



Compro Computer Services, Inc.
105 East Drive
Melbourne, FL 32904
Telephone: (321) 727-2211
Fax: (321) 727-7009
www.compro.net

LCRS Technical White Paper

Legacy Computer Replacement System (LCRS)

April 2009

Notices

©2009 Compro Computer Services, Inc. All rights reserved. No part of this document, including text, code examples, diagrams, or icons, may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording, or otherwise) without the prior written permission of Compro Computer Services, Inc.

Information in this document is subject to change without notice. Compro Computer Services, Inc. may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you license to these patents, trademarks, copyrights, or other intellectual property. Please send licensing inquiries to: Compro Computer Services, 105 East Drive, Melbourne, Florida 32904.

Limit of Liability/Disclaimer of Warranty: This document is licensed and/or sold "as is" without warranty of any kind, either expressed or implied, regarding the contents of this document, including but not limited to implied warranties for the book's quality, performance, merchantability, or fitness for any particular purpose. Neither Compro Computer Services, nor its dealers or distributors shall be liable to the purchasers or any other person or entity with respect to any liability, loss, or damage, caused or alleged to have been caused directly or indirectly by reliance upon the contents of this document.

Trademarks

Compro, the Compro logo, LCRS, and other branded items are trademarks or registered trademarks of Compro Computer Services, Inc.

AMD is a registered trademark of AMD Corporation.
Ethernet is a registered trademark of Xerox Corporation.
HP is a trademark of the Hewlett-Packard Company.
Intel is a registered trademark of Intel Corporation.
Linux is a registered trademark of Linus Torvalds.
SPARC is a trademark of SPARC International, Inc.
Sun and Solaris are trademarks of Sun Microsystems Corporation.
SuSE is a trademark of Novell Corporation.
Windows is a registered trademark of Microsoft Corporation.

All other product, service, and company names are trademarks or registered trademarks of their respective owners.

Compro Computer Services, Inc.
105 East Drive
Melbourne, Florida 32904

Table of Contents

GENERAL DESCRIPTION	5
COTS HARDWARE.....	5
INDUSTRY-STANDARD SOFTWARE	5
PCI-BASED INTERFACES	5
LEGACY SIMULATION SOFTWARE	5
FUNCTIONAL DESCRIPTION.....	7
LCRS BLOCK DIAGRAM	7
LCRS VIRTUAL ENVIRONMENT	8
LCRS CONFIGURATION.....	9
LCRS SOFTWARE PROGRAM STRUCTURE	9
BASIC SYSTEM PERIPHERALS.....	10
STANDARD LEGACY SYSTEM PERIPHERALS – UPDATE CONSIDERATIONS	10
I/O SUBSYSTEM PERIPHERALS	11
SOFTWARE TOOLS.....	12
DISTRIBUTED REAL TIME EXTENSIONS (DRTX) FOR DEVELOPMENT SYSTEMS.....	12
LCRS UTILITIES	13
MPXDISC.....	13
MPXTAPE (Optional).....	13
MPX_REGS and MPX_BREGS.....	13
MPXRSCH	14
LCRS CONFIGURATION FILE DIRECTIVES	14
BIXLOOP	14
IDELAY	14
MANAGERIAL CONSIDERATIONS: ADDRESSING PROGRAM REQUIREMENTS	14
MANAGERIAL CONSIDERATION – TECHNICAL	14
Expandability.....	15
Expandability Solution: ZI ³	15
ZI ³ Foundation: PCI Reflective Memory System	15
Performance.....	17
Peripheral Upgrades	17
MANAGERIAL CONSIDERATION – LOGISTICAL.....	17
Deployment Time	17
Maintainability	18
Configuration Management.....	18
Future Proofing	18
MANAGERIAL CONSIDERATION – BUDGETARY.....	18
Environment – LCRS is “Green”.....	18
Reliability.....	18
Training	19
Sparing.....	19
Long-Term Supportability.....	19

SECURITY CONSIDERATIONS FOR MILITARY INSTALLATIONS 20

SUMMARY 21

List of Figures

FIGURE 1. LCRS BLOCK DIAGRAM 7

FIGURE 2. LCRS VIRTUAL COMPUTER DIAGRAM 8

FIGURE 3. LCRS PROGRAM STRUCTURE 10

FIGURE 4. LCRS/MPX I/O TO LINUX DEVICE 11

FIGURE 5. DRTX FUNCTIONS 12

FIGURE 6. SECURE LCRS OPTION 21

GENERAL DESCRIPTION

The Legacy Computer Replacement System (LCRS) is an integrated collection of **commercial-off-the-shelf (COTS) hardware, industry-standard software, PCI-based interfaces, and legacy simulation software.** LCRS' function is *simulating* physical hardware, with one primary goal – *preserve customer application software and connected I/O subsystems while improving performance.*

COTS Hardware

Hardware components include powerful server-class SMP-style microprocessors and motherboards, quick-swap SATA disk drives, PCI slots, 19" rack-mountable chassis, slide-in power supplies, and modular I/O. With COTS comes lower up-front cost, improved performance, new functions, maintenance simplicity, modular sparing, easy expandability, dramatically better MTBF, and reduced facility load (power, HVAC, and floor space).

Industry-Standard Software

LCRS is simply a real time C application that runs under Linux utilizing a customized kernel to assure maximum real time performance. Linux tool suites are readily available for real time application development, debugging and tuning. Additionally, legacy tools (Orbis, MPX Toolkit, Volmgr, etc.) remain fully functional, supporting the original environment. Finally, utilities are available that facilitate file exchange between LCRS and Linux environments (DRTX, MPX Tape), with legacy file manipulation and management possible in the Linux environment using COTS tools.

PCI-Based Interfaces

Compro offers 100%-compatible PCI-based interfaces that replicate legacy hardware functionality. These boards replace legacy SelBUS and MPBus boards, such as the High Speed Device (HSD), Real Time Option Module (RTOM), Ethernet controller, and Eight Line Asynchronous (8LAS) controller. Other legacy controllers (for example, IOP, MFP, MCP, TLC, SCSI, LP/FDC, ADS, AI, RMS, all tape and disc) are replaced using built-in or COTS-PCI I/O. The design objective is *retaining customer I/O devices and connectivity wherever required.* This means existing legacy subsystem connectors plug into the LCRS, without modification, and simply work as before.

