

# Portable Checkout System

## Technical Paper

---

**August 25, 1999**

**Rodney D. Davis**  
Chief Technology Officer  
Command and Control Technologies Corporation  
1311 N. U.S. Highway 1, Suite 129  
Titusville, FL 32796  
407.264.1193  
[www.cctcorp.com](http://www.cctcorp.com)



## Proprietary Information Statement

All data contained in this document is proprietary and is the property of Command and Control Technologies Corporation of Titusville, Florida. It is furnished and received in confidence solely for informational purposes of the recipient. Permission to reproduce this document is hereby granted provided that this page and the copyright notice on page ii are included in full.



# TABLE OF CONTENTS

<b>1.0</b>	<b>ABSTRACT .....</b>	<b>1</b>
<b>2.0</b>	<b>INTRODUCTION .....</b>	<b>1</b>
<b>3.0</b>	<b>CONCEPT OF OPERATIONS OVERVIEW .....</b>	<b>2</b>
	<b>SYSTEM ARCHITECTURE.....</b>	<b>5</b>
4.1	REAL-TIME CONTROL SERVER.....	6
4.2	COMMAND AND CONTROL TOOLKIT™ .....	7
4.2.1	<i>Application Development Environment.....</i>	7
4.2.2	<i>Scripting Environment .....</i>	8
4.2.3	<i>System Management.....</i>	8
4.2.4	<i>Archival and Retrieval .....</i>	9
4.2.5	<i>Simulation, Replay, Training and Practice Facilities .....</i>	9
4.2.6	<i>Interface Processing.....</i>	11
4.2.7	<i>Event Processing.....</i>	11
4.3	GRAPHICAL USER INTERFACE .....	11
4.3.1	<i>Glg Toolkit™ .....</i>	12
4.3.2	<i>OMNI™ .....</i>	12
4.4	REAL-TIME CONTROL APPLICATION SOFTWARE .....	13
4.4.1	<i>T-Zero™ Automated Test Conductor.....</i>	13
4.4.2	<i>Countdown Management.....</i>	15
4.4.3	<i>Message Mechanism .....</i>	15
4.5	REAL-TIME WEB SERVER .....	16
<b>5.0</b>	<b>CONCLUSION.....</b>	<b>17</b>



designed to enable routine mobile mission processing by integrating all phases of vehicle development and operations under a common set of tools and processes. The approach to the PCS is based on an integrated suite of system capabilities, including real-time command and control, decision support, mission design, and post mission data reduction and analysis.

In the early 1980's, soon after completing the Shuttle Launch Processing System, NASA recognized that a pattern of common requirements exists across diverse test and checkout system implementations. Motivated by the need to reduce time and costs in new systems deployment, and enabled in large part by a maturing market in open system technologies, NASA set out to address this problem by initiating a research and development project called the Generic Checkout System (GCS).

GCS successfully demonstrated that modest real-time command and control performance was achievable in an open architecture, but more importantly, it demonstrated the potential for COTS to displace custom development and increase capability. GCS was a fully distributed architecture based on the UNIX operating system and ISO communications protocols. The open implementation opened the door to an ever-growing selection of COTS capabilities such as low cost parallel processing computers, database management systems, and GUI development tools.

The Partial Payload Checkout Unit (PPCU) was successfully implemented as a production system based on many of the concepts explored by GCS. With an initial price tag of approximately \$10 million, PPCU demonstrated a dramatic reduction in cost to deploy highly complex control system capabilities. Because no COTS product alternative was available, most of PPCU cost was consumed in the infrastructure software needed to manage, process, distribute, archive, retrieve, and display data and commands, and integrate the external orbiter emulation interfaces.

PPCU went on to further prove that the concept of a reusable architecture was viable for other test and checkout systems. Both the Clipper Graham (the Delta Clipper, also known as the DC-X) launch control system and the Space Station Test Checkout and Monitoring System were successfully derived from PPCU/GCS. These applications realized significant cost savings to an even greater degree than the original PPCU since much of the system infrastructure code was reusable.

However, limitations in the distributed communications design of PPCU prevented it from scaling to smaller sets of equipment or for significantly higher throughput applications. To address this limitation, a third generation research and development activity was taken on by NASA to build a highly scalable architecture called the Control and Monitoring Unit (CMU). CMU was engineered for topology transparency and performance efficiency, building on architectural lessons learned from PPCU. Possessing a full suite of command and control capabilities, CMU is scalable from a portable laptop configuration to a highly distributed parallel processing architecture. It can be deployed at a fraction of the cost of a PPCU-based system. In 1997 CMU was licensed from NASA as part of a technology transfer agreement and is now available commercially as part of the Command and Control Toolkit™ (CCTK) product line from Command and Control Technologies Corporation.

The availability of a complete command and control infrastructure such as is available in CCTK not only enables previously unattainable system deployment cost savings, it now makes it possible for organizations like NASA to focus on their core mission rather than tools development.

### **3.0 Concept of Operations Overview**

The operations concept for PCS is based on the notion of minimizing the test and integration campaign with a common set of tools and processes. For example, the PCS is designed to be operated with a minimal set of support staff and reduced flow time. Common activities in each phase of operation take no more than one operator to carry out. A strong configuration management capability allows the system to be rapidly reconfigured between mission operations. A single operator can reconfigure the PCS system from one experiment or spacecraft to another in under an hour.

Mission planning and configuration functions are linked with a relational database architecture. There are no pockets of automation requiring manual intervention for system functions. The information system functions are fully interactive, allowing for a smooth flow of information from mission planning through post-mission analysis. Configuring the PCS to support new vehicles is also driven by relational data base systems. New vehicle data can be automatically ingested from most common data formats.

The system supports most standard computer communications interfaces and the capability to

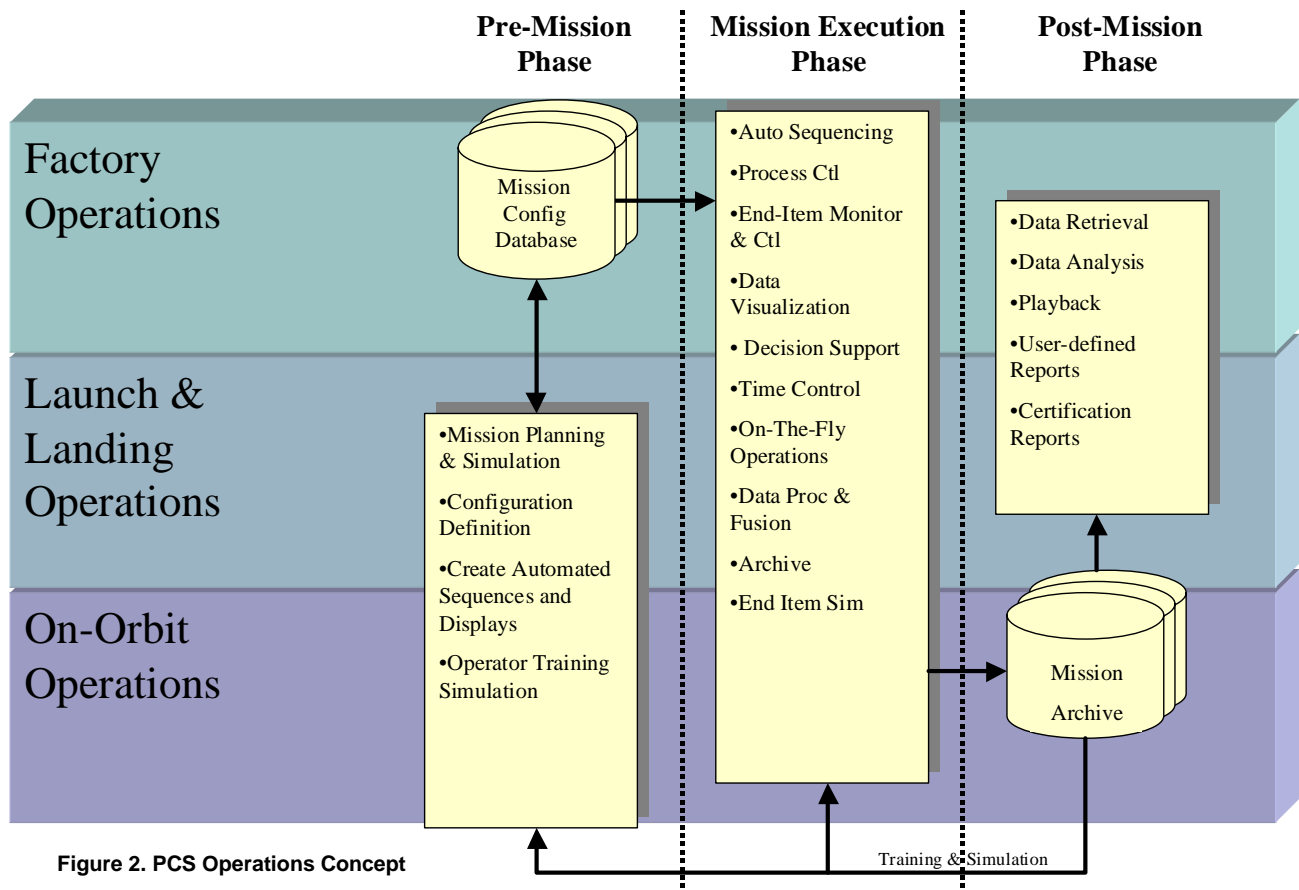


Figure 2. PCS Operations Concept

easily adapt to custom interfaces for non-standard services. This provides the widest possible support for new spacecraft, minimizing the impact of rework to support new hardware interfaces.

