PRINTED CIRCUIT BOARD DESIGN FOR FREQUENCY DISTURBANCE RECORDER

By

Lei Wang

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
In
Electrical Engineering

APPROVED:

_______________________________
Dr. Yilu Liu, Chairman

__________________  ____________________
Dr. Richard Conners        Dr. Virgilio Centeno

December 12, 2005
Blacksburg, Virginia

Keywords: Frequency Disturbance Recorder (FDR)
Frequency Monitoring Network (FNET)
Power system monitoring
Printed Circuit Board (PCB)
Hardware interface
MPC555
Printed Circuit Board Design for Frequency Disturbance Recorder

Lei Wang

ABSTRACT

The FDR (Frequency Disturbance Recorder) is a data acquisition device for the power system. The device is portable and can be used with any residential wall outlet for frequency data collection. Furthermore, the FDR transmits calculated frequency data to the web for access by authorized users via Ethernet connection. As a result, Virginia Tech implemented Frequency Monitoring Network (FNET) with these FDR devices. FNET is a collection of identical FDRs placed in different measurement sites to allow for data integration and comparison. Frequency is an important factor for power system control and stabilization. With funding and support provided by ABB, TVA and NSF the FDRs are placed strategically all over the United States for frequency analysis, power system protection and monitoring.

The purpose of this study is to refine the current FDR hardware design and establish a new design that will physically fit all the components on one Printed Circuit Board (PCB). At the same time, the software that is to be implemented on the new board is to be kept similar if not the same as that of the current design. The current FDR uses the Axiom CME555 development board and it is interfaced to the external devices through its communication ports. Even through the CME555 board is able to meet the demands of the basic FDR operations, there are still several problems associated with this design. This paper will address some of those hardware problems, as well as propose a new board design that is specifically aimed for operations of FDR.
ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my academic and research advisor Dr. Yilu Liu for her guidance and constant support in helping me to conduct and complete this work. I would also like to thank Dr. Richard Conners for his constant support and expertise in many different areas of the research. In addition, I want to thank Dr. Vigilio Centeno for serving on my advisory committee, as well as for his kind advice and help in the classroom and lab. Also, I want to thank Ryan (Jian) Zuo, Emily (Chunchun) Xu, Kevin (Zhian) Zhong and Bruce Billian for their generous advices and helpful comments regarding the development of Frequency Disturbance Recorder and FNET. Finally, with deepest love and appreciation, I would like to thank my father Yuwen Wang, my mother Zengjuan Li and my lovely sister Julie Wang for their consistent support and encouragements.
# TABLE OF CONTENTS

Chapter 1: Introduction ............................................................................. 1  
  1.1 Current FDR Hardware Architecture .............................................. 1  
  1.2 Problem Statement and Proposed Work ......................................... 2  
  1.3 Thesis Organization ..................................................................... 3  
Chapter 2: Axiom CME555 ................................................................. 5  
  2.1 MPC555 Processor ....................................................................... 8  
  2.2 External Memory ......................................................................... 11  
  2.3 Jumpers and Switches ................................................................. 11  
  2.4 Ports and Connectors ................................................................. 13  
Chapter 3: External Peripherals ............................................................ 14  
  3.1 Signal Conditioning Unit ............................................................. 14  
  3.2 ADC Unit ................................................................................... 14  
  3.3 Motorola M12+ Receiver ............................................................. 15  
  3.4 Serial to Ethernet Converter ....................................................... 17  
Chapter 4: FDR Software ................................................................. 18  
Chapter 5: Past and Related Efforts ..................................................... 21  
  5.1 Protel ......................................................................................... 21  
  5.2 MPC555 Support Pins ................................................................. 21  
Chapter 6: Printed Circuit Board Design – PB555 Prototype ................. 27  
  6.1 Prototype Proposal ................................................................. 27  
  6.2 PB555 board ............................................................................. 28  
  6.3 Prototype Development ............................................................. 30  
  6.4 LCD Interface ........................................................................... 32  
  6.5 LCD Test ............................................................................... 40  
  6.6 GPS Interface .......................................................................... 48  
  6.7 Signal Conditioning Unit and ADC Interface ............................... 52  
Chapter 7: Conclusions and Future Work ............................................ 54  
  7.1 Conclusions ............................................................................. 54  
  7.2 Future Work ............................................................................. 54
LIST OF TABLES