When peripheral updates are desired (typically console, disk, tape and printer), new devices are easily connected. – Yet the original application software functions with them transparently. Modern peripheral features (better performance, increased capacity and some additional functions) become available to the original application code.

Legacy Simulation Software

The legacy simulation software is essentially the LCRS application that runs under Linux. It is available to accommodate the entire line of SelBUS-based SEL/Gould/Encore "Concept/Series" systems including 32/2X, 32/5X, 32/6X, 32/7X, 32/8X, 32/9X, 20x0, RSX, MICROSel, and MULTISel. All original hardware characteristics are retained, including instruction set, registers, operation modes, memory, and I/O.

With respect to performance, LCRS easily exceeds legacy systems it replaces. One may conclude, however, that increased performance will “break” the original application because of timing differences. This is not necessarily so, based upon the following:

Real time applications are by nature “frame-based.” This means a given quantity of code must execute within a specific, cyclic time boundary. Even though code execution will occur much faster under LCRS, overall execution is still governed by frame frequency. The net result is more available processing time per frame (that is, “headroom”), permitting even more instructions per cycle.

Occasionally, an unfortunate coding practice uses “do-loops” to provide a fixed delay. Total delay time depends upon:

- a. loop iteration count
- b. instruction execution speed

Because instruction speed is hardware-dependent, this practice inhibits code transportability. To circumvent this issue, LCRS offers a special initialization directive (BIXLOOP) that specifically detects and “slows down” these loops to match original hardware performance. *Other instructions are unaffected.* The result is retained do-loop timing compatibility while obtaining increased performance headroom.

Note that the original application source code is *not* required; simply lifting a binary image from the legacy machine and loading it into LCRS suffices. This addresses problems where source code is lost or poorly documented, or re-coding proprietary machine-language assembly code is prohibitive.

To further clarify the LCRS concept, the following points are offered:

- LCRS is not a cross-compiler; no application recompilation (nor source code) is required nor performed.
- LCRS is not a re-hosting tool; existing applications run in their native legacy language (assembly, FORTRAN, etc.), and legacy operating system (MPX, RTM, XPM, or custom OS). Connected I/O operational characteristics and drivers remain the same.
- LCRS is not slow, like typical *emulators*; modern multi-core COTS processors result in magnitude-order application performance increase, maximizing frame headroom.

Bottom line, LCRS is a simulator, not an emulator; applications run in an environment that retains original computer hardware functional characteristics.

FUNCTIONAL DESCRIPTION

LCRS Block Diagram

As outlined in the discussion above, LCRS is a software implementation of a virtual Concept/Series computer system running in the Linux environment. The existing MPX-32 Operating System (for example) and user application run on that Virtual computer “system”. Figure 1 portrays the virtual replacement computer system structure.

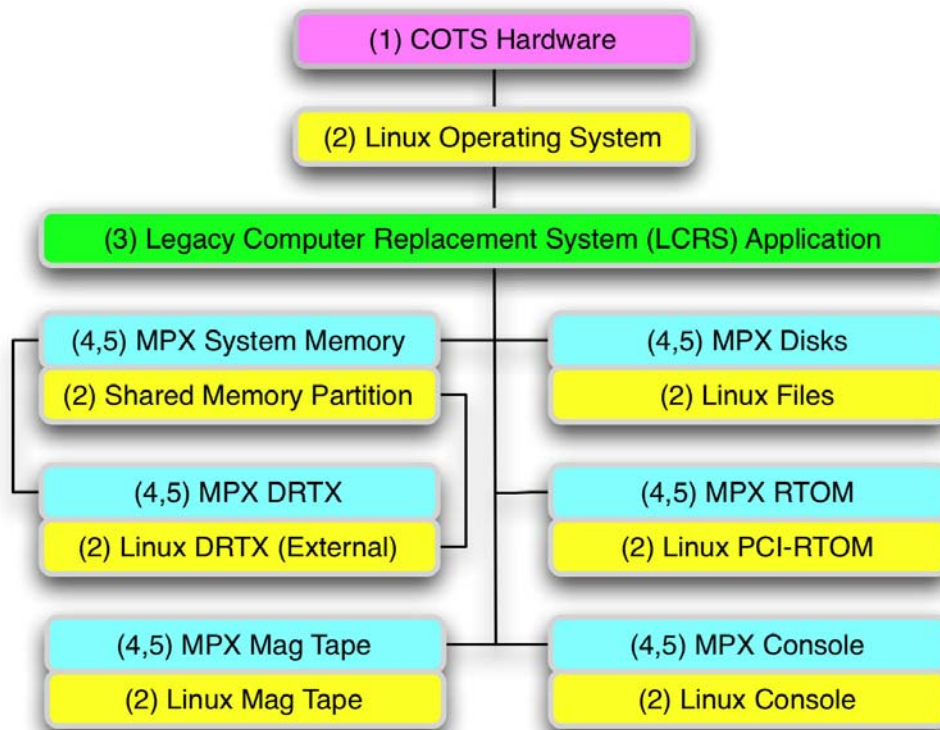


Figure 1. LCRS Block Diagram

As presented in Figure 1, multiple levels comprise the COTS computer environment:

- Level 1 is the COTS-based hardware system.
- Level 2 is the Linux Operating System running on its respective hardware. Note that Linux DRTX runs on an external node.
- Level 3 is the Legacy Computer Replacement System software application code (virtual Concept/Series computer) running as a Linux task.
- Level 4 is the standard MPX-32 Operating System running on the virtual Concept/Series computer system (and could be RTM, XPM or customized Concept/Series O/S).
- Level 5 is the user application software running in the standard MPX-32 environment.

LCRS Virtual Environment

Figure 2 functionally illustrates LCRS level 2-through-5 interaction and depicts Linux, MPX, legacy application and native task interplay.

