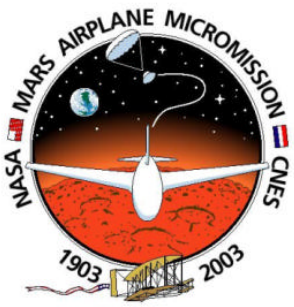
Mars Aerial Research Vehicle
M.A.R.V.



University of Colorado at Boulder

Undergraduate Aerospace Engineering
Ben Mottinger *Jason Muckenthaler*
Dan Wicklund *Corissa Young*

Undergraduate Computer Science
Colin Graham *Eric Schell*

Undergraduate Electrical Engineering
Ted Kutrumbos

Faculty Advisors
Dr. Brian Argrow
Dr. Jason Hinkle
Dr. John Sunkel

ABSTRACT

Mars Aerial Research Vehicle (MARV) is based on the NASA Langley mission Mars Airplane Package (MAP). The deployment sequence of the MAP was designed, as well as the method of separation from the aeroshell. To determine the stability behavior during separation from the aeroshell, a wind tunnel model of the airplane was constructed and tested for pitching moment. Also, a VxWorks based software system was implemented to provide video imaging and to control the airplane deployment and camera position. The ground software was written in Java to provide a portable data evaluation system.

TABLE OF CONTENTS

MISSION OVERVIEW

NASA has a vision statement that reads, “NASA is an investment in America's future. As explorers, pioneers, and innovators, we boldly expand frontiers in air and space to inspire and serve America and to benefit the quality of life on Earth.” This is a very bold statement that encompasses many areas of Space exploration. One specific area it speaks to is the Human Exploration and Development of Space (HEDS), which is one of the foci of the Mars Aerial Research Vehicle (MARV) project. The HEDS division of NASA has generated it’s own mission statement that reads, “To open the Space frontier by exploring, using and enabling the development of Space and to expand the human experience into the far reaches of Space.”

Human exploration of Space has truly been and continues to be a driving force for the entire Space industry. Not only has it been responsible for much of the advancement and development of Space exploration materials and techniques, but it has also been a significant contributor to the excitement and interest of the general public in that vast expansion referred to as “Space: the final frontier”. Sending people into Space has been the source of much pride for the United States since the early 1960’s. In order for the United States to remain a leader in the Space industry, we must continue the advancement of human exploration of Space.

Langley Research Center (LaRC) has proposed the development and launch of a mission that will include the powered flight of an aircraft in the atmosphere of Mars. The mission will demonstrate the technology needed to fly an aircraft on another planet for the purpose of collecting science data, as well as taking high-resolution pictures of the Martian surface. The mission, known as the Mars Airplane Package (MAP), will also celebrate the first powered flight of the Wright Brothers in 1903. The Mars Airplane will be taken to Mars on the 2003 Mars spacecraft mission. The transfer spacecraft will drop the airplane package into the atmosphere, where the airplane will separate from the aeroshell (heat shield) and deploy autonomously wings and a tail. The airplane will then attain and begin powered, level flight, which will last a minimum of 120 seconds. The airplane will continue flying until the end of the usable communications window, which is estimated to be 20 minutes. During the entire flight, science and engineering data from the airplane will be relayed to Earth via the orbiting satellite.

The MARV team selected the deployment of the wings of the MAP as the design project focus. This problem encompasses many different engineering disciplines including aerospace, computer science, and electrical. For this reason, the MARV team is comprised of 4 aerospace engineering students, 2 computer science engineering students, 1 electrical engineering student, a computer science advisor, an electrical advisor, a structural advisor, and an aerodynamic advisor. This interdisciplinary team has been able to complete a more comprehensive and complete design for the wing deployment of the MAP.

The MARV project was divided into four phases:

- *Phase I*: the preliminary design portion of the project. During this time the group researched the design (aerodynamic and software) of the MAP on a very high level.
- *Phase II*: focused on one aspect of the MAP design, specifically the deployment system.
- *Phase III*: hardware acquisition, machining, and construction
- *Phase IV*: integration and testing phase.

The end goal for the project was to design the wing packaging and wing deployment for the MAP, with the end result being a fully deployable wing with the accompanying actuator, microprocessor, and supporting software. The secondary goal for the deployable wing was to conduct wind tunnel testing of its pitch stability. A complete software architecture design was also developed for the MAP along with all of the accompanying electrical components that were necessary for integrating the aerospace portion of the project with the software (computer science) portion of the project.

Phase I involved extensive research of the MAP design proposed by NASA Langley including aerodynamic plane designs along with software architecture concepts. This research served as the basis for the remainder of the MARV project. Previously developed concepts (from LaRC) and ideas were expanded upon to include the design of the deployment mechanism and sequence for the wings of the MAP.

Phases II and *III* will see the MARV group divided three ways:

- The first group was comprised of three of the four aerospace engineering students and will focus on the structural design and building of the MAP wing model, followed by the design of the wing packaging and deployment system. The portion of the deployment system situated on the wings consisted of shape memory actuators.
- The second group had the remaining aerospace student and the electrical engineering student. Group number two designed the electrical deployment mechanism that was initiated by the software developed by group three. The complete deployment mechanism is comprised of two shape memory actuators (on the wings) and the supporting power electronics. This group was also responsible for creating the necessary circuitry for a digital camera, which was implemented because of the requirement of MAP to take high-resolution pictures of the surface of Mars.
- The third group comprised of the two remaining students – both Computer Science majors – designed and implemented the control software for the supporting electronics for the deployment mechanism and the control software for the digital video camera.

Phase IV involved testing of the individual components including software code, electrical circuits, and wing stability. Then, final integration produced the demonstration of a fully deployable wing system for the MAP project.

AERIAL VEHICLES AS PLANETARY EXPLORERS

The Mars Airplane Project (MAP) concept was designed as a low-cost atmospheric research vehicle. Although the mission requires interplanetary space travel, the MAP was conceived along with its transfer vehicle to be launched as a secondary payload aboard an Ariane 5 launch vehicle. The implications of launching an interplanetary mission as a secondary payload are profound for NASA's exploration program. Secondary payloads require a fraction of the cost-to-orbit that primary payloads do.

The drawback to secondary payloads has always been two-fold. Secondary payloads are essentially hitchhikers and thus do not have the command of the launch vehicle that the primary does. Interplanetary missions often have narrow launch windows, making it difficult to rely on the primary payload's schedule. In addition to the scheduling constraints, secondary payloads suffer from extreme volume and mass constraints.

The MAP was conceived as a small vehicle with a short mission lifetime, translating into very limited, but valuable science returns. The planetary transfer vehicle – the Mars Micromission Spacecraft (MMSC) – was designed to carry several modules and deposit them into the Martian Atmosphere using an aeroshell-parachute system similar to that used on the Viking Missions. The MMSC would then act as the main communications relay for the objects deposited onto the planet. Multiple missions would build on the communications coverage by supplanting the previous MMSCs already in Martian Orbit.

Because of the inexpensive, disposable, and simplistic nature of airplanes similar to MAP, multiple missions could be sent at low cost. These missions have potential for other planetary exploration missions as well. Due to the proximity of Mars and the relative similarity in atmosphere and gravity, the MAP was well suited to atmospheric exploration and imaging. However, the potential for this type of exploration is not limited to aircraft. Dirigibles also present definite possibilities for atmospheric exploration due to their significantly increased mission lifetime and payload capability.