Table 1 Summary of MPC555 Support Signal Connections..........................25
Table 2 Mapping of LCD port pins to MPC555 signals...............................34
Table 3 DMC20261NY-LY-AXE timing constraints..................................35
Table 4 Absolute maximum ratings for DMC20261NY-LY-AXE power supply.....39
Table 5 HCD44780 controller register selection......................................42
Table 6 Operation code for HCD44780 instructions..................................42
Table 7 Memory initialization modifications...........................................44
Table 8 M12+ connector pin information..............................................51

LIST OF ABBREVIATIONS

PCB: Printed Circuit Board
FDR: Frequency Disturbance Recorder
FNET: Frequency Monitoring Network
LCD: Liquid Crystal Display
ADC: Analog to Digital Converter
GPS: Global Positioning System
EPLD: Electrically Programmable Logic Device
LED: Light Emitting Diode
JTAG: Joint Test Action Group
BDM: Background Debug Monitor
PWM: Pulse Width Modulation
TPU: Time Processor Unit (MPC555 subsystem)
MIOS: Modular Input Output Subsystem (MPC555 subsystem)
QSMCM: Queued Serial Multi-Channel Module (MPC555 subsystem)
QADC: Queued Analog to Digital Converter (MPC555 subsystem)
TouCAN: Two 2.0B Controller Area Network (MPC555 subsystem)
DASM: Double Action Submodules (MPC555 subsystem)
PWMSM: Pulse Width Modulation Submodules (MPC555 subsystem)
MCSM: Modulus Counter Submodules (MPC555 subsystem)
PIOSM: Parallel Port Input Output Submodules (MPC555 subsystem)
GPIO: General Purpose Input Output
EEPROM: Electrically Erasable Programmable Read Only Memory
EPROM: Electrically Programmable Read Only Memory
RTS: Request to Send (Protocol signal)
CTS: Clear to Send (Protocol signal)
EMI: ElectroMagnetic Interference
TTFF: Time To First Fix
CPU: Central Processing Unit
SRAM: Static Random Access Memory
SCI: Serial Communication Interface
SPI: Serial to Parallel Interface
CMOS: Complementary Metal Oxide Semiconductor
TTL: Transistor Transistor Logic
DGPS: Differential Global Positioning System
TCP/IP: Transmission Control Protocol/Internet Protocol
UTC: Coordinated Universal Time
PPS: Pulse Per Second
1.1 Current FDR Hardware Architecture

The current FDR hardware is composed of the Axiom CME-555 board, an Analog to Digital Converter unit (ADC), a Global Positioning System unit (GPS), a signal-conditioning unit, a Serial to Ethernet Converter and a power supply [14]. See Figure 1 [14] for a block diagram of the overall system. The main algorithm is implemented in a microprocessor-based unit, where the frequency computation, timing synchronization and communication tasks are executed. The overall operation of the FDR can be described as the following: the transformer first lowers the power grid voltage to a reasonable level, then the signal conditioning and ADC unit process the signal for sending it to the microprocessor, and finally the microprocessor communicates to the GPS for timing synchronization and outputs the frequency estimation results to the server.

[14]. Figure 2 shows the outside view of a FDR unit. There is a Serial to Ethernet converter sitting on top of the unit. The embedded Serial to Ethernet converter makes the device networked, which allows for remote control and access to the device. Also, there
is a liquid crystal display (LCD) in front of the unit panel that shows real time data including current time, frequency, voltage and the number of satellites acquired. Furthermore, this particular design allows plenty of space for development and the frequency estimation can be used in applications that involve power system monitoring, protection and control.