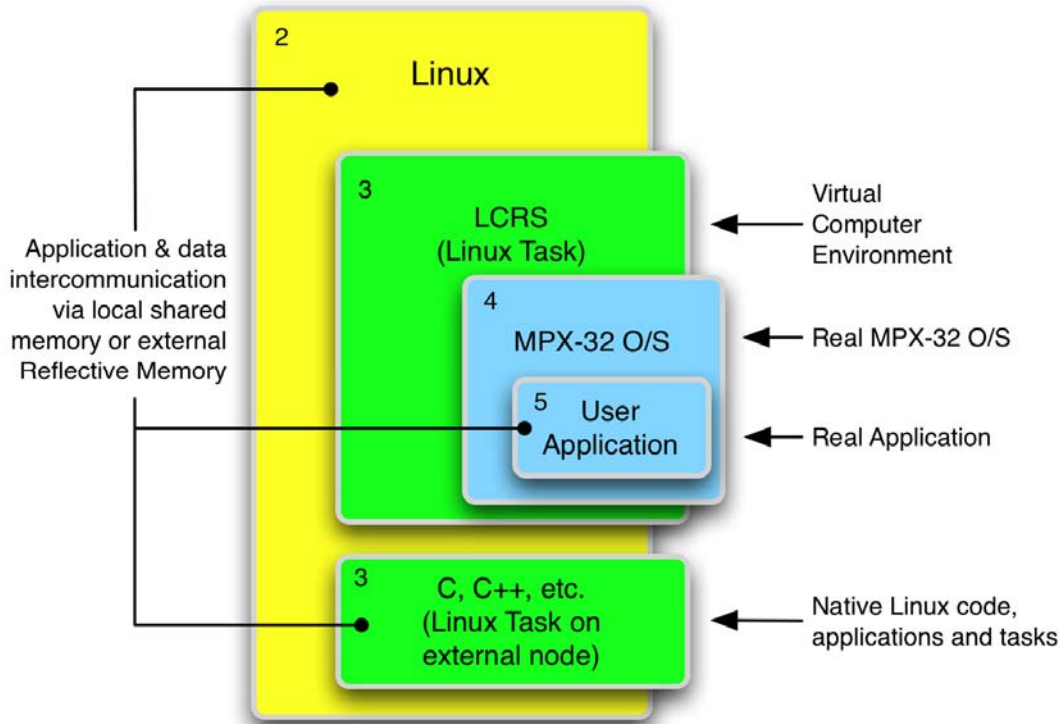


Figure 2. LCRS Virtual Computer Diagram

- Level 2: The Linux Operating System provides the foundation for all software presented to the COTS hardware system.
- Level 3: LCRS software is the virtual Concept/Series computer running as a Linux task.
- Levels 4 and 5: The actual legacy MPX-32 Operating System and user application software runs in the standard MPX-32 environment. This in turn runs on the virtual Concept/Series computer system.
- Level 3: Any desired new applications may be written in C, C++, etc., running as normal Linux tasks directly in the Linux environment. *These generally reside on a separate Linux node, coupled to LCRS via Reflective Memory. This eliminates LCRS performance perturbation.*

All LCRS and native communication is accomplished via shared memory. One or more additional, independent computer systems participate in this communication using Compro's PCI-based Reflective Memory System (PCI-RMS).

If desired, programmers may develop new applications and subsystems (for example, DBMS, TCAS, Wind-Shear, WRX, Client-Server models, etc.) using:

- Level 4: Gould FORTRAN (in the legacy MPX development environment)
or:
- Level 3: Modern ANSI FORTRAN and development tools running as Linux tasks
or:
- Level 3: Modern C ++ and development tools running as Linux tasks.

LCRS Configuration

LCRS uses a Linux-style configuration data file (.lcrsrc) to configure most of the system features at runtime. Configurable elements include:

- Memory size
- Simulated disk path names
- Device handlers
- Shared memory partition names to allow concurrent LCRS instances on one computer

LCRS features built-in parallelism that provides threads for the Concept/Series simulation, delivering true computational concurrency. There is one thread for each of the following simulation elements:

- CPU
- Internal Processing Unit (IPU), if configured
- Each I/O device

Symmetrical Multi-Processor (SMP) architecture takes advantage of this parallelism. The boot processor handles all non-real-time interrupts, vectoring all real-time interrupts to the designated real-time processor. This keeps LCRS CPU and LCRS IPU tasks bound to their own processors, shielded from unintended interrupts.

This segregation presents a virtual environment indistinguishable to the legacy system, permitting a high-fidelity replacement solution. The CPU/IPU biasing feature also functions in today's *multi-core* processors, effectively permitting a CPU/IPU configuration on one chip.

LCRS Software Program Structure

Figure 3 defines the high-level LCRS (virtual Concept/Series) programmatic structure. This program flow illustrates the software that generates a virtual Concept/Series computer, which runs a standard MPX-32 Operating System and legacy applications. The MPX-32 O/S and associated documentation remains exactly the same as on today's Concept/Series computer system.

The LCRS software reads the "hardware" configuration file to determine all Concept/Series attributes required for this system. It then performs a general system initialization, loads Linux driver code and starts the system threads. Threads include all console and control panel interface functions and "wait/no-wait" I/O interfaces.

The LCRS code then enters the Main Execution Loop performing instruction fetch, instruction decoding and vectoring. It concurrently processes any machine traps, I/O device interrupts and RTOM interrupts. Just as in Concept/Series hardware, LCRS software executes machine instructions, fetches operands, and performs required execute and store operations. The process repeats by branching to the beginning of the main execution loop.

