

**TECHNICAL REPORT
NATICK/TR-04/010**



AD A 420 411

PROJECT LISTEN, COMPUTE, SHOW (LCS) - MARINE

by
Victor Zue*
John Wroclawski*
and
Michael Bolotski**

***Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139**

****The MicroDisplay Corporation
San Pablo, CA 94806**

February 2004

Final Report
June 1998 – June 2001

Approved for public release; distribution is unlimited

Prepared for
**U.S. Army Research, Development and Engineering Command
Natick Soldier Center
Natick, Massachusetts 01760-5056**

DISCLAIMERS

The findings contained in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of trade names in this report does not constitute an official endorsement or approval of the use of such items.

DESTRUCTION NOTICE

For Classified Documents:

Follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For Unclassified/Limited Distribution Documents:

Destroy by any method that prevents disclosure of contents or reconstruction of the document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 18-02-2004		2. REPORT TYPE Final Report		3. DATES COVERED (From - To) June 1998 - June 2001	
4. TITLE AND SUBTITLE PROJECT LISTEN, COMPUTE, SHOW (LCS) - MARINE				5a. CONTRACT NUMBER C-DAAN02-98-K-0003	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Victor Zue,* John Wroclawski,* and Michael Bolotski**				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) *Laboratory for Computer Science, Massachusetts Institute of Technology, 200 Technology Square, NE43-538, Cambridge, MA 02139 **The MicroDisplay Corporation, 3055 Research Drive, San Pablo, CA 94806				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Sponsor: Defense Advanced Research Projects Agency (DARPA) Microsystems Technology Office (Elana Ethridge) 3701 North Fairfax Drive Arlington, VA 22203-1714				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) NATICK/TR-04/010	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited					
13. SUPPLEMENTARY NOTES Monitor: US Army Research, Development and Engineering Command, Natick Soldier Center, ATTN: AMSRD-NSC-TP-S (H. Girolamo), Natick, MA 01760-5056					
14. ABSTRACT This document is the final report for Project LCS-Marine, carried out by MIT Laboratory for Computer Science and the MicroDisplay Corporation. The goals of the project were to design and evaluate the usefulness of an integrated information delivery system utilizing a) multimodal human-oriented conversational applications and b) a compact handheld wireless terminal device with cell-phone form-factor but rich multimodal (audio, visual, gesture) human interface capabilities. The report describes our work in the key project areas of conversational systems architecture, conversational systems applications, wireless terminal system software, wireless terminal hardware architecture, networking and communications research and development, high resolution micro-display research, development, and fabrication, micro-display optics and driver electronics, and handheld terminal packaging and form-factor considerations.					
15. SUBJECT TERMS MARINE CORPS LISTEN COMPUTER PROGRAM TECHNOLOGY TRANSFER INFORMATION NEEDS CONVERSATIONAL SYSTEMS HUMAN-MACHINE INTERACTION SPOKEN LANGUAGE VOICE ACTIVATED SYSTEMS WIRELESS COMPUTING DEVICES HAND-HELD DEVICES DIALOGUE SYSTEMS HUMAN INTERFACE DESIGN					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 105	19a. NAME OF RESPONSIBLE PERSON
a. REPORT UNCLASSIFIED IED	b. ABSTRACT UNCLASSIFIED IED	c. THIS PAGE UNCLASSIFIED IED			Thomas Gilroy
					19b. TELEPHONE NUMBER (Include area code) 508-233-5855

Table of Contents

List of Figures.....	v
List of Tables	vii
Preface.....	viii
1. Overview	1
1.1 Project Summary.....	1
1.2 Project Accomplishments	2
1.3 What Was Not Achieved	4
2. Conversational System Architecture: Galaxy Communicator	5
3. Development of Human Language Technologies.....	10
3.1 Speech Recognition	10
3.2 Language Understanding	11
3.3 Language Generation	12
3.4 Speech Synthesis.....	12
4. Application Development	14
4.1 Jupiter.....	14
4.2 Pegasus.....	16
4.3 Voyager.....	16
5. Handheld Device System Design.....	19
5.1 Overview.....	19
5.2 Hardware Design.....	19
5.3 System Software	24
6. Communications and Networking.....	35
6.1 Overview	35
6.2 Voice Communication Strategy	37
6.3 Adaptive Channel Multiplexing.....	38
6.4 Web Performance and the Cache Information Protocol	41
7. Handheld Device Development.....	43
7.1 Goal.....	43
7.2 Scope of Work, Approach, and Results	44
7.3 Mechanical Design and Packaging	46

Contents (Cont'd)

7.4	Displays and Drive Schemes.....	50
7.5	Drive Electronics	56
7.6	Optics	63
7.7	Display Controller ICs	70
8.	References.....	85
Appendices.....		89
A. Additional Technical Documentation (copies available upon request)		90
B. Papers (copies available upon request)		91
C. Theses (copies available upon request)		93

List of Figures

Figure 1. Galaxy Communicator architecture for conversational systems.....	6
Figure 2. Example spoken dialogue between Jupiter and a user.....	15
Figure 3. Organization of the SX- 2 System.....	20
Figure 4. WP-1 Unit in the desired configuration.....	47
Figure 5. Initial model of the WP-3 configuration.....	48
Figure 6. WP-3 Handset Industrial Design	48
Figure 7. Mockup of WP-4 Packaging. The unit is shown in a closed configuration and is compared to a Nokia 6100 series cell phone. An open configuration is shown in Figure 21.	50
Figure 8. Analog Display architecture	52
Figure 9. Microscope picture of typical early defect on the G4 substrate.....	54
Figure 10. An Example Sample Ramp Conversion. The switches start out closed, so that the column capacitor tracks the rising ramp. When the broadcast time value exceeds the digital value stored in register t1, the comparator opens the switch, thus holding the voltage V1 on the rightmost column. The leftmost switch operates similarly until the time exceeds t3.	56
Figure 11. EK-2 Evaluation Board. Approximate size is 12" x 6".	58
Figure 12. BeltBox-1 in its boxed configuration. Only the top analog board can be seen.	59
Figure 13. Side view of the BB-4 stack. The digital board is at the top. Most of the area is the large FPGA and the two frame buffer memories.	61
Figure 14. BeltBoard-5. The four visible components are the FPGA (in a BGA package), the frame buffer memory, the chip-on-board BeltChip and the PIC controller.....	62
Figure 15. Block diagram of a BB-5. Notice the discrete components between the FPGA and the display connection. All of these are scheduled to be included in the next generation display.....	63

Figures (Cont'd)

Figure 16. Cross-section of the 1st generation eyepiece	65
Figure 17. Optical model of mini projector. The view is looking down from above the user's head. The microdisplay is in the upper left corner. The parabolic mirror on the right side flips down to the horizontal plane. The viewer's eye is in the lower left with the very large exit pupil. The image, to the eye, appears to be floating behind the parabolic mirror at a 20-inch distance with a 16-inch diagonal.....	67
Figure 18. The prototype of the flip out curved display compared to the WP-4 mockup.....	67
Figure 19. Mangin-type filp display optical diagram.....	68
Figure 20. Fiber Fuse-based Optics	69
Figure 21. WP-4 Mockup with Flip-out wedge optic	70
Figure 22. Block Diagram of BeltChip	72
Figure 23. Block diagram of BC-1 output circuitry	74
Figure 24. The BC-1B test board is shown above. Notice the size of the package relative to the BeltChip. Also notice the amount of space occupied by the miscellaneous discrete components and video header to the right of BC-1B and the power supply section on the left.....	75
Figure 25. Block Diagram of the processor/display interface. The display controller snoops the main memory bus for write commands to the frame buffer and generates write operations	78
Figure 26. MC-2 Internal Block Diagram.....	79

List of Tables

Table 1. Performance evaluation of Jupiter for sentences collected from naive users.	16
Table 2. Measured CDPD Performance, MIT, Summer 1999	36

Preface

This report outlines the research undertaken by the Massachusetts Institute of Technology (MIT) Laboratory for Computer Science, Cambridge, MA, and The MicroDisplay Corporation, San Pablo, CA, to design and evaluate the usefulness of an integrated information delivery system utilizing a) multimodal human-oriented conversational applications and b) a compact handheld wireless terminal device with cell-phone form-factor but rich multimodal (audio, visual, gesture) human interface capabilities. The project was completed during the period June 1998 to June 2001 under contract number C-DAAN02-98-K-0003, under the direction of the U.S. Army Research, Development and Engineering Command, Natick Soldier Center, Natick, MA, and the sponsorship of the Defense Advanced Research Projects Agency (DARPA).

PROJECT LISTEN, COMPUTE, SHOW (LCS) - MARINE

1. Overview

1.1 Project Summary

LCS/Marine is a project aimed at serving soldiers' future information needs. It sets forth a radically new human-machine interaction paradigm (Listen, Communicate, and Show) in which the computer can interact with the user naturally via spoken language instead of keyboard. The interaction promotes mobility, since compute and knowledge servers can be connected to the user via a phone or a computer network.

A key component of the project is the development of a light weight, handheld device, referred to in this report as the WebPhone or LCS/Marine device, which combines an intuitive, cell-phone form factor with a high-resolution visual display, additional human-oriented I/O capabilities including audio and gesture input, and substantial local computational power. It is intended to be a concept demonstration for use by the Marines.

The responsibilities of the MIT Laboratory for Computer Science (LCS) in this project were:

- Design and implement the overall system architecture,
- Develop the spoken language technologies for dialogue interactions in several applications,
- Develop communications and networking algorithms for the system,
- Develop overall system specifications and architecture for the LCS/Marine device,
- Select the subcontractor for the handheld device, subject to the approval of the sponsors,
- Work closely with the subcontractor in the design and manufacturing of the handheld device, and
- Work closely with Lockheed-Martin to help them port the spoken language technologies to applications of interest to the Marines.

This final report describes the work performed and technical accomplishments made through the course of the project, as summarized below. Rather than repeating the material contained in the quarterly progress reports, this final report summarizes the research conducted during the funding period, with pointers to the specific publications. Published papers and additional technical

documentation are listed and included for reference. Theses are also listed, and are available upon request.

1.2 Project Accomplishments

This section summarizes the major accomplishments of the project.

- Design and implementation of a **conversational systems architecture**. The Spoken Language Systems (SLS) Group at LCS has designed, implemented, and distributed the Galaxy Communicator architecture for dialogue interaction.
- Development and demonstration of **three conversational applications**. The LCS SLS group has developed initial versions of:
 - **Jupiter**, an audio-only conversational system for obtaining weather information.
 - **Pegasus**, a multi-modal conversational system for obtaining airline flight status information.
 - **Voyager**, a multi-modal conversational system for interacting with a rich database of maps and navigational information.
- **Technology transfer** of MIT research in conversational systems. Members of the LCS SLS Group worked closely with Lockheed-Martin's Advanced Technology Laboratory on technology transfer, resulting in the successful demonstrations of spoken dialogue interaction in the logistics domain, in conjunction with the US Marines.
- Development and implementation of several **novel networking protocols and algorithms** to optimize performance of interactive conversational applications in a bandwidth-poor wireless environment. The LCS Advanced Network Architecture (ANA) group developed and implemented
 - **Adaptive Channel Multiplexing**, a protocol and algorithm for combining several network channels with poor and varying performance into a single virtual channel with much higher and more predictable performance.
 - **Sub-Atomic Web Caching**, a new approach to optimizing the display of WWW data in bandwidth-poor environments by caching *portions* of each object at the client, and reusing these portions to construct new objects.
 - The **Cache Information Protocol** and its related **Information Representation Format**, networked system protocols allow application programmers to make use of Sub-Atomic Web Caching.
- Development and implementation of a **software architecture and system software** for the LCS/Marine device. The LCS ANA group developed an overall system software

architecture for the LCS/Marine handheld device, and implemented or adapted necessary software components to support device operation.

- Specification, design, and fabrication of a **hardware architecture and hardware design** for the LCS/Marine device. The LCS ANA group, working with engineers from our subcontractor MicroDisplay corporation, developed an overall hardware architecture for the LCS/Marine device, developed detailed designs for each hardware component and PCM assembly, and oversaw the fabrication of these components by a contract vendor.
- Advances in **display technology and optics** required to support usable high-resolution miniature displays. Project subcontractor MicroDisplay Corporation substantially advanced the state of the art in small LCOS displays and related optical components through their work on LCS/Marine. Specific accomplishments include:
 - **LCOS Display architecture:** MicroDisplay developed two digital LCOS display architectures within the LCS/Marine project:
 - The first architecture, with an external ramp generation circuit, was refined under the program, and later formed the basis for an SXGA (1280x1024) projection product.
 - The second architecture, based on a novel integrated ramp generation circuit is scheduled to form the core of the next generation projection product. This architecture is expected to scale to true HDTV resolution of 1920 by 1080.
 - **LCOS Display design:**
 - Two generations of 800 x 600 display substrates were fabricated and successfully used.
 - Physical layout techniques were developed that resulted in flat substrates.
 - Silicon fabrication processes were debugged in two different silicon foundries (Chartered Semiconductor and Amkor) to improve yield and image uniformity.
 - Liquid crystal modes were developed that eliminate flicker and image sticking, improve contrast to 150:1, and improve reflectivity for higher brightness, but remain at 5 V operation.
 - Cell assembly techniques were developed to reduce the severity of Newton rings, thus improving image quality, while maintaining cell gaps of one micron.
 - **Optics Design:**
 - Development of improved, fixed-focus optical design, more compact than the design existing at the start of the program. Additionally, designed for

to assembly in 15 minutes from a small number of pieces; the previous design required over two hours to assemble.

- Five different optics alternatives to an external eyepiece were explored and prototypes were built for two of them.
- **Technology transfer** of MicroDisplay developments in miniature LCOS displays. SVGA LCOS display developed for LCS/Marine successfully commercialized by MicroDisplay, with over 12,000 units shipped into headset products, both consumer and professional.
- **LCOS Display Controller IC design.** Three integrated circuits were developed by MicroDisplay to support the LCS/Marine handheld device.
 - An integrated **analog drive IC (BeltChip-1)** was developed and successfully used in the final prototype.
 - An integrated **one-chip digital controller (MC-1)** was developed and used for the VGA video input prototype phase. Several generations were developed and tested, with the final one performing display signal conditioning in the digital domain.
 - An integrated **one-chip digital controller (MC-2)** was developed and tested in the final StrongArm memory bus phase. MC-2 re-used several modules from the output side of the MC-1 implementation.
- **Handheld Device design and fabrication.** Working with LCS and subcontractor Virtual Vision, Inc., MicroDisplay developed two generations (WP-1, WP-3) of a LCS/Marine handheld device design. The second generation incorporated human-factors lessons from the first design, such as a rotating eyepiece and repositioned mouse pointer. It also supported the full device functionality such as PCMCIA cards, flash, and batteries. It was also designed to handle thermal loading from the integrated electronics.

1.3 What Was Not Achieved

The principal unachieved milestone of the LCS/Marine project was the development and fabrication of the final handset design, referenced as WP-4. The objective of this phase of the project was to package the microdisplay, optics, and electronics in a practical and intuitive cell-phone form factor. A mockup mechanical and optical design proposed by MicroDisplay subcontractor Virtual Vision (shown in Section 7.3.3 and Section 7.6.6) met the size target of the initial proposal and offered a usable optical design with strong human factors. However, development of a new all-digital display and conversion of the display controller FPGA into an ASIC form, coupled with a further round of electronics shrink and repackaging, with have been necessary before a working prototype could be built. Technical and management personnel and additional resources needed to complete these steps were not available within the scope of the contract.

2. Conversational System Architecture: Galaxy Communicator

This section describes the overall architecture and key components of the Galaxy Communicator system. This system implements the core conversational system capabilities that underpin the LCS Marine project.

2.1.1 Design Overview

Through our experience over the last decade in designing conversational systems, we have come to realize that an essential element in being able to rapidly configure new systems is to allow as many aspects of the system design as possible to be specifiable without modifying source code. To this end, we have redesigned our core architecture to support complex system configurations controlled by a run-time executable scripting language.¹ Using this new framework, we have been able to configure multi-modal, multi-domain, multi-user, and multilingual systems with much less effort than previously. We can now configure systems whose capabilities are well beyond what was previously considered feasible.

The resulting new architecture, Galaxy-Communicator [1], has been designated as the common architecture for the multi-site DARPA Communicator project in the United States (see Figure 1). A main goal of this program is to promote resource sharing and plug-and-play interoperability across multiple sites for the research and development of dialogue-based systems. MIT was given the responsibility of developing the architecture, which is being maintained and distributed from MITRE. At this moment, the Galaxy Communicator architecture and the associated library have been made Open Source, and more than 1,000 downloads have been made thus far.

¹ The development of the Galaxy Communicator architecture and the human language technologies have been co-funded by the DARPA Communicator Program sponsored by the Information Technology Office.

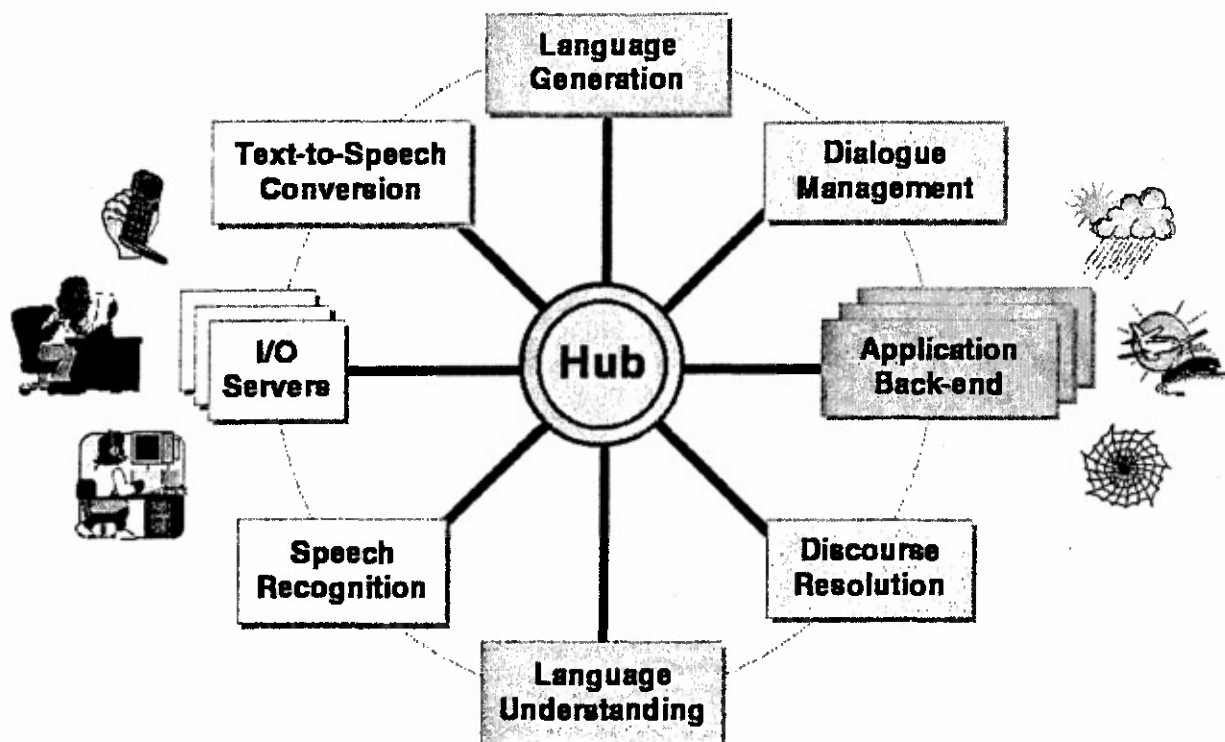


Figure 1. Galaxy Communicator architecture for conversational systems.

Galaxy-Communicator differs from its predecessors mainly in two ways: (1) a central hub handles all communications among the various servers via a standardized protocol, and (2) system control flow is maintained through a specialized run-time executable programming language interpreted by the hub. Galaxy was first introduced in 1994, as a client-server architecture for accessing on-line information using spoken dialogue [2]. Since then, it has served as the test bed for our research and development of human language technologies, resulting in systems in different domains (e.g., automobile classified ads [3], restaurant guide [4] and weather information [5]), different languages [6], and different access mechanisms [3; 4; 5]. In 1996, we made our first significant architectural redesign to permit universal access via any web browser [7]. The resulting Web-Galaxy architecture makes use of a "hub" to mediate between a Java GUI client and various compute and domain servers, dispatching messages among the various servers and maintaining a log of server activities and outputs.

In the process of developing dialogue control modules for various domains in Galaxy, we came to the realization that it is critical to be able to allow researchers to easily visualize program flow through the dialogue, and to flexibly manipulate the decision-making process at the highest level. To this end, we developed a simple high-level scripting language that permits boolean, string, and arithmetic tests on variables for decisions on the execution of particular functions. A domain-dependent dialogue control table specifies a set of sequential rules in this scripting language.

Generally, multiple rules fire in the course of a single turn. We found this mechanism to be very powerful, and were successful in incorporating it into our domain servers for weather, flight status, traffic and navigation information. We then began to contemplate the idea of

incorporating an analogous mechanism into the program control of the entire system, which was being maintained by the Galaxy hub. At about the same time, discussions were beginning on the possibility that Galaxy be designated as the reference architecture for the DARPA Communicator Program. It seemed possible for a scripting language, modeled after the dialogue tools developed for our domain servers, to support a programmable hub for the DARPA Communicator.

In the design of Galaxy-Communicator, we retained the notion of a central hub, but regularized the communication protocol between the hub and all servers, permitting users to configure "hub scripts" to easily specify the flow of information among servers performing their specialized tasks in the course of a dialogue turn. Analogous to the dialogue control table, sequential rules fire based on tests on hub variables. The hub variables are represented in a data structure that we call a "frame," which permits typed variables (e.g., string, integer, float, binary, and (recursively) frame) to be packaged together, manipulated, and transmitted.

An interesting research issue also addressed here is how the complex tasks of mixed-initiative dialogue systems should be partitioned into a set of semi-autonomous servers, each of which has clearly assigned roles. If the community intends to experiment with plug-and-play options, then it will be important to partition the space into servers in a consistent way. It is logical to define separate servers for speech recognition, natural language understanding, natural language generation, and speech synthesis. However, the components that deal with context resolution, response planning, and database retrieval are not necessarily organized the same way by different groups of researchers. In the systems we have thus far designed at MIT, the task of "turn management" is handled by a suite of domain-specific servers, as mentioned previously. Each of these servers is controlled by a separate dialogue control table. However, a single database server takes care of database needs for all of the domain servers, with capabilities of consulting both the Web and local relational databases. The turn managers routinely consult the database multiple times in the course of resolving a single user query. Discourse inheritance is managed separately from turn management, and the context record is updated after both the user turn and the system turn. All domains are handled by a single generic server, but controlled by domain-specific discourse tables [4].

2.1.2 The Hub Scripting Language

The Galaxy-Communicator system consists of a central hub that controls the flow of information among a suite of servers, which may be running on the same machine or at remote locations. The hub interaction with the servers is controlled via a scripting language. A script includes a list of the *servers*, specifying the host, port, and set of operations each server supports, as well as a set of one or more *programs*. Each program consists of a set of *rules*, where each rule specifies an *operation*, a set of *conditions* under which that rule should "fire," a list of input and output *variables* for the rule, as well as optional store/retrieve variables into/from the discourse history.

When a rule fires, the input variables are copied into a *token* and sent to the server that handles the operation. The hub expects the server to return a token containing the output variables at a later time. There is the option of no output variables, in which case interaction is one-way only. The input and output variables are all recorded in a hub-internal *master token*. The discourse history will also be updated, if the rule has so specified. The conditions consist of simple logical,

string, or arithmetic tests on the values of the typed variables in the master token. The hub communicates with the various servers via a standardized frame-based protocol.

Each individual user is associated with a unique *session*; user state information, such as the current language, domain, etc., is recorded via *session variables*. Each session is usually associated with a particular GUI and/or audio server. Discourse context is organized utterance-by-utterance within a session. Variables can be passed among different tokens associated with the same session via a device of prepending "hub session" to the key's name. Tokens associated with different sessions compete for available resources, and are queued up by the hub when requested servers are busy. The hub automatically garbage collects tokens when they are no longer active.

2.1.3 Program Flow Control

A simple communication protocol has been adopted and standardized for all hub/server interactions. Upon initiation, the hub first handshakes with all of the specified servers, confirming that they are up and running and sending them a "welcome" token that may contain some initialization information, as specified in the hub script. The hub then launches a wait loop in which the servers are continuously polled for any "return" tokens. Each token is named according to its corresponding program in the hub script, and may also contain a rule index to locate its place in program execution, and a "token id" to associate it with the appropriate master token in the hub's internal memory. The rule is consulted to determine which "OUT" variables to update in the master, and which variables, if any, to store in the discourse history or the log file. Following this, the master token is evaluated against the complete set of rules subsequent to the rule index, and any rules that pass test conditions are then executed. A top-level flag controls whether the program is to run in "single-threaded" or "multi-threaded" mode, where the former permits only a single rule to fire and the latter immediately executes *all* rules that fire. Servers other than those that implement user-interface functions are typically stateless; any history they may need is sent back to the hub for safekeeping, where it is associated with the current utterance. Common state can thus be shared among multiple servers. Furthermore, state is insensitive to server crashes.

To execute a given rule, a new token is created from the master token, containing only the subset of variables specified in the rule's "IN" variables. This token is then sent to the server assigned to the execution of the operation specified by the rule. If it is determined that the designated server is busy (has not yet replied to a preceding rule either within this dialogue or in a competing dialogue) then the token is queued up for later transmission. Thus the hub is in theory never stalled waiting for a server to receive a token. The hub then checks whether the server that sent the token has any tokens in its input queue. If so, it will pop the queue before returning to the wait loop.

2.1.4 Semantic Frame Representation

We expect that researchers utilizing the Galaxy system will be developing servers which will need to interface with a suite of existing servers already in place. In such cases, it is necessary for the servers to share a common language in the representations they jointly process. Researchers who choose to replace *all* the servers are free to use whatever meaning

representations they find convenient. However, if the intent were to replace a subset of servers, for example, a new dialogue manager or a new language generation server, then the new server would have to adopt the meaning representation protocol that was in use by the replaced component. Thus we think it is appropriate to provide a brief description of the meaning representation formats that have been adopted by our systems.

In the process of developing conversational systems in multiple domains over the last decade, we have constructed a minimal linguistic specification of a meaning representation that we feel is adequate for most applications of interest to us. Our TINA system [8] converts recognizer hypotheses into *semantic frames* in this format, and our context tracking component [9] depends critically upon this format for proper functioning. Our GENESIS [10] system can paraphrase semantic frames into multiple languages, not just natural languages but also into SQL, into a flattened "E-form" representation, and into waveform concatenations for our ENVOICE speech synthesizer [11].

We view the linguistic/semantic world as consisting of three main types of constituents, which we call *clause*, *topic*, and *predicate*. A clause constituent generally occurs at the highest level, and usually represents the high level goal of the user request, which could be, for example, "display," "record," "repeat," "reserve," etc. Topics generally correspond to noun phrases, and predicates are typically attributes, which could be expressed as verb phrases, prepositional phrases, or adjective phrases. A semantic frame is, then, a named and typed structure, with one of the above three types. Semantic frames also contain *contents*, and there is a library of tools available for manipulating the contents. Traditional linguistic contents include an optional topic and zero or more predicates. A frame can also contain a set of (key: value) pairs, where the key can be any symbol-string, and the value is one of: (1) an integer, (2) a string, (3) a semantic frame, and (4) a list of values in categories (1)-(3). We use the (key: value) notation for syntactic features such as number and quantification; there is also a distinguished "name" key for named entities. The (key: value) notation is very generic, and it has allowed us to represent almost any information we need to record, most especially database retrievals, in semantic frame format. For instance, the key "airline" has the value "United" as retrieved from the database. In fact, the token that is sent between the hub and the servers is also itself a [degenerate] instance of a semantic frame, although at the highest level it only utilizes the (key: value) feature of the frame.