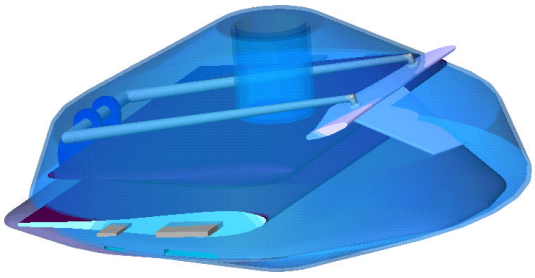


Figure 1: MAP in stowed configuration inside aeroshell (top). MAP separating from aeroshell (right).

MAP DEPLOYMENT DESIGN

The preliminary design for the Mars Air Plane (MAP) called for a disk shaped fuselage with deployable wings and deployable tail structure. The MAP shape was developed out of the general desire to maximize the scientific payload housed with in the fuselage. In order to maximize this payload and thus the scientific return from the mission, the wing area needed to be as large as possible. After looking at several possible deployment configurations for the MAP, it was decided that the wings needed to fold on top of each other:

This packaging method requires a more complicated wing deployment mechanism that will allow the wings to fold flat on top of each other. The most significant impact of this design decision is on the aerodynamics of the airplane during the deployment phase.

Deployment Design Assumptions

Because of the uncertainties associated with a free fall trajectory on Mars – wind, exact densities, exact orientation – assumptions were made to facilitate the design of the deployment sequence while not sacrificing requirements.

- Because of the unknown trajectory once the MAP is inside the atmosphere, it was assumed that the aeroshell's transverse velocity would remain much smaller than the vertical velocity during the free fall phase of the deployment.
- The orientation of the aeroshell is assumed to be roughly horizontal.
- The attitude of the aeroshell relative to the martian surface does not need to be known for a successful deployment and flight.

Pre-Deployment Conditions

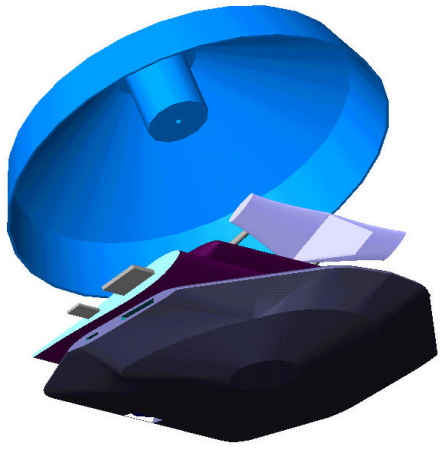
Altitude	6.5 Km
MACH Number	0.9
Vertical Velocity	~207 m/s
Transverse Velocity	~0 or Vh << Vz
Orientation	~ Horizontal

Post Deployment Conditions

Minimum Altitude	5.0 Km
MACH Number	0.8
Vertical Velocity	~207 m/s
Transverse Velocity	~0 or Vh << Vz
Orientation	~ Horizontal

Aeroshell Deployment

The mission profile calls for the airplane to be deployed from the aeroshell at close to a 90° angle of attack relative to the oncoming airflow and then assume steady level flight. The deployment phase thus has to be designed such that the airplane can separate from the aeroshell, deploy its tail and wings, and recover from the ensuing dive while still maintaining as much altitude as possible, while accounting for the fact that the desired orientation for deployment may be achieved with a only a certain uncertainty.



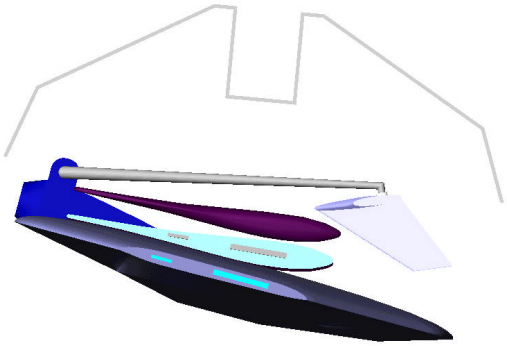


Figure 2: MAP being deployed from aeroshell top. The diagram shows the initial pitch angle.

The aeroshell deployment sequence was then designed so as to ensure that the MAP would attain a pitch forward dive in all but the most extreme circumstances upon separation from the top part of the aeroshell. Because the

airplane is essentially a flat disc upon ejection from the aeroshell, the aeroshell must have a mechanism to impart a nose-down pitch on the airplane.

The nose-down pitch would be accomplished by a set of separation springs. The springs would be set to kick the airplane out of the aeroshell with enough of a moment to prevent the MAP from falling back on itself. The springs would also serve to prevent the aeroshell from interfering with the deployment of the mechanisms.

Airplane Deployment Sequence

Time	Event	Description
Stage 1:		
$\mathbf{T_{-1}}$	Aeroshell Bottom Separation	Aeroshell Mechanism
Stage 2:		
$\mathbf{T_0}$	Aeroshell Top Separation	Release from MAP
Stage 3:		
$\mathbf{T_1}$	Tail Deployment	
$\mathbf{T_2}$	Wing 1 Deployment	
$\mathbf{T_3}$	Wing 2 Deployment	
Stage 4:		
$\mathbf{T_4}$	Horizontal Stabilizer Deflection	
$\mathbf{T_5}$	Ignite Engine	
$\mathbf{T_6}$	Begin Level Flight	

Immediately after the MAP is kicked out of the aeroshell, the tail assembly will deploy further pitching the nose of the aircraft down and causing the airplane to go into a nosedive. The reason for the desire to have the plane in a known orientation – specifically a nose dive – is to allow for the least amount of control needed while simultaneously allowing the vehicle to develop a velocity vector parallel to its nose as opposed to its belly, allowing the wings to produce lift and minimizing the amount of thrust necessary from the engine to produce this forward velocity.

At this stage of deployment, the MAP has a deployed tail and stowed wings and is free falling with a pitch down pitching moment. The wings are then deployed at a certain delta t from the tail to ensure that the tail has been locked into place and the wing deployment will not cause the plane to pitch unrecoverably.

The wing deployment presents one of the more interesting challenges for the deployment phase in terms of stability. The wings are deploying asynchronously in free fall. Therefore, the possibility exists for not only changes in the pitching moment that might be outside the correctibility range of the control surfaces, but also a rolling moment and associated coupled yawing moment.

THE MARV PROJECT

MARV is a quarter scale model of the MAP built to test the aerodynamics associated with the asynchronous and asymmetric nature of the aircraft during deployment. Specifically, MARV looks at the pitching moment during the wing deployment. MARV was built with the tail section fixed, but with deployable wings.

Test Article Design

The wind tunnel model design requirements were driven by the pitch moment measurement. Several test constraints including the test section size of the wind tunnel descoped the original model design. The wind tunnel test section limited the model to quarter scale, which imposed constraints on the size of deployment actuators and the release mechanism. Therefore, the wing deployment actuators (torsion springs), were mounted on the top of the fuselage. A small penalty in drag resulted from the external mounts, but the assembly and manufacturing complexity was reduced.

The original aeroshell volume constraint was also considered when designing the wing stowage. The wings fold flat on the fuselage, parallel to each other. This configuration requires a double hinge mechanism for the wing that folds on top of the other one (Fig. 3).