![Figure 2 - Picture of FDR](image)

1.2 Problem Statement and Proposed Work

The current FDR hardware design is not optimized for efficiency or cost. In addition, most of the hardware that is implemented within the device is unused. The CME555 board is not specifically aimed for FDR applications and the cost of the board is relatively high for the some of the unnecessary functionalities it provides. Also, the device is not so compact for installation in some circumstances. For every installation, the CME555 requires some on-board hardware configurations as well as establish connection to the GPS, ADC and the signal-conditioning unit. As a result, the need for optimizing FDR hardware has triggered the initiative for the design of a new FDR board. The main objective of the new FDR board is to integrate the GPS, ADC and signal
conditioning unit into one board, as well as to reduce the cost for manufacturing and installation of FDR. Also, the new FDR board needs to be able to support some of the basic characteristics of FDR. The board has to be able to accommodate a LCD of some kind that displays real time data and serial ports for GPS and server communication.

1.3 Thesis Organization

Much work has been done in the field of circuit design for FDR printed circuit board (PCB). This thesis presents a proposal for the design of a new FDR PCB, as well as a detailed procedure for establishing a final board design. Finally, results are presented for a PCB prototype implemented for interfacing to the LCD.

Chapter 1 is an overview of the background information for the project, including the current hardware architecture of FDR and problems that are associated with the current design.

Chapter 2 presents a detailed description of the current board Axiom CME555 that is currently implemented in the FDR. Also, some of the board design requirements are briefly discussed.

Chapter 3 goes through the hardware that are interfaced to the CME555 board.

Chapter 4 briefly describes the current FDR software architecture.

Chapter 5 investigates the MPC555 subsystems and the external logic that is needed for supporting the chip. Some of the earlier research efforts are presented with emphasis on the actual circuit design for MPC555.

Chapter 6 proposes a practical way of approaching the PCB design. Specifically, a prototype based on the PB555 board needs to be implemented for each of the external devices. The chapter concludes by discussing some of the results obtained from the LCD interface.

Chapter 7 summarizes the entire thesis work and suggests the future work.

Appendix A has a set of software initialization code for PB555. These initialization files are needed to setup configuration for the PB555 board. The Linker Command File was modified from that of CME555 provided by Metrowerks Codewarrior and the file is used to specify the location of RAM and ROM and other memory intensive parameters. Metrowerks provided the Runtime initialization file for
single chip operation, in this case the MPC56X single chip Runtime initialization file is used. Note that MPC56X has the same in-chip configuration as MPC555, so it can be used to replace the Runtime initialization file for the Axiom CME555. Furthermore, the memory configuration file is also based on a single chip MPC56x, this file allows for the proper display of memory during debugging.

Appendix B contains the code for driving the LCD to display characters. The C++ driver code in Appendix B was obtained from part of the FDR software, it simply displays the initializing messages to the LCD. The LCD initializing steps are referenced from the HD 44780 controller datasheet [37]. Also the EPLD (Erasable Programmable Logic Device) code in the end of Appendix B was received from Axiom, which is the same code that is implemented on the EPLD chip in the CME555 and used in the PB555 prototype.

Appendix C contains schematics of CME555 obtained from Axiom, which is referred to frequently throughout the thesis. Appendix C also shows the schematic for PB555 interfacing to the external components of FDR, the PB555 schematic was obtained from Axiom and the interfacing schematic was developed based on the discussions in Chapter 6 of this thesis.

Appendix D shows the current CME555 jumper and switches settings for FDR, which is included in the thesis to show how the board connections and configurations are setup in the current design.

Finally, Appendix E contains the netlisting files that were generated from the Protel schematics. The netlisting files are in EDIF format and they are separated into three different schematic files including the MPC555 chip connections, LCD and GPS interface connections and ADC and signal conditioning unit connections. The MPC555 netlisting was generated from the schematic in Figure 12 of Chapter 5 and the external device netlistings were generated from the PB555 external interface schematics in Appendix C. Appendix E was incorporated into this thesis to summarize the overall thesis work.
CHAPTER 2
AXIOM CME555

The Axiom CME555 board has a fully configured development system for the Motorola MPC555 PowerPC microcontroller. The system allows for Plug and Play and it has a large amount of external memory, 128K x 32 bits (512 Kbytes) Static RAM and 128K x 32 bits (512 Kbytes) Flash EPROM. In addition, the board has two memory sockets for additional memory expansion, two built-in RS-232 connectors on board, two TouCAN ports with transceivers, a LCD connector with contrast, a 16 key keypad connector, a large number of port connectors for MPC555 signals and a large prototyping area and Debug/JTAG Development connectors [1].

