EFTS Command Controller

From Concept to Operational System

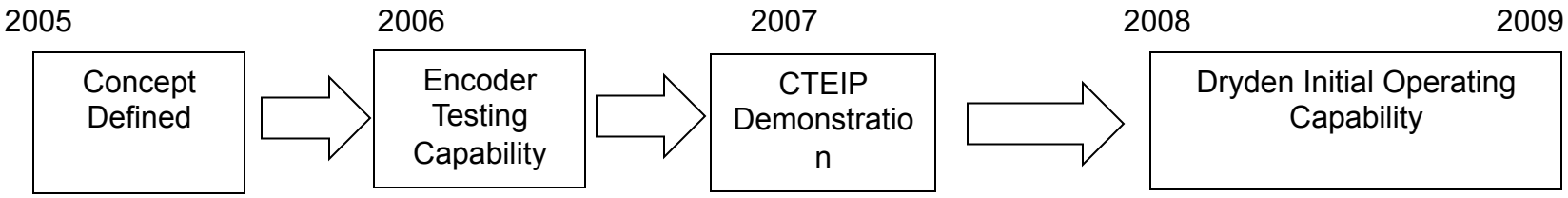
Dennis Arce

Bourne Technologies, Inc.

Scope of Brief

- Provide a short EFTS CC Introduction
- Describe the evolution of the CC and its functions
  - Phase 1: Prototype Capability
  - Phase 2: NASA Dryden Initial Operating Capability (IOC)

Development TimeLine



EFTS CC Prototype

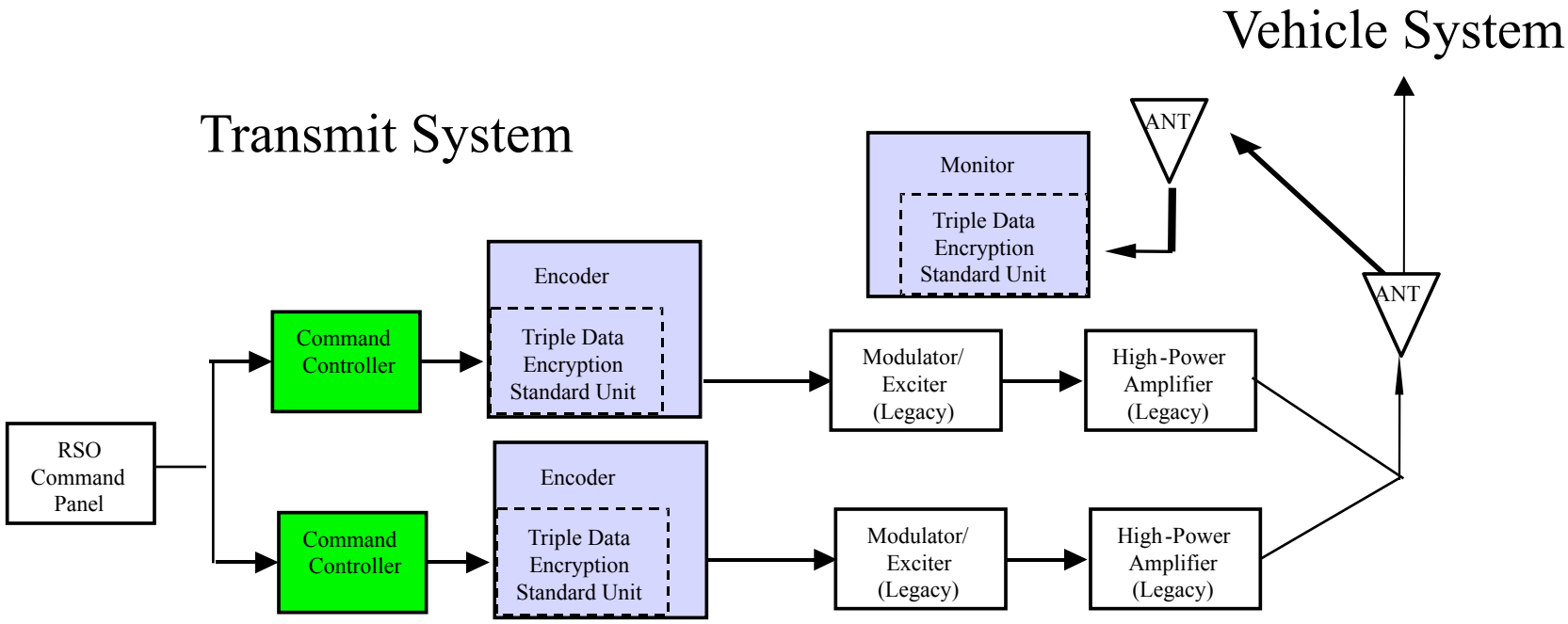
EFTS CC SVDI

SVDI= SINGLE VEHICLE DISCRETE INPUT

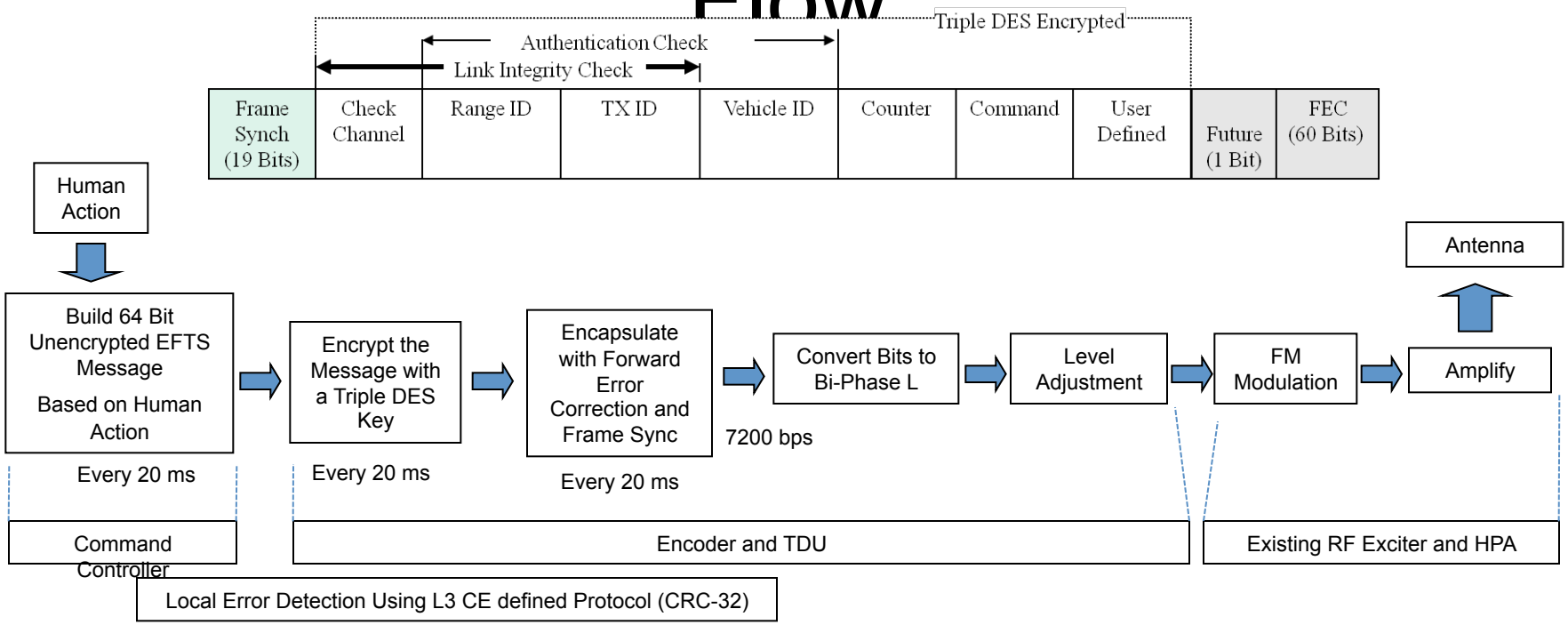
Current System Status

- NASA Dryden: EFTS CC SVDI's design has been validated against its requirements.
- EFTS CC SVDI has been integrated into the NASA Dryden infrastructure and has passed functional testing.
- EFTS System Wide Testing is being conducted at NASA Dryden in preparation for an EFTS Initial Operating Capability (IOC).
  - Testing Plan has been approved in concept
  - Pre-flight operations cards are being refined to include EFTS Mission types
  - EFTS System Testing Procedures have been developed and are being ironed out as part of a larger effort to include the EFTS system as a are being worked out.