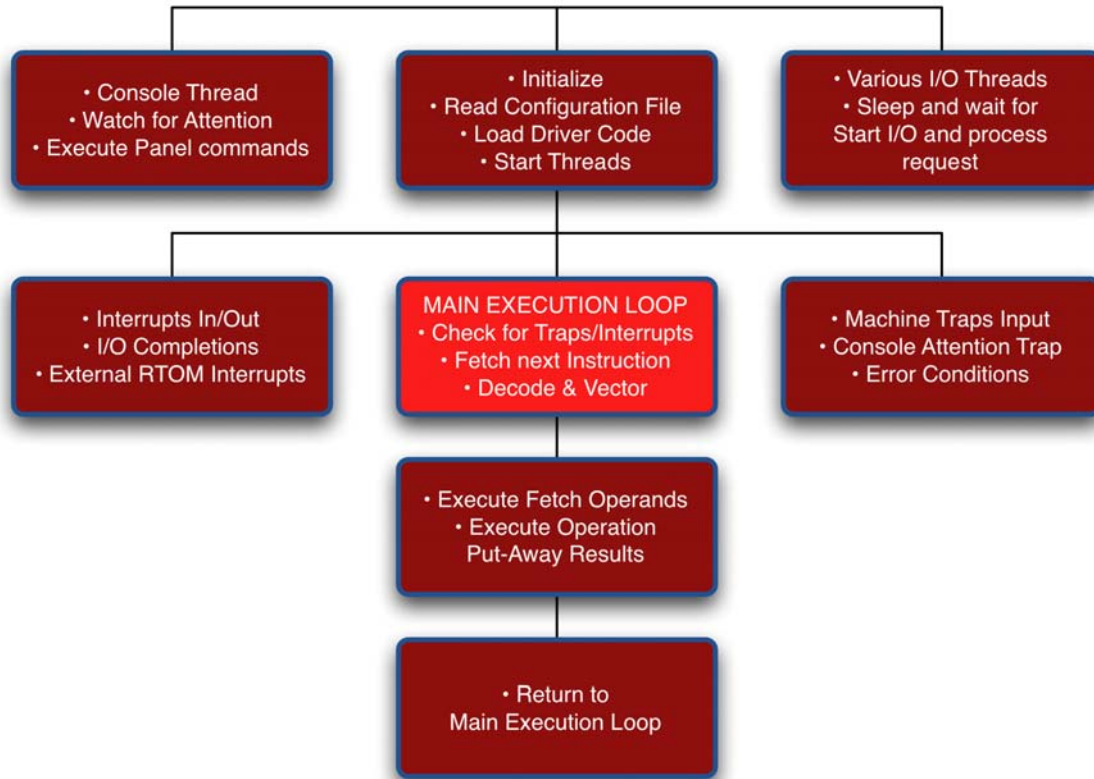


Figure 3. LCRS Program Structure

Basic System Peripherals

The MPX-32 system and data disks are actually Linux files in LCRS. The “Linux file disk” is partitioned with 768 blocks per sector, like the Concept/Series SMD disk. Disk data is accessed through the High-Speed Disk Processor (HSDP) handler, exactly like the Concept/Series MPX-32 system.

The 9-track ½" tape and associated Low-Speed Tape Processor (LSTP) is replaced with a state-of-the-art Digital Audio Tape (DAT) drive operating in the Level-2 Linux environment. The MPX-32 handler still issues tape commands and retrieves device status, however the physical tape device is the Linux-based DAT. The system can execute a Boot SDT from a DAT and can also generate a User SDT to a DAT. The DAT is also used for system backups in both the Linux and MPX environments. If desired, a SCSI-based 9-track tape drive is available, providing media compatibility with existing libraries.

A Linux console port replaces the Concept/Series Console terminal and the Linux PCI-based RTOM replaces the SelBUS based RTOM. However, all interfaces to these devices remain via the MPX-32 Console and RTOM handlers. MPX-32 memory space is a Linux shared memory partition accessible from both domains.

Standard Legacy System Peripherals – Update Considerations

Standard legacy system peripherals, including hard disk and magnetic tape drives, use proprietary SelBUS controllers. LCRS configurations substitute these SelBUS controllers with COTS controllers, using modern SATA or SCSI based peripherals.

In some rare circumstances, the legacy MPX-32 operating system may require an update “patch” to operate under LCRS using SATA or SCSI based COTS peripherals. In contrast, the customer application code will typically NOT require any modification to operate with modern COTS peripherals; the application still “sees” legacy tape or disk devices, while enjoying COTS peripherals’ improved performance.

I/O Subsystem Peripherals

Some of the PCI-based I/O devices that run under Linux, controlled by MPX-32 handlers executing in the LCRS environment, include:

- High Speed Device (HSD) interface
- Ethernet
- RS-232/422 interface
- Synchronous Communications Multiplexer (SCM)
- Asynchronous Data Set (ADS) interface
- SCSI
- 2345 Real Time Option Module (RTOM)
- Reflective Memory System (RMS)

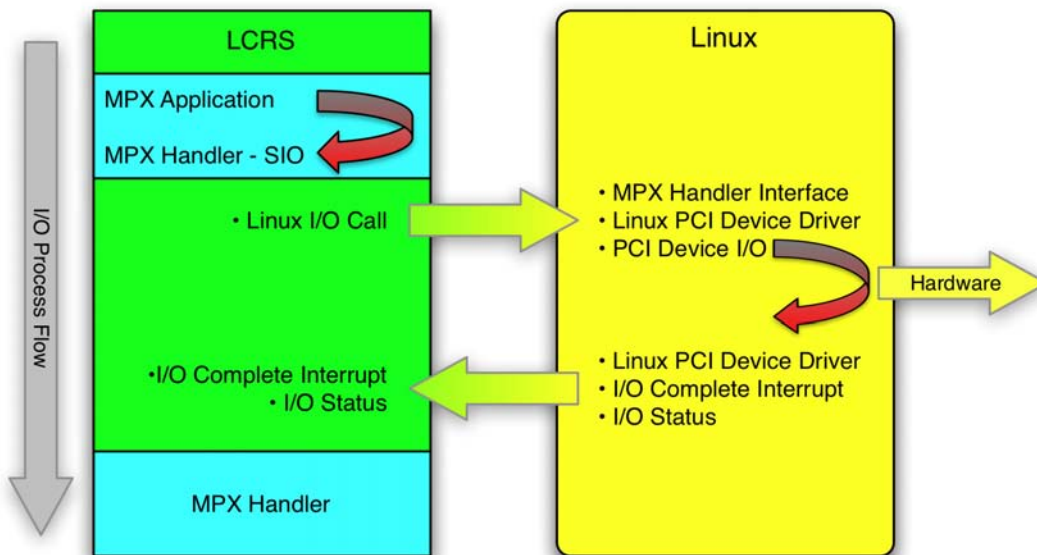


Figure 4. LCRS/MPX I/O to Linux Device

Figure 4 illustrates an MPX application running in LCRS (virtual Concept/Series system). The application calls the MPX handler to do a “Start I/O.” LCRS then performs a Linux I/O call to the Linux-domain MPX handler interface, which commands the Linux-PCI driver to perform standard PCI-device I/O.