Figure 2 depicts major operational phases and relationships between them. The following is a list of key PCS operational concepts:

- High degree of portability using COTS components
- Expedient support of multiple mission scenarios, concurrent test operations, and diverse configurations are managed via open databases and support tools. The PCS requires little or no mission specific software development for mission execution.
- Common software, tools and processes are utilized at all stages of a payload or vehicle's life-cycle, including: mission planning, mission/test configuration, operations, and post mission support. Further, these capabilities can be deployed at the factory, launch/landing site, and during on-orbit operations.
- Automation tools such as task sequencers and real-time script generators are used to minimize labor intensive tasks and efficiently support man-in-the-loop semi-automated operations.

- Internet-based information access allows data to be freely interchanged among PCS operators, users, and remote test support personnel with minimal infrastructure and cost.
- Simulation and training capabilities are built in to the PCS to allow rigorous checkout of systems and continuous development of operation team competence.
- The approach can be expanded to other elements of vehicle and ground systems automation to facilitate future integration of process changes and capabilities. Significant emphasis is placed on open systems technology, interface standards compliance, and integration of plug-in COTS capabilities. New and/or custom external interfaces can be added without altering the underlying architecture.

The general operational concepts described above are captured in a specific process flow derived for STS payloads in Figure 3, PPT Operations Concept Flow. This concept flow is based on the Portable Payload Tester, an advanced technology project sponsored by NASA-KSC. In this flow the entire life cycle of an STS payload is depicted in three major operational phases: Payload factory, launch site,

and on-orbit. This process flow emphasizes the iterative nature of payload operations, starting with mission planning, configuration definition, flowing to real-time operations like test and checkout, and completing with post-mission analysis and data reduction which feeds back to configuration definition. This process iteration assumes a common set of tools and processes that allow a “building-block” approach to defining and implementing automated operational scenarios. Once a common set of tools and processes are established across the life-cycle, dramatic cost savings become possible.

The first phase of factory operations is subsystem test and checkout. In this phase we are dealing with payload LRUs and components in a bench test setting. Since this is the first phase of operations we are dealing in rudimentary tasks such as basic command and data (instrumentation) interfaces, capturing those interfaces in a configuration database. We also seek to develop the test and checkout automation products for the first low level elements, including display components and initial control and monitoring algorithms. Operations at this phase of the payload life cycle are dynamic and need to support on-the-fly changes until LRU designs mature. Iteration at this phase allows the payload

and the supporting automation to mature together.

The second phase of factory operations is subsystem integration. In this phase we build on automation products developed in the previous phase, but with an increased focus on functional capabilities and inter-subsystem and GSE interfaces. We continue to refine the command and data configuration database and display components, reflecting increasing payload complexity. The primary objective at this level is to define the sequence of events, actions and response criteria that will enable the subsystem components to function as an integrated subsystem. Data fusion is also introduced for the first time. Automated sequences are developed, maintained, and preserved in a common test repository/library. When complete, this test sequence provides a level of validation for subsystem functionality. Interface and missing element simulation are integrated in to the test sequences to compensate for unavailable resources. All command and data transactions are digitally recorded. Seamlessly integrated post-test analysis tools enhance problem analysis and automated test documentation production.

The final phase of payload factory operations is integrated payload certification. In this phase we finalize the command and data configuration

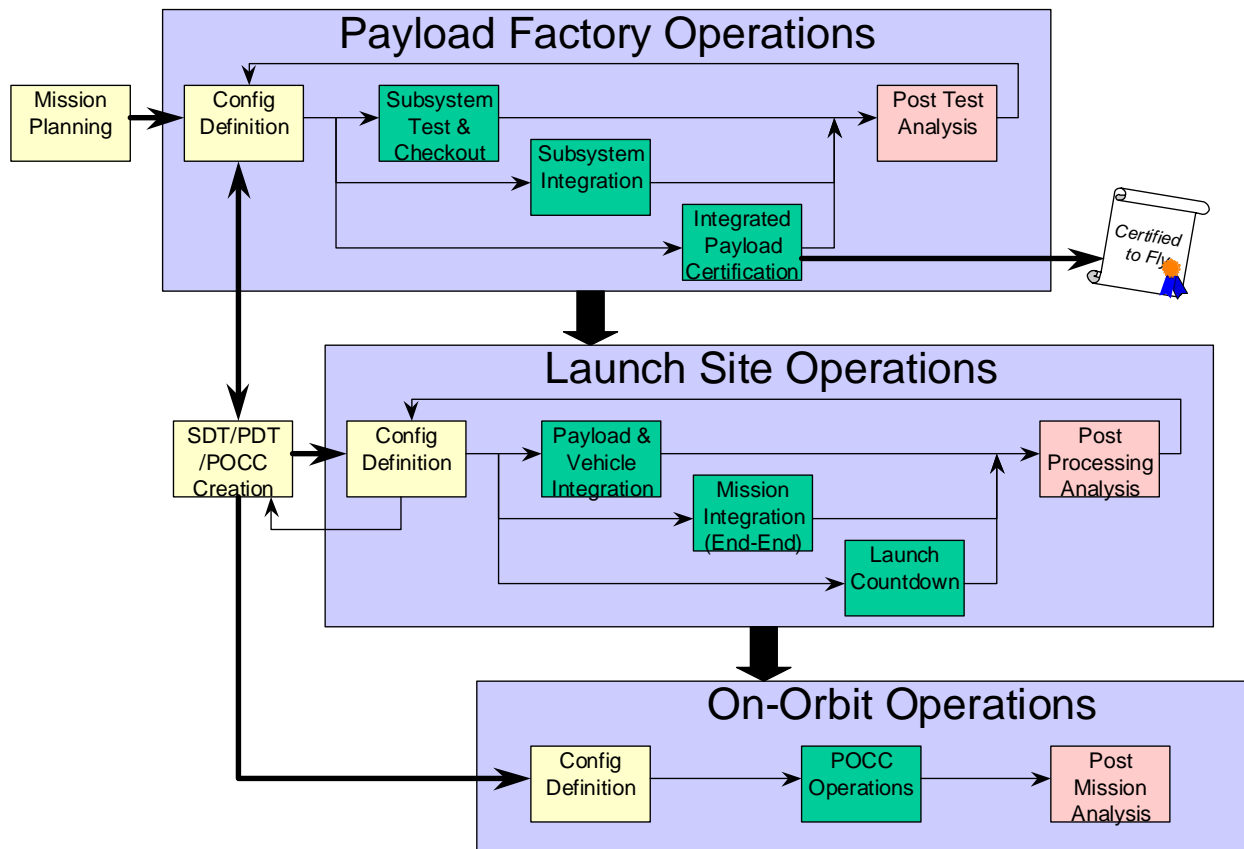


Figure 3, PPT Operations Concept Flow

database and export it into the POCC/PDT Annex. Payload to orbiter external interface verification is executed utilizing certified orbiter emulation interfaces and companion certified interface verification scripts which exercise STS ICD compliance. Certified reports are automatically generated from the test archive. As payload subsystems are integrated into a complete payload system, baselined subsystem models and automated sequences are integrated into mission sequences that are codified and repeatable. Inclusion of web based technology removes the geographic limitations of test and operations support personnel.

The transition of the payload from the factory to the launch site and to on-orbit operations represents a continuous process that is best implemented contiguously from an automation perspective. Operations are designed organically from the automated building blocks of previous phases and missions into increasingly higher levels of capability. Over time, the automation of payload operations becomes a process engineering activity rather than a software/computer technology activity.