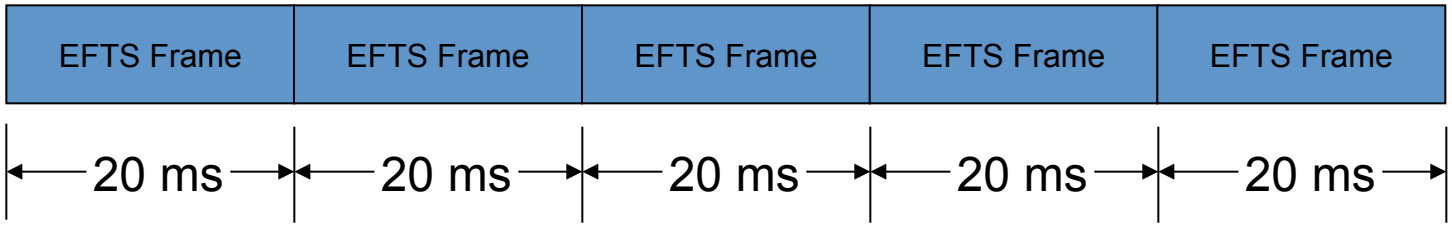
EFTS Transmit System Block Diagram



Simplified EFTS Transmit Logical Flow



EFTS Time Division Multiplexing



- Time Division Multiplexing
  - One Transmitter continuously transmits to all vehicles on one frequency
    - RF is continuously keyed.
    - Packets are continuously sent to keep FTR link synchronized.
    - If no command needs to be sent, a benign command (No Op) is sent.
  - Each Frame is a complete message that is authenticated at the receiver.
  - Frames are Continuously sent
    - Keeps Receiver Synchronized to the Ground
  - Each Packet:
    - Address a single vehicle for Commands (E.g. Arm)
    - All Vehicles for Link Integrity (Check)
  - No Provision for Multiple Simultaneous Transmitters at same Frequency
    - Make before Break or Break before Make must occur when geography mandates it.

L3 EFTS Encoder

- Developed by EFTS L3 CE as an operational product.
- Accepted and Qualified for General Use
- CSI Port for Configuration Software

L3 EFTS TDU

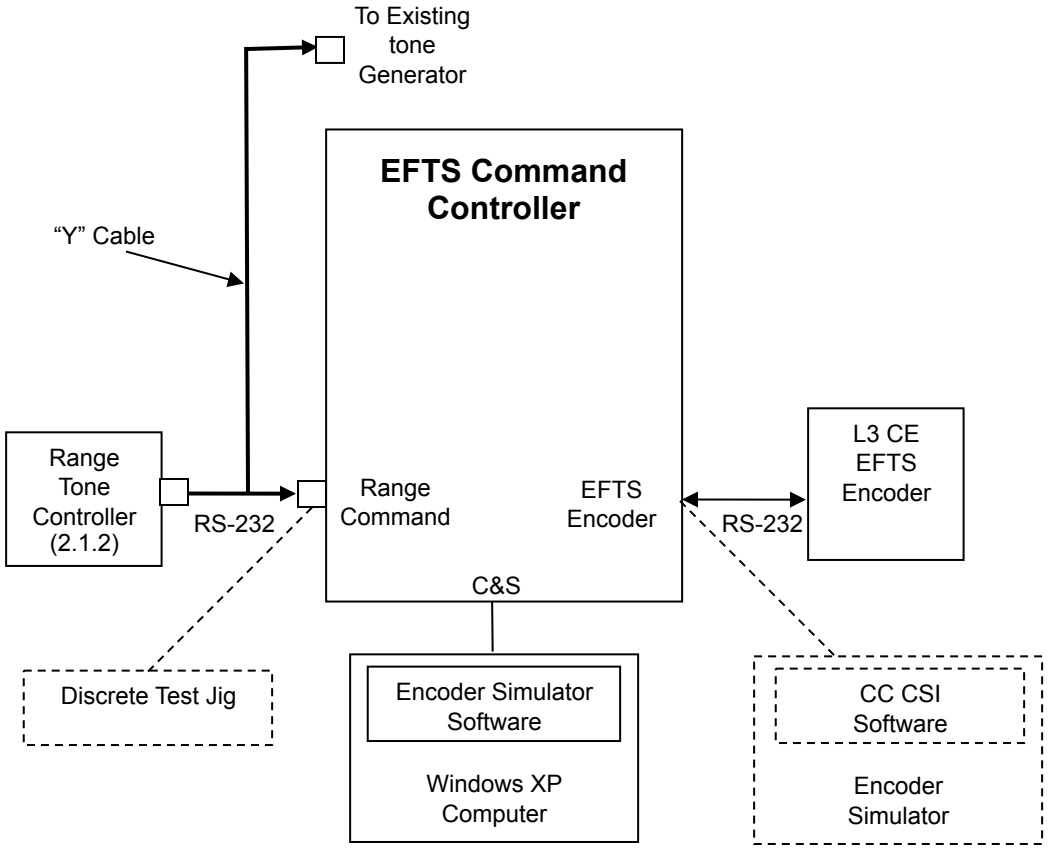
- TDU= Triple DES Unit
- DES= Data Encryption Standard per FIPS PUB 81
- Developed by EFTS L3 CE as an operational product for use with NSA Generated Key
- Controlled Cryptographic Device (CCI)
- Accepted and Qualified for General Use
- 50 Keys Each
- CSI Port Configuration Software through Encoder Interface
- Download Keys using DS-102, DTD

Development Phase 1
CTEIP Prototype

Reason for Prototype

- EFTS Program Developed several items for the EFTS system. The ground system interface for these device is the L3 CE EFTS Encoder/TDU combination.
  - Ranges decided to keep the development of the user interface portion for themselves to develop
    - Each Range has a different way of doing business
      - Test Ranges Typically test one vehicle at a time
      - Training Ranges may test multiple vehicles, or one at a time
      - Launch Ranges focus on single vehicle with wide area applications
  - EFTS Program needed to develop a device that would test the L3 CE Encoder prior to accepting it.
    - Development needed to be concurrent with Encoder so that interface approach could be independently verified. Development was based on an ICD.
  - EFTS Program needed to develop a device that could be integrated into various ranges for a CTEIP demonstration
    - Selection of the demonstration test range (Eglin/Tyndall) did not come until after requirements had been defined and development was under way.
    - Design needed to be flexible.

EFTS CC Interconnection



Functional Allocation

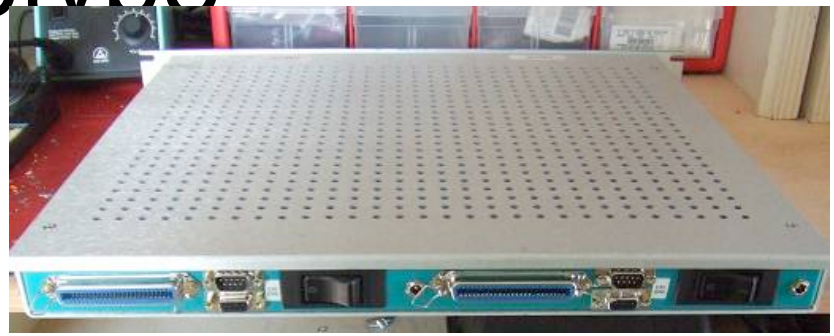
- **Command Controller-**
  - **Brief Description: Develops the 64-bit unencrypted EFTS message based on discrete inputs and hands message off to command controller every 20 ms using serial interface.**
  - RS-232 Control and Status Interface (CSI)
    - Allows Configuration of Mission Parameters (Range ID, Qty of Encoders and TXIDs, Qty of Vehicles and VIDs, Encoder Baud Rates)
    - Command Interface to Range Matrices (Which Tones equate to which commands).
    - Provides Error and Status Reporting (Encoder Link Errors, CC Errors)
    - Accessed through Windows GUI for Demo
    - Encoder Interface defintion
  - Interfaces with the Range I/O- Twenty Discrete Inputs
  - Interfaces with Encoder- RS-232 using Encoder Req/Send protocol
  - Develops 64-bit EFTS Message based in real-time based on rules.
- **Encoder/TDU**
  - **Brief Description: Encrypts 64 bit message using internal TDU, and develops full 144 bit message. Can hand off message to remote encoder or output on local BB output.**
  - RS-232 CSI
    - Allows Architecture Configuration (Central vs. Remote Encoders, Baud Rates, Where TDU is contained)
    - Error and Status Reporting
    - Accessed through Windows GUI for Demo
  - Contains and Communicates with TDU
    - Provides Encryption at CE or any Remote based on Architecture
  - Message Router: provides single interface to Command Controller to drive up to 5 local or Encoders.
  - Interfaces with Command Controller using RS-232 Req/Send protocol
  - Baseband Output provides level adjustment and filtering to drive FM Modulators.

