Space Shuttle Usage of z/OS

Did you know that the software running on the Shuttle's main computers is developed, built and tested on z/OS? Come to this session to learn about the languages, preprocessors, and compilers (custom and COTS) used to generate and test the Shuttle's Primary Avionics Software System (PASS). It will also show the hardware and unique access methods used to attach the Shuttle's onboard computers to the z900 to create a simulation environment for development and testing.

Space Shuttle Usage of z/OS

Jan Green

United Space Alliance, LLC

Jan.Green@usa-spaceops.com

March 5, 2009

Session 2819

USA™
United Space Alliance

Contents

Our Mission

United Space Alliance

Flight Software

To be the world-class software leader for human space flight providing software that is safe, on-time, error free, and cost-effective

Products and Customers

Ground Flight Control
MCC / KSC

Crew & Flight
Control
Training

Shuttle
Missions

Integrated HW/SW
Testing

Mission Support

Products

Flight / Onboard Products

Ground Products

Applications/ Support Tools

PASS and BFS
GPC Flight Software

Flight
Reconfiguration

Mission Specific
Data Products

Onboard &
Ground Products

C/C++
PL/1
IMS
Oracle
SQL

Mainframe

LAN

JAVA
HTML
HAL/S
Fortran
Assembler

Computing Platforms

Human-Rated
Software Validation

Applications / Support
Development

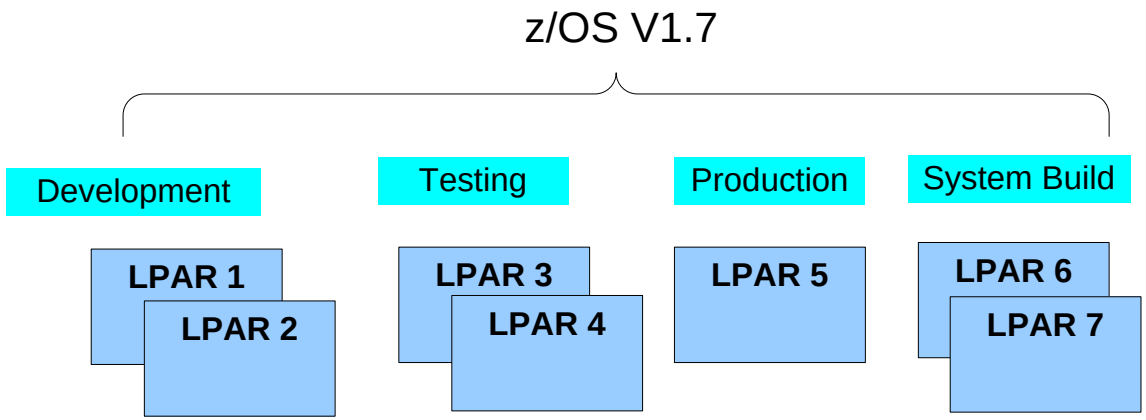
Modeling &
Simulation

Systems Engineering

Application Development

Facility Overview

z/900
Model 2064-102
78 MSUs (capped at 65 MSUs)
2 CPs, 1 IFL
16 Gig Memory
3.6 Terabytes DASD



Facility Overview

Major Products:

- JES3 (each LPAR is a standalone Global)
- IMS (vehicle and payload configuration information)
- Compilers (PL/I for MVS&VM, C/C++ V1.7, Assembler, VS FORTRAN 2.6, REXX V1.4)
- RACF
- OEM
  - Oracle (configuration management of all source changes)
  - CA Suite (Ops, TMS, etc)
  - EJES
  - Astute
  - Abendaid / Fileaid