#### 4.0 System Architecture

The PCS approach is based on the commercial Command and Control Toolkit™ product and supporting third party COTS products. Known as CCTK, this software package is specifically designed to integrate diverse multi-mission, multi-user applications. It also provides the common core around which application-specific modules and tools are integrated to produce turnkey systems such as the PCS.

The PCS architecture contains the features to perform all operations tasks from test design and preparation to test execution and post-test analysis as described in the previous section. It provides a flexible core capability from which legacy functions can integrate and evolve logically with new PCS capabilities.

Figure 4 shows the PCS top level architecture. The system is comprised of multiple distributed processing subsystems connected together using COTS industry standard interfaces, operating systems, and data visualization tools.

The three-tiered client server system configuration shown in Figure 4 is layered in order to optimize functional and geographical distribution of processing. Layering also inherently increases system reliability and determinism by grouping

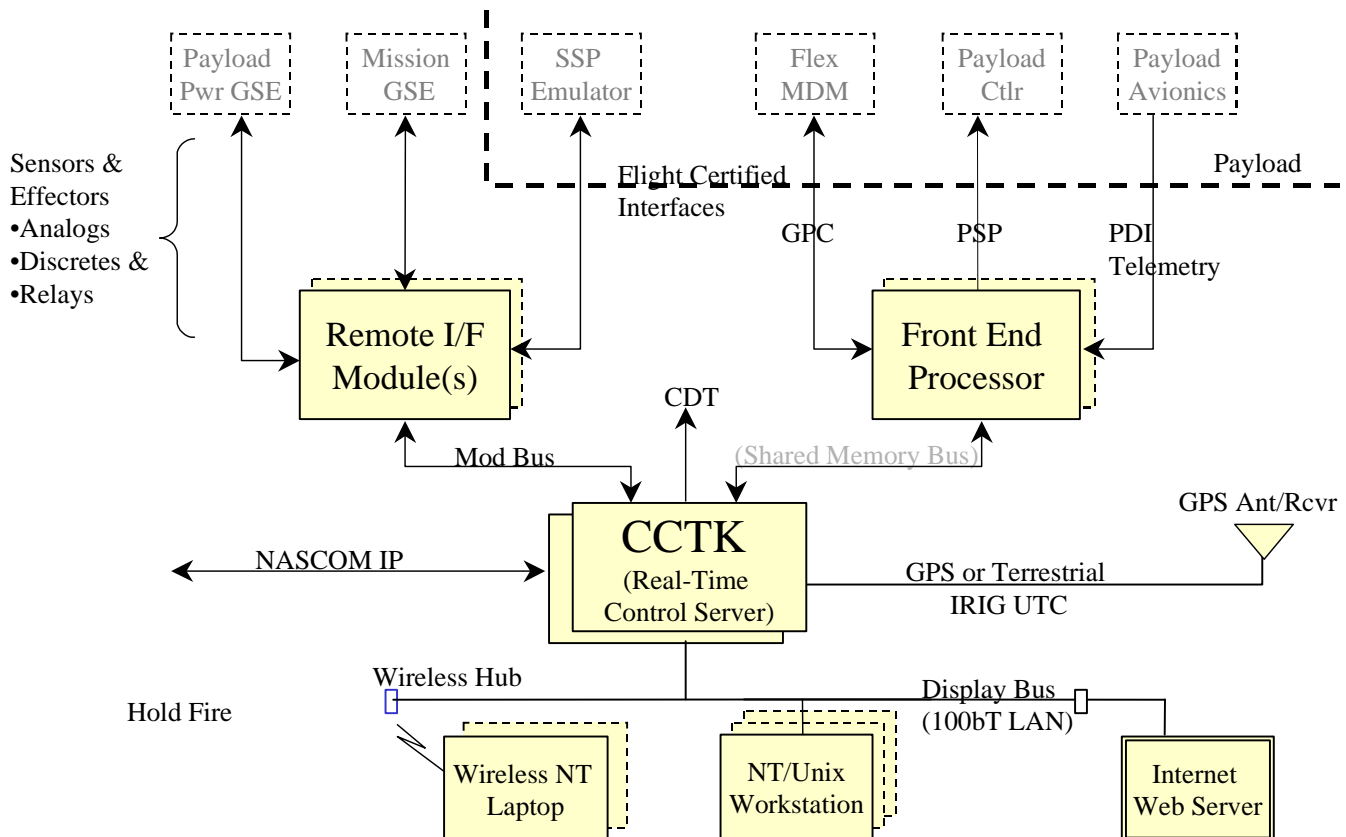


Figure 4. PCS System Architecture

cohesive/like forms of data processing, and minimizing the need for complex interfaces and

protocol transition. All layers are integrated across open interfaces that can be configured to a scalable variety of transport protocols, including Ethernet, FDDI, and ATM. This configuration can also be packaged for highly portable operations by configuring the Real-time Control Server (RTCS) in a high end laptop or suitcase computer configuration.

High speed vehicle data is acquired and decommutated from the external system interfaces, time tagged, and multiplexed into a normalized sequential time ordered data stream by the Front End Processor (FEP). FEP raw data is received and processed by the RTCS where it is converted to engineering units, linearized, limit checked, fused, recorded, and made available to the application environment.

Low level sensor and effector data are acquired via the Remote I/F Module (RIM). The RIM is comprised of one or more modules that perform analog and discrete I/O processing. The RIM can be distributed over long distances in order to accommodate remotely located end items like cameras, sensors, power supplies, and fluids GSE. Using the widely supported industry standard Modbus SCADA protocol, and off the shelf PLC components, the RIM provides a cost-effective approach to integrating support equipment control and monitoring.

The RTCS hosts the commercial T-Zero™ Automated Test Conductor program and other end item control and monitoring applications. The CCTK application environment provides data, command, and event software services for executing control and monitoring algorithms and serving graphic workstation clients with real-time graphical displays. Authentication services integrate security and access control to prevent unauthorized activities, including commanding. The application development environment provides the tools necessary to create new applications for mission specific control and monitoring algorithms. The RTCS also provides archive and retrieval services for all data, and operational transactions.

UTC acquisition is supported from GPS or terrestrial IRIG sources. The RTCS may be configured to provide CDT generation and distribution or receive CDT from an external master timing unit.

The CCTK graphic workstation (CGW) provides the user interface for navigation and presentation of real-time visual displays. Because the CGW is

based on the industry standard Microsoft NT operating system, additional third party tools are easily integrated.



Figure 5. Real-time Control Server Configuration

#### 4.1 Real-Time Control Server

The CCTK Real-time Control Server (RTCS) application-processing environment is based on the UNIX OS with strict compliance to POSIX standards. It can be hosted on a number of different open server environments including the Silicon Graphics Origin 200/2000, DEC Alpha, Sun Ultrasparc, and Linux server class platforms. The server host is based on scalable shared-memory multiprocessing machines with the performance capacity, features, and reliability needed to support the payload control and monitoring applications. Figure 5 illustrates an example configuration.

The combination of mature UNIX/POSIX computing services, real-time services, multi-language software development tools, and a commercial application library provides a comprehensive environment for creating and executing computation intensive real-time applications. SMP support (or equivalent) enables the PCS to cost effectively adapt to changing operational capacity needs for processing, memory and peripheral support.

Most COTS servers are designed to resist failures, coming standard with environmental sensors and a system controller to manage redundant power and cooling systems. Additional features for ECC memory and cache, dual SCSI controllers, and hot pluggable drives ensure uptime is maximized. Uninterrupted application availability is further enhanced with some commercial fail-over capabilities such as the SGI Failsafe™ software.

