TABLE OF CONTENTS

EXECUTIVE SUMMARY ................................................................................................................................. 1

1. INTRODUCTION ........................................................................................................................................... 2
   1.1. ACRONYM REFERENCE .......................................................................................................................... 3
   1.2. PROBLEM SPECIFICATION .................................................................................................................... 5
   1.3. PROJECT SELECTION ............................................................................................................................... 5
   1.4. CHALLENGE ......................................................................................................................................... 6
   1.5. TEAM MEMBERS .................................................................................................................................. 6

2. PROJECT REQUIREMENTS .............................................................................................................................. 8

3. PROJECT MANAGEMENT .............................................................................................................................. 8
   3.1. TEAM ORGANIZATION ........................................................................................................................... 8
   3.2. WORK BREAKDOWN STRUCTURE ......................................................................................................... 11
       3.2.1. Administration .................................................................................................................................. 12
       3.2.2. Business ......................................................................................................................................... 12
       3.2.3. Technical ....................................................................................................................................... 12
       3.2.4. Hours Spent ................................................................................................................................... 13
   3.3. SCHEDULE .......................................................................................................................................... 15
       3.3.1. Spring Semester .............................................................................................................................. 15
       3.3.2. Fall Semester .................................................................................................................................. 18
   3.4. CONTINGENCY ..................................................................................................................................... 22
   3.5. PROJECT BUDGET ................................................................................................................................. 22

4. PRELIMINARY RESEARCH ............................................................................................................................. 23
   4.1. PRODUCTS ............................................................................................................................................ 23
       4.1.1. Ultimate Wireless by GrandTec ........................................................................................................ 23
       4.1.2. PC on TV Player by D-Link .............................................................................................................. 24
       4.1.3. SlingCatcher by Sling Media .......................................................................................................... 24
       4.1.4. SRU4000 by Phillips ....................................................................................................................... 25
       4.1.5. Pro PC/TV Wireless by Altech ........................................................................................................ 25
   4.2. PATENTS .............................................................................................................................................. 26

5. PROPOSED SOLUTION ................................................................................................................................. 29

6. PRODUCT DESIGN ...................................................................................................................................... 34
6.1. SYSTEM DESCRIPTION

6.2. NORMS

6.3. CRITERIA

6.4. ALTERNATIVES AND ANALYSIS
   6.4.1. Wireless Protocol
   6.4.2. Development Platform
   6.4.3. Video Compression Algorithm

6.5. SYSTEM ARCHITECTURE AND DESIGN

7. PROTOTYPE DESIGN
   7.1. ALTERNATIVES AND ANALYSIS: PROTOTYPE
      7.1.1. Wireless Protocol
      7.1.2. Wireless Development Kit
      7.1.3. VideoLan Client (VLC)
      7.1.4. Video Compression Algorithm
      7.1.5. Video Processing Development Kit
   7.2. SYSTEM ARCHITECTURE AND DESIGN SUMMARY

8. IMPLEMENTATION
   8.1. WIRELESS SEND
   8.2. WIRELESS RECEIVE
   8.3. DIGI–TO–TI TRANSFER PROTOCOL
   8.4. VIDEO DECODING AND ENCODING

9. RESULTS
   9.1. TESTING
   9.2. DESIGN ANALYSIS

10. BUSINESS AND PRODUCTION
   10.1. BUSINESS PLAN
   10.2. COMPETITION RESULTS

11. MANUFACTURING PLAN
   11.1. PRODUCTION
   11.2. PRODUCT
   11.3. PRODUCT SUPPORT AND MARKETING

12. CONCLUSION
TABLE OF FIGURES

13. ACKNOWLEDGEMENTS

14. BIBLIOGRAPHY

15. APPENDIX

15.1. NCIIA Grant Proposal

15.2. Market Research Survey

15.3. Email from Staccato Communications

15.4. Wireless Receive Code

15.5. Video Processing Code

15.6. Senior Design Night Demonstration Perl Code

TABLE OF TABLES

TABLE 1-1 ACRONYMS

TABLE 3-1 Team Roles

TABLE 3-2 Project Milestones

TABLE 3-3 Contingency Plans

TABLE 3-4 Overview of Volturna’s Budget

TABLE 5-1 Competing Product Comparison

TABLE 7-1 Wireless Protocol Comparison

TABLE 7-2 MPEG-2 Operation Levels (6)

TABLE 7-3 Wireless Protocol Decision Matrix

TABLE 7-4 Video Compression Algorithm Decision

TABLE 7-5 DSP Decision Matrix

TABLE 8-1 DC7 Pin Header Information

TABLE 11-1 Packaging Costs for Product

TABLE 11-2 Product Bill of Materials

TABLE OF FIGURES

FIGURE 1-1 Wireless Media Transceiver Concept Design

FIGURE 3-1 Weekly Team Hours

FIGURE 3-2 Team Member Contribution to Weekly Hours

FIGURE 3-3 Spring Semester Administrative Schedule

FIGURE 3-4 Spring Semester Documentation Schedule
FIGURE 3-5 SPRING SEMESTER TECHNICAL SCHEDULE

FIGURE 3-6 FALL SEMESTER ADMINISTRATIVE SCHEDULE

FIGURE 3-7 FALL SEMESTER WEEKLY TASK SCHEDULE

FIGURE 3-8 FALL SEMESTER REPORTING SCHEDULE

FIGURE 3-9 FALL SEMESTER BUSINESS SCHEDULE

FIGURE 4-1 GRANDTEC ULTIMATE WIRELESS MODULES(1)

FIGURE 4-2 D-LINK DPG 1200(2)

FIGURE 4-3 SLING MEDIA'S SLING CATCHER - CONNECTION PANEL(3)

FIGURE 4-4 PHILLIPS SRU4000(4)

FIGURE 4-5 ALTECH Pro PC/TV WIRELESS(5)

FIGURE 6-1 HIGH-LEVEL BLOCK DIAGRAM

FIGURE 6-2 VIDEO PLAYBACK BITRATES (7)

FIGURE 6-3 FRONT AND BACK OF RECEIVER PACKAGE

FIGURE 7-1 TEXAS INSTRUMENT'S TMDXEVM355 - DM355 EVALUATION MODULE(12)

FIGURE 8-1 OPENING THE VLC PLAYER IN WINDOWS XP

FIGURE 8-2 OPENING THE "FILE" DROP-DOWN MENU AND SELECTING THE "OPEN NETWORK STREAM" OPTION

FIGURE 8-3 INSERTING THE CODE INTO THE "CUSTOMIZE" TAB IN THE VLC PLAYER

FIGURE 8-4 TOP VIEW OF W1-9C DEVELOPMENT BOARD(13)

FIGURE 8-5 TOP VIEW OF CONNECTCORE W1-9C MODULE(13)

FIGURE 8-6 INTERNET LAYER (GREEN), TRANSPORT LAYER (ORANGE), AND APPLICATION LAYER (BLUE) OF A UDP DATAGRAM

FIGURE 8-7 NETGEAR WGR614 WIRELESS ROUTER(14)

FIGURE 8-8 BOARD INTERCONNECT DESCRIPTION

FIGURE 8-9 DEVELOPMENT ENVIRONMENT OVERVIEW

FIGURE 8-10 DM355 INTERFACES(12)

FIGURE 8-11 DM355 AUDIO/VIDEO BUFFER STRUCTURE

FIGURE 8-12 DM355 SOFTWARE THREAD(16)

FIGURE 8-13 DM355 FRAME STACK(16)

FIGURE 9-1 SERVERTALK UDP CHAT PROGRAM

FIGURE 9-2 HERCULES UDP CHATTING PROGRAM

FIGURE 11-1 RECEIVER CASING

FIGURE 11-2 USB DONGLE PCB SCHEMATIC

FIGURE 11-3 RECEIVER PCB SCHEMATIC
EXECUTIVE SUMMARY

This report will provide the reader with an overview and justification of Team Volturna’s design decisions made in the research and development of the Volturna Wireless Media Player (VWMP). This will include sections covering the team’s selection of this project, identification of a problem, planning the most efficient and economically feasible solution, the design process, and the building and testing of the VWMP solution.

Team Volturna was formed with the entrepreneurial goal of developing a marketable product that would not only meet the requirements of Calvin College’s Senior Design course but be useful to the end customer. The inspiration for the VWMP came from observing the growing media trends such as YouTube and television (TV) stations making their shows available for online viewing. These trends show a higher number of consumers watching videos on personal computers (PCs) which typically have small screens designed for individual viewing. Research conducted by the group showed that there was a significant population of consumers who watched PC videos on a TV or had the desire to do so using the combination of an S-Video cable and an audio cable. This process involved the inconvenience running wires across a room typically filled with the obstacles of furniture and took away from the aesthetic nature of the room.

The VWMP is a device which enables users to stream audio and video from a PC to a TV, projector, or any device with RCA audio-video input ports. The VWMP consists of a single receiver unit which receives an input signal of audio and video broadcast over an 802.11g network, transforms that signal to a TV compatible format, and outputs the signal through RCA Audio/Video (RCA A/V) ports. This device will eliminate the clutter of running S-Video and line-in audio cables from a PC to a TV or similar display unit and allow customers to more easily share their digital media in group settings.
1. INTRODUCTION

Within the timeframe of September to May, Calvin College Engineering engages each of their senior level students in an extensive, multi-semester design project. The course, Engineering 339 and Engineering 340, or more commonly known as simply, “senior design,” allows the students to form teams and capstone projects. These projects are significant and reflect the students’ engineering concentrations. This project report explores the endeavors and final design of which one group within the 2007 - 2008 school year chose. The team, known as Team Volturna (also as Team 7), comprises of Gwendolyn Einfeld, Richard Kwakye, Charlie Reitsma, and Ted Worst. Each is a graduating senior engineering student with an electrical and computing concentration at Calvin College. Additional assistance in the business aspects of the project was enlisted by Leah Einfeld, an economics major, also at Calvin College. The project is known as the Volturna Wireless Media Player (VWMP).

Team Volturna was formed with the entrepreneurial goal of developing a marketable product that would not only meet the requirements of senior design, but be useful to the end customer. The inspiration for the VWMP came from observing the growing media trends such as YouTube and television (TV) stations making their shows available for online viewing. These trends show a higher number of consumers watching videos on personal computers (PCs) which typically have small screens designed for individual viewing.

The VWMP is a device which enables users to stream audio and video from a PC to a TV, projector, or any device with RCA audio-video input ports. The VWMP consists of a single receiver unit which receives an input signal of audio and video broadcast over an 802.11g network, transforms that signal to a TV-compatible format. This device will eliminate the clutter of running S-Video and line-in audio cables from a personal computer to a television, or similar display unit, and allow customers the ease of sharing their digital media in group settings.
# 1.1. Acronym Reference

### Table 1-1 Acronyms

<table>
<thead>
<tr>
<th>ACRONYM</th>
<th>TERM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC</td>
<td>Advanced Audio Coding</td>
<td>Audio compression format specified by MPEG-2 and 4</td>
</tr>
<tr>
<td>ADI</td>
<td>Analog Devices, Inc.</td>
<td>Semiconductor manufacturer</td>
</tr>
<tr>
<td>ARM9</td>
<td>Advanced RISC Machine</td>
<td>Microprocessor architecture used in the Digi and TI development kits</td>
</tr>
<tr>
<td>BOM</td>
<td>Bill of Materials</td>
<td>A parts listing of necessary components involved in a marketable product</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial, Off-the-Shelf</td>
<td>Readily available software and hardware</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
<td>Or processor; a collection of logic units able to execute computer programs</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital-to-Analog Converter</td>
<td>Device that interprets digital information as an analog signal</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
<td>Protocol used by network clients to obtain parameters for an IP network</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
<td>Microprocessor for digital signals (often media)</td>
</tr>
<tr>
<td>DVD</td>
<td>Digital Video Disc</td>
<td>Optical digital media storage device</td>
</tr>
<tr>
<td>EVM</td>
<td>Evaluation Module</td>
<td>Development kit hardware</td>
</tr>
<tr>
<td>FM</td>
<td>Frequency Modulation</td>
<td>Broadcasting technique emphasizing the varying of frequencies</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
<td>Network protocol for file transfer</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input &amp; Output</td>
<td>Hardware ports for general applications</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public License</td>
<td>Free public license for use in the GNU open source project</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
<td>In reference to audio and video signals, defined by higher resolution and quality than standards</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
<td>Network protocol for information transfer</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual Property or Internet Protocol</td>
<td>Legal property rights or an address associated with a computer on a network</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
<td>Computer network covering a small geographic area</td>
</tr>
<tr>
<td>MPEG</td>
<td>Motion Picture Experts Group</td>
<td>Family of video compression algorithms</td>
</tr>
<tr>
<td>ACRONYM</td>
<td>TERM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
<td>Protocol allowing consistent file sharing over a network</td>
</tr>
<tr>
<td>NTSC</td>
<td>National Television System Committee</td>
<td>Analog audio/video encoding and broadcast format used in the Americas</td>
</tr>
<tr>
<td>NXP</td>
<td>Next eXPerience</td>
<td>Semiconductor manufacturer</td>
</tr>
<tr>
<td>PAL</td>
<td>Phase Alternating Line</td>
<td>Analog audio/video encoding and broadcast format used in Europe and Asia</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
<td>Any computer with microprocessing ability, suited for single-user interaction</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
<td>A silicon (generally) board with wires printed on it, for electronics</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Lock Loop</td>
<td>Control system that generates a fixed relation to the phase of a reference</td>
</tr>
<tr>
<td>POSIX</td>
<td>Portable Operating System Interface</td>
<td>IEEE 1003 standard for interfacing with UNIX based operating systems</td>
</tr>
<tr>
<td>PPFS</td>
<td>Project Proposal and Feasibility Study</td>
<td>Report submitted by team in December 2007</td>
</tr>
<tr>
<td>RCA, RCA A/V</td>
<td>Radio Corporation of America (Audio/Video)</td>
<td>Standard component wires for audio/video (left audio, right audio, video)</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
<td>Range of oscillation rates anywhere between 3 Hz and 300 GHz</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computing</td>
<td>CPU design strategy emphasizing instruction set performance</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
<td>Core internet protocol providing reliable and in-order data delivery</td>
</tr>
<tr>
<td>TI</td>
<td>Texas Instruments</td>
<td>Electronics manufacturer</td>
</tr>
<tr>
<td>TS</td>
<td>Transport Stream</td>
<td>An MPEG stream type used for streaming MPEG-4 media</td>
</tr>
<tr>
<td>TV</td>
<td>Television</td>
<td>Audio and video viewing device</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
<td>Core internet protocol without handshaking</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
<td>Computer serial bus for interfacing with peripherals</td>
</tr>
<tr>
<td>VGA</td>
<td>Video Graphics Array</td>
<td>Analog computer graphics standard</td>
</tr>
<tr>
<td>VLC</td>
<td>Video LAN Client</td>
<td>Open source media player</td>
</tr>
<tr>
<td>VWMP</td>
<td>Volturna Wireless Media Player</td>
<td>Product described in this report</td>
</tr>
<tr>
<td>WMV</td>
<td>Windows Media Video</td>
<td>Video codec by Microsoft, specific to Windows Media Player</td>
</tr>
</tbody>
</table>
ACRONYM | TERM | DEFINITION
---|---|---
WUSB | Wireless Universal Serial Bus | Short-range, high bandwidth wireless protocol
X86 | N / A | Common CPU instruction set architecture

1.2. **Problem Specification**

Team Volturna’s senior design project is the research, design and prototype of a wireless, digital media system that will enable the streaming of digital audio and video signals from a computer to a display, such as a television or projector. The system consists of two major components, a transmitter and a receiver, with primary design work existing in the receiver unit. The transmitter prototype uses the standard 802.11b wireless protocols to transfer data. This protocol is compatible with the 802.11g protocol common on laptops today. The receiver is the primary design focus for the team. It takes a wireless digital audio and video signal and converts it to an output that can be read by a television display. The project essentially processes the digital media remotely; separate from the user’s PC. The popularity of YouTube, home videos, and picture slideshows demonstrate a movement towards storing media on computers rather than removable discs. Volturna’s product fills the need for a way to share computer stored media in a group setting. Preliminary market surveys reveal a diverse consumer base that spans genders, ages, and both the business and personal entertainment markets. Figure 1-1 shows the concept design of Volturna’s project, and how any user with simply general knowledge of a computer would be able to successfully achieve function out of the product.

![Figure 1-1 Wireless Media Transceiver Concept Design](image)

1.3. **Project Selection**

Team Volturna is especially interested in engineering entrepreneurship. One of the major components of this project was the development of a business plan, entry into three entrepreneurship competitions, and exploration of the requirements for starting a company. Prior to selecting a project, the team agreed that the end-product should be something that appealed to a wide-consumer base, and thus could be the first product of a new company. Numerous team brainstorming sessions led to the selection of a wireless media streamer. A subsequent
market survey demonstrated that the VWMP appealed to a wide variety of consumers and therefore would be a good starting product for an infant company. The full survey can be found in Appendix 15.2.

<table>
<thead>
<tr>
<th>Category</th>
<th>Responding “yes” (all respondents)</th>
<th>Interested in Purchasing</th>
<th>No Interest in Purchasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watch movies on PC</td>
<td>66.6%</td>
<td>100%</td>
<td>25%</td>
</tr>
<tr>
<td>Watch TV on PC</td>
<td>80%</td>
<td>100%</td>
<td>75%</td>
</tr>
<tr>
<td>Watch home videos on PC</td>
<td>88.9%</td>
<td>90%</td>
<td>60%</td>
</tr>
<tr>
<td>Use PC for presentations</td>
<td>90%</td>
<td>90%</td>
<td>90%</td>
</tr>
<tr>
<td>Use PC for slideshows</td>
<td>66.7%</td>
<td>90%</td>
<td>37.5%</td>
</tr>
<tr>
<td>Primary operating system</td>
<td>Windows XP (80%)</td>
<td>Windows XP (80%)</td>
<td>Windows XP (80%)</td>
</tr>
</tbody>
</table>
| Most influential consideration in purchasing | 1. Cost  
2. Operating system  
2. Operating system  
3. TV output | 1. Cost  
2. Operating system  
3. Size |

1.4. Challenge

This project presented team members with a number of challenges. The emphasis on entrepreneurship meant the team needed to explore the legal and practical aspects of establishing a viable company, as well as how to competitively produce and market a product. The technical part of the project required team members to learn network protocols and video encoding/decoding standards, as well as how to work with different microprocessors. It also challenged the team to work in the C programming language – a language that members were nominally acquainted with prior to the project. The final challenge the team had to overcome was that of completely planning a complex, nine-month project from concept to implementation and testing, while staying within a tight budget.

1.5. Team Members

Team Volturna is made up of five members – four senior engineers in the electrical and computer concentration, and one junior from the economics department. This section provides an introduction to each of the team members, including their area of oversight within the project. The team opted to have each member lead a component of the project, although all members contributed to all areas. Further explanation of the work done by each member is given in Section Error! Reference source not found.
The remainder of this report gives a complete discussion of the research and design work done by Team Volturna, as well as entrepreneurial endeavors undertaken by the team, and the team’s overall assessment of the project. Section Error! Reference source not found. explains the Senior Design course requirements that were the framework for the project. Section Error! Reference source not found. gives administrative information.
pertaining to the project – the work breakdown among team members, budget, contingency plans, and a project schedule for both the Fall 2007 and Spring 2008 semesters during which the project took place. This is followed by a brief introduction to similar products on the market (Section Error! Reference source not found.) and then a thorough discussion of Volturna’s design criteria for the VWMP in Section Error! Reference source not found.. The design sections begin

2. **PROJECT REQUIREMENTS**

This project was done as part of a requirement for the Calvin College Senior Design engineering course. Specific requirements included selecting a project of suitable complexity to warrant nine months’ work by a team of five people, and a number of presentations and reports. These deliverables appear in the scheduling part of this report.

Additionally, Volturna added an entrepreneurial emphasis to the course requirement. This focus added a number of requirements to the project, including the development of a feasible business plan with a three-year financial plan, a thorough exploration of intellectual property (IP) as far as the legal framework for patenting a design concept, and entry into entrepreneurial competitions.

3. **PROJECT MANAGEMENT**

3.1. **Team Organization**

Volturna has five members, each with different roles pertaining to the major three parts of the project: business, technical, and administration. These roles are described in
Table 3-1. With five team members and three major areas of concentration, the team decided to create “leads” in each area. Gwen and Ted were the leads for the administration area, Richard oversaw the technical area, and Charlie and Leah were responsible for the business area. Each member provided support in all areas, particularly the technical area, but the area lead was responsible for ensuring progress and coordinating the work of other team members in their area. All members are in the electrical engineering concentration except for Leah, who is an economics major. As such, her role on the team extended only over the business aspect of the project.
Table 3-1 Team Roles

<table>
<thead>
<tr>
<th>Member</th>
<th>Primary Role(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gwendolyn Einfeld</td>
<td>Documents/editing, scheduling, wireless transfer</td>
</tr>
<tr>
<td>Leah Einfeld</td>
<td>Business plan and development</td>
</tr>
<tr>
<td>Richard (Kwame) Kwakye</td>
<td>Overseeing project design, wireless transfer</td>
</tr>
<tr>
<td>Charlie Reitsma</td>
<td>Business plan and development, video decoding/encoding</td>
</tr>
<tr>
<td>Ted Worst</td>
<td>Team communication, video decoding/encoding</td>
</tr>
</tbody>
</table>
### 3.2. Work Breakdown Structure

Table 3-2 shows the status of current project milestones.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select and define project</td>
<td>September 2007</td>
<td>Complete</td>
</tr>
<tr>
<td>Create high level block diagram</td>
<td>September 2007</td>
<td>Complete</td>
</tr>
<tr>
<td>BizPlan 2007 Competition</td>
<td>November 2007</td>
<td>Complete</td>
</tr>
<tr>
<td>NCIIA Grant Proposal Submitted</td>
<td>December 2007</td>
<td>Complete</td>
</tr>
<tr>
<td>Project Proposal and Feasibility Study Submitted</td>
<td>December 2007</td>
<td>Complete</td>
</tr>
<tr>
<td>Purchase Wireless Development Kit</td>
<td>December 2007</td>
<td>Complete</td>
</tr>
<tr>
<td>Purchase Video Processing Development Kit</td>
<td>February 2008</td>
<td>Complete</td>
</tr>
<tr>
<td>Successful Video Encoding (NTSC)</td>
<td>March 2008</td>
<td>Complete</td>
</tr>
<tr>
<td>Successful Communication between PC and Wireless</td>
<td>March 2008</td>
<td>Complete</td>
</tr>
<tr>
<td>Successful audio streaming between transmitter and receiver</td>
<td>March 2008</td>
<td>Incomplete</td>
</tr>
<tr>
<td>Successful video streaming between transmitter and receiver</td>
<td>April 2008</td>
<td>Incomplete</td>
</tr>
<tr>
<td>Design review</td>
<td>May 2008</td>
<td>Incomplete</td>
</tr>
<tr>
<td>Senior Design Night and Presentation</td>
<td>May 2008</td>
<td>Complete</td>
</tr>
</tbody>
</table>
3.2.1. Administration

Administrative tasks included status reports, budgeting, scheduling, poster design, and documentation. Ted and Gwen were the primary point persons for this area. Ted prepared weekly status reports to be submitted to the team advisor. Ted was also in charge of poster design. Gwen was responsible for project scheduling, and compiling reports and other documentation. Both Ted and Gwen kept track of available funds and expenses. Although Gwen and Ted were leads for this area, Charlie and Richard were also involved. Charlie was responsible for updating the website regularly, as well as providing documentation pertaining to the business and technical aspects of the project. Richard was responsible for much of the technical documentation and was the primary meeting scribe. The administrative tasks were weighted equally over both the fall 2007 semester and spring 2008 semester.

3.2.2. Business

Business tasks included preparing a detailed business plan, preparing financial information for a startup company, and analyzing the legal requirements for both the company and the company products. Additionally, the business group oversaw the entries to the Calvin bizPlan 2007, Great Lakes Entrepreneurship Quest (GLEQ), and National Collegiate Innovators and Inventers Association (NCIIA) Grant competitions. Charlie was the contact person for all competitions. He also led the development of Volturna’s business plan and grant proposal. Leah provided advice on the financial planning for the company. Gwen, Richard, and Ted assisted in the writing of both the grant proposal and business plan, as well as participated in the bizPlan live competition. Gwen also developed a market survey and analyzed the data collected by the other team members. Business tasks took place primarily during fall 2007 as all competitions had fall semester deadlines. During the spring semester, business related tasks were limited to development of a manufacturing plan and review of the existing business plan.

3.2.3. Technical

Technical tasks included the research, design, implementation, and testing of the VWMP. Richard was the point person for this group. Because the technical design was so complex, the team felt it would be better for one person to oversee the work of all members to ensure that the design was headed in a good direction and that the work done by each member was useful to the final product. The fall 2007 semester was dedicated to the research and design of the prototype and product. During this time, Ted and Gwen worked on developing a plan for the video encoding and decoding, while Charlie and Richard worked on a wireless transmission solution. The team spent spring 2008 refining the VWMP design and building a prototype of the device. The teams shifted during this time as the video processor required a Linux operating system and Charlie was the only team member with Linux on his laptop. As a result, Gwen and Richard worked on the wireless implementation and the video decoding/encoding implementation was done by Ted and Charlie.
3.2.4. Hours Spent

Team Volturna spent 1209 hours outside of class working on this project. Just under two-thirds of those hours, 791, were spent during the Spring 2008 semester. Work the first semester focused on business planning, while second semester focused on design. Figure 3-1 shows the hours spent each week on the project. During the first semester the team had three major business-related deadlines – Great Lakes Entrepreneurship Quest, National Collegiate Innovators and Inventors Alliance Grant, and Calvin bizPlan, as well as a Project Proposal and Feasibility Plan. During the second semester the team’s only major deliverables were a working demonstration for Senior Design Night and a final report. The hours spent tend to fluctuate with these deadlines although the hours second semester increase almost each week.

![Figure 3-1 Weekly Team Hours](image)

Figure 3-1 Weekly Team Hours

Figure 3-2 shows each team member’s contribution each week. Overall, Charlie spent 310 hours, Gwen spent 364 hours, Ted spent 280 hours, and Richard spent 254 hours on project-related work over the semester. While Gwen, Ted, and Richard each spent approximately 70% of those hours during the second semester, Charlie put in only 55% of his hours the second semester. This was largely due to the business related activities occurring during the first semester.
Figure 3-2 Team Member Contribution to Weekly Hours
3.3. Schedule

The following images show the work breakdown schedule and Gantt chart for the team for both semesters. The images are broken into sections by type of task. These schedules were prepared using Microsoft Project.
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Resource Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Frontage</td>
<td>205 days</td>
<td>Fri 21/06</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>Document</td>
<td>195 days</td>
<td>Fri 21/06</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>Weekly Tasks</td>
<td>255 days</td>
<td>Fri 21/06</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>Reporting Process</td>
<td>232.7 days</td>
<td>Fri 21/06</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>Website</td>
<td>195 days</td>
<td>Fri 21/06</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>Update 1</td>
<td>1 hr</td>
<td>Fri 21/06</td>
<td>Charlie</td>
</tr>
<tr>
<td>07</td>
<td>Update 2</td>
<td>1 hr</td>
<td>Mon 3/09</td>
<td>Charlie</td>
</tr>
<tr>
<td>08</td>
<td>Update 3</td>
<td>1 hr</td>
<td>Mon 4/09</td>
<td>Charlie</td>
</tr>
<tr>
<td>09</td>
<td>Presentations</td>
<td>230.5 days</td>
<td>Fri 21/06</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Spring #1</td>
<td>9.48 days</td>
<td>Fri 2/06</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Plan</td>
<td>2 hrs</td>
<td>Fri 2/06</td>
<td>Ted, Kwanne</td>
</tr>
<tr>
<td>12</td>
<td>Present</td>
<td>10 mins</td>
<td>Mon 2/06</td>
<td>Ted, Kwanne</td>
</tr>
<tr>
<td>13</td>
<td>CEBAC Review</td>
<td>5.05 days</td>
<td>Fri 3/06</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Plan</td>
<td>2 hrs</td>
<td>Fri 3/06</td>
<td>Ted, Kwanne</td>
</tr>
<tr>
<td>15</td>
<td>Present</td>
<td>10 mins</td>
<td>Mon 3/06</td>
<td>Ted, Kwanne, Charlie</td>
</tr>
<tr>
<td>16</td>
<td>Final Presentation</td>
<td>3.15 days</td>
<td>Fri 4/06</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Plan</td>
<td>2 hrs</td>
<td>Fri 4/06</td>
<td>Ted, Kwanne, Charlie</td>
</tr>
<tr>
<td>18</td>
<td>Present</td>
<td>30 mins</td>
<td>Mon 4/06</td>
<td>Ted, Kwanne, Charlie</td>
</tr>
<tr>
<td>19</td>
<td>Panel Presentation</td>
<td>4.5 days</td>
<td>Mon 4/06</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Plan</td>
<td>3 hrs</td>
<td>Mon 4/06</td>
<td>Ted, Kwanne, Charlie</td>
</tr>
<tr>
<td>21</td>
<td>Present</td>
<td>30 mins</td>
<td>Tue 4/06</td>
<td>Ted, Kwanne, Charlie</td>
</tr>
<tr>
<td>22</td>
<td>Report</td>
<td>102.7 days</td>
<td>Mon 3/07</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Executive Summary</td>
<td>1 hr</td>
<td>Mon 3/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>24</td>
<td>Problem Description</td>
<td>30 mins</td>
<td>Mon 3/07</td>
<td>Ted</td>
</tr>
<tr>
<td>25</td>
<td>Problem Challenges</td>
<td>30 mins</td>
<td>Fri 3/07</td>
<td>Ted</td>
</tr>
<tr>
<td>26</td>
<td>Project Requirements</td>
<td>1 hr</td>
<td>Mon 3/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>27</td>
<td>Team Description</td>
<td>45 mins</td>
<td>Fri 3/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>28</td>
<td>Work Breakdown Structure</td>
<td>1 hr</td>
<td>Thu 3/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>29</td>
<td>Work Schedule</td>
<td>1 hr</td>
<td>Thu 3/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>30</td>
<td>Budget</td>
<td>45 mins</td>
<td>Thu 3/07</td>
<td>Ted</td>
</tr>
<tr>
<td>31</td>
<td>Concept</td>
<td>1.5 hrs</td>
<td>Mon 3/07</td>
<td>Charlie</td>
</tr>
<tr>
<td>32</td>
<td>Design Criteria</td>
<td>1.5 hrs</td>
<td>Thu 3/07</td>
<td>Ted</td>
</tr>
<tr>
<td>33</td>
<td>Finish Work</td>
<td>2 hrs</td>
<td>Fri 3/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>34</td>
<td>Design Decisions</td>
<td>03.18 days</td>
<td>Fri 3/07</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Testing Wireega</td>
<td>2 hrs</td>
<td>Wed 4/07</td>
<td>Kwanne, Charlie, Gwen, Ted</td>
</tr>
<tr>
<td>36</td>
<td>Business Outline</td>
<td>2 hrs</td>
<td>Thu 4/07</td>
<td>Charlie</td>
</tr>
<tr>
<td>37</td>
<td>Manufacturing Plan</td>
<td>2 hrs</td>
<td>Wed 4/07</td>
<td>Charlie</td>
</tr>
<tr>
<td>38</td>
<td>Conclusion</td>
<td>45 mins</td>
<td>Mon 4/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>39</td>
<td>Lessons Learned</td>
<td>45 mins</td>
<td>Mon 4/07</td>
<td>Charlie</td>
</tr>
<tr>
<td>40</td>
<td>Future Work</td>
<td>45 mins</td>
<td>Mon 4/07</td>
<td>Ted</td>
</tr>
<tr>
<td>41</td>
<td>Compile</td>
<td>2 hrs</td>
<td>Mon 4/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>42</td>
<td>Prepare #1</td>
<td>2 hrs</td>
<td>Mon 4/07</td>
<td>Kwanne, Ted, Kwanne, Charlie</td>
</tr>
<tr>
<td>43</td>
<td>Submit Draft</td>
<td>10 mins</td>
<td>Tue 4/07</td>
<td>Kwanne</td>
</tr>
<tr>
<td>44</td>
<td>Prepare #2</td>
<td>2 hrs</td>
<td>Thu 4/07</td>
<td>Kwanne, Ted, Kwanne, Charlie</td>
</tr>
<tr>
<td>45</td>
<td>Submit Find</td>
<td>10 mins</td>
<td>Fri 4/07</td>
<td>Kwanne</td>
</tr>
</tbody>
</table>