Shuttle Data Processing System

Shuttle Data Processing System

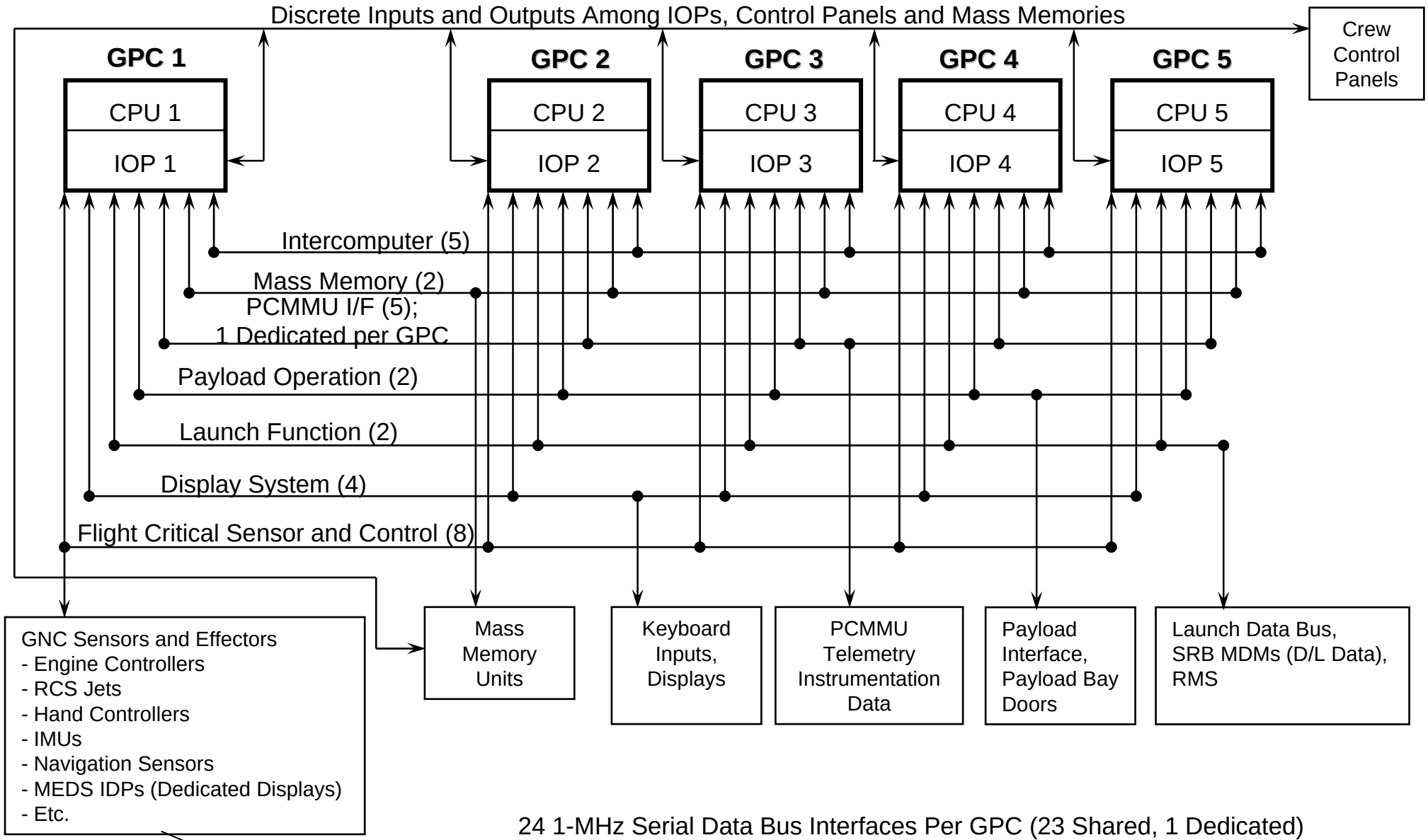
- AP-101S General Purpose Computers (GPC)
  - Radiation tolerant processor for real-time applications
  - Architecture evolved from the IBM System/360
  - Same processor family used in several military aircraft (B-1B,…)
  - Approximately 1.3 MIPS
  - CPU Oscillator 40 MHz
  - Memory – Static RAM CMOS technology
    - 262,144 FULLWORDS (32 BITS) = 1 MB
    - Access/Cycle Time = 250 nanoseconds

- Mass Memory
  - Solid State Memory contains several copies of executables
  - Avionics overlay software for each flight phase is loaded from here

Shuttle Data Processing System

- Serial Data Buses
  - 24 Data Buses - 1 MHZ Serial Channels
  - Unique bus using architecture that evolved into MIL-STD-1553
  - Any bus has a single GPC commander, all GPCs can listen on most buses
- Multiplexers/Demultiplexers (MDM)
  - Provide serial, analog, or discrete interface between end devices and serial data bus
- Multifunction Electronic Display Subsystem (MEDS)
  - Crew GPC display and keyboard interface
- Completely fly-by-wire system
  - Crew flight controls all electrically connected via GPCs
  - No direct mechanical or hydraulic linkages (except landing gear)