Selection of a platform

- Technical Need: Encoder Simulator and command Controller
- Initially developed windows based software
  - Few weeks of development indicated that it was not practical.
  - 1-2 ms timing was a problem (for me) on XP
  - If a problem with Windows occurred, could it be found?
  - Blue screen of death!
- Decided on dual approach
  - Embedded Device for Command Link
  - Windows based software for status and configuration.

EFTS Command Controller Prototype

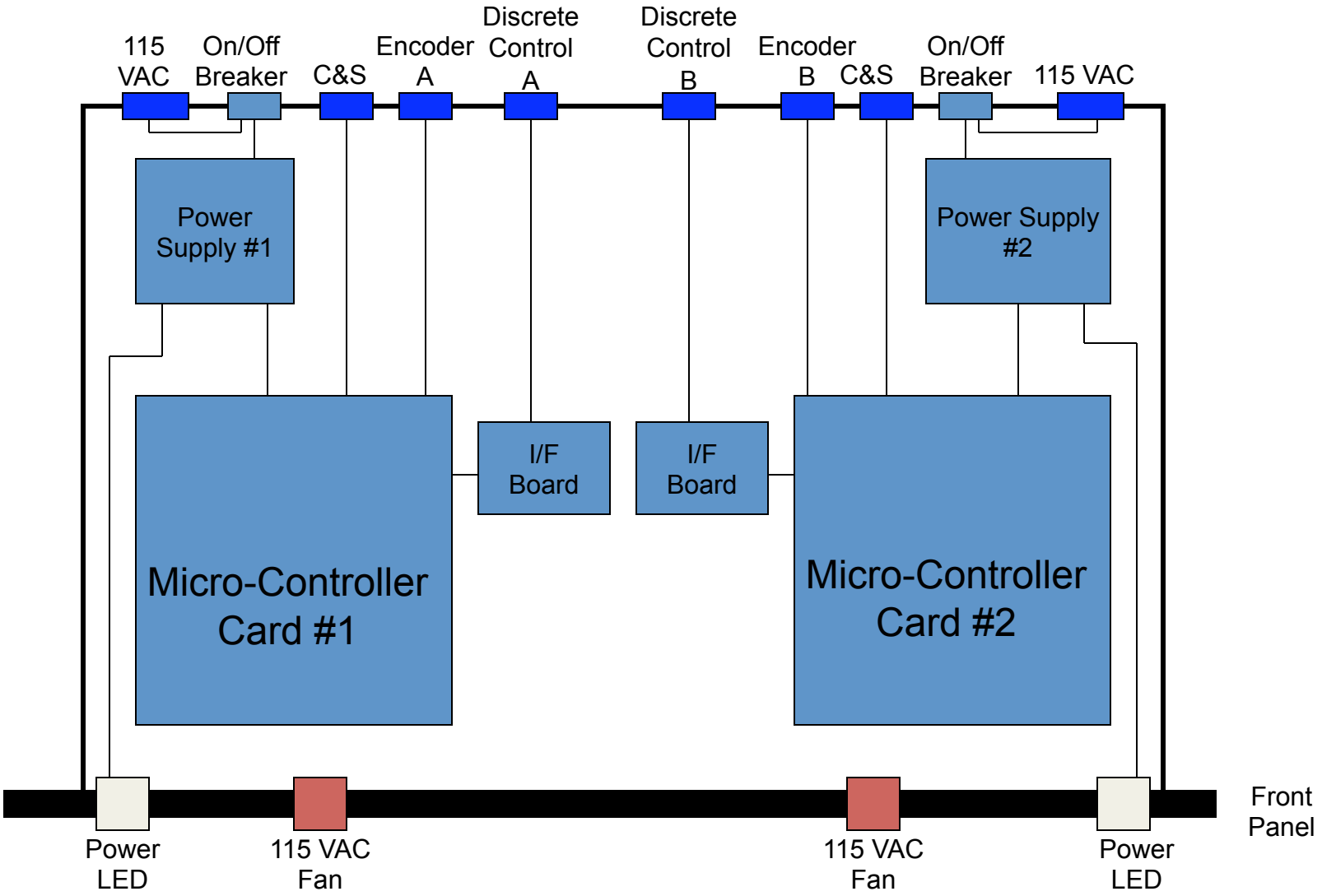
Front View



Rear View

- Developed by Bourne Technologies, Inc.
- 2 Units in 1 RU Chassis
- 115 VAC Input- Wall Unit
- CSI Port for Configuration Software