## 4.2 Command and Control Toolkit™

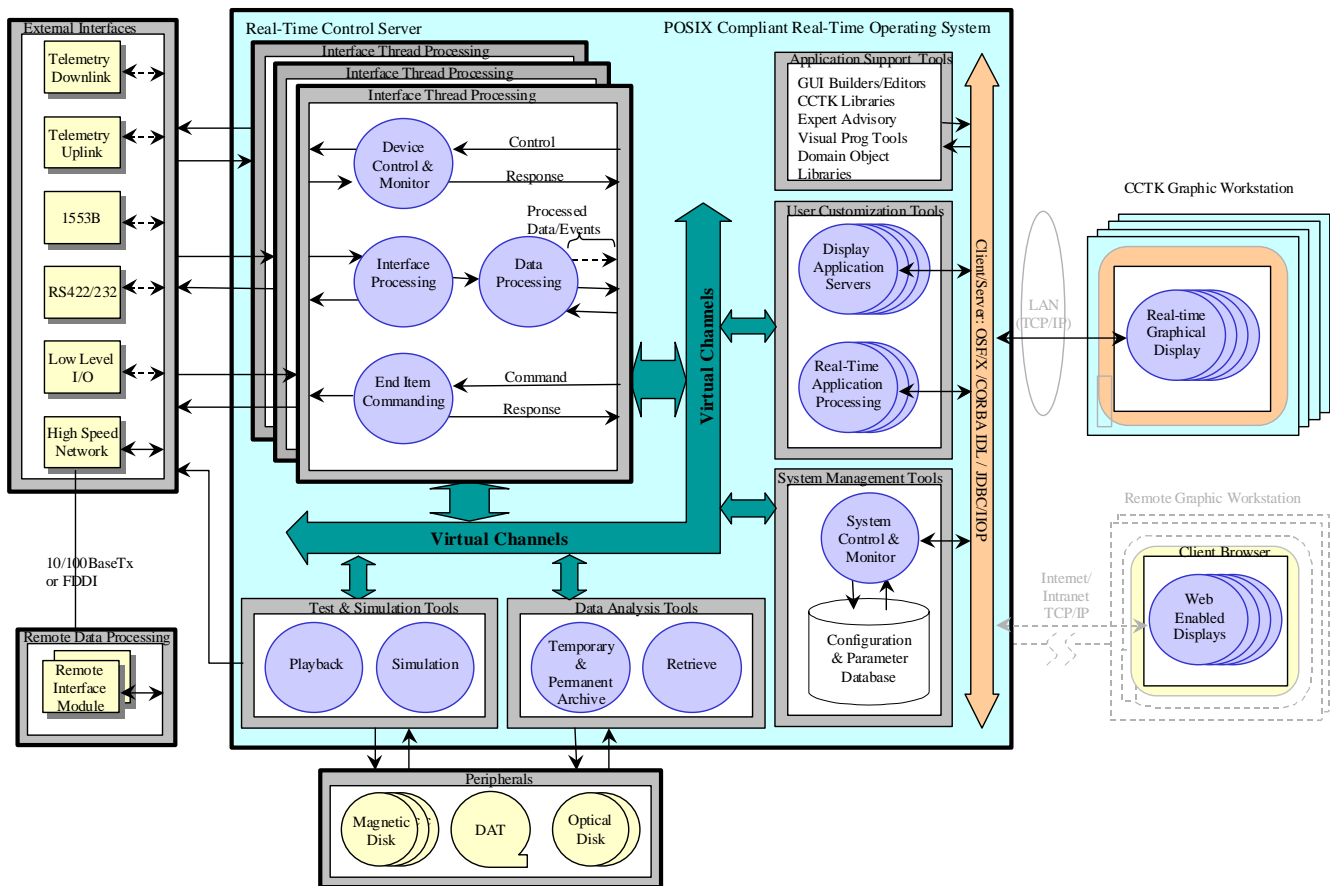
The Command and Control Toolkit™ product is a COTS command and monitoring software package that is hosted in part on the RTCS. This toolkit, based on proven software technology licensed from NASA, provides a customizable environment for real-time command and control applications. The CCTK is a high-speed, flexible data acquisition and control product that can be scaled to meet the performance requirements of all shuttle payloads. It provides a client server environment that is adaptable to stand alone control, monitoring, and simulation, or to distributed multi-operator integrated operations. Figure 6 shows the logical organization of the CCTK software. Since CCTK is designed to be a generic architecture, it can be adapted to a wide range of test/launch site applications. Systems based on this product are readily modified without recompilation to accommodate changing or new requirements typically encountered throughout the lifecycle of a payload program.

Configured as a PCS, CCTK will readily accommodate shuttle payloads and associated

ground systems as a missing element simulator functioning autonomously or man-in-the-loop. CCTK provides automated tools for sequencing, closed-loop finite-state-machine control, data fusion algorithms, data visualization, and multithreaded parallel processing. Many control and monitoring automation functions are achievable without application software using built-in command and monitoring interfaces and the data visualization tool set of Glg Toolkit™ (DataViews, and SL-GMS are also available as optional integrated data visualization tools).

### 4.2.1 Application Development Environment

The CCTK real-time services have a comprehensive C/C++/Java language application programming interface for processed data access, automated event processing, end-item commanding, user display dialogue, and system control. Application development is supported with application interface simulation, data analysis, and debug tools that shorten the pre-deployment time for user specific customization in the field. Applications can be efficiently developed off-line or in parallel with on-going operations.



Command and Control Technologies, 407.383.5282, info@cctcorp.com

Figure 6. Command and Control Toolkit Software Architecture

#### 4.2.2 Scripting Environment

The CCTK scripting environment provides the capability to process command line inputs as well as executing complete programs (See Figure 7).



Figure 7, Virtual Machine Scripting Environment

A visual, spreadsheet-oriented debugger is also provided for ease in program development and debugging. Some of the capabilities of the scripting environment include:

- Automation of command and data processing without code compilation, enabling on-the-fly operations
- Type free variables (all variables are implicitly floating point), commands, and data afford tremendous flexibility and simplicity in algorithm creation so significant automation can be achieved with no programming skills.
- Scripts can be executed from within the interactive spreadsheet GUI or executed from other scripts or applications
- Integrated checking for dimensional consistency and units balancing
- Interpretive subset of the ANSI Standard 'C' programming language allows for sophisticated algorithms

#### 4.2.3 System Management

CCTK provides the capability to reconfigure for each test/mission nominally within one hour. The CCTK provides simple database and script oriented interfaces for entry of test-dependent

parameters. This feature also allows users to enter data for new missions while processing another.

System control functions allow CCTK to be reconfigured from one mission configuration to another with just a few mouse clicks. A mission is defined in the configuration and parameter database, specifying logical resources, commands and data as shown in Figure 8. Data base utilities serve to normalize resources, data, and command parameter definitions and provide the reference structures necessary for binding commands and data to applications and displays during real-time operations. New command and measurement parameters can be added, deleted, or modified without impacting system or application software. The configuration and parameter database is based on the Microsoft Access database and provides the user with the ability to edit database files offline from the real-time system software.

The Access format provides an extra level of database management in that the databases can be separated from the CCTK system, stored offline, and transferred via standard email attachments to other users for editing purposes. Use of the popular Access editor for database editing will save training costs.

The database file structure is mission oriented. All programming elements and configuration files for a project are stored under a single file name for ease of database management and system re-configuration. New database configurations are easily ingested or exported to/from the common database file format or spreadsheet.

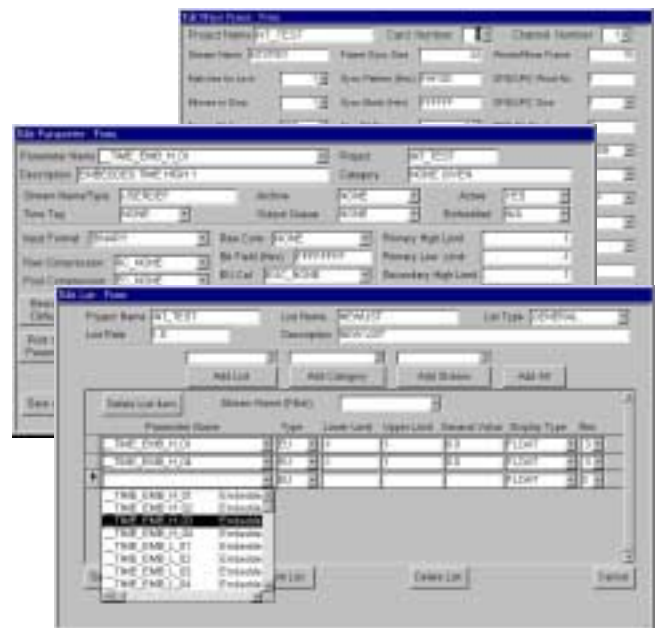


Figure 8, Configuration Database

#### 4.2.4 Archival and Retrieval

CCTK provides a highly robust data archiving capability that records all data, commands, events, and system messages in time ordered fashion for later retrieval. The toolkit provides client/server access to all archived information via retrieval tools that are accessible to all graphic workstation clients. Data may be accessed either as a processed retrieval or as a data playback. Separate archiving processes support the simultaneous continuous real-time data archiving and retrieval and viewing of previously archived data. The user can also display real-time data and view previously archived data at the same time on the same, or multiple display CRTs. These capabilities include:

- Processed data retrievals are initiated based on archival test/mission ID, start/stop time, and an array of user selectable access filters such as: information ID(s), source, events, limit violation, user, etc. Retrieval definitions are defined and executed or stored in a file for later execution or batch processing. Retrieval results are stored in an ASCII file which can be displayed, printed in tabular form, or plotted graphically
- Playback accepts data items, conditionals, start/stop time slices or any combination as input functions for playback parameters. In addition to the processed storage format, the CCTK archive process stores data in an object oriented data structure prior to parameterization, engineering unit conversion, or derived parameter calculation. The archive data structure consists of the raw data as it was directly received by the front-end processor. This data structure also includes embedded time data. Since this archive data structure closely mirrors the exact data structure that is ingested by the CCTK processing engine, the CCTK system employs the unique ability to simply instantiate a second data processing routine to process archived data for playback. This architecture gives the user the same data-driven processing environment of real-time data for use when playing back previously archived data. The same displays can be used in playback or real-time modes.