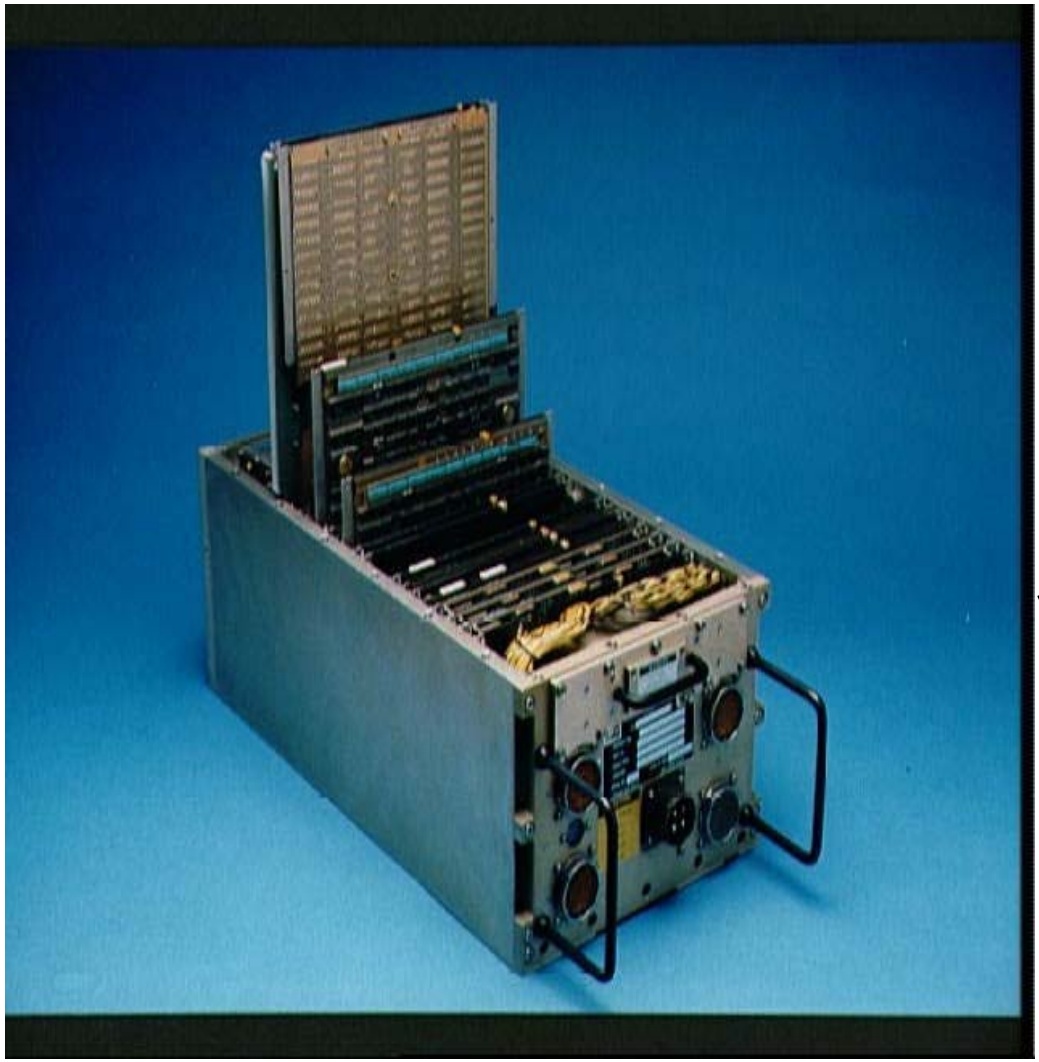
Shuttle Data Processing System



United Space Alliance

Cockpit View

Shuttle's General Purpose Computer (GPC)



The AP-101S GPC is 19.55 inches long, 7.62 inches high, 10.2 inches wide, and weighs 64 pounds.

It requires 550 watts of electrical power.

Shuttle GPC Flight Software System

The GPCs host the Primary Avionics Software System (PASS) and Backup Flight Software (BFS)

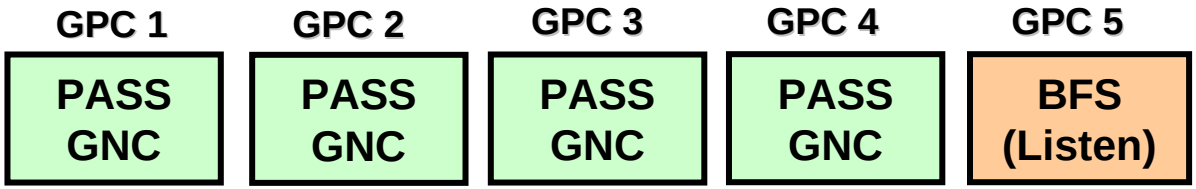
- **PASS is a full-function software system controlling all mission phases**
  - **Ascent, On-Orbit, Entry/Landing, and Ground Checkout overlays for Guidance, Navigation, and Control (GNC)**
  - **Separate Systems Management (SM) Overlay**
    - **Antenna management, power and life support functions, payload control, etc.**
  - **Over 450 distinct functional applications**
  - **Approx 400K SLOCs**
- **BFS is an independently developed and maintained software system which provides redundancy during critical ascent and entry phases.**
  - **Will be engaged in the event of a PASS redundant set failure**
  - **Provides Systems Management/Special Processing and Payload Support Capability which are not available in PASS during Ascent and Entry**

Shuttle GPC Flight Software System

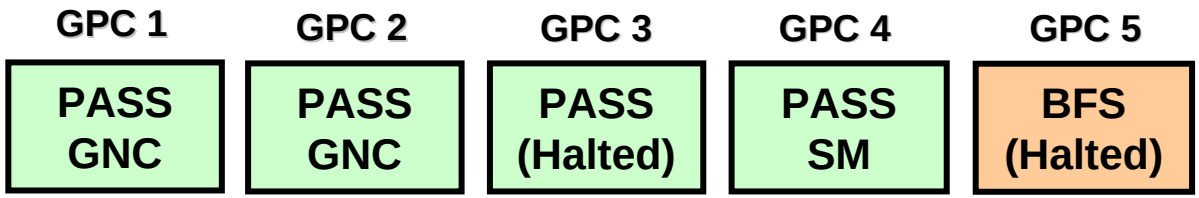
- Custom operating systems
  - Real-time, multitasking, interrupt-driven system with pre-emptive priority scheduling
- PASS and BFS are primarily coded in HAL/S
  - All application software and a major part of the system software
- Core OS functions are AP101 assembler code