**Project Spring WBS**

**Date:** Tue 4/22/06

**Task Progress:**

**Milestones:**

**Summary:**

**External Tasks:**

**Deadline:**
Figure 3-5 Spring Semester Technical Schedule

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Resource Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype</td>
<td>207.5 days</td>
<td>Fri 2/1/08</td>
<td></td>
</tr>
<tr>
<td>Purchasing</td>
<td>11 days</td>
<td>Fri 2/1/08</td>
<td></td>
</tr>
<tr>
<td>Research Wireless Boards</td>
<td>5 hrs</td>
<td>Fri 2/1/08</td>
<td>Charlie, Kwanza</td>
</tr>
<tr>
<td>Research Video Boards</td>
<td>5 hrs</td>
<td>Fri 2/1/08</td>
<td>Tom, Ted</td>
</tr>
<tr>
<td>Purchase Trible</td>
<td>1 hr</td>
<td>Wed 2/6/08</td>
<td>Tom, Ted</td>
</tr>
<tr>
<td>Purchase Digi board</td>
<td>1 hr</td>
<td>Wed 2/6/08</td>
<td>Charlie, Kwanza</td>
</tr>
<tr>
<td>Determine the hot function</td>
<td>2 hrs</td>
<td>Fri 2/8/08</td>
<td>Tom, Ted, Charlie, Kwanza</td>
</tr>
<tr>
<td>Wireframe Transfer</td>
<td>41.5 hrs</td>
<td>Wed 3/10/08</td>
<td></td>
</tr>
<tr>
<td>Familiarize with Digi board</td>
<td>4 hrs</td>
<td>Wed 3/10/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>UDP Code</td>
<td>28.5 days</td>
<td>Fri 3/1/08</td>
<td></td>
</tr>
<tr>
<td>Research UDP Protocol</td>
<td>5 hrs</td>
<td>Fri 3/1/08</td>
<td>Tom, Ted</td>
</tr>
<tr>
<td>Poisdecde</td>
<td>2 hrs</td>
<td>Tue 3/5/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>Coa</td>
<td>4 hrs</td>
<td>Thu 3/7/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>Test small transfer</td>
<td>4 hrs</td>
<td>Fri 3/8/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>Test VLC transfer</td>
<td>4 hrs</td>
<td>Wed 3/11/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>GPU build tree</td>
<td>34.5 days</td>
<td>Fri 3/11/08</td>
<td></td>
</tr>
<tr>
<td>Research GPU</td>
<td>3 hrs</td>
<td>Fri 3/11/08</td>
<td></td>
</tr>
<tr>
<td>Poisdecde</td>
<td>1 hr</td>
<td>Tue 3/15/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>Coa</td>
<td>2 hrs</td>
<td>Wed 3/15/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>Test email transfer</td>
<td>6 hrs</td>
<td>Thu 3/19/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>Test VLC transfer</td>
<td>4 hrs</td>
<td>Fri 3/20/08</td>
<td>Kwanza, Owen</td>
</tr>
<tr>
<td>Yellow Brick coding</td>
<td>16.5 days</td>
<td>Tue 3/25/08</td>
<td></td>
</tr>
<tr>
<td>Familiarize with Trible</td>
<td>4 hrs</td>
<td>Tue 3/25/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Input buttons</td>
<td>10.5 days</td>
<td>Wed 4/5/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Research board inputs</td>
<td>6 hrs</td>
<td>Wed 4/9/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Poisdecde</td>
<td>1 hr</td>
<td>Thu 4/10/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Hand audio</td>
<td>6 hrs</td>
<td>Thu 4/10/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Hand video</td>
<td>6 hrs</td>
<td>Fri 4/11/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Coa</td>
<td>6 hrs</td>
<td>Mon 4/14/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Ted</td>
<td>6 hrs</td>
<td>Mon 4/14/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>MPEGs encoding</td>
<td>0.5 days</td>
<td>Tue 4/16/08</td>
<td></td>
</tr>
<tr>
<td>Research MPEGs processing</td>
<td>6 hrs</td>
<td>Tue 4/16/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Poisdecde</td>
<td>1 hr</td>
<td>Wed 4/16/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Coa</td>
<td>4 hrs</td>
<td>Wed 4/16/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Test video</td>
<td>4 hrs</td>
<td>Wed 4/16/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Test audio</td>
<td>4 hrs</td>
<td>Thu 4/17/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>NTSC encoding</td>
<td>0.5 days</td>
<td>Thu 4/17/08</td>
<td></td>
</tr>
<tr>
<td>Research NTSC encoder</td>
<td>6 hrs</td>
<td>Thu 4/17/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Poisdecde</td>
<td>1 hr</td>
<td>Fri 4/18/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Coa</td>
<td>4 hrs</td>
<td>Fri 4/18/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Test video</td>
<td>4 hrs</td>
<td>Mon 4/21/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Test audio</td>
<td>4 hrs</td>
<td>Mon 4/21/08</td>
<td>Ted, Charlie</td>
</tr>
<tr>
<td>Interface</td>
<td>0 days</td>
<td>Tue 4/28/08</td>
<td></td>
</tr>
<tr>
<td>Test small transfer between boards</td>
<td>6 hrs</td>
<td>Tue 4/28/08</td>
<td>Owen, Ted, Charlie, Kwanza</td>
</tr>
<tr>
<td>Test large transfer between boards</td>
<td>6 hrs</td>
<td>Wed 4/29/08</td>
<td>Owen, Ted, Charlie, Kwanza</td>
</tr>
<tr>
<td>Test streaming between boards</td>
<td>6 hrs</td>
<td>Wed 4/29/08</td>
<td>Owen, Ted, Charlie, Kwanza</td>
</tr>
<tr>
<td>Documentation</td>
<td>106.5 days</td>
<td>Fri 5/1/08</td>
<td></td>
</tr>
<tr>
<td>Weekly Tasks</td>
<td>2.5 days</td>
<td>Fri 5/1/08</td>
<td></td>
</tr>
<tr>
<td>Reporting Presence</td>
<td>31.5 days</td>
<td>Fri 5/1/08</td>
<td></td>
</tr>
</tbody>
</table>

Project: Spring WIN
Date: Thu 4/24/08
### 3.3.2. Fall Semester

![Figure 3-6 Fall Semester Administrative Schedule](image)

- **Figure 3-6 Fall Semester Administrative Schedule**
Figure 3-7 Fall Semester Weekly Task Schedule
Figure 3-8 Fall Semester Reporting Schedule
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start</th>
<th>Work</th>
<th>Task Name</th>
<th>Start</th>
<th>Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Design</td>
<td>Wed 9/5/07</td>
<td>136</td>
<td>Business Plan</td>
<td>Thu 9/5/07</td>
<td>83.50</td>
</tr>
</tbody>
</table>
3.4. **Contingency**

In the event that the project was unable to be carried to completion, Volturna identified the goals in Table 3-3 as contingency plans. The top row in the table shows the minimum achievement that Volturna would consider to be acceptable for the project. Each successive layer adds complexity to the functionality of the device. Row 3 shows the “ideal” plan, this is the team’s initial goal. Row 4 goes beyond the initial goal, requiring higher quality streaming. This plan would be used if Volturna were to complete Plan 3 ahead of schedule.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (minimum requirement)</td>
<td>Audio streaming to TV with proper decoding.</td>
</tr>
<tr>
<td>2</td>
<td>Video streaming to TV with proper encoding and decoding or audio/video streaming with significant lag.</td>
</tr>
<tr>
<td>3 (ideal)</td>
<td>YouTube quality video and audio streaming to TV with proper encoding and decoding. Lag is less than a second.</td>
</tr>
<tr>
<td>4</td>
<td>DVD quality video and audio streaming to TV with proper encoding and decoding. Lag is less than a second.</td>
</tr>
</tbody>
</table>

3.5. **Project Budget**

Volturna’s final budget was modest considering the scope of the project. The team was able to purchase a total of three development kits, two of which were used in the final design. The total budget for the project was $1800. The Calvin College Senior Design Fund provided $300 and the team won another $1500 from the Calvin College bizPlan 2007 competition.

Initially, the team purchased a ZigBee kit for wireless transmission to start work on the prototype. However, actual testing resulted in a Zigbee transfer rate of 250kpbs as opposed to the previously expected 1Mbps. This bandwidth limitation was deemed insufficient to achieve acceptable video streaming. The Digi ConnectCore Wi-9C was purchased as an alternative. The final purchase was a Texas Instruments (TI) TMDXEVM355 evaluation module. This is a digital signal processor (DSP) that was used to implement all of the receiver-side video encoding and decoding. The overall cost of the project (not including the “free” man-hours of labor involved) was approximately $1500, equal to that of the business plan award. The team remains comfortably within the total fiscal budget, as seen in Table 3-4.


4. PRELIMINARY RESEARCH

4.1. Products

Team Volturna conducted a study of the prior work that has been conducted in this field. Products that offered similar services to electronics consumers include the recently introduced SlingCatcher, the PC-on-TV Player made by D-Link, and the Ultimate Wireless made by GrandTec USA. Other products studied are still in the research and development stages and were revealed at January 2008’s Consumer Electronics Show (CES), such as the SlingCatcher and the TiVO Desktop 2.6.

4.1.1. Ultimate Wireless by GrandTec

The Ultimate Wireless is a PC-to-TV converter package consisting of a transmitter unit and a receiver unit. Broadcasting over the 2.4 GHz frequency, the Ultimate Wireless has 4 selectable channels and broadcasts over a 125-150 foot range. The transmitter unit is equipped with a standard 15 pin VGA input and 3.5mm stereo line-in input port. The receiver unit has a 15 pin VGA output port, as well as a composite S-Video output port and RCA audio/video output ports. The Ultimate Wireless uses P.L.L. frequency generation and F.M. demodulation. By taking in only VGA and line-in audio signals on the receiver, the Ultimate Wireless eliminates the need for any digital to analog conversion. However, this package does require two units, shown in Figure 4-1 the transmitter and receiver, which require separate power sources and there is still the need to connect wires from the PC to the transmitter before broadcasting to the receiver. The entire package is available directly from the manufacturers for the price of $149.99.
4.1.2. **PC on TV Player by D-Link**

D-Link’s DPG-1200 or the PC-on-TV Media Player was newly introduced at the Consumer Electronics Show (CES) 2008 and was released in April 2008. The DPG-1200 transmits audio and video wirelessly over an 802.11g network, an 802.3u network and uses TCP/IP protocol. Features of the DPG-1200 include a remote control equipped with a scroll ball to allow use as a mouse and codec independence, allowing the user to play whatever is shown on the PC screen. The DPG-1200 however does not support DirectX applications, thus not all content on the PC is viewable on the TV. Also, to use this device wirelessly the user must already have a wireless network configured using a separate wireless router. Furthermore, the product has the rather high end price of $239.99 from the manufacturer.

4.1.3. **SlingCatcher by Sling Media**

Sling Media, a home entertainment electronics company, revealed a prototype of their SlingCatcher at the January 2008 CES. The product is very similar to the VWMP design and is expected to be released to market in the near future. Operating over an 802.11g wireless network, the SlingCatcher consists of a single receiver unit, equipped with RCA Audio/Video ports, a Digital Audio port, an S-Video port, an HDMI Output port, and Component Output for High Definition TVs. According to Sling Media the SlingCatcher will wirelessly transmit audio and video from a PC to a TV. In addition to its extensive output forms, the SlingCatcher also comes with a remote controller. A drawback of the SlingCatcher is its more advanced system requirements as compared to competitive products. It requires 512MB of internal memory if the user is running Windows XP on their PC and 1GB internal memory if the user is running Windows Vista. In addition to that the SlingCatcher requires the user...
possess a Broadband internet connection and a router either wired or wireless. An estimated price for this product is not yet available.

Figure 4-3 Sling Media’s Sling Catcher - Connection Panel

4.1.4. **SRU4000 by Phillips**

The SRU4000, Figure 4-4, is Phillips previously released Wireless Audio Video transmitter. It consists of two units; a sender and a receiver which both have the same ports. Solely equipped with RCA Audio and Video ports on the sender and receiver units, the SRU4000 is far less versatile than its competition and only transmits analog signals over RF frequencies. In order to transmit from a PC to a television using this product, the user would be required to purchase an additional S-Video to RCA Video cable which costs $50-$70. Both the receiver and sender units require separate powering.

Figure 4-4 Phillips SRU4000

4.1.5. **Pro PC/TV Wireless by Altech**

An improvement on Altech’s former Pro A/V unit, the Pro PC/TV Wireless, Figure 4-5, converts a VGA to a TV signal and transmits it wirelessly over a range of up to 100 ft. To its advantage this product comes with 4 selectable channels and requires no software installation. However, it does consist of two separate units requiring individual powering, it has a low transmission range of 100ft, the transmitter takes only VGA and RCA A/V
inputs and the receiver is only equipped with only RCA A/V ports. The Pro PC/TV wireless is limited in its input/output ports and this is in no way complimented by its price of $171.95 from the manufacturers.

Figure 4-5 Altech Pro PC/TV Wireless(5)

4.2. Patents

These patents were identified in the team’s December 2007 Project Proposal and Feasibility Study (PPFS) as potentially affecting the VWMP. Here is a review of those patents, along with a short discussion of how they may affect the final VWMP design.

PATENT 1 7,302,239
Wireless receiving device
ABSTRACT:
A wireless receiving device of the present invention is so arranged that (i) a high-frequency circuit section is provided on a front surface of a printed board and (ii) a digital video signal processing section and a wireless LAN digital signal processing section are provided on a back surface of the printed board. Further, an earth pattern is provided in a middle position between the front and back surfaces of the printed board, so that (i) the high-frequency circuit section and (ii) the digital video signal processing section and the wireless LAN digital signal processing section are shielded from one another using electromagnetic shielding by the earth pattern. Thus, when members that are commonly used for the wireless LAN and for wireless video reception are formed on a single printed board in order to reduce the size of the printed board, it is possible to effectively reduce the lowering of signal quality caused by the noise.

COMMENT: This patent no longer affects the team as it expressly covers a digital signal processor and wireless receiver on one board. Ours are separate devices and thus are not covered in this patent.

PATENT 2 7,301,928
Wireless packet transfer apparatus and method
ABSTRACT:
An error correction encoding rate selection table is provided in an error correction processing unit of a packet
transfer apparatus, and the table stores an error correction encoding rate preset to maintain a desired QoS in
correspondence with a protocol type and an application type. When a transmission packet is transferred to a
wireless transmission path, an encoding control unit judges the protocol type and application type of a
transmission packet from a header of the transmission packet, in accordance with a judgment result and the
error correction encoding rate selection table, an error correction encoding rate is selected, and the transmission
packet is subjected to error correction encoding and transferred.

COMMENT: Due to the team’s use of the UDP protocol for transmission, error correcting has been eliminated
from the design. Quick delivery of packets was determined to be more important than reliable delivery of
packets. Thus this patent which claims an error correction encoding scheme, does not cover the VWMP.

PATENT 3  7,260,360
Multi-band ZigBee transceiver supporting IEEE 802.15.4 wireless communication
ABSTRACT:
Disclosed herein is a multi-band ZigBee transceiver for supporting IEEE 802.15.4 wireless communications. In
the multi-band ZigBee transceiver, a Multi-Mode Modem (MMM) selects any one of a European version
standard using 860 MHz band, a US version standard using 920 MHz band, and a worldwide version standard
using 2.4 GHz ISM band among IEEE 802.15.4 standards. A frequency synthesizing unit variably adjusts a
carrier frequency according to the transmission standard. A transmission unit receives a digital modulated
signal, low-pass-filters the digital modulated signal with a bandwidth thereof being variably adjusted, and up-
converts the filtered digital modulated signal into an RF modulated signal corresponding to the selected
transmission channel. A receiving unit down-converts the RF modulated signal into a BB modulated signal
using the carrier frequency, low-pass-filters the BB modulated signal with a bandwidth thereof being variably
adjusted according to the selected transmission standard, converts the filtered BB modulated signal into a digital
modulated signal, and outputs the digital modulated signal to the MMM. A transmission/reception switch unit
outputs the RF modulated signal, input from the transmission unit, to an antenna, or outputs the RF modulated
signal, received from the antenna, to the receiving unit.

COMMENT: This patent is irrelevant as the time has opted against the ZigBee wireless technology in favor of
the 802.11 standard.

PATENT 4  7,302,241
Wireless receiver without AGC
ABSTRACT:
A **wireless** communication unit (300) incorporates a receiver comprising radio frequency circuitry (210, 220, 230, 240) for receiving a radio frequency signal and converting the radio frequency signal to a low frequency signal. A signal level adjustment circuit receives the low frequency signal and an analogue to digital converter (370), operably coupled to the signal level adjustment circuit receives an adjusted low frequency signal and providing a digital received signal. A signal processor (108) operably coupled to the analogue to digital converter (370) processes the digital received signal. The signal level adjustment circuit includes a low frequency amplifier (360) whose gain is arranged to be dependent upon a clip point of the analogue to digital converter (370).

**COMMENT**: It is unlikely, but still possible that this patent will affect the VWMP. It appears to claim a wireless communication unit containing an analog-to-digital converter. The VWMP in contrast uses a digital-to-analog converter. The team’s understanding of the TI chip is that the gain of the signal is **not** affected by the digital-to-analog converter which seems to be the main point of this patent.

**PATENT 5  6,941,114**

*USB-based wireless transmitting/receiving system*

**ABSTRACT:**
A USB-based wireless transmitting/receiving system includes a transmitting portion connected to at least one USB-based peripheral device for receiving a signal from the peripheral device and a receiving portion connected to a computer host and coupled to the transmitting portion in a wireless fashion for receiving and applying the signal from the transmitting portion to the computer host whereby a wireless communication is established between the computer host and the USB-based peripheral device. Both the transmitting portion and the receiving portion include a central processing unit for processing signals received, a receiving unit for receiving signals, a transmitting unit for transmitting signals and a power supply system for powering the transmitting portion or the receiving portion. The power supply system includes a regulation circuit adapted to connect to an electric main. The power supply system may selectively include a primary battery set, a secondary battery set and/or a solar cell system, all connected to a controller to which the regulation circuit is also connected for selectively powering the transmitting portion or the receiving portion.

**COMMENT**: This patent does not affect our prototype design which uses an 802.11 network, but it would likely affect our “ideal” design which would use the wireless USB protocol.

**PATENT 6  7,263,573**

*Wireless USB hardware scheduling*

**ABSTRACT:**
A USB-based wireless transmitting/receiving system includes a transmitting portion connected to at least one USB-based peripheral device for receiving a signal from the peripheral device and a receiving portion connected to a computer host and coupled to the transmitting portion in a wireless fashion for receiving and applying the signal from the transmitting portion to the computer host whereby a wireless communication is established between the computer host and the USB-based peripheral device. Both the transmitting portion and the receiving portion include a central processing unit for processing signals received, a receiving unit for receiving signals, a transmitting unit for transmitting signals and a power supply system for powering the transmitting portion or the receiving portion. The power supply system includes a regulation circuit adapted to connect to an electric main. The power supply system may selectively include a primary battery set, a secondary battery set and/or a solar cell system, all connected to a controller to which the regulation circuit is also connected for selectively powering the transmitting portion or the receiving portion.

COMMENT: As with patent 5, this patent does not affect our current prototype but still may affect the ideal WUSB design.

PATENT 7  7,269,400

*Wireless peripheral interface with universal serial bus port*

**ABSTRACT:**

A wireless peripheral interface is used for coupling with a Universal Serial Bus (*USB*) port for connecting a wireless peripheral with a host computer or controller. Wireless peripheral devices (e.g., keyboard, mouse, trackball, touch pad, joysticks, and game controllers) transmit communication signals, e.g., radio frequency (RF) signals, to the peripheral interface, which are received and processed into formats suitable for transmission to the host computer or controller via *USB*, either alone or in combination with other standard external bus systems, such as serial and PS/2.

COMMENT: This patent does not affect the prototype but may still affect the ideal WUSB design.

5. **PROPOSED SOLUTION**

Ideally Team Volturina would have a final product design that outweighs the advantages of its competition in terms of price, media input and output port options, package aesthetics, user friendliness, power consumption, range of operation and the extra setup requirements. For instance, the elimination of the need for a transmitter unit usually requires the use of an 802.11 network and with that comes the need for a router, ideally a wireless router.
Table 5-1 provides an overview of the competition’s strengths and weaknesses.
Table 5-1’s criteria of User Friendliness was determined based on the simplicity of use. That is, if the product requires several connections or more complex connections that the competition or if the product required the user to setup a LAN. Furthermore, it should be noted that the additional requirements implies peripherals or cables that are not included in the package. For instance the Phillips SRU4000 has an S-Video to RCA cable in its Additional Requirements section because this is required to connect to a TV with RCA input ports and this is not included in the package.

This provided Team Volturna a more defined idea of the most effective use of the team’s resources to gain a competitive advantage.
Table 5-1 Competing Product Comparison

<table>
<thead>
<tr>
<th></th>
<th>D-Link DPG-1200</th>
<th>Sling Media Sling Catcher</th>
<th>Altech Pro PC/TV Wireless</th>
<th>Phillips SRU4000</th>
<th>GrandTec Ultimate Wireless</th>
<th>Volturna VWMP (prototype design)</th>
<th>Volturna VWMP (production design)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless Protocol</td>
<td>802.11g</td>
<td>802.11g</td>
<td>802.11g</td>
<td>802.11g</td>
<td>802.11g</td>
<td>802.11g</td>
<td>Wireless USB</td>
</tr>
<tr>
<td>Package Aesthetics (Rating 1-4)</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Video Resolution</td>
<td>720 x526</td>
<td>N/A</td>
<td>2048x1536</td>
<td>Data Not Provided</td>
<td>1600x1200</td>
<td>2048x1536</td>
<td>2048x1536</td>
</tr>
<tr>
<td>Range</td>
<td>100 feet</td>
<td>100 feet</td>
<td>100 feet</td>
<td>125 feet</td>
<td>125-150 feet</td>
<td>100 feet</td>
<td>33 feet</td>
</tr>
<tr>
<td>Reciever Input Format</td>
<td>802.11g from PC</td>
<td>802.11g from PC</td>
<td>VGA, RCA A/V</td>
<td>RCA A/V</td>
<td>VGA, Line-in Audio</td>
<td>802.11g from PC</td>
<td>Wireless USB from Transmitter</td>
</tr>
<tr>
<td>Price</td>
<td>$239</td>
<td>Not yet available</td>
<td>$172</td>
<td>Not available</td>
<td>$149</td>
<td>$120 (Estimated)</td>
<td>$120 (estimated)</td>
</tr>
<tr>
<td>User Friendliness (Rating 1-4)</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Number of Units (i.e. transmitter, receiver, etc.)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Additional Requirements</td>
<td>Not DirectX Compatible, Wireless Router, 130MB Hard Disk, 512 RAM</td>
<td>Wireless Router, Broadband Internet, 1GB RAM</td>
<td>None</td>
<td>S-Video to RCA A/V Cable *</td>
<td>None</td>
<td>Wireless Router *</td>
<td>None, Windows XP or Newer</td>
</tr>
</tbody>
</table>

Note: In Table 6-1 an * indicates host PC requirements were not made available by manufacturer.

Upon close analysis of the competition’s data, Team Volturna’s proposed solution for a wireless transmission device was the single receiver unit, 802.11g, codec independent, and low cost design of the VWMP which would output via RCA Composite Audio and Video ports. Component Video is of lower quality than Composite
Video, therefore the VWMP will still be at a disadvantage to the D-Link product. However, this solution would be in the $120-160 range making it relatively low in cost when compared with the competition. The selection of Wireless USB as opposed to the conventional 802.11g method would eliminate the necessity of a processor on the receiver unit, potentially reducing the final product cost.

The option of 802.11g was selected for means of wireless transmission as this was found to be the best suited for the high bit-rate required to transportation digital data. The package aesthetics for the VWMP in the above table are for the final product as opposed to the prototype. Here Volturna has the advantage of knowing the competition, allowing us to develop more aesthetic, user-friendly packaging. Figure 11-1 displays the VWMP product package design.

The input data to the VWMP receiver will be delivered over an 802.11g network provided by the host PC. To facilitate ease of this step in product, Team Volturna shall provide a CD as part of the final package containing easy to install software which will setup a LAN recognizable by the VWMP receiver.

The VWMP’s RCA A/V output is compatible with all commercially available TVs and projectors that have RCA Composite A/V ports. One RCA A/V cable will be required to connect the receiver to the display device, but no S-Video cables or extended range cables will be needed, in contrast to the Phillips SRU4000. This does however mean the VWMP provides lower quality than that of an S-Video or Component equipped product such as the D-Link product. Additionally, the VWMP will not require an existing internet connection or a rather large 1 GB RAM as the SlingCatcher does. The prototype design VWMP will be compatible with any PC that can run Windows 2000 or newer, thus require 256 MB RAM, while the product design VWMP will be compatible with any PC equipped with a USB port. The VWMP’s advantage over D-Link’s DPG-1200 lies in the VWMP’s lower price, the D-Link product is the most expensive of the competition; this is most likely due to D-Link’s early manufacture of the product thereby granting them a monopoly. In addition the D-Link product uses the higher quality but more expensive output form of Component Video. Otherwise the DPG-1200 would have the advantage of a single additional VGA format output.

The single unit design of the VWMP eliminates the inconvenience of having to attach a transmitter unit to a PC, this is especially advantageous when the PC is a laptop that is often moved around. The use of a single unit also reduces the product’s overall power consumption. The DM-355 which will serve as the digital signal processor chip requires 3.3V, the Wi-9C also requires a 3.3V power supply. The prototype design for the VWMP is estimated to require a 3.3 power supply. The VWMP’s product design will also be codec independent, meaning it will transmit any audio and video that play from the speakers and screen of the PC. The coded independence will be achieved by using the open source software VLC which enables the user to broadcast the entire screen content, this eliminates the problem of a player with the required video codecs. In other words whatever is seen on the screened is broadcast. The VWMP’s competitive edge therefore lies in its low cost and simplicity.
6. PRODUCT DESIGN

6.1. System Description

Team Volturna is designing a low cost, aesthetically pleasing high audio/video quality, wireless USB multimedia device to link between a personal computer and a multimedia broadcasting module, such as a television or a projector. The basic overview of Team Volturna’s project can be seen in the functional block diagram in Figure 6-1. The idea behind this design is that the wired component could be replaced with a wireless connection and the device would work the same. The transmitter is responsible for encoding and transmitting the signal and the wireless receiver for decoding the received signal, so that the output signal from the receiver is identical to the audio/video signal. After the signal passes through the wireless receiver, it will be sent to a DSP that will convert it to a PAL or NTSC analog TV signal. Together, the wireless receiver and the DSP make up the device receiver. The drivers will act as an auxiliary graphics card and audio card of the PC so that the device captures the entire monitor image and sounds in the same way a wired projector or S-Video and line-in audio combination would. The designed software drivers will compress and encode the signal and then put it through to the transmitter which will send the signal to the receiver. The team’s vision of the transmitter is a device comparable to a pen drive or a USB 802.11g adapter. The receiver will be about the size of a wireless router.

Figure 6-1 High-Level Block Diagram

An alternative design approach involves directly connecting a Video Graphics Array (VGA) plug coupled with a line-in audio cable and transmitting both data lines via radio frequency to a receiver. This option could also be slightly modified by replacing the VGA plug with an S-Video plug. Both of these choices were eliminated based on the requirements for senior design. These alternatives involved a lot less technical workload and perhaps
might have been too simple to cover the class requirements. Also these options would involve the use of two peripheral ports on the PC which the team thought would compromise the design goal of an easy to use device. The alternative design also would also require an external power source for the transmitter.

6.2. Norms

Team Volturna has identified the design norms of transparency, integrity, and trust as especially applicable to their project. As a transparent design, the final product must be easy to use for a user with nominal technical experience. It must also be durable. The product should last for several years assuming normal intended use. In addition to being easy to use, the end product must have a design that implies its function, fulfilling the norm of integrity. The premise of the project is to promote human interactions and relationships by allowing users to easily share their stored media with others via a multimedia broadcasting device like a television or projector this would conform to the design norm of encouraging social interaction. A final design norm the team is committed to is design trust. The product must be reliable as the user is depending on it to faithfully transmit their multimedia. The product could be used in a corporate or business setting. This adds to the importance of user confidence in the product.

6.3. Criteria

The team’s product design criteria were chosen with the goals of marketability and cost-reduction, especially in light of the team’s entrepreneurial focus. The design must meet these objectives while successfully achieving its functional purpose. The intended purpose of the VWMP is to facilitate social interaction through the sharing of digital media. For this reason, the team felt that the integration of several key design norms into the entirety of the project were crucial to success. Conforming to these norms had the additional benefit of increasing marketability, a large influence in critical project decisions.

The product design criteria to remain video-player independent also involved the design transparency norm. Video player independence allows customers who may be working from a variety of operating systems (Windows, Mac OS, Linux, etc.) to use the product, as well as allows customers to transmit media regardless of its compatibility with Windows’ Media Player, iTunes, or any other proprietary media player option.

A third design criteria was robustness, hence the norms of trust and integrity. The team felt that the user of the device should be able to trust that the device would function properly every time it was used. Hence the design should be able to withstand reasonable temperature extremes and long periods of both use and non-use. Also, the design norm of stewardship factors into the project, as the team strives to remain under budget and to manage all team resources with minimal waste.

Finally, the team organization decision to work on each part of the design in small groups was made in acknowledgement of the humility design norm. This relates to the fact that the designers are imperfect and
possess the capability to make mistakes. By recognizing this, the team was able to compensate for potential errors in the design by team programming regularly checking each other’s work.

Specific design decisions include its mandatory ability to wirelessly stream data from a computer to any NTSC-ready display device. The prototype design must accomplish this through a wireless network. This ought not to interfere with other common use of the computer system.

I. Physical requirements were determined on the basis of marketability, that is, portability, aesthetics and durability. For example the requirement that the receiver case be mountable is linked to the team’s goal of having this device be used for presentations. A mountable case could be mounted along with a projector to a ceiling or wall. Based on the above criteria the team set the following physical requirements:

   A. Weight of the transmitter will not exceed 0.5 pounds. This will facilitate convenient pocket-size portability of the product, and a transmitter that can be easily passed around or attached to different laptops or desktops computers.
   B. Weight of the receiver will not exceed will not exceed 2.5 pounds. This will facilitate convenient portability of the product by hand or in a carrying case,
   C. Dimensions of the transmitter will not exceed .45 x 1.2 x 2 inches to be competitive with existing products.
   D. Dimensions of the receiver will not exceed 6 x 6 x 2 inches.
   E. Receiver housing will be surface mountable, using screws allowing customers to mount the device on the ceiling or wall near a projector or TV.

II. Power requirements were determined on the basis of power availability and typical power sources for similar device. An adapter will be needed to convert the AC power source into a DC source that is compatible with our digital hardware and chipsets. The specific adapter needed will be contingent on the power requirements of the hardware we select. Based on the aforementioned criteria the following power requirements were made:

   A. The transmitter will be powered through the USB port of the computer to facilitate easy use.
   B. The receiver will be powered via a 120 V, 60 Hz wall socket to conform to the North American AC power standard. Specific power requirements, such as allowable fluctuation from the nominal value, will be a design decision as it is dependent on the hardware selected.

III. The hardware requirements of the design for the transmitter and the receiver are as follows:

   A. The minimum acceptable bit-rate for a receiver will be 1 Mbit/s. This is the bit-rate associated with very low quality video conferencing.
B. The minimum transmission range will be 10 meters to ensure that the product can be used in an average sized-room to transmit between a PC and projector or TV.

C. Transmitter and receiver will facilitate wireless media transmission between a PC that has a USB drive, and can run VLC and a TV or projector.

D. The TV or projector will have RCA composite audio/video female ports.

D. Receiver will use a first-come, first-served handshaking protocol to avoid conflicts between received signals. That is, the receiver unit will establish a closed connection with the first device that requests it; until that connection is terminated no other device may connect to the receiver.

E. Receiver will have an external on/off power switch to conserve power when not in use. Note that the device can be ceiling mounted. If the user decides to mount the device on a ceiling then the user is responsible for a means to reach up to ceiling in order to switch of the device.

F. Receiver will have an LED to indicate power on status, receiving data status and idle status each of these states will be confirmed by a steady on LED. Figure 6-3 illustrates the design for the receiver front panel where the device status LED’s are located.

G. Receiver package will include a 3 ft RCA composite video and audio plug cable to connect to a TV or projector.

H. Wireless signal will comply with all relevant FCC regulations

IV. Software requirements of the system pertain to the written drivers and the options they enable for the transmitter to transmit audio and video from the PC to the receiver unit. Many of these requirements, such as the operating system, stem directly from Volturna’s initial market survey. Software requirements are:

A. The software will be designed for Windows XP home and office editions.

B. The VWMP will be transmit the entire PC screen and transmit any audio from the sound card via the VLC program. The VWMP will therefore be compatible with any video player software.

C. The software shall be user friendly. The user will be able to plug the transmitter in and follow a series of explicit on-screen setup instructions to begin using the device.

D. The software drivers shall compress the signal into a byte/second signal that is compatible with the wireless protocol.

E. The software drivers will encode the signal into a form that the transmitter recognizes as a wireless packet.

V. Functional requirements of the transceiver are outlined here:

A. The host PC computer will transmit its screen contents, including audio.
B. The output from the receiver will be an NTSC analog TV signal

VI. Labor and Parts: It is a design goal that total labor and parts required for the final design will cost no more than $130 per VWMP (receiver and transmitter) to make the product market competitive. However, pending the use of ultra-wideband wireless transmission, Volturna may reset this price cap. Regardless, Volturna will ensure that any new price cap is still market competitive for the product’s quality. Volturna also acknowledges that total labor and parts do not make the final retail cost and that there are other factors such as distribution and shipping costs involved in making that decision. Since the Volturna team is looking at entrepreneurship and the actual marketing of this product, hardware shall be selected such that it is available in quantities of 5000+. Section 11 gives more information on the selection of this number. This will make the design more readily adaptable to mass manufacture. We are also looking at using as many integrated circuits as possible to reduce circuit board complexity and population time.

6.4. Alternatives and Analysis

6.4.1. Wireless Protocol

For wireless media transmission, the data rate, range, cost and other factors are important specifications that Volturna considered in Table 7-1. Video streaming can be done at a number of rates depending on video quality. Four protocols were considered – WirelessUSB, Zigbee, 802.11g, and 802.11n (draft). These protocols are most suited to streaming video and are detailed in Error! Reference source not found..

Table 6-1 Wireless Protocol Comparison

<table>
<thead>
<tr>
<th></th>
<th>WirelessUSB</th>
<th>Zigbee</th>
<th>802.11g</th>
<th>Draft 802.11n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Rate</strong></td>
<td>480Mbps</td>
<td>1Mbps</td>
<td>54Mbps</td>
<td>300Mbps</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>3-10 meters</td>
<td>Up to 20 meters</td>
<td>Up to 30 meters</td>
<td>Up to 90 meters</td>
</tr>
<tr>
<td><strong>Typical Uses</strong></td>
<td>Media transmission</td>
<td>Home sensor networks/controls</td>
<td>Home/Business networks</td>
<td>Business networks</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>$6000-$30000</td>
<td>$250</td>
<td>$500</td>
<td>$1000-$10000</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>Same as wired USB</td>
<td>Variable, up to 128-bit encryption</td>
<td>WEP/WPA</td>
<td>WEP/WPA</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>Very new, hard to locate</td>
<td>Common, but difficult to find high bandwidth chips</td>
<td>Very common</td>
<td>Moderately easy to find</td>
</tr>
<tr>
<td><strong>Hardware Requirements</strong></td>
<td>Self-contained</td>
<td>Self-contained</td>
<td>IP – need small board computer</td>
<td>IP – need small board computer</td>
</tr>
</tbody>
</table>
Table 6-2 MPEG-2 Operation Levels (6)

<table>
<thead>
<tr>
<th>Name</th>
<th>Pixel/Line</th>
<th>Lines</th>
<th>Max Frame Rate (Hz)</th>
<th>Bit Rate (Mbit/s)</th>
<th>Buffer Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Level (LL)</td>
<td>352</td>
<td>288</td>
<td>30</td>
<td>4</td>
<td>475136</td>
</tr>
<tr>
<td>Main Level (ML)</td>
<td>720</td>
<td>576</td>
<td>30</td>
<td>15</td>
<td>1835008</td>
</tr>
<tr>
<td>High 1440 (H-14)</td>
<td>1440</td>
<td>1152</td>
<td>60</td>
<td>60</td>
<td>7340032</td>
</tr>
<tr>
<td>High Level (HL)</td>
<td>1920</td>
<td>1152</td>
<td>60</td>
<td>80</td>
<td>9781248</td>
</tr>
</tbody>
</table>

These different levels shown in
Table 6-2 are typically applied to the various settings depicted in Error! Reference source not found..