3. Development of Human Language Technologies

Conversational systems require the integration of several different human language technologies. In addition to developing the architecture for combining these technologies, we have continued to make progress in each individual area as well. The following sections describe some of the improvements we have made in the areas of speech recognition, language understanding (including discourse and dialogue), and language generation. This is followed by a brief description of the three applications that we have developed for demonstrating the underlying technologies.

3.1 Speech Recognition

The SUMMIT speech recognizer developed by our group uses a segment-based framework for its acoustic-phonetic representation of the speech signal. Feature vectors are extracted both over hypothesized segments and at their boundaries for phonetic analysis. The resulting observation space (the set of all feature vectors) takes the form of an acoustic-phonetic network, whereby different paths through the network are associated with different sets of feature vectors. We have implemented a probabilistic framework which allows us to compare different paths in our segment lattice by considering the entire acoustic observation space. The approach we have adopted is to add an extra lexical unit which matches with all segments which do not correspond to one of the existing units, i.e., all sounds which are not a phonetic unit, as they are too large, too small, overlapping, etc. It can be shown mathematically that this approach can be implemented by only considering the segments in a particular segmentation. This is done by normalizing the phonetic likelihood of a particular segment by the likelihood of the "not-unit" for that segment.

Another innovation achieved during this contract is the recasting of our recognizer into a finite-state transducers (FSTs) framework. FSTs are a very powerful representation for many of the constraints and operations used in a spoken language system [12; 13]. An FST consists of states and transitions, where a transition can have an input label, an output label, and a weight (often a probability). We have developed a fairly comprehensive FST library and set of tools, allowing the search components of the SUMMIT speech recognition system to operate on FSTs [14].

Examples of constraints that can be represented by various kinds of FSTs include language models, lexicons, phonological rules, and context-dependent labeling. The lexicon can be represented by an FST. Other examples include system output such as *N*-best lists, word/phone graphs, recognition paths, etc. By representing a diverse set of constraints and outputs in a common framework, powerful algorithms can be applied to any of them and they can be combined in interesting and unforeseen ways.

Constraints at different levels can be combined using an operation called composition. For example, SUMMIT typically operates on the following cascade of FSTs: $U = C \circ P \circ L \circ G$. Here, *C* represents a transducer relating context-independent phonetic labels on the right to context-dependent labels on the left, *P* the phonological rules, *L* the lexicon, and *G* the grammar or language model (e.g., *n*-gram or context-free grammar). Thus, the composition of the cascade of FSTs results in a *single* FST *U* relating context-dependent phonetic labels to word strings,

including the combination of all relevant constraints and their probabilities. Note that composition can be performed on-the-fly, meaning that only the parts of the result that are explored will be computed. SUMMIT's forward search component operates on such an FST representing all constraints and probabilities. In subsequent search passes, we often make use of additional FSTs, most notably to change the language model to a higher-order n -gram.

We have built up a fairly comprehensive set of tools and C++ API to construct and manipulate FSTs. Operations include determinization, minimization, epsilon-removal, composition, union, concatenation, and closure, many of which can be performed on-the-fly. Overall, the use of FSTs have been a big win for us. We have gained tremendous flexibility, while at the same time we have been able to make our systems more accurate and faster. The Jupiter weather information system is one such system benefiting from the use of FSTs during recognition.

3.2 Language Understanding

All of our spoken language systems make use of TINA for natural language understanding [8]. TINA is a probabilistic parser which operates top-down, using a stack-based control strategy which favors the most likely analyses. It is designed so that its grammar rules and associated probabilities can be automatically trained from a set of parsable sentences. This approach has many advantages, including ease of development, portability, and most important for use with a speech recognition system, low perplexity. We have, in fact, shown empirically that grammar probabilities can substantially reduce the perplexity of the resulting language model.

TINA uses context free rules and a feature passing mechanism to provide the maximum constraint at the lowest cost. TINA has a layered approach: the upper layers of rules are domain-independent, capturing notions like wh-question, imperative, subject, etc. The lower layers are written to capture the particular semantic categories that can occur. As a result, the category names of the parse tree carry all the information, both syntactic and semantic, necessary to interpret the meaning of the sentence. TINA employs a "robust parsing" strategy that attempts to piece together an understanding of the utterance from analyses of fragments. This strategy has significantly improved the system's ability to understand spontaneous speech that is often agrammatical.

One of the main areas of research in language understanding undertaken in this contract was in the area of discourse and dialogue. We explored several alternatives for discourse and dialogue modelling in the context of our Galaxy Communicator conversational systems since it was initially unclear to us where discourse should be maintained. Our goal was to achieve the principle that domain-dependent aspects should be maintained in external tables, with the program's role being to interpret the tables. Our systems now have the capability of handling most discourse phenomena that occur in the domains we have developed, controlled by a declarative discourse table that specifies inheritance properties for all classes within all domains. Note that the discourse module was also designed to be able to interpret attention-focussing multi-modal input (clicking as well as speaking).

3.3 Language Generation

During this contracting period, we have implemented a new version of our language generation component, Genesis, to resolve many of the original system's idiosyncrasies and shortcomings. The resulting system, Genesis-II [15], has fulfilled our main goals, which were to provide very straightforward methods for simple generation tasks, while also supporting the capability of handling more challenging generation requirements, such as movement phenomena, propagation of linguistic features, structural reorganization, generation from lists, and the context-dependent specification of word sense.

In Genesis-II, generation is controlled by a set of generation rules in conjunction with a lexicon mapping terminal strings to their surface form realization. In defining the syntax of the generation rules, we have focused on creating an expressive language with generalized mechanisms. This was achieved by carefully designing notations and commands in a generic way, so that they would enjoy wider utility. Thus, the same generation mechanism is used for clauses, predicates, topics, and keywords. Furthermore, users can explicitly (but recursively) specify the ordering of all constituents in the target string, allowing for the correct generation of strings that were impossible to generate in the original system. Similarly, notations and commands were carefully designed such that their utility would extend beyond the original requirement to other related aspects. For example, we added selectors to allow for context-sensitive word-sense disambiguation, which were also effective in prosodic selection for speech synthesis needs.

Genesis-II also provides generic mechanisms for handling conjunctions and wh-query movement. Since the mechanisms for wh-movement involve reorganizing frame hierarchies, they also turned out to be useful in the generation of foreign languages with substantially different syntactic organizations. Genesis-II allows frames to be grouped into class hierarchies associated with common generation templates, resulting in a significant size reduction over the corresponding rules in Genesis-I. For example, we were able to reduce the rule files for SQL in the Jupiter weather domain by 50%. This generality also leads to more efficient porting to new languages.

We have thus far used Genesis-II in a number of specific domains and languages, both formal and natural. In our Jupiter domain, weather reports are being translated into three languages besides English: Spanish, Japanese, and Mandarin Chinese. The quality of the translation is greatly improved over what could have been achieved using the original Genesis system. In the realm of formal languages, we use Genesis-II both to convert a linguistic frame into an E-form and to generate database queries, often represented in SQL. In our Mercury flight reservation domain, common generation rules are used for both speech and text generation, where selected entries in the speech lexicon can map directly to pre-recorded waveforms, selected for prosodic context when appropriate.

3.4 Speech Synthesis

For nearly a decade, we have been using a commercially available, off the shelf speech synthesizer (DECTALK) for delivering the verbal message to the user. While the resulting synthetic speech is quite intelligible, it lacks naturalness. As a result, we have recently developed

a corpus based speech synthesis system called ENVOICE [11], which delivers very natural sounding speech for the applications that we have developed.

4. Application Development

We proposed to implement three applications using the Galaxy Communicator architecture and the human language technologies that we have developed. In this section, we will describe briefly these three applications, one dealing with weather, one dealing with flight status, and one dealing with urban navigation.

4.1 Jupiter

Jupiter is a telephone-only conversational interface to weather information for more than 500 cities worldwide, and can provide information on a variety of general weather conditions, temperature, wind, unusual situations, advisories, etc. [5]. The weather information is obtained from four on-line sources on the Web, and is updated several times daily. Jupiter employs the Galaxy Communicator architecture, in which a user interacts with the system using a telephone. It serves as a platform for investigating several research topics. First, by using the telephone, we can empower a much larger population to access the wide range of information that is becoming available. In the scenario that we are putting into place, a user conducts "virtual browsing" in the information space without ever having to point or click, or even have a computer nearby. Second, displayless information access poses new challenges to conversational interfaces. If the information can only be conveyed verbally, the system must rely on the dialogue component to reduce the information to a digestible amount, the language generation component to express the information succinctly, and the speech generation component to generate highly natural and intelligible speech. Third, channel distortions place heavy demands on the system to achieve robust speech recognition and understanding. Finally, by applying human language technologies to understanding the "content," in this case the weather forecast, we can manipulate and deliver exactly the information that the user wants, no more and no less.

Weather information retrieved from the World Wide Web in textual format is processed by our natural language understanding component in order to achieve precise information delivery. The grammar is used to parse a new set of typically tens of thousands of sentences at every update, collected automatically. Typically, over 99% of unseen material parses successfully. These data are then used to update a relational database stored locally, and to answer specific questions.

To facilitate data collection, we have installed a toll free telephone number and made Jupiter continuously available so that naive users can call and inquire about the weather. As a consequence we have accumulated a large number of spontaneous utterances for training from real users. For example, over a two year period, we collected over 180,000 utterances from over 30,000 users, all without widely advertising the availability of the system. After only a few months of data collection, we had collected over 12,000 utterances for training purposes. These data have proved to be invaluable for system development, training, and evaluation. is an example dialogue between a naive user and Jupiter.

Jupiter:	Hi! Welcome to Jupiter, the MIT Lab for Computer Science weather information system. This call is being recorded for system development. You may hang up or ask for help at any time. How can I help you?
User:	CAN YOU TELL ME WHAT'S THE WEATHER LIKE IN BOSTON TODAY
Jupiter:	In Boston this afternoon. Partly sunny early. Then becoming cloudy with 30 percent chance of showers toward evening. Tonight: rain. Possibly heavy at times. Near 100 percent chance of rain. Can I help you with something else?
User:	YES WHAT ABOUT SAN FRANCISCO
Jupiter:	In San Francisco today: patchy fog near the ocean morning. Otherwise mostly sunny. Tonight: areas of coastal fog and low clouds. Otherwise fair. Is there something else I can do for you?
User:	YES HOW COLD IS IT IN DALLAS TEXAS TODAY
Jupiter:	In Dallas today: high upper 40s. Tonight: lows near 32. What else would you like to know?
User:	I'D LIKE TO KNOW WHETHER IT'S SNOWING IN DETROIT

Figure 2. Example spoken dialogue between Jupiter and a user.

Jupiter is a system that is under constant development. From time to time, however, we evaluate its performance on unseen test sets. We will report a snapshot of Jupiter's performance two years into its existence. The test data, consisting of 2,507 utterances, represents a collection of calls randomly selected over our data collection period. Of these, 2,003 were free of partial words, cross-talk, etc., and 1,793 were considered to be "in vocabulary" in that they contain no out-of-vocabulary words. About 72% of the in-vocabulary sentences (1,298) were from male speakers, about 21% (380) were from females, and 7% (115) from children.

Table 1 shows the system performance two years into the development of Jupiter. Out of a total of 2,507 recorded utterances, 269 contained no speech, and were eliminated from further consideration. Jupiter was able to answer nearly 80% (1,755 out of 2,238) of the utterances. The word and sentence recognition error rates for this subset were 13.1% and 33.9%, respectively. The corresponding utterance understanding error rate is 21.2%. Note that many utterances containing recognition errors were correctly understood. Approximately 5% (105 out of 2,238) of the utterances were rejected.

Table 1. Performance evaluation of Jupiter for sentences collected from naive users.

CATEGORY	ERROR RATE (%)
Word Recognition	13.1
Sentence Recognition	33.9
Understanding (from speech)	21.2

4.2 Pegasus

Pegasus is a conversational system that can answer natural queries about flight status for some 4,000 daily flights between some 50 cities in North America (e.g., "Are there any flights from Chicago to Boston arriving around 2 p.m.?"). The information is real, and is updated once every two minutes. Flight status information is kept current on flights throughout the entire calendar day. Users can find on-time information for any flight that has taken off and/or landed on the particular day they ask. Like Jupiter, Pegasus is available to general users via a toll free telephone number. The flight status information is integrated with flight schedule obtained from a SABRE database in order to offer on-time status.

We have developed more sophisticated mechanisms for dealing with errors caused by misrecognition of flight numbers. We were able to implement fairly complex algorithms quickly, directly as a consequence of the hub scripting capabilities of Galaxy Communicator. When the system proposes a query containing an airline and flight number that is not in our schedule or status databases, the N-best list is set aside while the system engages the user in a clarification sub-dialogue to elicit a source and destination city. The system then queries the database for a set of flight numbers consistent with those two cities and re-filters the N-best list based on these new flight numbers. After this short sub-dialogue and using this intelligence, the system is able to correct a flight-number misrecognition without ever revealing to the user that a mistake was made.

4.3 Voyager

A significant effort has been placed on the development of a new and improved Voyager system. Voyager has been one of the cornerstone systems of the Spoken Language Systems group since its inception in 1989 [16; 17]. The basic function of the system is to provide tourist and travel information for the city of Boston. In the past, the system has focused on static map-based information, allowing the user to ask about sites and landmarks, view maps, and obtain directions from place to place.

During this funding period, the Voyager system has undergone a major overhaul in two main areas. First, the system has been completely ported to the new Galaxy Communicator spoken language system architecture [1]. Second, it has been augmented with new, dynamic information about current traffic conditions (as provided by SmartRoute Systems). These two changes have allowed the system to become viable for use by the general public over a standard telephone line.

In developing the new Voyager system, various technical challenges have emerged. First, the complexity of the queries that can be asked of Voyager exceeds the complexity of queries

typically asked of our other systems [5]. This required that the flat key-value dialogue representation used in our previous systems be expanded to handle lists and hierarchical constructs. With this augmented dialogue structure complex requests, such as "Show me a map with museums and historic sites in Boston, Brookline and Cambridge", can be handled.

Using the request stated above the following semantic representation is generated for use by the dialogue manager and database server:

```
{c eform
  :action "display"
  :list ( {q list_item
           :category "museum" }
    {q list_item
     :category "historical_site"
     :in ( {q list_item
            :category "city"
            :name "Boston" }
      {q list_item
       :category "city"
       :name "Brookline" }
      {q list_item
       :category "city"
       :name "Cambridge" } ) } ) }
```

The inclusion of a hierarchical structure and lists in this query allows for a straightforward representation of the complex query which, despite its increased complexity, is easily handled by the back-end components of the system (i.e., the dialogue manager and the database server).

The particular query also contains an example of semantic ambiguity, which the turn manager is responsible for resolving. The hierarchical structure indicates that the parser interpreted the query as a request for "museums" first and then "historic sites in Boston, Brookline, and Cambridge" second. Although this particular parse is technically an acceptable interpretation, the more appropriate interpretation would attach the "in Boston, Brookline, and Cambridge" prepositional phrase to both "museums and historic sites". Currently, the turn manager has a set of heuristic rules for detecting and correcting these types of parse ambiguities.

Another technical challenge of Voyager is the ability to generate short and concise traffic reports from the traffic information reports provided by SmartRoute. The traffic reports are distributed in small capsules, each reporting the traffic conditions for a particular roadway segment. Typical users will ask for a traffic report covering multiple roadway segments. The challenge is to aggregate the traffic reports from these roadway segments without generating excessive or redundant information. To provide an example, consider the reports generated for the following user query, "What is the traffic like on Route 2?" The database returns reports for four different segments of road for this query. If the four reports are generated independently then the user might hear the following:

On Route 2 heading east from Route 495 to Route 95 the traffic is moving freely with an average speed of 41 miles per hour. On Route 2, heading east from Route 95 to Memorial Drive, the traffic is moving freely with an average speed of 36 miles per hour. From the Ground Round Rotary to Huron Avenue, there is a utility crew. Various lanes are restricted. You are advised to expect delays. On Route 2 heading west from Memorial Drive to Route 95 the traffic is moving freely with an average speed of 36 miles per hour. On Route 2 heading west from Route 95 to Route 495, the traffic is moving freely with an average speed of 46 miles per hour.

However, the system contains a series of rules designed to cut down on the verbosity of the traffic reports that are generated. These rules can apply various linguistic phenomena, such as segregatory coordination and ellipsis, as well as some task specific heuristics for eliminating redundant information. After applying these rules, the report above is reduced to the more concise report listed below:

On Route 2 heading east from Route 95 to Memorial Drive, the traffic is moving freely with an average speed of 36 miles per hour. From the Ground Round Rotary to Huron Avenue, there is a utility crew. Various lanes are restricted. You are advised to expect delays. Traffic on the other roadway segments for Route 2 is moving freely at or near the speed limit.

Various other research areas that have been investigated under Voyager include the development of a robust recognizer trained from out-of-domain data and the study of the dialogue management issues surrounding reading driving directions to a user over the phone.

5. Handheld Device System Design

5.1 Overview

This section presents an overview of the hardware and software system design of the WP-3 handheld device developed for the LCS/Marine project. As described previously, the high-level objective of this work was to produce a device with rich human interface capabilities, particularly including high-resolution visual display and audio input and output, in a compact telephone-like form factor. Other design requirements included wireless communication abilities, reasonable computational capabilities, and programming flexibility. Reasonable battery life was recognized as an additional goal, although traded off against the requirements listed above and the lack of ability to make use of certain custom technologies in a low-volume prototype design.

5.2 Hardware Design

The WebPhone handset hardware design is built around the Intel SA-1110 StrongArm processor, supported by a 2 MByte Flash memory and 32 MBytes of SDRAM. Two operational peripheral slots are provided. The primary I/O slot is a PCMCIA interface, intended for use with wireless LAN card communicating with a host computer over the wireless link. A CompactFlash interface accessible by the processor provides storage for both operating system software and application-specific data. An additional serial data interface is provided for debugging purposes only. An on-board frame buffer implements the graphics interface, through a high-speed link to the system processor. In the WebPhone, this requirement is met using an FPGA to snoop the processor bus for video-related data transfers to SDRAM memory. The video data snooped by the FPGA is then stored in two high-speed SRAM ICs, used as the frame buffer. The specifications also call for the interface to the MicroDisplay to be many-time programmable or reconfigurable. In the WebPhone architecture, the FPGA interfaces to both the SA-1110 bus and the display drive electronics and is configured during the hardware boot sequence. The WebPhone design includes the Philips model UDC1200, a 12-bit sigma delta codec.

The WebPhone system electronics physical implementation consists of a number of PCB assemblies, their associated components and the interconnections between them. An overview of the system hardware architecture is given in Figure 3. The design contains five board assemblies: the Main Board, the Riser Board, the Camera Board, the Pixie Board, and the Dongle Board. The first four boards are packaged inside the phone casing; the Dongle Board serves as a docking station that supports charging and serial communication.

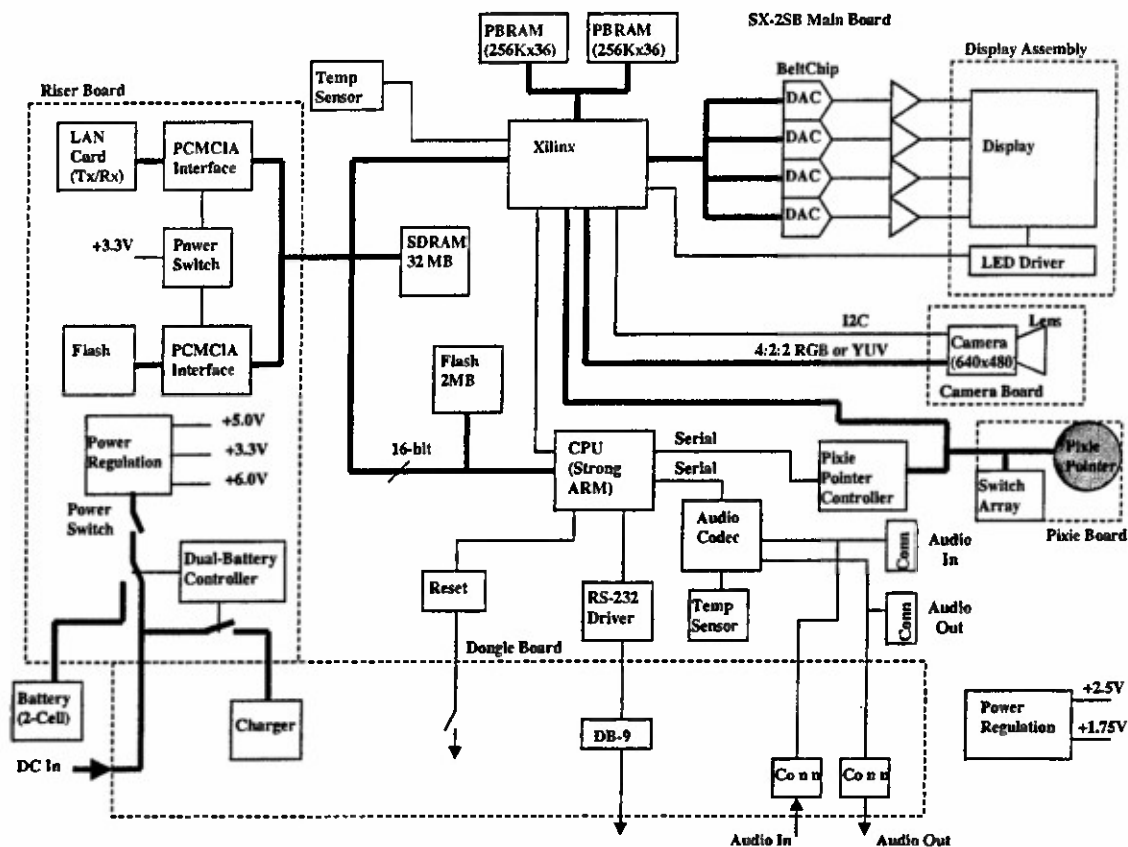


Figure 3. Organization of the SX- 2 System

The board most central to the electronics system is the SX-2SB Main Board. It is this board that houses the SA-1110 microprocessor, the SA-1110 memory and support electronics, the FPGA, the frame buffer memory ICs, the display drive electronics, the camera interface electronics, the pixie pointer and mouse buttons support electronics, the audio codec, and some of the power supply circuits.

The SX-2SB Riser Board houses the major system power circuits and the PCMCIA interface components and their associated connectors. A separate card for the PCMCIA components is required due to the bulk of the PCMCIA components. The interface between the Main and Riser Boards is through two connectors, one primarily for the PCMCIA signals and the other primarily for the supply currents and the serial interface. The following sections describe each of the WebPhone electronics subsystems in detail.

5.2.1 Power Supply and Clocking

System power is provided by a pair of Li-Ion cells connected in series, each providing a nominal voltage of 3.7 V. The use of two batteries in series, and the higher voltage they provide, reduces the complexity of the power supply circuits since all high-current regulation can be performed in a single direction, that is downwards. Alternatively, power can be supplied by the Dongle Board, which receives +12 V dc from a wall-wart. The Riser Board utilizes a PowerPath Controller from Linear Technologies to control power source selection. This device is also used to monitor

the charge state of the battery and to supply a charging current when the battery charge state is low and the handset is connected to the Dongle Board. The Riser Board uses several regulators to derive the required system voltages. The first is a 500 KHz switching regulator from Linear Technologies used to derive the 3.3 V system voltage for the Main Board digital electronics from the supply voltage selected by the PowerPath Controller. A second switching regulator from Linear Technologies derives the 5.0 V required by the Riser Board LANCard and certain of the Main Board digital and analog components, including some associated with the microdisplay, from the supply selected by the PowerPath Controller. A Maxim Power Switching Network IC is used to select the power source for each of the PCMCIA devices. The third regulator is a charge pump regulator that converts the output from the 3.3 V supply to 6.0 V, which is used by the microdisplay once it has been regulated to 5.0 V on the Main Board.

The SX-2SB Main Board requires a number of supply voltages, including the battery or Dongle voltage, 6.0 V, 5.0 V, digital 3.3 V, 2.5 V, and 1.75 V, as well as the separate 5.0 V and 3.3 V required by the components associated with the microdisplay. Of these, the battery voltage, 6.0 V, 5.0 V, and digital 3.3 V are provided by the Riser Board, while the rest are generated on the Main Board. To supply these voltages, the Main Board has several on-board regulators. The first of these is a 500 KHz switching regulator from Linear Technologies. This device converts the battery voltage supplied to this board to the 2.5 V that is required by the FPGA core. The second regulator is an adjustable LDO programmed to provide the 1.75 V required by the StrongArm core from the 2.5 V supply. The third regulator is an LDO that converts the 5.0 V supplied to the Main board from the Riser Board into the 3.3 V required by the op amp that derives the "MD_IDAC" voltage and the analog circuits of the BeltChip. The last of the Main Board regulators is an LDO that derives the 5.0 V supply required by the microdisplay and the ITO op amp from the 6.0 V that is derived on the Riser Board.

The WebPhone handset uses the Linear Technologies LTC1479 Power Path Controller to control the selection of the DC power source. The DC supply will be selected if available, if not, the battery will be selected. To guarantee that power is provided at power-up under all conditions, the LTC1479 comes up in the "3-Diode Mode". In this mode, the FET switches used to select between the battery and the DC supply are placed in a state in which they function as forward biased diodes, resulting in the supply with the highest voltage providing power to the system. This is not the normal mode of operation, and it is incumbent on the processor to pull the PMON_3DM_N line high once it is powered up. In normal operation, the LTC1479 monitors the battery and DC inputs and automatically selects the source of DC power based upon the availability of DC power.

When in the 3-Diode Mode, the LTC1479, holds the FET switch between the battery and the charger in the off state. Since the processor controls the state of the PMON_3DM_N line, the processor is required to hold this line in the high state in order for the battery to receive charging current. It is necessary, then, for the processor to be running during the charging process.

The CPU monitors the state of the battery voltage using the "PMON_LOBAT_N" line from the LTC1479. This line is high when the voltage provided by the battery is at or above 5.98 V and is low otherwise. As originally designed, the PMON_LOBAT_N signal is inverted and routed to the "BATT_FAULT" pin of the SA-1110. The "BATT_FAULT" pin of the SA-1110 places the processor in sleep mode when it is active, which occurs when the battery voltage is low. With

the processor in sleep mode, it cannot place the PMON_3DM_N line high as is required to charge the battery. This problem was resolved by routing the PMON_LOBAT_N signal along with the PMON_DCINGOOD signal into an OR gate and using the resulting signal, after being inverted, as the "BATT_FAULT" signal to the CPU.

There are two major clock sources for the SX-2SB. The first of these is the StrongArm PLL, which is referenced to a 3.6864 MHz external crystal. This reference frequency is multiplied up by this PLL to a core processor clock frequency of 206 MHz. The SDRAM memory bus frequency is programmable, but is set to $\frac{1}{4}$ of the processor core frequency on this board, or 51.5 MHz. This is the frequency of P_SDCLK1 provided to the FPGA for the clocking of its front-end modules. The second major clock source is the 24 MHz oscillator the output of which is provided to the FPGA. The FPGA uses an internal DLL to generate a 48 MHz clock for use by internal and external circuits. The external circuits that rely on this clock are the MoSys memory and the microdisplay and its associated components.

There are a number of more minor clocking sources. Besides the 3.6864 MHz crystal, the StrongArm processor relies on a 32.768 KHz crystal for use by its on-chip real-time clock (RTC). In addition, the USAR PixiPoint-Z relies on a 4 MHz crystal to derive the clocking signals required by its internal circuits. The last source of timing signals is the CMOS camera.

