# Avionics Modernization and the C-130J Software Factory

Richard Conn, Stephen Traub, and Steven Chung
*Lockheed Martin Aeronautics Company*

*The rollout of the first production C-130 aircraft, the C-130A, took place on March 10, 1955. Since then, more than 2,100 C-130s have been built in dozens of variations and are flown by more than 60 nations worldwide. They carry troops, vehicles, and armaments into battle. They drop paratroopers and supplies from the sky. They serve as airborne and ground refuelers. They serve as flying hospitals, hurricane hunters, and provide emergency evacuation and humanitarian relief. They perform airborne early warning and maritime surveillance. They've worn skis in Antarctica and have helped recover space capsules. In May 1992, the 2,000th C-130, a C-130H, was delivered. In September 1992, formal development of the C-130J began. Unlike its predecessors, the C-130J is a software intensive system employing modern avionics that have made significant improvements in its performance. By March 2001, the C-130J flew with a complete compliment of mission computer software setting 50 world records. This article presents insight into Lockheed Martin's modernization of the C-130 airlifter family.*

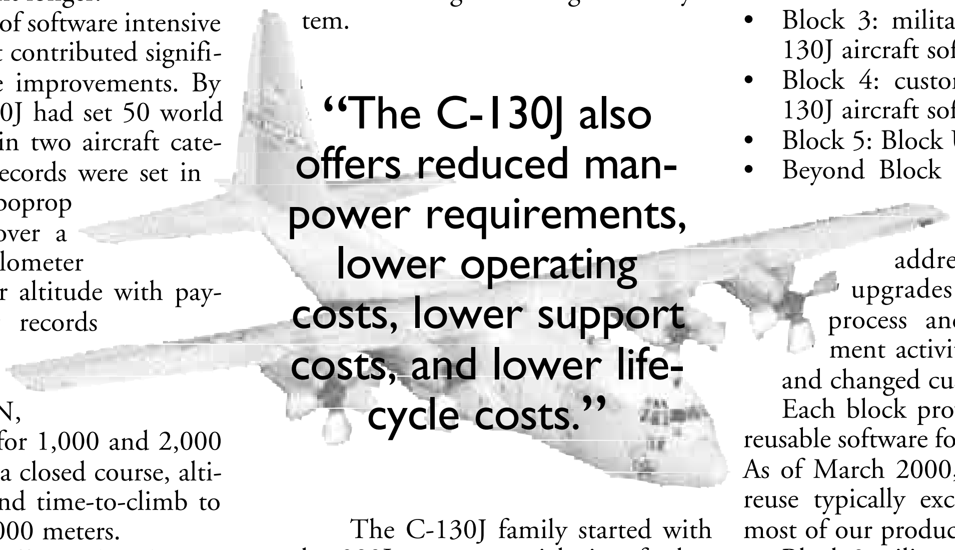
The C-130J looks like the earlier models, but it is really a brand new airplane with improved performance [1]. A key difference is that the C-130J is a software intensive system, where the earlier models were largely mechanical aircraft. Compared to the production C-130E, here are the C-130J improvements:

- Maximum speed is 21 percent greater.
- Climbing time is 50 percent less.
- Cruising altitude is 40 percent higher.
- Range is 40 percent longer.

The introduction of software intensive systems to the aircraft contributed significantly to all of these improvements. By June 1999, the C-130J had set 50 world aeronautical records in two aircraft categories. Twenty-one records were set in the Class C-1.N, Turboprop category for speed over a 1,000 and 2,000 kilometer closed course and for altitude with payload. The other 29 records were set in the Short Takeoff and Landing, Class N, Turboprop category for 1,000 and 2,000 kilometer speed over a closed course, altitude with payload, and time-to-climb to 3,000, 6,000, and 9,000 meters.

The C-130J also offers reduced manpower requirements, lower operating costs, lower support costs, and lower life-cycle costs. Here are the three key distinguishing features of the C-130J:

- A new propulsion system featuring four Full-Authority Digital Engine Control Allison AE2100D3 engines that generate 29 percent more thrust while increasing fuel efficiency by 15 percent.
- Advanced avionics technology featuring two holographic heads-up displays and four multifunctional heads-down Liquid Crystal Displays for aircraft flight control, onboard systems monitoring and control, and navigation; the displays are night vision imaging system compatible.
- Two mission computers and two backup bus interface units provide information flow and dual redundancy for the onboard systems, including an extensive integrated diagnostics system.