Generally video bit-rates are between 1.5Mbps and 20Mbps, although streaming of very low quality video can be done at 0.8Mbps as seen in Error! Reference source not found.. Zigbee is close to this lower band and therefore would provide the lowest quality video. WirelessUSB has a bit-rate of 480Mbps which would allow the VWMP to transmit a compressed HD video. The team’s analysis of four wireless alternatives – WirelessUSB, Zigbee, 802.11g, and 802.11n (draft) are shown in Error! Reference source not found.. In Table 7.2 Bandwidth refers to rate at which the protocol transmits data, this is usually measured in Mbps. The price row refers to the developmental kit price not the price of the actual final product based on this protocol. The price row is included so that this decision matrix assists not only in the selection of the most theoretically suitable protocol but also the most economically suitable. If need be such a row would be useful in selection of a more affordable protocol as in the case of this project. Security protocol refers to the protection the protocol offers to the data being broadcast, that is to encrypt it making it unreadable by undesired receiving devices. For the purpose of this project security is of little importance. Range refers to the distance across which the data can be broadcast using this protocol. Hardware refers to the quantity of hardware that is required for the use of this protocol; Volturna does not desire to use a protocol that requires too much additional hardware.

Table 6-3 Wireless Protocol Decision Matrix

<table>
<thead>
<tr>
<th>ALTERNATIVES</th>
<th>Wireless USB</th>
<th>Zigbee</th>
<th>802.11g</th>
<th>Draft 802.11n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Criteria</strong></td>
<td><strong>Decision Model</strong></td>
<td><strong>Bandwidth</strong></td>
<td><strong>Weight</strong></td>
<td><strong>Rating</strong></td>
</tr>
<tr>
<td><strong>&lt; 0.384 Mbps</strong></td>
<td>Video conference</td>
<td>(MPEG-4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>&lt; 1.5 Mbps</strong></td>
<td>Video in a window</td>
<td>(MPEG-1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1-2 Mbps</strong></td>
<td>HVS quality full screen</td>
<td>(MPEG-2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2-3 Mbps</strong></td>
<td>Broadcast NTSC</td>
<td>(MPEG-2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4-6 Mbps</strong></td>
<td>Broadcast PAL</td>
<td>(MPEG-2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>8-10 Mbps</strong></td>
<td>Professional PAL</td>
<td>(MPEG-2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>12-20 Mbps</strong></td>
<td>Broadcast HDTV</td>
<td>(MPEG-2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>27.5-40 Mbps</strong></td>
<td>DVB satellite multiplex</td>
<td>(MPEG-2 Transport)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>32-40 Mbps</strong></td>
<td>Professional HDTV</td>
<td>(MPEG-2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>34-50 Mbps</strong></td>
<td>Contribution TV</td>
<td>(MPEG-2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>140 Mbps</strong></td>
<td>Contribution HDTV</td>
<td>(MPEG-2-I)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>168 Mbps</strong></td>
<td>Raw NTSC</td>
<td>(uncompressed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>216 Mbps</strong></td>
<td>Raw PAL</td>
<td>(uncompressed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>270 Mbps</strong></td>
<td>Raw contribution PAL</td>
<td>(uncompressed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1-1.5 Gbps</strong></td>
<td>Raw HDTV</td>
<td>(uncompressed)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Initially, the team focused design efforts on the production design in which we would use the Wireless USB protocol for connection to a PC. This technology would allow the newer laptops to be able to use a high bandwidth protocol. By allowing this type of ease of use, the design would maintain transparency for the end user. Unfortunately, the relative newness of the technology presented prohibitively high costs; the lowest quotes received for a dev kit being over $20,000. This high cost caused the team to seek out other potential methods of funding. Professor VanderLeest notified us of the NCIIA E-team grants, which provide a maximum of $20,000. The team spoke with a few companies to see about a possible educational discount, and came across one, Staccato Communications, who stated that they might be able to offer a Wireless USB development kit, without any support in four months for about $17,000. This proposed cost would work in line with the team’s anticipated schedule and the NCIIA grant proposal notification.

6.4.2. Development Platform

For the product development, we would use the Staccato Communications Wireless USB development kit. The Staccato Communications system provides reference designs for their products and also demonstrated integrity by being honest about their situation, as well as the possible availability of a lower cost alternative that would fall within our price range around the time we would require it. This decision would provide us with the ability to make the design much more transparent for users, as the system would appear as a typical USB device, with all of the ease of use that accompanies it. Additionally, the Wireless USB development kit provided by Staccato Communications would include a transmitter for the Wireless USB protocol, thereby removing the need to find the scarce part.

6.4.3. Video Compression Algorithm

This decision was identical to the one made for the prototype design and can be found in section 7.1.4.

6.5. System Architecture and Design

The basic system architecture of the product would consist of a Wireless USB transmitter, which could be an add on much like a USB dongle, but likely will be embedded in future laptop computers. This will transmit the multimedia over the Wireless USB signal to the receiver. The basic idea of what the receiver would look like is shown in Figure 6-3.
Of important note on this is the red and white composite RCA connectors for stereo audio and the yellow RCA connector for composite, NTSC video. The receiver unit will also contain an internal antenna that will connect to the Wireless USB signal. The connect light will indicate that the device is connected to a transmitter, and the receive light will indicate that it is receiving video data. The power switch will control if the receiver is powered on or not, and the power light will indicate its current status. The power input will accept the input from a transformer which will provide 3.3V.

7. Prototype Design

7.1. Alternatives and Analysis: Prototype

7.1.1. Wireless Protocol
In order to start work on the prototype, the team chose to purchase a lower-bandwidth Zigbee development kit rather than an 802.11 development kit, as the Zigbee unit's conceptual similarity to the USB protocol would ease the transition to our desired Wireless USB protocol.

In mid-December of 2007, when the team submitted the grant proposal, the market had only two main competitors offering similar products: the FlyImage 5000 and SmartHome's 7743B. The decision on the grant proposal was made in late February. From the time we submitted the grant proposal in December to the time the NCIIA made the decision in late February, Las Vegas held the Consumer Electronics Show (CES), where many companies displayed products similar to ours. As a result, when the NCIIA returned their verdict on our grant proposal, they had this to say: "Reviewers agreed that there is a clear need for this technology but were concerned about the saturation of the market by other companies with more funding who have already begun working on this. They believed that it would take a large amount of capital and expertise to get a consumer
electronic product like this out into the marketplace quickly. They were however impressed with the team and encourage you to develop another project with these capable students."

As the team began work with the Zigbee wireless platform in early February, it became clear that we had made some errors in judgment regarding the capabilities of the board, as well as misleading information in the development kit datasheet. The issue stemmed from two main misjudgments:

- The expected throughput of the wireless protocol was 1Mbps, when in actuality, the maximum of the wireless protocol throughput is 1Mbps, and the average is much lower 25 kbps.
- The transmission through the Zigbee chip and out to the input/output ports would be 1Mbps, however, the actual ability of the Zigbee chip to accept data in, and output it through the system is around 25kbps.

Because of these assumptions, the kit would not suit our needs as we required at least 1Mbps video for the lowest quality – the lowest frame rate video that we would accept. This forced our team to reconsider 802.11, especially after recent similar products emerged at CES. The team then decided to spend an extra $500 towards an 802.11b, Net+OS development kit, but only after they had checked over the boards specifications to attempt to avoid the errors they faced with the Zigbee unit by calling the obtaining the benchmark test data with the dev kits average throughputs. This decision proved wise after finding out in mid March that we were denied the NCIIA grant.

7.1.2. Wireless Development Kit

Initially the MaxStream Zigbee development kit was selected as it was the only development kit the team could locate that used the Zigbee protocol and promised a 1Mbps throughput. When it became clear that even the MaxStream was not going come close to attaining that throughput, the team went with the next best alternative shown in Error! Reference source not found.. A rapid analysis of available 802.11 wireless single board computer development kits was immediately conducted by the team. The team considered a number of options, including the WSB-9452, the Telum ASLP10, Lantronix’s Wi-Port Embedded System Device Server, X-Scale’s PC104 PXA255 and PXA270 CU DIMM embedded SBC modules, the CM-X270W Computer-On-Module, and the Triton-270. The team contacted the manufacturers and resellers of the selected development kits via telephone where possible and used email otherwise. The team obtained price quotes and made enquiries about product specification sheets with emphasis on benchmark tests and final throughput rates.

Attro’s WSB-9452 is an excellent single board computer that is equipped with a core 2 duo processor. However, the team decided that a development kit with a core 2 duo processor would not be an economic solution to the problem as only the wireless functionality was needed.(8) General Electric’s Fanuc Embedded systems’ Telum ASLP10 is a single board computer equipped with an Intel Pentium M processor. This device would be capable of achieving our processing goals, but due to the lack of immediately available spec sheets for download and lack of immediate technical support, the team decided not to pursue this option. (9) Lantronix’s Wi-Port
Embedded System Device Sever was also considered for its 802.11 b and g standards. However, closer analysis of the device spec sheets showed that the available output was serial CMOS (Asynchronous) which would result in a total throughput of 300 bps to 921,600 bps. (10)

Ultimately it was Digi System’s ConnectCore Wi-9C wireless single board computer that was selected for the project prototype. Equipped with Ethernet and wireless 802.11b capability, with up to 55 shared General Purpose Input/Output Ports (GPIO’s) the Wi-9C would conveniently connect to the selected DM-355 video processor chip. When running in synchronous master mode, the Wi-9C would provide a total throughput of up to 11.25 Mbps. The Wi-9C was also provided the convenient C based programming platform Net+OS and had a total cost of $499.99. (11)

7.1.3. VideoLan Client (VLC)

The project required some sort of video stream to send the data across the wireless protocol. Initially, we didn't set up a strategic plan for how we would create this video stream, however, as we were preparing our presentation for the business plan, the idea of using a mock display of the final device to sell the idea was brought up. VLC is very commonly used open source media player, making it easy to edit if needed, which already had a module that facilitated streaming of a PC screen with audio. This made VLC ideal software for this project. Additionally, VLC supports not only MPEG-4 transport streams, but MPEG-2, H.264, and a number of others. Though these features were not listed in our criteria they would could still be useful in making possible future improvements. For prototyping and final product design this seemed to be the best route since the functionality we desired already existed to some degree and the player facilitates a variety of transfer protocols and video compression algorithms.

7.1.4. Video Compression Algorithm

Because of the 11 Mbps bandwidth constraint of the 802.11b protocol, the video stream must be compressed to maintain quality. Uncompressed video runs up to 90 Mbps, visually lossless compression, that is compression algorithms that appear identical to uncompressed video runs in the 5-10 Mbps bit-rate range, and video conferencing is done between 384 Kbps and 1.5 Mbps. The Zigbee protocol would have been done in the video conferencing range, but use of the 802.11b protocol allows us to operate in the “visually lossless” range.

There are hundreds of video compression algorithms available, the team decided to select seven very commonly used compression algorithms. This was done because the most common compression algorithms will be most compatible with most equipment and software thus would be easier to work with. Seven video compression algorithms were considered. Selection criteria included compression ratio, suitability for real-time processing, legal requirements for using the algorithm, and the constraints of our development kit options. The seven compression algorithms considered were H.261, H.263, H.264, MPEG-1, MPEG-2, MPEG-4, and WMV-9. The following sections briefly highlight the advantages and disadvantages of each.
H.261, H.263, H.264
Volturna researched three of the H family algorithms. H.261 was designed for two-way communication (i.e. video conferencing). It provides data rates in multiples of 64 Kbps. H.263 is an improvement of H.261, specifically designed for internet video. H.264 is based on MPEG-4. It can achieve the same quality as an MPEG-2 compression at half the bit-rate. However, it is well covered by patents which would require legal work and licensing fees that would go beyond this project's allotted budget.

MPEG-1
The MPEG-1 standard is the first developed by the Moving Picture Experts Group. The algorithm allows data rates up to 1.5 Mbps and was designed for CD-ROM applications. It is not widely used today, although it does form the basis of the VideoCD standard used in Asia. MPEG-1 is best used for digital storage, where latency is not an issue.

MPEG-2
MPEG-2 is a higher bit-rate, more advanced version of MPEG-1. The MPEG-2 standard can provide a bit rate from 1.5 to 15 Mbps. This is the format associated with DVDs. Additionally, the MPEG-2 algorithm can compress interlaced video and is the basis of the digital television standard. With MPEG-2, one can achieve 5 to 30 times compression. Further, this standard is well suited to real-time applications.

MPEG-4
MPEG-4 is a video codec especially suited for web multimedia. It provides better compression than MPEG-2 under non-real-time conditions and is comparable for real-time video processing. Additionally, MPEG-4 allows variable bit-rate encoding, as well as constant bit-rate encoding which means it can be used in very-low as well as much higher bandwidth applications. This is the standard encoding for internet video streaming. The strongest drawback to MPEG-4 is that it is well covered by patents, meaning licensing issues if used in a commercial product.

WMV-9
The Windows Media Video (WMV) standard is a proprietary standard developed by Microsoft for their Windows Media Player. Although this is a comparable codec to others, it has a large drawback in that very few media players can encode and decode a WMV file. Further, use of WMV files on operating systems outside of the Windows system is difficult. To maintain transparency and keep the VWMP as platform independent as possible, the WMV standard was not selected.

Decision
Table 7-1 shows the design criteria considered by the team and how each alternative matches that criteria. The most important criteria to this project were image quality, compatibility with many platforms (operating systems, video players, etc.), and associated legal issues. Compatibility goes back to Volturna’s desire that the VWMP be
platform independent, and the emphasis on selecting an algorithm that was not heavily licensed relates to the goal of starting a business. Entering a market that is already highly regulated is difficult due to the need to pay licensing fees and draw up costly legal contracts. Ultimately Team Volturna decided that MPEG-4 would be the best suited compression algorithm for this project.

In Table 7-3 Image Quality refers to the number of pixel the compression algorithm yields from an input file. That is, the more pixels on an output image the compression algorithm can give, the higher its image quality. File Size refers to the output file size that a compression algorithm yields, this also affects the rates at which it must stream data. That is to say if a compression algorithm has a large file size it will require larger data rates for streaming. Compatibility refers to the if the compression algorithm can be read by the most commonly used media players such as VLC, the iTunes player, Windows media player etc. Efficiency here refers to the ratio of input to output file size of the algorithm. Resolution refers to the size of the maximum screen or display size that the algorithm’s output can occupy. Legal issues refer to if a license must be obtained to use the compression algorithm, if there is a fee, if it is open source and free or if it can be used free and legally through certain programs. For instance the WMV compression algorithm is strictly for use with the Windows Media Player program.

Table 7-1 Video Compression Algorithm Decision

<table>
<thead>
<tr>
<th>Decision Model</th>
<th>H.261</th>
<th>H.263</th>
<th>MPEG-1</th>
<th>MPEG-2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Criterion</strong></td>
<td><strong>Weight</strong></td>
<td><strong>Rating</strong></td>
<td><strong>Score</strong></td>
<td><strong>Rating</strong></td>
</tr>
<tr>
<td>Image Quality</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>File Size</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Compatibility</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Efficient</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Resolution</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>No Legal Issues</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>High Definition</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>20</td>
<td>14</td>
<td>43</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decision Model</th>
<th>MPEG-4</th>
<th>H.264 / MPEG-4 AVC</th>
<th>WMV 9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Criterion</strong></td>
<td><strong>Weight</strong></td>
<td><strong>Rating</strong></td>
<td><strong>Score</strong></td>
</tr>
<tr>
<td>Image Quality</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>File Size</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Compatibility</td>
<td>4</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>Efficient</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Resolution</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>No Legal Issues</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>High Definition</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>20</td>
<td>23</td>
<td>68</td>
</tr>
</tbody>
</table>
7.1.5. **Video Processing Development Kit**

The DSP decision is one of the most important for determining the functionality of the final prototype. This is because the DSP decision affects the final cost, power consumption, size of the receiver unit and the types of compression algorithms that can be used. Both the design criteria listed in
Table 7-2 and the team’s budget figured in the final decision. Research into available DSPs and development modules revealed three major options: Texas Instrument’s DM355 family, Analog Devices’ Blackfin family, and NXP’s nexperia family. Devices from any of the families would have provided the required design functionality. To make a decision, Volturna considered some of the differences between the families and discovered that the TI was a slightly better option than the others. In Table 7-4 Dev Kit Supported refers to the availability of technical support from the developer. MPEG-4 refers to the ability to decode data from the MPEG-4 format. AAC audio refers to the ability to decode data from the AAC audio standard. Cost refers to the cost of a dev kit. High Definition refers to the ability of the Dev Kit to output HD compatible video formats. Low power refers to the dev kit having low power consumption.
Table 7-2 DSP Decision Matrix

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Weight</th>
<th>TI - TMS320DM355</th>
<th>ADI - Blackfin</th>
<th>NXP - PNX15xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tech Support</td>
<td>4</td>
<td>4 16</td>
<td>4 16</td>
<td>1 4</td>
</tr>
<tr>
<td>MPEG-4</td>
<td>3</td>
<td>4 12</td>
<td>3 9</td>
<td>4 12</td>
</tr>
<tr>
<td>AAC Audio</td>
<td>2</td>
<td>3 6</td>
<td>2 4</td>
<td>4 8</td>
</tr>
<tr>
<td>Dev Kit Cost</td>
<td>3</td>
<td>3 9</td>
<td>4 12</td>
<td>3 9</td>
</tr>
<tr>
<td>High Definition</td>
<td>3</td>
<td>4 12</td>
<td>3 9</td>
<td>4 12</td>
</tr>
<tr>
<td>Low Power</td>
<td>1</td>
<td>3 3</td>
<td>4 4</td>
<td>3 3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>16</strong></td>
<td><strong>21 70</strong></td>
<td><strong>17 66</strong></td>
<td><strong>19 60</strong></td>
</tr>
</tbody>
</table>

The most important criteria for the team was the existence of an appropriate DSP development kit, as well as easy available technical support from the manufacturer. The capability to encode into the NTSC form as well the ability to connect to the selected 802.11 kit were also taken into consideration but not placed in a decision matrix because all the boards were capable of doing these things. Furthermore, the costs of all the actual DSP chips, not including their development kits were all below $20, which in light of development kit prices the team decided could be overlooked for a prototype design. TI’s DM355 had this as well as ADI’s Blackfin, but the team was unable to locate a development kit for NXP’s PNX15xx series. Since Volturma does not have the time nor technical resources to effectively use a ball-grid array (BGA) or other chip without the development kit, the NXP was eliminated from the options. The DM355 development kit (TMDXEVM355) cost $640, while the Blackfin development kit (ADZS-BF533-EZLITE) costs approximately $450. Both were within the team’s budget. The PNX15xx series fully supported MPEG-4 video and accompanying AAC audio along with the DM355. The Blackfin family appears to support these as well, but not explicitly MPEG-4. Also, it was unclear how the ADZS-BF533-EZLITE utilized GPIO inputs, while the TMDXEVM355 provided clear documentation. Both DSPs support RCA connectors with NTSC format, though Texas Instruments’ DM355 was advertised specifically to support H. Though for the purpose of this project was not a requirement but was certainly a complimentary feature for possible future improvements. Processor power consumption was considered, but it was secondary to other decision criteria. Using
Table 7-2 as a guide, the choice was made to purchase the TMDXEVM355 evaluation module (Figure 7-1) for Texas Instrument’s DM355 DSP from Newark Corporation.

![TMDXEVM355 Evaluation Module](image)

**Figure 7-1 Texas Instrument's TMDXEVM355 - DM355 Evaluation Module(12)**

### 7.2. System Architecture and Design Summary

The prototype design for the wireless digital media system, outlined in the project requirements, is different than the intended final goal of a commercial off-the-shelf product outlined in the team’s business plan. This is primarily due to the constraints of development. For one thing, the purchase of new high bandwidth, high fidelity Wireless USB hardware remains well out of reach of the team’s budget. While low-cost components exist to manufacture such a product, the cost of development hardware and tools is prohibitive. The final prototype therefore uses the 802.11 protocol and since the design for an 802.11 transmitter is trivial, the team focused efforts on the receiver and on formatting the transmission data stream.

The overall vision of the system contains three components – a host PC with a wireless 802.11 card, a VWMP connected to the PC via a wireless router, and a display device such as a TV which will be wired to the VWMP using RCA A/V cables. The system will transmit both audio and video, and will do so by mirroring the contents of the PC’s screen on the display device.

Transmission from the PC to the VWMP is done via the VLC open source media software and work on that consists of modifying existing code to meet our needs. VLC takes the data on the desktop display, wraps it into an MPEG4 transport stream (TS) including both audio and video, and places the MPEG4 TS inside a User Datagram Protocol (UDP) Ethernet packet. It then sends the data over the wireless Ethernet network to the recipient server, or to a multicast Internet Protocol (IP) address to a client listening to that IP address.
The receiver is more complicated and consists of a wireless receiver and media processor. Two separate development kits are used due to the difficulty of finding one low-cost solution that has a digital to analog converter (DAC), NTSC encoder, MPEG-4 decoder, and wireless 802.11 capability. The first development kit is the Digi ConnectCore Wi-9C which has 802.11b capability. The ConnectCore receives the data stream in packets and unwraps the UDP, IP, and Ethernet headers from the data. The result is an MPEG4 TS packet which is split into 13 bit chunks and transferred to the TI GPIO pins. Before sending the TS packet, the ConnectCore communicates the length of the packet to the DM355 which eliminates the need for feedback and allows us to use a simple rising edge flag to signal the TI that new data is available on the GPIOs.

The TI DM355 digital processor reads the data via its development kit’s GPIO ports and recompiles the original MPEG-4 TS. It then passes this to its MPEG-4 co-processor which decodes the audio and video and hands it off to the DM355 backend. The backend consists of an NTSC encoder and a DAC. The video and audio are converted to analog and the video is encoded using the NTSC encoder. The DM355 outputs the signal using RCA A/V component jacks.

This prototype meets the criteria for function, but does have drawbacks as far as marketability and transparency. Using 802.11 requires the customer to either have an existing wireless network or buy a separate wireless router. This increases the complexity of the device for the consumer and reduces transparency.

8. Implementation

A router will be used to establish a Wireless Local Area Network (LAN). The router will then assign an IP address to the host PC and the Wi-9C. Using the open source software for the VLC player, the host computer will stream audio and video over a UDP stream to the static address assigned by the team to the Wi-9C. If the host PC does not have an on-board wireless card it will require the separate purchase of one.

A program written by team Volturna for the Wi-9C will serve 2 main functions, the first being to act as UDP Server, reading data in from the host PC’s UDP stream and secondly it will output the UDP stream data to the selected GPIO’s. This program is described in section 8.1.

The DM-355 development kit and Wi-9C will be connected via their GPIO’s. The specifics of this interface are described in section 8.3. The Wi-9C has 55 shared GPIOs and the EMV-355 has 14 free GPIO’s. GPIOs 25 to 38 on the Wi-9C are by default assigned to LCD functions. Using the UDP Server/GPIO output program written by team Volturna these will be re-assigned to output data from a UDP stream. The DM-355 will then convert the data received from its GPIO’s to the analog audio and video forms and output them via its RCA audio/video ports. This process is described in section 8.4.
8.1. Wireless Send

On the host PC the user would install a copy of the VLC player version (0.8.6+) or newer. The VLC player is free, open-source software available for download at http://www.videolan.org/.

Once the player has been installed on the host PC the user would open the VLC player program as shown in Figure 8-1.

![Figure 8-1 Opening the VLC Player in Windows XP](image_url)

The user would then click on the “File” drop-down menu, and select the “Open Network Stream” Option as shown in Figure 8-2.
Figure 8-2 Opening the "File" Drop-down menu and selecting the "Open Network Stream" option

In the “Customize” tab at the bottom of the tab that was just open insert the following code:

```
vlc screen:// :screen-fps=30 :screen-caching=100
:sout=#transcode{vcodec=mp4v,vb=2048,scale=1,acodec=mpga,ab=192,channels=2}:duplicate{dst=std{access=rtp,mux=ts,dst=192.168.1.3:10001}}
```

This functionality of this code will be explained briefly. The inserting of the code into the “Customize” tab is illustrated in Figure 8-3.

Figure 8-3 Inserting the code into the "Customize" tab in the VLC Player
The user would then click the “OK” button to start streaming to the desired Static IP address 192.168.1.3

In the above code the code that was entered into the “Customize” tab, the “screen://” command selects the desired input module, in this case the host PC screen. The “:screen-fps=30” command selects a 30 frames per second refresh rate which complies with NTSC video standard. The “:screen-caching=100” sets the internal caching to 100ms from the default 300ms setting, this tells the host PC to store a new snapshot of the screen into cache memory every 100ms. The “--sout” command tells the VLC the images will be output. The “#transcode{}” command tells the VCL program to transcode the input of screenshots to the parameters placed in the “{}” field.

Inserted into the “{}” parameter field, the “vcodec=mp4v” sets the video codec to MPEG-4 video. This is very convenient because by changing this field the output data stream can be placed in other compression algorithms such as MPEG-2 or any of the many other algorithms that the VLC player is compatible with. It should be noted that this only sets the output form of our stream; the end result of this code will still display the entire screen content ultimately displaying anything that can be played on the host PC screen.

Using this method the design product would truly be independent of the codec of the video the user decides to transmit. The “vb=4096” command sets the bit-rate of the transcoded video to (4Mb/s) which is within the limits of the prototype designs Wi-9C board throughput rate as well as the product designs Wireless USB throughput rate. The “acode=mpga” command merely sets the placeholder audio codec to mp3, however this line will need to be modified to transmit audio from the sound card. The “ab=256” sets the bit-rate of the transcoded audio to 256 Kb/s. The “scale=1”, “width=1280” and “height=800” commands set the transcoded video to 2048x1536 resolution.

The “:rtp” command tells the VLC player that output video from its encoder will be streamed. That is, it will use the Real Time Protocol (rtp). The “dst=192.168.1.3” command sets the destination Static IP address to the desired Wi-9c board address of 192.168.1.3. In the case of the design product this would be modified to output to the desired serial com port number that the USB transmitter device would be plugged into. The “port=10001” sets the destination port number for the Wi-9C board, this would also be modified in the case of the design product. The “access=udp” tells the VLC player that the data will be sent using the UDP protocol. The “mux=ts” tells the VLC player that the data will be streamed using MPEG-2 Transport stream, this is not to be confused with the MPEG-4 video compression being used.

**8.2. Wireless Receive**

A schematic of the ConnectCore Wi-9C development platform is show in Figure 8-4.
To implement the design, serial port A was used for debugging (highlighted in red) and the LCD header pins (blue) were used for interfacing with the TI board. The schematic of the ConnectCore Wi-9C itself is shown in Figure 8-5.
A JTAG debugger supplied with the development board connected the JTAG header outlined in green on Figure 8-5 via a ribbon cable to the USB port on the PC used for debugging. The ARM9 processor on the Digi ConnectCore came preloaded with Digi’s proprietary Net+OS operating system. Digi’s ESP development software was used to develop programs for the board. The ESP software is based on Sun’s Eclipse development environment. The board can be programmed in either the C or C++ language. This project was done primarily in C.

The Net+OS system has an existing 802.11 internet protocol (IP) stack. The software written for the Wi-9C monitors the stack and filters out User Datagram Protocol (UDP) packets. UDP packets are discrete blocks of data and require no handshaking between sender and receiver. The sender is not assured of packet reception. This protocol was deemed sufficient for this application since in streaming, speed is preferable over reliability. That is, if a packet is not received or has an error detected upon being received, the Wi-9C should ignore it and continue receiving the next packet. UDP packets consist of a packet header and the data as shown in Figure 8-6. If the datagram is originating with the host PC as in this application, the source address will be the host PC’s IPv4 address, and the destination will be the static IPv4 address assigned to the board. This address was set to 192.168.1.3. The 8-bit protocol needs to be the hexadecimal value 0x11 which represents the UDP protocol. The length of the header and data is determined at the time of sending and this computation is done by the VLC streaming software. The checksum is optional for the IPv4 suite and was neglected in the receiver since detection of an error will not change the ConnectCore software operation. The data consists of the MPEG-4 encoded video and audio from VLC.

<table>
<thead>
<tr>
<th>IPv4 Source Address (32 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 Destination Address (32 bits)</td>
</tr>
<tr>
<td>0 (8 bits)</td>
</tr>
<tr>
<td>Source Port (16 bits)</td>
</tr>
<tr>
<td>Length of UDP header and data (16 bits)</td>
</tr>
<tr>
<td>Data</td>
</tr>
</tbody>
</table>

Figure 8-6 Internet Layer (Green), Transport Layer (Orange), and Application Layer (Blue) of a UDP Datagram

The ports used for the UDP protocol were hardcoded to avoid the problem of either the host PC or ConnectCore being unaware of which port the other was using. The process of preparing ports for UDP transfer is as follows:
1. Open datagram socket
2. Allocate the receive and/or send buffer (depending on application)
3. Initialize the socket structure
4. Bind an IP address to a port

A sample of the code written to perform these operations is shown below.

```c
char *dataWlnGramBuffer; //Create send/receive buffer for wireless datagram socket
int sockWlnGram, rcvWlnAddrSizeGram; //socket and address size
struct sockaddr_in rcvWlnAddrGram; //address structure for socket
NaIamIfAddrInfo_t wlnAddrInfo; //address information structure

//check that wireless interface has IP addresses on IPv4
if (customizeIamGetIfAddrInfo("wln0", &wlnAddrInfo) != NA_IAM_STATUS_SUCCESS ||
    wlnAddrInfo.v4Info.IPV4_ADDR(ipAddress) == 0) {
    printf("naBypassApp: IPv4 address is not configured on wln0\n");
    return;
}

// open datagram socket for wireless port, AF_INET specifies this is IPv4, SOCK_DGRAM
// specifies this is a datagram socket and the ‘0’ tells the socket to figure out which
// protocol it is using on its own. If sockWlnGram < 0, an error has occurred
if ((sockWlnGram = socket (AF_INET, SOCK_DGRAM, 0)) < 0) {
    printf("naBypassApp: wireless datagram socket creation failed, errno = %d\n",
            getErrno());
    return;
}

//allocate buffer for receive
dataWlnGramBuffer = malloc(APP_BUFFER_SIZE);
if (dataWlnGramBuffer == NULL) {
    printf("naBypassApp: wireless datagram buffer allocation failure\n");
    return;
}

//bind wireless datagram socket
memset((char *)&rcvWlnAddrGram, 0, sizeof(rcvWlnAddrGram)); //clear memory at rcvWlnAddrGram
rcvWlnAddrGram.sin_family = AF_INET; //set address family
//set actual IP address
rcvWlnAddrGram.sin_addr.s_addr = htonl(wlnAddrInfo.v4Info.IPV4_ADDR(ipAddress));
rcvWlnAddrGram.sin_port = htons(DGRAM_WIRELESS_PORT); //set port

//bind socket
```
resultWlnGram = bind(sockWlnGram, (struct sockaddr *) &rcvWlnAddrGram,
   sizeof(rcvWlnAddrGram));

if (resultWlnGram < 0) {
  printf("naBypassApp: bind call for wireless failed, errno = %d\n", getErrno());
  return;
}

//set the size of the wireless address structure
rcvWlnAddrSizeGram = sizeof rcvWlnAddrGram;

The socket is now set up to listen on the port DGRAM_WIRELESS_PORT. We used a
DGRAM_WIRELESS_PORT value of 10001.

Having successfully set up and initialized the UDP ports that will be used to send and receive stream datagrams
from VLC and communicate with the host PC, the next step is to set up a wireless network. Initially the team
wanted to create an ad-hoc network between the ConnectCore board and the host PC. However, after significant
trouble connecting to and setting up the ad-hoc network, the team contacted a Digi technician via phone who
said that ad-hoc mode is not recommended for the ConnectCore as there are still many bugs in the code. As a
result, Volturna opted to pursue a wireless-LAN model. A NetGear WGR14 Wireless router was used to set up a
wireless LAN named “Volta”.

Figure 8-7 Netgear WGR614 Wireless Router(14)

A laptop running a freeware UDP program called “Hercules” put out by the HW Group was used for debugging.
The Hercules software allows a user to set up a UDP client and send and receive data with a server. In this case
the ConnectCore functions as a UDP server. To send and receive data, the UDP thread enters an infinite loop
which continuously checks the WIRELESS_DGRAM_PORT for data. This is done using the recv function
specified in the API Reference provided with the ESP software. The syntax for this function is

  int recvfrom(socket, buffer, length of buffer, flags, pointer to socket structure for ‘from’
  address, length of from address structure);

58
The recvfrom methods return an integer. A negative return value indicates there was an error. A value of 0 indicates no data was available. Positive return values indicate the number of bytes either received or sent. In pseudo-code the infinite loop looks like this:

```c
for (;;) {
    bytesReceived = recvfrom(wireless_socket, receive_buffer, buffer_length, 0, (struct *)
                               receive_address, sizeof (receive_address));
    if (bytesReceived > 0) {
        sendtogpio(receive_buffer, bytes_received);
    }
}
```

The sendtogpio(char * receive_buffer, int bytes_received) function is a custom function called whenever a new datagram is received. It takes the first bytes_received bytes of receive_buffer and sends those to a new buffer which handles the transfer of bits to the GPIO pins.