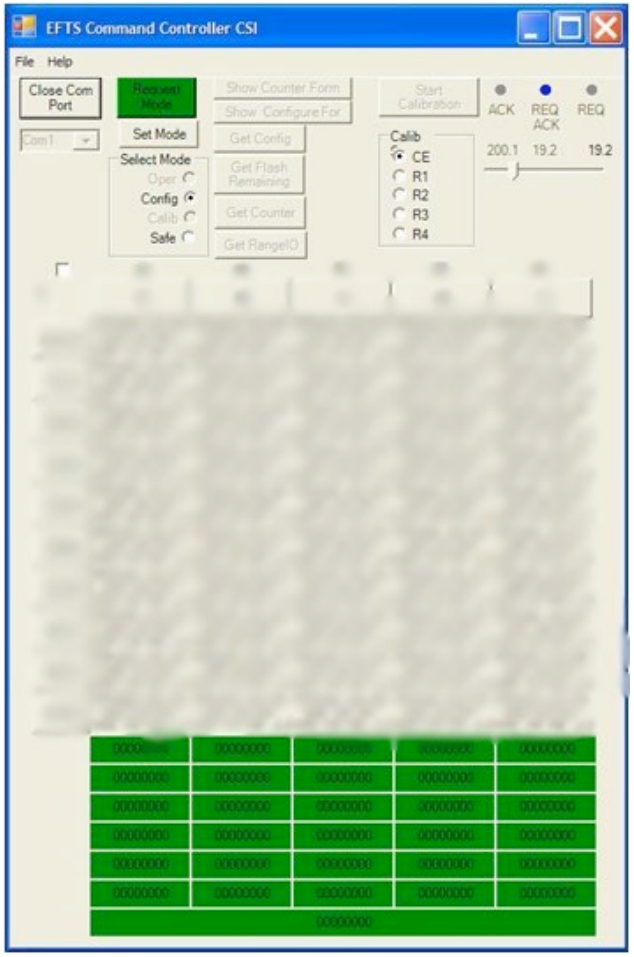
Command Controller Approach

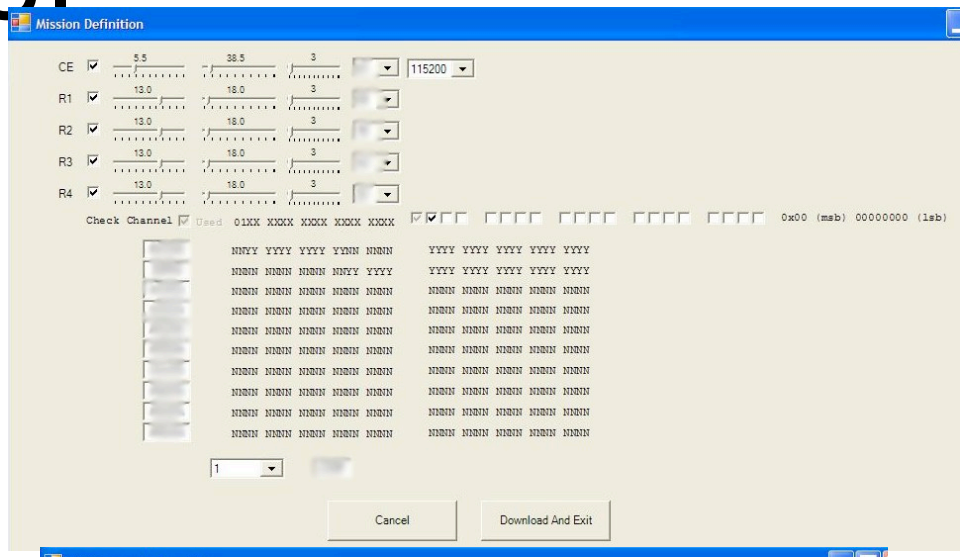


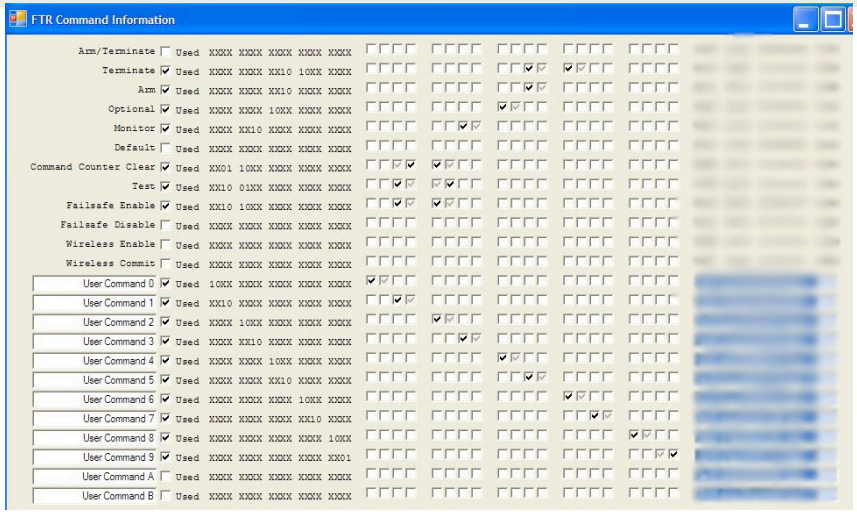
Micro Controller Card

- Use COTS Card
  - Atmel AT91SAM7S64-IAR.
  - Based on the Atmel AT91SAM7S64 microcontroller
  - Kit comes with a USB flash programmer
  - Kit comes with an evaluation version of a development system for this microcontroller
    - IAR Embedded Workbench, no royalties.
    - Evaluation of alternative development system was also briefly conducted (Keil uVision3)
    - others…

EFTS Command Controller GUI







Results of Prototype Development

- Unit was developed on Schedule and worked as designed
- Tested various configurations including
  - Single Vehicle Test
  - 2 vehicles test
  - Multiple-Sequenced Vehicle Test (1 at a time)
  - Command Priority Test
  - Arm/Terminate sequencing, failover, etc.
  - Various Captive Carry and 1 Live Fire Test
- Complete Success in support of Encoder Development and CTEIP-AMRAAM Demo

Phase II- NASA Dryden IOC

IOC= Initial Operating Capability

EFTS= Enhance Flight Termination System

CC= Command Controller

SVDI= Single Vehicle Discrete Input

Primary Purpose of CC SVDI

Command Activation

RSO @ BWB Panel

Command Translation

EFTS CC SVDI
(2 Per Chassis)