In an effort to maximize the system power savings, a p-FET power switch was added to the Riser Board SB. This p-FET is placed just after the Power Path Controller on this board, and so is able to remove the power source when either the battery or the Dongle are providing power to the handset.

5.2.2 Processor and Memory Subsystem

The microprocessor subsystem consists of the SA-1110 StrongArm microprocessor, a 2 Mbyte boot sector Flash memory, 4 KingMax 8M x 8 bit SDRAMs, the reset monitor IC, and several logic devices to handle supporting signals.

The SA-1110 is a highly integrated communications microcontroller containing a 32-bit StrongArm RISC processor core, system support electronics, multiple communications channels, an LCD controller, a memory and PCMCIA controller, and general purpose I/O (GPIO) ports. The processor provides power efficiency, low cost, and high performance, and is packaged in a 256-pin mini-BGA package. The processor uses a core voltage of 2.5 V and has an I/O voltage of 3.3 V.

The integrated clock generation circuits of the SA-1110 feature an on-chip PLL that provides the designer with tremendous flexibility in selecting the processor clock frequency and memory clock rates. In the WebPhone, the processor core clock is set to 206 MHz. The clock rate of the memory bus can be programmed to be either $\frac{1}{2}$ or $\frac{1}{4}$ of the processor core clock rate through register settings. In addition, the processor supports both 16-bit and a 32-bit data bus widths. Both bus widths were considered for the WebPhone and the 16-bit architecture was selected in an effort to reduce power consumption when compared to the 32-bit implementation. Regrettably, after MicroDisplay had committed to the 16-bit bus architecture, Intel published an Errata dated March 20, 2000 revealing that the SA-1110 does not reliably perform reads/writes from/to

internal registers immediately following SDRAM reads when the 16-bit processor bus architecture is implemented and when running at the full memory clock frequency. As a result, the bus speed of the SA-1110 is limited to 51.5 MHz in the WebPhone design, which is $\frac{1}{4}$ the processor clock rate. Unfortunately, this does result in a reduction in system performance, but does serve to reduce system power consumption. Using a bus speed of 51.5 MHz with a bus width of 16 bits provides a bus bandwidth of about 100 MB/sec, about twice that of the SX-1 board.

The processor has an address bus width of 26 bits and is compatible with ROM, Synchronous Mask ROM, Flash, DRAM, SDRAM, SRAM, and PCMCIA memory types. The processor features an 8-Kbyte data cache and a 16 Kbytes instruction cache. The Flash memory is the AMD Am29LV160D 16 Mbit CMOS boot sector Flash memory. The location of the boot code within the device is jumper-selectable on the SX-2 board. With the first setting, address 0x0000 0000 within the CPU address space is at the base of the Flash, while with the second setting, address 0x0000 0000 within the CPU address space is at address 0x0010 0000 within the Flash. The jumper effectively swaps the upper and lower halves of the Flash within the address space of the CPU.

The SDRAM used in the WebPhone is a set of 4 KingMax CMOS 8M x 8 Synchronous DRAM ICs. These devices are packaged in the 44-pin TinyBGA package and have an access time of 8 nsec. Each device is organized as a set of 4 banks, each of 4096 rows and 512 columns. From the viewpoint of the processor, the 4 SDRAM devices are configured into two DRAM banks of 8M x 16 bits, the first at processor bank 0 and the other at processor bank 2. The starting address of processor DRAM bank 0 is 0xC000 0000 and that of processor DRAM bank 2 is D000 0000. Therefore, the base address of each of the SDRAM banks within processor bank 0 are 0xC000 0000, 0xC020 0000, 0xC040 0000, and 0xC060 0000, and of each of the SDRAM banks within processor bank 2 are 0xD000 0000, 0xD020 0000, 0xD040 0000, and 0xD060 0000.

5.2.3 PCMCIA Interface

Two slots for external I/O devices are provided by the WebPhone. Electrically, both of these slots are standard 16-bit PCMCIA device slots. Mechanically, one slot is implemented as a standard 16-bit PCMCIA connector. This slot is designed for use by the radio interface, but may also be used by a serial line interface or Ethernet interface for testing and software download. The second slot is implemented in the CompactFlash mechanical format. The CompactFlash is electrically identical to the PCMCIA, but housed within a smaller physical package. This slot is intended for use by 32MB to 128MB CompactFlash memory cards. It provides the ability to configure the WebPhone for specific tasks. Use of the CompactFlash card to initialize the device is described in the System Software section.

The interface to each of these devices is implemented using a Link-Up LV1110VCA PCMCIA controller IC. This IC provides most functions of the PCMCIA interface, including buffering, interrupt management, and power control. A small number of additional gates and passive devices are required to complete the interface.

5.2.4 Display Interface

The WebPhone handset uses the MicroDisplay “Carradine” display mounted on a DC-31 board and housed in the “Looking Glass” eyepiece. Display data provided by the FPGA is converted from digital to analog data by a custom IC known as the BeltChip, described further in Section 7.7.1. The BeltChip consists of a front half and a back half. The front half, not used in the WebPhone architecture, contains the four ADCs and a PLL, while the back half contains the four DACs. Only the back half components are used in the WebPhone design.

The four BeltChip DACs require a number of external signals for proper operation. Besides the 8-bits of data required by the DACs, the BeltChip also requires a clock the rising edge of which is used to latch the data provided to the device. This clock, referred to as “MD_DACCLK”, is provided to the BeltChip by the FPGA and is 180 degrees out of phase to the clock used by the FPGA to provide the data. This guarantees the data will meet the set-up and hold time requirements. The BeltChip DACs also require a pair of enable signals that are inverses of one another. The signal “MD_DAC_ENP” is the positive of the two enable signals and “MD_DAC_ENN” is the inverse of that signal. “MD_DAC_ENP” is equal to the inverse of the “frame” signal, and this signal changes state with the start of each field. The last signal to the BeltChip is “MD_IDAC”, which is a constant voltage of 1.65V.

This is the function of the signals to the BeltChip DACs. Each of the four BeltChip DACs actually consists of a pair of DACs, one providing positive output current and the other providing negative output (or input) current. The positive DAC is enabled by the signal “MD_DAC_ENP” while the negative DAC is enabled by “MD_DAC_ENN” so that each DAC is used in alternating output fields. The DAC output pin, “IDAC+”, is routed to an op-amp buffer to convert the output current to a voltage. This op-amp is referenced to 1.65 V and the 1.47 Kohm resistor tied from the negative terminal of the op-amp to ground provides a bias current that results in an output voltage of 2.5 V for a DAC output current of zero. The output of the active DAC is provided through the “IDAC+” output pin, so that positive DAC outputs result in a voltage below 2.50 V and negative DAC outputs result in a voltage above 2.50 V. In this manner, every other field supplies a video output voltage above and then below the ITO voltage of 2.50 V. The output current of the inactive DAC is provided to the “IDAC-” output pin, providing a place to dump its output current.

5.3 System Software

System software used to operate the handheld device include an initialization program, an embedded operating system, a windowing and display management system, audio I/O software, and an embedded HTML display component (web browser). An overview of the client system software building blocks is shown in Figure XXX. Our strategy in the development of this software was to make use of existing standard software components to the extent possible, and then to modify these components and develop missing components to meet the specific needs of our handheld device. In this section, we discuss design choices for this software, with particular focus on non-standard modifications and original work undertaken for this project.

5.3.1 Device Initialization and Bootstrap Software

Device initialization and bootstrap software is responsible for initializing the raw hardware of the handheld device at startup, and for loading and initializing the device operating system. Initialization software also plays a critical role in device power management. The initialization software for the LCS/Marine device was developed from scratch at MIT.

5.3.1.1 Bootstrap Sequence

The core requirement imposed on the bootstrap code is that it correctly initialize the handheld hardware at system startup and on wake from sleep. We identify two additional requirements: that bootstrap operations be as fast as possible, and that the bootstrap process offers the flexibility to choose different system software as needed for different applications. Unfortunately these requirements are somewhat conflicting. We resolve these requirements by implementing the following bootstrap sequence in device firmware. The firmware itself is stored in the 2MB Flash ROM of the handheld device. The ROM space is broken into several partitions. The size and location of each partition is recorded in a fixed header stored in partition 0.

1. Cold Start. On reset of the CPU, the ROM partition table is examined, and the contents of ROM partition zero are executed directly from ROM.
2. Initialize processor and main memory systems. Code executed from ROM partition 0 resets CPU registers to a known state, and writes configuration and timing parameters controlling hardware access to the main DRAM array into StrongArm registers, making the DRAM available. A brief memory test is performed to check for gross failure of the PU subsystem. Note that this test is *not* designed to detect transient memory failures.
3. Configure PCMCIA slots. Timing and configuration parameters enabling CPU access to the PCMCIA and CompactFlash slots are set in StrongArm and LinkUp PCMCIA driver registers.
4. Examine PCMCIA slots for a valid bootable file storage device. The PCMCIA and CompactFlash slots are examined to see whether a hardware device is present. If so, several tests are performed. If a) the device located is a storage device containing valid filesystem partitions, and b) the first filesystem partition contains a file named "/kernel", an attempt to load and execute that file is made. If both slots contain a bootable device, the CompactFlash slot is favored over the PCMCIA slot.
5. If no bootable filesystem is found on a device in an I/O slot, the contents of ROM partition 1 are loaded into main memory and executed. It is assumed in this circumstance that ROM partition 1 contains a compressed operating system image. If no compressed OS image is found in ROM partition 1, or if the image fails to execute correctly, the device halts. No error indication is presently given.
6. The booted operating system takes control of the CPU from the boot code. The operating system may assume that the DRAM and PCMCIA/IO slots have been configured correctly, and that all other devices (display, audio, camera, pointing device, etc.) are reset. The operating system must take whatever action is needed to initialize these

elements of the hardware , and then locate and execute the programs that activate the devices functionality (ie, display manager, web browser, etc). In so doing, the operating system will typically re-examine the I/O slots for evidence of a device containing an initial filesystem. If no such device is found, the OS can fall back on a small compressed initial filesystem located in ROM partition 2. Although tightly size constrained, there is sufficient space in this filesystem to store further initialization code or simple diagnostics.

The high-level objective of this boot sequence is to allow special purpose or application-specific operating system software to be booted from a “personality card” that may be present in the handheld’s CompactFlash slot. Thus, different personality cards can configure the handheld to execute different core applications, use different communication links, and so on. This capability is essential to allow reuse of the handheld in widely varying environments.

Should no personality card be found, the handheld is still able to boot to a fully operational state using the internal ROM copy of the operating system. The standard ROM copy of the operating system initializes to a simple debugging monitor and display driver. The primary use of this version of the OS is to test and exercise new versions of display driver FPGA code provided by MicroDisplay.

Unfortunately, the step of examining the handheld I/O slots for a bootable filesystem is somewhat time-consuming in the circumstance that no filesystem is found. For this reason, the firmware implements a configurable bypass of steps 3 and 4 above, allowing direct boot into the ROM-based operating system when desired.

Finally, all partitions of the system Flash ROM may be rewritten under software control. Writing partition 1 (ROM operating system) or partition 2 (initialization filesystem storage) is straightforward. Writing partition 0 is dangerous because a failure will leave the device unusable, requiring hardware-level repair to install a new Flash ROM. For this reason partition 0 is write-locked in hardware, although the lock may be overridden under software control. To perform this function we developed a carefully written, single-function program to download and install partition 0 boot images in the device.

5.3.2 Operating System

The handheld operating system provides basic services to application programs, including memory management, filesystem management, networking and communications, peripheral device management, and so on. In contrast with many cellphone-like devices, which utilize a stripped down task-specific operating system, the richer functionality and broader purpose of our device demands that we make use of an efficient and customized, but flexible, operating system environment.

After review of the available alternatives, we adopted and customized a version of the Linux operating system for our handheld device. The primary arguments in favor of this decision were the ability of Linux to support the applications needed, commonality of environment with other software development efforts at MIT, and the highly functional networking environment supported. Primary drawbacks to the choice were the relative immaturity of Linux on the StrongArm platform, and the absence of some critical functions required by our application.

Work performed at MIT during the contract period to remedy these drawbacks is described below.

5.3.2.1 Status At Start Of Contract

Linux for the StrongArm processor is developed using an open community process. At the start of our work, a version of the linux kernel 2.2 had been developed for the DEC/Intel "Multimedia StrongArm Evaluation Board", and ported to a small number of other StrongArm embedded devices. A full distribution² of StrongArm linux was also available for a small general purpose computer, the Corel Systems "Netwinder". As custom hardware from the development effort described in Section 5.2 was not yet available, we acquired three Netwinders for use as development systems and two Evaluation Boards for use as development targets for embedded systems development.

Initial evaluation of StrongArm linux and the available development tools demonstrated that most basic functionality was present, but that some reliability and stability problems were also present. Also, some functions critical to our needs were not implemented. During the course of the contract period, however, significant stability improvements were made by other developers. We were able to benefit from these improvements by tracking the overall StrongArm Linux development effort closely. We also developed and extended the OS locally to meet our requirements. All portions of our work of value to those outside the project were made available to the larger community. A brief summary of our development work is given in the next sections.

5.3.2.2 Power Management

The StrongArm CPU hardware, although already a low-power device, supports several "sleep" and "system idle" capabilities to further reduce power drain in quiet periods. However, the Linux OS did not offer any support for the use of these capabilities. Initial design computations showed that the battery life of our handheld device operating at full power, without use of these modes, would be approximately one hour.

To increase performance in this area we added functionality to the Linux kernel and to our bootstrap code to make use of the power-saving capabilities of the StrongArm. Primary capabilities added include:

- Ability to power down peripheral devices not in use under software control. The devices supported include the display, audio subsystem, and I/O slots (PCMCIA and CompactFlash).
- Ability to adjust the CPU clock rate between a "full" rate for high performance and a "reduced" rate for power savings. The precise rates to be used are configurable in software; our standard settings were 206Mhz and 51.5 Mhz.
- Ability to sleep the main processor core and all peripheral devices. The StrongArm CPU supports a power-down "sleep" function in which the majority of the CPU core is shut

² A Linux "distribution" is a complete, integrated packaging of the Linux operating system kernel and related user programs, assembled and validated for a specific purpose by a distributor or software packager.

down, and the processor then idles waiting for an interrupt. Power draw in this mode is extremely low. However, use of the mode requires substantial operating system support. All application and OS state must be saved in permanent memory, peripherals must be powered down without loss of data, and main memory must either be powered off or placed into its own “sleep” mode. On restart, all hardware including main memory must be reconfigured and reinitialized, OS and application state restored, and user programs continued.

Overall, we were able to improve power drain under normal usage patterns by a factor of roughly three. A primary technical limitation was the amount of time required to bring the hardware out of full “sleep” mode. Because many steps are required, including the sequential reinitialization of several hardware subsystems, moving from full sleep to fully ready required approximately five seconds. This was unacceptable from a human factors perspective: our requirement is that the device should be ready to use by the time it reaches the user’s ear.

We continued to develop the power management and sleep-wake code in our system through the course of the contract. We implemented several intermediate sleep modes, in which some but not all of the hardware devices were power-controlled. A critical time saving was obtained by not powering down the display interface FPGA, thus removing the need to reprogram it on power up. An additional time saving was obtained by preserving most state in SDRAM operating in “sleep” mode, rather than writing application state to permanent memory. With these modifications, we were able to obtain a system wakeup time of approximately two seconds. The tradeoff was a significant increase in power drain during sleep, reducing the battery lifetime in sleep mode from approximately 30 hours to approximately 6 hours.

Power management, including sleep modes, dynamic CPU clock rate adjustment, and peripheral power control, has now been made available in the public distribution of StrongArm linux.

5.3.2.3 Display Controller

Linux incorporates a standard interface known as “fbdev” for memory-mapped “frame buffer” devices used as graphics displays. The programming model for this interface is intended to support industry-standard hardware using a multi-bit pixel model. The raw interface presented by the MicroDisplay field-sequential LCOS display is quite different. However, it is important for program portability that user programs see a standard interface.

We mapped the field-sequential LCOS display into a standard fbdev interface by combining the shadow framebuffer and FPGA hardware described in Section 5.2.4 with a newly implemented custom device driver. The core function of the device driver is to present the MicroDisplay video memory, which is actually implemented in the shadow frame buffer and not directly addressable, as an addressable region of Linux kernel address space. Two additional functions are supported:

- Standard fbdev I/O control (ioctl()) functions are used to implement a pseudo-color map, and to implement display synchronization with user programs to reduce flicker and tearing.

- Newly defined private `ioctl()` functions are added to allow user program control of LCOS-specific hardware functions, including ITO voltage adjustment.

Additionally, the display controller interface incorporates a method for user level programs to a) trigger reprogramming of the FPGA from the initial configuration code embedded in the firmware, or b) reprogram the FPGA directly, from configuration code supplied via a file download. The ability to reprogram the FPGA after system startup was provided to assist in debugging, and to allow for application-specific FPGA functions, such as image enhancement and processing, to be downloaded by applications. The ability to reprogram the FPGA from the initial code was added in order to provide a definitive mechanism to return the FPGA to a known initial state. This was found necessary to address problems in which erroneous control inputs would place FPGA internal state machines into an undefined and unrecoverable state.

Because this code is specific to the MicroDisplay LCOS display and FPGA code, it has not been incorporated into the main-line publicly available Linux distributions.

5.3.2.4 Audio Interface

To support user program access to our device's audio hardware, an audio device driver was implemented for the UXB1200 codec. At the time this work was undertaken, the Linux community supported two unrelated audio subsystems. The first, known as the "OSS-compatible system" was an internal extension and re-implementation of an older audio subsystem known as OSS; the Open Sound System. This sound system was considered standard, but had a number of known bugs and performance limitations. The second, known as ALSA (Advanced Linux Sound Architecture) was widely considered to be superior in design and implementation, but was not yet complete or in widespread use.

After comparison and code review we chose to port a version of ALSA to our StrongArm linux kernel, and implement within it audio support for our hardware. We determined that the design of OSS was insufficient for our requirement to support independent full-duplex input and output channels with the UXB codec, and that lower-level implementation decisions made by OSS developers made support for this codec difficult. We were able to develop reliable audio support in our kernel by adopting ALSA, but omitting parts that were incomplete or were not needed for our application. In addition to the UXB1200 device support, we added or completed several minor functional extensions to ALSA. The overall result of this effort was reliable, full-duplex audio support with low computational overhead and full control of the codec hardware to select available sample rates, adjust volume, etc.

ALSA has now been adopted as the standard sound architecture for most linux distributions, and is available in the public version of StrongArm linux. Support for the UXB series of codecs (1200, 1300, 1400) is incorporated into this sound architecture.

5.3.3 Windowing and Display Management Subsystem

The windowing and display management subsystem provides the underlying primitives needed to draw objects on the MicroDisplay, create, manage, and destroy windows, and perform related operations. The design of this subsystem is critical to the success of a visually oriented device

with a small, aggressive form factor. For the user, the system must produce, simple, intuitive, legible displays and support a minimalist, straightforward model for window management. The subsystem must also operate effectively in a low-resource environment.

The overall subsystem contains three component parts: the windowing toolkit, the graphics toolkit, and the font and type management subsystem. Our approach to implementing these components is described in the following sections.

5.3.3.1 Windowing Toolkit

The windowing toolkit provides the basic functions for drawing primitive objects, such as rectangles and polygons, on the display screen, and for managing and displaying *windows* of data, belonging to different applications, on the device. In contrast to the situation with a traditional PC, where rich functionality and artistic visual effects are important considerations, our requirements focus on clarity and legibility, and on low drawing overhead and use of CPU resources.

In our design, these capabilities are provided by an adapted version of the Microwindows toolkit. Microwindows is an open source project aimed at producing desktop-quality graphics functionality for small devices. Functions provided include graphics region support, clipping, drawing of lines, rectangles, circles, ellipses polygons and text, a color management model and pseudo-color palettes, image rendering (GIF/JPEG/etc), and blitting (copying and moving of pixel data). Above this, it provides a message-passing protocol interface, window creation and destruction, window exposure, hiding, and moving, window painting, and utility functions to maintain device and graphics contexts. Microwindows runs in 100-600k of memory, and including all associated libraries, is an order of magnitude smaller in code size than the X Window System traditionally used on Linux desktop computers.

Adaptation of Microwindows for our needs was straightforward. The Microwindows software is modular, in that choices about capabilities to support can be made at compile time. We specified a custom configuration containing support for core functionality and our display and pointing devices.

Microwindows is designed to interface with hardware displays through the Linux Frame Buffer interface discussed in Section 5.3.2.3. Immediately above this is a device-independent code layer that performs lowest-level basic drawing functions. In the distributed version of Microwindows, this layer has been tuned for use with standard graphics hardware on Intel x86 processors. To obtain satisfactory performance, we re-implemented this layer in a manner tuned to the StrongArm instruction pipeline, the timing requirements of our display hardware, and the (somewhat poor) performance of the StrongArm version of the gnu C compiler. These modifications gave us significant performance improvement across a standard mix of applications.

5.3.3.2 Graphics Toolkit

The graphics toolkit implements core elements of the visual user interface such as buttons and text boxes. The graphics toolkit also provides the basic mechanism used to lay out graphics objects on the display.

Graphics toolkit designers often face a tradeoff between the usability of the generated graphics and aesthetic considerations such as rich icons and three-dimensional appearance of buttons, scroll bars, and the like. A related tradeoff is that between quality of the displayed graphics objects and computational overhead. Our application differs from the normal general-purpose in that usability and low overhead are or critical, while rich elaborate graphics are not desirable. For this reason industry-standard graphics toolkits such as Qt or GTK were not suitable.

We instead adopted an experimental open-source toolkit known as FLTK (Fast Light Toolkit). FLTK is designed for embedded system interfaces, and presents an extremely simple, crisp visual model to the user. Within this paradigm, FLTK provides the following capabilities:

- Buttons
- Text Boxes
- Valuator
- Groups
- Box Types
- Labels and Label Types
- Widget Position, Sizing, and Layout
- Color Management

FLTK is designed to impose minimal code and data memory overhead. This is achieved primarily by splitting the implementation into many small objects and designing it in a heavily modular fashion so that unused functions are never referenced from the top-level dispatch tables, and thus do not get linked into the executable code. The result of this design strategy is that FLTK must be carefully configured for the specific application, but once configured is extremely small. Runtime memory usage is also minimized by careful design. On the StrongArm, code and data overhead obtained can be illustrated through the following statistics:

- `sizeof(Fl_Widget)` == 64 to 92 Bytes. This illustrates the contribution of each graphics object to runtime memory requirements.
- The "core" (the "hello" program compiled & linked with a static FLTK library and then stripped) requires 114K of code space. This illustrates the smallest practical program size.
- A sample program that includes every widget requires 538k of code space. This illustrates the worst-case contribution of the toolkit to program size.

5.3.3.3 Fonts and Type Management

Legibility of text is a critical usability criterion for a small handheld device. Achieving this is difficult due to:

- Small display size and relatively low resolution.
- Difficulty of positioning display correctly with respect to viewer's eye.
- Usage pattern requires achieving correct eye positioning rapidly.
- Limitations on quality, contrast, and brightness of display.

For these reasons, our windowing and display system focused heavily on achieving high legibility of displayed text. We achieved this through two key means:

1. Adoption of high-legibility fonts. Rather than using industry-standard fonts such as Helvetica and Times-Roman, we adopted throughout our interface fonts originally designed for enhancing readability of web pages on low-resolution displays. These fonts include the "Georgia" serif font and the "Verdana" and "Tahoma" sans-serif fonts, all developed and distributed by Microsoft for use with their web browsers.
2. Use of tuned anti-aliased font rendering throughout the system. Our type rendering engine based on the open-source FreeType 2 system. FreeType 2 is a software font engine designed to be small, efficient, highly customizable and portable, while capable of producing high-quality output (glyph images).

The following capabilities are provided by FreeType 2:

- Provides a simple and easy-to-use API to access font content.
- Designed for embedded systems. Does not use static writable data - can be run from ROM directly.
- Supports the following font formats:
 - TrueType fonts (and collections)
 - Type 1 fonts
 - CID-keyed Type 1 fonts
 - CFF fonts
 - OpenType fonts (TrueType and CFF variants)
 - SFNT-based bitmap fonts
 - X11 PCF fonts
 - Windows FNT fonts
 - BDF fonts (including anti-aliased)
 - PFR fonts
 - Type42 fonts (limited support)
- Capable of producing a high-quality monochrome bitmap, or anti-aliased pixmap, using 256 levels of grey. Substantial readability improvements over the 5 levels used by Windows 9x/98/NT/2000.
- Full, efficient TrueType bytecode interpreter.³ Able to produce excellent output at small point sizes.

Freetype provides a small number of tunable parameters for the rendering process. We attempted to adjust these parameters to obtain highest quality on the MicroDisplay LCOS display. However, a series of experiments revealed that the quality improvement was only minimally detectable to an average observer, and was highly dependent on the precise operating conditions of the display hardware. Small drifts in ITO voltage, resulting in slight changes in display contrast or flicker levels, reduced the effectiveness of this tuning. We concluded that use of the standard values was appropriate.

³ Note that commercial use of this capability requires the user to obtain licenses or permission to practice US patents 5155805, 5159668, and 5325479.

5.3.4 HTML Information Display

The visual information component of Galaxy, and thus LCS/Marine, applications is implemented by generating and displaying standard HTML WWW pages. An embedded web browser in the handheld device is responsible for rendering and displaying these web pages. Because few limits are placed on the generation of these web pages, the HTML display component must offer relatively rich functionality.

A number of lightweight embedded systems browsers were available at the time of system development. However, these browsers were typically limited in function. Examples of such limits include support of only a subset of the current HTML standard and lack of support for JavaScript. The resulting tradeoff is that these browser implementations are typically small, easy to retarget to new environments, and offer high performance.

One possible design strategy would thus be to adapt the desired applications to these limited browsers. This would typically involve rewriting generated web pages for correct display in the new environment, removing dependencies on unsupported functions, and evaluating the resulting loss of user capabilities.

We viewed this approach as undesirable. Because the overall goal of the project was to support applications that were also used in other environments, we concluded that the web browser and HTML display capabilities of the WebPhone should be as close as possible to those of browsers intended for more traditional environments – although the precise *form* of those capabilities should be optimized for the WebPhone environment. An example of this is our discussion of lightweight graphics object and text rendering in the previous section.

To meet this requirement, we adopted and adapted ViewML, a lightweight but capable web display engine. ViewML is built around three key open-source components; the KFM HTML display component developed by the KDE project, and javascript component from the same source, and the libwww core HTTP component supplied by W3C. ViewML provides full HTML 3.2 rendering capabilities in minimal footprint package. At the time of development, HTML 3.2 was sufficient to support all of the applications discussed in Section 4 without modification.

Adapted for our system, ViewML requires approximately 2MB of code memory for the core component, plus approximately 1MB additional memory for extensions to support our higher-quality text rendering interface. This contrasts with an estimate of 10-20 MB of code memory to support a richer full function web browser such as Mozilla.

Through the course of the project, Web technologies and standards continued to evolve. Web browser capabilities sufficient to meet our requirements in CY 1999-2000 are no longer sufficient. At the time of writing this report, the limitations of ViewML have become significant. Particularly, newer versions of the HTML standard, the spread of Cascading Style Sheets, and the continuing development of Javascript, render ViewML less and less capable of meeting the requirement to support a broad cross-section of web page designs. Replacement or upgrading of this software component would be essential to support a current generation WebPhone. Two alternatives are possible. First, a number of commercial embedded browsers have been developed and made available since the conception of this project. A possible alternative that preserved the benefits of our approach would be to upgrade ViewML to use the more modern

KHTML and KJS components now available from KDE. These components provide a lightweight but richly standards-compliant web information display framework; including support for HTML 4.0 and XHTML, CSS, ECMAScript (the standards-based successor to JavaScript), XML, and the W3C Document Object Model (DOM). The merit of this approach has recently been demonstrated by Apple Computer's selection of KHTML and KJS to form the core of their commercial Safari Web Browser. Because a broad base of commercial and open-source community developers have demonstrated strong commitment to the development of these components, we expect that they will continue to evolve as needed to quickly track further web standards development. This evolution is critical to the viability of a device that seeks to make use of general Web technologies.