#### 4.2.5 Simulation, Replay, Training and Practice Facilities

The PCS configuration provides a robust and configurable real-time simulation environment. It delivers a set of simulation tools that work interchangeably with application level simulation and with end-item interface simulation. The

simulation environment architecture is shown in Figure 9.

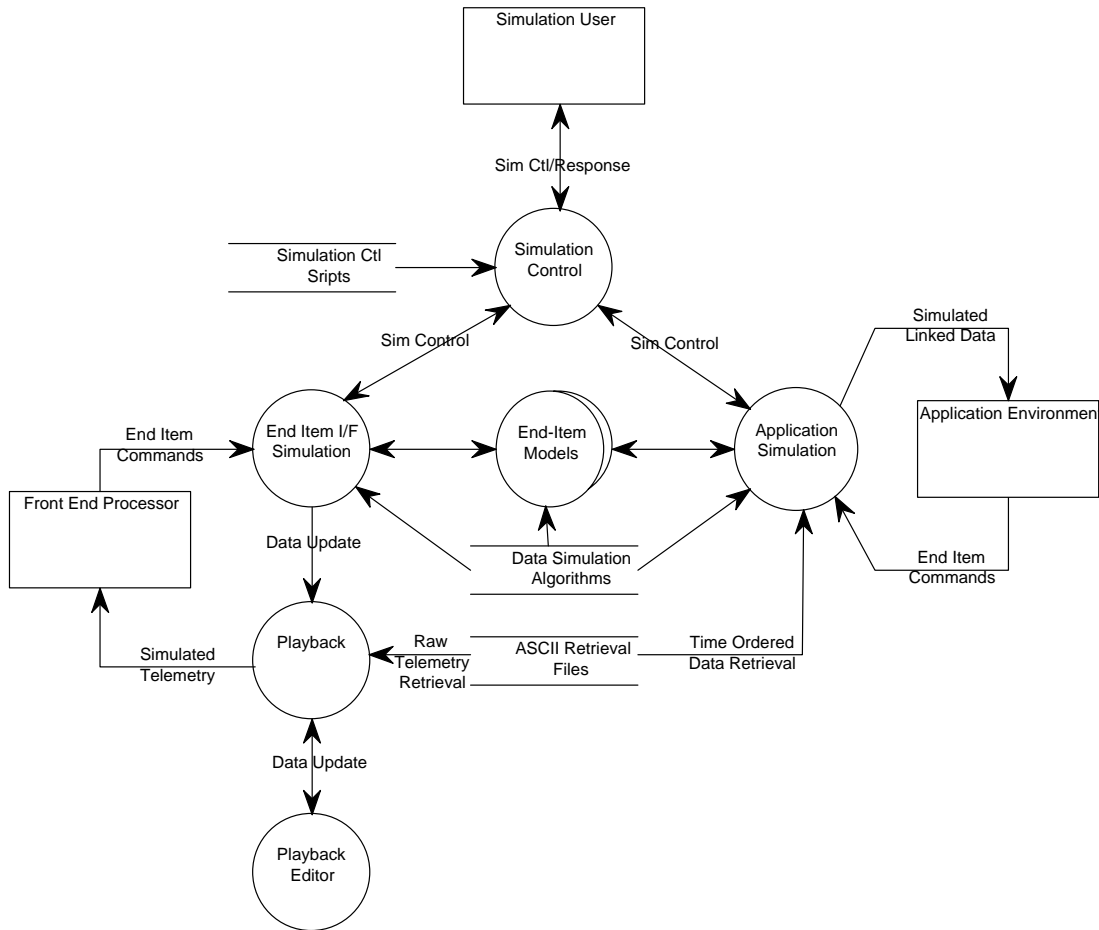
The CCTK simulation capability is designed to support three principal tasks:

- Application development and testing – Provide low level data dynamic simulation for checkout of applications, displays, and configuration databases.
- User training for mission operations - Provide nominal and non-nominal mission-based simulations for operations personnel training.
- System testing and validation - Provide configurable test scenarios for systems' problem resolution, testing, and operational certification.

The CCTK simulation is designed for flexibility, providing a range of mechanisms for realizing simulations. Simulations can be achieved using anyone of the following:

- CCTK Simulation Engine – An interpretive environment for creating basic dynamics utilizing a high level GUI. It requires no programming expertise. It functions by providing a spreadsheet like mapping between stimuli and measurement dynamics.
- Scripts – Since the CCTK simulation behavioral dynamics are implemented in an application library, script environments like the built-in CCTK “Virtual Machine” and Tcl/Tk can be used to create simulations on-the-fly.
- C/C++ or Java – Full native language support provides simulation flexibility only possible with a complete programming language. Also, since C/C++ is the native CCTK system language, performance and response times are optimized for demanding applications not appropriate for the interpretive options listed in the previous two bullets.
- Visual Modeling Tools – CCTK supports third party modeling tools such as MATRIX<sup>TM</sup>, and G2<sup>TM</sup> integrated via the CCTK real-time C/C++ application library. These tools offer alternative paradigms to simulation creation; however, there are performance limitations associated with these tools that need to be considered when applying them to real-time interactive models.

Three simulation modes are supported that allow simulation to range from simple command response behaviors to interactive dynamic end-item modeling:



**Figure 9. Simulation Architecture**

- Data simulation is used to map built-in or user defined data algorithms for manipulating data (ramp, sign wave, random, etc) over time. This method of simulation is ideal for application development and testing because it enables an application developer to work independently with a minimum of outside dependencies. Further, the simulation can be created and executed on-the-fly because data simulation can be executed interpretively (step-by-step) and can be driven largely by the command line or script. When created by script, data simulation scenarios can be saved to a file and reused later in integrated or regression testing.
- Low fidelity dynamic models are created using the built-in data simulation library or third party libraries. Models are somewhat more sophisticated than simple data simulation. These are used to mimic end items such as radar, propulsion systems, GSE, and navigation subsystems behavior. Measurement and command parameters can be organized into logical groupings of components, subsystems,

and systems that exhibit behavior in response to commands, and events such as time or user intervention.

- High fidelity dynamic models are created to accurately mimic dynamic end-item characteristics. This type of modeling is typically used in design analysis, representing non-linear behaviors, or when model components need to be complex and highly interactive. This level of modeling is best implemented using the CCTK application library, which can include specialized modeling tool integration.

CCTK has the unique capability to playback previously recorded data and commands scenarios into the system for interactions with models and/or into the external interfaces. The playback can interact dynamically with active simulation models. This mode of simulation provides powerful flexibility in creation and reuse of test scenarios to support the PCS.

#### 4.2.6 Interface Processing

The basic technology is adaptable to a large number of applications in large part because of the ease of integration of external interfaces. For the PCS, CCTK is configured to communicate commands and data to and from front end processors, telemetry acquisition systems, and other vehicle or spacecraft elements. Because interface-processing structures are defined in the configuration database, it is possible to define and allocate many interface processing configurations without any configuration or application specific software.

CCTK interface processing software includes real-time processing capabilities for handling numerous types of input data. Standard algorithms are provided with the system and include: first to ninth order polynomial conversions, data compression based on significant change value, limit checking, and many more applicable computations.

#### 4.2.7 Event Processing

The event processing capabilities of CCTK are well suited for handling both internally generated data triggered events as well as externally triggered events from any of the front end processor inputs. Any measurement in the system may be associated with an event by defining event conditions (sometimes called "exceptions") in the configuration database. Up to nine limit conditions may be defined indicating such things as high/low critical or high/low cautionary data ranges. During real-time operations, the system will automatically evaluate real-time data against the predefined limit conditions. The occurrence of an event can be used to initiate a reactive control sequence or GUI dynamic. For example, a discrete data input could trigger an exception when it becomes one (on) or zero (off). This input can be from an instrument, break wire, or a status bit defined in the downlink telemetry stream. This event triggers an application response to safe the test, initiate an event handler, and/or notify the system user. Exceptions are time-stamped and recorded for post mission analysis. Exceptions can also be associated with derived data.