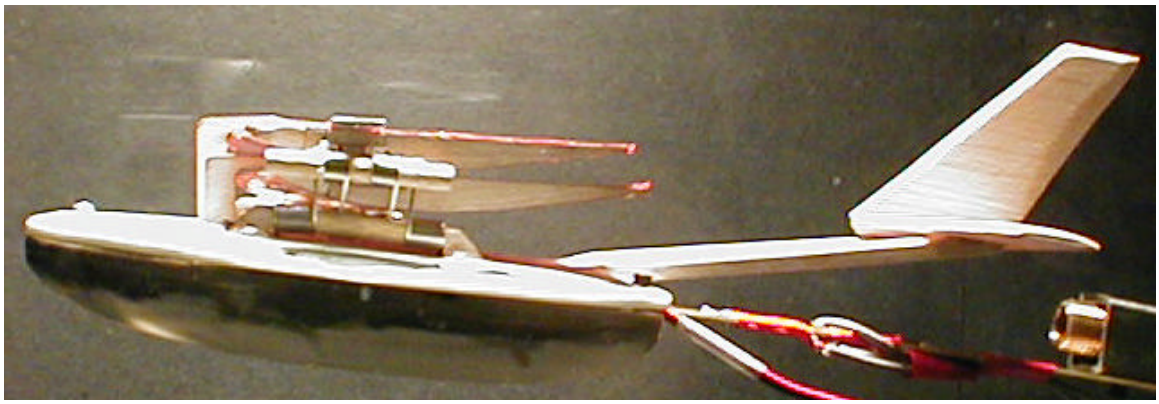


Figure 3: Stowed configuration showing double hinge mechanism

The release mechanism was also designed from a function/simplicity standpoint. The wings were held in the stowed position by lever arms that were actuated by NiTi shape memory alloy (SMA) electric pistons (see Fig. 4 & 5). The SMA pistons are activated by current, and the temperature change in the NiTi that results causes the alloy to contract in length, providing the linear force to pull the levers. When the alloy cools, it can be moved back to its original shape. The particular pistons used for the wind tunnel model provide about 1 lbf of force.

The fuselage was designed in two halves so that the SMA actuators could be mounted in the bottom half and accessed by removing the top.

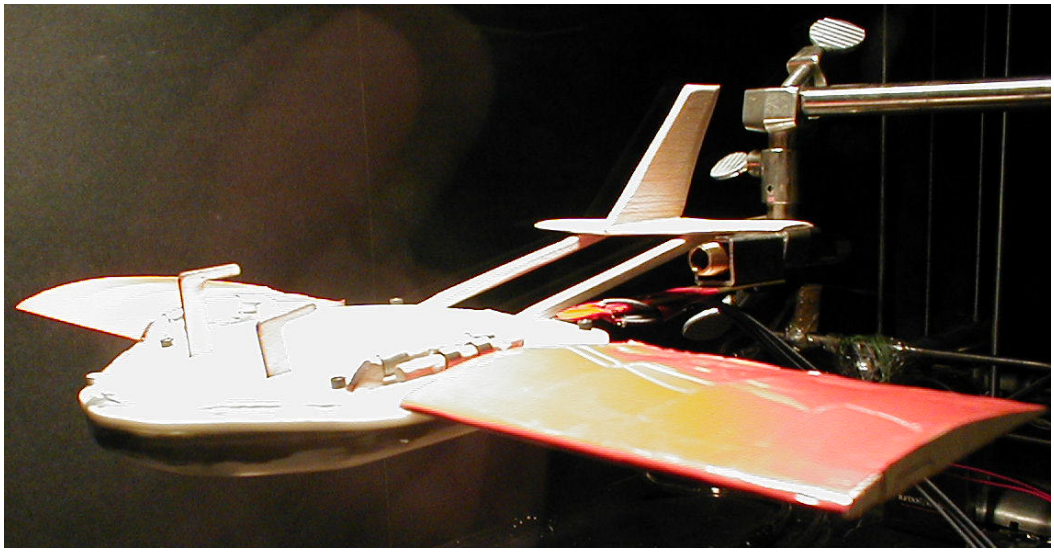


Figure 4: **MARV deployed in the wind tunnel**

Figure 5: **Wind tunnel model at 45^{o} angle of attack**

Test Article Manufacturing

The entire wind tunnel model and attachment hardware was manufactured by the MARV team at the University of Colorado. The fuselage was processed by a Rapid Prototype Device (RPD) using stereo lithography SolidWorks part files. Each half was processed in about 8 hours running continuously. The tail assembly was made from balsa wood by hand. The wing ribs were cut from maple wood on a Laser CNC to exactly match the Eppler 387 airfoil section (Fig. 5 shows wing ribs and spars). Covering for the wings was MonoKote™ model airplane iron-on material.

Experiment Overview

A pitch test was designed to determine the static pitching moment at angles of attack ranging from zero to 90 degrees (orthogonal to flow). By determining the static pitching moment of MARV at all angles of attack, a prediction of its behavior after aeroshell separation could be made. Additionally, a static pitch test was designed to measure the pitching moment during the wing deployment, to assist in determining pitching trends during wing deployment. A computer model was programmed to model the theoretical pitching moment of the static, deployed MARV at angles of attack up to 90 degrees, for comparison purposes with the measured wind tunnel data.

Experimental Setup

The testing was completed in a low-speed wind tunnel, with a 2x2 ft test section. The model was attached to the tunnel via a 15-cm rectangular cross-section, brass sting attached to a cross-flow bar. The sting was instrumented with four strain gauges, which measured strain along the axial direction of the sting. The moment measured at each strain gauge was determined via a calibration curve, created from tests during no-flow conditions. The full-velocity tests were run at 15m/s in both the (wings) stowed and (wings & tail) deployed configurations, with angles of attack varying from zero to 75 degrees.

The pitching moment was backed out of the measurements by taking strain measurements at two different points on the sting. The difference between the strains allowed the pitching moment to be differentiated from the moment in the sting due to the effective tip load (summed lift and weight of the model).

Experimental Results and Analysis

Flutter was encountered at angles of attack approximately 5 degrees on either side of stall, preventing the data precision obtained in other measurements. In the deployed configuration, MARV was found to stall at an angle of attack of approximately 12 degrees above the zero-lift AoA. In the stowed configuration, the stall point moved up to approximately 22 degrees above zero-lift.

The computer model was based on predictions of drag and lift, and is only accurate to use as a reference for pitching *trends*, as the ultra-high angle of attack regions of the model are relatively unexplored in terms of lift

and drag. Functions were developed based on approximate trends to model the lift coefficient and drag coefficients at angles of attack above stall. The computer model was based upon MARV dimensions, and flow characteristics present during the wind tunnel testing, for the sake of comparison with the measured data.

The model shows stable pitching trends from zero degrees up to stall, neutral pitch tendencies up to nearly 30 degrees AoA, and unstable pitch tendencies at higher angles of attack. These trends can be seen in Figure 6. The wind tunnel data obtained shows different pitching trends from the theory. The data shows unstable pitch tendencies throughout the range of angle of attack. The comparison between the wind tunnel data and the computer prediction is shown in figure 6.

Due to the aforementioned flutter at angles of attack around stall, the data in this range have a large uncertainty associated with them, which accounts for some of the discrepancy between the theoretical and predicted models. Additional error was encountered at large angles of attack because the flow over the tail was disturbed by the sting attachment prior to reaching the tail. This disturbed flow drastically reduced any lift acting from the tail, which would have helped to stabilize the aircraft. The model itself was comprised of many approximations, and, as previously stated, the figure below provides trends of the data and prediction.