Shuttle GPC Flight Software System

- Ascent/Entry Configuration

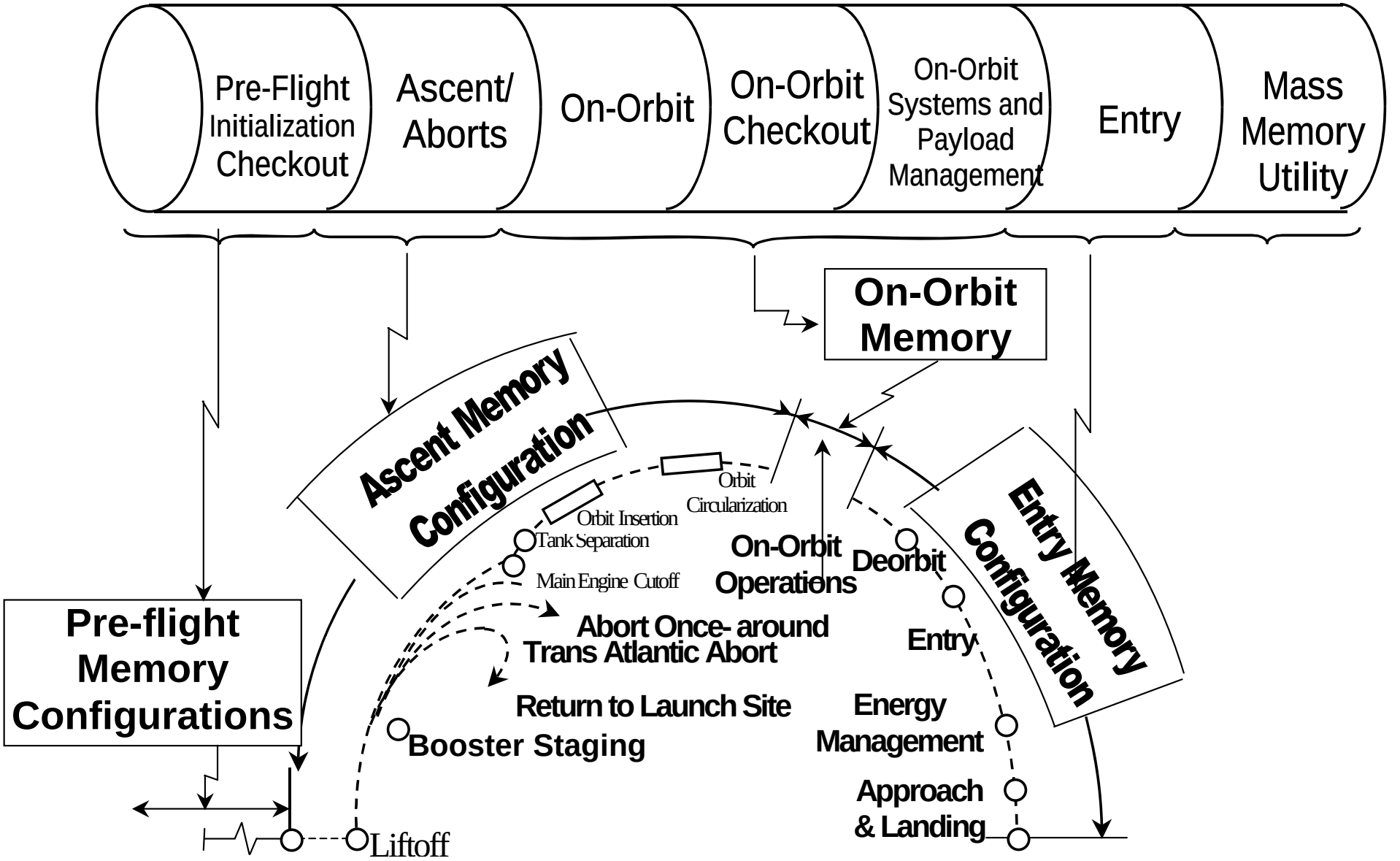


  - PASS controls the Flight Critical buses (flies the vehicle)
    - Synchronized redundant set execution of identical software
    - Two FC buses controlled by each PASS GPC
    - All PASS GPCs listen to all FC buses
    - OS maintains sync, ensures all redundant GPCs receive same inputs
  - Backup Flight Software (BFS) listens on FC buses
- Typical Orbit Configuration



  - FC buses divided between PASS GNC GPCs
  - Halted PASS GPC is first backup (orbit configuration loaded)