"The C-130J also offers reduced manpower requirements, lower operating costs, lower support costs, and lower life-cycle costs."

The C-130J family started with the 382J, a commercial aircraft that was created specifically to achieve Type Certification by the Federal Aviation Administration (FAA). FAA Type Certification was at Level A (the highest level) of the DO-178B standard. This milestone established that the C-130J family has complied with the safety critical requirements of the FAA should we later have a commercial customer. Once FAA Type Certification was achieved, the C-130J was derived from the 382J, establishing the military baseline software for all future variants of the aircraft. Each major version of software for the C-130J is called a block, and more than 96 percent of the 382J software (Block 2) was reused in creating the C-130J military baseline (Block 3). Ninety percent or more of the military baseline software (Block 3) has been reused so far for each variant of the aircraft (Block 4):

- Block 1: basic airworthiness software.
- Block 2: safety-critical 382J aircraft software.
- Block 3: military baseline of the C-130J aircraft software.
- Block 4: custom variants of the C-130J aircraft software.
- Block 5: Block Upgrade Program.
- Beyond Block 5: Hercules Improvement Plan for software/systems will address future C-130J upgrades as a continuous process and product impr-ovement activity and to address new and changed customer needs.

Each block provided a foundation of reusable software for the following blocks. As of March 2000, our level of software reuse typically exceeded 90 percent for most of our products:

- Block 3 military software baseline - 96 percent reused from Block 2.
- Block 4 software for the Royal Air Force - 95 percent reused from Block 3.
- Block 4 software for the Royal Australian Air Force - 95 percent reused from Block 3.
- Block 4 software for the United States Air Force - 97 percent reused from Block 3.
- Block 4 software for the Italian Air Force - 90 percent reused from Block 3.
- Block 4 software for the Tanker vari-

® *Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.*

ant - 90 percent reused from Block 3.
- Reuse on the C-27J aircraft, the C-5 Aircraft Modernization Program, and proposed for the C-130 Aircraft Modernization Program is yet to be measured but is expected to be equally high.

The first flight of the C-130J was April 1996 with a minimum of onboard software. The C-130J flew with a complete mission computer software suite (Block 5.3) in March 2001. The new software is expected to be installed in the deployed worldwide fleet of C-130J aircraft during a one-year period beginning the summer of 2001 after Air Force qualification testing is completed at the Air Force Flight Test Center at Edwards Air Force Base.

Plans for reuse of C-130J software and technology were laid out during the early days of the software development effort. The C-130J's advanced avionics technology and mission computer software are already being reused in the C-27J aircraft, the C-5 Aircraft Modernization Program, and Lockheed Martin's proposed Joint Strike Fighter. C-130J avionics and software reuse has also been proposed for the Lockheed Martin's C-130 Aircraft Modernization Program that is intended to incorporate newer technology into the older C-130 aircraft in the fleet.

## The C-130J Aircraft as a Software Intensive System

The C-130J aircraft is an integrated collection of software systems produced by more than 25 suppliers. These systems, which are developed in compliance with the Lockheed Martin C-130J Tier I Software Development Plan, are integrated with the devices on the aircraft such as the engines, pneumatics, flight station displays, and the radar. A common Tier I Software Development Plan helped to enforce commonality between all the suppliers, making integration of their products into the air vehicle easier.

The Lockheed Martin C-130J Software Integrated Product Team develops the air vehicle and ground-based data system software also in compliance with the Tier I plan. Thus Lockheed put the same commonality requirements on itself as it did its suppliers. All suppliers, including Lockheed, produced their own Tier II Software Development Plans per directions in the Tier I Software Development Plan.

The air vehicle software consists of the Mission Computer (MC) Operational Flight Program (OFP) and Bus Interface Unit (BIU) OFP. The MC OFP manages the overall software operations within the C-130J aircraft and executes within a normal or backup mode. Both modes of the MC OFP include the primary roles of maintaining a central database, providing executive control for all software functions, providing interfaces to the MIL-STD-1553 data buses, and performing fault detection/fault isolation.