The Linux-PCI device driver captures “I/O complete” and “status,” and passes the information back to the Linux driver interface in LCRS. The information is forwarded to the MPX handler and is passed to the application. Using this method, MPX-32 and the application software never “know” they are communicating outside the virtual Concept/Series environment.

SOFTWARE TOOLS

Distributed Real Time eXtensions (DRTX) for Development Systems

Designed to facilitate an excellent development capability, DRTX links the LCRS/MPX domain to the Linux domain using Linux (server) and MPX (client) software. Connectivity is established via Compro’s PCI Reflective Memory System (PCI-RMS).

The DRTX software environment:

- Provides bi-directional file transfer capability.
- Provides Terminal Services Manager (TSM) login, MPX Volume Manager access, etc.
- Executes or compiles MPX FORTRAN from Linux, returning results to the Linux system.
- Allows editing and source control under the Linux environment.

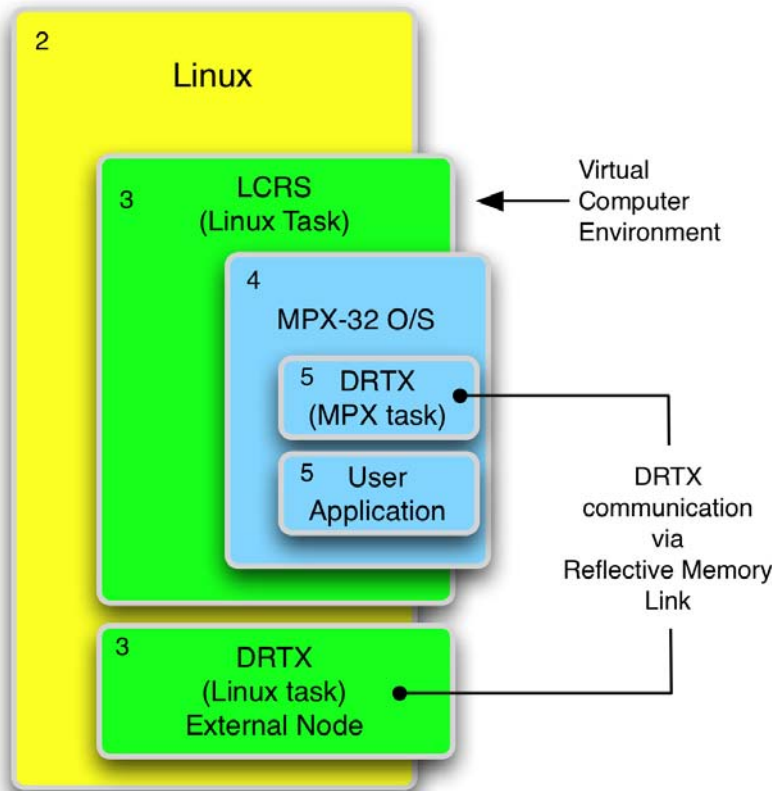


Figure 5. DRTX Functions

The DRTX software package includes interrelated components that execute on one or more Legacy Computer Replacement Systems (LCRS) running MPX, and one or more COTS processors running Linux applications.

DRTX permits MPX source code development, manipulation, storage, control, modification, and compilation in a Linux environment. This means MPX-like commands (such as TSM, Volume Manager, etc.) are available while in the Linux environment. For example, to perform an MPX-based “compile,” DRTX sends source to MPX (running in the LCRS) where it is compiled. Results are sent back to the Linux environment without user intervention. Communication between MPX/DRTX and Linux/DRTX domains may occur across legacy shared memory, or a PCI-based Reflective Memory System (PCI-RMS).

DRTX also supports file transfer, manipulation and management across both environments. This feature exploits modern Linux utilities, development tools, and file system architecture while retaining compatibility with legacy MPX file systems.

LCRS Utilities

LCRS includes several utilities useful for installation, debugging and maintenance. The following section provides a brief overview of some of the available utilities.

MPXDISC

The *mkdisc* utility creates and formats virtual disks (files) on the LCRS host system for use by the LCRS emulation.

MPXTAPE (Optional)

The *mpxtape* utility performs two main functions:

- Reads a physical (or virtual) MPX/LCRS VOLMGR or FILEMGR tape to the Linux file system.
- Writes a physical (or virtual) Linux-generated VOLMGR or FILEMGR tape to the MPX/LCRS file system.

Its primary purpose is moving source code from the MPX/LCRS environment to the Linux environment, and back. While in the Linux environment, source code modification and maintenance is available using the full compliment of modern Linux tools.

While *mpxtape* offers a simple method for manual (physical media) file transfer and data manipulation, it does not provide the advanced interactive features of DRTX. The *mpxtape* license is an optional LCRS component.

MPX_REGS and MPX_BREGS

These Linux command line utilities display the LCRS virtual machine CPU general purpose registers, and base registers. This information is handy when debugging applications.

MPXRSCH

This Linux command line tool searches the MPX O/S shared memory partition for a pattern within a 32-bit word. This is similar to the MPX O/S OPCON search command. *mpxsrch* is useful for physically validating memory activity.

LCRS Configuration File Directives

Invoked using a plain text file during LCRS startup, these directives offer useful diagnostic capabilities for isolating and resolving problems with application code execution in the LCRS environment.

BIXLOOP

Occasionally, programmers use an unfortunate technique for inserting delays into application code; instead of using system timers or counters, they use a “do loop” for a specified number of iterations. Although this works on the original platform, it fails when moving to any other hardware because each iteration depends upon instruction execution speed. This is especially problematic when source is unavailable, because detecting and correcting this problem in binary is difficult.

The BIXLOOP directive easily addresses this problem by detecting and delaying offending loops. The programmer simply specifies delay parameters, replicating the loop’s original duration. Most remarkably, *no other instructions are affected*. This means do-loops are correctable *without* overall performance penalty.