The gpio function does the following

1. Adds bytes_received to local stack (first-in-first-out, thus MUST be an 8-bit number)
2. Adds receive_buffer to local stack (first-in-first-out)
A GPIO thread handles the setting of GPIO pins via the following formula

1. for (; ;) {
2. if (gpio_buffer is not empty) {
   a. bytes = top byte of local stack
   b. set GPIOs = bytes
   c. set signal bit to 1 for TI
   d. pop first byte off local stack
   e. bits = bytes*8
   f. wait if necessary
   g. set signal bit to 0
   h. for (bits > 0) {
      i. set GPIOs = top 13 bits of local stack
      ii. set signal bit to 1 so TI knows signal is ready
      iii. pop off top 13 bits of local stack
      iv. bits = bits – 13
      v. wait if necessary
      vi. set signal bit to 0
   i. }
   j. set GPIOs all to 0
3. }
4. }

The GPIO thread will execute this loop for as long as the board is on and running. When the GPIO buffer is empty, the default is to set all GPIOs to 0. The TI only reads the GPIOs on the rising edge of the signal bit. The maximum rate for the GPIOs is limited at 10MHz as that is how fast the TI can read them. The ‘wait’ functions prevent the GPIO data from being replaced too quickly.

8.3. Digi – to – TI Transfer Protocol

The transfer between the Digi and TI boards is done via GPIO ports. The TI has only 14 ports so it is the limiting factor. To speed transfer, all 14 ports are used. Thirteen ports (bits) are used to transfer data, and the remaining one is used as a signal to the TI board that new data is available for reading. Each time the Digi board receives a packet, it extracts the data and initializes a transfer with the TI board by placing the length of the packet in bytes on the thirteen data pins, and setting the signal pin to high. The TI board reacts by reading the length and allocating a space on its buffer for a packet of that length. The Digi board then begins transferring the packet in thirteen bit chunks beginning with the least significant bits first. Each time a chunk of the packet is sent, the signal bit pulses high long enough for the TI to recognize the signal.
Table 8-8 Board Interconnect Description

The TI only reads data on the rising edge of signal. If the final chunk is less than 13 bits long, the most significant bit positions are padded with zeros. The TI board knows to throw these out because it keeps track of the length of packet it is looking for and how many bits it has received at all times. The TI recognizes a transfer complete each time it receives a packet of the expected length. The Digi recognizes a transfer complete by setting the signal pin to zero and leaving it there until a new packet is ready to be sent. If packets arrive faster than the TI board can read them and there is not enough space in the Digi’s buffer, the extra packets are dropped. The team reasoned that since there is already a level of unreliability with the UDP protocol, dropping additional packets would not cause significant quality issues. If the packets are regularly arriving too quickly for transfer, the VLC code will be modified to either send smaller packets or send packets less often. This is done by changing the frame rate and raw packet rate in the VLC code.

8.4 Video Decoding and Encoding

The TI DM355 development was done using the MontaVista Linux development environment which is designed for and came with the TI DM355. This requires a PC running a Linux operating system which led us to use Charlie’s laptop which was already running the Sabayon Linux distribution for the code development. The debugger system also requires a serial port for debugging and a network connection to for the DM 355 to retrieve the root NFS filesystem. Charlie’s laptop did not have a serial port so a Windows PC had to be used for running the code in a console window using a HyperTerminal client. This is shown in Figure 8-9 below.

Figure 8-9 Development Environment Overview

In order to retrieve data sent from the Wi-9c, the TI DM355 uses data input output pins on the DC7 pin header, which provides 14 GPIO pins. The DC7 pin header is located on the area indicated in Figure 8-10 below.
In order to interact with the data on the DC7 pin header, software from the DM355 EVM Support forum target content software references the memory locations found in the evmdm355_gpio.h. Using these memory references, the information on the input output pins becomes available to the software. The software initializes these input output pins by calling the functions `EVMDM355_init()`, found in the evmdm355.c file and `EVMDM355_GPIO_init()`, found in the evmdm355_gpio.c. The data on each individual pin can be retrieved then by calling the function `EVMDM355_GPIO_getInput(Uint16 number)` with the number of the pin as the argument (Uint16 number). Pin numbers are shown in Table 8-1.
Table 8-1 DC7 Pin Header Information

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Signal</th>
<th>Pin #</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GIO54</td>
<td>2</td>
<td>GIO67</td>
</tr>
<tr>
<td>3</td>
<td>GIO65</td>
<td>4</td>
<td>GIO31</td>
</tr>
<tr>
<td>5</td>
<td>GIO63</td>
<td>6</td>
<td>GIO64</td>
</tr>
<tr>
<td>7</td>
<td>Ground</td>
<td>8</td>
<td>Ground</td>
</tr>
<tr>
<td>9</td>
<td>GIO62</td>
<td>10</td>
<td>GIO61</td>
</tr>
<tr>
<td>11</td>
<td>GIO60</td>
<td>12</td>
<td>GIO59</td>
</tr>
<tr>
<td>13</td>
<td>GIO58</td>
<td>14</td>
<td>GIO57</td>
</tr>
<tr>
<td>15</td>
<td>GIO56</td>
<td>16</td>
<td>GIO032</td>
</tr>
<tr>
<td>17</td>
<td>Ground</td>
<td>18</td>
<td>Ground</td>
</tr>
<tr>
<td>19</td>
<td>VCC_3V3</td>
<td>20</td>
<td>VCC_3V3</td>
</tr>
</tbody>
</table>

A function return of ‘1’ indicates that the signal is high, whereas a return of ‘0’ indicates a low.

Data is received in chunks through the pins and when a new data packet is received, the software recognizes the first packet as containing the number of bytes in the total packet. This number is stored in a counter. The software then stores the subsequent chunks into the GPIO packet buffer starting from the least significant bit of the packet. When the number of bits in the counter matches the number of total bits received, the TI flags the packet as complete. This system does not take into account lost packets at this time. At this point, the entire GPIO packet buffer is written to the MPEG4 packet buffer. Assuming a frame rate of 25fps, and the GPIOs operating at the maximum 10MHz, the TI will be able to receive 400 13-bit chunks of data per frame. This amounts to 650 bytes, although the first 13 bits will be the packet size, so only 648 (the floor of 650-1.5) bytes of data can be received. At a 20 fps datarate, 811 bytes can be received per frame. Once the GPIO packet buffer receives the data, our software would parse the packet from its MPEG4 encoded scheme into its audio and video components, and place it on an audio buffer and video buffer respectively. Figure 8-11 below shows the structure of the system.

![DM355 Audio/Video Buffer Structure](image_url)

Figure 8-11 DM355 Audio/Video Buffer Structure
In this figure, the display buffer represents all of the information that will be sent out to the user including both audio and video data.

The basic software thread of the DM355 video processing system used by the encode/decode software demo program which can be found in Appendix 0 follows the general scheme shown in Figure 8-12.

Once the software begins, the main thread will start to place the data from the GPIO pins onto the display buffer as indicated earlier. This section of the code would be written by the team and interact with . This display buffer is read during the display thread of the software. Once the data is in the display thread, it is then sent to the MPEG4 decoder which processes the data. Data is retrieved from the display buffer in approximated sets of what is called a frame, which is the amount of data that can be displayed on the screen at any given time. This is done through a file loader algorithm. This algorithm pulls in roughly one frame of data from the display buffer every time it finishes displaying a frame. This allows the system to always have the next frame of data available in memory before it is to be displayed on the system. The basic structure of the frame memory stack is shown in Figure 8-13.
Figure 8-13 DM355 Frame Stack(16)

The frames are then sent to the decoder algorithm provided with the MontaVista development environment, which decodes the MPEG4 frame. The MPEG4 decoder then takes the decoded frame and sends it through to the NTSC video processing unit.

9. **Results**

9.1. **Testing**

In order to ensure that the design is working properly, the team would use an incremental testing scheme. As parts are completed, the design of those pieces would be tested for accuracy with the specifications of each part. As a result, the testing would begin by ensuring all code is working by testing for compilation and by debugging through it to ensure that the proper processes are starting. Further debugging is required for testing which calls to examine buffers.

The first of the hardware testing will be done on the wireless board to ensure that the UDP protocol is correctly being decoded. This consists of sending a UDP packet and ensuring that the data we receive back is the same as the data sent. In order to ensure this is done, we would use the wireshark Ethernet protocol program to verify the data sent and compare that to the resulting data on the memory registers.

The second part we would check to ensure that it was working properly is the GPIO size measurement output on the Wi-9c. We would check the output using a protocol analyzer and if the size of the bytes of the protocol matches the number we receive out on the system’s GPIOs then we have validated this step.

Once this step is complete, we can begin testing to ensure that the GPIO pins are outputting the entire packet and correctly. The best way to do this is to test the GPIO input on the TI DM 355 first and ensure that that is working properly. To test the TI DM355, we would use a 3.3V power supply and place it on the pins of the DM355’s DC7 interface and compare those to the results we had in software. Secondly, we would check to see
that the software is returning the correct number of bytes. We can then set the number of bytes to a relatively small number, such as 8 bytes which will ensure that there will be multiple strings sent. Once we have signaled a byte number by setting the most significant bit high, we can then record the data we send and ensure that the software on the DM355 only counts the total number of bytes we sent it and then begins again. Once we have verified this behavior, we can then connect the Wi-9C and the DM355 together.

With the DM355 and the Wi-9c connected, the next step is to send a single UDP packet. When the Wi-9c receives this packet, it should send the packet to the DM355 and the contents of the UDP packet should be placed on the GPIO buffer. We can then write the GPIO buffer to a file and compare the output of the packet to the encapsulated data reported by wireshark.

If the encapsulated data reported by wireshark is the same, we must then test the MPEG4 packet buffer. To ensure that we are getting the correct data into the MPEG4 packet buffer, we can compare the output of the MPEG4 packet buffer to the entire encapsulated packet data as reported by wireshark. If these two match we then proceed to testing the audio and video splitting algorithm.

To ensure that the splitting algorithm is working properly, we would attempt to send just audio from the Video Lan Client. If the audio buffer fills and the video does not, we have success so far. If we then test the video portion by sending simply video and only the video buffer fills, we can then conclude that the video split alone is successfully working. Once we have tested these two components individually, we can then test them combined. In order to do this, we would send an audio and a video file from VLC. If both the audio and video are filling completely, it is likely that we have success. A potential pitfall here is that there may be potential overlap of the audio and video signal, causing incorrect data to be written. If the same audio and video files are used to check the parts as to check the whole, the data written on the individual buffers can be checked to verify if it is the same as the data as was written in the single stage.

At this point, the best way to test the output would be to see if the display is correctly showing the video. If the display does not correctly show video, check to ensure that the data being sent to the entire system is correct, then check the video and audio display buffers again. If this is not working still, we would check to ensure that the display is still working with the sample data.

9.2. Design Analysis

The current design for our system would work well for use by a technically knowledgeable person. At its current state, it requires that users are aware of what an IP address is and how to set and find one. If we were to do this again, we would make sure to use a multicast IP address. By using a multicast IP address, the design would be much easier to detect using a computer software program, and thereby allow easy usage for the non-technologically inclined population. The design overall could be vastly improved were we to integrate the DM355 chip and a wireless network chip onto the same board. This would eliminate the need for using external
GPIO pins and also eliminate the GPIO packet buffer as the UDP protocol stack could be decoded from the wireless network buffer stack and then placed directly onto the MPEG4 stack.

Using the testing strategy above as a guide for the test driven development of our prototype unit, the team was initially unable to successfully wirelessly broadcast UDP data between the host PC and the Wi-9C. The team then attempted to do this over an Ethernet broadcast; this too was unsuccessful. The team then further backtracked and attempted to broadcast data over Ethernet between the host PC and the Wi-9C using an FTP (File Server Protocol) server. This was successful.

With the lessons learned from using Ethernet to broadcast data using an FTP, the team then progressed to broadcast data wireless using an FTP. This too was eventually successful.

The next step was to attempt broadcasting UDP data between the host PC and the Wi-9C again. This still proved to be problematic. The team then acquired the commonly used open source software programs ServerTalk and Hercules which would setup UDP chat servers on the host PC and another PC which would be used in debugging. ServerTalk and Hercules are shown in Figure 9-1 and Figure 9-2 respectively.

![Figure 9-1 ServerTalk UDP chat program](image)
Using these programs to debug the UDP streaming code, the team was able to debug to the point that the Wi-9C successfully received the UDP data as it was supposed to. Unfortunately within the allotted time the team was unable to send the UDP data received from the host PC to the output of the GPIO pins of Wi-9C.

The DM355 the team was able to successfully encode a file into the MPEG-4 format using code provided with the DM355. The team was then able to successfully access and decode this file using the DM355 wired directly to the PC via serial cable and to output that file successfully to the RCA A/V ports of a TV.

The team was further able to wirelessly access the encoded file on the host PC wirelessly and output this to the RCA A/V ports of a TV. This however, required the use of a router.

10. BUSINESS AND PRODUCTION

10.1. Business Plan

Since entrepreneurship was a central focus of Volturna’s project, a business plan was created. This plan was submitted to the Great Lakes Entrepreneurship Quest, a Grand Rapids area entrepreneurship competition, the Calvin bizPlan 2007 competition, and the NCIIA grant committee along with the team’s grant request. The following pages contain this business plan.
10.2. Competition Results

Volturna won the Calvin College bizPlan 2007 competition. These winnings provided the bulk of the team budget. Volturna did not win the Great Lakes Entrepreneurship Quest or NCIIA grant request.
Confidentiality Agreement:

This message is intended exclusively for the Calvin College Senior Design Course. This communication may contain information that is proprietary, privileged, confidential or otherwise legally exempt from disclosure. If you are not affiliated with the Calvin College Senior Design Course, you are not authorized to read, print, retain, copy or disseminate this message or any part of it. If you have received this message in error, please notify the sender immediately and delete all copies of this message.
Executive Summary

Volturna – The Company

Volturna seeks to create a product that will enable computer users to quickly and easily send their multimedia to a television. This technology eliminates the need to connect a mess of cables in order to display a computer screen directly to a TV and makes the process as easy as a few clicks of a mouse. While there are currently similar products on the market, these often require an oversized box connected to a computer. With our solution, nearly all future laptops will connect automatically, and current computers require a device as small as a thumb drive.

Management

Management Team

Charlie Reitsma, CEO of Volturna, currently attends Calvin College and will graduate in May of 2008 with a Bachelor of Science in International Engineering and a emphasis on business management. His past work experiences include implementation of web-based lab management software, design and installation of a $30,000 door entry system and oversight of home remodel projects. Gwendolyn Einfeld, the Vice-President of Volturna, is an undergraduate student at Calvin College. She intends to graduate in May of 2008 with a degree in Engineering with an Electrical and Computer Concentration. Gwen works as an employee for the Center for Social Research at Calvin College, where she does analysis of social trends. She recently spent time working on real-time disk scheduling at the University of California, Santa Cruz. Leah Einfeld acts as the Financial Chair of Volturna and plans to graduate from Calvin College in May 2009 with a Bachelors degree in Economics and minors in Business and French. She currently works as the Account Financial Assistant for TrimQuest. Richard Kwakye acts as Lead Design Engineer. He plans to graduate in May of 2008 with a Bachelor of Science in Engineering with and Electrical and Computer Concentration. He has previously worked for Byce and Associates as an IT technician and Electrical Engineer. Ted Worst joins the team as Internal Operations Manager. He currently studies towards a Bachelor of Science in Engineering at Calvin College, and has previous experience with quality assurance at GE aviation.

Advisory Board

Dr. Steven H. VanderLeest is Calvin College’s Engineering Department Chair and works a consultant in safety-critical embedded systems design for DornerWorks Limited and acts as a business and technical mentor. Tim Theriault, a Senior Management and Strategic direction Engineer at General Electric (GE) Aviation, acts as a technical consultant. Paul Holtrop works with Volturna to create business strategy. Paul works for Impact Management as a consultant and has led roles in materials, logistics, and production management. Andrew DeVries assists Volturna in creating marketing strategy. He has spent 15 years as Director of Marketing and V.P. of Sales for U.S. Exchange.

Market Analysis

The market for this type of product is in its infancy. Currently, laptop and device manufacturers are focusing on the technology behind our design. This will enable future computers to interact directly with the Volturna media receiver. Based on preliminary market surveys, the target market for this device spans to people from 16-60 years old. Current designs similar to the design we propose are offered at nationwide retailers such as Home Depot. These designs often range from $100-$180. Table 1 shows the preliminary market survey results.
In order to protect its ideas as well as its intellectual property, Volturna hopes to patent any relevant designs, as well as maintain confidentiality with producers.

Similar products to our initial target design currently are on the market, but Volturna's offering simplifies the connection and will make it easier for use with a laptop. Current devices such as the FlyImage 5000 and SmartHome's 7743B require bulky connections to your computer. Volturna's design will use a device the size of a USB key fob. Dell and Lenovo have started putting the technology in their laptops as well, and most laptops will have the technology in the next few years, making Volturna's device much easier to use than both the FlyImage and SmartHome versions.

Product

Basic Product Description

Our product will ease the connection to display your computer screen directly on your TV. Current products are cumbersome and hard to use, and require a large unit attached to both the computer and the TV. Current products on the market such as the FlyImage 5000 and

![Figure 1](image)

SmartHome's 7743B require large units as well as multiple cables connected from the computer to the unit. Volturna’s Wireless Receiver connects through just a small device directly to the computer without any cables. This allows the technology to be portable as well as easy to use. Figure 1 illustrates the device and connections.

<table>
<thead>
<tr>
<th>Category</th>
<th>Respondents Interested (11)</th>
<th>Respondents Interested (20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watch movies on PC</td>
<td>100%</td>
<td>70%</td>
</tr>
<tr>
<td>Watch TV on PC</td>
<td>100%</td>
<td>80%</td>
</tr>
<tr>
<td>Watch home videos on PC</td>
<td>95%</td>
<td>85%</td>
</tr>
<tr>
<td>Use PC for presentations</td>
<td>60%</td>
<td>90%</td>
</tr>
<tr>
<td>Use PC for slideshows</td>
<td>64%</td>
<td>70%</td>
</tr>
<tr>
<td>Other Use</td>
<td>9% (Games)</td>
<td>5%</td>
</tr>
</tbody>
</table>

The market for this portable display technology is expected to be worth over one billion dollars within the next five years. With over 20 million unique users a day visiting video hosting sites such as YouTube.com, the market for video watching devices continues to grow, and based on our market survey results, those who watch movies on their computers show strong interest in this product.

Our initial target market based on our survey data is users who watch movies, television and home videos on their computers. In order to target this market, we intend to place ads with popular video hosting sites such as YouTube.com and Yahoo! video. This will put our product directly in front of our market’s eye. Additionally, we will participate in consumer electronics trade shows to demonstrate our product to a wider consumer audience.

The current market is ripe for this technology. While in the past, computers had little wireless connections, the laptop computers of today and tomorrow connect to everything from a cell phone to the internet. The newest laptops and computers are starting to add a new technology called Wireless USB that will enable easy connections to devices without attaching them to a computer. By using the capability of this technology in its infancy, Volturna will capture a large amount of the market share for this specialized product before its competitors have the capability to. This will give the company a deep foothold in the market and allow further expansion and innovation with this new technology.
Additionally, the latest laptops from Dell and Lenovo have the ability to use the Volturna Wireless Receiver without an external transmitter and most computer manufacturing companies are working towards incorporating this technology in future products.

**Business Model**

**Income Generation**

In order to ensure income for Volturna, we intend to outsource product manufacture and sell goods to customers either through consumer electronics vendors such as Circuit City and Best Buy or direct to consumers through a company website. By producing the product for electronics vendors, the company would be paid on a contractual basis for the goods in large quantities. The consumers would handle all of the sales aspects regarding the product. By selling the product direct to consumers, the company would except payment by credit card to ensure prompt payment to the company.

**Potential Customers**

The customers for this design include computer video watchers in the age range from 16-60 of any ethnicity. These consumers have a computer and often a high speed internet connection. The vast majority of Volturna’s customers fall within the middle to high income range. Customers who fall in the high income range have high definition home theater systems, and require high quality pictures.

**Opportunity**

Alternative methods of generating income include licensing designs to manufacturers and assisting other design firms to develop products based on Wireless USB.

Volturna expects an initial growth rate of $200,000 in its first year, reaching to $1 million in its third year of operations.

**Competition**

**Competition Analysis**

Current market competitors include Flystream, AraneusUSA, and SmartHome. The SmartHome 7743B currently enjoys a large corner of the market, and can be found through Circuit City. This company receives around 50% of the revenue of this type of product. The SmartHome product requires a large connection box, and has low picture quality and interferes with cordless telephones. It typically costs around $160. The AraneusUSA product is the most viable product on the market and offers the best ease of use and highest overall customer satisfaction, but is also the most expensive of the units with a price of around $180, and still requires connection cables between the computer. The Flystream compares to the SmartHome in regards to quality and accounts for about 25% of the market share.

The current advantage over the competitors is that the device will not require users to connect a large box to their computers, which is quite burdensome for a portable computer such as a laptop. Additionally, the cost for our product is expected to be much lower than the competition.

The key factors to consumer acceptance of the product are that the picture quality be exceptional, and that the product does not interfere with cordless telephones and other wireless signals.

**Financials**

**Preliminary Financial Status**

The preliminary financial needs are based on the requirements for the product design. These needs are outlined in Table 2. The initial design investment has been obtained through Calvin College’s BizPlan2007, where the team received
first place. Additional funding is needed for a Wireless USB Development kit which will enable the product to fulfill market requirements including high quality. Preliminary financial revenue and operating costs are outlined in Table 2.

**Exit Strategy**

To exit this market, the company would rely on a sale of the intellectual property. Based on the engineering time expected for the product, the value of the intellectual property is $200,000. This should allow the company to recoup its initial investment amount. A secondary option is a sale of the entire company to recoup shareholder investment. The first foreseeable exit points are after the initial design is completed. Another possible exit point is after the first manufacturing contract expires. These both insure minimal losses for the company.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>TIMING</th>
<th>FUNDING</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Investment</td>
<td>November 2008 – COMPLETED</td>
<td>COMPLETED</td>
<td>Begin design work</td>
</tr>
<tr>
<td>Wireless USB Development Kit</td>
<td>January - March 2008 - In</td>
<td>$25,000</td>
<td>Product Design Equipment</td>
</tr>
<tr>
<td>Legal Counsel</td>
<td>February - March 2008</td>
<td>$25,000</td>
<td>Patent and design protection</td>
</tr>
<tr>
<td>Design Prototyping</td>
<td>March - August 2008</td>
<td>$50,000</td>
<td>Test Final Design for sale</td>
</tr>
<tr>
<td>Bring to Market</td>
<td>June - December 2008</td>
<td>$80,000</td>
<td>Begin Commercial Location and Sales</td>
</tr>
</tbody>
</table>

---

For the years ended:

<table>
<thead>
<tr>
<th>Summary</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Income</td>
<td>118,050</td>
<td>579,900</td>
<td>1,038,900</td>
<td>1,454,100</td>
<td>1,869,300</td>
</tr>
<tr>
<td>Total Expenses</td>
<td>169,550</td>
<td>370,500</td>
<td>326,700</td>
<td>326,700</td>
<td>326,700</td>
</tr>
<tr>
<td>Total Gross Income</td>
<td>287,600</td>
<td>950,400</td>
<td>1,365,600</td>
<td>1,780,800</td>
<td>2,196,000</td>
</tr>
<tr>
<td>Ending Cash Balance</td>
<td>148,050</td>
<td>727,950</td>
<td>1,766,850</td>
<td>3,220,950</td>
<td>5,060,250</td>
</tr>
</tbody>
</table>
11. MANUFACTURING PLAN

11.1. Production

Volturna’s prototype is, in its own sense, a feasibility study. The use of development kits and demo code reflects upon the products development phase. However, the Wireless-USB product is the main focus for production. Volturna expects to sell roughly 2500 units the first year, eventually reaching 4000 units sold in the second year and 5600 sold in the third year. These numbers are estimates based on demand of similar products on the market and the expectation that as company recognition grows, sales will rise also.

Manufacturing will be contracted. Volturna would like to find a domestic manufacturer for their product in light of the recent economic turndown. Staying in the country may help save jobs, and that would be a side-benefit to Volturna’s venture. Additionally, because domestic companies must adhere to US quality standards, it will be easier for us to monitor quality. Finally, shipping for our relatively low volume of products will be cheaper.

Both the USB dongle and VWMP receiver will be custom made. A power adapter for the VWMP and A/V cables for connecting the receiver to the TV will be included. Additionally, customers will receive an instruction booklet and installation CD. Table 11-1 shows some of these estimated costs.

<table>
<thead>
<tr>
<th>Item</th>
<th>Model</th>
<th>Manufacturer</th>
<th>Cost</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box for packaging</td>
<td>S-10714</td>
<td>ULine</td>
<td>$3.55 (100 min)</td>
<td>10”x8”x4”, contains suspension insert</td>
</tr>
<tr>
<td>Instruction booklet</td>
<td>-</td>
<td>-</td>
<td>$1500/5000</td>
<td>Estimated based on online suppliers</td>
</tr>
<tr>
<td>Installation CD</td>
<td>-</td>
<td>-</td>
<td>$450/1000 CDs</td>
<td>Estimated based on online suppliers</td>
</tr>
</tbody>
</table>

Packaging, printed circuit-board (PCB) printing and assembly will be done via a contractor. The team will provide initial customer support, marketing, and oversee product deployment.

11.2. Product

The VWMP product will consist of two parts that will both be manufactured rather than third-party provided. Wireless USB dongles are still difficult to find and quite costly to purchase wholesale.
Table 11-2 shows the basic bill of materials for the VWMP. Each device will require one of each of the materials shown, with the exception of the antenna and wireless USB chip which will need to be in both the dongle and receiver. The total material cost (excluding shipping) as listed is $73.48. It is likely this cost could decrease as the receiver case for example can be found from a manufacturer who specializes in large bulk orders. Also the manufacturer(s) we decide to contract with for PCB printing and product assembly may have knowledge of lower cost bulk options for these parts. However, this does not take into account the cost of PCB printing, the cost of the manufacturers and the costs of packaging listed in Table 11-1. Volturna’s target price for the product is $130 and the material cost approaches this price.
<table>
<thead>
<tr>
<th>Part</th>
<th>Model</th>
<th>Manufacturer</th>
<th>Cost</th>
<th>Minimum Order</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless USB Chip</td>
<td>SC3503 Ripcord1</td>
<td>Staccato Communications</td>
<td>~$10</td>
<td></td>
<td>Band: 3.1-4.8GHz</td>
</tr>
<tr>
<td>Antenna</td>
<td>445-3385</td>
<td>TDK Corporation</td>
<td>~$4.004</td>
<td>1500</td>
<td>3.1-5.2GHz, 2.0dBi gain</td>
</tr>
<tr>
<td>Dongle Case</td>
<td>P-1304TX</td>
<td>PolyCase</td>
<td>~$0.58</td>
<td>100</td>
<td>ABS plastic, 0.45”x1.2”x2”</td>
</tr>
<tr>
<td>USB Plug</td>
<td>74032-001LF</td>
<td>FCI</td>
<td>~$0.27218</td>
<td>4800</td>
<td>Type A Male</td>
</tr>
<tr>
<td>USB Controller</td>
<td>ISP1582BS-T</td>
<td>NXP Semiconductors</td>
<td>~$3.15788</td>
<td>2000</td>
<td>Serial interface</td>
</tr>
<tr>
<td>Receiver Case</td>
<td>AG-85 Keyhole Wall Mount</td>
<td>PolyCase</td>
<td>~$5.39</td>
<td>100</td>
<td>ABS plastic, Wall mountable, 5”x7.5”x1.75”</td>
</tr>
<tr>
<td>Power Adaptor</td>
<td>412-106054</td>
<td>Xicon</td>
<td>~$3.63</td>
<td>100</td>
<td>2.1mm barrel plug, 120VAC wall adaptor, Output: 5V, 500mA</td>
</tr>
<tr>
<td>Barrel Plug</td>
<td>CP-004A</td>
<td>CUI Inc</td>
<td>~$0.3569</td>
<td>12000</td>
<td>2.1mm inner diam.</td>
</tr>
<tr>
<td>Component Cables</td>
<td>55-862B</td>
<td>Ram</td>
<td>~$3.55</td>
<td></td>
<td>A/V red/yellow/white</td>
</tr>
<tr>
<td>Microprocessor</td>
<td>i.MX27</td>
<td>FreeScale</td>
<td>~$15.168</td>
<td>1000</td>
<td>32bit ARM processor w/ video codecs</td>
</tr>
<tr>
<td>RCA Jacks Connectors</td>
<td>RCJ-31234</td>
<td>CUI Inc</td>
<td>~$0.66766</td>
<td>2500</td>
<td>Red, White, Yellow</td>
</tr>
<tr>
<td>Audio/Video DAC</td>
<td>THS8200PFP</td>
<td>Texas Instruments</td>
<td>~$7.50</td>
<td>500</td>
<td>11 bit, made for component video conversion</td>
</tr>
<tr>
<td>NTSC Encoder</td>
<td>AD723ARU</td>
<td>Analog Devices, Inc</td>
<td>~$5.1956</td>
<td>2500</td>
<td></td>
</tr>
</tbody>
</table>


The FreeScale microprocessor was chosen as a representative microprocessor since Volturama will need to do prototyping with the Wireless USB chip before the final product can be designed. The microprocessor was also selected because it contains MPEG and H.16x family codecs. Buying directly from Freescale will likely be cheaper than the Digi-key cost. The PolyCase cases were chosen both for our ability to add a company logo to them and for their aesthetic shapes.
Figure 11-1 Receiver Casing

The product will require two PCBs. Because information needs to flow in one direction, the wiring should not be overly complex and a 2-layer board should be sufficient for the receiver. It is possible that the limited size of the dongle will necessitate a 4 layer PCB. An order of 2500 2-layer receiver boards and 2500 2-layer dongle boards from PCB Express would cost $6,179.75 or approximately $2.50 per product. This brings the total cost from above to $74.98. Adding a 20% contingency/shipping to that brings the estimated total cost of production to $89.98.

Figure 11-2 USB Dongle PCB Schematic

DONGLE: POLYCASE P-1304TX INNER
DIMENSIONS: 29 x 49 x 9 mm
Even though costs are approaching the $130 price tag, Volturna still recommends that the product be sold at $130 wholesale and that rather than increase the price, cheaper components and manufacturing methods be found through working with manufacturers and contracting with part suppliers. Increasing the price much higher will mean the product loses its distinction as a low cost product compared to competition.

11.3. Product Support and Marketing

Product support will initially be provided via an online site and by the team directly. As the volume of sales increases throughout the first and second years, the team will hire a small team of two to three customer service representatives who will be available from 8AM-5PM, Mon-Fri. Assuming that 25% of the expected 5000 first year customers request service, and each customer service call lasts approximately 15 minutes, 312.5 hours of customer service time will be required. Assume a pretty even spread over the year, this amounts to 6 hours of customer service time per week. One person will be able to handle that inquiry level, and a temp can be hired for the Christmas/January season when customer service requests tend to peak.
Marketing will be done by the Volturna team. The internet will be the primary marketing tool. Volturna will use online ads, as well as online sellers such as amazon.com to market their devices. Pending their willingness to contract with the team, companies such as newegg.com and tigerdirect.com will also be used as product deployment points.

12. **CONCLUSION**

12.1. **Lessons Learned**

Throughout the timeline of the project, the team learned a lot about overall design of a product, from its inception as a business idea through to the completion of the design. Calvin College’s senior design course and project taught the team about considering all aspects of a project, including costs, competitors, marketing, and the work necessary to get a product from its inception to being a real product. Providing team members with all the tools necessary to accomplish the end goals is essential for the success of any project. These tools include purchasing power, information, decision making ability, deadlines, time, technical understanding, and many others. Without all the necessary tools, progress can stagnate, hindering further progress and hampering deadlines. With Volturna’s Wireless Media Player, some of the issues facing the project were end prototype completion, maintaining potential intellectual property, and design satisfaction. A fully functional prototype is essential to prove the feasibility of the product both conceptually and technically. Also, the development phase ought to materialize the original requirements at the minimum. Throughout the development of a prototype, understanding where intellectual property exists is a valuable resource to the developers - to legally accredit ownership of the technology (patents, trade secrets, etc). Although Volturna did not retain any new, patentable material, the importance of recognizing intellectual property remains. Because of the team's desire to create a marketable product and potentially a company based around their design, maintaining potential intellectual property was important and led the team to pursue Calvin's BizPlan2007 competition. This decision contributed to delaying necessary prototype design decisions, and compressed the majority of the design work into the second semester. Later, uncertainty on the interconnect between the two development boards caused a delay in ordering the parts necessary which further pushed back the development timeline. As a result, the team learned that drilling down into the hard details to find the relevant information followed by communicating what was learned back to the team and ensuring that it is understood is essential to the overall progress of the team. The team learned that working as a team means to rely on the strengths of every member. Some members are better at ensuring things get done, some are better at writing reports, some are better at dealing with technical details, and some are better at dealing with the everyday tasks.
Learning to use individual skills to the benefit of the team provided one of the best lessons for Volturna. Additionally, the team learned that it is essential to focus on creating workable timetables and to provide ample extra time with which to accomplish tasks. Lastly, it is essential to create a plan, and dynamically work within the plan. Although the plan may need change, it is important to strive to keep the level of progress at the point it should be.

12.2. Future work

In applications development involving hardware and software, such as the Volturna Wireless Media Player, there exists the possibility of improvement. Given enough time and testing, Volturna’s project would be tweaked to become more efficient with its power and data management. This would be confined within the tools and kits readily available. Although the project is proven feasible, it remains classified as a prototype. This classification evolved out of Calvin College sponsored BizPlan2007, of which the team received a $1500 boost in funding for the project. The manufacturing scheme detailed how it would be feasible to manufacture initial versions of marketable designs of the project (that do not include unnecessary development kit components). More work would be done to devise engineering manufacturing schemes to convert the prototype into a commercial off-the-shelf (COTS) product.