Physically, the CME555 has a board size of 7” x 7.5” [1]. Figure 3 [1] shows the physical allocation of different devices on the board. Specifically, the MPC555 chip is located close to the center of the board. In addition, there are rows of headers along the sides of the chip. Located below the MPC555 chip, there are several external memory chips along with latches and an EPLD chip. At the very bottom of Figure 3, there is the LCD and keypad port along with the contrast potentiometer. To the right of MPC555, there is a large amount of breadboard and expansion space alongside with connectors, testing LEDs and switches. At the very top of Figure 3, there are two RS232 connectors, a RS232 transceiver chip, several jumpers and switches. On the left side of MPC555 there are jumpers, external memory configuration switches and various connectors. More significantly, the BDM port is located just left of MPC555.

The physical architecture of CME555 is not optimized for the general operation of FDR. It is obvious that the CME555 board has a large amount of external memory that is not needed for the operation of FDR. In addition, there is the CAN (Controller Area Network) communication interface, the keypad interface, the external memory configuration switches, the testing LEDs and switches not used for the operation of FDR. An optimized PCB for the FDR would eliminate all of this unnecessary hardware to allow for a cheaper, cleaner and simpler solution. A replacement of the CME555 board would also affect the software that is currently used. Since the current FDR software is specifically for the MPC555-based system, the new board design will have to be MPC555-based as well to accommodate the software. Furthermore, to allow for
reusability of the current FDR software, the new PCB needs to be compatible with the current software. As a result, a good starting point in the design of FDR board would be a review of the CME555 hardware. One of the most significant aspects of the CME555
board is the large amount of memory it has. To better understand the memory structure of CME555, a memory map of the board is shown in Figure 4 [1].

<table>
<thead>
<tr>
<th>Internal Memory</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000 0006 FFFF</td>
<td>CMFI Internal Flash Memory Array</td>
</tr>
<tr>
<td>0007 0000 002F BFFF</td>
<td>Reserved</td>
</tr>
<tr>
<td>002F C000 003F 97FF</td>
<td>On-chip Control and Status Registers (see Figure 5)</td>
</tr>
<tr>
<td>003F 9800 003F FFFF</td>
<td>Internal SRAM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Memory</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0040 0000 007F FFFF</td>
<td>40 0000 – 40 FFFF for 64 K device 40 0000 – 47 FFFF for 512 K device CS 0 – External EPROM – 16 bit M-SEL devices U2-U3 (shipped with monitor program here)</td>
</tr>
<tr>
<td>0080 0000 00BF FFFF</td>
<td>CS1 – External SRAM – 32 bit RAM-SEL devices U4-U7 512K bytes at 80 0000 – 87 FFFF on this board</td>
</tr>
<tr>
<td>00C0 0000 00FF FFFF</td>
<td>CS2 – External Flash Memory – 32 bit FLUSH-SEL devices U15-U16 512K bytes at C0 0000 – C7 FFFF on this board (located on the board underneath the external EPROM)</td>
</tr>
<tr>
<td>0100 0000 013F FFFF</td>
<td>CS3 – External Peripheral Devices</td>
</tr>
<tr>
<td>0140 0000 FFFF FFFF</td>
<td>Un-Mapped Address Space</td>
</tr>
</tbody>
</table>

**Figure 4 - CME555 memory map**

Figure 4 shows the internal memory map of MPC555 as well as the external memory map of CME555 board. The external memory mapping is especially important since it would affect where the FDR software is placed as well as where the LCD is memory mapped. As seen in Figure 3, starting at memory location 0x00400000 there is the optional external memory expansion slot. Axiom ships the CME555 board with a monitor program preprogrammed within EPROM (Electrically Programmable Read Only Memory) units installed in the optional memory sockets. Memory block starting 0x00800000 to 0x00FFFFFF is dedicated to the static RAM (Random Access Memory) and Flash memory, each taking up 512 Kbytes of memory. Finally, memory location starting 0x01000000 to 0x013FFFFF is for keypad and LCD interfacing. For the new PCB design, some of the external memory can be removed. However, for software
compatibility issues, the original LCD memory allocation needs to stay intact. The ideal PCB design would have the LCD at the same memory location as the CME555 board.