IDELAY

Another debugging aid, IDELAY offers a convenient method of slowing down the *entire* simulated LCRS environment. IDELAY adds a delay between every instruction fetch, providing a *performance* environment identical to the original legacy hardware. IDELAY is useful for top-level debugging, determining whether *any* timing issue is present in the original application code. IDELAY is used independently of BIXLOOP.

MANAGERIAL CONSIDERATIONS: ADDRESSING PROGRAM REQUIREMENTS

This section identifies and explores customers’ typical primary program requirement categories: **technical**, **logistical**, and **budgetary**. It presents each section with specific objectives, and offers recommendations for each.

Managerial Consideration – Technical

This section defines requirements involving sheer technical capabilities, including expandability, performance and peripherals.

Expandability

A primary motivation for replacing host computer systems generally originates from new simulation requirements (for example, Ground Proximity Warning System (GPWS), Traffic Collision Alert & Avoidance System (TCAS), Weather Radar (WRX), windshear, IOS, etc.).

These capabilities are available for existing simulators via:

- a) New simulation code
- or:
- b) New connected subsystems

However, each of these solutions poses potential problems. Existing simulations may not have enough spare frame time (headroom) or application memory space to accommodate these new features. Alternatively, new subsystems usually involve an interface protocol that perturbs the host's performance, and/or require complex simulation code modification and driver insertion. These problems are especially prevalent in older host computers.

Expandability Solution: ZI³

Zero Incremental Integration Impact (ZI³) is a concept made possible by Compro's PCI-RMS, described below. Essentially, ZI³ easily permits adding new features and functions to existing systems (referred herein as "host systems"), *without* affecting host system performance or operation.

Although Ethernet is a viable solution in commercial network environments, it is usually unattractive for real time applications. Ethernet is inherently non-deterministic; guaranteeing data packet frequency and latency is difficult. Ethernet also requires adding a driver that consumes CPU time. In a simulator host, this time may not be available, and may prevent normal simulation operation. Ethernet also lacks inter-system interrupt mechanisms often required in real time environments.

Because PCI-RMS operates autonomously by automatically reflecting pre-defined memory regions, add-on systems simply "see" data in their local memory. PCI-RMS uses a stand-alone engine that re-distributes memory data without using any host system CPU cycles. Connected systems simply "snoop" reflected memory via a "back-door" mechanism. With ZI³, new simulator functions like GPWS, TCAS, WRX, wind shear, and IOS are available without requiring host application recoding or affecting host performance.

One example of ZI³ in action is a nuclear power plant monitoring system. The system had been rigorously certified, and any new capabilities threatened a potentially costly recertification. The customer needed remote monitoring and control, and relied upon ZI³ for the solution. Using the PCI-RMS Bridge, Compro added a network of Windows PCs interconnected via PCI-RMS boards. Using the PCI-RMS API, the PC application simply identified memory regions in the plant monitoring system. All data appearing in those memory regions appeared in PC local memory, where a new PC application collected and analyzed the data, redistributing it to plant administration executives for real-time status.

For flight simulators, ZI³ can eliminate potentially costly FAA recertification when adding new system functions.

ZI³ Foundation: PCI Reflective Memory System

The foundation of ZI³, Compro's PCI-Reflective Memory System (PCI-RMS) is a distributed multi-node shared memory system, providing an extremely high speed, low latency communication backbone for interconnected computer systems running demanding high performance applications.

PCI-RMS Functional Description

PCI-RMS can support up to 255 systems requiring simultaneously access to the same data. This requires a PCI-RMS board to be installed in each system. Each PCI-RMS board is interconnected via thin co-axial copper (optional fiber optic) cable using the Fiber Channel physical-layer protocol in a ring topology. Each PCI-RMS board includes an external power supply, keeping the ring “alive” if one of the nodes fails.

Each PCI-RMS board identifies one or more memory regions to which any system can write and transparently reflect data into all interconnected PCI-RMS boards. Each system’s reflected memory region is defined as “shared data” available to applications.

Applications simply write to local memory and the PCI-RMS hardware automatically transmits the data to all connected computers’ local memory, with zero CPU overhead at extremely high speeds. Essentially, each cooperating system in the real time cluster has a duplicate copy of the shared data, eliminating all contention normally associated with conventional shared memory approaches.

PCI-RMS is compatible with computers running Linux, Windows, Solaris and Tru64 Linux operating systems with an available PCI slot.

PCI-RMS Error Detection and Interrupts

PCI-RMS implements an extensive set of error detection and recovery techniques assuring data integrity. These include Cyclic Redundancy Checks (CRC) on serial data frames, bus and parity error reporting, visual status indicators, and software-accessible status registers.

To validate data communication between connected PCI-RMS nodes, a hardware interrupt may be generated to report serial link or data transfer errors. Mailbox interrupts are available for asynchronous notification upon successful data arrival, including DMA transfer completion.

PCI-RMS in Real Time Applications

PCI-RMS eliminates non-deterministic operation caused by memory access contention, normally resulting in random program execution blocking. PCI-RMS consumes no CPU overhead for memory access because writes to memory are automatically distributed to other connected systems by hardware.

Memory reads are executed in local RAM eliminating memory access contention delays. Data reflection is automatic, concurrent and passive, requiring zero additional application software cycles to update all systems that share this common data.

PCI-RMS DMA “Memory Channel” Function

PCI-RMS includes built-in Direct Memory Access (DMA) and Memory Channel (Direct Serial Write) functions. Large data blocks (up to 224 MB) can be reflected to multiple connected computers by copying data from a source computer’s local memory to a reflected memory region. Data is then transparently moved to remote system(s) local memory using the Reflective Memory System (RMS) DMA engine.

The result is improved performance because:

- The remote system can now obtain data from *local* memory, which can be cached.
- The data transfer overhead associated with PCI to RMS latency is eliminated.

This capability frees processors to perform computation while I/O is offloaded to the PCI-RMS DMA engine.

Performance

LCRS addresses the performance (headroom) requirement handily. Sheer instruction execution speed benchmarks indicate at least 12x performance increase when replacing a 32/77 with LCRS. Conservatively, a 32/77 that uses 80% frame time will use less than 10% frame time under LCRS.

This provides dramatically more capability, permitting new code insertion without performance penalty. While specific performance improvement metrics cannot be predicted due to configuration variables, LCRS will almost always result in performance improvement.