Events enter into the system in one of three ways. They can be (1) the result of user-generated manual commands selected from a menu, (2) CCTK-generated system events or exceptions caused by data input changes as described above, or (3) a temporal event such as the countdown time reaching a specific value. Regardless of the event source, CCTK provides a wide variety of response actions and facilities to

program the resulting action. These facilities include a built-in scripting environment that provides a means to alter the action without recompiling, reloading or restarting the system. The event can trigger a user-defined application program, allowing the user to create functions requiring fast reactive control. Both applications and scripts can execute end item commands to trigger devices like cameras or video controls.

#### 4.3 Graphical User Interface

The CCTK graphic interface provides the client user interface for interacting with the PCS. The PCS may have any number of graphic workstation clients connected via standard TCP networks. User access and permissions control functions are defined pre-mission and invoked via user logon authentication. The CCTK software package includes a graphical user interface constructed in a main menu format to give the user a single menu for access to all system functions, including configuration definition, system control and monitoring, system displays, application displays, data retrievals, and other functions.

Any standard or user created screen is accessible by any user/client in the system. All screens are pre-configured and selectable from the system desktop menu. In fact, any authorized user may create their own custom screen(s) and bind it to real-time data within the system using the integrated GUI builder.

Because all real-time data is ingested and processed by the data processing engine, any client connected to the RTCS can access any data defined in the configuration database including calibration, visibility, and statistics data.

One of the standard system displays is the scrolling strip chart, which can be viewed and printed from any Microsoft NT client. The CCTK software allows the user to create a single display screen containing up to 64 parameters displayed as scrolling strip-charts. Users can use the scrolling strip-chart displays in real-time mode, retrieval mode, or both. Data-driven displays allow the user to see individual data samples versus time in a scrollable X/Y plot.

Application displays are typically dynamic graphic interfaces designed to provide intuitive visual interfaces for end-item control and monitoring. They may be created by the system user with the Glg Toolkit or other similar data visualization package such as DataViews or SL-GMS. Some of the general capabilities of the application displays include:

- Real-time schematic model views of vehicle and ground subsystems for logical and intuitive operations.
- Selective integration of manual controls for individual device control and monitoring, and capability to initiate automated sequences for closed loop hands-off control, providing operations personnel optimum flexibility in operations support.
- On screen capability for information hiding and drill down details provide for timely tuning of display details to fit the operations need in real-time.
- Graphics, color, and digital dynamics are used for instrumentation readout, prompts, command responses, event/state and limit violation indicators, and health and status conditions.
- A critical, cautionary, and informational messaging panel is integrated for posting significant event details to all operations personnel.

#### 4.3.1 Glg Toolkit™

Real-time visual display creation and editing is provided via the integrated Glg Toolkit. The Glg displays are actually created and executed on the RTCS and displayed remotely on any CGW console position via X-client emulation software provided by Exceed™ from Hummingbird Software.

The Glg Toolkit is a high-performance dynamic graphics system for real-time applications. It is especially suited for creating real-time applications with a high volume of data that needs to be represented graphically. The Glg Toolkit provides an interactive environment for creating dynamic 2D and 3D graphics in a point and click editor, and a powerful but simple resource-based API for supplying dynamic data and integrating graphics into the PCS application.

The Glg Toolkit consists of the following major components:

- GLG Graphics Builder - The Glg Graphics Builder is an interactive 2D and 3D graphical editor with a point and click interface. It is used to create custom dynamic visualization components for Glg run-time, C/C++ library, Glg Java Bean or Glg ActiveX control. The features of the Glg Builder include 3D dynamics, real-time constraints, complex resource hierarchies, and double-buffered display to eliminate flickering.
- Real-time Dynamic Displays Engine - A library of classes and functions for embedding

dynamic graphics into an application. It provides functionality for connecting graphics to the application data and includes C, C++, Java and ActiveX interfaces.

- Glg and Java Web Tools - The Glg Java Bean, Netscape plug-in and ActiveX control may be used for creating graphical data-driven web applications that work with both Netscape and Internet Explorer web browsers.
- Glg Tools for UNIX and real-time operating systems - The Glg Toolkit provides a cross-platform graphical environment that can be used on a wide variety of platforms: UNIX, VMS, real-time operating systems, and Windows. Applications built using the Glg Toolkit and Glg drawings may be deployed on any of these platforms using the same set of Glg tools.
- Glg Pre-Built Visualization Components Library - In addition to the custom visualization components that can be created by the PCS user using the Glg Graphics Builder, the Glg Toolkit provides a large collection of pre-built reusable visualization components: 2D and 3D graphs, meters, dials, and other items developers can use. These components can be customized interactively in the Glg Graphics Builder. New components or component panels can be created by modifying or combining existing components.
- Glg Tools and Utilities - A set of utilities for importing maps and CAD drawings, handling large scrolling tables, generating data for prototyping, and other functions.

The diverse collection of pre-built visualization components, flexible deployment model, GIS tools and rapid prototyping features, cross-platform and web deployment capabilities make the Glg Toolkit an ideal tool for building PCS applications ranging from simple animations to complex system monitoring.

#### 4.3.2 OMNI™

OMNI™ provides the means to visualize the results of complex simulations involving spatial relationships between user-defined objects such as satellites, aircraft, ground sites, and missiles. OMNI provides a high-end modeling capability for data analysis and display of a single target or group of targets.

OMNI's "look from anywhere, to anywhere, at any time" philosophy enables mission designers and range analysts to receive and understand complex information in a way that cannot be

achieved through more traditional presentation techniques. Some of its key features include:

- 3D and 2D simulation, modeling, and visualization
- Ground, air, and space-based sensors, and communications volumes
- Satellite orbital mechanics
- User-defined tracks (aircraft, ground vehicles, ships, etc.)
- Visibility statistics and graphics
- Ground and sky traces
- Optional Programmer's Interface for data exchange with other applications

#### 4.4 Real-time Control Application Software

PCS requirements for real-time control (RTC) and monitoring is implemented based on the generic RTC Application Architecture shown in Figure 10. This architecture provides a simple model of the interfaces and relationships between the applications, CCTK real-time services, and CCTK customization tools.

The RTC application environment provides a

homogeneous interface for data and event access, command issuance, application interaction, and creation of derived data. Control applications can be created independently of the underlying interface processing and communication protocols. Further, the underlying UNIX operating system provides services for parallel processing, file system access, and inter-process communications. Any number of RTC applications and monitor and control client displays may run concurrently. Typical example applications include end-item controllers such as power, avionics, and hydraulics.

In many cases it is possible to meet all of a particular application's needs without writing any application software simply by creating a monitor and control display using the GUI builder.

##### 4.4.1 T-Zero™ Automated Test Conductor

Large, complex group activities such as vehicle checkout and launch processes are traditionally coordinated verbally, often with the help of written procedures. No capability has previously existed for visualizing the test/launch process itself or for automated coordination of group activities. The T-Zero Automated Test Conductor solves these problems by providing a visual, role-based, interactive timeline in the familiar PERT and Gantt