2.1 MPC555 Processor

The MPC555 from Motorola is a 32 bit embedded microcontroller [24]. It is used most extensively in the automotive applications, such as engine control, transmission control, suspension and stability. It is also used in the robotics and avionics control industry. However, in the recent years, the MPC555 is rapidly being integrated into applications such as avionics, controls, analysis equipment, robotics and power management. In the case of frequency disturbance recorder, all of the frequency estimations are done within the MPC555 chip. The clock speed of MPC555 is able to run as high as 40MHz. And with 448 Kbytes of embedded FLASH memory and 26 Kbytes of Static RAM [24], the MPC555 allows for ease of modification or upgrading of the software implementation. Moreover, the MPC555 supports a wide range of on-board peripherals, these include third generation Time Processor Units (TPU3), two TouCAN Controller Area Network (CAN 2.0B) modules, and dual queued analog to digital converter (QADC) [24]. However, for the purpose of FDR operations, only the TPU3 is used. Each TPU includes its own RISC core and memory system and functions like another microcontroller within the MPC555. The TPU’s sole purpose is to provide timing functions capable of processing up to 20 million instructions per second with a 40 MHz system clock. The Controller Area Network (CAN) is a broadcast-based communication protocol. It is often utilized in automotive and industrial networking. Finally, the QADC within the MPC555 has 32 channel internally multiplexed or 81 externally multiplexed with 10 bit resolution with 5 μS conversion time. It is fully capable of simultaneously sampling two analog inputs [24]. However, since the internal QADC only allows up to 10 bits conversion with internal sample/hold, an external ADC is used instead to tolerate more precise frequency estimates.

To allow for communication with external devices, the MPC555 has many different kinds of Input/Output interfaces. The 24 address and 32 data pins can be used for general purpose I/O (GPIO). Also, many peripheral pins can be used for general purpose I/O as well when not used for primary function. Out of the 18 channels Modular
I/O System (MIOS) in the MPC555, 10 are used for double action submodules (DASM), 8 are dedicated to pulse width modulation (PWM) operations [24]. Also included in the MIOS are two 16 bit modulus counter submodules (MCSM) and two parallel port I/O submodules (PIOSM). For serial communication interface (SCI) and serial peripheral interface (SPI), the MPC555 introduces the Queued Serial Multi-Channel Module (QSMCM). The Queued Serial Peripheral Interface (QSPI) provides full-duplex communication port for peripheral expansion or inter-processor communication [24]. Also, the QSPI has four programmable peripheral-select pins that supports up to 16 devices. Each SCI provides 16 register receive buffer and 16 register transmit buffer, as well as separate transmitter and receiver enable bits and double buffering of data.