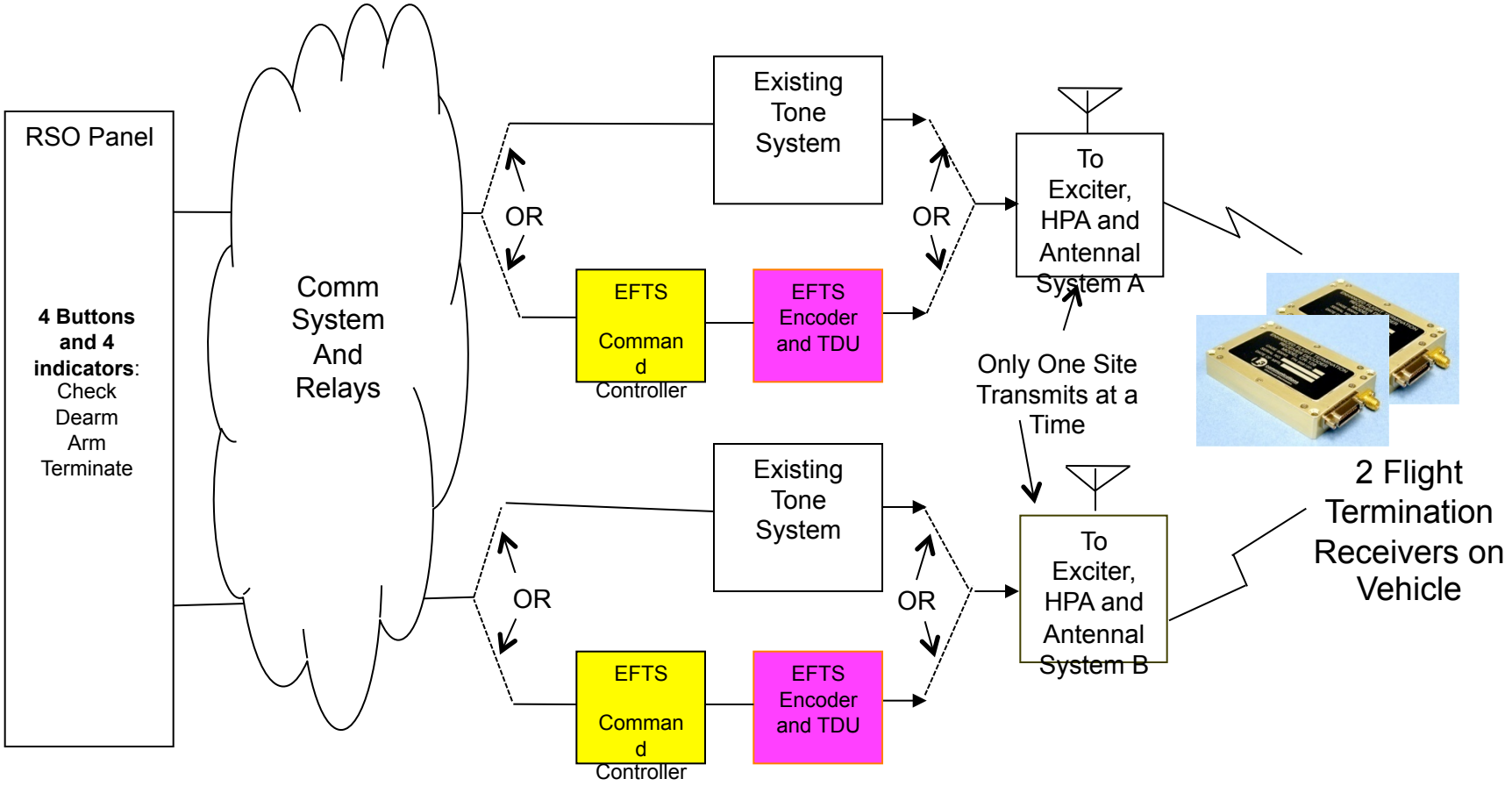
Command Encryption and Formatting

L3-CE Encoder with TDU

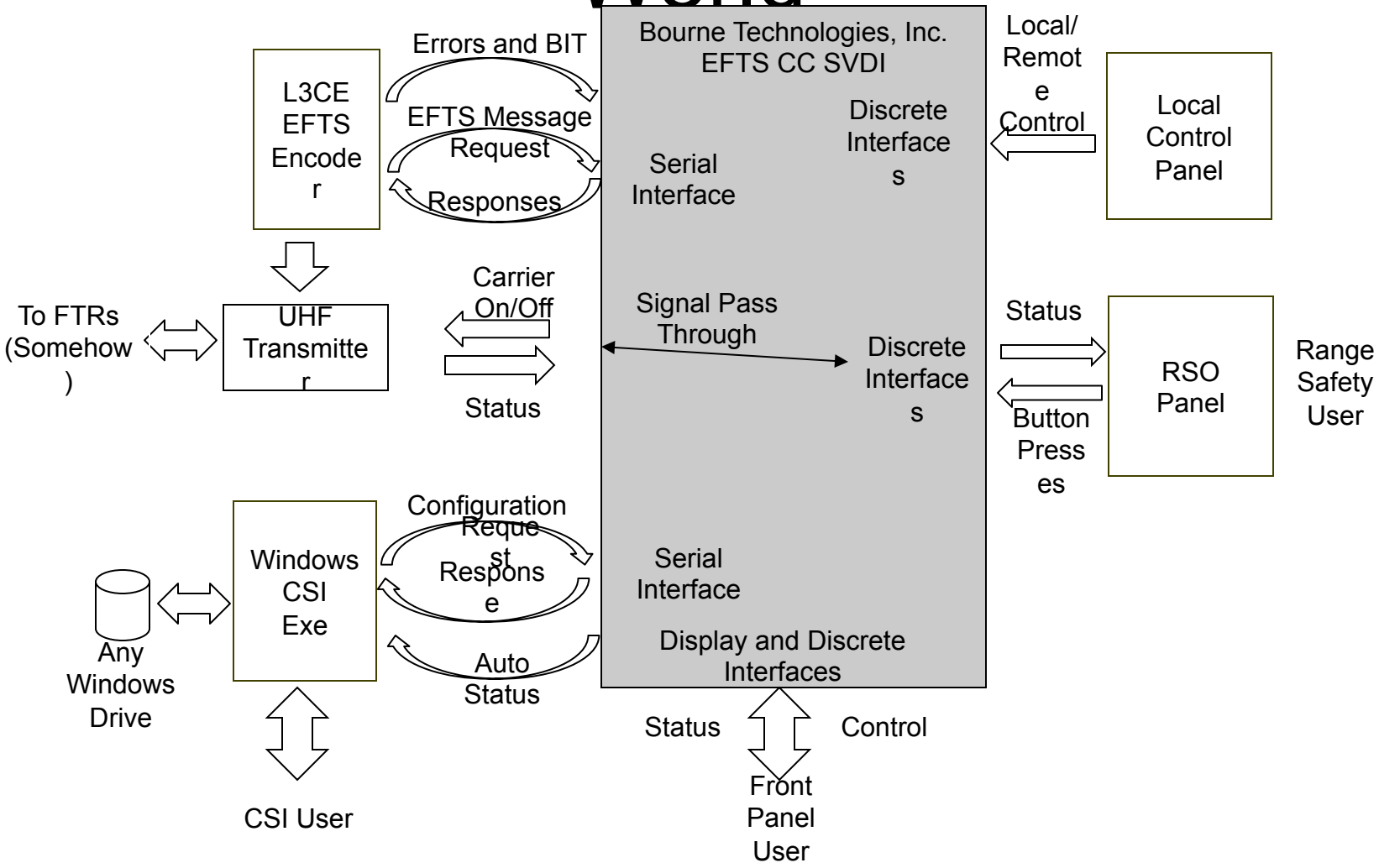
To FM
Exciter, HPA
and Antenna

- The CC SVDI is primarily a Command Translator-
  - Translates RSO's Button Presses to EFTS Message for Flight Termination
  - Interfaces with RSO Panel and L3-CE Encoder
    - RSO Interface: Command Input are dry relay contacts that provide +5V when button is being pressed
    - L3-CE Encoder interface is 3 Wire RS-232Serial 115.2kbps 8N1. Protocol defined by L3 CE. "See L3-CE Encoder IDC, Rev J, dated 31 March 2006"