6. Communications and Networking

This section describes work undertaken within the LCS/Marine project to support the communications and networking requirements of the WebPhone device. This work included both engineering effort needed to adapt and utilize existing standards and protocols when possible, and a research effort needed to realize new capabilities not available at the time of project start. The first subsection briefly outlines our overall networking framework. Discussion of newly developed networking functions and capabilities is presented in the remaining subsections.

6.1 Overview

The LCS/Marine handheld device uses wireless networking capabilities to communicate with

- Galaxy conversational system servers, to support the core functions of its conversational applications.
- Sources of multimedia data (maps, charts, etc.) to present data to the user as required by the application.

This network channel must support both voice and traditional data communication, and it must offer the highest possible performance to successfully support graphics, video and multimedia data displays.

The scope of the LCS/Marine project did not include the development of new lower layer networking technologies. Instead, our requirement was to make use of available wireless networking capabilities to the extent possible. At the start of the contract period, many wireless networking service providers were predicting deployment of medium speed wide area wireless services (so-called 2.5G and 3G technologies) within the contract lifetime. However, no such service was actually operational. The project's initial strategy was to partner with wireless network operators (e.g. Bell Atlantic Mobile, now Verizon) and wireless network technology developers (e.g. Qualcomm) to take advantage of these deployments as they occurred.

Early in the project period, after discussions with several wireless service operators and the technical staff of two equipment developers (Qualcomm and Nokia) MIT concluded that the predicted deployments of wide-area medium-speed access services were unlikely to materialize within the contract lifetime. We therefore adopted a revised strategy, with work focused on the existing Cellular Digital Packet Data (CDPD) wide area service, and 802.11 local area service. Unfortunately, the performance of CDPD service is extremely limited.

Table 2 summarizes the results of a series of experiments to measure available performance between a mobile CDPD node located at MIT and a fixed server node also located at MIT.

Table 2. Measured CDPD Performance, MIT, Summer 1999

Average Throughput	7.3 kilobits/sec
Worst Case Throughput (with functional connection ⁴)	1.2 kilobits/sec
Average Latency	350 milliseconds
Worst Case Latency	672 milliseconds

This level of performance is extremely limiting. As a result, connection of the LCS/Marine device directly to a Galaxy server over a single CDPD network link was not useful. We discuss an alternative strategy for use of CDPD channels in Section 6.3.

LCS/Marine supports two overall networking models. In the *individual model*, a single handheld device communicates directly with a conversational system server over a wide area wireless network. In the *group model*, one or more LCS/Marine devices communicate with a local access server using a short-range high-speed wireless network. The access server, in turn, communicates with the conversational system server over a longer-range, and presumably slower speed, wireless network. The access server may also provide functions such as web data caching and proxying, to improve overall performance.

Our system design allows for the same LCS/Marine device to support either operating mode, and to switch between operating modes and different wireless communication environments “on the fly”. Ultimately, our design envisions a handheld device containing either multiple radios, each supporting a specific communications standard (e.g. GPRS, CDPD, 802.11) or a single *software defined radio*, capable of being reprogrammed to operate in different communications environments as required. However, limitations of the radio technology available during the contract period precluded these options. Instead, our design implements a limited version of this flexibility by providing a standard PCMCIA slot for the radio device, and allowing the user to swap radio devices as needed to support different communication environments. The sequence is as follows:

1. User places LCS/Marine device in “sleep” mode by pressing button.
2. User removes currently installed radio card and inserts card needed for use in new environment.
3. User “wakes” device with button press. Device configures new radio card and re-establishes active network connections using new radio channel.

It is important to note that this operation does not interrupt or terminate active conversational application sessions. The user of an application may simply pick up where they left off before swapping the radio card.

⁴ In our experiments, it was not uncommon for CDPD connections to fail after some period of time. The measurements in this table do not include variations introduced by failed connections; however, in practice these failures have a strong further effect on performance.

LCS/Marine uses standard Internet (TCP/IP) protocols for all voice and data communication. All basic functions of TCP/IP are supported natively by the Linux operating system. We extended these functions as required to support the requirements of LCS/Marine for mobility and operation in performance-challenged environments.

We utilized Mobile IP protocols to support the radio swapping function described above. Mobile IP allows an IP-based device to move from one IP network to another without disrupting existing TCP connections, and thus without disrupting existing Galaxy sessions. Although most wireless networking technologies, including CDPD and 802.11, allow for roaming among the different base stations of a single wireless network, Mobile IP provides the critical additional function of mobility *between* different wireless networks, different technologies, and different network providers.

During the contract period, Mobile IP protocols saw continuing development in the IETF. Additionally, Mobile IP support in the Linux operating system was incomplete and somewhat unreliable at the start of our work. Implementation of a complete, tested, and up to date Mobile IP protocol suite was beyond both the scope of the LCS/Marine project and our available resources. We concluded that such an implementation would eventually be developed by others, but not in time for our use. We instead implemented a minimal subset of the protocols, sufficient for our requirements in the context of a proof-of-concept design. Of particular note, we did not implement those portions of the protocol suite intended to provide security in commercial environments, nor those intended to support extremely rapid mobility between different networks. Our implementation was sufficient to allow seamless movement between networks using the three-step radio replacement strategy described above.

At the time of writing this report, Linux Mobile IP is developing rapidly, and has essentially been brought into compliance with the current standards.

6.2 Voice Communication Strategy

A key function of the LCS/Marine network is to transmit digitized audio between the handheld device and the Galaxy application server. This audio channel is used to carry voice messages from the user to the Galaxy applications, and to carry audio responses from applications to the user. In both directions, quality of the transmitted audio is a concern.

The standard method for carrying audio in IP packets is to represent the audio as digital data, and then to transmit the data in a "real-time" stream using the UDP and RTP protocols. The characteristics of real-time data streams are that the data is delivered as quickly and accurately as possible, but that no end-to-end reliability *guarantees* are implemented by the network protocol. This transmission method is considered most appropriate for continuous media such as audio and video. The reason for this is that support for reliable data delivery introduces significant *delay variation* into the channel, making accurate rendition of the isochronous real-time data impossible.

However, in channels with high packet loss, such as some wireless data links, the damage to the real-time audio signal from lost packets can be severe. In channels with high loss but also high bandwidth, this problem can be overcome with Forward Error Correction. However, if both high

loss and low bandwidth must be accommodated, the problem is fundamental. Particularly in the CDPD case, this describes our situation.

We therefore implemented two separate strategies for transmission of the audio portion of LCS/Marine application data. The first followed the standard approach. Digitized audio data from the codec was encapsulated in standard RTP packets and transmitted using UDP datagrams. This strategy was employed whenever a high-quality network path was available.

Our second strategy took advantage of the request-response nature of Galaxy application interactions. Rather than transmitting the audio data using a real-time channel, we collected audio data at the sending side until a pause in the data was detected, and then transmitted the data to the receiving side using a fully reliable non-real-time channel implemented using TCP. In essence, voice data was delivered reliably to the remote side on a phrase by phrase basis. At the receiving side, the phrases were reassembled into continuous speech and passed on to the next system module.

We evaluated two methods for identifying the phrase boundaries and triggering phrase transmission. The first of these was an energy detector adopted from that used for a similar purpose in the VAT audioconferencing application. This voice energy detection and sampling system operates by measuring the energy in an audio input signal, weighted towards the energy distribution of human speech. It implements "guard buffering" on either side of a phrase as detected by the energy detector, to avoid clipping the start or end of a spoken utterance.

We determined that the energy detector worked well in office-like environments. However, it proved sensitive enough to noise that it was not successful in noisy environments. For this reason, we also implemented an explicit push-to-talk mode, with the energy detector replaced by a push-to-talk button. Retaining the guard buffers again eliminated voice phrase dropouts resulting from late push or early release of the button. While explicit push-to-talk has the obvious disadvantage of requiring additional control input from the user, it greatly increases the reliability of the audio transmission, and hence the understanding, process in sufficiently noisy environments.

6.3 Adaptive Channel Multiplexing

This section describes our work on an *adaptive channel multiplexing* strategy for combining several wireless network channels with highly variable characteristics to form a single, higher bandwidth channel. We developed this strategy in response to the severe performance limits of CDPD wireless networking, and the inability of this networking to support the LCS/Marine device. However, the approach has broad applicability

The limited bandwidth of current wide-area wireless access networks (WWANs) is often insufficient for demanding applications, such as streaming audio or video, data mining applications, or high-resolution imaging. Inverse multiplexing is a standard application-transparent method used to provide higher end-to-end bandwidth by splitting traffic across multiple physical links, creating a single logical channel. While commonly used in ISDN and analog dialup installations, current implementations are designed for private links with stable channel characteristics.

Unfortunately, most wide area wireless network technologies use shared channels with highly variable link characteristics, including bandwidth, latency, and loss rates. To address this problem we have developed a novel *adaptive* inverse multiplexing scheme for WWAN environments, termed *Link Quality Balancing*, which uses relative performance metrics to adjust traffic scheduling across bundled links. By exchanging loss rate information, we compute relative short-term available bandwidths for each link. A brief overview of the approach, and some information about our implementation for LCS/Marine, are given below. Full details and performance evaluation are found in [18], included with this report.

6.3.1 Overview of Link Quality Balancing

A large number of wide-area wireless access network (WWAN) technologies have recently emerged, including Metricom's Ricochet Packet Radio network, *Global System Mobile* (GSM) [19], IS-95 [20], and Cellular Digital Packet Data (CDPD) [21]. The defining characteristic of WWANs is their use of a shared channel with long and variable round-trip times (RTTs), typically on the order of 500ms, coupled with a relatively low and variable bandwidth (usually in the tens of Kb/s). Additionally, many of these wireless technologies support link-level ARQ schemes for reliable transmission. While such mechanisms provide better line error characteristics, they combine with channel access contention to introduce significant delay variability.

The low bandwidths provided by WWAN technologies are often insufficient to support demanding applications, such as voice recognition and high-resolution imaging. In these contexts, one obvious solution is to spread connections over multiple physical links. By *striping* data over many physical channels (called a *bundle*), possibly by breaking packets up into fragments, it is possible to present a single logical channel with increased available bandwidth. When using such techniques, however, variations in the channel characteristics of any particular link can have catastrophic impact on the performance of the bundle as a whole. In particular, variations in the perceived transmission delay of an individual member link may cause delays in demultiplexing and packet assembly. In the absence of sophisticated scheduling techniques to balance delay, bandwidth variability between member links can cause the performance of many transport protocols to be throttled by the slowest member link, even when traffic is allocated in proportion to link speeds. Similarly, losses on one link may prevent packet reassembly, increasing loss rates markedly. These *amplification effects* motivate our design of an adaptive inverse multiplexing algorithm in WWAN networks.

Inverse multiplexing is fairly common in current network technologies. Most implementations, however, assume physical transport mechanisms with constant bit rates and stable link characteristics such as those found in circuit switched networks [22, 23, 24]. This is not the case with CDPD. Commercial CDPD networks use an IP-based transport mechanism such as TCP or UDP to transport data. These mechanisms are subject not only to the expected loss and delay characteristics of the Internet, but additional WWAN-specific issues as well, such as increased round-trip times and delay variability. The added synchronization requirements of an inverse multiplexing scheme only compound any losses or variability in bandwidth or delay.

Reliable transport protocols require reasonable limits on packet reordering, jitter, and loss in order to function optimally. In particular, TCP has been shown to be extremely sensitive to

variations in delay, as they affect both its ACK-based clock and RTT estimates for packet retransmission [25, 26, 27]. The Fast Retransmit algorithm [RFC2001] also makes assumptions about the level of packet reordering, and induces spurious retransmissions in the face of excessive reordering. It is therefore important to ensure that any multi-link scheme limit packet reordering and delay jitter as much as possible.

The multiplexor must then schedule packet fragments so that they are received and reassembled in roughly the same order they arrived, with a minimum of added delay. This requires balancing the fragment distribution according to the current *bandwidth*delay* product of each individual link. Due to the depth of network queues relative to the *bandwidth*delay* product, and channel access asymmetries inherent in CDPD, it is extremely difficult to obtain accurate measurements. Instead, we use relative performance metrics to adjust the traffic distribution between links, and allow the end-to-end transport protocol's bandwidth probing algorithm to function as it would normally. We term this dynamic adaptation *Link Quality Balancing*.

6.3.2 LCS/Marine Implementation

We implemented Link Quality Balancing, as described in [18], for use with the LCS/Marine "group communication model" described in Section 6.1. The specific usage is as follows:

- A number of LCS/Marine handheld devices communicate with a *Local Access Station*, typically vehicle mounted, using high-speed wireless LAN technology.
- The Local Access Station communicates with the Galaxy Application Server using multiple CDPD channels and the algorithm described above.
- The Galaxy Application Server has access to high-speed fixed (wired) networks, and communicates with its back-end databases and data sources using these networks.

In this model, the communication bottleneck is between the Local Access Station and the Galaxy server, and we make use of multiple WWAN channels to improve performance.

In our implementation, the Local Access Station was deployed on a Hummer vehicle. WWAN communication was supported by a bank of four Sierra Wireless MP200 CDPD modems. The modems were connected via a Specialix SIO serial board to a PII/400 PC running FreeBSD 2.2.7., modified to support our LQB algorithm. The other end of the virtual link was a second PII/400 PC running FreeBSD 3.2. This machine acted as a terminator for the multiplexed virtual link, and as a router for the resulting packets into the MIT network and SLS Galaxy servers. One interface of this machine was connected directly to the Bell Atlantic CDPD network over a 56K PVC Frame-Relay circuit. Another interface was connected to the MIT LCS wired network. LQB was implemented as a user-level PPP daemon that runs PPP [RFC1661] with HDLC framing [RFC1662] over UDP. The CDPD modems operate on Bell Atlantic Mobile's digital cellular network. During initialization, the mobile multiplexor scans for available CDPD channels, and allocates the four best channels to the MP200s. Each modem locks on its prescribed channel, and begins PPP link establishment negotiation with the wired multiplexor/demultiplexor to provide the virtual data path. In average conditions, bandwidths of 28-33kbits/s were obtained from the four striped CDPD channels.

6.4 Web Performance and the Cache Information Protocol

The critical performance limitation facing the LCS/Marine device is an inability to make effective use of the MicroDisplay for displaying rich or varying content, due to the low bandwidth and high latency experienced in wide area networks. We explored several strategies to address this problem by using device-local resources, such as power or disk space, in an intelligent manner to reduce network traffic where possible. A key result of our research is a new approach to this problem, based on applying a component model to web content – images, text, maps, and the like. After subdividing, describing and transmitting web content, generally assumed to be atomic objects, as its component parts, the client is able to reuse previously transmitted elements stored in its local cache in servicing other requests. To facilitate this sub-atomic exchange of information, we developed a new protocol, the Cache Information Protocol (CIP), which uses an application-independent means of describing content. This protocol is designed to integrate easily with a primary end-to-end protocol, such as HTTP, while providing the flexibility and compactness required. We evaluated the protocol through analysis and through two experiments based on LCS/Marine Galaxy applications, and showed a significant performance improvement in those applications.

A brief summary of this work is presented below. Full details may be found in [28] and [29].

In resource constrained environments, tradeoffs are central to the goal of optimizing performance. This phenomenon is particularly important in the case of wireless devices, forced to deal with the scarce resources as bandwidth and power. The problem becomes even more critical in the context of the LCS/Marine handheld device, where low network bandwidth, high network latency and low power use must be traded off against the potential of the high resolution MicroDisplay to provide rich visual data presentation.

While various levels of information can be used in making these decisions, an application level understanding of the events occurring can enable a greater level of performance through greater optimizations and increased flexibility. Our approach is to examine the structure of content at an application-level of a wireless web-browsing client. In particular, we examine applications such as those supported in the Galaxy system, in which the server dynamically produces multimedia content, such as images or audio files, in response to user queries. For this class of applications, it is often the case that much of the content generated draws on the same component resources, such as map icons, table elements, and the like. However, this structure is obfuscated from the client because the client receives only the final displayed object, which is assumed to be atomic.

We improve performance by exposing this internal structure and exploiting the associated performance benefits. In our architecture, the server has the ability to explicitly inform the client of the information required to synthesize a given object. It also has the capacity to provide the client with the various raw sub-components that are used in the synthesis, along with meta-data to enable their use. This knowledge enables a seamless shift of content generation from the server to the client (here, the LCS/Marine handheld) within the content of an end-to-end protocol such as HTTP. More importantly, it facilitates the reuse of these component pieces by the client, thereby reducing network usage and latency. Because content generation at the client is not subject to network bottlenecks, performance is dramatically higher. Exposing the component

parts of web content, thus violating the assumption that they are atomic objects, is the key idea of the process we call sub-atomic caching.

The result of this work is a flexible and application-independent architecture for enabling the client (here, the LCS/Marine device) and server (here, the Galaxy application server) to perform and reason about these tradeoffs. Key to this result is our design and implementation of the Cache Information Protocol, which allows a seamless migration of responsibility between client and server. This facilitates the flexibility required to optimize performance for different applications and using different network technologies. We also designed and implemented an application-independent Information Representation Format, which is used by the CIP protocol to provide the client with the meta-data required to use sub-atomic components. Our experiments showed that this format is sufficiently flexible for the task and enables a compact representation of this information.

7. Handheld Device Development

7.1 Goal

The goal of this portion of the project was to produce a compact multi-media capable computer in a telephone handset form-factor. Key components of the device were to include a low-power, high resolution, full-color miniaturized display, audio input and output capability, a pointing device, wireless communication capability and a powerful internal microprocessor. The unique liquid-crystal-on-silicon (LCOS) technology was to provide a display with high resolution and low power to enable viewing standard World-Wide Web screens.

Wireless handheld connectivity to the Internet or intranets is actively being pursued by many research efforts. Most of them concentrate on enhancing conventional digital cellular telephones with a small direct view display. The main problem with this approach is the display resolution limit. A 2-inch wide display in direct view at reasonable distances from the eye limits the field of view to 10-12 degrees. At a limit of 0.05 degrees per resolvable pixel, this translates into approximately 240 pixels. Most graphic web content would have to be re-written for such low resolutions. Although there are efforts to perform text translation automatically, conversion of graphical interfaces would be prohibitively expensive. The other problem with the approach of enhancing a telephone's intelligence is that users will come to expect full browser functionality, including full Java interpreter, pointer devices and potentially high computing capability for 3D graphics.

An alternative to enlarging a cell phone is to shrink a handheld computer into a telephone package and add a near-eye virtual-image microdisplay-based display system. This approach was pursued by the DARPA LCS/Marine project at the Laboratory for Computer Science in collaboration with the MicroDisplay Corporation and is described in this report. The LCS/Marine Communicator device contains a StrongArm microprocessor running a standard Linux multi-tasking operating system. The device accepts off-the shelf commercial PCMCIA cards as the wireless communication link. The communication modes supported are those available in this formfactor. During the course of the project these modes included 11 Mb/sec 801.11b wireless LAN and 19.2 Kb/sec wide area CDPD modem. Voice conversations are digitized and sent on top of the data medium. Human interaction is done with a combination of a track pointer, two mouse buttons, and voice input. There are no integrated keyboard interfaces. The device has other useful features such as a CompactFlash slot for low power, application-specific data loads, and a CMOS camera for image capture and transmission. This section describes the design of the handheld device (WebPhone) and the special purpose and custom components required to implement it, with primary focus on display, optics, and display driver electronics technology. Other aspects of the device electronics were implemented in a custom design built from off the shelf components, and are described in the section of this report addressing System Design.

7.2 Scope of Work, Approach, and Results

This section outlines the scope of work undertaken to develop the WebPhone device at MicroDisplay Corporation. It also outlines the principle development strategy and milestones, and lists the results achieved.

7.2.1 Scope of Work

The technical tasks undertaken to develop the handheld device for this project included:

- Designing a silicon substrate for an liquid-crystal-on-silicon (LCOS) display
- Developing high performance liquid crystal modes, assembly methods, and drive schemes.
- Designing a sequence of successively smaller and lower power drive electronics, culminating with a drive ASIC.
- Designing several generations of mechanical packaging for the handset and evaluating usage modes and human factors.
- Designing an operating-system independent display subsystem.
- Designing a compact optical eyepiece for a reflective display.
- Designing the core electronics, including StrongArm processor board, display drivers, audio, I/O subsystem, power controller, etc.

7.2.2 Approach

The technical development proceeded in five parallel tracks: LCOS display, display controller, computing platform, mechanical packaging, and optical subsystem. The approach changed somewhat through the life of the project because of what was learned.

The milestone deliverable approach, in which the parallel development tracks synchronized were:

System	Subsystem	Notes
WP-1 Human Factors prototype	Display	800 x 600 analog LCOS display
	Controller	PC video (VGA) input stage, discrete analog output, analog controls
	Computer	Laptop
	Mechanical	WP-1 rigid housing; display holder only
	Optics	V2 eyepiece
WP-2 Table-top development system (not handset)	Display	800 x 600 analog LCOS display
	Controller	Shadow frame buffer, discrete analog output, analog controls
	Computer	StrongArm SA-1100
	Mechanical	N/A
	Optics	V2 eyepiece
WP-3 Large, "carphone" form factor handset	Display	832 x 632 analog LCOS display, improved brightness
	Controller	Shadow frame buffer, analog ASIC output, digital controls
	Computer	StrongArm SA-1110
	Mechanical	WP-3 Case: rotating eyepiece, heat sink, battery provision, etc
	Optics	V2 eyepiece
WP-4 Not implemented	Display	832 x 632, digital input, low power
	Controller	ASIC
	Computer	StrongArm SA-1110
	Mechanical	Extending package
	Optics	Wedge optics

7.2.2.1 Phase 0: Start of Program

Much of the technology taken for granted at the end of the project simply did not exist at its start. As the program started, the highest resolution LCOS display demonstrated by MicroDisplay had a resolution of 640 by 480, flickered after an hour of operation, and had poor uniformity. The drive electronics occupied a page-sized circuit board and dissipated 12 W. The adjustable focus eyepiece was bulky and fully resolved pixel sizes of no smaller than 15 microns.

7.2.2.2 Phase 1: Human Factors Prototype

In this phase, an SVGA display was designed and fabricated, compact color drive electronics were designed, an improved eyepiece was designed and assembled, and a mechanical housing was completed. The resulting handset, when attached to a portable computer, could be used to perform human factors studies.

7.2.2.3 Phase 2: Table-Top Development Platform

In this phase, the identical mechanical case was attached to a tabletop StrongArm development system. This configuration would be used to develop software and optimize interfaces for the small screen and non-desktop pointer control configuration.

7.2.2.4 Phase 3: Large “Carphone” Sized Handheld

In this phase, the tabletop development system was to be integrated into the casing. In addition, the mechanical case would be re-engineered to address all the human interface issues identified in phase 1.

7.2.2.5 Phase 4: Compact Sized Handheld

In this phase, a standalone handheld device of more usable size and improved usability was to be developed, making use of advances in display technology, display driver electronics, and packaging developed earlier in the project. This portion of the work was not completed.

7.2.3 Results

The program has resulted in the development of:

- Custom 800 by 600 resolution LCOS display with 12.55-micron pixels with 24-bit capability and a contrast ratio of over 100, capable of supporting full motion video.
- Four generations of drive electronics that reduced size from a shoebox to a credit card and power consumption from 12W to 1.9 W.
- A low-power digital-input high-performance LCOS architecture suitable for both near-eye and projection applications.
- Several iterations of device packaging, addressing optics design and human factors issues in a near-eye high-resolution display.

7.3 Mechanical Design and Packaging

The WebPhone housing went through two design iterations and one proposed future design. The WP-1 and WP-3 were built (WP-2 was a small modification of WP-1).

7.3.1 WP-1

The initial WebPhone mechanical design is shown in Figure 4. The body is a single piece, with a fixed viewing angle. There is no provision for inserting PCMCIA cards or compact flash. Similarly, there is no provision for cooling. The WP-1 was intended to house only the video circuitry and plug into the video output port of a PC. The fixed eyepiece is in a lookdown configuration, and not obstructing forward view.



Figure 4. WP-1 Unit in the desired configuration

Due to human resources issues, the first iteration of the WP-1 did not contain the drive electronics, but only the display, eyepiece, and mouse pointer. The system configuration consisted of the WP-1, an external drive electronics box (BB-3), and a laptop computer. As a result, the combined system was too bulky to carry. A subsequent iteration integrated the BB-4 drive electronics into the casing.

7.3.2 WP-3

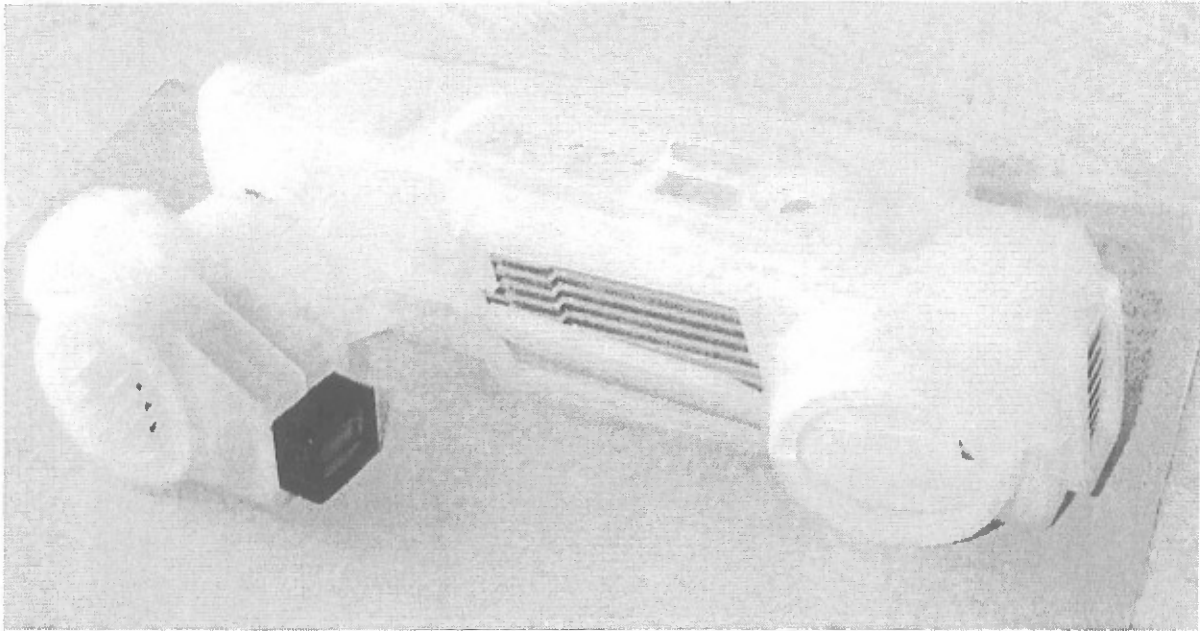


Figure 5. Initial model of the WP-3 configuration

A photograph of the initial mechanical design is shown in Figure 5. The unit appears as a standard handset-sized telephone, but with an adjustable eyepiece assembly. The bottom of the rotating assembly contains the digital camera unit. The earpiece is visible at the upper left. On the top of the unit are the slots for the track pointer and two rectangular mouse buttons. Additional controls to the right of the mouse buttons are for emergency display adjustments. Also visible are the openings for the CompactFlash card on the lower right and for the PCMCIA card on the lower left.