The memory map of MPC555 [24] microcontroller is shown in Figure 4. The available memory space for development includes 448 Kbytes of flash memory – address space 0x00000000 to 0x0006FFFF and 26 Kbytes of SRAM – address space 0x003F9800 to 0x003FFFFF. Control registers and IMB2 (intermodule bus) modules take up 64 Kbytes. In addition, Figure 6 [24] shows a block diagram of MPC555 with all of the subsystems and their bus connections shown.
<table>
<thead>
<tr>
<th>Address Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00 0000</td>
<td>CMF Flash A 256 Kbytes</td>
</tr>
<tr>
<td>0x04 0000</td>
<td>CMF Flash B 192 Kbytes</td>
</tr>
<tr>
<td>0x07 0000</td>
<td>Reserved for Flash (2.6 Mbytes – 16 Kbytes)</td>
</tr>
<tr>
<td>0x2F C000</td>
<td>USIU &amp; Flash Control 16 Kbytes</td>
</tr>
<tr>
<td>0x2F C000</td>
<td>USIU Control Registers 1 Kbyte</td>
</tr>
<tr>
<td>0x2F C800</td>
<td>FLASH Module A (64 bytes)</td>
</tr>
<tr>
<td>0x2F C840</td>
<td>FLASH Module B (64 bytes)</td>
</tr>
<tr>
<td>0x2F C880</td>
<td>Reserved for USIU</td>
</tr>
<tr>
<td>0x30 0000</td>
<td>UIMB Interface &amp; IMB3 Modules (32 Kbytes)</td>
</tr>
<tr>
<td>0x30 0000</td>
<td>DPTRAM Control (12 bytes)</td>
</tr>
<tr>
<td>0x30 2000</td>
<td>DPTRAM (6 Kbytes)</td>
</tr>
<tr>
<td>0x30 4000</td>
<td>TPU3_A (1 Kbyte)</td>
</tr>
<tr>
<td>0x30 4400</td>
<td>TPU3_B (1 Kbyte)</td>
</tr>
<tr>
<td>0x30 4800</td>
<td>QADC_A (1 Kbyte)</td>
</tr>
<tr>
<td>0x30 4C00</td>
<td>QADC_B (1 Kbyte)</td>
</tr>
<tr>
<td>0x30 5000</td>
<td>QSMCM (4 Kbytes)</td>
</tr>
<tr>
<td>0x30 6000</td>
<td>MIOS1 (4 Kbytes)</td>
</tr>
<tr>
<td>0x30 7080</td>
<td>TouCAN_A (1 Kbyte)</td>
</tr>
<tr>
<td>0x30 7480</td>
<td>TouCAN_B (1 Kbyte)</td>
</tr>
<tr>
<td>0x30 7884</td>
<td>Reserved (1920 bytes)</td>
</tr>
<tr>
<td>0x30 7F80</td>
<td>UIMB Registers (128 bytes)</td>
</tr>
<tr>
<td>0x30 7FFF</td>
<td></td>
</tr>
<tr>
<td>0x30 8000</td>
<td>Reserved for IMB3 (480 Kbytes)</td>
</tr>
<tr>
<td>0x38 0000</td>
<td>SRAM Control A (8 bytes)</td>
</tr>
<tr>
<td>0x38 0008</td>
<td>SRAM Control B (8 bytes)</td>
</tr>
<tr>
<td>0x38 0010</td>
<td>Reserved (485.98 Kbytes)</td>
</tr>
<tr>
<td>0x3F 9800</td>
<td>SRAM A (10 Kbytes)</td>
</tr>
<tr>
<td>0x3F C000</td>
<td>SRAM B (16 Kbytes)</td>
</tr>
</tbody>
</table>

Figure 5 - MPC555 memory map
2.2 External Memory

The Axiom CME555 board features two AM29LV2008 Flash Memory chips from AMD, four KM681002AJ Static RAM chips from Samsung and two on-board Option Memory EPROM banks for 16 bit wide memory. The Option Memory EPROM banks come with two NM27C256 memory chips. Each of these memory banks can be configured individually to operate from the MPC555 chip selects CS0, CS1, or CS2.

2.3 Jumpers and Switches

CME555 board provides many switches and jumpers for external hard reset configuration, external memory device selection and configuration, LCD and keypad port configuration, MPC555 power supply options, serial port options, as well as LEDs, test switches and potentiometer for development purposes.

Mode Switch 1, Mode Switch 2 and Config Switch provide the external Hard Reset Configuration Word when enabled [1]. The switches provide a logic 0 when it is in off position and a logic 1 when it is in the on position. Config Switch 4 is used to enable
External Configuration, when it is turned on, a Hard Reset signal will enable the External Reset Configuration Word from Mode Switches 1 and 2 on to the Bus for input by the MPC555. Config Switch 5 enables the EPEE signal to program the on-chip CMF flash [1]. Config Switch 6 provides the Vpp supply connection for programming the internal Flash EPROM Memory module. Config switches 1 to 3 provide the clock selection for the board [1].