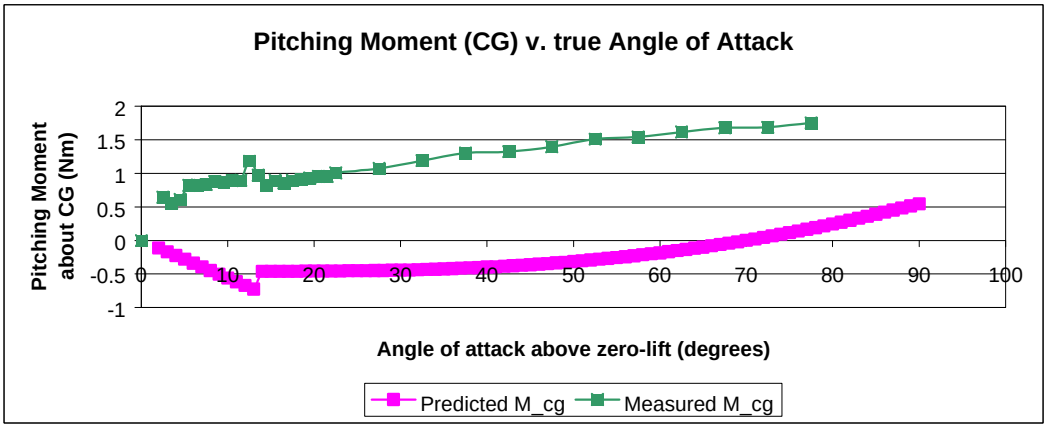


Figure 6: Measured and predicted pitching moments about MARV cg

The instability in the wind tunnel testing is likely due to two factors: the first is that the contribution of the body to lift was neglected, when it was found to represent a significant portion of the total lift. The second factor is that the tail was not sized for stability purposes in the MARV design. Contributing further to this factor is the disturbed flow from the sting, which was seen by the tail at high angles of attack.

The instability encountered in the wind tunnel testing would result in a massive pitch up moment during the wing deployment of the MAP, possibly sending the aircraft into an unrecoverable spin. Redesign of the MAP would be necessary to provide a large pitch down moment, even at ultra-high angles of attack. This negative pitching moment would serve to place the MAP into a dive, which would provide stability during the pull-out maneuver and engine firing. This pitch down moment would partly come from the separation springs in the aeroshell, and could be increased by increasing the tail size. Also, moving the cg forward would ensure dramatic stability increases.

ELECTRONICS SYSTEM:

The electronics system also known as the Control and Sensing System (CSS) is the integration point between the MARV flight software and the MARV plane (Figure 7). The main goal of the CSS is to control the deployment of the wings of the MARV plane when signaled from the flight software. The CSS also controls the position of the video acquisition unit. The CSS is best explained if broken into four subsystems; the micro-controller, the serial interface, camera controller, and the power amplifier.

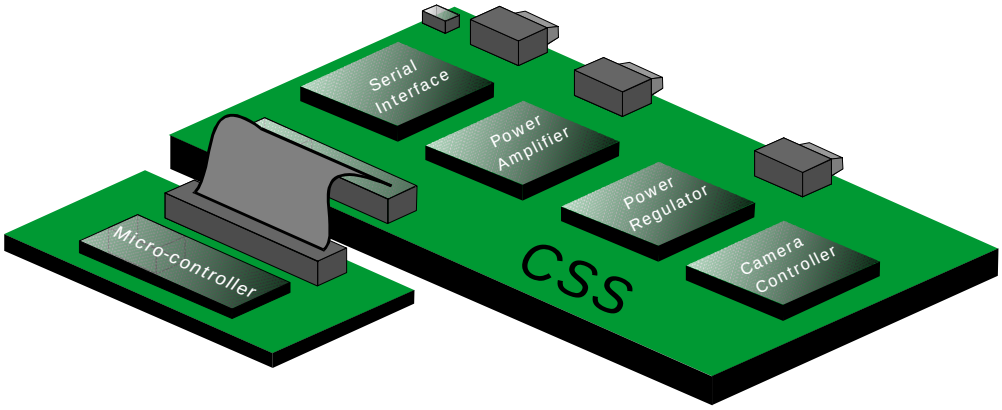


Figure 7: Control and sensing System (CSS)

Micro-controller

The micro-controller is the brain of the electronics system. The micro-controller must be capable of converting an acsii signal from the flight software into a TTL voltage level that will control the other subsystems. It was found that the best suited micro-controller for this project is the OOPic from savage Innovations (Figure 8). The OOPic is the first micro-controller that uses an object oriented programming language. The ease of programming in an object oriented language drastically reduced the programming time and allowed more time to be devoted to the actual electronics. The OOPic communicates with the flight software, which is located on a x86 target running VxWorks through a RS232 serial connection. The control and sensing system (CSS) recognizes acsii character sets sent by the VxWorks target. The communication between the target and the CSS is separated into three parts:

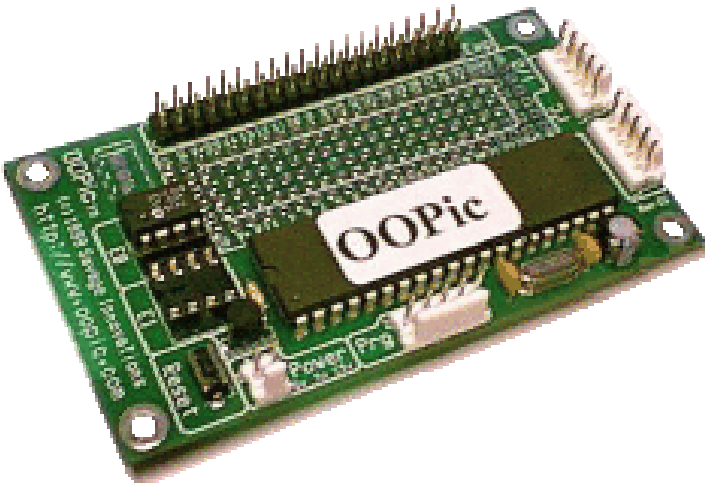


Figure 8: OOPic Micro-Controller

Initiation sequence:
On power up the CSS sends the target:

Ok
&

Upon the reception of "&" the CSS is ready to be controlled.

Camera control:
Camera control is the following:

>xxzz

Where acsii character ">" activates the control routine. Pan is controlled by "xx", which represents a two digit decimal number. Tilt is controlled by "zz", which also represents a two digit decimal number.

Pan values range from 10 (right limit) to 52 (left limit), with the value 31 being the center of rotation.
Tilt values range from 10 (top limit) to 52 (bottom limit), with the value 25 being parallel to the base.
Each character is echoed back to the target after being processed by the CSS.

Deployment Control:

Deployment control is the following:

>d (to start deployment)

s (to stop deployment)

Where acsii character ">" activates the control routine, and the acsii character "d" commands the deployment of the wings.

Each character is echoed back to the target after being processed by the CSS. The character "s" will be echoed when the CSS stops supplying power to the Signiture Memory Aloys (SMA) which deploy the wings of the MARV plane.

Serial interface:

The serial input and output signals from the OOPic are TTL level signals providing 0 and 5 volts. The TTL signals from the OOPic must be converted to +/-12 volts to register as serial communication on the VxWorks target. Conversion to RS232 signals was done with the MAX203 a TTL to RS232 signal converter chip, which will provide the voltage conversion to +/- 12 volts as well as providing the required signal inversion. The MAX203 is connected to the OOPic though I/O lines 22 and 23. The data transmitted by the OOPic is sent serially through the UART's transmit line located on I/O line 22. Data is received by the UART through the receive line located on I/O line 23.