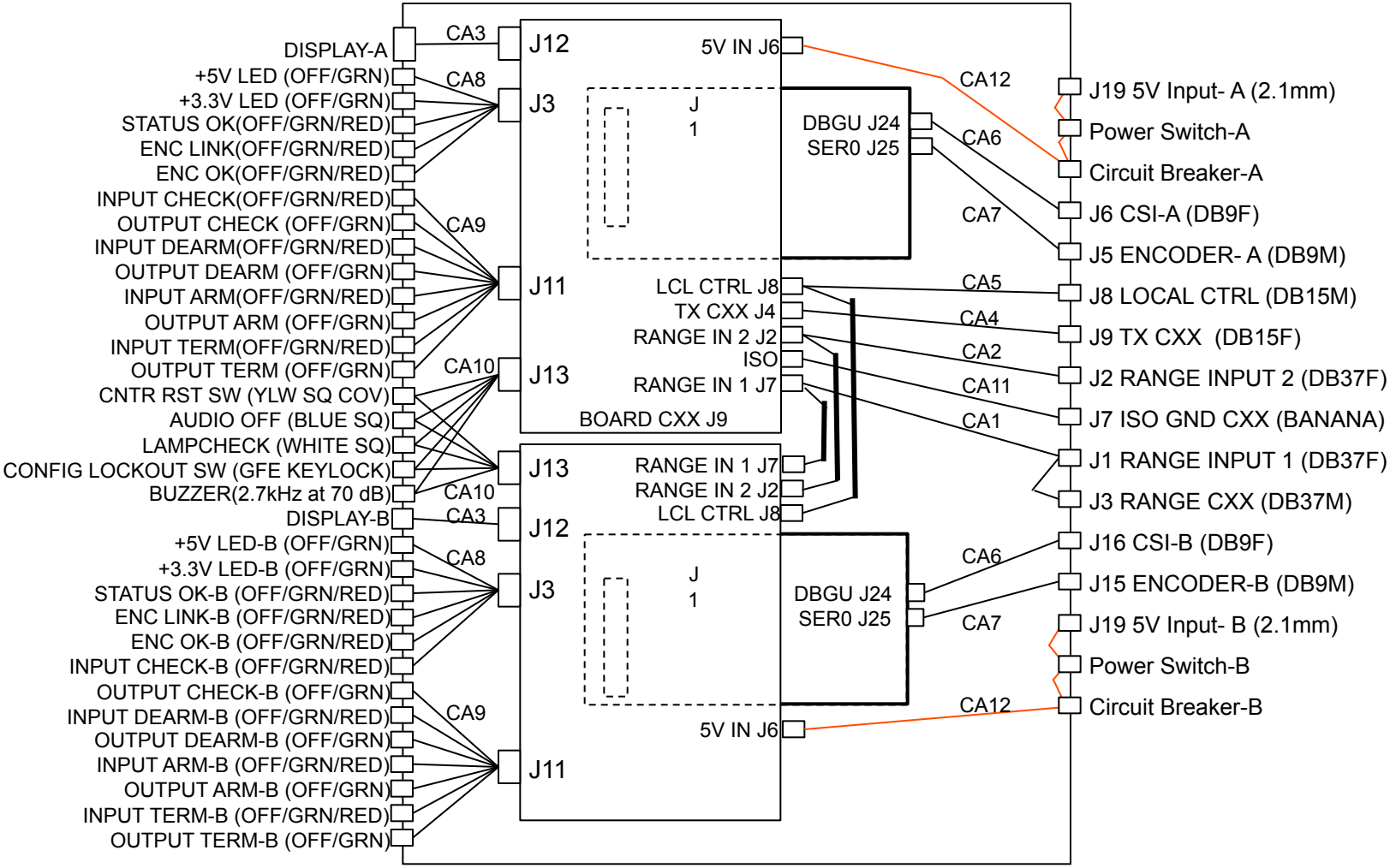
Simplified Dryden Transmit Architecture



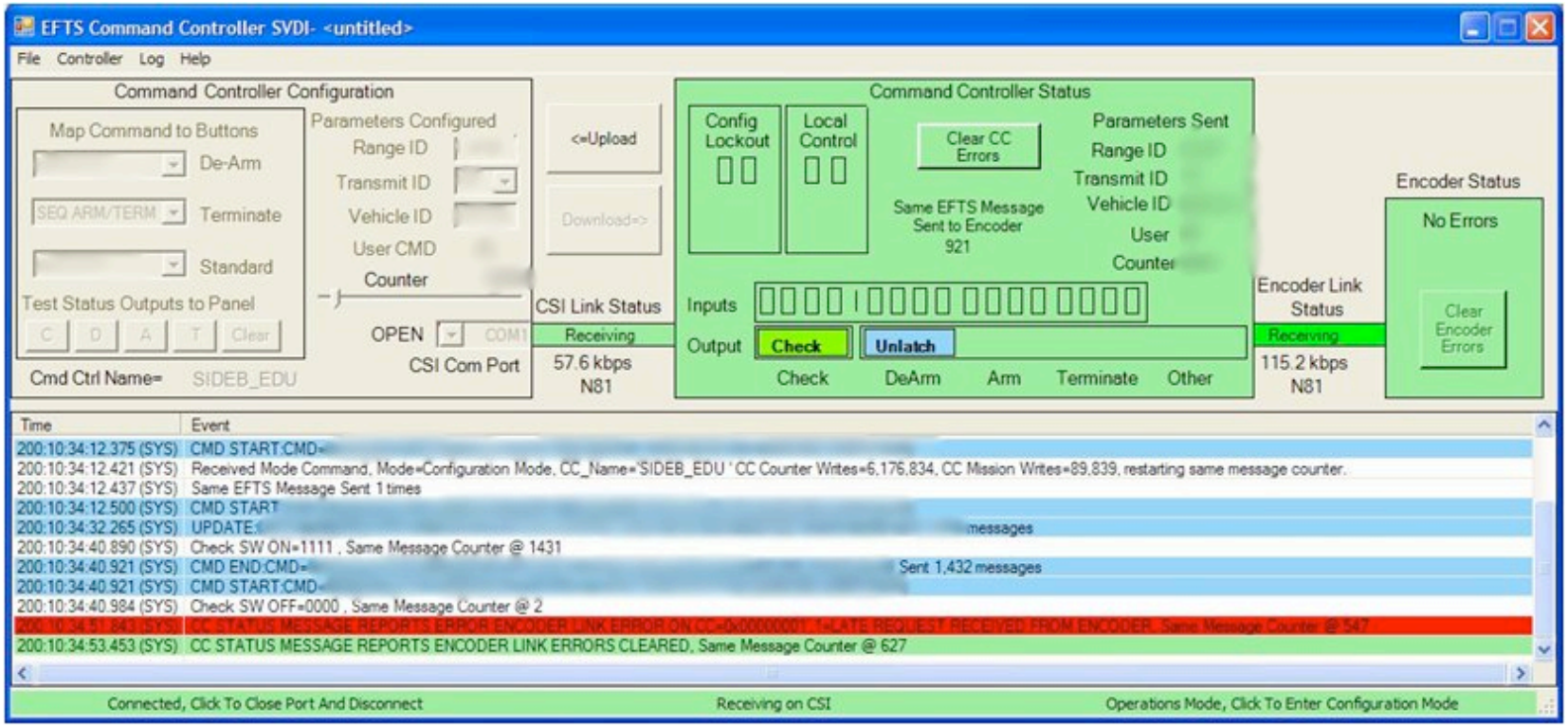
How Each CC SVDI Sees the World



EFTS CC SVDI Internal Design



Configuration and Status Software



Development Challenges

- Requirements Definition
  - Team was familiar with both EFTS and IRIG
  - Agreement on User Interface and Range Interface
  - Documented in an FPRD that was reviewed at SDR/PDR and finalized at CDR
- Technical
  - Not too many Technical Risks- Low Tech, High Reliability Approach
  - Atmel/ARM Processor Learning Curve- Simplified by Examples
  - Simple Opto-isolator interface to RSO Panel
  - Simple Serial Interfaces (RS-232 to Encoder and RS-232 to Control/Status Interface)
  - Uses “C” the premier language of embedded system
    - Subset to meet MISRA-C requirements for safety critical systems
  - All Static Memory Allocation- No dynamic memory allocated during operations
  - Small ARM based microcontroller at 18.432MHz can meet timing requirements.
  - No operating system means that programmer is in control!
  - ~5K Lines of embedded code (All independently reviewed)
- Verification and Validation
  - Hurdle of Validating EFTS CC SDVI has been passed
  - Hurdle of EFTS on Range is current step
  - Aided by :
    - Use of No Operating System, Use of MISRA “C”
    - Getting Agreement on what level of validation is necessary
  - Document! Document! Document!
  - Test! Test! Test!

Reviews and Documents

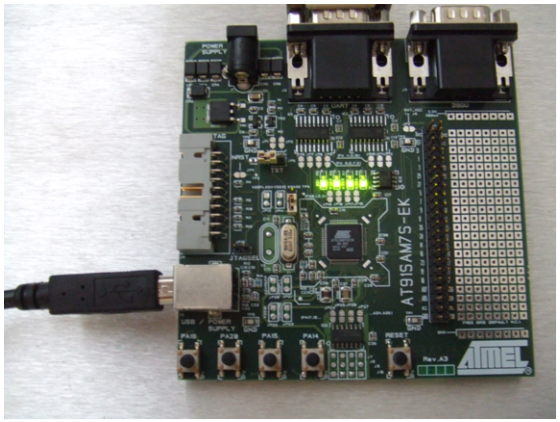
- External Reviews:
  - SDR/PDR: Requirements and initial design
  - CDR: Prototype was ready for production
  - Code Walkthrough
- Internal Reviews:
  - NASA set up a Review board to independently validate that the documentation and testing has been properly conducted.

- Documents:
  - Requirements for System
  - Functional and Performance Requirements Document (FPRD) for all EFTS components of system
  - Software Requirements Specification (SRS) for CC
  - Analysis of Alternatives (AoA) for System
  - Design Review Packages for all EFTS equipment
  - Training Guide for EFTS equipment
  - Software Design Document (SWDD) for CC
  - Software Hazard Analysis for CC
  - User's Guide for CC
  - System Verification Plan
  - System Verification Report
  - Reliability Analysis for System and CC
  - COMSEC Briefing (CSOP-13, dated 14 Feb 2007)
  - Acceptance test reports, user's guides for EFTS hardware
  - EFTS ConOps

Development Support Items

Range I/F Test Jig for Eglin/ Tyndall

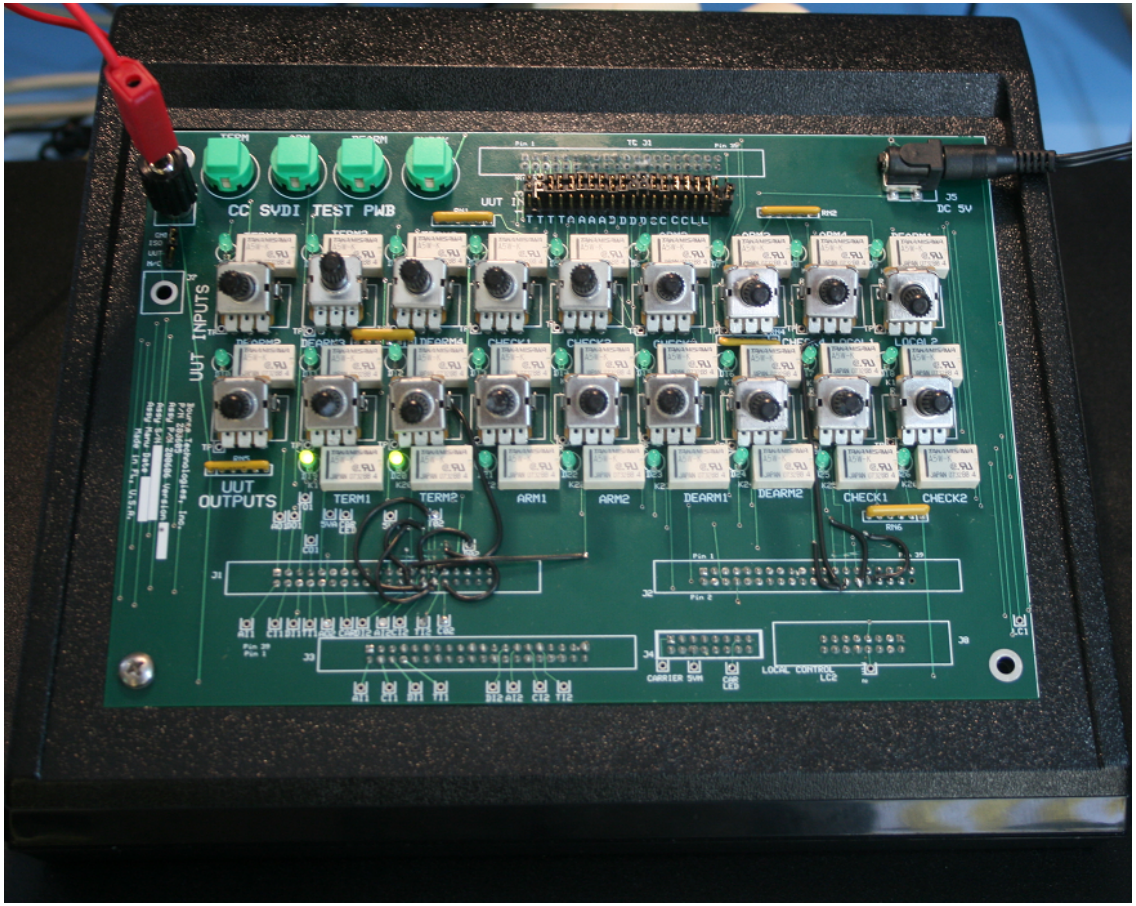
Encoder Simulator



Encoder Simulator
Based on
L3 CE Encoder
Interface Control
Document (ICD)