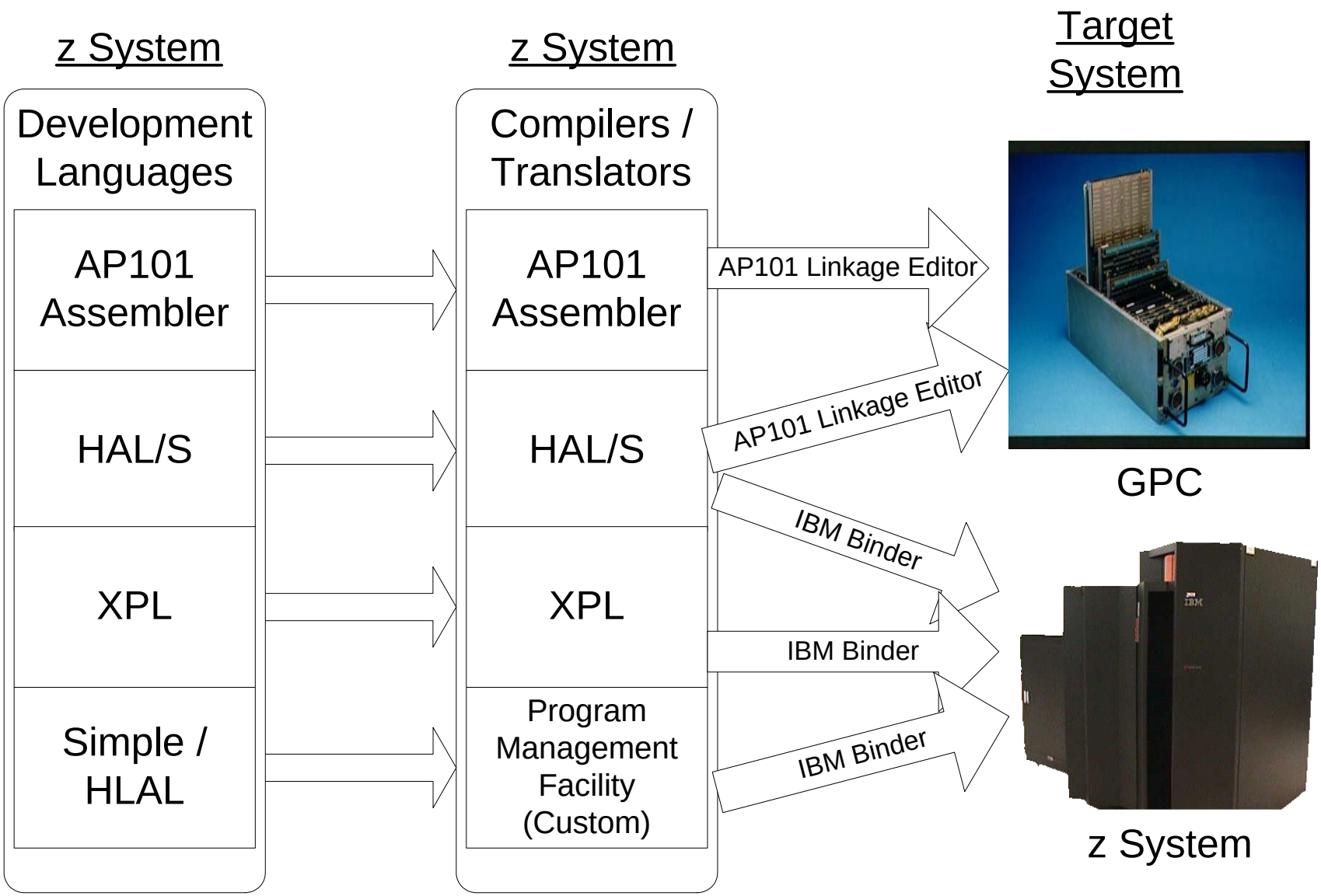
Phases of PASS



United Space Alliance

Languages and Compilers

Custom Language and Compiler Overview



United Space Alliance

HAL/S Language

- The HAL/S language was developed under NASA direction in the early 1970's for aerospace applications
  - Resembles PL/I
  - Designed for reliability and readability
- Major language features
  - Modular, structured programming language
  - Facility to schedule tasks based on priority
  - Vectors and matrices as standard data types
    - Standard vector and matrix operators (dot and cross products, matrix multiply and transpose, etc.) built in
  - Common data pools
  - Compiler-enforced restrictions on problematic language constructs (e.g., GO TO)
    - Different restrictions for flight code and tool code
  - *Does not support dynamic allocation of memory*

HAL/S Sample Code

```
DECLARE S4_SUBSCRIPT INTEGER;
DECLARE ENTRY_NAME CHARACTER(*);
DECLARE INTEGER, SAVE, TEMP;
SAVE = INDEX(ENTRY_NAME,'$');
IF ENTRY_NAME$(SAVE+1) = '(' THEN
DO;
TEMP = SAVE +2;
DO WHILE ENTRY_NAME$(TEMP) >= '0' OR ENTRY_NAME$(TEMP) <= '9';
 TEMP = TEMP + 1;
END;
ENTRY_NAME = ENTRY_NAME$(1 TO SAVE+1) ||
             CHARACTER(S4_SUBSCRIPT) ||
             ENTRY_NAME$(TEMP TO #);
END;
ELSE
ENTRY_NAME = ENTRY_NAME$(1 TO SAVE) ||
             CHARACTER(S4_SUBSCRIPT);
```

HAL/S Compiler

- The HAL/S Compiler system consists of
  - A seven phase language processor which produces object modules compatible with AP-101 Shuttle Support software
  - A comprehensive run-time library which provides an extensive set of mathematical, conversion, and language support routines
  - Can also produce object modules targeted for z/OS

- Additional trivia…
  - Written in XPL, another custom language
  - Objects targeted for the GPC are linked using the AP 101 Linkage Editor, another custom tool
  - Interfaces to Flight OS Based on the IBM System/4 Pi Model AP-101 (aka AP101S)

AP101 Assembler & Linkage Editor

- The AP101 Assembler
  - was created in the mid 1970s
  - based on IBM OS/360 Assembler
  - is written in IBM 390 Assembler
  - runs under z/OS

- The AP101 Linkage Editor
  - Provides the capability of combining and resolving external references of separately assembled program modules to produce an executable load module for the AP-101s Computer
  - created in the mid 1970s
  - is written in IBM 390 Assembler

  ➢ Creates the overlays used during different flight phases

XPL Language

- Developed at the Stanford Computation Center, the General Motors Research Laboratory, and at the University of California at Santa Cruz
- Derived from PL/I
- Some differences between XPL and the PL/I constructs with the same form:
  - Declaration is mandatory
  - Arrays are restricted to one dimension and the lower bound of all arrays is implicitly zero
  - There are no predefined abbreviations (use character, not char)
  - Only types fixed, character, and bit are provided
  - Bit strings are substantially different
  - Do loops have only positive steps
  - Procedures are not recursive and have only value (evaluated) parameters

XPL Sample Code

```
/* SUBROUTINE TO FIND OCCURRENCE OF ONE CHAR STRING IN ANOTHER. */
CHAR_INDEX:
  PROCEDURE (STRING1, STRING2) FIXED;
DECLARE
  STRING1    CHARACTER,
  STRING2    CHARACTER,
  L1         FIXED,
  L2         FIXED,
  I          FIXED,
  POS        FIXED,
  FOUND      BIT(1);
L1    = LENGTH (STRING1);
L2    = LENGTH (STRING2);
POS   = -1;
FOUND = FALSE;
I     = 0;
IF (L2 <= L1) THEN DO;
  DO WHILE (I <= (L1 - L2)) & (¬ FOUND);
    IF (SUBSTR (STRING1, I, L2) = STRING2) THEN DO;
      POS   = I;
      FOUND = TRUE;
    END;
    I = I + 1;
  END;
END;
RETURN POS;
```

XPL Compiler

- Written in XPL
- Can produce IBM 390 compatible object modules
  - IBM Binder used for linking

SIMPLE / HLAL

- Z/OS Assembler Applications written in our Houston's High Level Assembly Language (HLAL)

- SIMPLE is a pre-processor, producing IBM Assembler instructions from HLAL statements
  - Also produces a "pretty print" formatted listing with block indention and numbering

Tools for Building, Reconfiguring, and Testing Flight Software

Application Tools

There are over 100 application tools on z/OS that support the development, reconfiguration, build, and test of the Flight Software.

- ❖ These tools directly affect the content of the shuttle mass memory or are used to analyze the content and correctness of the flight software.