Camera Control:

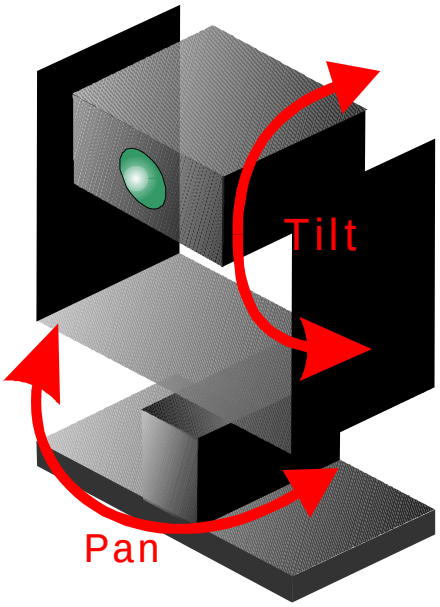


Figure 9: Camera control assembly

The camera control unit controls the tilt and pan rotation of a NTSC digital video camera (Figure 9). Two hobby servos are used to rotate the camera. The OOPic sends a Pulse Width Modulated (PWM) signal from I/O lines 31 and 30. By modulating the width or the pulse sent to the servos the servos can be precisely positioned to the desired location. The signal from line 31 controls the pan variable of the camera and the signal from line 30

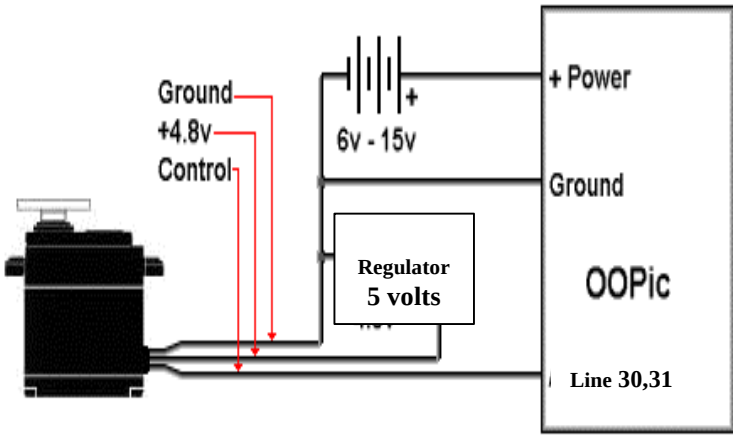


Figure 10: Servo connection diagram

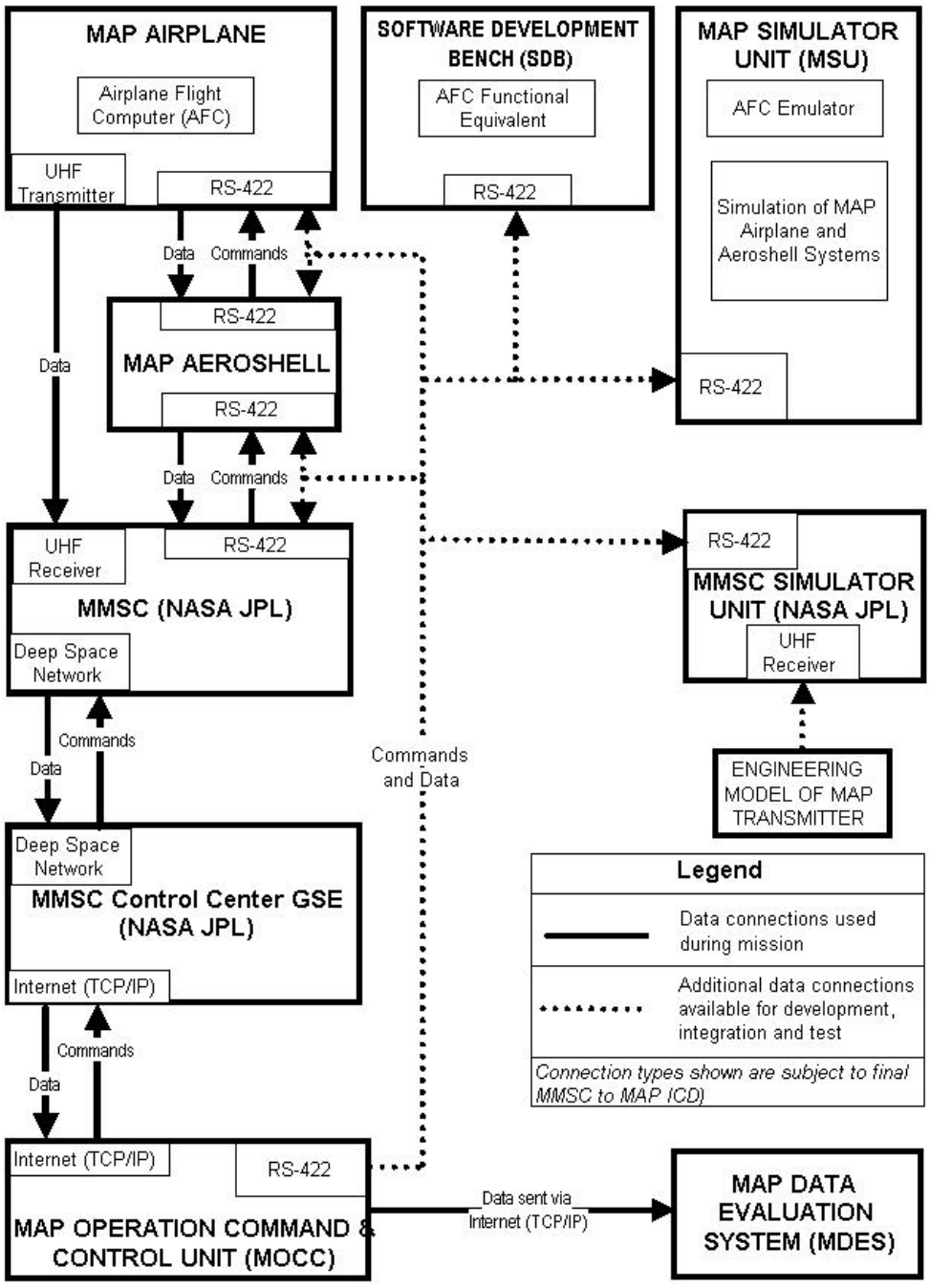
controls the tilt variable. Since the OOPic can only source 0.20 amps an external driver is needed to power the servos, this is done by using an LM780 to convert the 12 volt source used by the deployment controller to a 5 volt, 1 amp source to drive the servos (Figure 10).

Deployment Control:

The deployment of the MARV plane wings is controlled from the OOPic's I/O line 15. The signal from the OOPic is TTL and can only source 0.20 amps, while the SMA's used for wing deployment call for 10 amps. Therefore there needs to be additional circuitry to provide the 10 amps needed to deploy the wings. To source the power needed to provide the 10 amps two 12-volt dry cell battery are used. The TTL signal from I/O 15 of the OOPic is used to switch a relay, which closes the loop and allows the SMA's to draw 10 amps from the batteries. LM338s are used to regulate the current flowing from the batteries so that only 5 amps go to each of the SMA's.

