferences: Mission Ops | TCP/IP Settings | Startup STOL | Shutdown STOL

Export Window



Playback Control Window



Software Reuse

- Heritage of existing code
  - Flight: CPT, DSU, LENA, ICO
  - Ground based: Aurora

- Modules Reused
  - Packet Identifier 50%
  - Data Extractor 100%
  - Data Converter 100%
  - Limit Checker 100%
  - Log Handler 100%
  - Display Handler 100%

Software Platform

- Language - C, in-line assembly
- Compiler - Code Warrior 11
- Operating System - Mac OS 7.6 or greater

- Documentation
  - Software Requirements Specification APL
  - Software Development Plan AERO
  - Software Test Plan AERO
    - Requires TIMED spacecraft emulator

Hardware Platform

- Platform (development and testing/target)
  - PowerMac 6500 (300 MHz)
  - 128 Mbytes RAM
  - 4 GBytes local hard disk
- Additional hardware
  - 28.8 kbps modem
  - 100 Mbit FAST ethernet
  - Power Key Pro (external WatchDog timer)

Target Computer Loading

- Storage Requirements (24 hour)

	Pre-Delivery	Post-Delivery
– Data file	87.6 MBytes	0.00 MBytes
– Log file (5 min. interval) 0.04 MBytes		0.04 MBytes
– Housekeeping file	0.43 MBytes	0.05 MBytes
– Trend file	0.43 MBytes	0.05 MBytes
Total	88.5 MBytes	0.14 MBytes

Basic Operating Assumptions

- Power
- Active Telephone Line
- Active Ethernet Connection
- MOC and MDC functioning
- Data Processing POC functioning

Design Issues

- FTP commands to MOC
  - FETCH (external application) to do FTP, AppleEvents to handshake between Engineering POC software and FETCH
  - Embed FTP within Engineering POC software