The BIU OFP operates in conjunction with the MC OFP in performing the integration of the C-130J avionics. The BIU OFP operates within a normal mode or an MC backup mode. The primary roles of the BIU OFP during normal mode operations are monitoring health, storing and validating critical data, and providing interfaces to non-MIL-STD-1553B data sources. The primary roles of the BIU OFP during MC backup mode operations include acquiring the role as bus controller and performing critical functions.

The ground-based data system software includes the Ground Maintenance System (GMS) and the Organizational Maintenance System (OMS). The GMS is a ground-based computer system that provides a central database for maintaining line-replaceable unit (LRU) configuration information and archived aircraft history for each tail number in the C-130J fleet or squadron. The GMS processes the maintenance-related data recorded to on-board removable memory modules on the C-130J aircraft.

The GMS provides an automated or manual flight crew maintenance debrief function and reads data stored on the removable memory module. The GMS validates the downloaded data, runs automatic fault isolation routines, calculates health and usage parameters, and generates maintenance work orders as required. The system processes structural and engine data to monitor component life and supports configuration control and status reporting of the air vehicle. The GMS maintains a variety of printed reports to support aircraft maintenance. The GMS is also hosted on the Portable Maintenance Aid, which is loose equipment for each C-130J. This capability is provided to support the need to forward deploy the aircraft for operations away from its home base.

The OMS provides the user interface between the maintainer and the C-130J aircraft systems for performing organizational level maintenance on the aircraft. The OMS supports the maintainers by accessing electronic technical orders, troubleshooting aircraft failures, evaluating status of aircraft systems, checking configuration of aircraft systems, and uploading and downloading files to and from the aircraft systems. The GMS interfaces with the OMS for maintenance work order processing, status reporting of maintenance actions performed, and recording of diagnostic data during ground maintenance.

## The Software Factory

In the culture of our aircraft manufacturing facility, software is a part on the aircraft, tracked just like the engines, pneumatic systems, and radar systems. The C-130J Software Integrated Product Team operates a software factory that produces the air vehicle and ground-based data system software parts and approves the software parts for all computerized devices on the aircraft. The air vehicle software parts are written in Ada (250,000 lines of code), and the ground-based data system software parts are written in C++ and a fourth generation language (400,000 lines of code total) for each aircraft. Each software part has a part number, a set of associated drawings, and an assembly (such as a removable memory module). The drawings associated with each software part include the following:

- Software Item Drawings assign a unique part number to each computer software configuration item that is 1) installed on the aircraft, 2) used to create or prepare a part for aircraft installation, or 3) used to install or transfer a software item into an aircraft part. The notes on each Software Item Drawing describe 1) the host hardware part number, 2) the image file names and software version identities or a reference to the document containing specific software configuration information (i.e. version description document), and 3) the software-to-software compatibility dependencies.
- Software Assembly Drawings are produced for each software assembly (integrated collection of software items). A Software Assembly Drawing describes 1) a software assembly used in the production of a deliverable part, or 2) a software assembly delivered to a customer. Software Assembly Drawings assign a unique part number to each release of each software assembly. The parts list in the Software Assembly Drawing describes the software items (by part number and location code) contained on the assembly and the specific media (i.e., 3.5-inch diskette, 4mm tape, etc.) of which the assembly is made. The notes on the Software Assembly Drawing describe

1) the configuration of any vendor-supplied software items (i.e., reference to Vendor's Version Description Document), 2) the specific software assembly instructions used to create the software assembly, and 3) the contents of the label placed on the completed software assembly.

- Software Assembly Instruction Drawings are produced for each deliverable software assembly. The Software Assembly Instruction Drawing describes the required hardware equipment, software environment, personnel, access privileges, and detailed procedures necessary to produce the software assembly.
- Software Installation Instruction Drawings are produced for each software item installed into a deliverable part. The Software Installation Instruction Drawing describes the required hardware equipment, software environment, personnel, access privileges, and detailed procedures necessary to install the software item(s) into the host hardware part.
- Software Index Drawings facilitate the identification of customer deliverable software on each aircraft model, thus allowing the software design organization to control interim software releases to production aircraft without changing the master index for production software releases that are not delivered to a customer.
- Software Control Drawings are produced for each C-130J customer. The Software Control Drawing details the software and hardware combinations delivered to each customer. The body of the Software Control Drawing contains the following information for each deliverable software item: 1) find number, 2) software description, 3) identification of the software manufacturer, 4) software part number, 5) software version identity, 6) the aircraft model, version, serialization usage of the software/hardware combination, 7) note references, 8) hardware description, 9) identification of the hardware manufacturer, and 10) the host hardware part number. Notes in the Software Control Drawing describe: 1) which software items are loadable in the field and 2) any software compatibility/usage limitations.