- **Simulates Electrical RS-232 CC interface of 2 L3 CE Encoders.**
  - **Timing is same**
- **Can Generator Errors that Real Encoder cannot generate without failure**
- **Used COTS Arm Card from Same Family as the Command Controller**
  - **ATMEL ARM PROCESSOR**

Range Interface Test Box



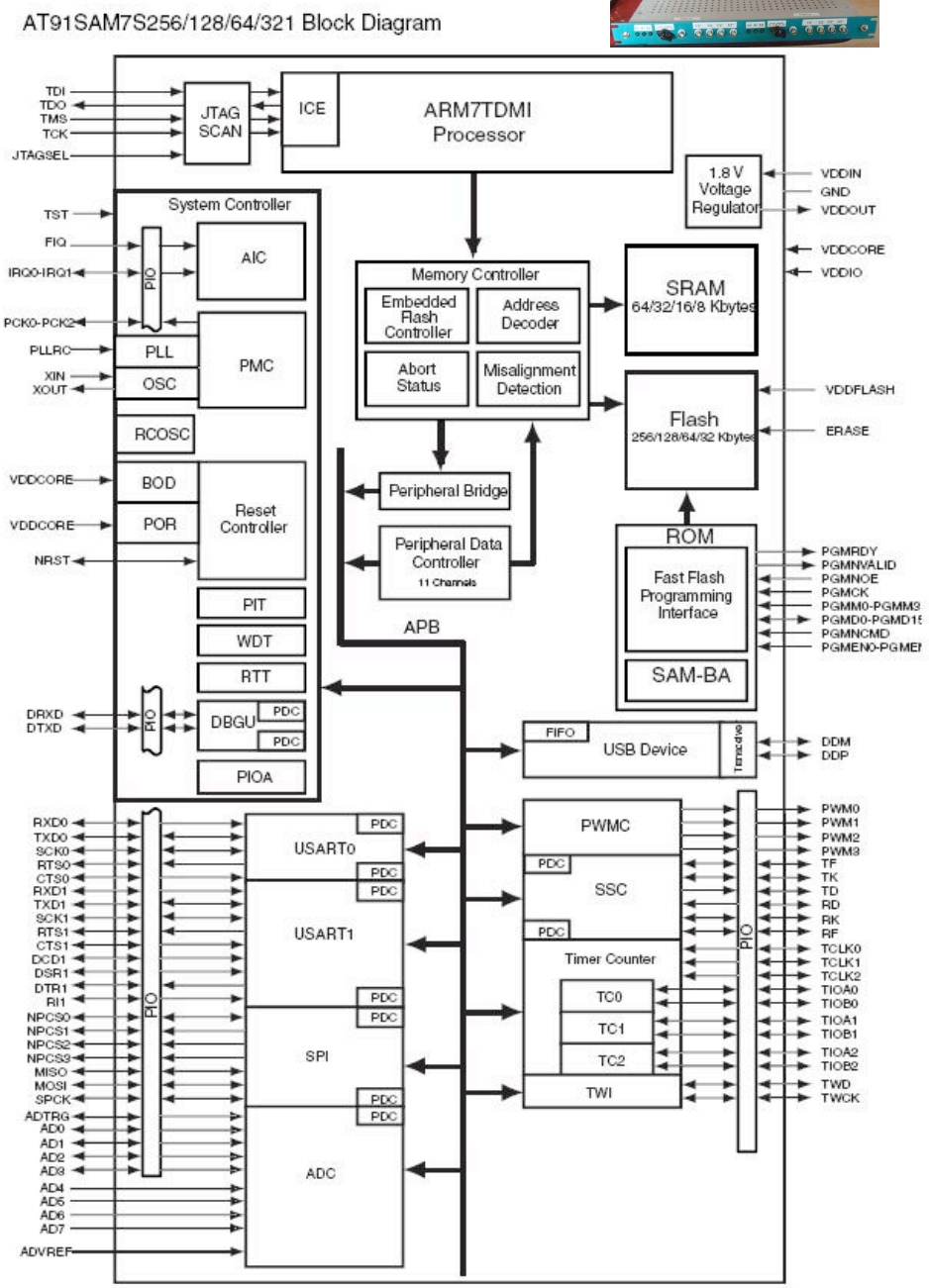
Backup Slides

What is ARM?

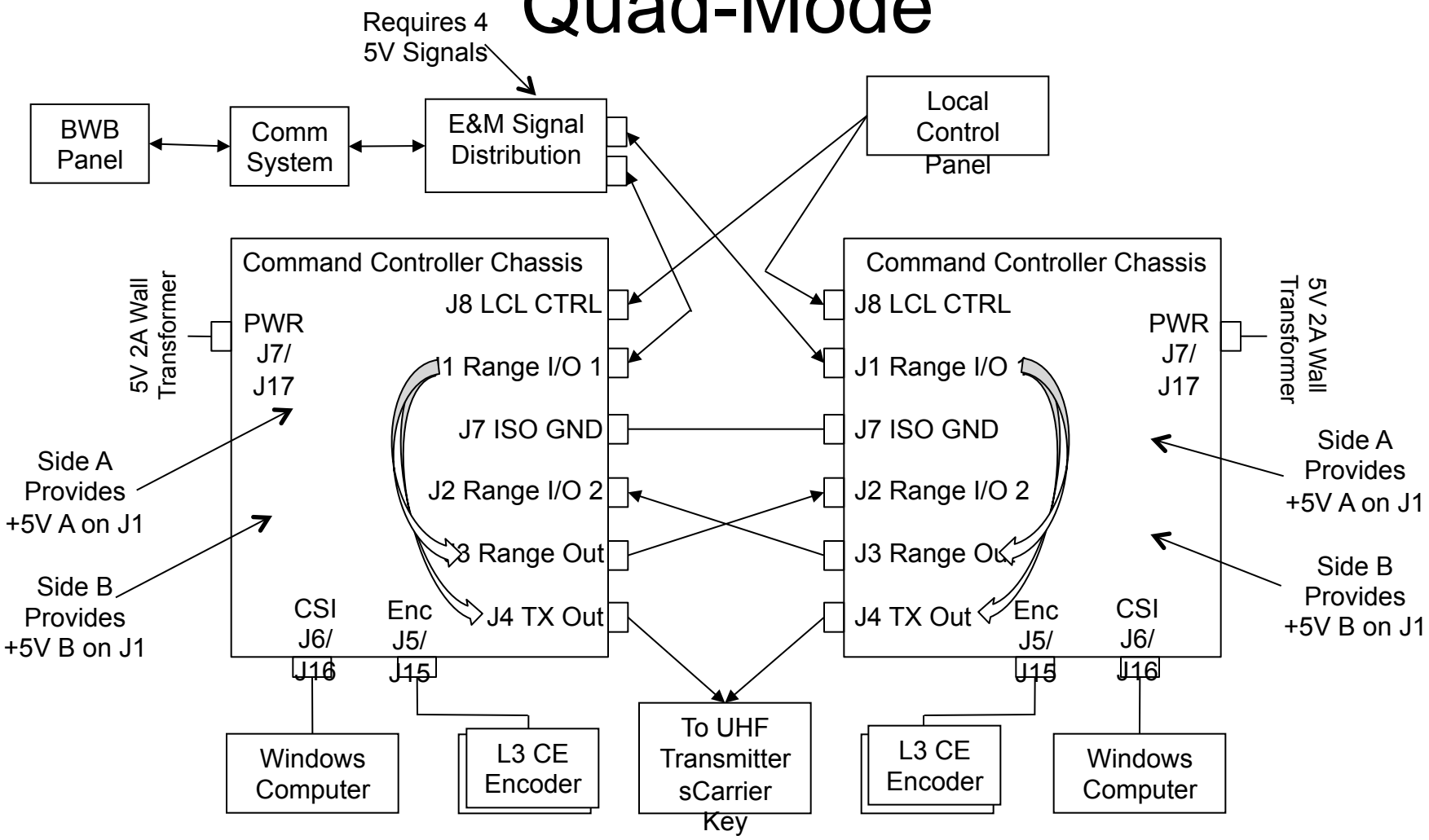
- ARM is a Reduced Instruction Set Computing (RISC) processor that is the becoming one of the more popular processors in the world.
  - The RISC architecture is widely used in embedded systems because of it's simplicity, predictability, and very low cost.
  - Smaller instructions often have data and logic embedded in them.
- The ARM company holds the rights to it's implementation, and sells intellectual property to chip manufacturers such as:
  - Intel
  - Atmel
  - Texas Instruments
  - RF MicroDevices
- ARM processors are very inexpensive compared to general purpose processors like the Intel x86
- ARM processors are used in the **iPod** from Apple, and the **Gameboy Advance** from Nintendo, and the soon to be released **Google** Android Netbook.
- Use of 32 bit processor over a 8 or 16 bit processor streamlined some calculations, including the 32-bit CRC that is calculated each 20 ms
- Many processors, including ARM, have no built in support for division
  - "C" software libraries are available for integer division
  - There is no division used in the application