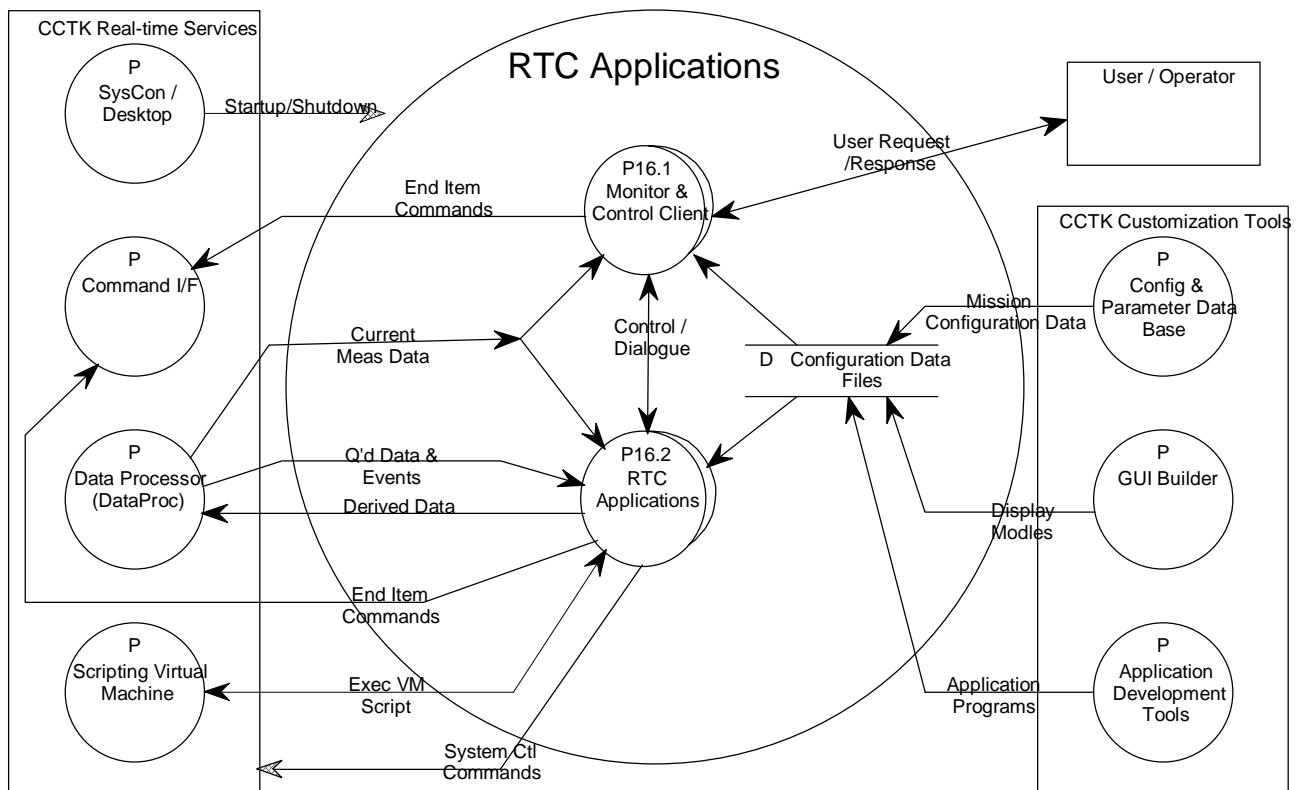


Figure 10. Generic Real-Time Control Application Architecture

chart formats. The timelines are assigned to each activity participant, console, or operator and animated in an intuitive fashion to reflect the ongoing operation. Symbols representing key events are placed along each timeline to denote process milestones, executable procedures, event status, and group decisions and interactions. The test or launch process is then 'conducted' as activity team members sequentially interact/click their respective symbols. Using T-Zero, the process designer can gate activities of one or more roles by linking them to real-time events or symbols on other time lines.

T-Zero is comprised of two principle elements:

**Automated Test Conductor** - Serves as the primary runtime interface of T-Zero.

**Conductor Toolkit** - Used off-line to create the visual sequence timelines.

Figure 11, Automated Test Conductor Interface – PERT View depicts the T-Zero Automated Test Conductor utilized in a simplified launch sequence. The interface resides in a standard dialog with familiar window controls and menus. Additional buttons complement the timeline symbols and provide a drag-and-drop capability

when constructing process timelines. The timelines representing the test/launch sequences for each role appear in a scrollable window whose upper border depicts countdown time. There are three console positions shown in the Figure 11 example: Test Conductor, Vehicle Engineer, and GSE Engineer. Consequently, there are three timelines containing sequences of milestones, events, executable procedures (or scripts), and group interactions/decisions. The diagonal lines, from a symbol in one timeline to a symbol in another, prevent execution/interaction of the target symbol until the successful realization of the pre-requisite symbol. Clicking on a role button brings up a chat/video interface for communication with colleagues.

In Figure 11, the role name appears on a button to the left of each timeline. Note that most symbols on the Test Conductor timeline are milestones. Symbols appearing on the other timelines include "test procedure" depicted by a document icon with the letter 'T', and "confirmation" depicted by a circle surrounding a dot. Realistic timelines would also include the other interactive symbols appearing in the toolbox. These include events such as concurrence, decision, document, person/group,

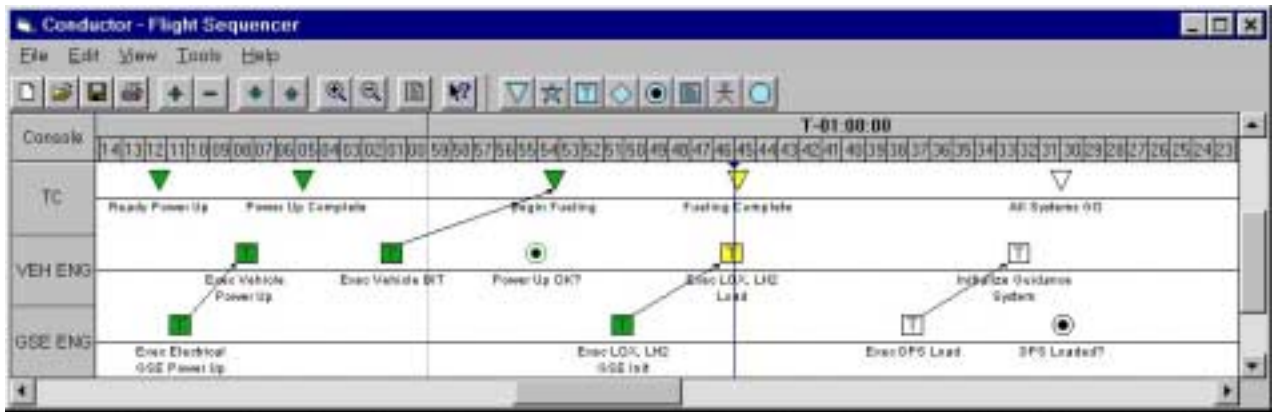


Figure 11, Automated Test Conductor Interface - PERT View

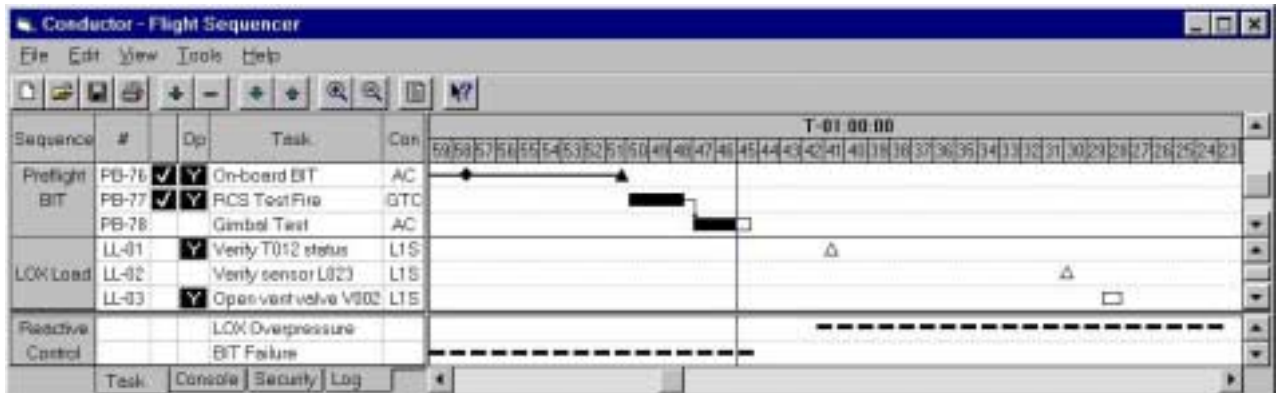


Figure 12, Automated Test Conductor Interface - Gantt View

and status.

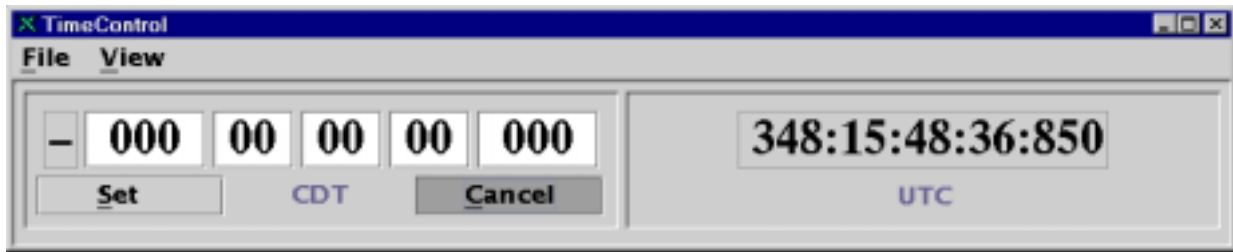


Figure 13. CCTK Time Control User Interface

Actions successfully completed are denoted by green-filled symbols, actions in progress appear in yellow, while actions not yet done appear outlined in black. A moving vertical line indicates current countdown time.