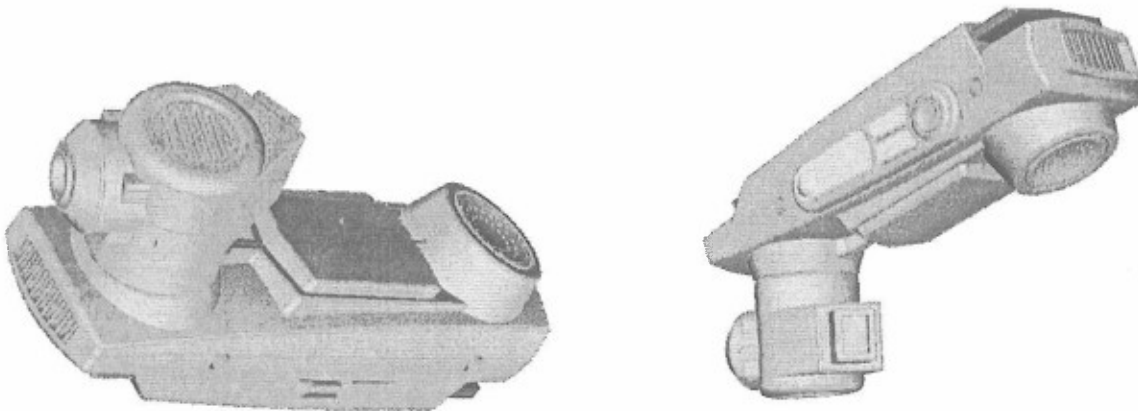


Figure 6. WP-3 Handset Industrial Design

Two views of the CAD model are shown in Figure 6. Some of the features that can be seen on the side view are the turret with the eyepiece, the circular joystick with the three buttons, the

earpiece, battery door cover, the heatsink ribs, a PCMCIA card protruding from the housing. The bottom view shows the serial and charging port cover.

The body of the handset is constructed of 11 molded plastic parts with a machined aluminum heatsink providing support for the other major body components. This heatsink runs the length of the handset and was designed to dissipate the approximately 7 W that the electronics within the unit were estimated to consume. The microdisplay and associated optics lie within the turret assembly visible in the figures, allowing the user to position the viewing optics in a manner that is the most comfortable for them. The camera assembly lies in this same turret assembly, consisting of the CMOS camera and its associated lens. The earpiece lies at the top end of the handset opposite the microdisplay turret and houses the speaker. The battery pack lies under the cover that lies between the microdisplay turret and the earpiece. The WebPhone also provides a microphone integral to the handset and located at the base of the turret housing the microdisplay; a position that is intended to optimize voice signal quality. The PCMCIA components are visible in the views from the backside of the handset. Plugging in from the top of the unit is the PCMCIA LAN card, while the CompactFlash card plugs in from the opposite end.

Referring to the figures, on the top edge of the handset near the earpiece lies the mouse pointer button for cursor control, consisting of a pixie-pointer with a "push-to-z" feature effectively providing a third mouse button. The right and left mouse buttons are adjacent to the mouse pointer button towards the Microdisplay end on the handset. The raised button that is in line with the mouse pointer button and mouse buttons but close to the microdisplay turret assembly is the "wake-up" button. This button lies under a removable rubber cover. This same cover shields 6 additional buttons that were originally intended to provide a means to adjust the microdisplay parameters, including ITO voltage, brightness, and contrast. In consideration of the consistent characteristics of the displays provided with these handsets, the display control buttons were deemed unnecessary and the handsets are to be delivered with the rubber covers fixed in place.

7.3.3 WP-4

The WP-4 mechanical design was not completed for this project. The architecture was to be based on the "slide out" design implemented as a mockup by Virtual Vision. The next generation of system circuit boards after the SX-2 would have required a volume of approximately 25% larger in length and thickness than the model shown below.

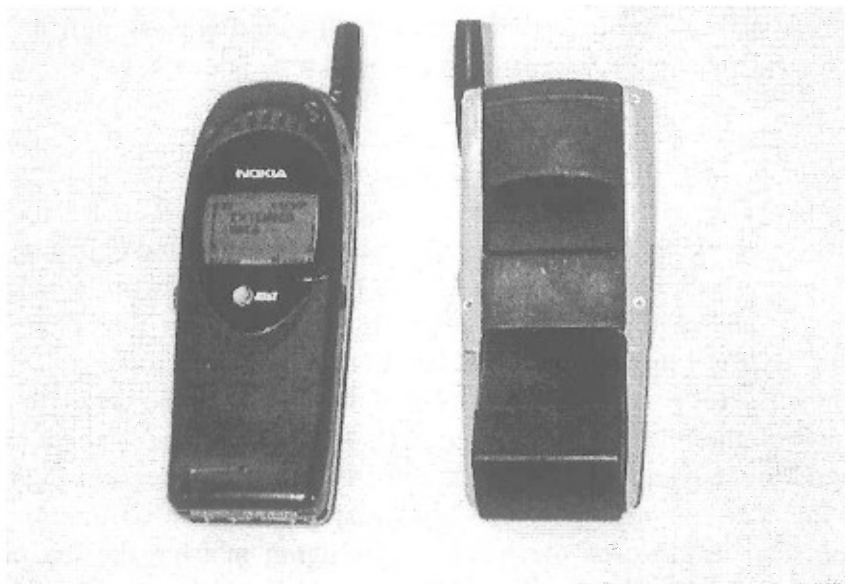


Figure 7. Mockup of WP-4 Packaging. The unit is shown in a closed configuration and is compared to a Nokia 6100 series cell phone. An open configuration is shown in Figure 21.

7.4 Displays and Drive Schemes

A high quality LCOS display requires bringing together several pieces of technology. The display substrate must be designed for low power and good yield; the semiconductor process must be optimized for flat, reflective, and mechanically robust die; a high contrast, fast-switching liquid crystal operating mode must be designed; and a repeatable cell assembly procedure must be developed. Over the course of the program, all of those separate components were achieved and demonstrated. However, the program was halted before a handset that incorporates all these improvements was completed. This section describes these achievements.

7.4.1 Background

7.4.1.1 LCOS Technology

An LCOS display consists of a three-element stack: a metallized silicon die, liquid crystal material, and a transparent conductive cover. The display uses a twisted-nematic (TN) liquid crystal (LC) in which the LC molecules twist between the two electrodes, rotating incident light to pass through a laminated polarizer. When an electric field is applied, the molecules orient vertically, destroying the twist, and not rotating the light. A thin film polarizing beam splitter (PBS) in the optical path passes light whose polarization has been rotated from horizontal to vertical. Light that is not rotated is transmitted back into the illumination optics, and results in a dark pixel. Light that is partially rotated to either vertical or horizontal appears gray.

From a circuit viewpoint, the brightness of a display at a particular pixel is controlled by applying a voltage between the glass and pixel electrodes. The charge-storage-based pixel consists of a CMOS transmission gate and a capacitor formed by both diffusion/substrate and inter-metal

dielectric structures. There is an additional non-linear capacitor formed by the pixel-LC-coverglass structure.

7.4.1.2 Field Sequential Color

An LCOS microdisplay as described above is intrinsically a grayscale device, but can be used to generate color images with a field-sequential color (FSC) technique. In this method, each image frame interval is divided into three fields corresponding to the three color components of the image (red, green, and blue). Each interval consists of three phases:

1. Update the pixel elements with the current color component.
2. Wait for the liquid crystal material to change state.
3. Illuminate the display with the current color light source.

The time required for the three phases determines the maximum refresh frequency and therefore the image quality of the display. The standard update rate for a MicroDisplay LCOS device is 60 Hz frame rate (which corresponds to 180 Hz field rate). Although the device can operate at over 90 Hz, the lower frequency reduces overall power and produces more saturated colors. The nominal 180 Hz field rate corresponds to 5.55 milliseconds (ms) per field, of which approximately 3 ms is required for drawing, 2 for the LC transition, and 0.5 for the LED illumination. The faster the pixel array is updated, the more time the LC material has to transition, and the more accurate the colors. This video bandwidth is challenging, as the native pixel clock is 160 MHz.

7.4.1.3 Analog Interface Architecture

Figure 8 shows a typical analog display architecture in which the video signal drives a horizontal wire connected to each pixel column via an analog switch. The switches are controlled by a shift register containing a single high bit that connects each column in turn to the video wire. A similar vertical shift register connects all pixels in the active row to their column wires. As each analog pixel value is placed on the video wire, it is sampled by the column wire and the currently active pixel. After all pixels in a row have been loaded, the vertical shift register advances and the process repeats.

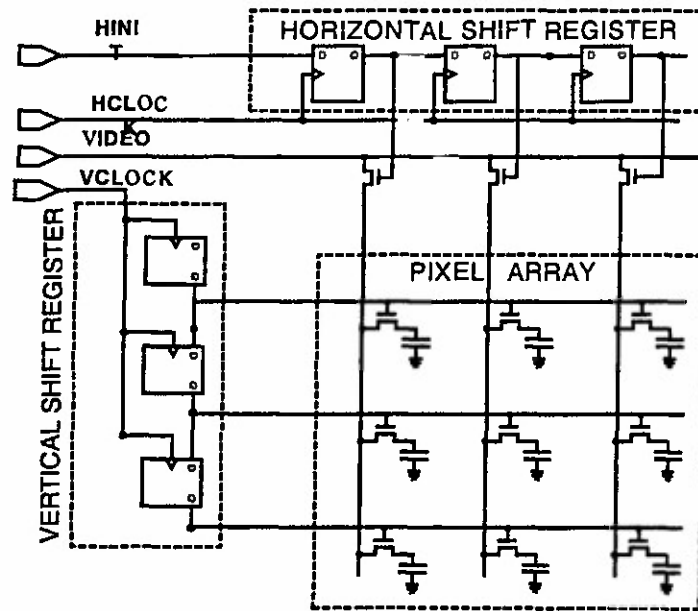


Figure 8. Analog Display architecture

In medium resolutions such as VGA and higher, the single video wire shown in Figure 8 is split into several parallel wires that are connected to interleaved columns. This results in a reduction of signal bandwidth on each wire. The reduction helps in two ways. First, system clock speeds are reduced from 160 MHz to a more manageable 40 MHz (in a 4-input design). Second, the long video and column wires on the display chip have a large RC (resistance/capacitance) time constant. Allowing longer propagation times results in more accurate pixel voltages.

Displays with analog interfaces pose several problems for portable system designs. First, the driver circuitry requires multiple digital-to-analog converters. The signal paths between the DACs and the display must be kept noise free. In the current LC mode as little as 16 mV of noise can lead to an error of 1 part in 128, reducing bit precision to six bits per pixel. From a systems point of view, several off-chip analog circuits must operate at the 5V required by the LC material rather than the 3.3 or lower voltages required by the rest of the digital drive circuitry. This leads to increased power consumption. Integrating the digital to analog conversion on the display chip reduces the display's noise susceptibility and reduces system power.

From another practical viewpoint, drive chips for a digital display are more economical, since need not integrate any analog component and therefore can be done with a range of digital implementation technologies such as field-programmable gate arrays or metal-mask gate arrays.

7.4.2 Generation 1: MD800G6

7.4.2.1 Silicon Substrate

During the first three months of the program, MicroDisplay rapidly designed, fabricated and tested an 800x600 display, MD800G6, with 12.55-micron pixels. The G4 display had several improvements over the previous generation of 640 x 480 (VGA) displays. The pixel pitch (and

therefore shift register pitch) was reduced to 12.55 microns, keeping the identical diagonal. The bonding pad layout was redesigned to reduce bond wire lengths and therefore the required size of display carrier. Both displays have been demonstrated to be fully functional on the first pass. The VLSI design proved to be the most straightforward step.

7.4.2.2 Semiconductor Process

Although the first assembled displays produced reasonable images, results on subsequent wafers were worse. There were a number of process issues.

1. Custom pixel etch step. Due to the low power and low-voltage requirements of the system, all 5 V of pixel voltage needed to be applied across the LC material. Voltage drops across the typical protective passivation material were not acceptable. An extra pixel etch process step was added to etch the passivation over the active pixel region. Since passivation etch steps are usually applied over large metal bonding pads, which stop the etching process, this process step is usually not required to be well controlled. In the case of the display, the pixel etch etches away at the dielectric between pixels, causing overetch of the inter-metal dielectric between the top two layers of metal. In other parts of the display, the passivation was underetched, causing non-uniform pixel heights and therefore varying electric fields. We performed successful joint process experiments with the foundry to refine the etch step.
2. Interaction of topology and process. The unusual topology of the display substrate, consisting of very large sheet of metal 2 pierced regularly by vias, affected several other process steps in the typical foundry sequence. Various solvents spread slower than expected and collected in places. The result was poor mechanical strength in the vias and therefore many unconnected pixels. This effect, of groups of white pixels, was termed the "starry night" effect and eventually eliminated, by changing both the design and the process steps.
3. Wafer-scale planarity effects. The display substrates proved to have non-uniform planarity. After considerable investigation, we determined that the effect was partially caused by the irregularity of the multi-project wafer. The G4 display was one of four displays in the reticle, and the topology above and below the display was not identical. Since large-scale topology produced by the chemical-mechanical polishing (CMP) method could be affected by underlying features over scales of over 1 mm, interaction with features of adjacent, different displays could lead to non-planar devices. The fix is to use single-display wafer designs, and to ensure that the topology on all four sides of the display is identical out to at least 1 mm.

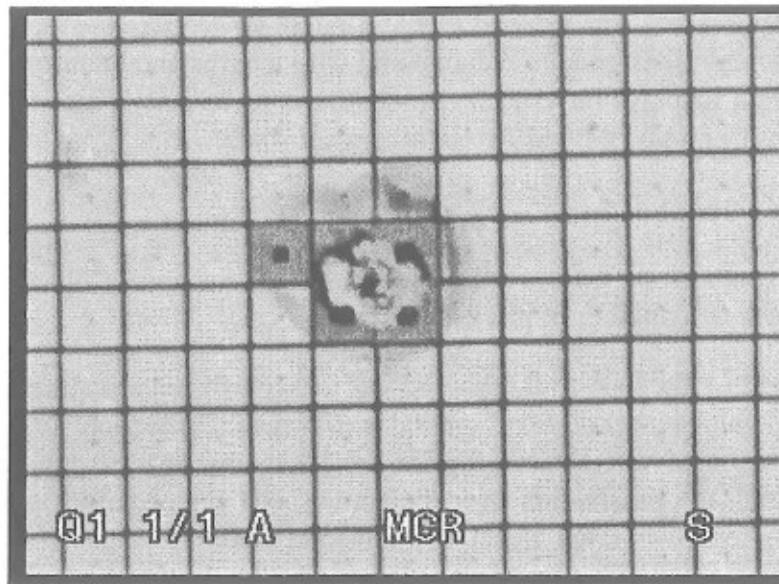


Figure 9. Microscope picture of typical early defect on the G4 substrate

Troubleshooting these effects required considerable effort. An engineer was dispatched to Singapore for over a week to work at the foundry. Multiple scanning electron microscope (SEM) images were taken at a local failure analysis lab. Experiments involving layout interactions required new mask design and fabrication. This research step was not expected and proved both expensive and time-consuming.

7.4.2.3 Liquid Crystal Mode and Assembly Method

The high-performance LC mode developed early in the project changed relatively little in theory. However, the LC material was changed from passive matrix to active matrix material. Most of the subsequent effort consisted of fine-tuning the real-world parameters such as polyimide thickness; exact rubbing direction; and cell gap control to match the theoretical model. When prepared properly, the active matrix material virtually eliminated image sticking. Contrast and color saturation also improved due to the faster switching time of the material.

Considerable characterization infrastructure needed to be developed to analyze experimental data. We set up spectral contrast and switching speed measurement systems. The ITO bias voltage on the displays also needed to be measured because the assembly process introduced sufficient variability that the built-in battery voltage (ITO bias) would drift on some displays. This drift leads to flickering displays. The assembly procedure was eventually made more robust. Surprisingly, bulk wafer-level cell assembly produced results superior to manual assembly due to consistent cell gaps and uniform film coating.

7.4.2.4 Drive Schemes

The most significant innovation in the drive scheme is the implementation of the Flash Clear technique that was theorized just prior to the program start. The technique is described in detail

in the Evaluation Kit documentation in the auxiliary package. The basic idea of the technique is to take advantage of the non-symmetric switching time of the display, and start the slow transition in parallel with updating the display pixel data. The refinement added during the course of the program is to use an intermediate voltage instead of an extreme voltage to optimize the average transition case.

There was also considerable experimentation with the use of the "flipping ITO" drive scheme. Previous schemes have fixed the glass electrode (ITO) voltage at 2.5V and centered the inverting video signal on 2.5V. This requires a 5V range, which is at the limit of the semiconductor process. To limit mechanical damage to the display surface and improve yield we experimented with retaining passivation over the pixels. This results in a voltage drop across the passivation material and therefore requires a higher applied voltage. We modified the drive scheme to "flip" ITO between 0 and 5 V, and invert the signal appropriately from the extremes. The driving scheme worked well, but some sort of surface property interaction between the passivation and the polyimide coating resulted uncontrollable ITO drift and therefore increasing flickering over time. In another experiment with a particular LC recipe, we increased the field rate from 180 Hz to 240 Hz to reduce flicker. Flicker was greatly reduced, but at the cost of poorer color saturation and increased power dissipation.

7.4.3 Foundry Availability Issues

In early fall of 1999 we were notified that our foundry, Chartered Semiconductor, was going to stop supplying MicroDisplay with wafers. We spent the next three months selecting a foundry, creating technology files, and analyzing the effects of porting the display design to these foundry candidates. We ported various sections of the display design to technologies from United Microelectronics (UMC), Fujitsu Microelectronics, Amkor Technologies, and American Microsystems (AMI). After our evaluation, we settled on Amkor as the best selection, followed by AMI.

This foundry effort was substantial. Not only did we create technology files for four different processes, but we also created "merged" technology files so that displays developed in an "Amkor-AMI" tech file could be fabricated in either process (after appropriate IO pad substitutions). We performed significant simulation work, circuit extraction, and floorplanning studies.

An SVGA display with larger pixels was for fabrication in early February of 2000. That design incorporated various circuit elements, such as an LVDS receiver. We received that design back on April 8 and have verified that the display works as expected. The LVDS receiver worked at over 100 MHz.

7.4.4 Generation 3: MD832P3

The next generation digital display being developed by MicroDisplay uses a sampled ramp digital-to-analog converter. In this scheme, voltage conversion is transformed to a time-based conversion as shown in Figure 10. A voltage ramp is distributed on a wire shared by all column switches. At the start of the ramp, all columns connect to the video line and track the slowly rising ramp. The digital value corresponding to the ramp voltage is broadcast to comparators at

each column that disconnect their column when the ramp exceeds the stored pixel value. The voltage held by the column is the D/A conversion result. The data distribution and voltage ramp phases are pipelined, slowing the ramp time to line rates. Arbitrary DAC transfer curves including gamma and LC correction can be implemented by altering the ramp voltage and count synchronization.

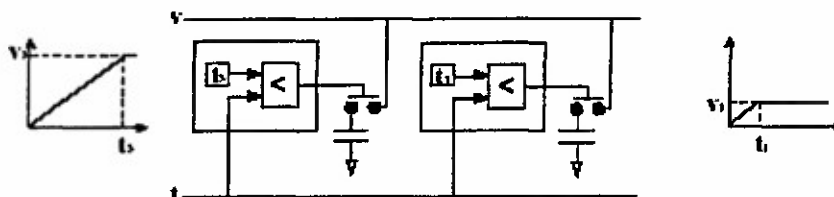


Figure 10. An Example Sample Ramp Conversion. The switches start out closed, so that the column capacitor tracks the rising ramp. When the broadcast time value exceeds the digital value stored in register t_1 , the comparator opens the switch, thus holding the voltage V_1 on the rightmost column. The leftmost switch operates similarly until the time exceeds t_3 .

7.5 Drive Electronics

7.5.1 Overview

A number of drive electronics circuit boards were developed throughout the program. Each time, the drive unit decreased in size and power consumption, yet increased in capability. Most of these were later converted into commercial evaluation kits for the larger size displays.

Board	Power	Size
EvalKit-2	14 W	Tabletop model, shoebox size. Modified from existing design.
BeltBoard-1	4.2W	Paperback book-size stack of two boards.
BeltBoard-4	2.6W	Smaller, cigarette pack size stack of two boards
BeltBoard-5	1.8 W	2" x 2" stack of two boards. Contains BeltChip-1.

Aside from the very first grayscale unit developed for rapid prototyping purposes, all other boards were for color display. Since the microdisplay is intrinsically grayscale, color is achieved by displaying successive red, green, and blue components of an image and illuminating the display with the appropriate color light source. This technique is called *field-sequential color* (FSC). There are four basic components in any subsystem that converts from an RGB signal to an FSC signal:

- Analog-to-digital converter to sample the pixel data
- Frame buffer memory to store the incoming RGB data and output a single color at a time

- Digital-to-analog converter to drive the microdisplay
- A digital controller to coordinate the three other subsystems and provide timing signals to the display.

BeltBoard-1 reduced size and power by replacing four separate digital controller chips (one for each color plus a display controller) and three memory chip groups with a single FPGA and a single memory chip group.

BeltBoard-4 reduced size and power by eliminating an analog signal conditioning stage (not described) before by digital computation, shrinking the power supply section, and reducing pixel precision to 18 bits per pixel

BeltBoard-5 reduced size and power by merging most analog functionality into a single custom analog ASIC and halving the frame buffer capacity to optimize for mostly static images.

7.5.2 EvalKit-2 (EK-2)

An existing 640 x 480 tabletop evaluation kit was modified to support 800 x 600 resolution for this program. Specifically we added analog circuitry to support the flash-clear feature required for high resolution. We also developed a number of mechanisms to compensate for the increased noise generated by the higher-speed operation. In addition, the prototyping area on this board was used to develop driver circuitry for clock distribution over a long signal cable. The modified board displayed an 800 x 600 image, but was plagued by analog noise caused by the higher clock rates of the SVGA resolution. The EK-2 circuitry is shown in Figure 11.

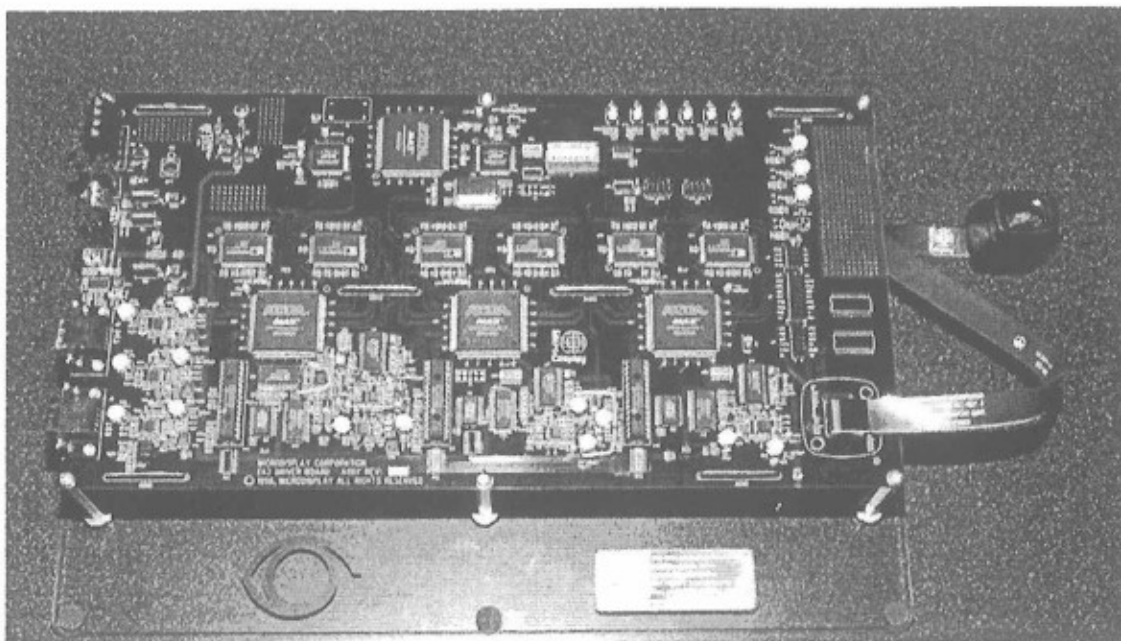


Figure 11. EK-2 Evaluation Board. Approximate size is 12" x 6".

7.5.3 BeltBoard-1

BeltBoard-1 was designed to consist of two separate circuit boards, one for the analog section and one for the digital section. This partitioning had several benefits. First, the design, layout, assembly, and test processes could be overlapped for the two boards, reducing overall completion time. Second, high frequency noise caused by the digital section could be isolated from the analog section by having to travel through the vertical connector. Finally, subsequent optimizations to either the analog or digital sections could be implemented faster as long as the interface was preserved. The initial BeltBoard-1 unit was demonstrated on schedule, in the middle of February 1999. Minor improvements were made in the analog boards, and the release version was shipped in March of 1999. The BeltBoard-1, in its external box, is shown in Figure 12. The approximate size is 6" x 3"; roughly one quarter that of the previous design.

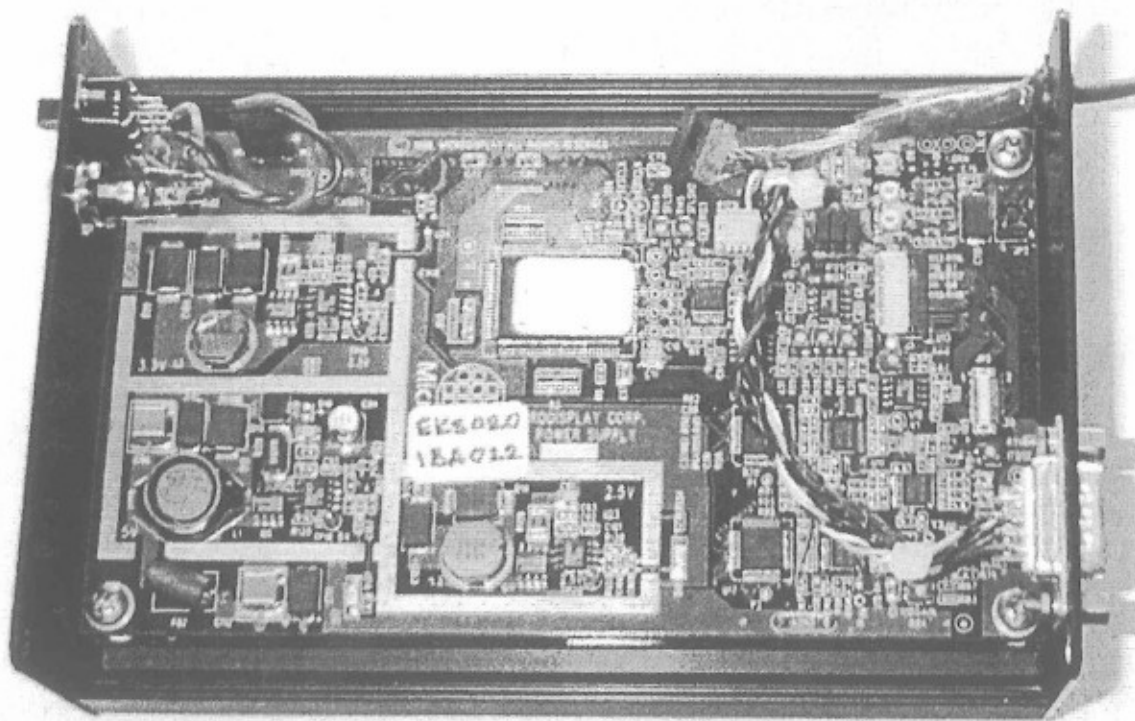


Figure 12. BeltBox-1 in its boxed configuration. Only the top analog board can be seen.

7.5.3.1 Features

BB-1 incorporates a number of features that allow rapid experimentation with liquid crystal recipe modifications, drive schemes, and general usability.