The people who work in the C-130J Software Factory are collectively called knowledge workers, and they serve in many distinct roles such as software product managers, software requirements engineers, software development engineers, software test engineers, software process engineers, software quality assurance specialists, and documentation specialists. These knowledge workers are tied together through a digital nervous system (DNS), a term coined by Bill Gates of Microsoft [2]:

> "A DNS comprises the digital processes that closely link every aspect of a company's thoughts and actions. Basic operations such as finance and production, plus feedback from customers, are electronically accessible to a company's knowledge workers, who use digital tools to quickly adapt and respond. The immediate availability of accurate information changes strategic thinking from a separate, stand-alone activity to an ongoing process integrated with regular business activities."

## Reuse

Software reuse has been at the heart of the C-130J Software Factory since development of the C-130J aircraft began in 1992. The program started with domain analysis and engineering, looking at what could be reused from other programs, defining the domain of the C-130J, and creating reusable assets that have been exploited throughout the program. The cost of developing air vehicle and ground-based data system software is the primary reason for Lockheed's aggressive efforts to achieve real, effective reuse. Reuse has significantly lowered the life-cycle cost and program risk.

Many products of the C-130J Software Factory were designed from the beginning to be reusable:

- Template-Based Design: Six domain-specific design patterns were originally created to serve as class definitions for all device interfaces to the MC OFP and the BIU OFP. Since 1992, three more design patterns were created to address new technology transition, bringing the total to nine design patterns. Courseware was prepared to document these design patterns and teach newcomers how to use the patterns. The productivity gains, improved reliability, and reduced testing overhead provided by applying template-based design were observed throughout the development of the software.
- Source Code: For many device interfaces, source code used for other device interfaces could be reused with very minor modification. In addition, source code from previous blocks could be reused extensively on later blocks (note the reuse figures between Blocks 2 and 3 and Blocks 3 and 4, see page 19).
- Test Scripts: Due to the definition of the classes of device interfaces, test scripts could also be reused. Requirements-based testing also helped by supporting automated generation of test cases directly from the requirements specifications.
- Documentation: Delivered and internal documentation was designed to be reusable, facilitating its production from one software build to the next.
- Software Development Domain Specific Kits (DSKs): Commercially-available DSKs, such as Microsoft Visual Studio .NET and Microsoft Visual Basic for Applications, greatly enhance productivity. We also employ homegrown DSKs, such as our Data Collection System Version 3, which is a DSK designed to build data collection applications.
- Common Software Development Tools: Our Environment and Tools Working Group establishes a set of common software development tools, such as Rational APEX and Cadre Teamwork for use on several Lockheed Martin programs. We save cost in terms of both purchase price and training, and we gain by having more readily interchangeable personnel. Reuse is also enhanced in that tool-specific conversions are reduced or eliminated should an asset produced by one program be adopted by another.
- Domain Knowledge: Knowledge captured during the early domain analysis and engineering activities was stored in courseware, reusable as a teaching instrument throughout the life of the program.

## Challenges

The C-130J aircraft denotes a cultural change in a significant part of a major corporation from producing largely mechanical aircraft to producing software intensive aircraft. Such a change takes time for the culture to adapt, and there are many challenges that both the management and technical communities within that culture must face. These are the challenges faced by the C-130J Software Integrated Product Team:

- Building safety critical, high integrity [3] software for an aircraft with corporate funding (the development of the

| Statistic Tracked | 1998 | 1999 | 2000 |
|---|---|---|---|
| Number of changes processed | 2,430 | 2,350 | 2,115 |
| Number of engineering software builds | 240 | 300 | 330 |
| Number of software qualification tests | 79 | 85 | 81 |
| Number of pages of documentation produced | 472,500 | 564,200 | 531,010 |
| Number of software tests executed | 700,450 | 798,683 | 751,700 |
| Test success percentage | 98.27% | 98.75% | 99.00% |

Table 1: *Modern Avionics in the C-130J has Contributed to its Improved Performance*

C-130J was done without funding from external sources, such as the United States government), the corporate investment and risk were high.