However, there currently exists technology capable of performance much greater than that of the prototype, but can only be acquired with adequate funding and commitment. During the decision making process, the team spoke with a few select companies to inquire about a possible educational discount. One in particular was Staccato Communications, who stated that they might be able to offer a Wireless USB development kit, for about $17,000 (not including active support). Due to obvious cost and funding restraints, the team chose to purchase a lower-bandwidth yet effective 802.11 development kit to initialize work on a prototype as soon as possible. However, the future looks promising in terms of the marketability of the design, which in turn would garner further funding support in terms of business startups. This funding, accompanying an already functional prototype, would instigate further high definition and high bandwidth upgrades to the product. An initial, intuitive upgrade to the prototype would utilize the new IEEE 802.11n wireless protocol, debuting in 2009. This would allow a range increase up to seventy meters as opposed to thirty eight of 802.11b. Also, this would drastically increase throughput to about 70 Mbps as opposed to 23 Mbps. Additionally, 802.11n offers backwards compatibility with 802.11b, making an upgrade relatively easy. Although no 802.11n development kits are currently for sale, an estimated cost for one would be at least $500. Further upgrades would include an investment into the new Wireless USB protocol, as well as a possible upgrade in video encoding algorithm to WMV9 (which would require expensive licensing for manufacturing). WMV9 would provide additional quality and efficient compression for video transmission. The Wireless USB protocol
would allow 110 Mbps at a range of ten meters, or up to 480 Mbps at a range of three meters. This even surpasses the 802.11n protocol and without the unnecessarily large range. As mentioned before, the Wireless USB development costs remain significant; approximately $17,000. In the end, while these changes would equally require change in hardware, the feasibility and experience remains. Despite the change in hardware for future prototypes and improvements, development kits contain similar software in the development package. The initial prototype mentioned in this report provides the inception for future application study.

13. ACKNOWLEDGEMENTS

Volturna would like to thank Professor Steven VanderLeest for his role as team advisor, as well as the Calvin BizPlan sponsors, in particular Mr. Brett Logan, for their financial support that made this project possible. The team also thanks Mr. Tim Theriault for his industry perspective and advice.

14. BIBLIOGRAPHY


2. D-Link. PC-on-TV Media Player. [Online] [Cited: April 19, 2008.]


7. Fairhurst, Dr. Gorry. Summary of MPEG Compression Capability. [Online]


15. APPENDIX

15.1. NCIIA Grant Proposal

1.0 INTRODUCTION: THE PROBLEM & PROJECTED SOLUTION

The rise of digital multimedia has led to personal computers becoming one of the primary multimedia viewing mediums with YouTube.com users watching over 70 million videos a day. Unfortunately, PCs provide only a limited ability for group viewing, and connecting a PC to a TV requires an intimate knowledge of computer settings and a number of extra cables. To address this issue, we aim to create an easy-to-use wireless audio and video transmitter and receiver for sending video and audio from a PC to a television.
To create this product, we seek to establish a consumer electronics design company, Volturna, specializing in the licensing of designs. Volturna would create marketable products, and then license the rights to produce those designs to manufacturers.

2.0 HISTORY AND CONTEXT

Five students from Calvin College, each with an interest in entrepreneurship, have joined together to form a team, Volturna. They have come up with a product that would meet the requirements for a senior engineering project course at Calvin College – one which market research shows a strong demand for with 100% of respondents who frequently watch movies on their PCs having interest in purchasing. The idea is a device to easily send sound and video from a personal computer directly to a television, without cables. This device would use technology built into computers to send the video wirelessly to a device attached to the TV. The computer would recognize the device similar to that shown in Figure 2-2-1 and users would be able to connect to it with the click of a button. The basic idea is shown in Figure 2-2-2.

Table 2-2-1 Marketing Survey Results

<table>
<thead>
<tr>
<th>Category (Number of Respondents)</th>
<th>All Respondents (20)</th>
<th>Respondents who indicated interest in buying (11)</th>
<th>Respondents who indicated no interest in buying (8)</th>
</tr>
</thead>
</table>

Figure 2-2-1: Computer Recognition of Device

Figure 2-2-2 Product Concept
Watch movies on PC  70%  100%  38%
Watch TV on PC  80%  100%  75%
Watch home videos on PC  85%  95%  75%
Use PC for presentations  90%  60%  90%
Use PC for slideshows  70%  64%  38%
Other Use  5%  9% (Games)  0%

Most influential consideration in purchasing:
- Operating System (1)
- Cost (2), Operating System (2), TV Output (3)
- Cost (2), Device Size (3)
- Operating System (1), Cost (1), Operating System (1), Device Size (3)

In fall of 2007, Volturna entered a business plan competition, BizPlan2007, at Calvin College in which the team generated a business plan (attached as an appendix) and participated in a live business plan presentation. The group of seven judges, comprised of business owners and entrepreneurs, awarded the team first place in that competition, after which they took the team out for dinner and offered advice and assistance.

3.0 INTRODUCTION: THE PROBLEM & PROJECTED SOLUTION

3.1 Members

Gwendolyn R. Einfeld, a resident of Seattle, Washington, acts as team project manager, planning project schedules and assigning weekly tasks. Gwendolyn also works on the wireless reception and decoding portions of the research and design, as well as oversees product marketing. She plans to graduate in May of 2008 with a Bachelor of Science in Engineering, with a concentration in electrical and computer engineering from Calvin College.

Leah M. Einfeld, also from Seattle, Washington, takes her role as the team’s financial advisor. She prepares financial forecasts for the team and oversees budget development. Leah, in her junior year at Calvin College, invests her studies towards a degree in Economics.

Richard K. Kwakye, originating from Ghana, manages device design to ensure that the design meets project requirements and team quality standards. He acts as the meeting recorder, contacts potential suppliers, and is responsible for video compression technology and ensuring image quality. Richard graduates from Calvin College in spring of 2008 holding a Bachelor of Science in Engineering with a concentration in electrical and computer engineering.

Charlie S. Reitsma, of Cedar Lake, Indiana, manages Volturna’s business development. His responsibilities include business plan creation and compilation, website design, and developing Volturna’s business strategy. He also works with Leah to ensure that the business financial forecast aligns with the technical requirements of the project – development costs, materials, etc. As far as research and design, Charlie works on the PC interface and wireless transmission. Charlie plans to graduate from Calvin College next spring with a Bachelor of Science in Engineering with a concentration in international electrical and computer engineering.
Theodore A. Worst, of Grand Rapids, Michigan, manages team communications. His responsibilities include task tracking, setting of weekly goals, and generation of weekly status reports. He is working with the research and design of the digital-to-analog converter. Ted graduates in May of 2008 with a Bachelor of Science in Calvin College's electrical and computer engineering concentration.

3.2 Advisors and Mentors

Dr. Steven H. VanderLeest is the primary advisor for Team Volturna. Dr. VanderLeest's consulting work includes Tandem Computers Inc, Smiths Aerospace and Laser Alignment Control in the areas of digital design, software design, ASIC design and laser control. Currently, Dr VanderLeest acts as the chair of Calvin College's Engineering Department and as an industrial consultant in the field of safety-critical embedded systems design for DornerWorks Limited.

Tim Theriault, a senior engineer involved with the management and strategic direction in digital technology development at General Electric (GE) aviation, acts as an additional engineering mentor.

Paul Holtrop assists the team in creating a viable business strategy. Paul works for Impact Management and has worked within business and industry for more than 20 years, and has taught and implemented lean manufacturing technology while leading teams at a Fortune 500 company. He has a broad range of operations experience, holding various roles in engineering, front-line supervision, materials management, logistics management, and production management.

4.0 WORK PLAN AND OUTCOMES

The initial prototype design has been started and will be completed before June of 2008. The team already started designing and ordering parts, and has made contact with companies offering the technology necessary to create a prototype and verified a purchasing option with Staccato Communications (see appendix). As the team moves closer to design completion, we continue to seek out funding.

The team plans to form a business at the successful completion of the project. The team has already completed a number of business plans as well as entered Calvin College's BizPlan2007 competition in which the panel of judges, all of whom are entrepreneurs, awarded Volturna first place out of 11 teams. The team is currently working on its entry for the Great Lake Entrepreneurship Quest.

Volturna needs additional funding in order to purchase equipment that will enable the design of a product with the quality consumers need. This equipment will enable the development of high-quality video and sound from the receiver meeting the market requirement of television quality or better. It is imperative that Volturna receives this funding in order to start a successful venture.

On receipt of additional funding, Volturna intends to expand its prototype's capabilities. Volturna then plans to consult with our legal counsel for a patent on its prototype to protect its intellectual and if applicable apply for a provisional patent in March of 2008 and a full patent in August (see appendix for legal counsel information). Volturna further plans to maintain confidentiality for the designs in order to maintain exclusivity. Once the team has completed the prototype in May, we intend to give a formal public presentation to disseminate lessons learned and jump-start the business, start a commercial location, seek out a licensor for the initial proprietary design, and start design of an MP3 player that
connects wirelessly to both your computer and speakers to bring the music with you as you move from room to room. The timeline for this is shown in Figure 4.0.1.

To verify the technical feasibility of the design, the team used existing computer technology to test the concept by sending audio and video directly from one computer over a wireless signal to another computer. The test demonstrated that the technology we hope to use will be a viable way to transmit audio and video, and that the project will be technically feasible.

Similar products to our initial target design currently are on the market, but Volturna's offering in simplifies the connection and will make it easier for use with a laptop. Current devices such as the FlyImage 5000 and SmartHome's 7743B require bulky connections to your computer. Volturna's design will use a device the size of a USB keyfob. Dell and Lenovo have started putting the technology in their laptops as well, and most laptops will have the technology in the next few years, making Volturna's device much easier to use than both the FlyImage and SmartHome versions.

The team plans to create a Limited Liability Company centered around the licensing of its proprietary designs. We estimate this licensing to account for $200,000 of income for each product, as well as $40,000 for each engineering design. The team estimates that it will begin the marketing and licensing of a new design every 4 months, with development spanning a six month period. The projected financial outlook using this model is shown in Table 4.0.1.

<table>
<thead>
<tr>
<th>Table 4.0.1: Projected Financial outlook</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>2008</strong></td>
</tr>
<tr>
<td><strong>Net Income</strong></td>
</tr>
<tr>
<td><strong>Total Expenses</strong></td>
</tr>
<tr>
<td><strong>Total Gross Income</strong></td>
</tr>
<tr>
<td><strong>Ending Cash Balance</strong></td>
</tr>
</tbody>
</table>

Current competition in the market for Volturna comes mainly from internal design firms within product manufacturers such as D-Link and Linksys who do most of their consumer electronics design in house.
Other competition comes from design companies such as Hyphen Design, Lunar Design, and Fredricks Design. These companies thrive on creating innovative ideas to sell to companies.

As part of the first entrepreneurship competition at Calvin College as well as one of the first senior design teams seeking to create an entrepreneurial venture, Volturna's experiences will encourage other teams to follow in its footsteps. By documenting our team's steps to creating a business and mentoring future teams, Volturna will enable other groups to start on the entrepreneurial journey.

Finally, the team continues to find a benefit in learning about other fields of study. The team must work together to create innovative products while maintaining sound business principles, stretching the team beyond their training and forcing them to consider all the issues relevant to both engineering and business.

5.0 **Evaluation and Sustainability Plan**

Volturna will evaluate its internal success based on the following factors:

1. Design of a device capable of receiving audio and video wirelessly
2. Submitting a provisional patent application
3. Give a public presentation of the technology involved in the design
4. Present the idea to venture capitalists

After the grant ends, the project may continue in a number of ways. First, if successful the team will develop a business. Once the team is able to secure funding for day-to-day operations, they will establish a commercial location. The team will then pursue the sale of their designs, and start development on new designs. Secondly, the team will mentor teams hoping to pursue entrepreneurship. This might also include providing financial backing to students if the business is successful. The equipment the team purchases with project funds will become the property of Calvin College and will be available to any engineer teams who wish to use them, and will provide future teams with the capability to push themselves to the next level of innovation.
15.2. Market Research Survey

What is your age bracket?
___ <18 ___ 19-25 ___ 26-35 ___ 36-55 ___ >55

What is your gender?
___ Male ___ Female

Do you use a computer to view any of the following multimedia? (check all that apply)
___ Movies
___ Home or Online videos (for example, YouTube)
___ TV shows
___ Presentations (i.e. PowerPoint)
___ Slideshows
___ Other (describe) _________________________
___ None

What is your primary operating system?
___ Windows XP
___ Windows Vista
___ Mac OS
___ Linux
___ Other (describe) _________________________

Would you purchase a device to allow you to wirelessly play multimedia (audio and video) from your computer on a TV or projector?
___ Yes
___ No

If yes, for what purposes would you use such a device? (check all that apply)
___ Movies
___ Home or Online videos (for example YouTube)
___ TV shows
___ Presentations (i.e. PowerPoint)
___ Slideshows
___ Other (describe) _________________________

If yes, what would the primary use of the device be?
___ Personal (entertainment, etc)
___ Business or Academic (meetings, presentations, etc)

Rate the top 3 considerations that would affect your decision to purchase a wireless video transmitter.
___ Interfaces with my operating system of choice
___ Cost
___ Size/portability
___ TV output options (S-Video, RCA, etc)
___ Ability to interface with a projector
___ Supports HD
___ Other (describe) _________________________
15.3. Email from Staccato Communications

Charlie Reitsma <CharlesReitsma@gmail.com>

---

RE: WUSB Development kit information

2 messages

---

Eckart Voskamp <eckart.voskamp@staccatocommunications.com> Thu, Nov 29, 2007 at 4:35 PM

To: Charles Reitsma <charlesreitsma@gmail.com>

Dear Charles,

Thank you for your inquiry and your interest in Staccato products. The list price for our development kit is $25K and includes 2 staccato DVK boards, 1 SBC (single board computer), all software drivers, all software source code, documentation and upgrade support for one year. At this time, we don't have an educational pricing and since the technology is still fairly new, demand is currently very high. However, in 4 month time frame I might have a more relaxed situation and we could work something out in the range of 30%-50% - which would include limited support from our side only.

I hope this helps a bit.

Best Regards,

-- Eckart

Eckart Voskamp
Director of Sales
North America and Europe
Staccato Communications
6195 Lusk Blvd., Suite 200
Hello,

My name is Charlie Reitsma, and I was hoping your company might be able to help me out. I'm currently interested in starting an entrepreneurial venture through my college that would use Wireless USB technology. So far, the team that is working on this project with me has entered a business plan competition and won first place - an award of $1500. We're currently pursing a grant to generate funding for the Wireless USB components necessary to create our design, and for the grant, it would be nice if we were able to give a good estimate for the price of a Wireless USB development kit. Could you please let me know what price range might be expected for a Wireless USB development kit from Staccato Communications for educational and business purposes? We hope to be able to purchase a development kit within the next 4 months. Your help is greatly appreciated!

Sincerely,
Charlie Reitsma
Calvin College

Charlie Reitsma <CharlesReitsma@gmail.com>  Fri, Nov 30, 2007 at 11:49 AM

To: Eckart Voskamp <eckart.voskamp@staccatocommunications.com>

Dear Eckart,

Thanks so much for your response. We were quite please to hear that it might be possible to work something out. The prices you proposed would be very affordable with our grant proposal, and we certainly hope to be
able to work something out. I hope you don't mind, but I'm going to attach your email as a part of the grant proposal to demonstrate feasibility. Thanks again for your willingness to help us out, we will be sure to stay in touch!

Sincerely,
Charles Samuel Reitsma
Calvin College

[Quoted text hidden]
15.5. Wireless Receive Code

CSocket.h

#ifndef CSOCKET_H
#define CSOCKET_H

#ifdef __cplusplus
extern "C"
{
#endif

#include <sockapi.h>
#endif

#define BOOL int

/* CSocket class declaration */
class CSocket
{
  public:
  // Construction & destruction
  CSocket();
  BOOL Create(int type);
  BOOL Create(int type, unsigned short port, unsigned long addr = 0);
  virtual ~CSocket();

  // Attributes
  operator SOCKET() const { return fd;}
  BOOL Attach(SOCKET hsocket);
  SOCKET Detach();
  BOOL GetPeerName(unsigned short &port, unsigned long &addr);
  BOOL GetSockName(unsigned short &port, unsigned long &addr);
  BOOL SetSockOpt(int opt_name, const void* opt_val, int opt_len, int level = SOL_SOCKET);
  BOOL GetSockOpt(int opt_name, void* opt_val, int *opt_len, int level = SOL_SOCKET);
  virtual BOOL Accept(CSocket& connected_socket,unsigned short &port, unsigned long &addr);
  BOOL Bind(unsigned short port, unsigned long addr);

  // Operations
  virtual BOOL Close();
  BOOL Connect(unsigned short port, unsigned long addr);
  BOOL Listen(int backlog = 5);
  virtual int Receive(char *buf, int buflen, int flags = 0);
  int ReceiveFrom(char *buf, int buflen, unsigned short &port, unsigned long &addr, int flags = 0);
  virtual int Send(const char* buf, int buflen, int flags = 0);
  int SendTo(const char* buf, int buflen, unsigned short port, unsigned long addr, int flags = 0);
  enum { receives = 0, sends = 1, both = 2 };  
  BOOL ShutDown(int how = sends);

  private:
    SOCKET fd;
    CSocket(CSocket &so);  /* no implementation */
    void operator=(const CSocket &so);  /* no implementation */
};

#ifdef __cplusplus
}
#endif
#endif
#include <Npttypes.h>
#include "CSocket.h"
#ifdef SOCKADDR sockaddr

/* CSockAddrIn structure declaration */
struct CSockAddrIn
{
  short   sin_family;    // address family
  unsigned short   sin_port;    // port
  struct   in_addr  sin_addr;  // ip address
  char    sin_zero[8];

  /* Adding constructors to sockaddr_in structure */
  CSockAddrIn() {sin_family = AF_INET;}
  CSockAddrIn(unsigned short port, unsigned long addr = 0)
  {
    sin_family = AF_INET;
    sin_port = port;
    sin_addr.s_addr = addr;
  }
};

/* Construction */
*/
* Function: CSocket::CSocket()
* Description:
* This is a default constructor
* Parameters: none
*/
CSocket::CSocket()
{
  fd = 0;
  errno = 0;
}

/* Functions: BOOL CSocket::Create(int type)
*    BOOL CSocket::Create(int type, unsigned short port, unsigned long addr)
* */
BOOL CSocket::Create(int type) {
  if ((fd = socket(AF_INET, type, 0)) < 0)
    fd = 0;

  return (fd > 0);
}
BOOL CSocket::Create(int type, unsigned short port, unsigned long addr) {
  CSockAddrIn sin(port, addr);

  if ((fd = socket(AF_INET, type, 0)) < 0)
    fd = 0;
if (fd) {
    if (bind(fd, (SOCKADDR *) &sin, sizeof(sin)))
    {
        closesocket(fd);
        fd = 0;
    }
    return (fd > 0);
}

// Destruction
/*
 * Function: CSocket::~CSocket()
 * Description:
 * Destructor
 */
CSocket::~CSocket()
{
    if (fd)
        closesocket(fd);
}

// Attributes
/*
 * Functions: BOOL CSocket::Attach(SOCKET hsocket)
 * Description:
 * This routine attaches a socket handle to this CSocket object
 * Parameters: hsocket - socket handle
 * Return Values: TRUE
 */
BOOL CSocket::Attach(SOCKET hsocket)
{
    fd = hsocket;
    return TRUE;
}

/*
 * Functions: SOCKET CSocket::Detach()
 * Description:
 * This routine detaches a socket handle from this CSocket object
 * Parameters: none
 * Return Values: a detached socket handle
 */
SOCKET CSocket::Detach()
{
    SOCKET so = fd;
    fd = 0;
    return so;
}

/*
 * Functions: BOOL CSocket::GetPeerName(unsigned short &port, unsigned long& addr)
 * Description:
 * This routine gets the port and IP address of the peer socket
 * to which this socket is connected.
 * Parameters:
 * port - reference to store the peer port
 * addr
 */
BOOL CSocket::GetPeerName(unsigned short &port, unsigned long& addr)
{
    if (fd)
    {
        CSockAddrIn sin(port, addr);
        int addrrlen = sizeof(sin);
        if (getpeername(fd, (SOCKADDR *) &sin, &addrrlen) == 0)
        {
            addr = sin.sin_addr.s_addr;
            port = sin.sin_port;
            return TRUE;
        }
    }
    return FALSE;
}

BOOL CSocket::GetSockName(unsigned short &port, unsigned long& addr)
{
    if (fd)
    {
        CSockAddrIn sin(port, addr);
        int addrrlen = sizeof(sin);
        if (getsockname(fd, (SOCKADDR *) &sin, &addrrlen) == 0)
        {
            addr = sin.sin_addr.s_addr;
            port = sin.sin_port;
            return TRUE;
        }
    }
    return FALSE;
}

BOOL CSocket::GetSockOpt(int opt_name, void* opt_val, int *opt_len, int level)
{
    return FALSE;
}
BOOL CSocket::GetSockOpt(int opt_name, void* opt_val, int* opt_len, int level)
{
    if (fd && getsockopt(fd, level, opt_name, (char*)opt_val, opt_len) == 0)
        return TRUE;
    return FALSE;
}

/*
 * Functions: BOOL CSocket::SetSockOpt(int opt_name, const void* opt_val, int opt_len, int level)
 * Description:
 *   This routine sets a socket option.
 *   
 * Parameters:
 *     opt_name    - option name
 *     opt_val     - option value pointer
 *     opt_len     - size of opt_val in bytes
 *     level       - option level
 * Return Values:
 *     TRUE
 *     FALSE
 */
BOOL CSocket::SetSockOpt(int opt_name, const void* opt_val, int opt_len, int level)
{
    if (fd && setsockopt(fd, level, opt_name, (char*)opt_val, opt_len) == 0)
        return TRUE;
    return FALSE;
}

// Operations
/*
 * Functions: BOOL CSocket::Accept(CSocket& connected_socket,unsigned short &port, unsigned long &addr)
 * Description:
 *   This routine accepts a connection on a socket. It creates a new socket for
 *   first pending connection and attaches it to connected_socket.
 *   The accepted socket (connected_socket) cannot be used to accept more connections.
 *   The original socket remains open and listening.
 *   Applies only to SOC_STREAM sockets.
 *   
 * Parameters:
 *     connected_socket - A reference identifying a new socket that is available for connection
 *     port             - reference to store the peer port
 *     addr             - reference to store the peer IP address
 *   
 * Return Values:
 *     TRUE
 *     FALSE
 */
BOOL CSocket::Accept(CSocket& connected_socket,unsigned short &port, unsigned long &addr)
{
    if (fd)
    {
        CSockAddrIn sin(port,addr);
        int addrlen = sizeof(sin);
        SOCKET so = 0;
        if (((so = accept(fd, (SOCKADDR *) &sin, &addrlen)) > 0)
            if (connected_socket.Attach(so))
return TRUE;
  }
  }
  return FALSE;
}

/*
 * Functions: BOOL CSocket::Bind(unsigned short port, unsigned long addr)
 * Description:
 *  This routine binds this socket to the port and IP address.
 * *
 * Parameters:
 *  port    - port to bind
 *  addr    - IP address to bind
 * *
 * Return Values:
 *  TRUE
 *  FALSE
 */
BOOL CSocket::Bind(unsigned short port, unsigned long addr)
{
  if (fd)
  {
    CSockAddrIn sin(port,addr);
    if (bind(fd, (SOCKADDR *)&sin, sizeof(sin)) == 0)
      return TRUE;
  }
  return FALSE;
}

/*
 * Functions: BOOL CSocket::Close()
 * Description:
 *  This routine closes this socket.
 * *
 * Parameters:
 *  none
 * *
 * Return Values:
 *  TRUE
 *  FALSE
 */
BOOL CSocket::Close()
{
  if (fd && (closesocket(fd) == 0))
  {
    fd = 0;
    return TRUE;
  }
  return FALSE;
}

/*
 * Functions: BOOL CSocket::Connect(unsigned short port, unsigned long addr)
 * Description:
 *  This routine connects this socket to the network node
 *  with the specified port and IP address
 * *
 * Parameters:
 *  port    - port to connect

BOOL CSocket::Connect(unsigned short port, unsigned long addr)
{
    if (fd)
    {
        CSockAddrIn sin(port, addr);
        if (connect(fd, (SOCKADDR *)&sin, sizeof(sin)) == 0)
            return TRUE;
        return FALSE;
    }
}

BOOL CSocket::Listen(int backlog)
{
    return fd && (listen(fd, backlog) == 0);
}

BOOL CSocket::ShutDown(int how)
{
    return fd && (shutdown(fd, how) == 0);
}

int CSocket::Receive(char *buf, int buflen, int flags)
{
    // Implementation goes here...
}
/* flags - the possible flags are: MSG_PEEK, MSG_OOB. */

int CSocket::Receive(char *buf, int buflen, int flags)
{
    if (fd)
        return recv(fd,buf,buflen,flags);

    return 0;
}

/* Functions: int CSocket::Send(const char* buf, int buflen, int flags)
 * Description:  
 *      This routine sends data on a connected socket.
 *
 * Parameters:
 *      buf     - pointer to the data to send
 *      buflen  - number of bytes to send
 *      flags   - the possible flags are: MSG_DONTROUTE, MSG_DONTWAIT.
 *  Return Values:
 *      number of bytes received
 */
int CSocket::Send(const char* buf, int buflen, int flags)
{
    if (fd)
        return send(fd,(char *)buf,buflen,flags);

    return 0;
}

/* Functions: int CSocket::ReceiveFrom(char *buf, int buflen, unsigned short &port, unsigned long &addr, int flags)
 * Description:  
 *      This routine receives incomming data and captures the address the data came from.
 *
 * Parameters:
 *      buf     - pointer to the buffer to receive data.
 *      buflen  - size of the buffer in bytes
 *      port    - reference to store the port
 *      addr    - reference to store the IP address
 *      flags   - the possible flags are: MSG_PEEK, MSG_OOB.
 *  Return Values:
 *      number of data bytes received
 */
int CSocket::ReceiveFrom(char *buf, int buflen, unsigned short &port, unsigned long &addr, int flags)
{
    int cnt = -1;
    if (fd)
    {
        CSockAddrIn sin(port,addr);
        int addrlen = sizeof(sin);
        if ((cnt = recvfrom(fd, buf, buflen, flags,(SOCKADDR *)&sin, &addrlen)) >= 0)
        {
            addr = sin.sin_addr.s_addr;
            port = sin.sin_port;
        }
    }
    return cnt;

    /* Functions: see if I can add "here we are" to this */
int CSocket::SendTo(const char* buf, int buflen, unsigned short port, unsigned long addr, int flags)
{
    int cnt = -1;
    if (fd)
    {
        CSockAddrIn sin(port, addr);
        //cnt = sendto(fd, (char *) buf, buflen, flags, (SOCKADDR *)&sin, sizeof(sin));
        cnt = sendto(fd, (char *) buf, buflen, flags, (SOCKADDR *)&sin, sizeof(sin));
    }
    return cnt;
}

GPIO.h

#ifndef GPIO_h
#define GPIO_h

#include <stdio.h>
#include <stdlib.h>

void setGPIO (char * buffer, int bufSize);  //turn this into constantly set GPIO values to UDP
void testTwo (unsigned long threadInput);
#endif

GPIO.cxx

#include <stdio.h>
#include <stdlib.h>
#include <tx_api.h>
#include <string.h>
#include "GPIO.h"
#include "i2c_api.h"
#include "i2c_eeprom.h"
#include "nagpio.h"
#define WRITE_BUFFER_LENGTH 10
#define READ_BUFFER_LENGTH 10
extern TX_SEMAPHORE semaphore_0;

char sendLength(int bytes) {
    char * binaryBytes = "0000000000000000"
    int remainder = 0;
    int num = bytes;
    for (int i = 13; i > 0; i--) {
        remainder = num % 2;
        binaryBytes[i] = remainder;
        num = num >> 1;
    }
}
printf("setting length\n");
return * binaryBytes;
}

void setGPIO (char * retValue, int bytes)
{
#if defined (BSP_ARM9)
#ifdef NA_CC9C_OLD_CARRIER
char writeBuffer[WRITE_BUFFER_LENGTH], readBuffer[WRITE_BUFFER_LENGTH];
unsigned int writeBufferLength=2;
const unsigned char addr = 0xA0;
#endif
#endif
NAI2CInit(I2C_MASTER, 0);       //must be called to use these methods
/* Code to start the user application goes here. */
printf("Running thread #1!\n");
naGpioSetOutput(GPIO_NS9360_0, 1);
naGpioSetOutput(GPIO_NS9360_1, 0);
naGpioSetOutput(GPIO_NS9360_2, 0);
naGpioSetOutput(GPIO_NS9360_3, 0);
naGpioSetOutput(GPIO_NS9360_4, 0);
naGpioSetOutput(GPIO_NS9360_5, 0);
naGpioSetOutput(GPIO_NS9360_6, 0);
naGpioSetOutput(GPIO_NS9360_7, 0);
naGpioSetOutput(GPIO_NS9360_8, 0);
naGpioSetOutput(GPIO_NS9360_9, 0);
naGpioSetOutput(GPIO_NS9360_10, 0);
naGpioSetOutput(GPIO_NS9360_11, 0);
naGpioSetOutput(GPIO_NS9360_12, 0);
naGpioSetOutput(GPIO_NS9360_13, 0);

//call the length function, returns the length in bits
char length;
length = sendLength(bytes);
//test this portion should reDefine the GPIO functions
//you can find GPIO names in nagpio in: include/netos71/src/bsp/platforms/connectcorewi9c

// naGpioSetOutputValue(GPIO_NS9360_0, 1);
// naGpioSetOutputValue(GPIO_NS9360_1, length[1]);
// naGpioSetOutputValue(GPIO_NS9360_2, length[2]);
// naGpioSetOutputValue(GPIO_NS9360_3, length[3]);
// naGpioSetOutputValue(GPIO_NS9360_4, length[4]);
// naGpioSetOutputValue(GPIO_NS9360_5, length[5]);
// naGpioSetOutputValue(GPIO_NS9360_6, length[6]);
// naGpioSetOutputValue(GPIO_NS9360_7, length[7]);
// naGpioSetOutputValue(GPIO_NS9360_8, length[8]);
// naGpioSetOutputValue(GPIO_NS9360_9, length[9]);
// naGpioSetOutputValue(GPIO_NS9360_10, length[10]);
// naGpioSetOutputValue(GPIO_NS9360_11, length[11]);
// naGpioSetOutputValue(GPIO_NS9360_12, length[12]);
// naGpioSetOutputValue(GPIO_NS9360_13, length[13]);
// //wait for a bit
// naGpioSetOutputValue(GPIO_1, 0);
// int bits = bytes*8-13;
// //wait for a bit
// for (int i = 1; i <=bits; i+13) {
// printf("setting GPIOs\n");
// naGpioSetOutputValue(GPIO_1, 1);
// naGpioSetOutputValue(GPIO_2, retValue[i]);
// naGpioSetOutputValue(GPIO_3, retValue[i+1]);
// naGpioSetOutputValue(GPIO_4, retValue[i+2]);
// naGpioSetOutputValue(GPIO_5, retValue[i+3]);
// naGpioSetOutputValue(GPIO_6, retValue[i+4]);
// naGpioSetOutputValue(GPIO_7, retValue[i+5]);
// naGpioSetOutputValue(GPIO_8, retValue[i+6]);
// naGpioSetOutputValue(GPIO_9, retValue[i+7]);
//     naGpioSetOutputValue(GPIO_10, retValue[i+8]);
//     naGpioSetOutputValue(GPIO_11, retValue[i+9]);
//     naGpioSetOutputValue(GPIO_12, retValue[i+10]);
//     naGpioSetOutputValue(GPIO_13, retValue[i+11]);
//     naGpioSetOutputValue(GPIO_14, retValue[i+12]);
//     naGpioSetOutputValue(GPIO_1, 0);
// }
// printf("Done GPIOing...");
#elif defined (BSP_ARM9)
// printf ("Running BSP_ARM9");
// NA12Clinit(I2C_MASTER, 0);  //must be called to use these methods
#ifdef NA_CC9C_OLD_CARRIER
printf("Running something crazy");
for (;;)
{
  tx_thread_sleep(100);
  /* enable I2C EEPROM write control. */
  naGpioSetOutput(GPIO_WCTL, 0);
  writeBufferLength=4;
  writeBuffer[0] = (char) rand() * 4 % 200;
  writeBuffer[1] = 0x45;
  writeBuffer[2] = 0x46;
  writeBuffer[3] = 0x47;
  writeBuffer[4] = 0x48;
  naI2CEE_Write(addr, writeBuffer[0], &writeBuffer[1], writeBufferLength);
  writeBufferLength=4;
  memset(readBuffer, 0, READ_BUFFER_LENGTH);

  naI2CEE_Read(addr, readBuffer, writeBuffer[0], 4);
  naGpioSetOutput(GPIO_1, 0);
  naGpioSetOutput(GPIO_2, 0);
  naGpioSetOutput(GPIO_3, 0);
  naGpioSetOutput(GPIO_4, 0);
  naGpioSetOutput(GPIO_5, 0);
  naGpioSetOutput(GPIO_6, 0);
  naGpioSetOutput(GPIO_7, 0);
  naGpioSetOutput(GPIO_8, 0);
  naGpioSetOutput(GPIO_9, 0);
  naGpioSetOutput(GPIO_10, 0);
  naGpioSetOutput(GPIO_11, 0);
  naGpioSetOutput(GPIO_12, 0);
  naGpioSetOutput(GPIO_13, 0);
  naGpioSetOutput(GPIO_14, 0);
  //call the length function, returns the length in bits
  char length = sendLength(bytes);
  //test this portion should reDefine the GPIO functions
  //you can find GPIO names in nagpio in: include/netos71/src/bsp/platforms/connectcorewi9c
  naGpioSetOutputValue(GPIO_1, 1);
  naGpioSetOutputValue(GPIO_2, length[1]);
  naGpioSetOutputValue(GPIO_3, length[2]);
  naGpioSetOutputValue(GPIO_4, length[3]);
  naGpioSetOutputValue(GPIO_5, length[4]);
  naGpioSetOutputValue(GPIO_6, length[5]);
  naGpioSetOutputValue(GPIO_7, length[6]);
  naGpioSetOutputValue(GPIO_8, length[7]);
  naGpioSetOutputValue(GPIO_9, length[8]);
  naGpioSetOutputValue(GPIO_10, length[9]);
  naGpioSetOutputValue(GPIO_11, length[10]);
  naGpioSetOutputValue(GPIO_12, length[11]);
  naGpioSetOutputValue(GPIO_13, length[12]);
  naGpioSetOutputValue(GPIO_14, length[13]);
  //wait for a bit
naGpioSetOutputValue(GPIO_1, 0);
int bits = bytes*8-13;
// wait for a bit
for (int i = 1; i <=bits; i+13) {
printf(“setting GPIOs\n”);
naGpioSetOutputValue(GPIO_1, 1);
naGpioSetOutputValue(GPIO_2, retValue[i]);
naGpioSetOutputValue(GPIO_3, retValue[i+1]);
naGpioSetOutputValue(GPIO_4, retValue[i+2]);
naGpioSetOutputValue(GPIO_5, retValue[i+3]);
naGpioSetOutputValue(GPIO_6, retValue[i+4]);
naGpioSetOutputValue(GPIO_7, retValue[i+5]);
naGpioSetOutputValue(GPIO_8, retValue[i+6]);
naGpioSetOutputValue(GPIO_9, retValue[i+7]);
naGpioSetOutputValue(GPIO_10, retValue[i+8]);
naGpioSetOutputValue(GPIO_11, retValue[i+9]);
naGpioSetOutputValue(GPIO_12, retValue[i+10]);
naGpioSetOutputValue(GPIO_13, retValue[i+11]);
naGpioSetOutputValue(GPIO_14, retValue[i+12]);
// wait
naGpioSetOutputValue(GPIO_1, 0);
// wait
}
#endif
#endif
#endif

void testTwo (unsigned long threadInput)
{
    int ret;
    /* Code to start the user application goes here. */
    printf (“Running thread #2\n”);
#if defined (BSP_ARM9)
    #ifdef NA_CC9C_OLD_CARRIER
        ret = naGpioSetOutput(GPIO_I2C_LED_N0, 1);
        if (ret) {
            printf(“naGpioSetOutput(GPIO_I2C_LED_N0, 1) returns %d\n”, ret);
            naGpioSetOutput(GPIO_I2C_LED_N1, 1);
            naGpioSetOutput(GPIO_I2C_LED_N2, 1);
            naGpioSetOutput(GPIO_I2C_LED_N3, 1);
            naGpioSetOutput(GPIO_I2C_LED_N4, 1);
            naGpioSetOutput(GPIO_I2C_LED_N5, 1);
            naGpioSetOutput(GPIO_I2C_LED_N6, 1);
            naGpioSetOutput(GPIO_I2C_LED_N7, 1);
        }
    #else
        ret = naGpioSetOutput(GPIO_I2C_0, 1);
        if (ret) {
            printf(“naGpioSetOutput(GPIO_I2C_0, 1) returns %d\n”, ret);
        }
        ret = naGpioSetOutput(GPIO_I2C_1, 1);
        if (ret) {
            printf(“naGpioSetOutput(GPIO_I2C_1, 1) returns %d\n”, ret);
        }
        naGpioSetOutput(GPIO_I2C_2, 1);
        if (ret) {
            printf(“naGpioSetOutput(GPIO_I2C_2, 1) returns %d\n”, ret);
        }
        naGpioSetOutput(GPIO_I2C_3, 1);
        if (ret) {
            printf(“naGpioSetOutput(GPIO_I2C_3, 1) returns %d\n”, ret);
        }
        naGpioSetOutput(GPIO_I2C_4, 1);
        if (ret) {
            printf(“naGpioSetOutput(GPIO_I2C_4, 1) returns %d\n”, ret);
        }
    #endif
#else