1. A battery-operated evaluation was run on the BeltBoard system. The results were excellent with the system able to run 1.5 hours on 8 AAA cells.
2. A series of test were also run using the long-cable wherein the displayed image exhibited no deterioration because of the extended cable length.
3. Test modes such as constant intensity screens and grayscale ramps were built into the FPGA to allow field calibration without an image source.
4. The BeltBoard now detects an absent video signal and shows a test mode image. This is critical to preserving displays, as the liquid crystal display must be driven to a net zero DC signal, even in the absence of valid video. Extended periods of a DC bias across the LC can permanently damage the display. The test pattern is now displayed whenever the video source is disconnected to inform the user that it is the video source rather than the display that is inoperable
5. An on-board PIC microprocessor now controls a number of user-configurable display parameters such as brightness and contrast, as well as some parameters related to the display "recipe" that should be invisible to the user. These latter parameters have been made programmable to allow flexibility in future display assembly techniques.

6. Over a dozen parameters are now controlled from the four buttons and four switches on the BeltBoard. As this flexibility requirement did not exist during the initial board specification, the BeltBoard was not designed with sufficient user feedback mechanisms. The only available method to tell the user that the parameter currently being modified is, say contrast rather than brightness, is to flash the LEDs in a specific pattern. This is an inefficient feedback mechanism
7. The BeltBoard can now mirror the display in the X and Y directions.

7.5.3.2 Problems and Possible Solutions

The main problem with the BB-1 design is the large number of analog parameters that needed to be manually adjusted and had to match very precisely. Each of the four video channels had a four-stage operational amplifier sequence that performed the signal conditioning required by the LC material. A resistance mismatch caused by mechanical disturbance (such as being shaken while in transit), or by thermal effects, resulted in the four separate analog video channels having slightly different gains and offsets. This was seen as a visual artifact of vertical stripes every four pixels.

A related cause of visual artifacts was that the displays being produced at the time exhibited "ITO drift". In this phenomenon, a charge would build up over time inside the LC cell, creating the effect of a battery in series with the pixel electrode. As a result, electric field during different inversion phases was not balanced, and flicker was evident. The short-term solution to this effect was to tune the ITO voltage control to compensate for this internal battery. The miniaturized potentiometers were difficult to adjust, and frequently an adjacent control, such as brightness or contrast was adjusted instead of ITO.

Due to the analog configuration method, it was very difficult to maintain a factory calibration setting, as the gain and offset (contrast and brightness) parameters were coupled, and modifying one affected the other. Excessive adjustment required lab bench recalibration.

Finally, the analog output stages controlled brightness and contrast by fitting a gain and offset line to the reflectance transfer curve of the LC material. Since that curve is not linear, any particular setting was a compromise between the number of black and white levels available in the image.

Most of these issues could be addressed by performing the LC signal conditioning digitally, storing the factor configuration parameters digitally in non-volatile memory, and eliminating all but one of the analog transformation stages. The final stage is still required to amplify the voltage output of the digital-to-analog converters. An additional benefit of a digital drive scheme would be the ability to control the brightness and contrast through the digital input.

7.5.4 BeltBoard-4

The most user-visible change from BB-1 to BB-4, aside from the smaller size and improved image quality, was that the display could be configured from a PC with a graphical user interface via a serial port. Configuration settings were stored.

The key idea of DSC is to perform the signal conditioning with a look-up table that maps the desired digital intensity values to the appropriate voltage levels for the LC material. The values in the table are computed by a microcontroller, based on button-press information, and then stored in a look-up table in the Altera FPGA. This relatively expensive computation, involving multiplications and divisions is performed by the microcontroller instead of in the FPGA hardware. Since these computations occur at human response rates (as the sliders are manipulated) the microcontroller has no trouble keeping up. The lookup technique avoids power and space-hungry arithmetic circuits in the FPGA.

Some of the settings that can be controlled with this method are brightness, contrast, the electro-optical response of the liquid crystal, and the tuning of the liquid crystal voltage range. The results so far have been very good, resulting in easily adjustable and high quality displays when used with the BeltBoard drivers.

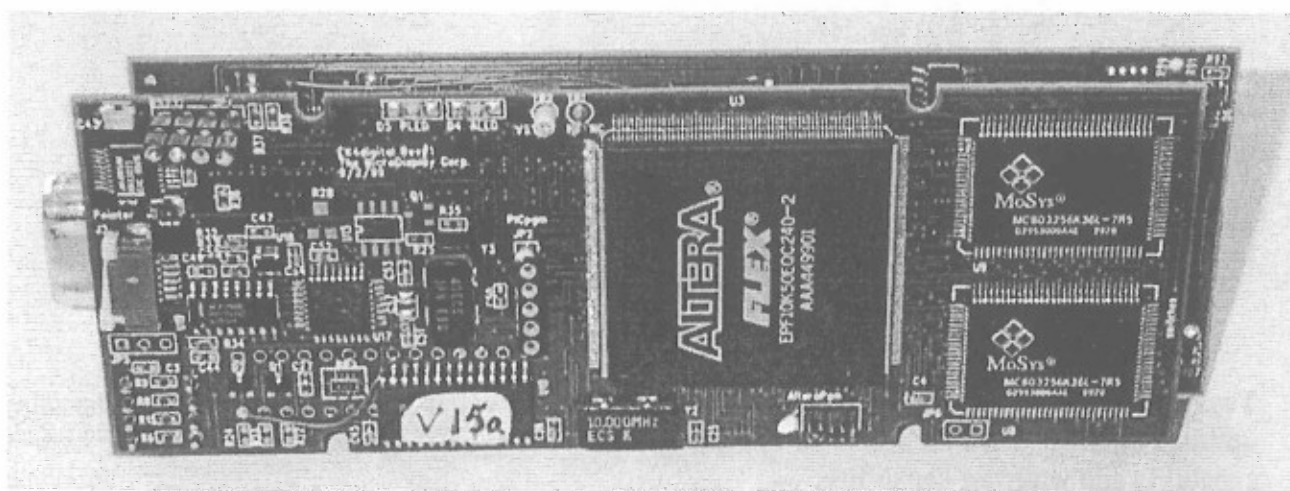


Figure 13. Side view of the BB-4 stack. The digital board is at the top. Most of the area is the large FPGA and the two frame buffer memories.

The four-stage analog pipeline that performed the computation of signal gain, offset, ITO inversion and final gain has been replaced with a single gain stage. This has reduced power and circuit board area significantly. The power dissipation of BB-4 is 2.8 W.

7.5.5 BeltBoard-5

BeltBoard-5 is the most compact color drive circuit yet. As a result of the high level of integration in the BeltChip, the use of a ball-grid array package for the FPGA, and new high-density memory, and custom miniaturized connectors, the entire control board now measures only 2" by 2". The board is powered by one of several power supply boards, designed to stack vertically with the main board

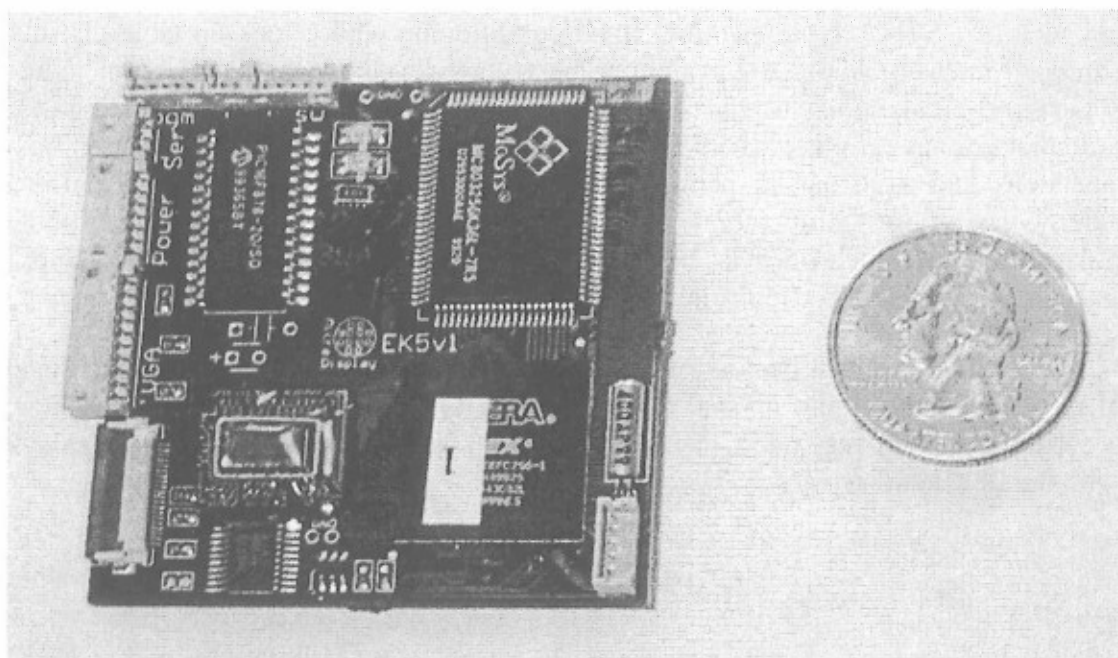


Figure 14. BeltBoard-5. The four visible components are the FPGA (in a BGA package), the frame buffer memory, the chip-on-board BeltChip and the PIC controller

Two power supply boards were built to date. The BB5/PW1 is based on linear regulators (size optimized). BB5/ PW2 is based on switching regulators (power optimized). Both have battery capability and wallwart capability.

The design met the 2.5" x 2.5" size target. The actual size for each board in the stack is actually 2" x 2". The circuit board works as expected, and supports a modular power supply system. All drive circuitry is on one PCB; all power supply circuitry is on another. Thus, the tradeoff between the large size of a switching power supply and the power inefficiency of a linear power supply can be made application specific.

The power target of 2W for the drive electronics was also met and bettered. We measured 1.86 W on the drive electronics board. The existing power supply module is somewhat inefficient and dissipates 0.54 W on its own, for a power conversion efficiency of 78%, and a system power dissipation of 2.4W (1.4 A at 1.7V). With a pair of 1400 mAh rechargeable NiMh AA batteries, the display system should have a lifetime of just below two hours.

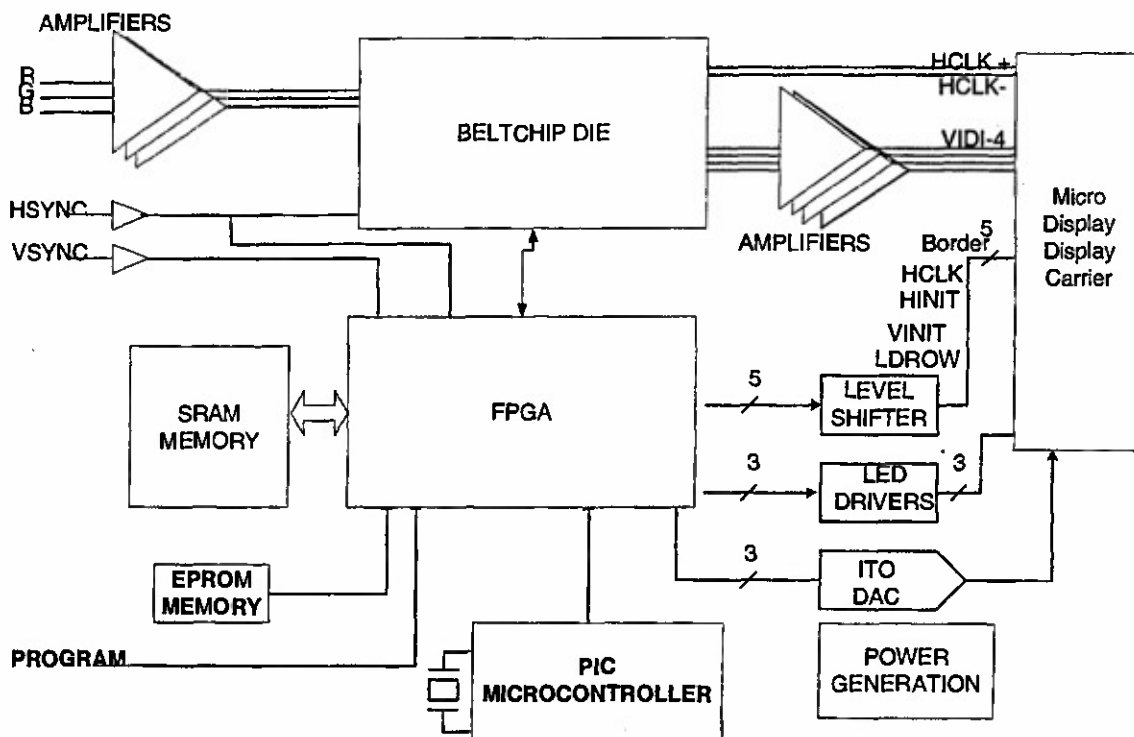


Figure 15. Block diagram of a BB-5. Notice the discrete components between the FPGA and the display connection. All of these are scheduled to be included in the next generation display.

The main problem with the BeltBoard is the assembly and reliability difficulties with the compact chip-on-board assembly. The packaging vendor took four weeks to assemble 9 boards because the wirebond task was outsourced to another specialist. After debugging the board, we had to change one bonding pad location, and shortly thereafter had 6 operational boards. However, within three days, all but one suffered from intermittent failures, all related to bond wire connections to the BeltChip. Some of them were made to work as long as a finger was pressing on the die. Of the batch of nine assembled boards only one worked reliably.

7.6 Optics

The design of a compact direct-view display with high usability was a principal goal of the project. A number of optics approaches were explored and developed.

7.6.1 Design Goals and Constraints

The optical system for the WebPhone has a number of challenging constraints. It must provide full color, high-resolution images, with low power consumption, large field of view, comfortable prolonged viewing, and tolerance to head motion. Compact sub-inch reflective LCOS devices offer the required image quality at low power. However, the sub-inch diagonal display must be enlarged in some way to allow for easy viewing and text readability.

An 800 by 600 pixel (SVGA resolution) device with 12.55-micron pixels was selected for this project. The eye can easily perceive 0.05 degrees as one pixel. Therefore, the display subtends 40 degrees of arc; yielding horizontal field of view is +/- 20 degrees.

For comfortable viewing over extended periods of time, long eye relief is necessary. To understand this requirement, consider reading text such as email for 15 minutes through a camcorder viewfinder. The precise positioning of one's eye, and its prolonged separation from the rest of the world, together make reading under such circumstances unpleasant and lead to eyestrain. It is therefore desirable to peer from a distance towards a display: to not be cut-off from the rest of the world in an obtrusive way. Ultimately, a see-through display will allow the least intrusion between the viewer and the natural world. It has been historically difficult to design such "augmented reality" eyepieces that in addition allow the user to see high contrast images in bright sunlight. Operation of the device in bright sunlight is as a higher priority than transmissive or augmented reality mode operation. As a result, the scope of this project confined itself to exploration of opaque type eyepieces and display schemes.

To increase usability of the display, a large exit pupil is necessary. Typical eyepieces, such as telescopes, require a user to be very precise located. For extended viewing times that allow for user mobility, such as walking, a large viewing zone or exit pupil is necessary. The minimum exit pupil was set at one inch after a review of the literature on eyepieces.

7.6.2 Initial Viewer Prototypes

The initial viewer in the handset prototype shown in Figure 1 consists of a standard MicroDisplay eyepiece, which consists of illumination optics, a beam splitter, a microdisplay and a magnifying lens. An on-board LED without a lens was used. The light emitting from the LED spreads into a cone of approximately +/- 20 degrees. This cone angle is limited by the refractive index difference between the air and the LED substrate. The exit angle of the light is confined by Snell's law; all other light from the LED is totally internally reflected. A schematic diagram of this eyepiece is shown in Figure 16.

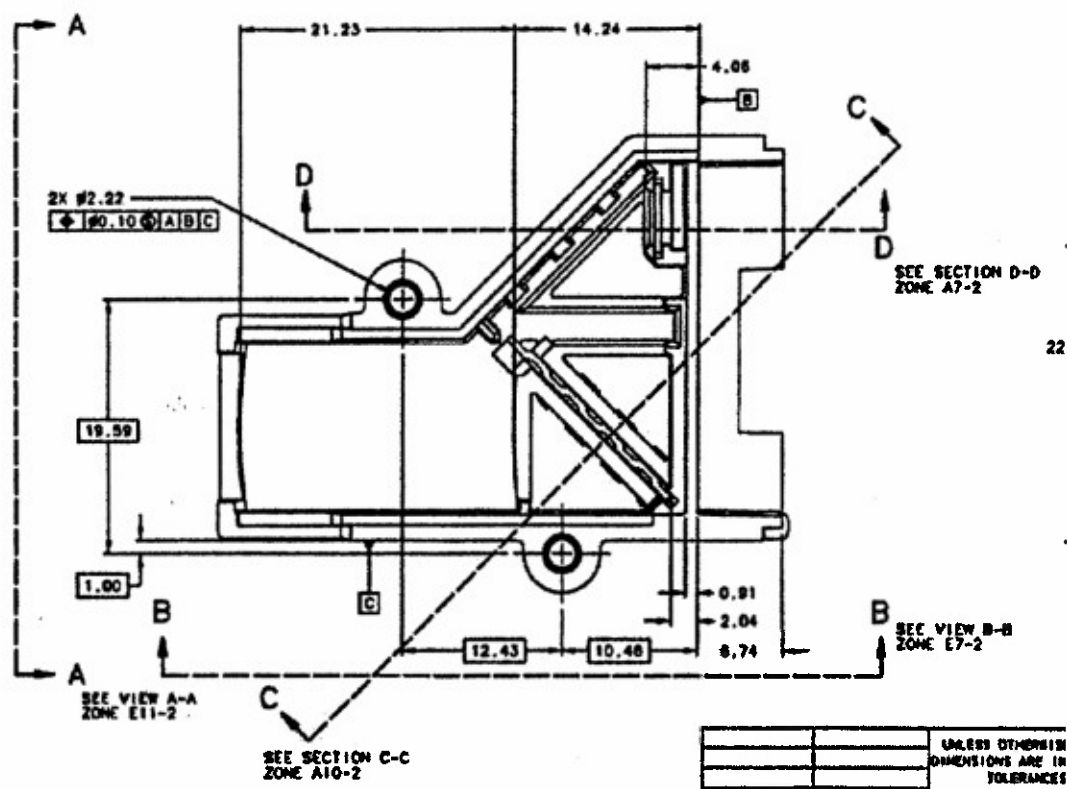


Figure 16. Cross-section of the 1st generation eyepiece

Three LEDs, one for red, green and blue respectively, are placed within 1mm of each other on the LED substrate. This slight displacement in the LEDs ordinarily would lead to color shift in the viewer's plane as a function of eye position. At one spot the blue LED appears brighter, at another the red. To solve this color shift problem a weak diffuser is placed within 1mm of the LED bank.

As the light continues on its path it next encounters a fold mirror at 45 degrees, which causes a 90-degree change of direction for the light. Next, a Fresnel lens is used to collimate. The Fresnel lens has a focal length matched to the distance from to the LED, in our case approximately 15mm. The light is then sent through another weak diffuser to smooth out some of the locally bright and dark ridges created in the light field by the Fresnel lens. A polarizer is the final component in this sandwich, which absorbs all but the vertically polarized illumination. It is necessary to polarize the light prior to incidence on the liquid crystal on silicon (LCOS) device. See Section 3.1 for details on LCOS operation.

The now polarized light next encounters a polarizing beam splitter. A thin-film type polarizing beam splitter was chosen because of its good performance over a wide angular range. Conventional prism type beams splitters have narrow angular ranges of performance. In addition, they are heavy, being made of glass, and expensive.

The polarized light now is reflected down to the microdisplay LCOS device. As described in section 3.1 the light which is rotated by 90 degrees corresponds to the bright pixels and the non-

rotated light corresponds to the dark pixels. The partially rotated light corresponds to the various shades of gray pixels.

The light emerges from the microdisplay and unrotated light is reflected back into the polarizer sandwich. The rotated light is transmitted through the thin film polarizing beam splitter and into the eyepiece lens. After several design iterations on the lens we have chosen a stock Kellner triplet with the wide, sharp field of view, excellent aberration correction and relatively long eye-relief. The resultant field of view is ± 22.5 degrees, with an eye relief of 1 inch. The exit pupil has a diameter of 0.5 inches.

7.6.3 Mini-Projector Flip Display

The "straight-through" eyepiece design described in Section 7.6.2 suffers from several drawbacks. First, it is difficult to package in a small form-factor. More critically, it is difficult to use. This is due to the limited eye relief and exit pupil, which demand accurate positioning of the device with respect to the user's eye. This rigid positioning requirement was found to make it extremely difficult to "pick up the phone and go".

To both extend the eye relief and exit pupil, we have chosen to adopt a flip-phone where the final piece in the eyepiece is the flip-out like component of a standard cell-phone. The use of the flip allows a larger optical system to be used. It can be flipped into place, while keeping the size of the phone small when the flip is folded back into the phone.

Rather than using the flip surface directly in front the viewer's eyes thus obscuring the head-on view of the world from the user, the user peers slightly downwards into the flip surface. The flip surface is a concave mirror and acts as the final lens of the optical system. The flip-out piece of the cell-phone is far enough away from the eye to allow comfortable viewing while keeping the display optics small enough to occupy a physically small volume. The flip is approximately 2.5 inches away from the viewer's eyes, but the viewer perceives an image that fills a 16 inch diagonal approximately 20 inches away: a ± 19 degree field of view. The viewer doesn't focus on the flip surface, but through the flip surface onto the virtual image created by the optical system.

The light that is transmitted into the eye first encounters a relay lens, and then an off-axis parabolic mirror. The lens diagram is included below. Such a catadioptric system allows a long optical path through the phone without taking up lots of volume. The microdisplay is embedded near the earpiece of the phone allowing a long separation between the relay microdisplay and the final folded mirror. This design is essentially a mini-projection optical system in a telephone. The system differs slightly from a true projector in that the final optical surface, the catadioptric mirror, has optical power. Thus, the image appears behind the flip of the phone and is enlarged accordingly. The current eye relief of this system is 65 mm and the exit pupil is 25mm. The field of view of ± 19 degrees and the image appears 20 inches away with a 16-inch diagonal.

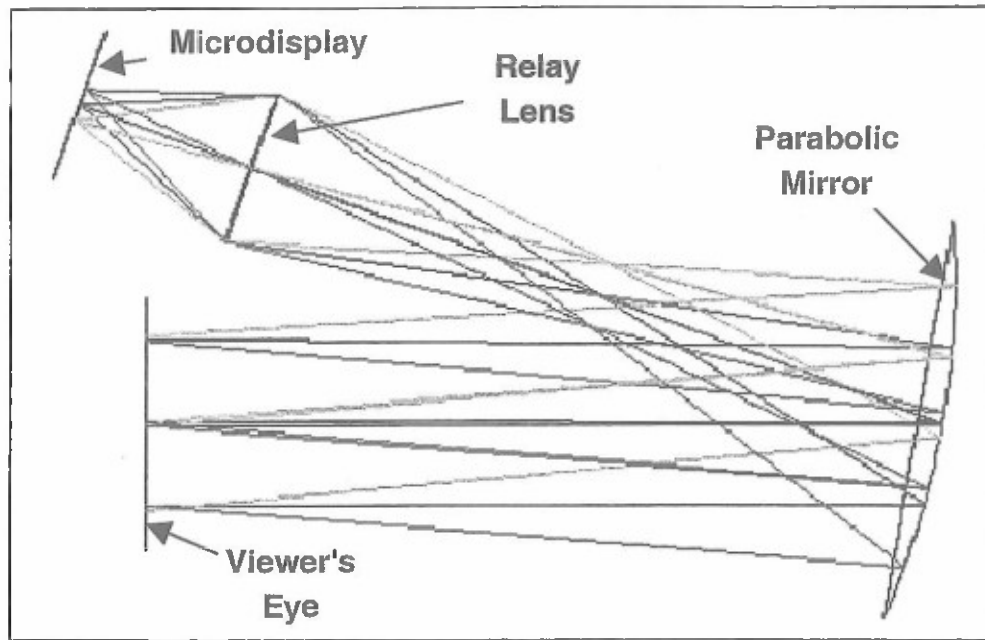


Figure 17. Optical model of mini projector. The view is looking down from above the user's head. The microdisplay is in the upper left corner. The parabolic mirror on the right side flips down to the horizontal plane. The viewer's eye is in the lower left with the very large exit pupil. The image, to the eye, appears to be floating behind the parabolic mirror at a 20-inch distance with a 16-inch diagonal.

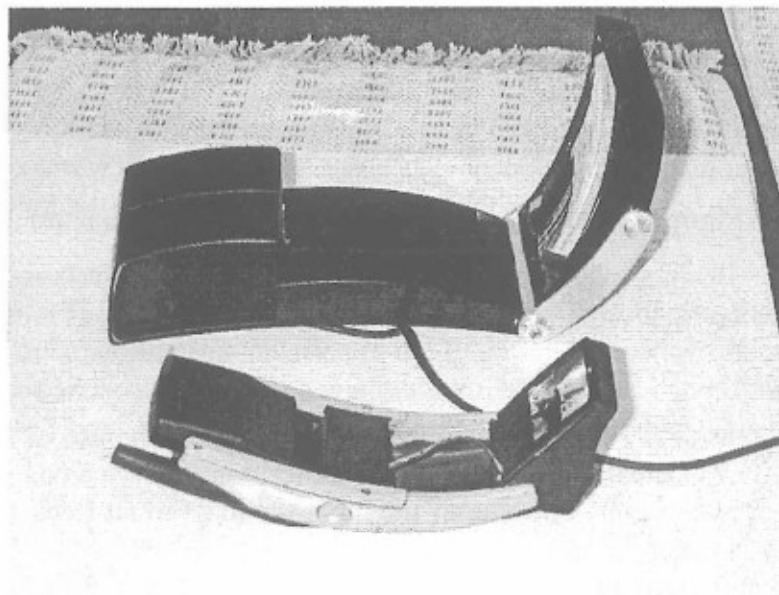


Figure 18. The prototype of the flip out curved display compared to the WP-4 mockup

7.6.4 Mangin-Type Flip Display

Another design evaluated in the course of the project involves the use asymmetric lenses: lenses with front and back surfaces of differing curvature, with an aluminum coating on the back curved surface of the lens. Such optical components are traditionally called Mangin mirrors after their inventor. Our exploration has concentrated on using such components at an oblique incidence. Essentially, a traditional lens is sliced at an angle such that light is coupled into the first refractive surface at a grazing angle of 10-20 degrees from horizontal. Incident light at such a steep angle is totally internally reflected (TIR) back to the second surface. This second surface is a curved mirror. Light incident on that curved mirror is then redirected according to Snell's law back to the first surface, but on this pass the light is transmitted through this surface, rather than undergoing a TIR, because the angle of incidence now allows transmission (rather than total internal reflection). Now the light continues it's path and is focused into the exit pupil and then into the eye.

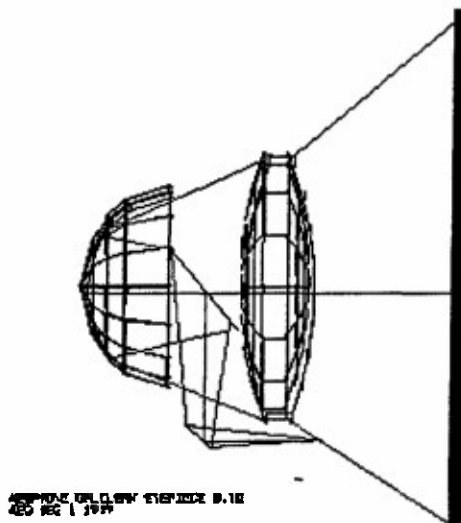


Figure 19. Mangin-type flip display optical diagram

7.6.5 Optics Approach 3: Fiber Fuse

A third approach evaluated in the course of the project involves the use of fiber optics to carry the image from display to the user's eye. The basic concept is to place a compact fiber optic fuse directly in front of a microdisplay. Some of the highest end "Virtual Reality" systems use such an element to increase the angle of view and magnification. The field sequential illumination can be achieved with by coupling LEDs through either the top or bottom using edge-lit illumination. For example, a leaky mode plate can be employed. Several field sequential modes of MD illumination will be explored. Additionally, an optional black and white mode is available which requires no LEDs but uses ambient illumination and works well in high brightness applications.