MARV SOFTWARE

As with any advanced flight system such as the MAP, which requires various modes of operation, communication between components, and the ability to be reprogrammed after launch, one of the core underlying systems is the software. For a system such as the MAP, software is preferred, if not necessary, to perform such functions as real-time control and data processing. For the MARV project, only a subset of the MAP software architecture was necessary. When possible, the software for MARV was designed to implement the actual mission requirements as much as possible. The subset of functionality that we chose to design and implement involves the deployment system, the video imaging system, and the core software required to schedule multiple tasks, control the timing of mode transitions, and telemeter data. The way in which our software system fits into the system level data flow can be seen from the following diagram:



We are running the AFC software on a x86-based computer using the COTS real-time operating system VxWorks. We are connecting the AFC directly to the MOCC, which is a Unix application running on Solaris. The MDES is a Java applet, which can run on any host.

Imaging is a major focus of the MAP mission. Partly to address this important area, MARV has also focused on imaging. The Airplane Flight Computer (AFC) acquires image data (frames) and sends them via the MAP Operation Command and Control System (MOCC) to the MAP Data Evaluation System (MDES). One purpose of choosing video imaging was to explore the usefulness of a compression algorithm that only sends data that has changed since the last frame was sent.

For frame acquisition, we purchased a Hauppauge WinTV camera and framegrabber card based on the bt878. This affordable COTS solution has the benefit of available Linux drivers. Our first plan was to write a VxWorks driver for the card and acquire frames locally. When completing the driver became unfeasible, we resorted to doing the frame acquisition on a Linux x86-based computer. The frames are sent from the Linux machine to the AFC via a TCP/IP socket. In the AFC software, the frame grabber driver provides a top half interface as if the frames were acquired locally but the bottom half of the framegrabber driver actually receives the frames via the socket connection. On the AFC, a video imaging task consumes the frames by performing the change-only compression algorithm and sends the resulting data to the telemetry task for delivery to the GSE.

The idea of a change-only compression algorithm was explored by the MARV team in an attempt to reduce the amount of image data that would need to transmitted. The basic principle involved in this algorithm is that it is only necessary to transfer the data for a pixel if that pixel has changed since the last frame. The pixel is determined to be changed if the numerical difference is exceeds a predefined threshold. There is a tradeoff, however, between sending only the changed pixels and just sending the entire frame because when there are a large number of changed pixels, the amount of data that would need to be transferred would actually be greater than that for a normal full frame. This is because when a full frame is sent, it is not necessary to send the locations for each individual pixel because they are sent in order, without skipping any pixels. When sending the change-only data, however, the pixel location along with the actual image data needs to be transmitted. In our algorithm, we encode the location in 17 bits (for 320x240 pixels) for both color and grayscale modes. For the color mode we encode each of the three color components in 5 bits for a total size of 32 bits. For the grayscale mode, the lone intensity value is reduced to 7 bits for a total size of 24 bits.

The achitecture of the software was designed for the MAP mission. A subset of this achitecture has been implemented for the MARV project. The following modules have been implemented for MARV:

Application : Mission Operations : Mode Management

Application : Mission Operations : Deployment

Application : Science : Video Imaging

OS : I/O : Digital I/O

OS : I/O : Framegrabber

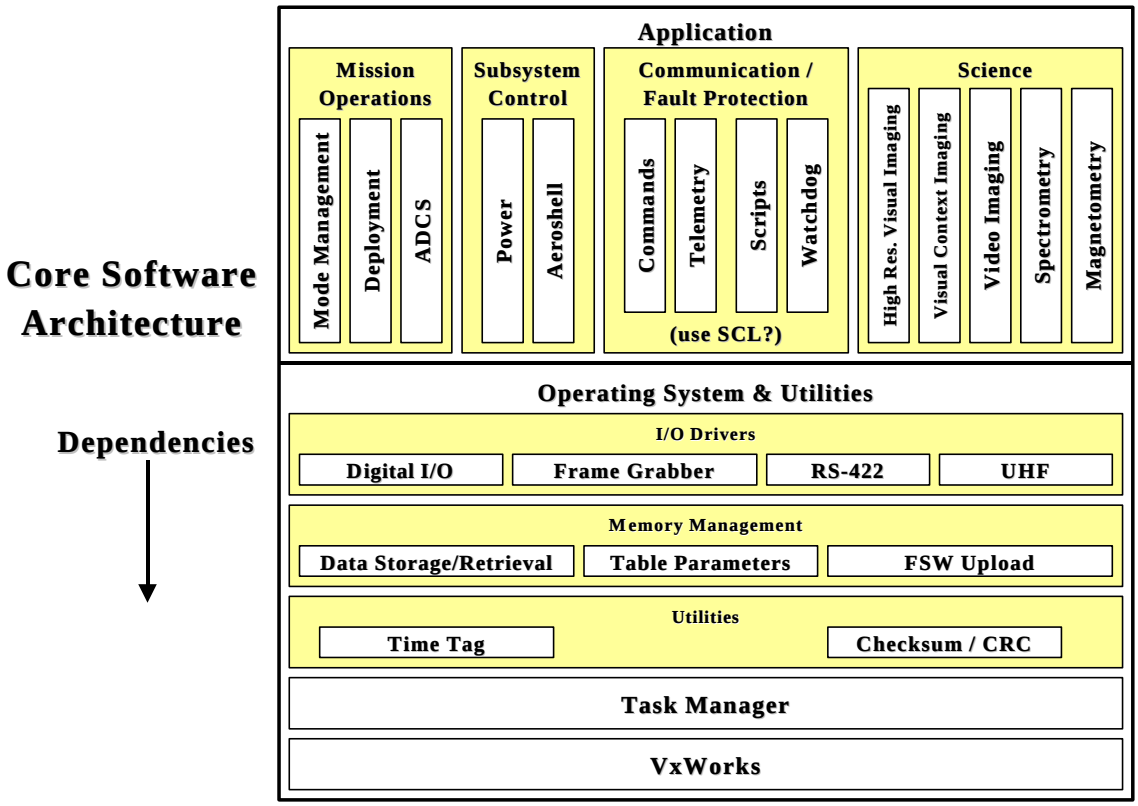
OS : I/O : Ethernet (instead of UHF / RS-422)