Peripheral Upgrades

One problem with legacy hosts is legacy peripherals. Traditionally maintenance-intensive electro-mechanical devices, these often present support staff the most problems. Parts availability, training, reliability and performance pose escalating challenges.

By using LCRS, peripheral upgrade issues (current and future) are largely eliminated. Because LCRS “tricks” the application into “thinking” it is still working with legacy peripherals, new devices are easily added, now and in the future, without requiring application modification. LCRS simply re-maps legacy I/O functions to new controllers.

For example, old CDC-style SMD interfaces now speak SATA; Dataproducts printer interfaces map to Serial, Ethernet or High Speed Parallel supporting modern laser printers; Pertec magnetic tape interfaces now speak via SCSI to 9-track or DAT; proprietary Asynchronous Data Set (ADS), Synchronous Communications Multiplexer (SCM) or Asynchronous Interface (AI) communication controllers map to COTS serial PCI-I/O boards; physical punched card decks (via TLC card reader) are supported using virtual card files; and the list continues.

Ongoing LCRS updates assure compatibility with current and future peripherals, without affecting your legacy host application.

Managerial Consideration – Logistical

Major life cycle cost items include initial deployment, maintainability and configuration management. LCRS offers significant advantages over typical re-host solutions, as described below.

Deployment Time

LCRS is inherently rapidly deployable because existing subsystems remain in place, and existing application code remains intact. Typical installations require one week on-site preparation; one week installation; and one week certification, acceptance and basic orientation.

Most work occurs without interfering with normal operations. Compro engineers can schedule activities around production. Also, LCRS and legacy systems are easily switched via simple pre-installed A/B selectors. Legacy system operation is quickly resumed during any deployment phase, and A/B selection promotes quick LCRS/legacy operational comparison.

Maintainability

Because LCRS uses COTS hardware, maintainability is drastically improved. Systems include hot-swap disk drives, fans, and slide-out power supply modules. Controllers are now easily replaced PCI boards. Alternatively, entire spare systems can be pre-configured with internal PCI controllers, permitting entire rack-mount system replacement for fast mean-time-to-repair (MTTR).

Configuration Management

Because the same LCRS hardware can replace the entire SEL/Gould/Encore product line, a common configuration philosophy can apply to the entire replaced population. The only differences may occur in the quantity and type of PCI controllers, but the basic LCRS host remains the same.

With a greatly reduced spares population and highly modular components, traditional configuration management is greatly simplified. System tools are available that report internal hardware configuration data, even remotely. A central location can query remote systems, identify non-compliant items, and deploy updated hardware, firmware or software as needed. A total population configuration repository can reside at headquarters, permitting cost-effective and efficient updates.

Future Proofing

LCRS is largely platform independent. Compro has delivered LCRS on HP Alpha/Tru64, Sun SPARC/Solaris, Intel, and AMD (Linux) systems. Compro's experience migrating across several platforms underscores our ability to accommodate future platforms, while retaining LCRS' advantages.

LCRS processor hardware offers 64-bit CPUs. When future requirements demand expanded capability, typically via large physical RAM and increased floating-point precision, LCRS will be ready.

Managerial Consideration – Budgetary

Budgetary items, including environment, reliability, training and sparing factor heavily when deciding to re-host or replace. LCRS presents compelling arguments for a simplified replacement strategy.

Environment – LCRS is “Green”

Increasingly, environmental impact is an important component in any replacement strategy. Manufactured during eras of relative environmental dispassion, legacy SEL/Gould/Encore systems were large, loud, hot, power-hungry and anything but “green” by today's standards.

In contrast, LCRS can easily replace a room full of massive computer bulk with a rack of small, quiet, cool, power-efficient, environment-friendly nodes. The return-on-investment (ROI) savings through reduced HVAC, maintenance, space, and power consumption alone can justify LCRS within its product life cycle.

Reliability

COTS-based LCRS reliability far exceeds legacy equipment. Some items are so reliable that only failure projection algorithms can estimate product life; actual stress testing is often impractical because it would consume years. Old sparing paradigms can be relaxed when supporting LCRS because of this improved

mean-time-between-failure (MTBF). Whereas a 1:3 (spare:system) ratio was once common for mission critical sparing, 1:5 or greater is now practical.

LCRS hardware is also far more modular than legacy systems'. The lowest-replaceable-unit (LRU) is now an entire peripheral or subsystem. Even an entire LCRS node is economically spared as an LRU. LRUs are manufactured in clean rooms and high-precision factories. Higher LRU level means less field-intrusion, thereby increasing reliability.

Training

Typical legacy training classes, available only from the original equipment manufacturer (OEM) could last weeks or months. Proprietary hardware and software posed unique technical challenges. Technician attrition meant losing valuable "brain trust" resources, resulting in unacceptable risk during system failures.

Because LCRS relies largely on COTS, most technicians will already have familiarity with the hardware. Standard PC concepts (PCI, BIOS, SATA, etc.) apply to LCRS, simplifying conceptual understanding. LCRS is also highly modular, so isolating problems to LRUs is quick and simple. Built-in "health-check" diagnostics (power-on-self-test, or POST) quickly identify and isolate problems.

With respect to proprietary LCRS hardware elements (PCI-RTOM, PCI-HSD, PCI-RMS), these items include complete diagnostics, and are easily replaced. Classes are also available.

Compro offers comprehensive training for LCRS software, including installation, configuration and system administration. In many cases, training is sufficient for a few key personnel, who then re-disseminate their knowledge to fellow engineers.

Sparing

LCRS is very modular, using a building-block approach for configuring a variety of systems. A small spares complement (host PC, PCI-RTOM, PCI-HSD, PCI-8LAS, PCI-Ethernet) can support a vast replacement population.

Some completely different spares may still retain functional compatibility. For example, SATA disk drives, although available in several sizes, keep a common interface. This qualifies inevitably higher-capacity drives as viable spares for older LCRS systems.

Long-Term Supportability

Commercial "commodity" computing products (that is, COTS) provide many technological benefits, but rapidly change specifications. Typical government programs, with decades-long lifecycles and upgrade/spares purchases distributed over several years, may find it impossible to obtain products matching original configuration specifications. This complicates configuration management, creates software incompatibilities, and may make repairs impossible.