#endif

```
naGpioSetOutput(GPIO_I2C_5, 1);
if (!ret)
    printf("naGpioSetOutput(GPIO_I2C_5, 1) returns %d.\n", ret);

naGpioSetOutput(GPIO_I2C_6, 1);
if (!ret)
    printf("naGpioSetOutput(GPIO_I2C_6, 1) returns %d.\n", ret);

naGpioSetOutput(GPIO_I2C_7, 1);
if (!ret)
    printf("naGpioSetOutput(GPIO_I2C_7, 1) returns %d.\n", ret);
#endif
#endif

ret = naGpioSetOutput(GPIO_ERROR_LED, 1);
if (!ret)
    printf("naGpioSetOutput(GPIO_ERROR_LED, 1) returns %d.\n", ret);

for (;;)
{
    tx_thread_sleep(5);
    ret = naGpioSetOutputValue(GPIO_ERROR_LED, 1);
    if (!ret)
        printf("naGpioSetOutputValue returns %d. \n", ret);
    tx_thread_sleep(50);
    ret = naGpioSetOutputValue(GPIO_ERROR_LED, 0);
    if (!ret)
        printf("naGpioSetOutputValue returns %d. \n", ret);
#endif
#if defined (BSP_ARM9)
#if define NA_CC9C_OLD_CARRIER
    ret = naGpioSetOutputValue(GPIO_I2C_LED_N3, 0);
    if (!ret)
        printf("naGpioSetOutputValue returns %d. \n", ret);
    tx_thread_sleep(5);
    ret = naGpioSetOutputValue(GPIO_I2C_LED_N3, 1);
    if (!ret)
        printf("naGpioSetOutputValue returns %d. \n", ret);
#else
    ret = naGpioSetOutputValue(GPIO_USER_LED1, 0);
    if (!ret)
        printf("naGpioSetOutputValue returns %d. \n", ret);
    tx_thread_sleep(50);
    ret = naGpioSetOutputValue(GPIO_USER_LED1, 1);
    if (!ret)
        printf("naGpioSetOutputValue returns %d. \n", ret);
#endif
#endif

Root.cxx
#endif NETOS_GNU_TOOLS
#include <stdio.h>
#else
#include <iostream>
#endif
#include "appdown.h"
#include "CSocket.h"
/* UDP Echo server thread function */
void UdpEchoThread(unsigned long port);
BOOL UdpEchoTest(unsigned short port);

/* Set this to 1 to run the system POST tests during startup. */
const int APP_POST = 0;

/* Set this to 1 to run the manufacturing burn in tests. */
int APP_BURN_IN_TEST = 0;

/* Maximum number of UDP echo server threads */
#define NPORTS 4

/* Local ports */
unsigned long port[NPORTS] = {10000,10001,10002,10003};

/* Thread size */
#define STACK_SIZE 4096

/* Thread control blocks */
TX_THREAD udp_echo_thread[NPORTS];

/* Global C++ object to determine startup time */
ApplicationDownCnt app_down;

extern "C"
void applicationTcpDown (void)
{
    /* Count number of seconds. */
    app_down.Update();
}

/* Function: void applicationStart (void) */
/* Description:
   * This routine is responsible for starting the user application. It should
   * create any threads or other resources the application needs.
*/

ThreadX, the NET+OS device drivers, and the TCP/IP stack will be running when this function is called. This function uses global C++ object: PrintApplicationDown app_down.

Parameters: none
Return Values: none

extern "C"
void applicationStart (void)
{
    void *stack;
    char *app_name;
    int rc,prio;
    int i;
    #ifdef NETOS_GNU_TOOLS
        using namespace std;
    #endif
    /* Print how long it took to start */
    app_down.Name(app_name);
    #ifndef NETOS_GNU_TOOLS
        printf("Application %s started in %d secs\n", app_name, (int)app_down);
    #else
        cout << "Application " << app_name << " started in " << (int)app_down << " secs" <<endl<<endl;
    #endif
    delete [] app_name;

    /* Start UDP echo server threads */
    for (i = 0; i < NPORTS; i++)
    {
        stack = malloc (STACK_SIZE);
        if (stack == NULL)
        { netosFatalError ("Unable to allocate thread stack.", 5, 5); }
        prio = APP_DEFAULT_API_PRIORITY;
        rc = tx_thread_create (&udp_echo_thread[i], /* control block for thread*/
            "UDP Echo Thread", /* thread name*/
            UdpEchoThread, /* entry function*/
            port[i], /* parameter - port*/
            stack, /* start of stack*/
            STACK_SIZE, /* size of stack*/
            prio, /* priority*/
            prio, /* preemption threshold */
            1, /* time slice threshold*/
            TX_AUTO_START); /* start immediately*/
        if (rc != TX_SUCCESS)
        { netosFatalError ("Unable to create thread.", 5, 5); }
    }
    tx_thread_suspend(tx_thread_identify());
}

/*
 * Function: void UdpEchoThread(unsigned long port)
 *
 * Description:
 * This is the UDP Echo thread function.
 * It creates a socket and sends back received datagrams.
 * This function uses C++ local object:
 * object CSocket so.
Parameters: port - thread input, indicating the port to bind a socket to.
Return Values: none*/

```c
void UdpEchoThread(unsigned long port) {
    CSocket so;
    char buffer[1500];
    int dgram_cnt = 0;
    #ifdef NETOS_GNU_TOOLS
        using namespace std;
    #endif
    #ifdef NETOS_GNU_TOOLS
        printf("UDP Echo Started on Port %d\n", port);
    #else
        cout << "UDP Echo Started on Port " << port << endl;
    #endif
    int bufsize = 8192;
    /* Create a UDP socket */
    if (!so.Create(SOCK_DGRAM, htons((unsigned short) port))) {
        #ifdef NETOS_GNU_TOOLS
            printf("%d: Create Error %d\n", port, getErrno());
        #else
            cout << port <<": Create Error " << getErrno() << endl;
        #endif
        return;
    }
    if (!so.SetSockOpt(SO_SNDBUF, &bufsize, sizeof bufsize, SOL_SOCKET)) {
        #ifdef NETOS_GNU_TOOLS
            printf("%d: SetSockOpt Error %d\n", port, getErrno());
        #else
            cout << port <<": SetSockOpt Error " << getErrno() << endl;
        #endif
        return;
    }
    if (!so.SetSockOpt(SO_RCVBUF, &bufsize, sizeof bufsize, SOL_SOCKET)) {
        #ifdef NETOS_GNU_TOOLS
            printf("%d: SetSockOpt Error %d\n", port, getErrno());
        #else
            cout << port <<": SetSockOpt Error " << getErrno() << endl;
        #endif
        return;
    }
    /* Echo received datagrams */
    while (1) {
        char *pbuffer = &buffer[0];
        unsigned long peer_addr;
        unsigned short peer_port;
        int cnt;
        char *retval = "You Sent "; //found this line prints character before data is returned
        //get that data after to go to GPIO's
        if ((cnt = so.ReceiveFrom (pbuffer, 1500, peer_port, peer_addr)) < 0) {
            #ifdef NETOS_GNU_TOOLS
                printf("%d: ReceiveFrom Error %d\n", port, getErrno());
            #else
                cout << port <<": ReceiveFrom Error " << getErrno() << endl;
            #endif
        }
    }
}
```
cout << port << ": ReceiveFrom Error " << getErrno() << endl;
#endif
return;
}

if (cnt > 0) {
    //GPIO::setGPIO();
    setGPIO(pbuffer, cnt);
}
if ((cnt = so.SendTo (retval, cnt, peer_port, peer_addr)) < 0)
{
    #ifndef NETOS_GNU_TOOLS
        printf("%d: SendTo %d,%d Error %d\n",port,peer_addr,peer_port,getErrno());
    #else
        cout << port << ": SendTo " << peer_addr << "," << peer_port << " Error " << getErrno() << endl;
    #endif
    return;
}
if ((++dgram_cnt % 1000) == 0)
{
    #ifndef NETOS_GNU_TOOLS
        printf("Received %d Datagrams on Port %d\n",dgram_cnt,port);
    #else
        cout << "Received " << dgram_cnt << " Datagrams on Port " << port << endl;
    #endif
}

---

Root.c

#include <stdio.h>
#include <stdlib.h>
#include <tx_api.h>
#include <sockapi.h>
#include <appservices.h>
#include "bsp_api.h"
#include "pthread.h"
#include "appconf.h"
#include "server2.h"

#define STACK_SIZE 8192
TX_THREAD udp_server;
extern void server(ULONG thread_input);

/* Set this to 1 to run the manufacturing burn in tests. */
int APP_BURN_IN_TEST = 0;

int applicationTcpDown (void)
{
    static int ticksPassed = 0;
    ticksPassed++;

    /* Function: void applicationTcpDown (void)
     * Description:
     * This routine will be called by the NET+OS root thread once every
     * clock tick while it is waiting for the TCP/IP stack to come up.
     * This function can increment a counter everytime it's called to
     * keep track of how long we've been waiting for the stack to start.
     * If we've been waiting too long, then this function can do something
     * to handle the error.
     * This function will not be called once the stack has started.
     * Parameters: none
     * Return Values: none
     */

    void applicationTcpDown (void)
    {
        static int ticksPassed = 0;
        ticksPassed++;
void applicationStart (void)
{
    /* Initialize the system services for the application. */
    initAppServices();
    printf("%s Ready\n", APP_DIALOG_APP_NAME);
    int result;
    void *stack;
    stack = malloc(STACK_SIZE);
    if (stack == NULL) {
        printf("UDP Thread Stack failure.\n");
        return;
    }
    printf("Creating UDP thread\n");
    result = tx_thread_create(    &udp_server,    //pointer to thread location
        "UDP Server Thread",  //name of thread
        main2,    //thread entry function
        0,    //input passed to entry function
        stack,    //pointer to stack location in memory
        STACK_SIZE,    //stack size
        NABspMediumPriority,  //priority
        NABspMediumPriority,    //preemption threshold
        5,    //time slices
        TX_AUTO_START); //thread starts immediately
    if (result != TX_SUCCESS) {
        free(stack);
        printf("UDP thread create failure.\n");
        return;
    }
    printf("Thread execution complete, now suspended\n");
}

Send.c
#include <stdio.h>
#include <stdlib.h>
#include "tx_api.h"
#include "bsp_api.h"
#include "sockapi.h"
#include "eth_api.h"
#include "nabypassapp.h"
#include "trsocket.h"
#include "iam.hh"
define APP_EVENT  1

static char appSendThreadStack[4096];
static TX_THREAD appSendThread;
static TX_EVENT_FLAGS_GROUP appEvents;

#if BSP_ENABLE_ETHERNET_BYPASS == FALSE
static void appSendTask (ULONG arg) {
    char *dataBuffer;
    unsigned long eventFlags;
    struct sockaddr_in sendAddr;
    int result, sockFd, recvDataSize, sendDataSize;
    appSocketDataType *sockData = (appSocketDataType *) arg;

    /* Get send address and data size from the argument */
    sendAddr = sockData->destAddr;
    /* Send */
    result = tx_socket_sendto(sockFd, sendDataSize, sendAddr, &dataBuffer);
    if (result < sendDataSize) {
        sockData->destAddr = sendAddr;
        sockData->sendDataSize = sendDataSize - result;
        appSendTask(arg);
    }
}
#endif

recvDataSize = sockData->sendSize;

/* open a UDP socket */
sockFd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockFd < 0)
{
    printf("naBypassApp: socket failed, errno = %d\n", getErrno());
    return;
}

/* Disable UDP checksum */
tfSetTreckOptions(TM_OPTION_UDP_CHECKSUM, 0);

/* Allocate a data buffer */
dataBuffer = malloc(APP_BUFFER_SIZE);
if (dataBuffer == NULL)
{
    printf("naBypassApp: allocation failure\n");
    return;
}

/* Assign a client port to send data to */
sendAddr.sin_port = htons(APP_CLIENT_PORT);
while (1)
{
    /* Send a packet */
    sendDataSize = sendto(sockFd, dataBuffer, recvDataSize, 0, (struct sockaddr *) &sendAddr, sizeof sendAddr);
    if (sendDataSize != recvDataSize)
    {
        printf("naBypassApp: sendto returned %d, errno = %d\n", sendDataSize, getErrno());
    }
}

#else
#define ETH_HEADER_LENGTH   14
#define IP_HEADER_LENGTH    20
#define UDP_HEADER_LENGTH   8
#define MAX_IP_PACKET_SIZE  1500

/* All offsets are in bytes */
#define IP_VER_OFFSET       0
#define IP_TTL_OFFSET       8
#define IP_PROT_OFFSET      9
#define IP_SRCADDR_OFFSET   12
#define IP_DESTADDR_OFFSET  16
#define IP_IDENT_OFFSET     4
#define IP_LEN_OFFSET       2
#define IP_CHECKSUM_OFFSET  10
#define IP_VER_HLEN         0x4500
#define MY_TTL              60

#define UDP_LENGTH_OFFSET   (IP_HEADER_LENGTH + 4)
#define UDP_CHECKSUM_OFFSET  (IP_HEADER_LENGTH + 6)
#define IP_VER_HLEN         0x4500
#define MY_TTL              60

/* Should not really use arbitrary IP identifier, might be a conflict with TCP/IP stack */
#define IP_IDENTIFIER       0x1234

static naNetMsgPtr sendPacketHandles[NUM_BYPASS_TX_BUFFERS];
static void initSendBuffer(char *buffer,
                struct sockaddr_in *destAddr,
                struct sockaddr_in *srcAddr,
                char *destMac)
{
    unsigned short *pshort = (unsigned short *) buffer;
    unsigned char  *pchar  = (unsigned char  *) buffer;

    memcpy(buffer, destMac, MAC_ADDR_LENGTH);       /* Init MAC address */
    pshort[ETH_TYPE_OFFSET/2] = ETH_TYPE_IP;        /* Ethernet type == IP */

    pchar += ETH_HEADER_LENGTH;
    pshort = (unsigned short *) pchar;

    memset(pchar, 0, IP_HEADER_LENGTH + UDP_HEADER_LENGTH);
    pshort[IP_VER_OFFSET/2]   = IP_VER_HLEN;       /* Version, IP Header len */
    pchar[IP_TTL_OFFSET]  = (unsigned char) MY_TTL; /* TTL */
    pchar[IP_PROT_OFFSET] = IP_TYPE_UDP;            /* IP type == UDP */
    memcpy(&pchar[IP_SRCADDR_OFFSET], &srcAddr->sin_addr , 4); /* Source IP address */
    memcpy(&pchar[IP_DESTADDR_OFFSET], &destAddr->sin_addr, 4); /* Dest.  IP address */
    pshort += IP_HEADER_LENGTH/2;
    *pshort++ = srcAddr->sin_port;                  /* Source port */
    *pshort   = destAddr->sin_port;                 /* Dest   port */
}

static void txComplete(naNetMsgPtr msgHandle)
{
    char   * dataPtr;
    unsigned length;

    if (naNetMsgGetDataPtr(msgHandle, &dataPtr, &length) == ETH_SUCCESS)
    {
        unsigned short * ipheader = (unsigned short *)(dataPtr + IP_HEADER_LENGTH);
        ipheader[IP_IDENT_OFFSET/2] = 0;
    }
}

static unsigned long appNobufferCnt = 0;

static naNetMsgPtr getAvailBuffer(void)
{
    static int       bufferIndex = 0;
    unsigned short * ipHeader;
    char   * dataPtr;
    unsigned length;

    naNetMsgPtr msgHandle = sendPacketHandles[bufferIndex];
    if (naNetMsgGetDataPtr(msgHandle, &dataPtr, &length) != ETH_SUCCESS)
        return NULL;
    return NULL;
}
ipHeader = (unsigned short *)(dataPtr + IP_HEADER_LENGTH);
while (ipHeader[IP_IDENT_OFFSET/2])
{
    appNobufferCnt++;
    tx_thread.sleep(1);
}

if (++bufferIndex == NUM_BYPASS_TX_BUFFERS)
    bufferIndex = 0;

return msgHandle;

/*
 * Allocate and initialize packets and then run a loop to send a packet
 * on a timer event. Reuse preallocated packets
 */
static void appSendTask (ULONG arg) {
    unsigned short      ipIdent = IP_IDENTIFIER;
    struct sockaddr_in  srcAddr;
    unsigned short      ipPacketSize;
    NaIamIfAddrInfo_t   addrInfo;
    int  i, result, dataSize;

    appSocketDataType  *sockData = (appSocketDataType *) arg;
    /* extract data size from the argument */
    /* calculate the size of IP packet */
    ipPacketSize = (unsigned short) dataSize + IP_HEADER_LENGTH + UDP_HEADER_LENGTH;

    /* Use APP_CLIENT_PORT as dest. port */
    * Dest. IP address in already in sockData->destAddr
    */
    sockData->destAddr.sin_port = htons(APP_CLIENT_PORT);

    /* Use IP address of eth0 as source IP address */
    if (customizeIamGetIfAddrInfo("eth0", &addrInfo) == NA_IAM_STATUS_SUCCESS &&
        addrInfo.v4Info.IPV4_ADDR(ipAddress))
    {
        srcAddr.sin_addr.s_addr = addrInfo.v4Info.IPV4_ADDR(ipAddress);
    }
    else
    {
        printf("naBypassApp: IPv4 address is not configured on eth0\n");
        return;
    }

    /* Use APP_SEND_PORT as source port */
    sockAddr.sin_port = htons(APP_SEND_PORT);

    /* Allocate and initialize send packets. In this example packets are reused. */
    for (i = 0; i < NUM_BYPASS_TX_BUFFERS; i++)
    {
        char *data_ptr = NULL;
        unsigned length;
        /* Allocate a packet of a send size. */
        sendPacketHandles[i] = naNetMsgAllocate(ipPacketSize + ETH_HEADER_LENGTH);
        if (sendPacketHandles[i] == NULL)
        {
            printf("naBypassApp: Memory allocation error \n");
            return;
        }
        /* Get a data pointer of a packet to initialize packet data. */
        The data pointer must point to the Ethernet header when packet

* is transmitted.

if (naNetMsgGetDataPtr(sendPacketHandles[i], &data_ptr, &length) != ETH_SUCCESS)
{
    printf("naBypassApp: naNetMsgGetDataPtr failed \n");
    return;
}
/* Initialize packet data area that is the same for every packet*/
initSendBuffer(data_ptr, &sockData->destAddr, &srcAddr, sockData->destMacAddr);
}

while (1)
{
    unsigned short * ipheader;
    unsigned short checksum;
    char *data_ptr = NULL;
    unsigned length;

    /* Get available packet */
    naNetMsgPtr msg_handle = getAvailBuffer();

    /* Get a data pointer to update packet data */
    if (naNetMsgGetDataPtr(msg_handle, &data_ptr, &length) != ETH_SUCCESS)
    {
        printf("naBypassApp: naNetMsgGetDataPtr failed \n");
        return;
    }
    /* Initialize fields that change with every packet */
    ipheader = (unsigned short *)(data_ptr + ETH_HEADER_LENGTH);

    /* Write IP identifier and IP size fields */
    ipheader[IP_IDENT_OFFSET/2] = ipIdent++;
    ipheader[IP_LEN_OFFSET/2] = ipPacketSize;

    /* Increase IP identifier */
    if (ipIdent == 0) ipIdent++;

    /* Write UDP size field */
    ipheader[UDP_LENGTH_OFFSET/2] = dataSize + UDP_HEADER_LENGTH; /* UDP length */

    /* Don't do UDP checksum - zero it */
    ipheader[UDP_CHECKSUM_OFFSET/2] = 0;

    /* Calculate IP header checksum */
    checksum = ~cksum(ipheader, IP_HEADER_LENGTH);
    if (checksum == 0xFFFF)
    {
        checksum = 0;
    }
    ipheader[IP_CHECKSUM_OFFSET/2] = checksum;

    /* Send a packet. txComplete would be called after the packet is transmitted.
     * This particular txComplete would write 0 to the IP identifier to mark a
     * packet as available.
     */
    result = naEthBypassTransmit(msg_handle, txComplete);
    if (result)
    {
        printf("naBypassApp: NAEthBypassTransmit failure: errno: \n");
    }
}
#endif
/* Create send thread */
void appSend (void *uarg)
{
  int result;
  memset (&appSendThread, 0, sizeof appSendThread);
  /* first create the idle thread */
  result = tx_thread_create (&appSendThread, /* pointer to thread */
    "SendThread", /* name */
    appSendTask, (ULONG)uarg, /* entry function and input */
    (VOID *) appSendThreadStack, /* pointer to thread stack */
    sizeof(appSendThreadStack), /* stack size */
    2, 2, /* priority and preempt threshold */
    1, TX_AUTO_START); /* time slice and auto start */
  if (result != TX_SUCCESS)
  {
    printf("naBypassApp: failed to create idle thread!\n");
  }
}

Server2.h
#ifndef SERVER2_H_
#define SERVER2_H_
#include <sockapi.h>
#include <appservices.h>
#include <stdio.h>
#include "pthread.h"

void main2(unsigned long thread_input);

#endif /*SERVER2_H_*/

Server2.c
#include <sockapi.h>
#include <appservices.h>
#include <stdio.h>
#include "pthread.h"

main2() {
  int sd;
  struct sockaddr_in server;
  char buf[512];
  int rc, nb;
  fd_set read_set;
  struct timeval wait;

  server.sin_family = AF_INET;
  server.sin_addr.s_addr = htonl(INADDR_ANY);
  server.sin_port = htons(12345);

  sd = socket (AF_INET,SOCK_DGRAM,IPPROTO_UDP);
  //bind(UDPsocket, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) < 0
  bind ( sd, (struct sockaddr *) &server, sizeof(server));

  // for (;;) {
    //    rc = recv (sd, buf, sizeof(buf), 0);
    //    buf[rc] = (char) NULL;
    //    printf("Received: %s\n", buf);
nb = select (FD_SETSIZE, &read_set, (fd_set *) 0, (fd_set *) 0, &wait);
printf("Received: %sn", nb);

Starting loop
for (;;) {
    wait.tv_sec = 1;
    wait.tv_usec = 0;
    FD_ZERO (&read_set);
    FD_SET (sd, &read_set);
    nb = select (FD_SETSIZE, &read_set, (fd_set *) 0, (fd_set *) 0, &wait);
    if (nb <= 0)
        printf("error occurred\n");
    if (FD_ISSET(sd, &read_set))
        printf("Received data\n");
}
15.6. Video Processing Code

evmdm355_gpio.h

/*
 *  Copyright 2005 by Spectrum Digital Incorporated.
 *  All rights reserved. Property of Spectrum Digital Incorporated.
 */

/*
 *  GPIO header file
 */

#ifndef GPIO_
#define GPIO_

#include "evmdm355.h"

#define GPIO_IN                 1
#define GPIO_OUT                0

#define GPIO0                   0x00
#define GPIO1                   0x01
#define GPIO2                   0x02
#define GPIO3                   0x03
#define GPIO4                   0x04
#define GPIO5                   0x05
#define GPIO6                   0x06
#define GPIO7                   0x07
#define GPIO8                   0x08
#define GPIO9                   0x09
#define GPIO10                  0x0A
#define GPIO11                  0x0B
#define GPIO12                  0x0C
#define GPIO13                  0x0D
#define GPIO14                  0x0E
#define GPIO15                  0x0F
#define GPIO16                  0x10
#define GPIO17                  0x11
#define GPIO18                  0x12
#define GPIO19                  0x13
#define GPIO20                  0x14
#define GPIO21                  0x15
#define GPIO22                  0x16
#define GPIO23                  0x17
#define GPIO24                  0x18
#define GPIO25                  0x19
#define GPIO26                  0x1A
#define GPIO27                  0x1B
#define GPIO28                  0x1C
#define GPIO29                  0x1D
#define GPIO30                  0x1E
#define GPIO31                  0x1F
#define GPIO32                  0x20
#define GPIO33                  0x21
#define GPIO34                  0x22
#define GPIO35                  0x23
#define GPIO36                  0x24
#define GPIO37                  0x25
#define GPIO38                  0x26
#define GPIO39                  0x27
#define GPIO40                  0x28
#define GPIO41                  0x29
#define GPIO42                  0x2A
#define GPIO43                  0x2B
#define GPIO44                  0x2C
#define GPIO45                  0x2D
```c
#define GPIO46                  0x2E
#define GPIO47                  0x2F
#define GPIO48                  0x30
#define GPIO49                  0x31
#define GPIO50                  0x32
#define GPIO51                  0x33
#define GPIO52                  0x34
#define GPIO53                  0x35
#define GPIO3V3_0               0x36
#define GPIO3V3_1               0x37
#define GPIO3V3_2               0x38
#define GPIO3V3_3               0x39
#define GPIO3V3_4               0x3A
#define GPIO3V3_5               0x3B
#define GPIO3V3_6               0x3C
#define GPIO3V3_7               0x3D
#define GPIO3V3_8               0x3E
#define GPIO3V3_9               0x3F
#define GPIO3V3_10              0x40
#define GPIO3V3_11              0x41
#define GPIO3V3_12              0x42
#define GPIO3V3_13              0x43
#define GPIO3V3_14              0x44
#define GPIO3V3_15              0x45

/* ------------------------------------------------------------------------ *
*  Prototypes                                                              *
* ------------------------------------------------------------------------ */
int16 EVMDM355_GPIO_init( );
int16 EVMDM355_GPIO_setDirection ( Uint16 number, Uint8 direction );
int16 EVMDM355_GPIO_setOutput    ( Uint16 number, Uint8 output );
int16 EVMDM355_GPIO_getInput     ( Uint16 number );
#endif

evmdm355_gpio.c

/*
 * Copyright 2005 by Spectrum Digital Incorporated.
 * All rights reserved. Property of Spectrum Digital Incorporated.
 */

/*
 * GPIO implementation
 */

#include "evmdm355_gpio.h"

/* ------------------------------------------------------------------------ *
*                                                                            *
*  _GPIO_init( )                                                            *
*                                                                            *
* ------------------------------------------------------------------------ */
int16 EVMDM355_GPIO_init()
{
    /* Free GPIO from emulation */
    GPIO_PCR = 1;
    return 0;
}