MicroController

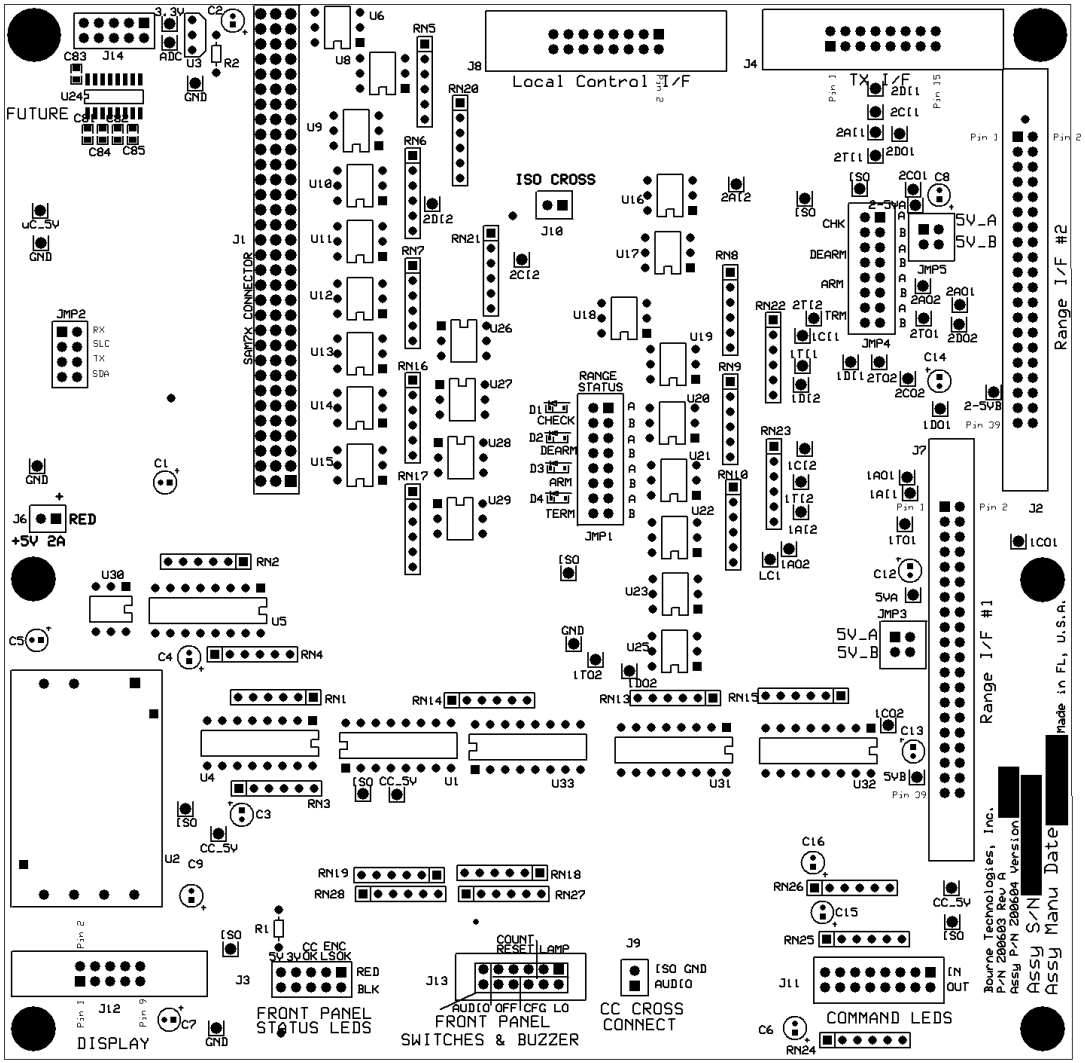
- AT91SAM7S64
- Arm7TDMI Processor
  - Built in ICE
  - Thumb Mode
  - 55 MHz
- Built in Flash
- Built in SRAM
- 2 USARTS for RS-232
  - Command Controller
  - Configuration Port
- Up to 32 Digital I/Os for Discrete Inputs
  - For OptoIsolated Control Input
- 11 Channel Peripheral DMA Controller with Memory Controller
  - Limits processor interrupt cycles required for transfers between Memory and I/O
- I²C (or Two Wire Interface) for EEPROM
  - For Command Counter Values
- Future
  - USB Device [alternate interface]
  - A/D Converters [Instrumentation]
  - Atmel family has more similar devices based on the ARM family and similar peripheral controls.



System Interconnect Diagram Quad-Mode



Interface Board Layout



Development Tools

- Embedded Software
  - Languages MISRA C and ASSEMBLY for ARM
  - Embedded Software- IAR EMBEDDED WORKBENCH for ARM
    - C Compiler
    - Assembler
    - Linker
- MS Visual Studio 2008 for CSI
  - C# and .NET 3.5SP1
  - Some Win API Calls for RS-232 Serial Port
- PCB Design
  - Express PCB Schematic
  - Express PCB Layout
- Chassis Panels
  - Front Panel Express
- Source and Document Control
  - Subversion (SVN) Server on Windows XP (Raid 5 System)
  - Tortoise SVN for Client
- Documents:
  - MS Word for Development and Production Documents
  - MS Powerpoint for Presentations
  - MS Excel for Parts Lists and Bill of Material
  - Any can be provided in ".pdf" form for deliverable
- Hardware Debugging
  - Logic Analyzer (HP 1672D)
  - Oscilloscope (Instek GOS-620)
  - Others (multimeter, adjustable power supply…)

Embedded Software development

- Software is written in “C” and “Assembly”
- Tool->IAR Embedded Workbench
  - Multi-File Editor and Source Object Browser
  - Project Builder and Linker
  - ARM “C” Compiler
  - ARM Assembler
  - Integrated JLink USB/JTAG debugger allows in circuit emulation to debug the processor.
  - Integrated Help
  - Optional MISRA “C” Compiler was used for CC SVDI
- Documentation
  - Atmel has extensive documentation on the product
  - ARM Processor is very well documented
  - IAR Support is excellent

Summary of Embedded Software Footprint

Latest Build Results:

- 23 460 bytes of readonly code memory (Onboard Code)
- 128 bytes of readwrite code memory (Onboard Code for CRC storage at factory only)
- 1 482 bytes of readonly data memory (Onboard Data Constants)
- 25 531 bytes of readwrite data memory (Onboard Data)

Estimated Code Size:

~ 5000 Lines of Code in .c files + header definitions.