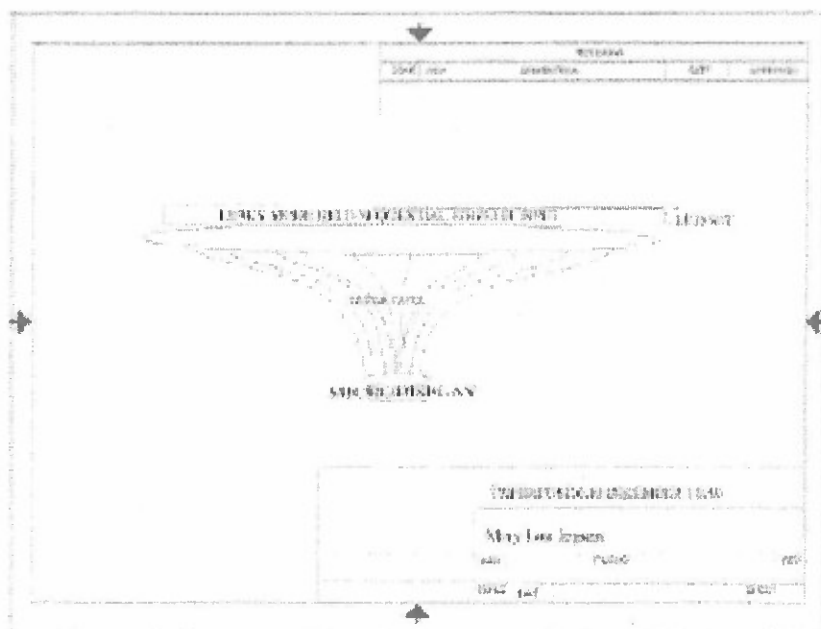


Figure 20. Fiber Fuse-based Optics

Fused, tapered fiber optical systems are usually expensive. One approach is a monolithic etch of a plastic material to lower manufacturing cost. This manufacturing path is novel and thus high-risk, but high potential payoff for not just present application.

Our experimentation has shown that the fiber optic taper design will not work well as a standalone solution. This is because the Coverglass thickness of the microdisplay is set to increase, due to manufacturing needs relating to die flatness. To use the fiber optic taper solution in a monolithic mode we would need to use a thinner cover glass of approximately 0.3mm. We are currently using 0.7 mm and are very likely to switch to 1.2 mm thickness coverglass.

Although it would be possible to experiment with even thinner coverglass (0.3mm), that approach would have other problems, and would be difficult to produce reliably for dozens of deliverable displays. This approach was abandoned.

7.6.6 WP-4 Optics: Wedge Optics

Some preliminary work was performed on the WebPhone-4 handset. At the time of termination of the project, the active approach was based on a wedge optic, in which the display is aligned

with the short edge of a curved wedge-shaped optic, as shown below. The advantage of this approach is that the wedge could flip in and out of the casing when not in use.

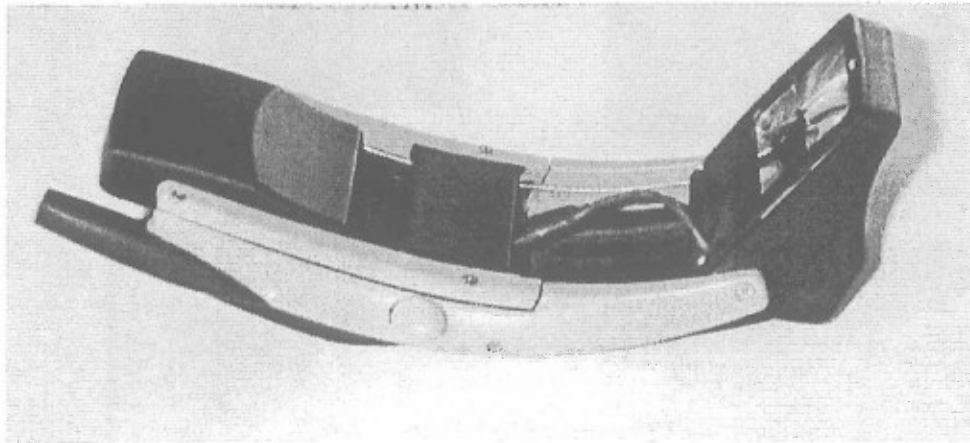


Figure 21. WP-4 Mockup with Flip-out wedge optic

Evaluation of this approach using a display mockup and lens system designed under contract by Virtual Vision was promising. The display mockup consisted of a functional wedge optic lens, “driven” by an image on photographic slide film and a transmissive illuminator. Experiments showed that many users, including several with corrective glasses or contact lenses, were able to adapt quickly to this display approach, and found it intuitive to use. Experiments did reveal the need for an adjustable locking device to control and customize the angle of the open display flip for different users.

7.7 Display Controller ICs

This section describes the Integrated Circuits developed in the course of the project.

7.7.1 Analog System: BeltChip

The original goal for the program was to provide a single-chip controller that incorporated all functionality except for the frame buffer memory. However, as the program went on it became clear that committing the digital control logic to silicon too early was a poor idea, since the drive schemes were in constant flux. The architectural decision was made to develop the system as a two-part solution: a low-power integrated analog subsystem, and a digital component that could be prototyped in an FPGA. There are several methodological advantages to this approach. For example, the digital logic can be operated at a lower voltage than the analog subsystem to optimize power.

The analog chip replaced most of the analog board of BB-1, and contained the following subsystems:

1. Triple Flash 5-bit Analog to Digital converter.
 - Samples the analog RGB input of a standard PC signal.
 - Features flash architecture and high-accuracy resistor ladder.

2. Quad Digital-to-Analog Converter

- One for each of the four video input channels of the display
- Supports up to 65 MHz output rate for experimenting with higher frame rates
- Low-power data conversion at 3.3V; only final gain stage requires a single 5V external opamp.
- Off-chip resistor for process compensation setting.

3. PLL Pixel Clock synthesizer

- Synthesizes pixel clock from external VGA sync signals
- Accommodates nonstandard video timing through support for external divider.
- Outputs frequency-doubled pixel clock to control high-speed external frame buffer SRAM

4. Crystal oscillator

- Robust self-biasing XTAL oscillator requires compact external crystal.
- Can be overdriven by external oscillator

5. LVDS output driver

- Low-power, low-noise clock signal over a long cable
- Biased by on-chip bandgap reference

6. Internal ROM Gamma tables

- Compensates for non-linear liquid crystal response
- Bypass capability in case of changing liquid crystal formulations.

7.7.1.1 BeltChip

The BeltChip design was developed as part of an integrated effort to reduce the power, size, and cost of a suitable high-speed DAC required to drive an LCOS SVGA-resolution microdisplay. Its primary function is to sample the four channels of 8-bit field sequential color (FSC) video data presented to the chip simultaneously and generate the corresponding analog data to be provided to the display. The simultaneous four-channel presentation of video data reduces by a factor of four the clock rate required to paint a field, or, alternatively, reduces by a factor of four the amount of time required to paint the display.

The BeltChip contains a set of three video-rate ADCs, intended to digitize a 1.0 V_{pp} video signals at rates of up to 65 MHz. These devices are not used in the SX-2 design. In order to achieve these speeds, a flash-type converter is employed. A total of 31 compare-and-latch blocks feed directly into a ROM encoder. A polysilicon ladder network provides the reference step potentials to each comparator.

Although also not used in the SX-2 boards, the BeltChip contains an internal PLL that is intended to generate the pixel clock from hsync. Depending on the divider selected, 800, 1056, or 1344, the PLL generates a 25.157 MHz, 40 MHz, or 65 MHz pixel clock.

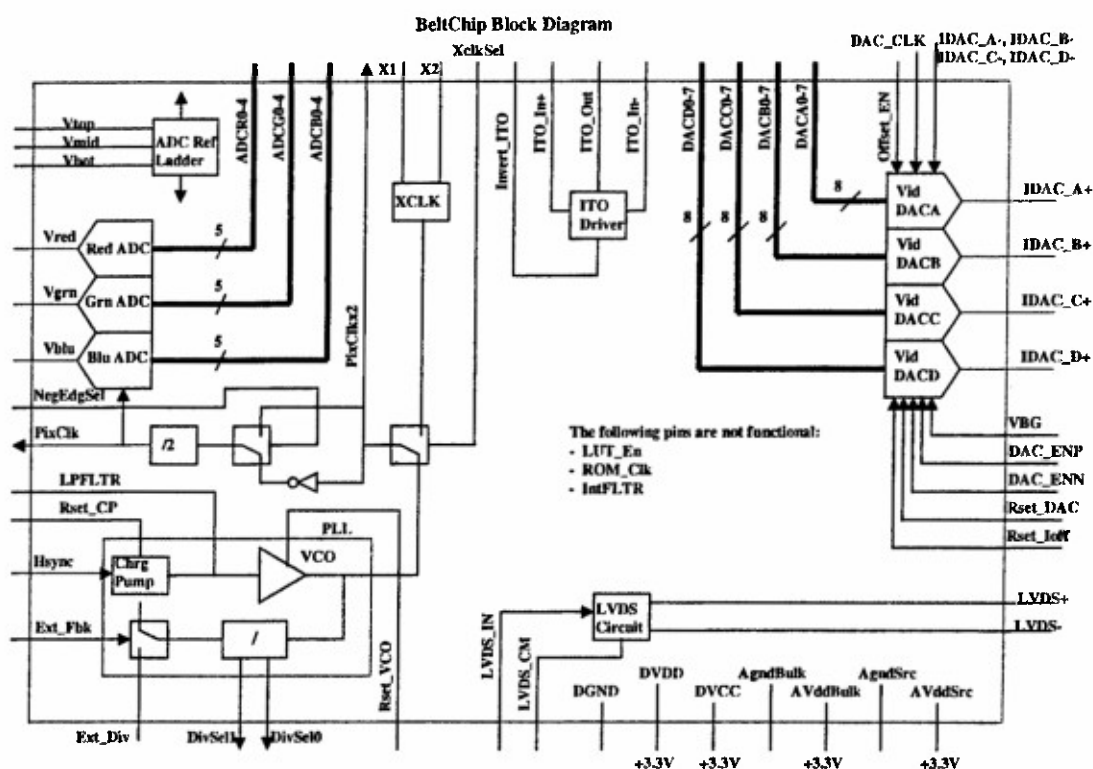


Figure 22. Block Diagram of BeltChip

The architecture of the BeltChip is shown in Figure 22. The device contains 3 8-bit ADCs, one for each component color, a single PLL, and four 8-bit DACs. For the SX-2 boards, only the 8-bit DACs are used. The hclk signal is provided to the LVDS receiver in the BeltChip, however, the LVDS receiver is not used in this design. That is, the LVDS outputs from the BeltChip are left open.

Figure 23 shows details of the critical display driver components of this device. Each of the BeltChip DACs consists of a pair of sub-DACs; a P-DAC and an N-DAC, each with a resolution of 8 bits. The P-DAC is used to source current from the output node, while the N-DAC is used to sink current from the output node. Since it is necessary to flip the video signal about the ITO voltage (nominally 2.5 V), the P-DAC is used below ITO and the N-DAC is used above ITO. Nominally, the outputs of the DAC pair will be biased at half V_{DD} , or 1.65 V, thereby balancing the voltage seen across the N-devices or P-devices in their corresponding sub-DAC. An external amplifier is used to shift the center point up to 2.5 V and amplify the output swing to the full 0.0 V to 5.0 V range required by the display.

Figure 23 also shows a video waveform generated by a DAC and then amplified to the full-scale 5.0 V signal. Note that when the output voltage is equal to the ITO voltage, the resulting video

will be white. Alternatively, when the output voltage is at either of the extremes from ITO, that is 0.0 V or 5.0 V, the resulting video will be black.

The two switching current sources shown connected to the outputs of the sub-DACs are used to produce a black level offset to the video signal. Only one current source is on at any given time. The current source connected to 3.3 V is on at the same time that the P-DAC is sourcing current and the current source connected to ground is on at the same time that the N-DAC is sinking current. An external resistor is used to set this current level for the desired black level offset.

To reduce the overall size of each DAC block, the decoders are split into groups of $\langle 7:3 \rangle$ and $\langle 2:0 \rangle$. The upper bits enable current sources that are equivalent to 8 LSB current sources. The lower bits enable 7 individual LSB current sources. This way, each DAC requires only 38 current sources (31 upper-bit sources and 7 lower-bit sources) rather than the 255 that would otherwise be required. The price paid for doing this is a noticeable increase in the charge injection at the output node during "multiple-of-8" code transitions and the possibility of non-monotonicity. However, since the entire DAC is now smaller, device matching improves, and the chance of non-monotonic behavior is very small as long as careful layout is employed.

An external resistor sets the reference current to all the DACs, thus providing a means to adjust DAC gain.

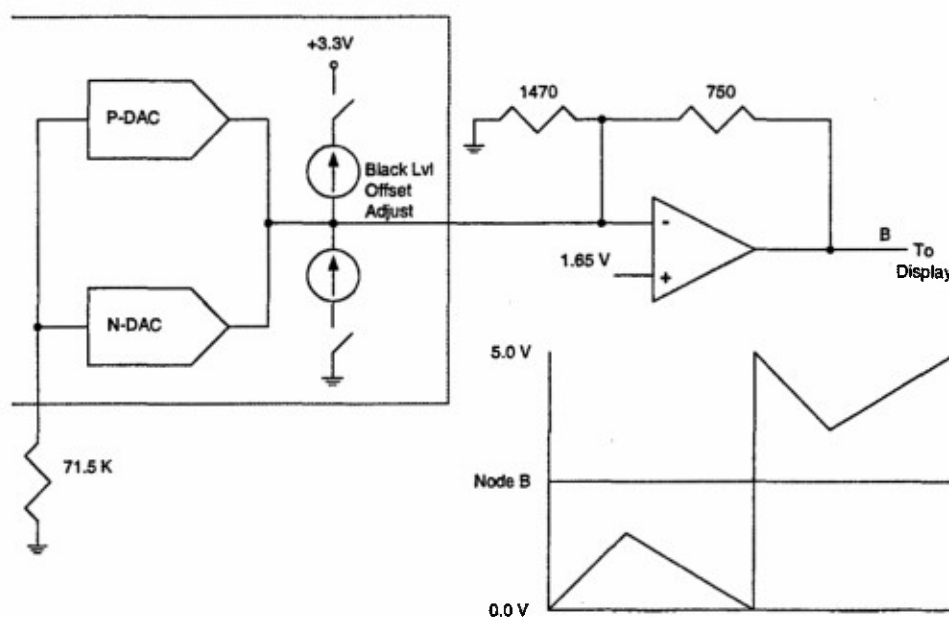


Figure 23. Block diagram of BC-1 output circuitry

7.7.1.2 BeltChip-1

The first step of this project was selection of the semiconductor process. We selected a Taiwan Semiconductor (TSMC) 0.35-micron process for two main reasons. First, the 0.35-micron technology consumes less power than a 0.5-micron alternative yet allows reasonably safe analog design. Finer process technologies such as 0.25-micron often require several iterations for analog designs. Second, for small quantities, prototyping runs through the MOSIS VLSI brokerage service were relatively frequent and had good support. The selection was finalized in December 1998, and the somewhat aggressive tapeout target date was set for the March 24th, 1999 multi-project run.

We made good progress through January and February, completing the schematics and simulations of most of the analog systems. We also completed the layout technology rule file for the process. The design entered the layout stage in early March and ran into some delays. For example, we had to port the input/output pads to this process. In another set of delays, the public-domain digital tool chain, called Alliance, that we began using had multiple bugs that needed to be worked around. The final set of delays occurred during the system integration stage, when all the functional blocks needed to be integrated. As the design was analog full custom, considerable manual work was required to route signals around noise-sensitive components. As a result, the design was submitted one day late. Unfortunately, MOSIS informed us that our design was dropped from the run basically over a space issue. By dropping our single design from this shared run, MOSIS was able to include three smaller designs to their reticle. The next TSMC run was not going to occur for another 3 months. We then began the search for a process to port the design to.

7.7.1.3 BeltChip-1B

We were able to obtain a reservation on a commercial multi-project through United Microelectronics (UMC), on a similar 0.35-micron process, with a May 10, 1999 deadline. The following changes were required to port the mostly analog design to a new process:

1. Remove I/O pads from one side of the design to fit into a UMC standard die footprint. This resulted in compaction of the rest of the pads and increased wiring.
2. Elimination of the on-chip VCO capacitor as the UMC process lacked a second level of polysilicon used for accurate capacitors in precision analog circuits.
3. Re-design, simulation and verification of all analog circuits for the poorer current drive performance of the UMC transistors characteristics.
4. Porting and shrink of the UMC-provided input-output pads.
5. Implementation of a new technology file.
6. Re-design of the LVDS pad architecture.

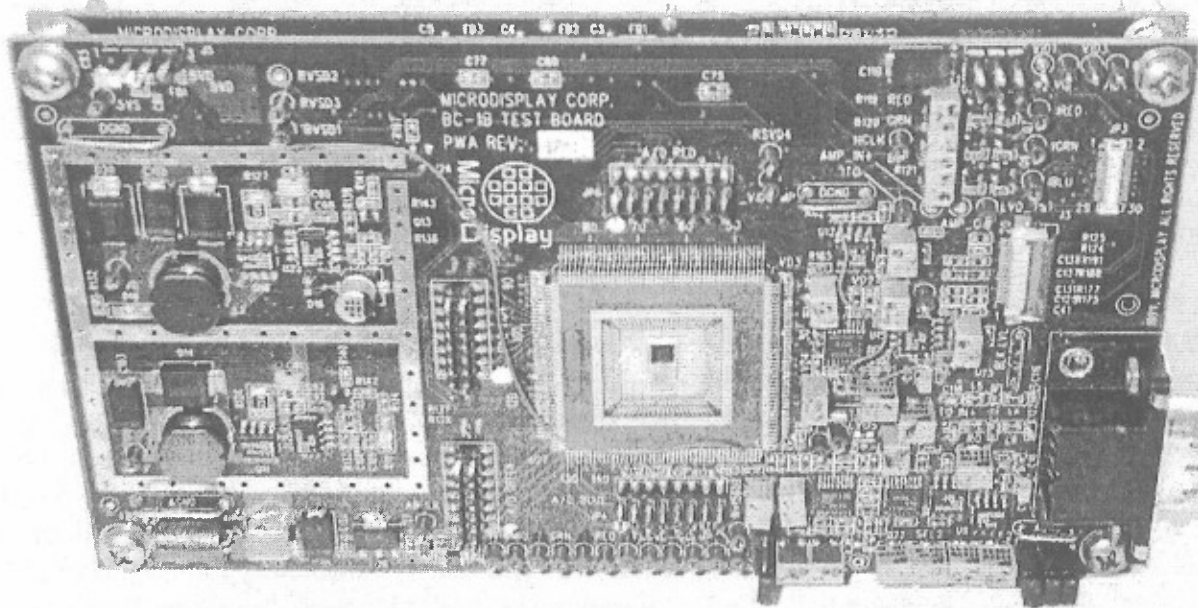


Figure 24. The BC-1B test board is shown above. Notice the size of the package relative to the BeltChip. Also notice the amount of space occupied by the miscellaneous discrete components and video header to the right of BC-1B and the power supply section on the left.

This porting process was completed in a remarkable four weeks by a team of only five engineers. The design was submitted for fabrication on schedule. All circuit elements were found to work.

There was considerably more work required before the chip could be tested. A package needed to be selected; arrangements with wire-bonding services needed to be completed, and a test board had to be designed, fabricated and tested. The test board was designed to replace the analog subassembly of BB-1 and therefore could directly plug into the digital subassembly.

We received the die two months after the originally quoted thirty-day turn. Despite the rushed porting effort, the device worked well.

7.7.1.4 Technical Learning from BeltChip-1B

A number of lessons were learned after incorporating BC-1B into a PCB system.

1. There are still a considerable number of discrete components surrounding the chip, as can be seen in the picture of the test board. These components are not sophisticated, but still consume a large amount of circuit board area. Some subsystems that could easily be integrated into the next BeltChip were the input video clamps, the slow DAC providing the ITO voltage, and the LED drivers.
2. The phase locked loop in BC-1B operates correctly but is very sensitive to the ground noise. Despite careful layout of BeltBoard-5, there is occasional timing jitter that corresponds to digital circuit activity. The voltage-controlled oscillator should be redesigned to improve noise immunity.

3. In our experience with BeltBoard-5, most of the circuit board problems, both in assembly time and reliability had to do with the chip-on-board assembly. The next generation of packaging should involve a commercial packaging vendor to assemble BeltChip-2 in a compact package such as a ball-grid array, rather than the larger quad flat pack that we used initially.
4. The 5-bit A/D digitizers produced minor image artifacts on natural scenes. 18-bit color, using 6-bit DACs is preferred.

7.7.1.5 Non-Technical Delays

In late June 1999, we were informed that UMC, our foundry, delayed the tapeout of our multiproject run by a month because another customer had design rule violations in their circuit. Design rule violations are significant not simply because the faulty circuit may not work, but mostly because features that are too small may cause small pieces of photoresist breaking off and contaminating other designs on the same reticle. Unfortunately, the foundry did not inform us of this delay until a week before the chip was due to arrive. An additional delay occurred when the boat of multiproject wafers were accidentally damaged and had to be re-fabricated.

This incident points out the risk and invisible costs associated with the low price of multi-project runs. Although the BeltChip has only cost \$10,000 to fabricate, it was delayed by over four months relative to a conventional full wafer run that would cost approximately \$72,000. In addition, four engineers worked for a month porting the design to a different process after our designed was bumped from the initial multi-project run. Future decisions to use multi-project runs should be made very carefully.

7.7.1.6 BeltChip-2

The first step in designing BC-2 was unfortunately the identification of a foundry. To ensure a timely fabrication cycle, we decided to use a dedicated chip run rather than a multi-project run. UMC initially indicated that they would accept such a design and several weeks later declined. We again had to research compatible processes. After several weeks, we selected a 0.35-micron process from American Microsystems (AMI).

We also performed preliminary simulations of the process port effects on the phase locked loop and the 8-bit DACs. The AMI transistors do not perform as well as the UMC transistors and would require a substantial redesign effort in the case of the delicate phase-locked circuitry. The DAC design was sufficiently robust to require only layout changes.

We completed an architectural study of the 6-bit ADC, redesigned the architecture, wrote linearity analysis software, and completed circuit simulations in the AMI 0.35u process. The change to the ADC architecture was substantial. To avoid simply doubling the size and therefore power of the 5-bit ADC, we chose a folded architecture, with separate paths for the coarse and fine bits.

7.7.2 Digital Controller: MC-2

The Display Controller IC is responsible for interfacing the display device to the CPU and related digital components. This device was designed by MicroDisplay engineers in close cooperation with MIT system designers. The objective of the development effort was to design a controller with two dominant characteristics:

- High performance: Performance should be sufficient to allow acceptable display of motion video images on the WebPhone.
- Appropriate programming model: The systems programmer should see the display as a reasonably standard video subsystem. Oddities and unusual characteristics of the field-sequential LCOS display should be hidden to the extent possible, oddities

7.7.2.1 Theory of Operation

The interface between the processor and display is critical to high performance in graphics rendering operations. The most obvious one is to timeshare the frame buffer between the display controller and the processor. This architecture was rejected for two reasons. First, the display update timing is critical, so the processor would have been locked out of the frame buffer while the controller was updating the display. This would have resulted in a 20% idle time by the processor. Second, to sustain the 40 MHz, 32 bit frame buffer read time, the color components are separated so that a single memory operation retrieves four consecutive pixels that are to be drawn. However, the vast amount of highly optimized existing frame buffer graphics code operates only with an RGB formatted frame buffer. Thus, if the processor needed to read a pixel to perform some sort of data manipulation, the memory hardware would have to do three random accesses, which would be prohibitively slow for full motion video operations.

The eventual solution was to implement a shadow frame buffer as shown in Figure 25. In this design, an interface ASIC observes the main memory bus of the SA-1100. Any write transactions to a predetermined memory location corresponding to the main frame buffer (MFB) in the DRAM based memory are snooped, and the equivalent memory locations in the shadow frame buffer are updated. Any read operations are performed on the MFB, which is organized in the conventional RGB memory format.

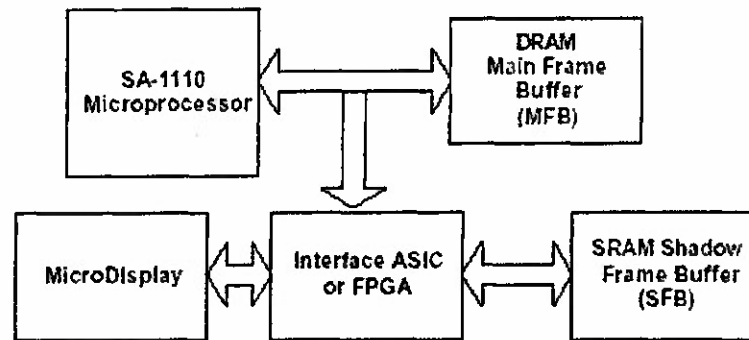


Figure 25. Block Diagram of the processor/display interface. The display controller snoops the main memory bus for write commands to the frame buffer and generates write operations

Since the interface ASIC is the only device that accesses the shadow frame buffer (SFB), its memory organization can be optimized for the field-sequential color organization. Other optimizations are possible. For example, to reduce power, the interface ASIC may choose to place the SFB in a standby mode after the pixels are updated. Since pixel draw time is typically only 50% of the total frame time, the SFB power consumption can be substantially reduced.. This technique will only work if there no frame buffer updates by the processor.

In another application, the interface ASIC can reduce power even further by not storing the actual image data, instead retaining an optimized version. If gray scale operation is selected, the interface ASIC can compute the luminance from each snooped RGB triplet and store only one byte instead of two. In addition, the number of pixel updates and therefore the number of memory accesses of the SFB is reduced by a factor of three. Note that this image conversion occurs in the interface ASIC and is therefore application independent. The interface ASIC must be able to generate an interrupt that forces the processor to redraw the entire frame buffer. Such a redraw event is required only when the user switches between display modes.

The initial prototype of the FPGA code was based on the code developed for the SX-1 board. This code did not meet the WebPhone performance requirements, in particular, the performance of the original snooper, which was not able to keep up with even non-burst continuous writing of video data without losing a fair amount of the data. A second set of code was developed and written in VHDL that corrected these problems. This new code features new snoopers, both SRAM and SDRAM; deeper FIFOs into and out of the SRAM frame buffer; an improved arbitration scheme to govern the read and writes from/to the SRAM frame buffer; and a new display controller. This section describes the second generation MC-2 implementation.