/* ------------------------------------------------------------------------ *
*                                                                            *
*  _GPIO_setDirection( number, direction )                                 *
*                                                                            *
* number    <- GPIO#                                                      *
* direction <- 0:OUT 1:IN                                                  *
*                                                                            *
* ------------------------------------------------------------------------ */
```
Int16 EVMDM355_GPIO_setDirection( Uint16 number, Uint8 direction )
{
    Uint32 bank_id = ( number >> 5 );
    Uint32 pin_id = ( 1 << ( number & 0x1F ) );
    Uint32* gpio_dir = ( Uint32* )( GPIO_BASE + GPIO_DIR_BASE + ( bank_id * GPIO_BASE_OFFSET ) );
    if ( ( direction & 1 ) == GPIO_OUT )
        *gpio_dir &= ~( pin_id ); // Set to OUTPUT
    else
        *gpio_dir |= ( pin_id ); // Set to INPUT
    return 0;
}

/* ------------------------------------------------------------------------ */
/*                                                                          */
/*  _GPIO_setOutput( number, output )                                       */
/*                                                                          */
/*      number   <- GPIO#                                                   */
/*      value    <- 0:LOW 1:HIGH                                            */
/*                                                                          */
/* ------------------------------------------------------------------------ */
Int16 EVMDM355_GPIO_setOutput( Uint16 number, Uint8 output )
{
    Uint32 bank_id = ( number >> 5 );
    Uint32 pin_id = ( 1 << ( number & 0x1F ) );
    Uint32* gpio_out = ( Uint32* )( GPIO_BASE + GPIO_OUT_DATA_BASE + ( bank_id * GPIO_BASE_OFFSET ) );
    if ( ( output & 1 ) == 0 )
        *gpio_out &= ~( pin_id ); // Set to LOW
    else
        *gpio_out |= ( pin_id ); // Set to HIGH
    return 0;
}

/* ------------------------------------------------------------------------ */
/*                                                                          */
/*  _GPIO_getInput( number )                                                */
/*                                                                          */
/*      number   <- GPIO#                                                   */
/*                                                                          */
/*      Returns:    0:LOW                                                   */
/*                  1:HIGH                                                  */
/*                                                                          */
/* ------------------------------------------------------------------------ */
Int16 EVMDM355_GPIO_getInput( Uint16 number )
{
    Uint32 input;
    Uint32 bank_id = ( number >> 5 );
    Uint32 pin_id = ( number & 0x1F );
    Uint32* gpio_in = ( Uint32* )( GPIO_BASE + GPIO_IN_DATA_BASE + ( bank_id * GPIO_BASE_OFFSET ) );
    input = *gpio_in;
    input = ( input >> pin_id ) & 1;
    return input;
}

evmdm355.c

/*
 * Copyright 2007 by Spectrum Digital Incorporated.
 * All rights reserved. Property of Spectrum Digital Incorporated.
 */
/*
 * Board Setup
*/
#include "evmdm355.h"
#include "evmdm355_i2c.h"

/* _wait( delay )
   * Wait in a software loop for 'x' delay
   */
void _wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ ){ };
}

/* _waitusec( usec )
   * Wait in a software loop for 'x' microseconds
   */
void _waitusec( Uint32 usec )
{
    _wait( usec * 8 );
}

/* EVMDM355_init( )
   * Setup I2C, MSP430, & I2C GPIO Expander
   */
Int16 EVMDM355_init( )
{
    /* PSC, PLL, DDR, & AEMIF are already configured in the GEL file. Those
     * settings are not re-initialized here */
    EVMDM355_I2C_init( ); // Setup I2C
    return 0;
}

evmdm355.h

/* Copyright 2007 by Spectrum Digital Incorporated.
   * All rights reserved. Property of Spectrum Digital Incorporated.
   */

/* Definitions & Register
   */
#ifndef EVMDM355_
#define EVMDM355_

/* Variable types
   */
#define Uint32 unsigned int
#define Uint16 unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char

/* ------------------------------------------------------------------------ *
*                                                                          *
*  Software Breakpoint code                                                *
*      Uses inline assembly command                                        *
*                                                                          *
* ------------------------------------------------------------------------ */
#define SW_BREAKPOINT    asm( " .long 0xE1200070" );

/* ------------------------------------------------------------------------ *
*                                                                          *
*  AEMIF Controller                                                        *
*                                                                          *
* ------------------------------------------------------------------------ */
#define AEMIF_BASE              0x01E10000
#define AEMIF_AWCCR             *( volatile Uint32* )( AEMIF_BASE + 0x04)
#define AEMIF_A1CR              *( volatile Uint32* )( AEMIF_BASE + 0x10)
#define AEMIF_A2CR              *( volatile Uint32* )( AEMIF_BASE + 0x14)
#define AEMIF_A3CR              *( volatile Uint32* )( AEMIF_BASE + 0x18)
#define AEMIF_A4CR              *( volatile Uint32* )( AEMIF_BASE + 0x1C)
#define AEMIF_B1RR               *( volatile Uint32* )( AEMIF_BASE + 0x40)
#define AEMIF_B1MR               *( volatile Uint32* )( AEMIF_BASE + 0x44)
#define AEMIF_EIMSR              *( volatile Uint32* )( AEMIF_BASE + 0x48)
#define AEMIF_E1MCR              *( volatile Uint32* )( AEMIF_BASE + 0x4C)
#define AEMIF_NANDPCR            *( volatile Uint32* )( AEMIF_BASE + 0x60)
#define AEMIF_NANDFSR            *( volatile Uint32* )( AEMIF_BASE + 0x64)
#define AEMIF_NANDECC2           *( volatile Uint32* )( AEMIF_BASE + 0x70)
#define AEMIF_NANDECC3           *( volatile Uint32* )( AEMIF_BASE + 0x74)
#define AEMIF_NANDECC4           *( volatile Uint32* )( AEMIF_BASE + 0x78)
#define AEMIF_NANDECC5           *( volatile Uint32* )( AEMIF_BASE + 0x7C)

#define AEMIF_MAX_TIMEOUT_8BIT  0x3FFFFFFC
#define AEMIF_MAX_TIMEOUT_16BIT 0x3FFFFFFD

#define EMIF_CS2                2
#define EMIF_CS3                3
#define EMIF_CS4                4
#define EMIF_CS5                5
#define EMIF_CS0_BASE           0x02000000
#define EMIF_CS1_BASE           0x04000000
#define EMIF_NAND_MODE          1
#define EMIF_NORMAL_MODE        0

/* ------------------------------------------------------------------------ *
*                                                                          *
*  Device System Controller                                                *
*                                                                          *
* ------------------------------------------------------------------------ */
#define SYSTEM_BASE         0x01C40000
#define PINMUX0             *( volatile Uint32* )( SYSTEM_BASE + 0x00)
#define PINMUX1             *( volatile Uint32* )( SYSTEM_BASE + 0x04)
#define PINMUX2             *( volatile Uint32* )( SYSTEM_BASE + 0x08)
#define PINMUX3             *( volatile Uint32* )( SYSTEM_BASE + 0x0C)
#define PINMUX4             *( volatile Uint32* )( SYSTEM_BASE + 0x10)
#define BOOTCFG             *( volatile Uint32* )( SYSTEM_BASE + 0x14)
#define ARM_INTMUX          *( volatile Uint32* )( SYSTEM_BASE + 0x18)
#define EDMA_INTMUX          *( volatile Uint32* )( SYSTEM_BASE + 0x1C)
#define DDR_SLEW             *( volatile Uint32* )( SYSTEM_BASE + 0x20)
#define TIMER64_CTL         *( volatile Uint32* )( SYSTEM_BASE + 0x24)
#define DEVICE_ID           *( volatile Uint32* )( SYSTEM_BASE + 0x28)
#define VDAC_CONFIG         *( volatile Uint32* )( SYSTEM_BASE + 0x2C)
#define TIMER64_CTL         *( volatile Uint32* )( SYSTEM_BASE + 0x30)
# define USBPHY_CTL *( volatile Uint32* ) ( SYSTEM_BASE + 0x34 )
# define MISC *( volatile Uint32* ) ( SYSTEM_BASE + 0x38 )
# define MSTPRI0 *( volatile Uint32* ) ( SYSTEM_BASE + 0x3C )
# define MSTPRI1 *( volatile Uint32* ) ( SYSTEM_BASE + 0x40 )
# define VPSS_CLKCTL *( volatile Uint32* ) ( SYSTEM_BASE + 0x44 )
# define DEEPSLEEP *( volatile Uint32* ) ( SYSTEM_BASE + 0x48 )
# define DEBOUNCE0 *( volatile Uint32* ) ( SYSTEM_BASE + 0x50 )
# define DEBOUNCE1 *( volatile Uint32* ) ( SYSTEM_BASE + 0x54 )
# define DEBOUNCE2 *( volatile Uint32* ) ( SYSTEM_BASE + 0x58 )
# define DEBOUNCE3 *( volatile Uint32* ) ( SYSTEM_BASE + 0x5C )
# define DEBOUNCE4 *( volatile Uint32* ) ( SYSTEM_BASE + 0x60 )
# define DEBOUNCE5 *( volatile Uint32* ) ( SYSTEM_BASE + 0x64 )
# define DEBOUNCE6 *( volatile Uint32* ) ( SYSTEM_BASE + 0x68 )
# define DEBOUNCE7 *( volatile Uint32* ) ( SYSTEM_BASE + 0x6C )
# define VPPIOCR *( volatile Uint32* ) ( SYSTEM_BASE + 0x70 )

/* ----------------------------------------------------------------------------------------------------- */
/* DDR Controller */
/* ----------------------------------------------------------------------------------------------------- */
# define DDR_REG_BASE 0x20000000
# define DDR_DDRVTPER *( volatile Uint32* ) ( 0x01C4004C )
# define DDR_DDRVTFR *( volatile Uint32* ) ( 0x01C42038 )
# define DDR_SDRSTAT *( volatile Uint32* ) ( DDR_REG_BASE + 0x04 )
# define DDR_SDBCR *( volatile Uint32* ) ( DDR_REG_BASE + 0x08 )
# define DDR_SDRCR *( volatile Uint32* ) ( DDR_REG_BASE + 0x0C )
# define DDR_SDTIMR *( volatile Uint32* ) ( DDR_REG_BASE + 0x10 )
# define DDR_SDTIMR2 *( volatile Uint32* ) ( DDR_REG_BASE + 0x14 )
# define DDR_VBPR *( volatile Uint32* ) ( DDR_REG_BASE + 0x20 )
# define DDR_IRR *( volatile Uint32* ) ( DDR_REG_BASE + 0x20 )
# define DDR_TM0 *( volatile Uint32* ) ( DDR_REG_BASE + 0x20 )
# define DDR_TM0R *( volatile Uint32* ) ( DDR_REG_BASE + 0x20 )
# define DDR_DDRPHYCR *( volatile Uint32* ) ( DDR_REG_BASE + 0x20 )
# define DDR_VPPIOCR *( volatile Uint32* ) ( DDR_REG_BASE + 0x20 )

#define DDR_BASE 0x80000000 // Start of SDRAM range
#define DDR_SIZE 0x08000000 // 128 MB

/* ----------------------------------------------------------------------------------------------------- */
/* EDMA Controller */
/* ----------------------------------------------------------------------------------------------------- */
# define EDMA_3CC_BASE 0x01C00000
# define EDMA_3CC_CCCFG *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )

/* Global Controller */
# define EDMA_3CC_ER *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_ERH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_ECR *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_ECRH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_ESR *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_ESRH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_CER *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_CERH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_EER *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_EERH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_EECR *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_EECRH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_EESR *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_EESRH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_SER *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_SERH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_SECR *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_SECRH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_IER *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_IERH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_IECR *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
# define EDMA_3CC_IECRH *( volatile Uint32* ) ( EDMA_3CC_BASE + 0x0000 )
#define EDMA_3CC_IERSR *( volatile Uint32* )( EDMA_3CC_BASE + 0x1060 )
#define EDMA_3CC_IERSRH *( volatile Uint32* )( EDMA_3CC_BASE + 0x1064 )
#define EDMA_3CC_IPR *( volatile Uint32* )( EDMA_3CC_BASE + 0x1068 )
#define EDMA_3CC_IPRHI *( volatile Uint32* )( EDMA_3CC_BASE + 0x106C )
#define EDMA_3CC_ICR *( volatile Uint32* )( EDMA_3CC_BASE + 0x1070 )
#define EDMA_3CC_ICRHI *( volatile Uint32* )( EDMA_3CC_BASE + 0x1074 )
#define EDMA_3CC_IEVAL *( volatile Uint32* )( EDMA_3CC_BASE + 0x1078 )
#define EDMA_3CC_QER *( volatile Uint32* )( EDMA_3CC_BASE + 0x1080 )
#define EDMA_3CC_QERHI *( volatile Uint32* )( EDMA_3CC_BASE + 0x1084 )
#define EDMA_3CC_QEECR *( volatile Uint32* )( EDMA_3CC_BASE + 0x1088 )
#define EDMA_3CC_QEESR *( volatile Uint32* )( EDMA_3CC_BASE + 0x108C )
#define EDMA_3CC_QSER *( volatile Uint32* )( EDMA_3CC_BASE + 0x1090 )
#define EDMA_3CC_QSECR *( volatile Uint32* )( EDMA_3CC_BASE + 0x1094 )

#define EDMA_TC0_BASE 0x01C10000
#define EDMA_TC1_BASE 0x01C10400

/* ------------------------------------------------------------------------ *
*                                                                          *
*  GPIO Control                                                            *
*                                                                          *
* ------------------------------------------------------------------------ */
#define GPIO_BASE               0x01C67000
#define GPIO_PCR                *( volatile Uint32* )( GPIO_BASE + 0x04 )
#define GPIO_BINTEN             *( volatile Uint32* )( GPIO_BASE + 0x08 )

// For GPIO[0:31]
#define GPIO_DIR_BASE           ( 0x10 )   // Direction Cntl
#define GPIO_OUT_DATA_BASE      ( 0x14 )   // Output data
#define GPIO_SET_DATA_BASE      ( 0x18 )   // Set data
#define GPIO_CLR_DATA_BASE      ( 0x1C )   // Clear data
#define GPIO_IN_DATA_BASE       ( 0x20 )   // Input data
#define GPIO_SET_RIS_TRIG_BASE  ( 0x24 )   // Set rising edge intr
#define GPIO_CLR_RIS_TRIG_BASE  ( 0x28 )   // Clear rising edge intr
#define GPIO_SET_FAL_TRIG_BASE  ( 0x2C )   // Set falling edge intr
#define GPIO_CLR_FAL_TRIG_BASE  ( 0x30 )   // Clear falling edge intr
#define GPIO_INSTAT_BASE        ( 0x34 )   // Intr status
#define GPIO_BASE_OFFSET        0x28

#define GPIO_01_BASE            ( GPIO_BASE + 0x10 )
#define GPIO_23_BASE             ( GPIO_01_BASE + GPIO_BASE_OFFSET )
#define GPIO_45_BASE             ( GPIO_23_BASE + GPIO_BASE_OFFSET )
#define GPIO_6_BASE              ( GPIO_45_BASE + GPIO_BASE_OFFSET )

// For GPIO[32:63]
#define GPIO_DIR01 *( volatile Uint32* )( GPIO_BASE + 0x10 )
#define GPIO_OUT_DATA01 *( volatile Uint32* )( GPIO_BASE + 0x14 )
#define GPIO_SET_DATA01 *( volatile Uint32* )( GPIO_BASE + 0x18 )
#define GPIO_CLR_DATA01 *( volatile Uint32* )( GPIO_BASE + 0x1C )
#define GPIO_IN_DATA01 *( volatile Uint32* )( GPIO_BASE + 0x20 )
#define GPIO_SET_RIS_TRIG01 *( volatile Uint32* )( GPIO_BASE + 0x24 )
#define GPIO_CLR_RIS_TRIG01 *( volatile Uint32* )( GPIO_BASE + 0x28 )
#define GPIO_SET_FAL_TRIG01 *( volatile Uint32* )( GPIO_BASE + 0x2C )
#define GPIO_CLR_FAL_TRIG01 *( volatile Uint32* )( GPIO_BASE + 0x30 )
#define GPIO_INSTAT01 *( volatile Uint32* )( GPIO_BASE + 0x34 )

// For GPIO[32:63]
#define GPIO_DIR23 *( volatile Uint32* )( GPIO_BASE + 0x38 )
#define GPIO_OUT_DATA23 *( volatile Uint32* )( GPIO_BASE + 0x3C )
#define GPIO_SET_DATA23 *( volatile Uint32* )( GPIO_BASE + 0x40 )
#define GPIO_CLR_DATA23 *( volatile Uint32* )( GPIO_BASE + 0x44 )
#define GPIO_IN_DATA23 *( volatile Uint32* )( GPIO_BASE + 0x48 )
#define GPIO_SET_RIS_TRIG23 *( volatile Uint32* )( GPIO_BASE + 0x4C )
#define GPIO_CLR_RIS_TRIG23 *( volatile Uint32* )( GPIO_BASE + 0x50 )
#define GPIO_SET_FAL_TRIG23 *( volatile Uint32* )( GPIO_BASE + 0x54 )
#define GPIO_CLR_FAL_TRIG23 *( volatile Uint32* )( GPIO_BASE + 0x58 )
#define GPIO_INSTAT23 *( volatile Uint32* )( GPIO_BASE + 0x5C )

// For GPIO[64:70]
#define GPIO_DIR45          *( volatile Uint32* )( GPIO_BASE + 0x60 )
#define GPIO_SET_DATA45     *( volatile Uint32* )( GPIO_BASE + 0x64 )
#define GPIO_CLR_DATA45     *( volatile Uint32* )( GPIO_BASE + 0x68 )
#define GPIO_IN_DATA45      *( volatile Uint32* )( GPIO_BASE + 0x6C )
#define GPIO_SET_RIS_TRIG45 *( volatile Uint32* )( GPIO_BASE + 0x70 )
#define GPIO_CLR_RIS_TRIG45 *( volatile Uint32* )( GPIO_BASE + 0x74 )
#define GPIO_SET_FAL_TRIG45 *( volatile Uint32* )( GPIO_BASE + 0x78 )
#define GPIO_CLR_FAL_TRIG45 *( volatile Uint32* )( GPIO_BASE + 0x7C )
#define GPIO_INSTAT45       *( volatile Uint32* )( GPIO_BASE + 0x80 )

#define I2C_BASE            0x01C21000
#define I2C_OAR             *( volatile Uint32* )( I2C_BASE + 0x00 )
#define I2C_ICIMR           *( volatile Uint32* )( I2C_BASE + 0x04 )
#define I2C_ICSTR           *( volatile Uint32* )( I2C_BASE + 0x08 )
#define I2C_ICCLKL          *( volatile Uint32* )( I2C_BASE + 0x0C )
#define I2C_ICCLKH          *( volatile Uint32* )( I2C_BASE + 0x10 )
#define I2C_ICCNT           *( volatile Uint32* )( I2C_BASE + 0x14 )
#define I2C_ICDDR           *( volatile Uint32* )( I2C_BASE + 0x18 )
#define I2C_ICSAR           *( volatile Uint32* )( I2C_BASE + 0x1C )
#define I2C_ICDXR           *( volatile Uint32* )( I2C_BASE + 0x20 )
#define I2C_ICCMDR          *( volatile Uint32* )( I2C_BASE + 0x24 )
#define I2C_ICICSR          *( volatile Uint32* )( I2C_BASE + 0x28 )
#define I2C_ICCMDQ          *( volatile Uint32* )( I2C_BASE + 0x2C )
#define I2C_ICPSC           *( volatile Uint32* )( I2C_BASE + 0x30 )
#define I2C_ICPID1          *( volatile Uint32* )( I2C_BASE + 0x34 )
#define I2C_ICPID2          *( volatile Uint32* )( I2C_BASE + 0x38 )

#define ICOAR_MASK_7        0x007F
#define ICOAR_MASK_10        0x03FF
#define ICSAR_MASK_7         0x007F
#define ICSAR_MASK_10        0x03FF
#define ICSTR_SADDR         0x0050
#define ICSTR_SDIR          0x0400
#define ICSTR_NACKINT       0x0200
#define ICSTR_BB            0x1000
#define ICSTR_RSFULL        0x0800
#define ICSTR_XSMT          0x0400
#define ICSTR_AAS           0x0200
#define ICSTR_ADD           0x0100
#define ICSTR_SCD           0x0020
#define ICSTR_ICXRDY        0x0010
#define ICSTR_ICRDRY        0x0008
#define ICSTR_ARDY          0x0004
#define ICSTR Nack          0x0002
#define ICSTR_AL            0x0001

#define ICMCR Nackx         0x0800
#define ICMCR_FREE          0x0400
#define ICMCR_STT           0x0200
#define ICMCR_IDLEEN        0x0100
#define ICMCR STF           0x0800
#define ICMCR_RST           0x0040
#define ICMCR_TRX           0x0200
#define ICMCR_XA            0x0100
#define ICMCR_RM            0x0080
#define ICMCR_DLB           0x0040
#define ICMCR IRS           0x0020
#define ICMCR STB           0x0010
#define ICMCR PDP           0x0008
#define ICMCR BC_MASK       0x0007
INTC controller

Controls the Interrupts

# define INTC_BASE 0x01C48000
# define INTC_FIQ0 *( volatile Uint32* )( INTC_BASE + 0x00 )
# define INTC_FIQ1 *( volatile Uint32* )( INTC_BASE + 0x04 )
# define INTC_IRQ0 *( volatile Uint32* )( INTC_BASE + 0x08 )
# define INTC_IRQ1 *( volatile Uint32* )( INTC_BASE + 0x0C )
# define INTC_FIQENTRY *( volatile Uint32* )( INTC_BASE + 0x10 )
# define INTC_IRQENTRY *( volatile Uint32* )( INTC_BASE + 0x14 )
# define INTC_EINT0 *( volatile Uint32* )( INTC_BASE + 0x18 )
# define INTC_EINT1 *( volatile Uint32* )( INTC_BASE + 0x1C )
# define INTC_INTCTL *( volatile Uint32* )( INTC_BASE + 0x20 )
# define INTC_EABASE *( volatile Uint32* )( INTC_BASE + 0x24 )
# define INTC_INTPR10 *( volatile Uint32* )( INTC_BASE + 0x30 )
# define INTC_INTPR11 *( volatile Uint32* )( INTC_BASE + 0x34 )
# define INTC_INTPR12 *( volatile Uint32* )( INTC_BASE + 0x38 )
# define INTC_INTPR13 *( volatile Uint32* )( INTC_BASE + 0x3C )
# define INTC_INTPR14 *( volatile Uint32* )( INTC_BASE + 0x40 )
# define INTC_INTPR15 *( volatile Uint32* )( INTC_BASE + 0x44 )
# define INTC_INTPR16 *( volatile Uint32* )( INTC_BASE + 0x48 )
# define INTC_INTPR17 *( volatile Uint32* )( INTC_BASE + 0x4C )

# define MCBSP0_BASE 0x01E02000
# define MCBSP0_DRR_32BIT *( volatile Uint32* )( MCBSP0_BASE + 0x00 )
# define MCBSP0_DRR_16BIT *( volatile Uint16* )( MCBSP0_BASE + 0x00 )
# define MCBSP0_DRR_8BIT *( volatile Uint8* )( MCBSP0_BASE + 0x00 )
# define MCBSP0_DXR_32BIT *( volatile Uint32* )( MCBSP0_BASE + 0x04 )
# define MCBSP0_DXR_16BIT *( volatile Uint16* )( MCBSP0_BASE + 0x04 )
# define MCBSP0_DXR_8BIT *( volatile Uint8* )( MCBSP0_BASE + 0x04 )
# define MCBSP0_SPCR *( volatile Uint32* )( MCBSP0_BASE + 0x08 )
# define MCBSP0_RCR *( volatile Uint32* )( MCBSP0_BASE + 0x0C )
# define MCBSP0_XCR *( volatile Uint32* )( MCBSP0_BASE + 0x10 )
# define MCBSP0_SRGR *( volatile Uint32* )( MCBSP0_BASE + 0x14 )
# define MCBSP0_MCR *( volatile Uint32* )( MCBSP0_BASE + 0x18 )
# define MCBSP0_PCR *( volatile Uint32* )( MCBSP0_BASE + 0x24 )

# define MCBSP1_BASE 0x01E04000
# define MCBSP1_DRR_32BIT *( volatile Uint32* )( MCBSP1_BASE + 0x00 )
# define MCBSP1_DRR_16BIT *( volatile Uint16* )( MCBSP1_BASE + 0x00 )
# define MCBSP1_DRR_8BIT *( volatile Uint8* )( MCBSP1_BASE + 0x00 )
# define MCBSP1_DXR_32BIT *( volatile Uint32* )( MCBSP1_BASE + 0x04 )
# define MCBSP1_DXR_16BIT *( volatile Uint16* )( MCBSP1_BASE + 0x04 )
# define MCBSP1_DXR_8BIT *( volatile Uint8* )( MCBSP1_BASE + 0x04 )
# define MCBSP1_SPCR *( volatile Uint32* )( MCBSP1_BASE + 0x08 )
# define MCBSP1_RCR *( volatile Uint32* )( MCBSP1_BASE + 0x0C )
# define MCBSP1_XCR *( volatile Uint32* )( MCBSP1_BASE + 0x10 )
# define MCBSP1_SRGR *( volatile Uint32* )( MCBSP1_BASE + 0x14 )
# define MCBSP1_MCR *( volatile Uint32* )( MCBSP1_BASE + 0x18 )
# define MCBSP1_PCR *( volatile Uint32* )( MCBSP1_BASE + 0x24 )

# define MCBSP_SPCR_FREE 0x02000000
# define MCBSP_SPCR_SOFT 0x01000000
# define MCBSP_SPCR_FRST 0x00800000
# define MCBSP_SPCR_GRST 0x00400000
# define MCBSP_SPCR_XSYNCERR 0x00800000
# define MCBSP_SPCR_XEMPTY 0x00400000
# define MCBSP_SPCR_XRDY 0x00200000
# define MCBSP_SPCR_XRST 0x00100000
# define MCBSP_SPCR_DXENA 0x00080000
# define MCBSP_SPCR_ABIS 0x00040000
# define MCBSP_SPCR_RSYNCERR 0x00020000
# define MCBSP_SPCR_RFULL 0x00000000
```c
#define MCBSP_SPCR_RRDY 0x00000002
#define MCBSP_SPCR_RRST 0x00000001

/* ------------------------------------------------------------------------ *
*                                                                          *
*  MMC Controller                                                          *
*                                                                          *
* ------------------------------------------------------------------------ */
#define MMC_BASE                0x01E11000
#define MMC_MMCCTL              *( volatile Uint32* )( MMC_BASE + 0x00 )
#define MMC_MMCCLK              *( volatile Uint32* )( MMC_BASE + 0x04 )
#define MMC_MMCST0              *( volatile Uint32* )( MMC_BASE + 0x08 )
#define MMC_MMCST1              *( volatile Uint32* )( MMC_BASE + 0x0C )
#define MMC_MMCIN               *( volatile Uint32* )( MMC_BASE + 0x10 )
#define MMC_MMCITOR             *( volatile Uint32* )( MMC_BASE + 0x14 )
#define MMC_MMCXODD             *( volatile Uint32* )( MMC_BASE + 0x18 )
#define MMC_MMCXBLEN            *( volatile Uint32* )( MMC_BASE + 0x1C )
#define MMC_MMCXNLK             *( volatile Uint32* )( MMC_BASE + 0x20 )
#define MMC_MMCXDRR             *( volatile Uint32* )( MMC_BASE + 0x24 )
#define MMC_MMCXDXR             *( volatile Uint32* )( MMC_BASE + 0x28 )
#define MMC_MMCXCMD             *( volatile Uint32* )( MMC_BASE + 0x2C )
#define MMC_MMCXARGHL           *( volatile Uint32* )( MMC_BASE + 0x30 )
#define MMC_MMCXRSPO1           *( volatile Uint32* )( MMC_BASE + 0x34 )
#define MMC_MMCXRSPP3           *( volatile Uint32* )( MMC_BASE + 0x38 )
#define MMC_MMCXRSPP4           *( volatile Uint32* )( MMC_BASE + 0x3C )
#define MMC_MMCXRSPP45          *( volatile Uint32* )( MMC_BASE + 0x40 )
#define MMC_MMCXRSPP67          *( volatile Uint32* )( MMC_BASE + 0x44 )
#define MMC_MMCXCDX             *( volatile Uint32* )( MMC_BASE + 0x48 )
#define MMC_MMCXCIHR            *( volatile Uint32* )( MMC_BASE + 0x4C )
#define MMC_MMCXSDIOCTL         *( volatile Uint32* )( MMC_BASE + 0x50 )
#define MMC_MMCXSDIOST0         *( volatile Uint32* )( MMC_BASE + 0x54 )
#define MMC_MMCXSDIOIEN         *( volatile Uint32* )( MMC_BASE + 0x58 )
#define MMC_MMCFIFOCTL          *( volatile Uint32* )( MMC_BASE + 0x5C )

/* ------------------------------------------------------------------------ *
*                                                                          *
*  MemStick controller                                                     *
*                                                                          *
* ------------------------------------------------------------------------ */
#define MS_BASE                 0x01E12000
#define MS_MSCMD                *( volatile Uint32* )( MS_BASE + 0x00 )
#define MS_MSDATA               *( volatile Uint32* )( MS_BASE + 0x04 )
#define MS_MSSTAT               *( volatile Uint32* )( MS_BASE + 0x08 )
#define MS_MSSYST               *( volatile Uint32* )( MS_BASE + 0x0C )
#define MS_MSMC                 *( volatile Uint32* )( MS_BASE + 0x20 )