These applications include:

- Preprocessors
  - Create data tables that are part of the Flight Software

- Flight Software Build and Reconfiguration Tools
  - Recreates data tables based on unique flight characteristics
  - Puts together components of the FSW
  - Creates the Mass Memory image which is installed on the shuttle
  - Mass Memory compare utilities

Application Tools

- Verification Autoscorers
  - Automated verification of FSW data components
- Configuration Management
  - Integral to entire process
- Simulator and Models
- Test Tools
  - Simulation post processors, data reduction, plotting
- C++ Reusable Components
  - String handling, Dynamic Dataset Allocation I/F, Containers, Reports, etc

Shuttle Flight Software Simulator

Flight Software Simulator

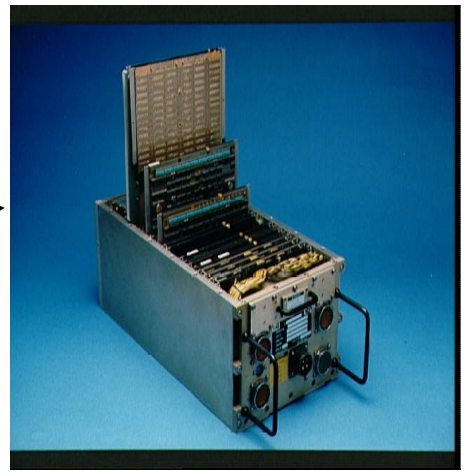
- **Provides a full Shuttle simulation capability with real Flight Software (FSW) and real Shuttle General Purpose Computers (GPCs) running in a simulated environment**
  - **All external devices and buses are modeled within the simulator**
- **Used by FSW developers and verifiers to test the Flight Software before it is delivered to the end users/customers**
- **Used during flights if required to build/test a patch to the Flight Software**
- **Runs under z/OS**

Simulator Component Configuration

Z/900

FEID

GPCs



Mainframe

- Models cyclic processing and synchronization with GPC FSW
- Performs command processing and sequencing
- Provides user controlled diagnostic data collection and logging

Flight Equipment Interface Device

- Custom hardware and software
- Multitasking high data throughput
- Collects diagnostic data from the GPC
- Passes model output to the GPC
- Passes GPC output to the models

General Purpose Computer

- Flight-equivalent hardware executing the Shuttle Flight Software
- I/O goes over a MIA adapter to interface with the FEID instead of a MIA to a shuttle device.

Flight Equipment Interface Device (FEID)

- Utilizes the main controller from the System/7 computer
- 360/50 frame

- Connection to z900 recently converted from Bus and Tag (Parallel) to FICON
  - The parallel channel FEIDs were defined as CTC on a 3088 control unit
  - The FICON FEIDs are defined as FCTC
    - Programming on Virtex 2 Pro chip accomplished using Very High Density Logic (VHDL)

- Attaches to the GPC via the Avionics Ground Equipment (AGE) Interface

- Responds to GPC bus requests for data

- Manages GPC Start/Stop Synchronization

- Monitors effective address registers to know what instructions are executing and what data is being accessed

Simulator Software Overview

- Primarily Assembler Application
  - Cannot run under Language Environment (LE) due to extensive internal error handling and unique architecture
- Runs as a batch job
  - Not technically real-time, but it appears that way to the system under test
  - Multi-tasking (about 10 major tasks and many other minor tasks)
- Unique access method to communicate with FEID
  - EXCP level
  - Attention interrupt handler and I/O appendages
  - One dedicated channel for input from FEID
    - Never ending read channel program (under normal conditions)
    - PCI Interrupts used to tell access method where the channel program is
  - Another dedicated channel for both input and output
    - Attention Interrupts used to signal error conditions

Unique Characteristics

- The simulator can stop and start the FSW execution such that the FSW has no noticeable perturbations to its functionality or timing.
  - Controlled at the hardware level by the FEID.
    - GPC hardware allows complete stopping at any instruction
  - Supports robust diagnostic capabilities
    - Can stop the FSW and extract/patch registers and data based on instruction execution, memory reference, and I/O execution
    - Can extract/patch register and data information during trace and single step
    - Can apply multiple I/O faults including killing the communication on the bus entirely

Unique Characteristics

- The math models of the simulator comprise the full set of environmental, Guidance Navigation and Control (GNC), and interface models needed to make the FSW think it is flying a Shuttle.
  - The simulator does non-time-critical calculations while the FSW is running
  - When the FSW finishes its minor cycle, the simulator does its time critical calculations.
  - The simulator can stop the FSW to catch up.

Unique Characteristics

- The simulator user can create checkpoints of the simulation and can then restart the simulation at a later time.
  - Copies data from GPC, FEID, and simulator application to tape, including state vectors, active commands, registers, etc. (Checkpoint)
  - Used later as the starting point of other simulations (Restart)
    - Allows any user to resume a simulation at a later phase and optionally modify the commands for additional testing
    - Allows simulations of various parts of a flight without having to go through launch every time
  - The run started by a checkpoint will have identical results as the original job at flight times after the checkpoint.