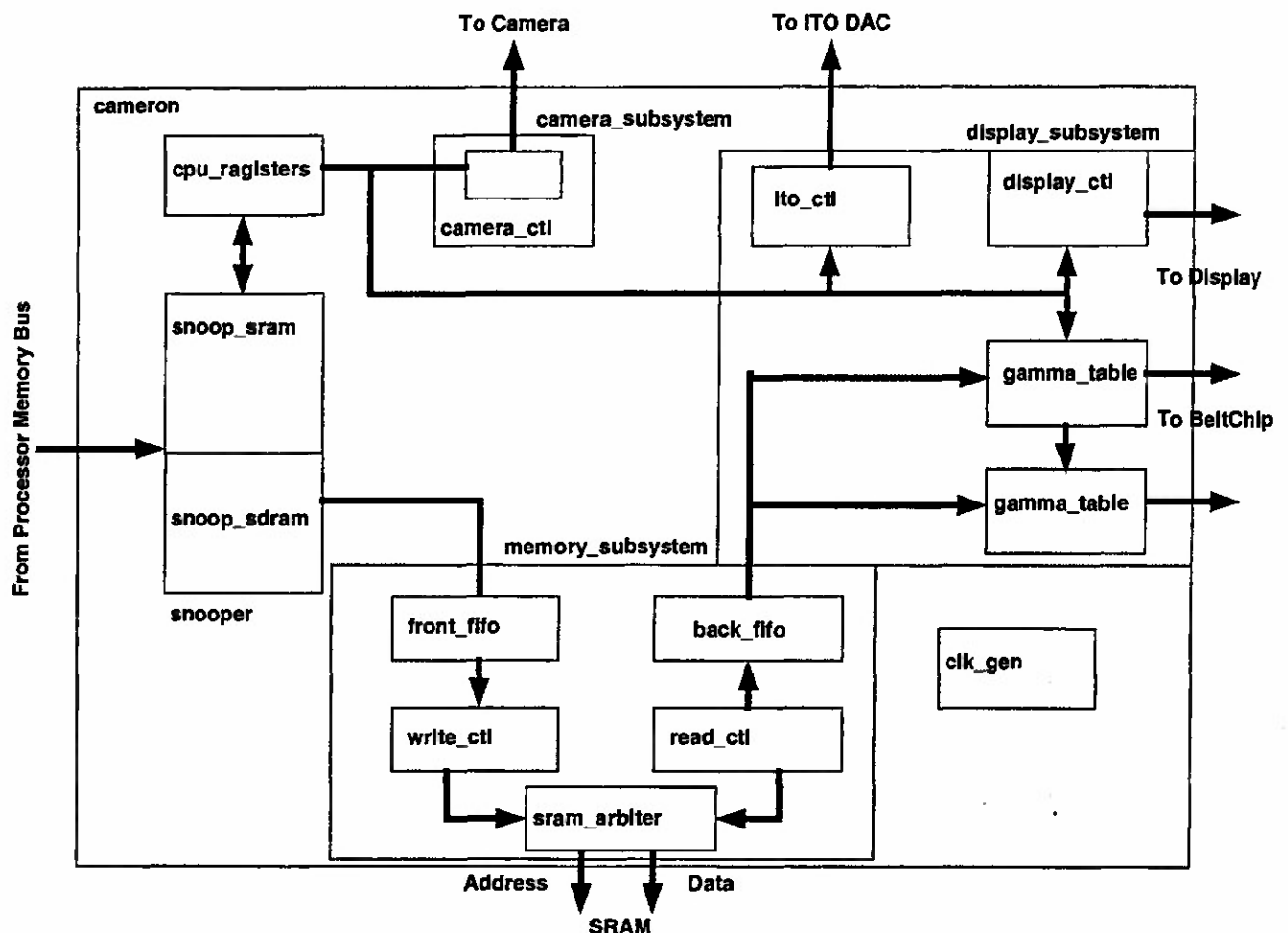


Figure 26. MC-2 Internal Block Diagram

7.7.2.2 MC-2 Module Descriptions

The resulting FPGA code consists of 6 major functional blocks and 14 individual modules. The architecture of the code is provided in Figure 26. The primary function of the FPGA is to monitor the processor bus for transfer of video data to the SDRAM memory, to place this data temporarily into a frame buffer, then to write that video data to the microdisplay. Additional functions includes the control of the ITO voltage provided of the microdisplay; control of brightness, contrast, and gamma correction parameters of the microdisplay; the control of the camera; and the transfer of camera video data to the processor. The following sections describe the functionality of each of the VHDL blocks and modules that make up the FPGA code.

Referring to the block diagram, the FPGA code is composed of 6 functional blocks. From the standpoint of video data flow, the FPGA code is divided conceptually into front, middle, and back portions. The modules snoop_sdram and front_fifo are part of the front end and are clocked with SDCLK from the CPU. The modules front_fifo, write_ctl, sram_arbiter, read_ctl, and back_fifo are part of the middle portion and are clocked with MEMCLK, which is shared with

the SRAM. The back portion contains the modules back_fifo, gamma_table, and display_ctl are clocked with DISP_CLK, which is half the frequency of MEMCLK.

The snoop block contains two modules, snoop_sram and snoop_sdram. The module snoop_sdram, clocked off the SDCLK from the CPU, monitors the processor bus for video data transactions within the designated video data address space of the SDRAM. The base of the video address space is at 0xC040 0000 and it extends to address 0xC04E A5FF. When a write transaction to the SDRAM occurs, the module snoop_sdram acquires the data and its corresponding address. Bus transactions within the video data address space result in the generation of a write enable signal.

The data, address, and write enable signal are routed to the memory_subsystem block, containing the modules front_fifo, write_ctl, sram_arbiter, read_ctl, and back_fifo. The front_fifo module is simply a wrapper for an instance of the Virtex FIFO available from Xilinx. This is a fully asynchronous dual-port FIFO instantiated with a width of 40 bits and a depth of 511 samples. The input side of this FIFO is clocked using the SDCLK from the CPU, while the output side is clocked using MEMCLK, the same clock as is used by the SRAM. Video data and addresses from the snoop_sdram module are acquired by the module front_fifo during the write enable signal provided by that module. Data are read from front_fifo under the control of the read enable signal provided by the write_ctl module. Front_fifo also provides a 9-bit status word to write_ctl indicating the amount of data contained in it. This information is used by the write_ctl module to decide whether or not to request SRAM bandwidth.

The module write_ctl performs a number of functions. First, it reformats the address and data words provided by front_fifo so that the red, green, and blue portions of the video data word can be partitioned in the SRAM. This partitioning placing red data into the lower half of the first SRAM chip, green data into the upper half of the first SRAM chip, and blue data into the lower half of the second SRAM chip. This module relies on a separate state machine module, write_ctl_fsm_hack, to aid in performing this function. In addition, write_ctl generates SRAM read requests to the sram_arbiter module when the amount of data stored in front_fifo reaches a preset level, as indicated by its 9-bit status word. If sram_arbiter grants this request, write_ctl generates a burst of data/address pairs that are routed through sram_arbiter to the SRAM. This module also has a stale counter that will request SRAM bandwidth even though front_fifo is not filled to that preset level in the event the data has sat in front_fifo for a pre-determined period of time.

The module sram_arbiter is central to the efficient use of SRAM bandwidth. This module relies on the state machine module sram_arbiter_fsm to balance the competing requirements of the front and rear FIFOs. As mentioned previously, the module write_ctl provides a write request signal to sram_arbiter when front_fifo becomes nearly full. In addition, the module read_ctl provides a read request to sram_arbiter if back_fifo requires data. This read request can have one of two priorities. The lower priority is used when rear_fifo has room to accept video data from the SRAM. The higher priority is used when rear_fifo is in danger of becoming empty. Sram_arbiter relies on sram_arbiter_fsm to decide which of these competing requirements needs servicing at any given time. In addition, sram_arbiter multiplexes the data, address and control signals from the write_ctl and read_ctl modules, depending on the mode of operation. This

module also provides the necessary delays to the data, address words, and control signals to and from the SRAM so that latency and read/write turn around times can be minimized.

The module `read_ctl` is the next module in the video data path. This module generates SRAM read requests to the `sram_arbiter` module when the amount of data stored in `back_fifo` reaches one of two preset levels, as indicated by its 9-bit status word. These read requests can be one of two priorities, depending on the amount of data contained in `back_fifo`. If `sram_arbiter` grants this request, `read_ctl` generates a burst of sequential addresses that are routed through `sram_arbiter` to the SRAM. This module then receives 32-bit video data words, representing four pixellets, from that frame buffer through `sram_arbiter`. The video data are read out in bursts and by color. That is, `read_ctl` generates sequential addresses that correspond to one of the three colors of data and does so until all the data of that color have been read out of SRAM. The starting address of a color generated by `read_ctl` is determined by a color signal and a `start_new_field` signal from the `display_ctl` module. This technique is used in support of the field sequential color approach. This module relies on a separate state machine module, `read_ctl_fsm_hack`, to aid in the generation of the control signals to the SRAM and the module `back_fifo`.

The `back_fifo` module is simply a wrapper for an instance of the Virtex FIFO available from Xilinx. This is a fully asynchronous dual-port FIFO instantiated with a width of 32 bits and a depth of 511 samples. The input side of this FIFO is clocked using the `MEMCLK`, the same clock that is provided to the SRAM. The output side is clocked using `DISP_CLK`, the same clock used to drive the display. Video data from the `read_ctl` module are acquired by `back_fifo` during the write enable signal provided by the `read_ctl` module. Data are read from `back_fifo` under the control of the read enable signal provided by the `display_ctl` module. `Back_fifo` also provides a 9-bit status word to `read_ctl` indicating the amount of data contained in it. This information is used by the `read_ctl` module to decide whether or not to request SRAM bandwidth.

The video data are then routed to the `display_subsystem` block, containing the modules `ito_ctl` and `display_ctl` and two instances of the module `gamma_table`, which is the last module in the video data path. As mentioned previously, video data is presented by the SRAM as 32-bit words, representing 4 8-bit pixellets. These are ultimately presented by the FPGA to the BeltChip as four simultaneous video channels, representing row-adjacent pixellets, which is the presentation the display expects as well. Each instance of `gamma_table` contains an instance of a 512x8 dual-port SRAM, with each of the two ports dedicated to a video channel. Therefore, the first instance of `gamma_table` contains the LUTs for video channels A and C, while the second instance contains the LUTs for video channels B and D. Even though the video data is of 8-bit resolution, the actual resolution is no better than that supplied originally by the CPU, which is only up to 6 bits. Therefore, only 64 values need to be contained in the LUT for each video channel to meet the resolution requirements. So, of the 512 available bytes in each of the dual-port SRAMs used in `gamma_table`, only 128 bytes are used; 64 for each of the two channels it represents.

The function of `gamma_table` is fairly straightforward. The 8-bit video data is used as an address to access the corresponding LUT data. The module is configured to self-load upon power-up with a linear transfer function that inverts the input data. That is, video data values that are near 0x00 are translated into values close to 0xFF, and vice-versa for input video values near 0xFF. This

process adjusts the data to be more suitable for the BeltChip, with the result that video data presented by the CPU that is near 00 0000 represents black and values near 11 1111 represent white.

The module `display_ctl` is responsible for generating the sync and clock signals required by the microdisplay and for providing the control signals required by the module `read_ctl` to support the selection of the appropriate SRAM data. The module defines the duration of each of the three color fields, with each field divided into time slots for the writing of data to the display, for the settling of the LCD material, and for the illumination of by the individual LEDs. Row and column counters define the length of each row and the durations of each of these time slots. As a means of providing color balance control, the module supports the programming of the individual LED on-times for the red, green and blue LEDs. These on-times are set to a default value on power-up that is equal to 44% of the field duration.

The second module of the snoop block is called `snoop_sram`. This module monitors the processor bus for SRAM read and write transactions within the address space of CS1_N of the processor (0x0800 0000 to 0x1000 0000). When such a transaction occurs, the data, address, and type of transaction are acquired. The snooped data word, its address, a `read_write` signal, and a `snoopdone` signal are provided by `snoop_sram` to the `cpu_registers` module, where the transaction is analyzed.

The module `cpu_registers` sorts through the SRAM transactions snooped by `snoop_sram`. Like the `snoop_sram` module, `cpu_registers` is clocked using `SDCLK` from the CPU. If the transaction was a write, the address is examined and the data are stored in the appropriate register. In the event the transaction is for the purpose of updating the ITO voltage or for sending a command to the camera, an appropriate update signal is generated and sent to the module controlling that function. Since the `ito_ctl` and `camera_ctl` modules are clocked using `DISP_CLK`, the update signals are generated from a re-clocked version of the `snoopdone` signal. In the event a write transaction is intended for the video LUT, both the data and a portion of the address are both stored and a `LUT_wreq` signal is generated. As presently configured, a read transaction is ignored; there are no registers that can be read. A status register may be implemented at a later time that provides a measure of the amount of data contained in `front_fifo` to aid the CPU in throttling the video data writes, in the event this becomes necessary once CPU burst writes are implemented.

As mentioned previously, commands to be directed to the camera are sorted by the `cpu_registers` module by means of the address presented by the processor. This 16-bit command word is composed of an 8-bit register destination address and an 8-bit data value. This 16-bit word, along with the `OVT_update` signal, is routed from the `cpu_registers` module to the `camera_ctl` module within the `camera_subsystem` block. `Camera_ctl` reformats the 16-bit parallel word into an I²C serial data stream composed of an 8-bit header and the 8-bit register and 8-bit data word. `Camera_ctl` relies on two state machine modules, `camera_scl` and `camera_ctl_fsm_hack`. The function of `camera_scl` is to generate a 260 KHz timing interval that is active for the duration of a go signal provided to it by `camera_ctl`. This module outputs two timing signals at a 260 KHz rate, `scl_high` and `scl_mid_low`. `Scl_high` is high for half the duration of the timing interval, while `scl_mid_low` defines the transition points for the serial data stream. The module `camera_ctl_fsm_hack` defines the present activity of the I²C serial port, with the states being Idle,

Start, Xmit, Ack, and Stop. The module defines the sequence of events on the I²C port, from Idle, to Start through the transmission of the 3 bytes with an acknowledge event after each byte, to Stop and back to Idle. Camera_ctl ties these two state machines together and interfaces them to the rest of the code.

The ITO voltage is controlled by a 12-bit DAC over a QSPI-compatible three-wire interface, providing an output voltage range of 0.0 to 5.0 V in 1.22mV steps. Upon power-up, the module generates a QSPI command to the ITO DAC placing the ITO voltage at its mid-scale value (= 2.5V). When the module receives an ITO_update command, the 12-bit data work on the ITO_value_in bus is latched into a shift register and the state machine is started. The state machine cycles through the states Idle, Start, SetData, and SetClk, providing a clock of frequency 7 MHz and a serial data stream from the shift register such that the clock changes from high to low while the data bit is stable.

The module clk_gen contains the clocking and reset circuits and provides system reset, SDCLK, MEM_CLK, and DISP_CLK to the remaining modules of the FPGA. The system reset signal is formed from the logical OR of the input reset signal, the DLL lock signal, and a reset signal that is generated by a counter circuit for a short period of time after the 1x clock from the DLL has started. SDCLK is formed from SDCLK1_IN from the CPU with the use of a global clock buffer, while DISP_CLK is the 1x clock output from the DLL, which accepts SYS_CLK_IN from the external crystal at a frequency of 50 MHz. The clock MEM_CLK is taken from the 2x output of the DLL, providing the 100 MHz clock for the memory.

7.7.2.3 FPGA Hardware Implementation

The FPGA used in the WebPhone is a Xilinx XCV300FPGA456AFP. This device has a core voltage of 2.5 V and supports any one of several logic levels to interface to the device in a feature that Xilinx refers to as SelectI/O. In the WebPhone application, the FPGA I/O standard is LVTTL, providing an output source voltage of 3.3 V and requiring a 3.3 V I/O supply. A 2.5 V regulator is located on the Main Board to supply the FPGA core voltage and derives the required voltage directly from the battery voltage.

This Xilinx FPGA provides the user with a large number of programming methods to choose from. The method selected for use in the WebPhone design is called "SelectMAP".. In this method, FPGA bit data are written into the FPGA 8-bits at a time by the StrongArm CPU at an address of 0x1000 0000 and at a frequency of 667 KHz (or a period of 1.5usec). The Xilinx configuration file turned out to be approximately 220K bytes, resulting in a load duration of 0.3 seconds.

7.7.2.4 Snooping Scheme Design Issues

The random-access write nature of the frame buffer store creates new challenges when combined with the color-sequential nature of the microdisplay. Unlike a conventional video input source, where the entire image is replaced, only one pixel may be updated at a time. Depending on the memory organization, the memory requires either a read-modify-write capability, or a byte-enable (or byte-protect) capability.

There are a number of memory snooping scheme options. Each trades off processor performance for power consumption. Three options were explored in detail.

1. **No color separation.** In this high-performance, high-power configuration, each 16-bit RGB pixel is stored as a half-word in memory. The write transfers are relatively efficient, requiring only one 32-bit write operation per 16-bit pixel. On the other hand, during a read transfer, two 32-bit read operations are required to obtain the 4-pixel quad that is delivered to the display per cycle. At a snoop bandwidth of 40 MHz, and display output frequency of 40 MHz, the overall required memory bandwidth is 120 MHz. As the SRAM is rated to 133 MHz, this is feasible. This scheme also supports the original 32-bit bus width of the SX-1 design, since the snooped word can be written to memory in a single cycle. The disadvantage of this scheme is high power consumption of the 120 MHz bus traffic and I/O circuitry. Even with an static video display, which is the most frequent mode of operation, the shadow frame buffer is running at 80 MHz.
2. **Color separation, unpacked.** In this scheme, the 16-bit pixels are separated by color, so that 3 separate writes are required for every snooped pixel. The net effect is to triple the snoop (write) bandwidth. On the other hand, each read operation retrieves four pixels from the memory. This method requires 3 x 40 MHz on the read side but only 40 MHz on the write side, for a total of 160 MHz peak bandwidth. This is not sustainable by the 133 MHz SRAM and therefore the processor may need to be stalled in the case of peak burst transfers. On the other hand, with a static video display, the shadow frame buffer is running at 40 MHz. The other disadvantage of this scheme is that it requires 50% more memory, since 24 bits instead of 16 are required per pixel.
3. **Color separation, packed.** In this scheme, the 6-bit color components are packed into the 36-bit word of the memory. This scheme has the best static video power performance, since the frame buffer needs to be accessed only at 66% of 40 MHz, or 26.7 MHz. The problem is with the snoop bandwidth, for any one snooped pixel, three words have to read, modified, and written, resulting in a net bandwidth requirement of 240 MHz on the input side and 26.7 MHz on the output side. Clearly, peak video updates are not sustainable.

The second scheme was selected as the best compromise of power and update speed. The processor stalls are very unlikely unless video is being computed from internal registers. For bit blit operations or for image decompression, memory reads are required. As soon as reads occur on the snooped memory bus, the FIFOs have enough time to be written to the shadow memory.

7.7.3 Programming Model

This FPGA provides the programmable memory, registers, and control ports seen by the system software and used to implement the display subsystem. A full description of this programming model is given in the supplementary documentation, attached.

This document reports research undertaken at the U.S. Army Research, Development and Engineering Command, Natick Soldier Center, Natick, MA, and has been assigned No. NATICK/TR-04/1010 in a series of reports approved for publication.

8. References

1. S. Seneff, P. Schmid, E. Hurley, R. Lau, C. Pao, and V. Zue, "GALAXY-II: A Reference Architecture for Conversational System Development," *Proc. ICSLP '98*, pp. 931-934, Sydney, Australia.
2. D. Goddeau, E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue, "GALAXY: A Human Language Interface to Online Travel Information," *Proc. ICSLP '94*, pp. 707—710, Yokohama, Japan, Sept. 1994.
3. H. Meng, S. Busayapongchai, J. Glass, D. Goddeau, L. Hetherington, E. Hurley, C. Pao, J. Polifroni, S. Seneff, and V. Zue, "WHEELS: A Conversational System in the Automobile Classifieds Domain," *Proc. ICSLP '96*, Philadelphia, PA, pp. 542-545, Oct. 1996.
4. S. Seneff and J. Polifroni, "A New Restaurant Guide Conversational System: Issues in Rapid Prototyping for Specialized Domains," *Proc. ICSLP '96*, pp. 665-668, Philadelphia, PA, Oct. 1996.
5. V. Zue, S. Seneff, J. Glass, L. Hetherington, E. Hurley, H. Meng, C. Pao, J. Polifroni, R. Schloming, and P. Schmid, "From Interface to Content: Translingual Access and Delivery of On-Line Information," *Proc. Eurospeech '97*, pp. 2227—2230, Rhodes, Greece, Sept. 1997.
6. C. Wang, J. Glass, H. Meng, J. Polifroni, S. Seneff, and V. Zue, "YINHE: A Mandarin Chinese Version of the GALAXY System," *Proc. Eurospeech '97*, pp. 351—354, Rhodes, Greece, Sept. 1997.
7. R. Lau, G. Flammia, C. Pao, and V. Zue, "WEBGALAXY - Integrating Spoken Language and Hypertext Navigation," *Proc. Eurospeech '97*, pp. 883—886, Rhodes, Greece, Sept. 1997.
8. S. Seneff, "TINA: A Natural Language System for Spoken Language Applications," *Computational Linguistics*, Vol. 18, No. 1, pp. 61—86, 1992.
9. S. Seneff, D. Goddeau, C. Pao, and J. Polifroni, "Multimodal Discourse Modelling in a Multi-User Multi-Domain Environment," *Proc. ICSLP '96*, pp. 192—195, Philadelphia, PA, Oct. 1996.
10. J. Glass, J. Polifroni and S. Seneff, "Multilingual Language Generation Across Multiple Domains," *Proc. ICSLP '94*, pp. 983—986, Yokohama, Japan, Sept. 1994.
11. J. R. Yi and J. R. Glass, "Natural-sounding Speech Synthesis using Variable-length Units," *Proc. ICSLP '98*, pp. 1167-1170 Sydney, Australia, Nov. 1998.
12. F. Pereira and M. Riley, "Speech recognition by composition of weighted finite automata," in *Finite-State Language Processing*, E. Roche and Y. Schabes, Eds., pp. 431-453. MIT Press, Cambridge, MA, 1997.
13. M. Mohri, M. Riley, D. Hindle, A. Ljolje, and F. Pereira, "Full expansion of context-dependent networks in large vocabulary speech recognition," in *Proc. ICASSP*, Seattle, WA, 1998, pp. 665-668.

14. J. Glass, T.J. Hazen, and L. Hetherington, "Real-time telephone-based speech recognition in the Jupiter domain," in *Proc. ICASSP*, Phoenix, AZ, 1999, pp. 61-64.
15. L. Baptist and S. Seneff, "Genesis-II: A Versatile System for Language Generation in Conversational System Applications," *Proc. ICSLP '00*, Vol. III, pp. 271--274, Beijing, China, Oct. 2000.
16. V. Zue, "Toward Systems that Understand Spoken Language," *IEEE Expert*, pp. 51-59, February, 1994.
17. J. Glass, G. Flammia, D. Goodine, M. Phillips, J. Polifroni, S. Sakai, S. Seneff, and V. Zue, "Multilingual Spoken-Language Understanding in the MIT Voyager System," *Speech Communication*, Vol. 17, No. 1, pp. 1-18, March 1995.
18. A. Snoeren, "Adaptive Inverse Multiplexing for Wide-Area Wireless Networking," *Proc. IEEE GlobeCom*, Rio de Janeiro, Brazil, December 1999.
19. M. Rahnema. "An Overview of the GSM System and Protocol Architecture." *IEEE Communications Magazine*, April 1993.
20. Electronic Industry Alliance/Telecommunications Industry Association. "IS-95: Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System," 1993.
21. Wireless Data Forum. "Cellular Digital Packet Data System Specification, Release 1.1," January 1995.
22. H. Adishesu, G. Parulkar, and G. Varghese. "A Reliable and Scalable Striping Protocol." *Proc. ACM SIGCOMM*, August 1996
23. Bandwidth On Demand Interoperability Group. "Interoperability Requirements for Nx56/64 kbit/s Calls." September 1992.
24. C. B. Traw and J. Smith. "Striping within the Network Subsystem." *IEEE Network*, 1995.
25. B. Bakshi, P. Krishna, N. Vaidya, and D. K. Pradhan. "Improving Performance of TCP over Wireless Networks." *Proc. ICDCS*, May 1997.
26. H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. "Improving TCP/IP Performance over Wireless Networks." *Proc. ACM MOBICOM*, November 1995.
27. H. Balakrishnan, V. Padmanabhan, and R. Katz. "The Effects of Asymmetry on TCP Performance." *Proc. ACM/IEEE MOBICOM*, September 1997.
28. M. Afegan and J. Wroclawski, "A Framework for Content Management in a Wireless Environment", ANA Research Group Technical Report, September 2000.
29. M. Afegan, Intelligent Management of Content by Application Layer Protocols in a Mobile Environment, BA/MA Thesis, Harvard College Department of Computer Science, May 2000.

30. S. Seneff, R. Lau, and J. Polifroni, "Organization, Communication, and Control in the GALAXY-II Conversational System," *Proc. EUROSPEECH 99*, Budapest, Hungary, September, 1999.
31. A. Bakre and B. Badrinath. "I-TCP: Indirect TCP for Mobile Hosts." *Proc. ICDCS*, May 1995.
32. A. Demers, S. Keshav, and S. Shenker. "Analysis of a Fair Queuing Algorithm," *Journal of Internetworking Research and Experience*, September 1989.
33. R. Durst, G. Miller, and E. Travis. TCP Extensions for Space Communications. In *Proc. ACM/IEEE MOBICOM*, September 1996.
34. M. Mehta and N. Vaidya. "Delayed Duplicate-Acknowledgements: A Proposal to Improve Performance of TCP on Wireless Links." Technical Report, Computer Science Dept., Texas A&M University, February 1999.
35. V. Paxson. End-to-End Internet Packet Dynamics. In *Proc. ACM SIGCOMM*, September 1997.
36. P. Ssinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks." In *Proc. ACM/IEEE MOBICOM*, August 1999.

PAGE INTENTIONALLY LEFT BLANK

APPENDIX A
ADDITIONAL TECHNICAL DOCUMENTATION

APPENDIX A

Additional Technical Documentation (included with report)

M. Bolotski and D. Elmore, "WP-3 FPGA Programmers Guide V2.0", MicroDisplay Corp. Describes the hardware programming interface of the WP3 display controller hardware.

M. Bolotski and D. Elmore, "WP3 FPGA Design V2.0", MicroDisplay Corp. Detailed design description of the WP3 digital controller FPGA.

SX2 Schematic Package, MicroDisplay Corp and MIT LCS ANA Group; primary design by Andrew "Bunnie" Huang. Schematics for the PCB assemblies used in the WP3 handheld device.

APPENDIX B
PAPERS

APPENDIX B

Papers (copies available upon request)

M. Afergan and J. Wroclawski, "A Framework for Content Management in a Wireless Environment", ANA Research Group Technical Report, September 2000.

S. Kamppari and T.J. Hazen, "Word and Phone Level Acoustic Confidence Scoring," *Proc. ICASSP2000*, Istanbul, Turkey, June 2000.

E. Sandness and I.L. Hetherington, "Keyword-based Discriminative Training of Acoustic Models," *Proc. 6th International Conference on Spoken Language Processing*, Beijing, China October 2000.

A. Snoeren, "Adaptive Inverse Multiplexing for Wide-Area Wireless Networking," *Proc. IEEE GlobeCom*, Rio de Janeiro, Brazil, December 1999.

J. Yi, J. Glass and L. Hetherington, "A Flexible, Scalable Finite-State Transducer Architecture for Corpus-Based Concatenative Speech Synthesis," *Proc. 6th International Conference on Spoken Language Processing*, Beijing, China October 2000.

APPENDIX C
THESES

APPENDIX C

Theses (copies available upon request)

M. Afergan, Intelligent Management of Content by Application Layer Protocols in a Mobile Environment, BA/MA Thesis, Harvard College Department of Computer Science, May 2000.

S. Bauer, Capturing Application and Network Adaptivity: Time Variations and Adaptation Paths. M.S. Thesis, MIT Department of Electrical Engineering and Computer Science, February 2001.

T. Burianek, Building a Speech Understanding System Using Word Spotting Techniques, M.Eng. Thesis, MIT Department of Electrical Engineering and Computer Science, July 2000.

S. Kamppari, Word and Phone Level Acoustic Confidence Scoring for Speech Understanding Systems, M.Eng. Thesis, MIT Department of Electrical Engineering and Computer Science, September 1999.