- Reducing risk and life-cycle cost for a software intensive system with a 30-year life span by achieving effective software reuse.
- Designing a software intensive system that is adaptable to changing technology during a 30-year life span.
- Meeting the requirements of FAA Type Certification.
- Controlling changes and software versions in light of thousands of requirements against multiple baselines for multiple customers, and creating different builds for different customers concurrently – satisfying the needs of a diverse group of customers, each with their own unique requirements during a 30-year life span.
- Achieving Capability Maturity Model® Level 3 and ISO 9001 certifications and continuing the investment needed to maintain these certifications.

From a broad perspective, the challenges may be grouped into four areas: software reuse, process, certification (for CMM Level 3, ISO 9001, and the FAA), and culture. Within the domain of our company (aircraft development and manufacturing), these challenges were addressed from the point of view of the pre-software intensive culture that was already in place:

- Software reuse was one of the easier challenges to address. The concept of line replaceable units (LRUs) was already in management's minds from a hardware perspective, so adding software parts as LRUs was not a significant leap. Neither was viewing those software parts as complex parts containing smaller component parts. Domain engineering was done at the beginning of the program, at a time when the development laboratories were not yet ready and the systems engineers were engaged in design and simulation. Ideas were also picked up from other existing aircraft programs, adding credibility to our domain engineering effort.
- Introducing a software process orientation was also an easier challenge to address. Management was already aware of manufacturing process concepts, so software development process concepts were not a significant leap in the early stages. A common Software Engineering Process Group was readily established to share ideas and infrastructure between the various software development Integrated Product Teams, such as the C-130J, F-22, C-5 AMP, and C-27J.

  The primary obstacle to our process definition efforts arose when management implemented a lean initiative to reduce waste in both the hardware and software processes. In the efforts to completely document the processes, it became evident how expensive a complete process description would be to produce. In describing our software development processes down to the level of following the trail of paper and electronic data between people's desks, the C-130J Software Integrated Product Team alone ended up with 114 distinct processes in a hierarchy that was three levels deep.

  This collection of process descriptions was a small part of the overall detailed process description for the development and manufacturing of the entire aircraft, which is currently incomplete and estimated to be between 3,000 and 5,000 distinct processes. The effort to create the detailed process description for the hardware side is continuing as we are moving to CMMI adoption.
- Certification activities were more challenging than software reuse and process. Our lean effort described in the previous bullet was a significant aid in our CMM Level 3 certification activities, and applying web technologies to describe our processes allowed us to present this information from the point of view of a CMM assessor, organized by Key Process Area and Key Practice. The introduction of automated data collection during the last three years has made it much easier to produce the evidence demanded by the CMM assessors, but gathering more than 300 artifacts for a CMM assessment is still a daunting task. The challenge of FAA Type Certification was similar to CMM Level 3 certification, and the ISO 9001 certification challenges fell nicely into place as our CMM Level 3 certification challenges were addressed.
- The cultural shift required by management to understand the issues and culture of the software engineers was our greatest challenge. Management expectations were originally high that software engineers could possess the same domain knowledge as systems engineers, and this was simply not the case. The mindset of someone with a master's degree in mechanical or electrical engineering, especially if that degree was granted more than 10 years ago, is fundamentally different from the mindset of a contemporary software engineer.

  Attempts were made to have systems engineers perform software engineering work – the success of these attempts was mixed. Over time, systems engineers and software engineers gradually came to understand each other's mindsets, but occasional personnel turnover disrupted this understanding; we found a continual need to reeducate engineers on both sides.

  Likewise, management's acceptance of software engineering concepts has been gradual, again requiring reeducation with personnel turnovers. After a decade, the three groups – management, systems engineering, and software engineering – still do not completely accept each other's mindsets. We expect this cultural difference to continue for some time to come.

The following statistics are noted in the more than 5 million source lines of code delivered to date: The C-130J software has been built for a 30-year life span. A lot can change in terms of the demands placed on the C-130J aircraft and its mission during these many years. Incorporation of a Global Air Traffic Management system and a comprehensive software maintenance plan are two of the efforts currently underway, and software production is continuing with a projection of more than 9 million lines of code delivered by the end of 2001. New missions, different requirements from new customers, changing requirements from existing customers, and the introduction of even newer technology to the aircraft are the key factors causing this software growth. Continual process improvement, particularly through the C-130J Digital Nervous System, is underway, and increasing levels of capability maturity, through CMM Level 4 to Level 5, are planned.