/* ------------------------------------------------------------------------ *
*                                                                          *
*  PLL1 Controller                                                         *
*      Generates DSP, ARM clocks                                           *
*                                                                          *
* ------------------------------------------------------------------------ */
#define PLL1_BASE               0x01C40800
#define PLL1_PID                *( volatile Uint32* )( PLL1_BASE + 0x000 )
#define PLL1_RSTYPE             *( volatile Uint32* )( PLL1_BASE + 0x004 )
#define PLL1_PLLCTL             *( volatile Uint32* )( PLL1_BASE + 0x010 )
#define PLL1_PLLM               *( volatile Uint32* )( PLL1_BASE + 0x110 )
#define PLL1_PLLDIV1            *( volatile Uint32* )( PLL1_BASE + 0x118 )
#define PLL1_PLLDIV2            *( volatile Uint32* )( PLL1_BASE + 0x11C )
#define PLL1_PLLDIV3            *( volatile Uint32* )( PLL1_BASE + 0x120 )
#define PLL1_POSTDIV            *( volatile Uint32* )( PLL1_BASE + 0x128 )
#define PLL1_BPDIV              *( volatile Uint32* )( PLL1_BASE + 0x12C )
#define PLL1_PLLCMD             *( volatile Uint32* )( PLL1_BASE + 0x138 )
#define PLL1_PLLSTAT            *( volatile Uint32* )( PLL1_BASE + 0x13C )
#define PLL1_PLLCMD             *( volatile Uint32* )( PLL1_BASE + 0x148 )
#define PLL1_PLLSTAT            *( volatile Uint32* )( PLL1_BASE + 0x14C )
#define PLL1_SYSTEM              *( volatile Uint32* )( PLL1_BASE + 0x150 )
#define PLL1_PLLDIV4            *( volatile Uint32* )( PLL1_BASE + 0x160 )
#define PLL1_PLLDIV5            *( volatile Uint32* )( PLL1_BASE + 0x164 )
```

/* ------------------------------------------------------------------------ *
*                                                                          *
*  PLL2 Controller                                                         *
*      Generates DDR2, VPBE clocks                                          *
*                                                                          *
* ------------------------------------------------------------------------ */
#define PLL2_BASE               0x01C40C00
#define PLL2_PID                *( volatile Uint32* )( PLL2_BASE + 0x000 )
#define PLL2_RSTYPE             *( volatile Uint32* )( PLL2_BASE + 0x0E4 )
#define PLL2_PLLCTL             *( volatile Uint32* )( PLL2_BASE + 0x100 )
#define PLL2_PLLM               *( volatile Uint32* )( PLL2_BASE + 0x110 )
#define PLL2_PLLDIV1            *( volatile Uint32* )( PLL2_BASE + 0x118 )
#define PLL2_PLLDIV2            *( volatile Uint32* )( PLL2_BASE + 0x11C )
#define PLL2_PLLDIV3            *( volatile Uint32* )( PLL2_BASE + 0x120 )
#define PLL2_POSTDIV            *( volatile Uint32* )( PLL2_BASE + 0x128 )
#define PLL2_BPDIV              *( volatile Uint32* )( PLL2_BASE + 0x12C )
#define PLL2_PLLCMD             *( volatile Uint32* )( PLL2_BASE + 0x138 )
#define PLL2_PLLSTAT            *( volatile Uint32* )( PLL2_BASE + 0x13C )
#define PLL2_CKEN               *( volatile Uint32* )( PLL2_BASE + 0x148 )
#define PLL2_CKSTAT             *( volatile Uint32* )( PLL2_BASE + 0x14C )
#define PLL2_SYSTAT             *( volatile Uint32* )( PLL2_BASE + 0x150 )
#define PLL2_PLLDIV4            *( volatile Uint32* )( PLL2_BASE + 0x160 )
#define PLL2_PLLDIV5            *( volatile Uint32* )( PLL2_BASE + 0x164 )

/* ------------------------------------------------------------------------ *
*                                                                          *
*  PSC ( Power and Sleep Controller )                                      *
*                                                                          *
* ------------------------------------------------------------------------ */
#define PSC_BASE                0x01C41000
#define PSC_INTEVAL             *( volatile Uint32* )( PSC_BASE + 0x018 )
#define PSC_MERRPR0             *( volatile Uint32* )( PSC_BASE + 0x040 )
#define PSC_MERRPR1             *( volatile Uint32* )( PSC_BASE + 0x044 )
#define PSC_MERRCR0             *( volatile Uint32* )( PSC_BASE + 0x050 )
#define PSC_MERRCR1             *( volatile Uint32* )( PSC_BASE + 0x054 )
#define PSC_PERRPR              *( volatile Uint32* )( PSC_BASE + 0x060 )
#define PSC_PERRCR              *( volatile Uint32* )( PSC_BASE + 0x068 )
#define PSC_EPCPR               *( volatile Uint32* )( PSC_BASE + 0x070 )
#define PSC_EPCcR               *( volatile Uint32* )( PSC_BASE + 0x078 )
#define PSC_PTCMD               *( volatile Uint32* )( PSC_BASE + 0x120 )
#define PSC_PTCSTAT             *( volatile Uint32* )( PSC_BASE + 0x128 )
#define PSC_PDSTATT0            *( volatile Uint32* )( PSC_BASE + 0x200 )
#define PSC_PDSTAT1             *( volatile Uint32* )( PSC_BASE + 0x204 )
#define PSC_PDCCTL0             *( volatile Uint32* )( PSC_BASE + 0x300 )
#define PSC_PDCCTL1             *( volatile Uint32* )( PSC_BASE + 0x304 )
#define PSC_MDSSTAT_BASE        { PSC_BASE + 0xA00 }
#define PSC_MDCCTL_BASE         { PSC_BASE + 0xA00 }
#define PSC_MDSSTAT_DSP          *( volatile Uint32* )( PSC_MDSSTAT_BASE + ( 4 * 39 ) )
#define PSC_MDCCTL_DSP          *( volatile Uint32* )( PSC_MDCCTL_BASE + ( 4 * 39 ) )
#define PSC_MDSSTAT_IMCOP        *( volatile Uint32* )( PSC_MDSSTAT_BASE + ( 4 * 40 ) )
#define PSC_MDCCTL_IMCOP        *( volatile Uint32* )( PSC_MDCCTL_BASE + ( 4 * 40 ) )

/* Power Domains */
#define ALWAYSON_POWER_DOMAIN   0
#define DSP_POWER_DOMAIN        1

/* Module Domains */
#define LFSC_VPSSMSTR 0 // VPSS Master
#define LFSC_VPSSSLV 1 // VPSS Slave
#define LFSC_TPC 2 // TPC
#define LFSC_TPTC0 3 // TPTC0
#define LFSC_TPTC1 4 // TPTC1
#define LFSC_EMAC 5 // EMAC
#define LFSC_EMAC_WRAPPER 6 // EMAC.WRAPPER
#define LFSC_USB 9 // USB
#define LFSC_ATA 10 // ATA
#define LFSC_VLYNQ 11 // VLYNQ
#define LFSC_DDR_EMIF 13 // DDR.EMIF
#define LFSC_AEMIF 14 // AEMIF
#define LPSC_MMC_SD 15  // MMC_SD
#define LPSC_ASP 17  // McBSP
#define LPSC_I2C 18  // I2C
#define LPSC_UART0 19  // UART0
#define LPSC_UART1 20  // UART1
#define LPSC_UART2 21  // UART2
#define LPSC_SPI 22  // SPI
#define LPSC_PWM0 23  // PWM0
#define LPSC_PWM1 24  // PWM1
#define LPSC_PWM2 25  // PWM2
#define LPSC_GPIO 26  // GPIO
#define LPSC_TIMER0 27  // TIMER0
#define LPSC_TIMER1 28  // TIMER1
#define LPSC_TIMER2 29  // TIMER2
#define LPSC_SYSTEM_SUBSYS 30  // SYSTEM SUBSYSTEM
#define LPSC_ARM 31  // ARM
#define LPSC_DSP 39  // GEM
#define LPSC_VICP 40  // IMCOP

/* ------------------------------------------------------------------------ *
 *                                                                          *
 *  PWM Controller                                                          *
 *                                                                          *
 * ------------------------------------------------------------------------ */
#define PWM0_BASE 0x01C22000  
#define PWM0_PID *( volatile Uint32* )( PWM0_BASE + 0x00 )
#define PWM0_PCR *( volatile Uint32* )( PWM0_BASE + 0x04 )
#define PWM0_CFG *( volatile Uint32* )( PWM0_BASE + 0x08 )
#define PWM0_START *( volatile Uint32* )( PWM0_BASE + 0x0C )
#define PWM0_RPT *( volatile Uint32* )( PWM0_BASE + 0x10 )
#define PWM0_PER *( volatile Uint32* )( PWM0_BASE + 0x14 )
#define PWM0_PH1D *( volatile Uint32* )( PWM0_BASE + 0x18 )

#define PWM1_BASE 0x01C22400  
#define PWM1_PID *( volatile Uint32* )( PWM1_BASE + 0x00 )
#define PWM1_PCR *( volatile Uint32* )( PWM1_BASE + 0x04 )
#define PWM1_CFG *( volatile Uint32* )( PWM1_BASE + 0x08 )
#define PWM1_START *( volatile Uint32* )( PWM1_BASE + 0x0C )
#define PWM1_RPT *( volatile Uint32* )( PWM1_BASE + 0x10 )
#define PWM1_PER *( volatile Uint32* )( PWM1_BASE + 0x14 )
#define PWM1_PH1D *( volatile Uint32* )( PWM1_BASE + 0x18 )

#define PWM2_BASE 0x01C22800  
#define PWM2_PID *( volatile Uint32* )( PWM2_BASE + 0x00 )
#define PWM2_PCR *( volatile Uint32* )( PWM2_BASE + 0x04 )
#define PWM2_CFG *( volatile Uint32* )( PWM2_BASE + 0x08 )
#define PWM2_START *( volatile Uint32* )( PWM2_BASE + 0x0C )
#define PWM2_RPT *( volatile Uint32* )( PWM2_BASE + 0x10 )
#define PWM2_PER *( volatile Uint32* )( PWM2_BASE + 0x14 )
#define PWM2_PH1D *( volatile Uint32* )( PWM2_BASE + 0x18 )

#define PWM_PID ( 0x00 )
#define PWM_PCR ( 0x04 )
#define PWM_CFG ( 0x08 )
#define PWM_START ( 0x0C )
#define PWM_RPT ( 0x10 )
#define PWM_PER ( 0x14 )
#define PWM_PH1D ( 0x18 )

/* ------------------------------------------------------------------------ *
 *                                                                          *
 *  Timer Controller                                                        *
 *                                                                          *
 * ------------------------------------------------------------------------ */
#define TIMER0_BASE 0x01C21400  
#define TIMER0_EMMGT *( volatile Uint32* )( TIMER0_BASE + 0x04 )
#define TIMER0_TIM12 *( volatile Uint32* )( TIMER0_BASE + 0x10 )
#define TIMER0_TIM34 *( volatile Uint32* )( TIMER0_BASE + 0x14 )
#define TIMER0_PRD12 *( volatile Uint32* )( TIMER0_BASE + 0x18 )
#define TIMER0_PRD34 *( volatile Uint32* )( TIMER0_BASE + 0x1C )
#define TIMER0_TRC *( volatile Uint32* )( TIMER0_BASE + 0x20 )
#define TIMER0_TGCR        *( volatile Uint32* )( TIMER0_BASE + 0x24 )
#define TIMER1_BASE        0x01C21800
#define TIMER1_EMUMGT      *( volatile Uint32* )( TIMER1_BASE + 0x04 )
#define TIMER1_TIM12       *( volatile Uint32* )( TIMER1_BASE + 0x10 )
#define TIMER1_PRD12       *( volatile Uint32* )( TIMER1_BASE + 0x14 )
#define TIMER1_PRD34       *( volatile Uint32* )( TIMER1_BASE + 0x1C )
#define TIMER1_TGCR        *( volatile Uint32* )( TIMER1_BASE + 0x24 )
#define TIMER1_WDTCR       *( volatile Uint32* )( TIMER1_BASE + 0x28 )

#define TIMER2_BASE        0x01C21C00
#define TIMER2_EMUMGT      *( volatile Uint32* )( TIMER2_BASE + 0x04 )
#define TIMER2_TIM12       *( volatile Uint32* )( TIMER2_BASE + 0x10 )
#define TIMER2_PRD12       *( volatile Uint32* )( TIMER2_BASE + 0x14 )
#define TIMER2_PRD34       *( volatile Uint32* )( TIMER2_BASE + 0x1C )
#define TIMER2_TRC         *( volatile Uint32* )( TIMER2_BASE + 0x20 )
#define TIMER2_TGCR        *( volatile Uint32* )( TIMER2_BASE + 0x24 )

#define UART0_BASE          0x01C20000
#define UART0_RBR           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_THR           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_IER           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_IIR           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_FCR           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_LCR           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_MCR           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_LSR           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_DLL           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_DLH           *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_PID1          *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_PID2          *( volatile Uint32* )( UART0_BASE + 0x00 )
#define UART0_PWREMU_MGMT    *( volatile Uint32* )( UART0_BASE + 0x00 )

#define UART1_BASE          0x01C20400
#define UART1_RBR           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_THR           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_IER           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_IIR           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_FCR           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_LCR           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_MCR           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_LSR           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_DLL           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_DLH           *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_PID1          *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_PID2          *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART1_PWREMU_MGMT    *( volatile Uint32* )( UART1_BASE + 0x00 )
#define UART_LSR   ( 0x14 )
#define UART_DLL   ( 0x20 )
#define UART_DLH   ( 0x24 )
#define UART_PID1  ( 0x28 )
#define UART_PID2  ( 0x2C )
#define UART_PWREMU_MGMT ( 0x30 )

/* ------------------------------------------------------------------------ *
 *                                                                          *
 *   VPSS Control                                                           *
 *                                                                          *
 * ------------------------------------------------------------------------ */
#if define VPSS_CLK_CTRL */ volatile Uint32* )0x01C40044

/* ------------------------------------------------------------------------ *
 *                                                                          *
 *   VPBE Control                                                           *
 *                                                                          *
 * ------------------------------------------------------------------------ */
#define VPBE_BASE 0x01C70400
#define VPBE_PID */ volatile Uint32* ) ( VPBE_BASE + 0x00 )
#define VPBE_PCR */ volatile Uint32* ) ( VPBE_BASE + 0x04 )

/* ------------------------------------------------------------------------ *
 *                                                                          *
 *   VPBE On-Screen Display                                                 *
 *                                                                          *
 * ------------------------------------------------------------------------ */
#define VPBE_OSD_BASE 0x01C70200
#define VPBE_OSD_MODE */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x00 )
#define VPBE_OSD_VIDWIN1M0 */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x04 )
#define VPBE_OSD_OSDWIN1M0 */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x08 )
#define VPBE_OSD_OSDWIN1M1 */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x0C )
#define VPBE_OSD_RECTCUR */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x10 )
#define VPBE_OSD_VIDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x18 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x20 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x24 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x28 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x30 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x34 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x38 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x40 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x44 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x48 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x4C )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x50 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x54 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x58 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x5C )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x60 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x64 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x68 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x70 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x74 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x78 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x7C )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x80 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x84 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x88 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x90 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x94 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0x98 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0xA0 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0xA4 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0xA8 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0xAC )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0xB0 )
#define VPBE_OSD_OSDWIN1OFST */ volatile Uint32* ) ( VPBE_OSD_BASE + 0xB4 )

130
#define VPFE_VENC_RAMPRT  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0C0 ) )
#define VPFE_VENC_DACSTST  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0C4 ) )
#define VPFE_VENC_YCOLVL  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0C8 ) )
#define VPFE_VENC_SCPRG  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0CC ) )
#define VPFE_VENC_CBVS  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0D0 ) )
#define VPFE_VENC_CMPNT  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0E0 ) )
#define VPFE_VENC_ETMG0  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0E4 ) )
#define VPFE_VENC_ETMG1  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0E8 ) )
#define VPFE_VENC_ETMG2  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0EC ) )
#define VPFE_VENC_ETMG3  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0F0 ) )
#define VPFE_VENC_DACSEL  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x0F4 ) )
#define VPFE_VENC_ARGBX0  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x100 ) )
#define VPFE_VENC_ARGBX1  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x104 ) )
#define VPFE_VENC_ARGBX2  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x108 ) )
#define VPFE_VENC_ARGBX3  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x10C ) )
#define VPFE_VENC_ARGBX4  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x110 ) )
#define VPFE_VENC_DRGBX0  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x114 ) )
#define VPFE_VENC_DRGBX1  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x118 ) )
#define VPFE_VENC_DRGBX2  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x11C ) )
#define VPFE_VENC_DRGBX3  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x120 ) )
#define VPFE_VENC_DRGBX4  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x124 ) )
#define VPFE_VENC_VSTARTA  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x128 ) )
#define VPFE_VENC_OSDCLK0  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x12C ) )
#define VPFE_VENC_OSDCLK1  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x130 ) )
#define VPFE_VENC_HVLDCL0  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x134 ) )
#define VPFE_VENC_HVLDCL1  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x138 ) )
#define VPFE_VENC_OSDHADV  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x13C ) )
#define VPFE_VENC_SCTEST0  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x140 ) )
#define VPFE_VENC_SCTEST1  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x144 ) )
#define VPFE_VENC_SCTEST2  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x148 ) )
#define VPFE_VENC_VTEST0  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x14C ) )
#define VPFE_VENC_VTEST1  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x150 ) )
#define VPFE_VENC_VTEST2  (*( volatile Uint32* ) ( VPFE_VENC_BASE + 0x154 ) )
#define VPFE_CCDC_FMT_ADDR5 *( volatile Uint32* )( VPFE_CCDC_BASE + 0x78 )
#define VPFE_CCDC_FMT_ADDR6 *( volatile Uint32* )( VPFE_CCDC_BASE + 0x7c )
#define VPFE_CCDC_FMT_ADDR7 *( volatile Uint32* )( VPFE_CCDC_BASE + 0x80 )
#define VPFE_CCDC_PRGEVEN_0 *( volatile Uint32* )( VPFE_CCDC_BASE + 0x84 )
#define VPFE_CCDC_PRGEVEN_1 *( volatile Uint32* )( VPFE_CCDC_BASE + 0x88 )
#define VPFE_CCDC_PRGODD_0 *( volatile Uint32* )( VPFE_CCDC_BASE + 0x8c )
#define VPFE_CCDC_PRGODD_1 *( volatile Uint32* )( VPFE_CCDC_BASE + 0x90 )
#define VPFE_CCDC_VP_OUT *( volatile Uint32* )( VPFE_CCDC_BASE + 0x94 )

/* ------------------------------------------------------------------------ */
/*                                                                          */
/*   VPFE Hardware 3A ( Auto-Focus / WB / Exposure )                          */
/*                                                                          */
/* ------------------------------------------------------------------------ */
#define VPFE_H3A_BASE 0x01C70080
#define VPFE_H3A_PID *( volatile Uint32* )( VPFE_H3A_BASE + 0x00 )
#define VPFE_H3A_PCR *( volatile Uint32* )( VPFE_H3A_BASE + 0x04 )
#define VPFE_H3A_AFPAX1 *( volatile Uint32* )( VPFE_H3A_BASE + 0x08 )
#define VPFE_H3A_AFPAX2 *( volatile Uint32* )( VPFE_H3A_BASE + 0x0c )
#define VPFE_H3A_AFPAXSTART *( volatile Uint32* )( VPFE_H3A_BASE + 0x10 )
#define VPFE_H3A_AFIRSH *( volatile Uint32* )( VPFE_H3A_BASE + 0x14 )
#define VPFE_H3A_ABFUST *( volatile Uint32* )( VPFE_H3A_BASE + 0x18 )
#define VPFE_H3A_AFCOEFO10 *( volatile Uint32* )( VPFE_H3A_BASE + 0x1c )
#define VPFE_H3A_AFCOEFO32 *( volatile Uint32* )( VPFE_H3A_BASE + 0x20 )
#define VPFE_H3A_AFCOEFO54 *( volatile Uint32* )( VPFE_H3A_BASE + 0x24 )
#define VPFE_H3A_AFCOEFO76 *( volatile Uint32* )( VPFE_H3A_BASE + 0x28 )
#define VPFE_H3A_AFCOEFO98 *( volatile Uint32* )( VPFE_H3A_BASE + 0x2c )
#define VPFE_H3A_AFCOEF010 *( volatile Uint32* )( VPFE_H3A_BASE + 0x30 )
#define VPFE_H3A_AFCOEFO110 *( volatile Uint32* )( VPFE_H3A_BASE + 0x34 )
#define VPFE_H3A_AFCOEFO132 *( volatile Uint32* )( VPFE_H3A_BASE + 0x38 )
#define VPFE_H3A_AFCOEFO154 *( volatile Uint32* )( VPFE_H3A_BASE + 0x3c )
#define VPFE_H3A_AFCOEFO176 *( volatile Uint32* )( VPFE_H3A_BASE + 0x40 )
#define VPFE_H3A_AFCOEFO198 *( volatile Uint32* )( VPFE_H3A_BASE + 0x44 )
#define VPFE_H3A_AFCOEFO1010 *( volatile Uint32* )( VPFE_H3A_BASE + 0x48 )
#define VPFE_H3A_AEWWIN1 *( volatile Uint32* )( VPFE_H3A_BASE + 0x4c )
#define VPFE_H3A_AEWINSTART *( volatile Uint32* )( VPFE_H3A_BASE + 0x50 )
#define VPFE_H3A_AEWINBLK *( volatile Uint32* )( VPFE_H3A_BASE + 0x54 )
#define VPFE_H3A_AEWSUBWIN *( volatile Uint32* )( VPFE_H3A_BASE + 0x58 )
#define VPFE_H3A_AEWBUFST *( volatile Uint32* )( VPFE_H3A_BASE + 0x5c )

#define VPFE_H3A_BASE 0x01C70080
#define VPFE_H3A_PID *( volatile Uint32* )( VPFE_H3A_BASE + 0x00 )
#define VPFE_H3A_PCR *( volatile Uint32* )( VPFE_H3A_BASE + 0x04 )
#define VPFE_H3A_AFPAX1 *( volatile Uint32* )( VPFE_H3A_BASE + 0x08 )
#define VPFE_H3A_AFPAX2 *( volatile Uint32* )( VPFE_H3A_BASE + 0x0c )
#define VPFE_H3A_AFPAXSTART *( volatile Uint32* )( VPFE_H3A_BASE + 0x10 )
#define VPFE_H3A_AFIRSH *( volatile Uint32* )( VPFE_H3A_BASE + 0x14 )
#define VPFE_H3A_ABFUST *( volatile Uint32* )( VPFE_H3A_BASE + 0x18 )
#define VPFE_H3A_AFCOEFO10 *( volatile Uint32* )( VPFE_H3A_BASE + 0x1c )
#define VPFE_H3A_AFCOEFO32 *( volatile Uint32* )( VPFE_H3A_BASE + 0x20 )
#define VPFE_H3A_AFCOEFO54 *( volatile Uint32* )( VPFE_H3A_BASE + 0x24 )
#define VPFE_H3A_AFCOEFO76 *( volatile Uint32* )( VPFE_H3A_BASE + 0x28 )
#define VPFE_H3A_AFCOEFO98 *( volatile Uint32* )( VPFE_H3A_BASE + 0x2c )
#define VPFE_H3A_AFCOEFO1010 *( volatile Uint32* )( VPFE_H3A_BASE + 0x30 )
#define VPFE_H3A_AFCOEFO110 *( volatile Uint32* )( VPFE_H3A_BASE + 0x34 )
#define VPFE_H3A_AFCOEFO132 *( volatile Uint32* )( VPFE_H3A_BASE + 0x38 )
#define VPFE_H3A_AFCOEFO154 *( volatile Uint32* )( VPFE_H3A_BASE + 0x3c )
#define VPFE_H3A_AFCOEFO176 *( volatile Uint32* )( VPFE_H3A_BASE + 0x40 )
#define VPFE_H3A_AFCOEFO198 *( volatile Uint32* )( VPFE_H3A_BASE + 0x44 )
#define VPFE_H3A_AFCOEFO1010 *( volatile Uint32* )( VPFE_H3A_BASE + 0x48 )
#define VPFE_H3A_AEWWIN1 *( volatile Uint32* )( VPFE_H3A_BASE + 0x4c )
#define VPFE_H3A_AEWINSTART *( volatile Uint32* )( VPFE_H3A_BASE + 0x50 )
#define VPFE_H3A_AEWINBLK *( volatile Uint32* )( VPFE_H3A_BASE + 0x54 )
#define VPFE_H3A_AEWSUBWIN *( volatile Uint32* )( VPFE_H3A_BASE + 0x58 )
#define VPFE_H3A_AEWBUFST *( volatile Uint32* )( VPFE_H3A_BASE + 0x5c )

#define EVMDM355_wait _wait
#define EVMDM355_waitusec _waitusec

/* Board Initialization */
Int16 EVMDM355_init( );

/* Wait Functions */
void _wait( Uint32 delay );
void _waitusec( Uint32 usec );

#endif
```c
#ifndef I2C_
define I2C_
#include "evmdm355.h"

/* ------------------------------------------------------------------------ */
/*  Prototypes                                                            */
/* ------------------------------------------------------------------------ */
Int16 EVMDM355_I2C_init( );
Int16 EVMDM355_I2C_close( );
Int16 EVMDM355_I2C_write( Uint16 i2c_addr, Uint8* data, Uint16 len );
Int16 EVMDM355_I2C_read ( Uint16 i2c_addr, Uint8* data, Uint16 len );
#endif

/* _I2C_init( )                                                            */
Int16 EVMDM355_I2C_init( ) {
  I2C_ICMDR = 0;                // Reset I2C
  I2C_ICPSC = 26;               // Config prescaler for 27MHz
  I2C_ICCLKL = 20;              // Config clk LOW for 20kHz
  I2C_ICCLKH = 20;              // Config clk HIGH for 20kHz
  I2C_ICMDR |= ICMDR_IRS;        // Release I2C from reset
  return 0;
}

/* _I2C_close( )                                                           */
Int16 EVMDM355_I2C_close( ) {
  I2C_ICMDR = 0;                      // Reset I2C
  return 0;
}

/* _I2C_reset( )                                                           */
Int16 EVMDM355_I2C_reset( ) {
  EVMDM355_I2C_close( );
}
```

Copyright 2007 by Spectrum Digital Incorporated. All rights reserved. Property of Spectrum Digital Incorporated.
EVMD355_I2C_init();
return 0;
}

/* ------------------------------------------------------------------------ *

*I2C_write( i2c_addr, data, len ) *
*I2C write in Master mode *
*i2c_addr <- I2C slave address *
data <- I2C data ptr *
len <- # of bytes to write *
*------------------------------------------------------------------------ */

Int16 EVMD355_I2C_write( Uint16 i2c_addr, Uint8* data, Uint16 len )
{
  Int32 timeout, i;
  I2C_ICCNT = len; // Set length
  I2C_ICSAR = i2c_addr; // Set I2C slave address
  I2C_ICMDR = ICMDR_STT // Set for Master Write
      | ICMDR_TRX
      | ICMDR_MST
      | ICMDR_IRS
      | ICMDR_FREE;
  _wait( 10 ); // Short delay
  for ( i = 0 ; i < len ; i++ )
  {
    I2C_ICDXR = data[i]; // Write
    timeout = i2c_timeout;
    do
      {
        if ( timeout-- < 0 )
          {
            EVMD355_I2C_reset( );
            return -1;
          }
      } while ( ( I2C_ICSTR & ICSTR_ICXRDY ) == 0 );// Wait for Tx Ready
  
  I2C_ICMDR |= ICMDR_STP; // Generate STOP
  return 0;
}

/* ------------------------------------------------------------------------ *

*I2C_read( i2c_addr, data, len ) *
*I2C read in Master mode *
*i2c_addr <- I2C slave address *
data <- I2C data ptr *
len <- # of bytes to write *
*Returns: 0: PASS *
-1: FAIL Timeout *
*------------------------------------------------------------------------ */

Int16 EVMD355_I2C_read( Uint16 i2c_addr, Uint8* data, Uint16 len )
{
  Int32 timeout, i;
  I2C_ICCNT = len; // Set length
  I2C_ICSAR = i2c_addr; // Set I2C slave address
  ...
I2C_ICMDR = ICMDR_STT               // Set for Master Read
    | ICMDR_MST
    | ICMDR_IRS
    | ICMDR_FREE;

_wait( 10 );                        // Short delay

for ( i = 0 ; i < len ; i++ )
{
    timeout = i2c_timeout;
    /* Wait for Rx Ready */
    do
    {
        if ( timeout-- < 0 )
        {
            EVMDM355_I2C_reset( );
            return -1;
        }
    } while ( ( I2C_ICSTR & ICSTR_ICRRDY ) == 0 );// Wait for Rx Ready

    data[i] = I2C_ICDRR;            // Read
}

//I2C_ICMDR |= ICMDR_STP;             // Generate STOP
return 0;

/* ------------------------------------------------------------------------ *
*                                                                          *
*  _I2C_read_variable( i2c_addr, data, len )                               *
*                                                                          *
*      I2C read in Master mode                                             *
*      i2c_addr <- I2C slave address                                       *
*      data    <- I2C data ptr                                             *
*      len     <- # of bytes to write                                      *
*                                                                          *
* ------------------------------------------------------------------------ */
Int16 EVMDM355_I2C_read_variable( Uint16 i2c_addr, Uint8* data, Uint16 len )
{
    Int32 timeout, i;

    I2C_ICCNT = len;                    // Set length
    I2C_ICSAR = i2c_addr;               // Set I2C slave address
    I2C_ICMDR = ICMDR_STT               // Set for Master Read
        | ICMDR_MST
        | ICMDR_IRS
        | ICMDR_FREE;

    _wait( 10 );                        // Short delay

    for ( i = 0 ; i < len ; i++ )
    {
        if ( i == 1 )
        {
            len = data[0];         // Set length
            I2C_ICCNT = len;
        }

        timeout = i2c_timeout;
        do
        {
            if ( timeout-- < 0 )
            {
                EVMDM355_I2C_reset( );
                return -1;
            }
        } while ( ( I2C_ICSTR & ICSTR_ICRRDY ) == 0 );// Wait for Rx Ready
```c
    data[i] = I2C_ICDRR; // Read
}
I2C_ICMDR |= ICMDR_STP; // Generate STOP
return 0;
```

Rules.make (Included in MontaVista development environment, with slight modifications)

```make
# The installation directory of the DVSDK dvsdk_1_30_00_23.
DVSDK_INSTALL_DIR=${HOME}/dvsdk_1_30_00_23

# For backwards compatibility
DVEVM_INSTALL_DIR=${DVSDK_INSTALL_DIR}

# Where the Codec Engine package is installed.
CE_INSTALL_DIR=${DVSDK_INSTALL_DIR}/codec_engine_2_00

# Where the XDAIS package is installed.
XDAIS_INSTALL_DIR=${DVSDK_INSTALL_DIR}/xdais_6_00

# Where the DSP Link package is installed.
LINK_INSTALL_DIR=${DVSDK_INSTALL_DIR}/NOT_USED

# Where the CMEM (contiguous memory allocator) package is installed.
CMEM_INSTALL_DIR=${DVSDK_INSTALL_DIR}/cmem_2_00

# Where the codec servers are installed
CODEC_INSTALL_DIR=${DVSDK_INSTALL_DIR}/dm355_codecs_1_06_01

# Where the RTSC tools package is installed.
XDC_INSTALL_DIR=${DVSDK_INSTALL_DIR}/xdc_3_00_02_11

# Where Framework Components product is installed
FC_INSTALL_DIR=${DVSDK_INSTALL_DIR}/framework_components_2_00

# Where DSP/BIOS is installed
BIOS_INSTALL_DIR=${DVSDK_INSTALL_DIR}/

# The directory that points to your kernel source directory.
LINUXKERNEL_INSTALL_DIR=/opt/mv_pro_4.0.1/montavista/pro/devkit/lsp/ti-davinci

# The prefix to be added before the GNU compiler tools (optionally including
# path), i.e. "arm_v5t_le-" or "/opt/bin/arm_v5t_le-".
MVTOOL_DIR=/opt/mv_pro_4.0.1/montavista/pro/devkit/arm/v5t_le
MVTOOL_PREFIX=${MVTOOL_DIR}/bin/arm_v5t_le-

# Where to copy the resulting executables and data to (when executing 'make
# install') in a proper file structure. This EXEC_DIR should either be visible
# from the target, or you will have to copy this (whole) directory onto the
# target filesystem.
EXEC_DIR=/home/gilkyboy/workdir/filesys/opt/hello
```

Makefile (mostly original, with some parts from MontaVista development environment)

```make
ROOTDIR = ./
include $(ROOTDIR)/Rules.make

RELTARGET = release/$(TARGET)

# The prefix to be added before the GNU compiler tools (optionally including
# path), i.e. "arm_v5t_le-" or "/opt/bin/arm_v5t_le-".
MVTOOL_DIR=/opt/mv_pro_4.0.1/montavista/pro/devkit/arm/v5t_le
MVTOOL_PREFIX=${MVTOOL_DIR}/bin/arm_v5t_le-
```
all: evmdm355.o evmdm355_gpio.o evmdm355_i2c.o main.o
    @echo Linking $@ from ^..,
    $(MVTOOL_PREFIX)gcc -o GPIO_test main.o evmdm355.o evmdm355_gpio.o evmdm355_i2c.o

main.o: evmdm355_gpio.h evmdm355_gpio.c evmdm355.h evmdm355.c evmdm355_i2c.h evmdm355_i2c.c
    @echo Building $(TARGET) from $(^).
    $(MVTOOL_PREFIX)gcc -Wall -DDM355 -Dti_sdo_ce_osal_Memory_USEDEPRECATEDAPIS=1 -I$(LINUXKERNEL_INSTALL_DIR)/include -c main.c

evmdm355.o: evmdm355.h evmdm355.c
    @echo Building $(TARGET) from $(^).
    $(MVTOOL_PREFIX)gcc -Wall -DDM355 -Dti_sdo_ce_osal_Memory_USEDEPRECATEDAPIS=1 -I$(LINUXKERNEL_INSTALL_DIR)/include -c evmdm355.c

evmdm355_i2c.o: evmdm355.h evmdm355.c evmdm355_i2c.h evmdm355_i2c.c
    @echo Building $(TARGET) from $(^).
    $(MVTOOL_PREFIX)gcc -Wall -DDM355 -Dti_sdo_ce_osal_Memory_USEDEPRECATEDAPIS=1 -I$(LINUXKERNEL_INSTALL_DIR)/include -c evmdm355_i2c.c

evmdm355_gpio.o: evmdm355_gpio.h evmdm355_gpio.c evmdm355.h evmdm355.c evmdm355_i2c.h evmdm355_i2c.c
    @echo Building $(TARGET) from $(^).
    $(MVTOOL_PREFIX)gcc -Wall -DDM355 -Dti_sdo_ce_osal_Memory_USEDEPRECATEDAPIS=1 -I$(LINUXKERNEL_INSTALL_DIR)/include -c evmdm355_gpio.c

install:
    @echo Installing $(TARGET) target files to $(EXEC_DIR).. 
    $(VERBOSE) install -d $(EXEC_DIR)
    $(VERBOSE) install $(RELTARGET) $(EXEC_DIR)
    $(VERBOSE) install -m 444 $(TARGET).txt $(EXEC_DIR)

clean:
    @echo Removing generated files..
    \rm *.o /*- GPIO_test

main.c

/* Main.c is a test of the basic GPIO functions
 * Written by Charlie Reitsma
 ************************************************/

#include <stdio.h>

/* Include the header files for the dm355 interface */
#include "evmdm355.h"
#include "evmdm355_i2c.h"
#include "evmdm355_gpio.h"

int main(void)
{
    printf("I ran, I'm running");

    // this code was in the development stages and the commenting out is for debugging purposes.

    //EVMDM355_init();

    return 0;
}
/**
 * EVMDM355_GPIO_getInput
 * number   <- GPIO#
 * 0: low
 * 1: high
 */

EVMDM355_GPIO_getInput( 31 );

return 0;
}

Compiler.rules

C_FLAGS += -Wall -DDM355 -Dti_sdo_ce_osal_Memory_USEDEPRECATEDAPIs=1
CPP_FLAGS += -I$(DVSDK_INSTALL_DIR)/demos/utils/include \
    -I$(LINUXKERNEL_INSTALL_DIR)/include
LD_FLAGS += -lfreetype -lpng -ljpeg -lpthread

# Comment this out if you want to see full compiler and linker output.
VERBOSE = @

COMPILE.c = $(VERBOSE) $(MVTOOL_PREFIX)gcc $(C_FLAGS) $(CPP_FLAGS) -c
LINK.c = $(VERBOSE) $(MVTOOL_PREFIX)gcc $(LD_FLAGS)

# Target tools
XDC_TARGET = gnu.targets.MVArm9

# Platform (board) to build for
XDC_PLATFORM = ti.platforms.evmDM6446

# Package path for the XDC tools
XDC_PATH = $(CODEC_INSTALL_DIR)/packages;$(CE_INSTALL_DIR)/packages;$(LINK_INSTALL_DIR)/packages;$(XDAIS_INSTALL_DIR)/packages;$(CMEM_INSTALL_DIR)/packages;$(FC_INSTALL_DIR)/packages

# The XDC configuration tool command line
CONFIGURO = XDCPATH="$(XDC_PATH)" $(XDC_INSTALL_DIR)/xs xdc.tools.configuro
15.7. Senior Design Night Demonstration Perl Code

test.pl

#!/usr/bin/perl

use File::Basename;
#use File::Copy;
use Net::Telnet ();

SNFS_Share = "C:\NFSShare";
$Local_Sub_Directory = "opt\data";
$Remost_Sub_Directory = "opt/data";
$TI_Telnet_IP = "192.168.1.6";
$Telnet_username = "root";
$Telnet_password = "";

$Init_Command = "/opt/dvsdk/loadmodules.sh";

$SCD_Command = "cd /opt/dvsdk";

$Run_Command = "/decode -v ./data/videos/davinsideffect_ntsc.mpeg4 -s ./data/sounds/davinsideffect_g711";

use Net::Telnet ();
@output = $t = new Net::Telnet (Timeout => 10,
   Prompt => \S*[#>\s+S/mi]?
   print @output;
@output = $t->open($TI_Telnet_IP);
   print @output;
@output = $t->waitfor("/login: $/i");
   print @output;
@output = $t->print($Telnet_username);
   print @output;
@output = $t->waitfor("/S*[#>\s+S/mi]?
   print @output;
@output = $t->print($Init_Command);
   print @output;
@output = $t->waitfor("/S*[#>\s+S/mi]?
   print @output;
@output = $t->print($CD_Command);
   print @output;
@output = $t->waitfor("/S*[#>\s+S/mi]?
   print @output;
@output = $t->print($Run_Command);
   print @output;

@@output = $t->timeout(undef);
    print @output;

@@output = $t->print($Run_Command);
    print @output;

@@output = $t->waitfor('/\S*[\#:>]\a*$/mi')
    print @output;

# $telnet->open('camel.perflect.com');
# $t->login($Telnet_username, 'password');

# $telnet = new Net::Telnet ( Timeout=>10, Errmode=>'die');
# $telnet->open('camel.perflect.com');
# $telnet->waitfor('/login: $/i');
# $telnet->print('bilbo');
# $telnet->waitfor('/password: $/i');
# $telnet->print('baggins');
# $telnet->waitfor('/$ S/i');
# $telnet->print('who');

# $output = $t->waitfor('/\S*[\#:>]\a*$/mi');
# print $output;

# @output = $t->timeout(undef);
# print @output;

# $output = $t->print($Run_Command);
# print @output;

# $output = $t->waitfor('/\S*[\#:>]\a*$/mi');
# print @output;

# $output = $t->timeout(undef);
# print @output;

# $output = $t->print($Run_Command);
# print @output;

# $output = $t->waitfor('/\S*[\#:>]\a*$/mi');
# print @output;

# $telnet->open('camel.perflect.com');
# $t->login($Telnet_username, 'password');

# $telnet = new Net::Telnet ( Timeout=>10, Errmode=>'die');
# $telnet->open('camel.perflect.com');
# $telnet->waitfor('/login: $/i');
# $telnet->print('bilbo');
# $telnet->waitfor('/password: $/i');
# $telnet->print('baggins');
# $telnet->waitfor('/$ S/i');
# $telnet->print('who');

# $output = $t->waitfor('/\S*[\#:>]\a*$/mi');
# print $output;

# $telnet->open('camel.perflect.com');
# $t->login($Telnet_username, 'password');

# $telnet = new Net::Telnet ( Timeout=>10, Errmode=>'die');
# $telnet->open('camel.perflect.com');
# $telnet->waitfor('/login: $/i');
# $telnet->print('bilbo');
# $telnet->waitfor('/password: $/i');
# $telnet->print('baggins');
# $telnet->waitfor('/$ S/i');
# $telnet->print('who');

# $output = $t->waitfor('/\S*[\#:>]\a*$/mi');
# print $output;
real.pl

foreach $argnum (0 .. $#ARGV) {

    my ($newfile, $file_dir, $file_suffix) = fileparse($ARGV[$argnum], qr/\.[^.]*/);

    print "The file suffix is $file_suffix\n";

    if ($file_suffix =~ /.mpeg4/) {
        $video_file = "$newfile$file_suffix";
        print "Video file\n";
    }
    elsif ($file_suffix =~ /.g711/ ) {
        $audio_file = "$newfile$file_suffix";
        print "Audio file\n";
    }
    elsif ($file_suffix =~ /.txt/ ) {
        # $audio_file = "$newfile$file_suffix";
        print "Text file\n";
    }

    print "Filepath is $ARGV[$argnum]\n";
    #print "content-type: text/html \n\n"; #The header
    #$filetobecopied = $ARGV[$argnum];
    #$newfile = "C:\$filetobecopied";
    print "Newfile is $newfile\n";
    print "Newfile directory is $NFS_Share\$Local_Sub_Directory\$newfile$file_suffix\n";
}
system("copy $ARGV[$argnum] $NFS_Share\$Local_Sub_Directory\$newfile$file_suffix");

#copy($ARGV[$argnum], $Local_Sub_Directory/$newfile") or die "File cannot be copied.

}

if ($video_file && $audio_file) {
    print "There is an audio and a video file\n";
    $Run_Command = "/opt/dvsdk/decode/ -v $video_file -s $audio_file"
}
elseif ($video_file) {
    print "There is a video file\n";
    $Run_Command = "/opt/dvsdk/decode/ -v $video_file"
}
elseif ($audio_file) {
    print "There is an audio file\n";
    $Run_Command = "/opt/dvsdk/decode/ -s $audio_file"
}


---

**test.bat1**

:: name fredarg.bat
:: parameters:
:: -1 A simple switch (using no following arguments)
:: -2 <arg> The name2 parms (using 1 following argument)
:: -3 <arg arg> The name3 parms (using 2 following arguments)
:: you may have any number of parameters

:: init variables
set name1=no
set name2=
set name3=

goto loop
:shift3loop
shift
:shift2loop
shift
:shiftloop
shift
:loop

:: here test for expected args
if "%1" == "-1" set name1=yes
if "%1" == "-1" goto shiftloop

if "%1" == "-2" set name2=%2
if "%1" == "-2" goto shift2loop

if "%1" == "-3" set name3=%2 %3
if "%1" == "-3" goto shift3loop

if "%1" == "" goto print

echo Usage error
echo UNRECOGNIZED ARG %1'
goto done

:print
echo name1 = "%name1%"
echo name2 = "%name2%"
echo name3 = "%name3%"
:: process args here...
pause
CALL "C:\Program Files\Windows NT\Accessories\wordpad.exe"