OS : Utilities : Time Tag

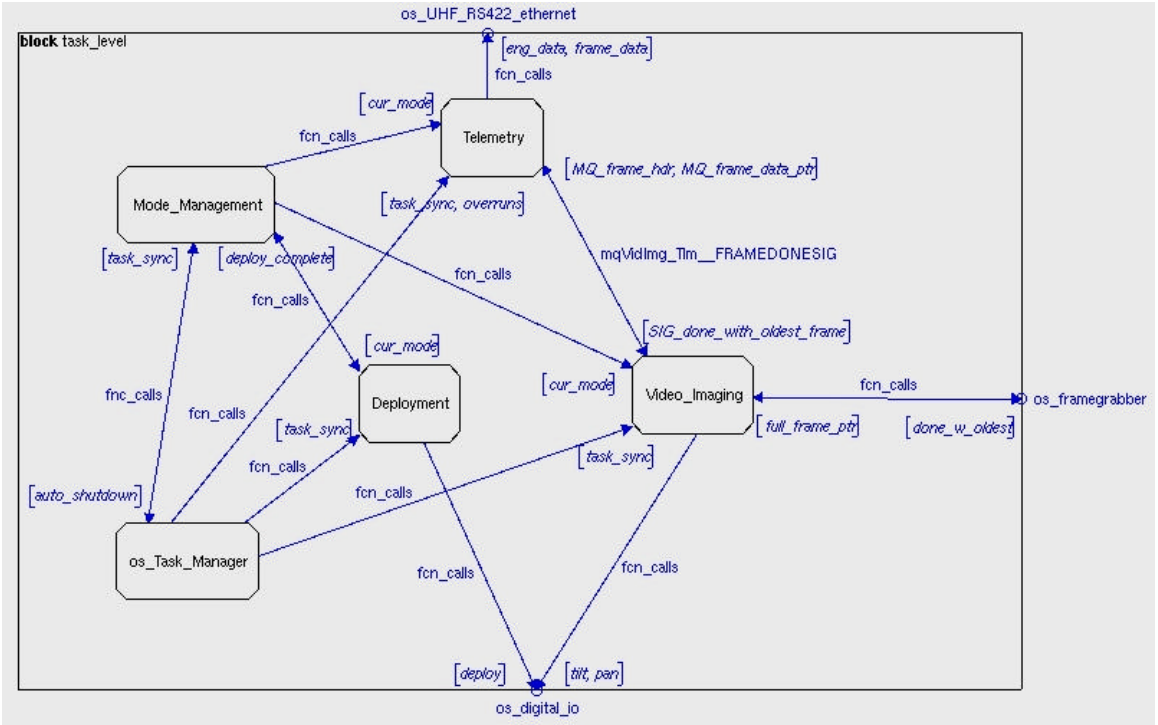
OS : Utilities : Task Manager

OS : VxWorks (configured)

A diagram of the core software architecture can be seen below. It illustrates the dependency hierarchy; os/utils are not dependent on application modules etc. As listed in the previous section, only some of the units on the core software architecture diagram have been implemented for MARV.



Another important figure from the detailed design of the AFC software is the task level data flow seen here:



This figure illustrates the data flow and relationships between all of the MARV AFC tasks. The external interfaces to the AFC are via the ethernet, framegrabber and the digital I/O.

The mode management task controls the mode timing and transitions between the following major modes: Manual, Aeroshell Deployment, Airplane Deployment, Level Flight and Final Flight. It provides the current mode to the other three aplication tasks. The deployment task handles the timing and commanding to control the airplane deployment sequence. The telemetry task gathers data from the other tasks and sends the engineering and frame data to the Ground Support Equipment (GSE). The video imaging task, as discussed previously, receives data from the framegrabber driver, performs the change-only compression algorithm and sends the data to the telemetry task. Actually, only pointers to the large image frames are sent between the tasks. Telemetry task receives a frame header and a ponter to data. The pointer is either to compressed data or a full frame. When done transmitting a frame, the telemetry task signals the video imaging task to announce that a frame can be discarded.

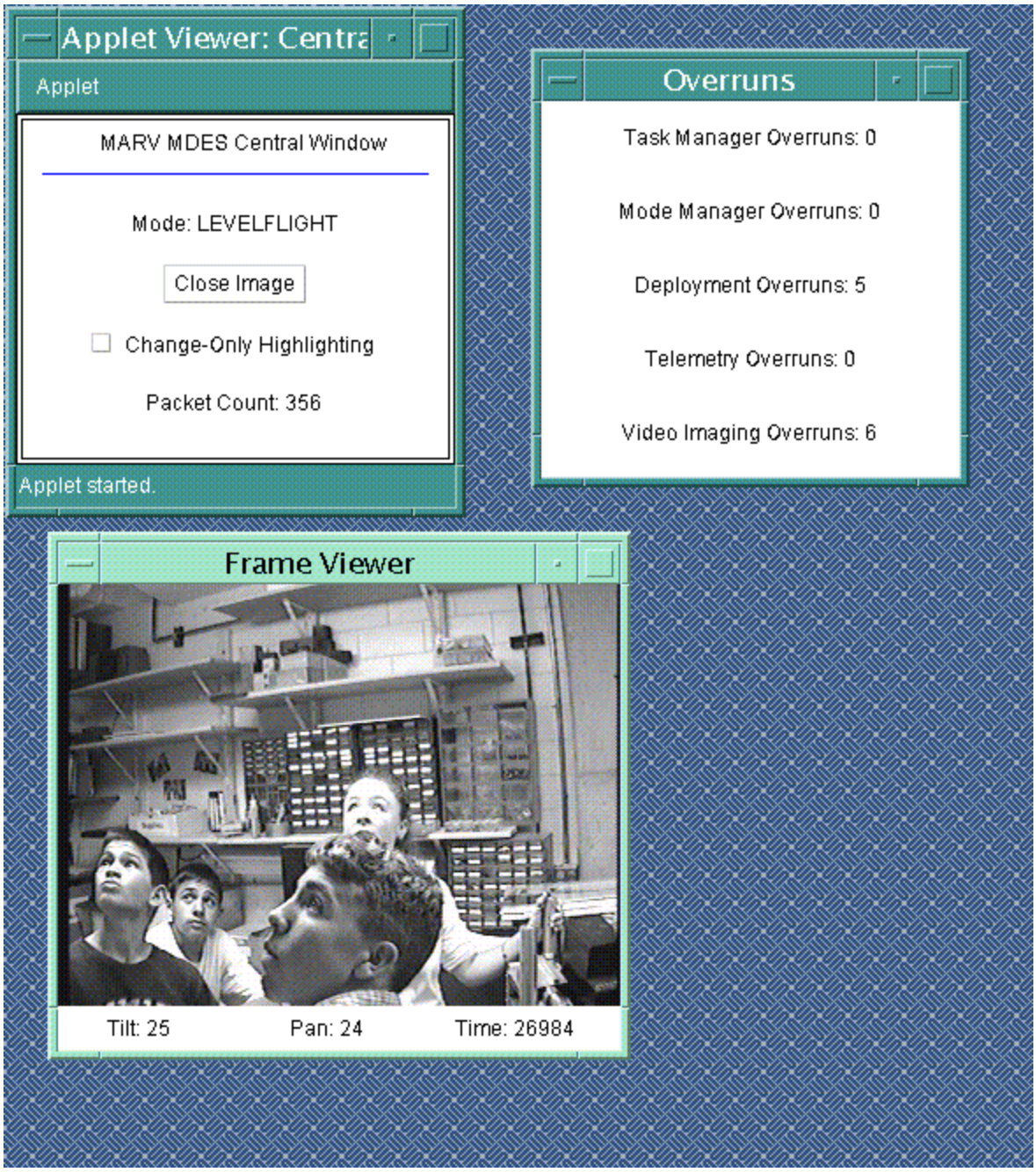
We also chose to focus on the GSE side of the mission and in particular the MDES. Connecting to the GSE to send command and control data to the MMSC is the MOCC. While we did not focus on the commanding of the MAP via the ground station, another responsibility of the MOCC is sending the MAP data sets to the MDES via internet. In our system, the MOCC connects directly to the AFC via a socket connection, and passes on all the data it receives from the AFC to the MDES. In both the mission requirements and our implementation, MDES is then responsible for reconstructing the MAP science and engineering data to its original form, displaying it, as well as storing it for later retrieval.

In our implementation, the MDES is a Java applet that is accessible via the Internet. Most web browsers currently cannot view it because they only support Java 1.1.5 at the latest, while the MDES includes classes from Java 1.2. Java's appletviewer is a viable alternative, however, and can be easily installed by any user.