Fortunately, Compro understands these challenges, and has devised programs that eliminate the COTS disadvantage: TOPS and GLTS.

Technology Obsolescence Protection Service (TOPS) Option

TOPS guarantees that all major system subassemblies will remain supportable as long as the systems are supported under a TOPS-enhanced maintenance contract with Compro. Here's how TOPS works:

- Systems are purchased from Compro including an annual fee-per-node TOPS contract.
- When any major system assembly is designated "end-of-life" by the original equipment manufacturer, and spare parts are in demonstrably short supply, Compro will identify a freely interchangeable and readily available (that is, "fungible") replacement.
- For a small fee, Compro will deliver the upgraded replacement system assembly upon a mutually agreeable schedule.
- At no additional charge, Compro will provide remote integration services and support when necessary to assure compatibility with the customer's application and existing 3rd party I/O.

Compro is the final designator of assembly obsolescence. The TOPS option may be added to any standard Compro remote or on-site maintenance plan. TOPS is available only with contiguous service, and any service lapse may invalidate TOPS availability.

Guaranteed Long-Term Support (GLTS) Option

In addition to TOPS, Compro offers the Guaranteed Long-Term Support (GLTS) option for three to ten years. To qualify, simply pre-pay a TOPS-enhanced maintenance program for a period of three years (minimum) to ten years (maximum), and Compro will procure and warehouse spare parts based upon engineering analysis or available manufacturer MTBF statistics. This assures parts availability well into typical program life cycles.

GLTS works in conjunction with TOPS, meaning that any TOPS-qualified end-of-life replacements will also be procured and warehoused for the duration of the GLTS period.

SECURITY CONSIDERATIONS FOR MILITARY INSTALLATIONS

Modern computer systems used in U.S. armed forces' environments must meet Information Assurance (IA) security requirements. To meet this demand, Compro offers the Secure Legacy Computer Replacement System option (S-LCRS) that can be added to any existing or future LCRS configuration.

S-LCRS uses a separate single-point-of access Secure Linux Gateway (SLG) computer system running an IA approved version of Linux. This SLG connects to one or more Embedded LCRS/Application Processors (ELAPs) that execute legacy SEL/Gould/Encore application code. All console traffic to the ELAP is via the SLG; therefore all system access is captured, logged, and archived.

The ELAP executes a combination of customized Linux kernel, LCRS application, legacy MPX-32 operating system, and legacy binary application code. Because of the custom kernel, the ELAP is truly an "embedded" solution.

Figure 6 illustrates the Secure LCRS functional concept.

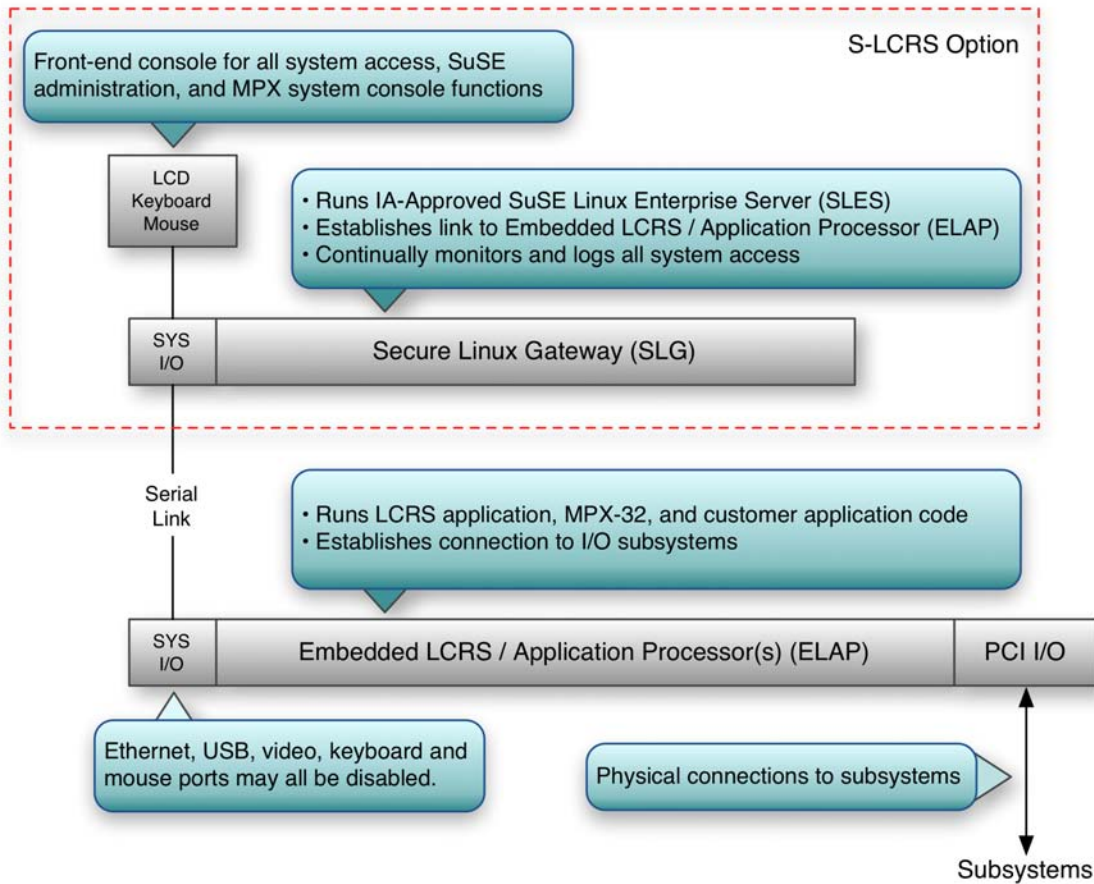


Figure 6. Secure LCRS Option

SUMMARY

The most important factors when considering system replacement are risk and cost. LCRS virtually eliminates risk by preserving known-functional application code, and reduces cost by eliminating application software re-hosting. In addition, LCRS permits simultaneous venturing into the native COTS world by providing a rich, deterministic real-time development environment that cooperatively executes with the LCRS environment.