Simulator User Interface

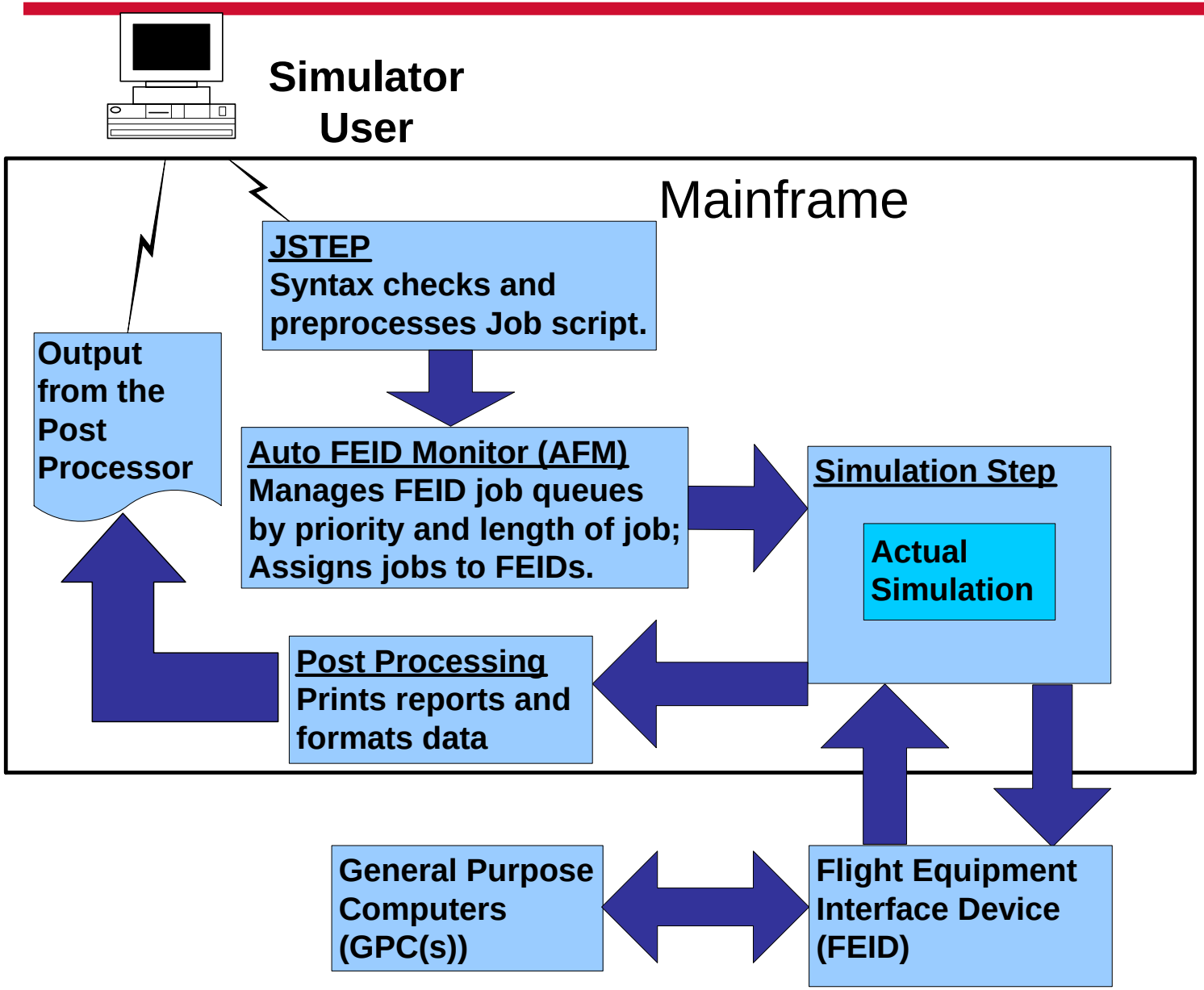
User Interface is called the Simulator Control Language (SCL)

- **Primarily written in Assembler; Some PL/I**
- **The SCL is a high level testing language that provides for symbolic substitution and conditional branching.**
  - **Batch command language, breakpoints, data collection/manipulation, checkpoints, etc**
- **SCL is syntax checked and converted into data tables and files**
  - **Stored in a partitioned dataset for later use by the simulation job.**

Post Processing

- Primarily written in Fortran and PL/I
- All of the snapped and I/O logged data is placed on the log tape.
- This log tape is available for any user to post process and see the results.
- The post processing toolset allows the user to see data in its raw format (hex), its engineering units, or be plotted.
- The users may write their own post processing routine and link it with the system version allowing them to manipulate the output data into any desired format.

Job Execution



Simulation job steps designed for efficient thru-put

- JSTEP – offline compilation of test instructions
- AFM – totally automated job scheduling and monitoring based on priority
- Simulation – simulation utilizing the FEID
- Post Processing – offline data analysis and reduction

Software Development and Build Tools

Tool Languages and Compilers

Language	Compiler	% Total SLOCS
Assembler	High Level Assembler V1.5	48
C++	C/C++ V1.7	15
PL/I	PL/I for MVS and VM 1.1	9
XPL[1]	XPL[1]	9
ISPF Dialog Services	ISPF V5.7	7
C	C/C++ V1.7	5
HAL/S[2]	HAL/S[2]	3
FORTRAN	VS FORTRAN V2.6	1
REXX	REXX Compiler V1.4 (some interpreted only)	1
SQL	Oracle	1

[1] Language and Compiler maintained in Houston
[2] Language and Compiler developed in Cambridge and maintained in Houston

Builds

- The builds for both FSW and critical tools are carefully controlled.
  - The changed source code is inspected in rigorous review processes and submitted to the build process.
  - The build process checks each individual module being changed for authorizing Change Requests
    - Configuration Management data is in an Oracle database
  - The build process uses the submitted source code to build the executable software
  - Provides unwavering reliability that the source code reviewed is accurately represented by the executable code built.