Figure 12 shows an alternate representation of the simple countdown sequence depicted as a traditional Gantt chart. This view is particularly useful when visualizing sequences of significant complexity. Task details can be interactively hidden or exposed using hierarchies or roll-ups, much like is commonly used in commercial project management packages like Microsoft Project.

Because T-Zero interacts with real-time data and events, a high degree of automation and autonomous control is achievable by embedding decision-based logic in the timeline and task symbol attributes. Every symbol becomes a virtual finite state machine that can trigger any number of possible actions.

#### 4.4.2 Countdown Management

The PCS provides timing precision of up to one microsecond for time stamping event correlation. MTU Universal Time Code (UTC) synchronization and Countdown Time (CDT) synchronization are provided by standard CCTK Time Services. Once the time has been received by the PCS it is normalized as processed data and made available for use anywhere in the system or display at any workstation via the network interfaces.

Additionally, the PCS operators may choose to use the available time management services of CCTK for CDT control. These services can be configured to control an internal CDT or the MTU (this assumes external control interfaces are available). CCTK provides the ability for the operator or a real-time application to start, hold, resume and reset CDT as shown in Figure 13. The time services can provide a backup time source in case of an MTU failure or detected time shift. The PCS can be commanded to revert back to using the MTU at any time.

#### 4.4.3 Message Mechanism

A PCS message mechanism is also provided (See Figure 14 below). The System Message Window allows the PCS user to track system messages that display various types of system and user communications and acknowledgments. The window is a graphical interface activated by any system user from the System Application Menu that displays four types of system messages:

- CRITICAL. (red) Critical system messages are those messages that need immediate attention from the user. Examples of critical system messages are cases when a particular subsystem suddenly malfunctions or stops reporting valid health or an application terminates unexpectedly.
- CAUTIONARY. (yellow) Cautionary system messages are those messages which the user should acknowledge, but do not demand immediate attention. Examples of cautionary system messages are cases when a subsystem loses data or when health counts are not changing.
- INFORMATIONAL. (blue) Informational system messages are those messages that are merely acknowledgments and informational data that the user may want to know, but can be acknowledged at the user's discretion. Examples of informational system messages are cases when the subsystem is successfully activated.
- USER COMMUNICATION. (user defined color) User communications messages are those messages generated by system operator. The sending operator may designate messages to be broadcast (default) or point to point based on user address. The user may designate a message as CRITICAL, CAUTIONARY, or INFORMATIONAL when the message is generated.

Messages may be viewed in two ways: categorized by message type or ordered by time.

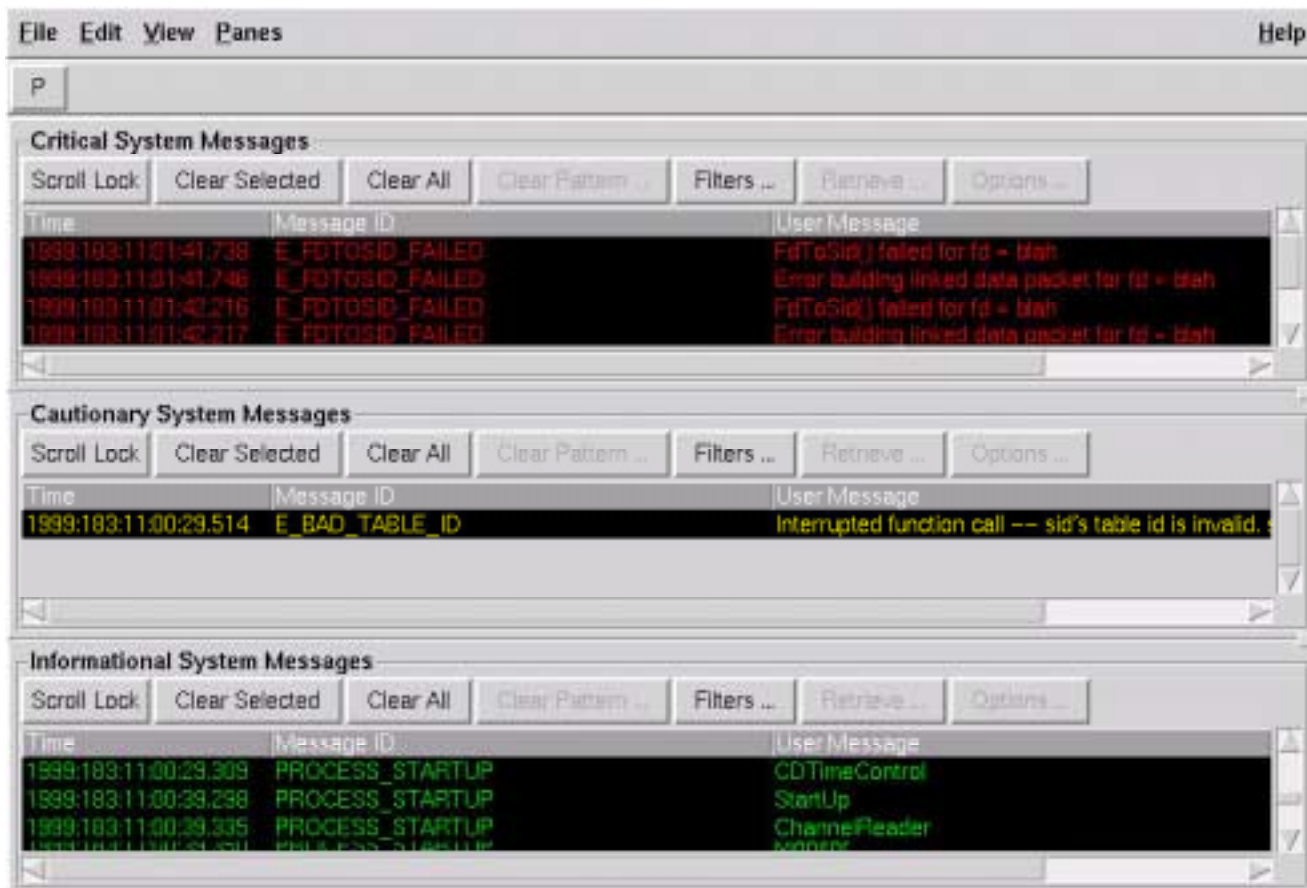


Figure 14. System Message Display

The format is chosen by selecting the SHOW BY: TIME/CLASS button. In the class format the system messages of each criticality class are shown in separately controlled lists. In the time format the system messages are grouped together so that the order in which the messages are received is more easily determined. The default format is by class.

The messages that populate the message windows are managed in the configuration database. The configuration database provides a simple table that maps the events that cause the messages to be generated to the text which is actually displayed in the message windows. The system administrator has the ability to modify existing messages or create new ones. New messages are typically introduced as new user created applications are created and deployed. This feature provides a tremendous amount of flexibility in tuning the system operations to user needs.

All system messages are recorded (by default) and may be retrieved near-real-time or post mission.

## 4.5 Real-time Web Server

CCTK supports the wide distribution of information, quickly and inexpensively, by allowing operators and remote clients to access data via an intranet or the Internet. Access is provided through the CCTK Real-time Web Server that can be configured to run locally on the

RTCS or remotely via COTS web server software. The server may optionally be configured for secure operation via a firewall. Java displays developed with the Glg Toolkit can be quickly configured to run within the PCS or via the web. Other display tools, such as DataViews, can also be used.

This combination of inexpensive, widely used, and readily available computing resources simplifies maintenance and makes it easier to adapt to new requirements. With the current development of an Internet protocol via satellite, global communication and distribution of data will be much more efficient and inexpensive. The potential of Internet technology will continue to expand, and its applicability to space payload systems will increase as the industry transitions to a more commercial orientation.

## **5.0 Conclusion**

CCT has joined in partnership with NASA to make PCS a reality for Space Shuttle payload operations. A subset of the final capability has been demonstrated as part of the Portable Payload Tester project completed at KSC in 1998. The PPT system successfully demonstrated most of the operational capabilities described in this paper. Remaining work for PCS includes implementation of a complete set of orbiter payload interface services.

The software checkout technology upon which PCS is based is a general-purpose test and simulation package. This underlying technology will enable streamlined payload processing with new and innovative concepts of operations for future missions, including factory to launch site portability and internet-based ground operations. As an early success story from NASA's Spaceport Technology Center at KSC, this technology promises to play a key role in spaceport operations of the future.