The applet can retrieve its input data from any of three source types, depending on the parameters it is passed. The first such method is to make a socket connection to the MOCC in which, the data is input and processed in real-time. If the MDES gets behind in reading the data, the MOCC will drop data that it is unable to write to the socket, but the system is setup the reduce the possibility of that situation occurring. While in the socket mode, the MDES writes all the data that it reads to a file on the applet server, which can be accessed and replayed using one of the other two methods. Since the second and third input methods involve reading from stored data files, they are therefore not real-time like the previous method. Playing the frames back with the same frequency as they were recorded was not accomplished, but MDES does attempt to maintain accurate timing. These two methods get an input data stream from a URL and a local file, respectively, but do not store anything. The URL method works effectively while working on a high-speed network, but generally it is preferred to store the data file locally, and use the third method.

Once a data input stream is started for each of the three methods, the subsequent execution of the applet is virtually identical, disregarding any waits associated with reading the actual data. Each packet that is sent by the AFC contains a sync pattern that the MDES locates to synchronize itself. After reading the pattern, the MDES reads in the various engineering and science data from the packet header, which it will subsequently display for the user. If an image frame is included, it will use that data to update the video image viewer, reading either a full frame or decoding the change-only data. As stated, if the applet is reading from a socket it will then store the data for each packet to a file on the applet server.

The appearance of the MDES can be seen with the following screen shot:



One additional feature of the MDES display that needs to be described is the "Change-Only Highlighting" checkbox. With this component, the user is allowed to highlight the pixels that were sent as change-only data. Besides being an interesting feature, this component proved value in the process of designing and implementing the pixel compression algorithm, and gave a feeling for how much data reduction was being achieved. When the right balance was achieved, only pixels for objects on the screen that were moving would be sent.

Although there are sometimes limitations in execution speed by using a Java applet, it was a fairly easy platform choice. The two obvious factors that played into the decision were the portability achieved by using a Java implementation, and the ability to run the MDES from any computer on the Internet.

The system we created is limited by various factors inherent in our design choices and our execution environment. The first such limitation is caused by the 10 Megabit Ethernet network in which the targets are

located. Due to the high amount of data that we attempt to transfer in real-time, a faster network would possibly have allowed for a higher frame rate, closer to what was included in the original requirements for the MAP. While such an increase in network bandwidth would have allowed the Linux / VxWorks / Solaris machines to perform at closer to their processing capabilities, other factors involving the displaying of the video would also have come into play. Due to limitations in Java's execution speed as well as the inherent overhead in actually sending the video data to the user's screen, it is uncertain what the peak level of data rate / frame rate would be within our system.

Structured software engineering methods were followed in the development of the MARV software. We began with a requirements definition based strongly on MAP documentation from NASA Langley. High level design was completed along with various design reviews. During implementation, modules were incrementally tested and code reviews were conducted. We used the configuration management software CVS (Concurrent Versions System), which provides an unreserved checkout mechanism that facilitated simultaneous work on the software system.

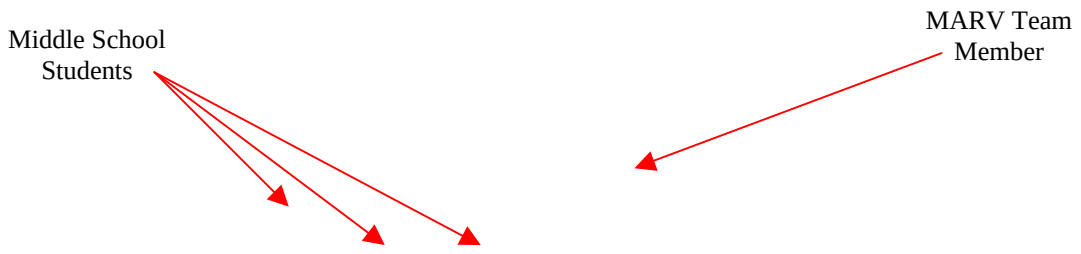
CONCLUSIONS

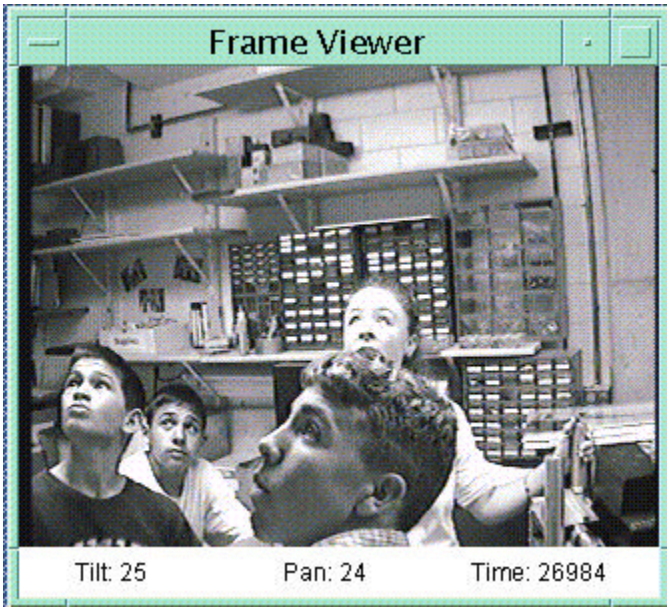
Wind tunnel testing was completed of a 1/4-scale model of the Mars Airplane (MARV), and the pitching moment was compared to a computer model prediction. The predicted pitching moment was found to be stable up to stall, and unstable at higher angles of attack. The wind tunnel data, however, showed static instability at every angle of attack from zero to 75 degrees. This instability is likely due to insufficient area in the horizontal stabilizers, as well as disturbed flow over the tail during testing. Further instability is due to the neglect of body lift in the predictions, as well as an aft cg position. This instability should be overcome through an increase in tail size, a forward movement of the center of gravity, and the use of separation springs to impart a downward pitching moment during aeroshell separation.

OUTREACH ACTIVITIES

The environment of the Senior Design Lab (SDL) class promoted outreach to other engineering students, industry, and the community. As students who participated in SDL, we were required to give multiple presentations pertaining to the progress of our project including a Conceptual Design Review, a Preliminary Design Review, a Critical Design Review, and a Delta Critical Design Review. These reviews were presented to other engineering students in the SDL class (approximately 50 students), faculty advisors, and the Engineering Advisory Council (EAC). The EAC consists of variety of engineers from industry that advise the College of Engineering on multiple issues regarding academics, student life, funding sources, and leadership organizations.

This spring two groups of students toured our laboratory facilities and were exposed to our design project. The first group consisted of approximately 150 high school students from disadvantaged areas in Colorado. Group number two was comprised of middle school students from both urban and rural areas. These students received a brief overview of the MARV project and a demonstration of the digital camera ability. The picture below (taken with the digital camera and captured with the MDES) shows part of the middle school group that toured our labs.





Finally, each spring during Engineering Days (E-Days) the University of Colorado Engineering Council (UCEC) sponsors a Design Expo. The MARV team attended this years expo which was held on April 29, 2000. This expo was open to all engineering design teams including chemical engineering, freshman projects, mechanical engineering, aerospace engineering, computer science, and electrical engineering. The community and industry representatives were also invited to attend the expo, and this year approximately 600 people attend, ranging from children to parents and from faculty to industry leaders.