Program Management Facility (PMF)

- Unique software that controls assembly, compilations, and link-edits of all PASS FSW or critical tool software in a single batch job and step
- All modules defined in a “System Inventory” that directs PMF on which translators / linkage editors / options to use and the expected condition codes
  - System Inventory is maintained in an Oracle database
- Invokes a Source Dependency Scanner, when applicable, for use in determining the order of module processing during a software build and any modules that need to be recompiled due to inverted cross references
- Invokes preprocessors such as Simple, when applicable
- Invokes (Attaches) the appropriate translator(s)
- Invokes (Attaches) the appropriate linkage editor(s)
- Returns CC 8 if expected condition codes not received

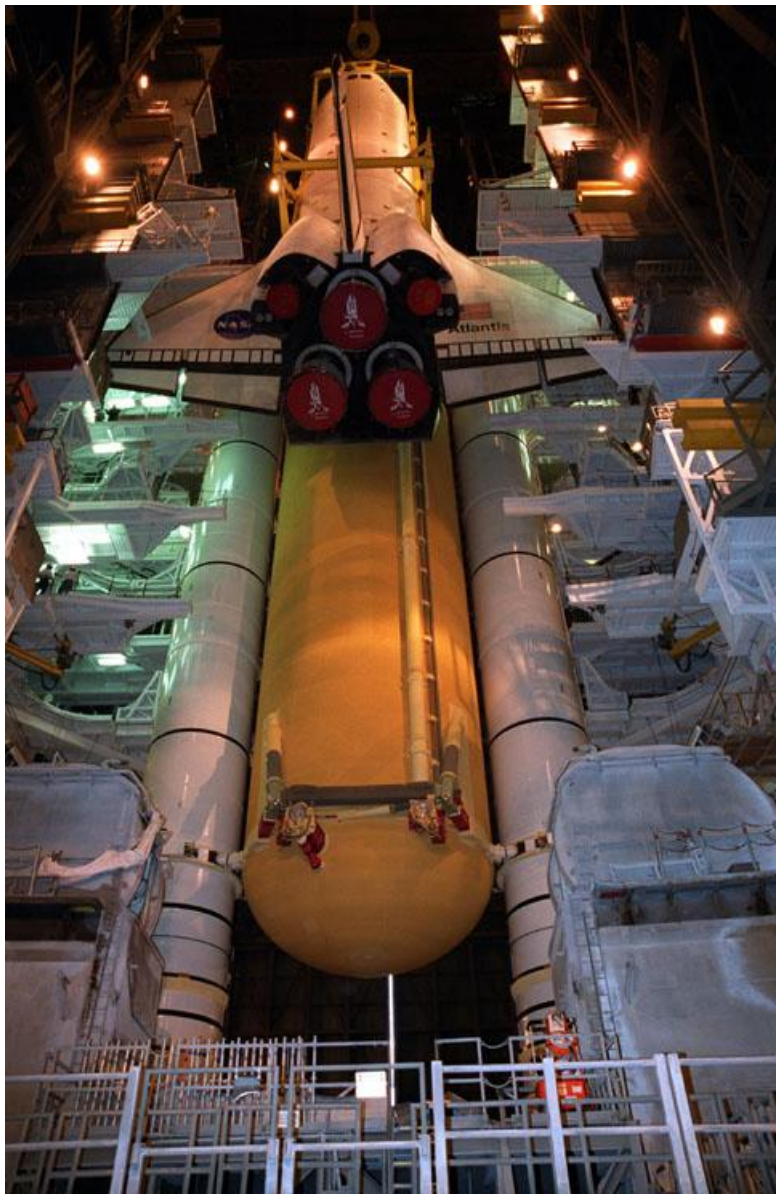
Test of Time

- A majority of the tools developed have existed since the 1970s

- COTS software used by our critical application tools are at supported levels
  - Establishment and enforcement of standards ensures standard interfaces are used
  - Z/OS software upgrades are carefully planned and tested
  - All application software is recompiled when major upgrades are made to the compilers

Fun Facts and Acronyms

SHUTTLE STACK (ORB / ET / SRB / MLP)



SHUTTLE FUN FACTS

- It takes only about 8 minutes to accelerate to a speed of more than 17K MPH.
- Each Shuttle Main Engine weighs about 1 / 7th as much as a train engine but delivers as much horsepower as 39 locomotives.
- The turbopump of the Shuttle Main Engine is so powerful it could drain an average family-size swimming pool in 25 seconds.
- The LH2 in the main engine is -423 degrees F and when burned with LO2, the temperature in the combustion chamber reaches +6000 degrees F.
- The energy released by the 3 Main Engines is equivalent to the output of 23 Hoover dams.
- Each of the SRBs burns 5 tons of propellant per second; a total of 1.1+ million lbs in 2 minutes. The plume of flame ranges up to 500 ft. long.
- The temperature of the combustion gases in the SRB reach +6100 degrees F; two-thirds the temperature of the surface of the sun.
- The four engines of a Boeing 747 produce 188,000 lbs of thrust; the two SRBs are more powerful than 35 jumbo jets at takeoff.
- If their heat energy could be converted to electric power, two SRBs firing for two minutes would produce 2.2 million kilowatt hours of power, enough to supply the entire power demand of 87,000 homes for a full day.

Acronyms

Acronym	Meaning
AGE	Avionics Ground Equipment
BFS	Backup Flight Software
CTC	Channel to Channel
ET	External Tank
FC	Flight Computer
FEID	Flight Equipment Interface Device
GNC	Guidance, Navigation, and Control
GPC	General Purpose Computer
GPS	Global Positioning System
HAL/S	The real-time aerospace programming language for Shuttle
HLAL	High Level Assembly Language
IDP	Integrated Display Processor
IMU	Inertial Measurement Unit
IOP	I/O Processor

Acronyms

Acronym	Meaning
MDM	Multiplexer/Demultiplexer
MEDS	Multifunction Electronic Display Subsystem
MIA	Multiplexer Interface Adapter
MLP	Mobile Launch Platform
OMS	Orbital Maneuvering System
ORB	Orbiter
PASS	Primary Avionics Software System
PCMMU	Pulse Code Modulation Master Unit
PMF	Program Management Facility
RCS	Reaction Control System
SIMPLE	Structuring Interpreter for a Macro Processing Language Extension
SLOC	Source Lines Of Code
SM	Systems Management
SRB	Solid Rocket Booster