## Lessons Learned

Many lessons were learned during the last decade of the C-130J software development. Here are some key lessons:

- Objectives and requirements must be nailed down specifically from the beginning. It is never possible to get the requirements right the first time if the problem is of any significant degree of complexity. Requirements traceability and requirements grading are required. Conduct software product evaluations on requirements as intensely as you would review the code.
- You can never have too many simulations or laboratory resources.
- Software engineering capability maturity is not enough by itself to improve the quality of an integrated system like an aircraft. Systems engineering and management capability maturity are also required.
- Driving a product by schedule is unavoidable. Be prepared to deal with it and be prepared to adapt when the schedule slips. Define all your processes and measure their performance. Remember that the last process in the sequence is not necessarily the source of the problem when a schedule slips.
- Automate testing as much as possible. Always plan on running a test again. Always base test cases on requirements, trace test cases to those requirements, and employ automated tools to build your test cases from your requirements specifications when possible.
- Successful reuse requires a significant up-front cost and an effective, compelling producer/consumer model that makes it economically viable. Management must see reuse values and accept the costs as well as the benefits.
- Measurement comes with capability maturity, but no measurements can replace the in-depth, detailed knowledge of the people on the development line. Management must journey to the (software) factory floor before they can really understand the issues.◆

## References

1. Lockheed Martin. C-130J Hercules Web site, <www.lmasc.com/c-130j/index.htm>.
2. Bill Gates. Business @ The Speed of Thought – Using a Digital Nervous System, Warner Books, 1999, <www.speed-of-thought.com>.
3. Sutton, James (Lockheed Martin), and Carre, B.A. (Praxis Critical Systems). Achieving High Integrity at Low Cost: A Constructive Approach, ERA 1995 Conference, London, United Kingdom.

## About the Authors

Richard L. Conn has more than 20 years experience in software engineering and project management. Conn is currently the software process engineer for the C-130J Airlifter at Lockheed Martin Aeronautics Company. He graduated with bachelor's and master's degrees in computer science from Rose-Hulman Institute of Technology in 1976 and the University of Illinois in 1978, respectively. Conn was an Army officer from 1978-82 at the Army's Satellite Communications Agency and the Air Force Institute of Technology, where he taught computer science. Conn was a member of the Federal Advisory Board for Ada and a distinguished reviewer of the Department of Defense's Software Reuse Technology Road Map.

Lockheed Martin Aeronautics Company
86 South Cobb Drive
Dept. 70-D6, Mail Zone 0674
Marietta, GA 30063-0674
Phone: (770) 494-1670
Fax: (770) 494-1345
E-mail: richard.l.conn@lmco.com

Stephen M. Traub has more than 20 years experience in software engineering and project management. Traub is currently the software designated engineering representative at Lockheed Martin Aeronautics Company on behalf of the Federal Aviation Administration. Graduating from Elon University in North Carolina in 1984, Traub worked for Unisys from 1980-1984 as the principal software engineer for Weapons Assignment tasks for several Navy shipboard systems. He has been at Lockheed Martin since 1984, first working on the C-5B aircraft, and then working on the C-130J in the roles of Mission Computer Software Development lead, software product manager, and Software Integrated Product Team lead.

Lockheed Martin Aeronautics Company
86 South Cobb Drive
Dept. 70-D6, Mail Zone 0674
Marietta, GA 30063-0674
Phone: (770) 494-1670
Fax: (770) 494-1345
E-mail: stephen.m.traub@lmco.com

Steven J. Chung has 18 years of experience in software engineering and project management. Chung is currently the Software Integrated Product Team lead for the C-130J Airlifter at Lockheed Martin Aeronautics Company. Graduating from the University of South Florida in 1983, he worked for Honeywell Space Systems as a software engineer on the Space Shuttle and the Advanced Space Communications Technology programs and E-Systems on a real-time communications network. Chung came to the C-130J program at Lockheed in 1996 as a staff engineer and was promoted to Software Integrated Product Team lead in 2001.

Lockheed Martin Aeronautics Company
86 South Cobb Drive
Dept. 70-D6, Mail Zone 0674
Marietta, GA 30063-0674
Phone: (770) 494-1670
Fax: (770) 494-1345
E-mail: steven.j.chung@lmco.com