External memory device selection is done with RAM-SEL, FLUSH-SEL and M-SEL Jumpers. The RAM-SEL jumper selects which chip select (CS) accesses the on-board Static RAM memory bank. This memory bank is often used for program debugging before programming into internal or external flash memory. The FLUSH-SEL jumper selects which chip select accesses the on-board EPROM memory bank. The memory bank is available for user to provide additional program code or data space and optional Boot start memory space. Finally, the M-SEL jumper is used to select which chip select accesses the on-board Option Memory EPROM bank. Once the external memory selection jumpers are removed, the memory space specified by that particular jumper will be disabled and becomes idle. However, the default memory selection sets the Static RAM to CS1, Flash EPROM to CS2 and the Option Memory EPROM to CS0. The MEM_OPT jumper selects the type of devices installed on the two optional memory sockets. Axiom has provided CME555 board with a simple Monitor program in the EPROM that is installed in these optional sockets. Finally, the MEM_VOLT Option jumper is used for selection of 5V or 3.3V for the power supply of the optional memory devices installed.

The rest of CME555 jumpers include configuration for LCD port, keypad port, serial port and MPC555 power supply. Default configuration has jumpers 10, 11 and 12 installed. Jumper 10 allows for selection of display power pin polarity on the LCD port, jumper 11 enables CS3 to be used for LCD and keypad port and jumper 12 enables IRQ4 to be used as a key input interrupt service. MPC555 cut-away jumpers are used for power supply configuration and are rarely used. Jumpers 5 to 9 control serial port configurations. COM1 and COM2 provide serial connection to the MPC555 internal serial ports SCI1 and SCI2 respectfully. SCI1 is connected to the RS232 transceiver device and SCI2 can be configured to use that same transceiver. If RTS/CTS handshaking
is needed, jumpers 5 through 7 and jumper 9 allows TPU channel 14 or 15 to be used as CTS or RTS signal. Lastly, jumper 8 can be installed to allow SCI2 to be connected to COM2. Option header jumper 5 provides connections to certain I/O ports of MPC555 or access to user components by other signals. See Appendix C for connections of jumper 5. As a conclusion, the current FDR operates with the CME555 jumper and switch settings shown in Appendix D.

2.4 Ports and Connectors

The LCD Port on the CME555 board provides a flexible connector to attach 80 character display modules, 160 character display modules and some graphics display modules with embedded drivers. For those specific applications of the FDR, an 80 characters display module is required to display all the data. The LCD and Keypad ports are memory mapped to chip select 3, or memory space 0x01000000 to 0x013FFFFF. Serial communication is implemented on COM1 and COM2 serial ports. MPC555 provide SCI1 and SCI2 for transmitting and receiving data bits. The CME555 uses a transceiver to translate the TTL voltage levels to RS232 voltage levels for data transmission to peripheral devices. Software development on the CME555 is usually performed using a BDM tool connected to the on-board BDM port connector. This provides real time access to all hardware, peripherals and memory on the board. BDM software such as Metrowerks Codewarrior allows for high-level source code debugging. During the development of the current FDR, the BDM port is accessed using Wiggler from Macraigor Systems Inc. The new board design will also seek to use the BDM port as the interface for software development. Finally, the CME555 provides port connectors for all the MPC555 subsystems and signals. See Figure 3 for each of the ports available on board.
CHAPTER 3
EXTERNAL PERIPHERALS

3.1 Signal Conditioning Unit

As seen in Figure 1, the signal from the transformer is the input for the signal conditioning system. The signal conditioning system includes a voltage divider, a passive low-pass filter, and an output bias circuit. The system must first take the 12Vrms signal from the transformer and attenuate it down to 20V peak to peak. Then the filter takes the 20V peak to peak signal as the input and outputs the appropriate signal that is readable by the ADC. The filter provides anti-aliasing and gets rid of high frequency noise [9]. As a result, the ADC gets a clean analog signal for conversion. See Appendix C for circuit diagram of signal conditioning unit.

3.2 ADC Unit

The AD976A is a high speed, low power, 16 bit sampling, analog to digital converter that can operate from a single +5 volt power supply. The AD976A takes standard +/-10 volt input range. With a 100/200kSPS throughput rate and a parallel interface [10], the AD976A is capable of connecting directly to microcontrollers. In the case of overall FDR system, (see Figure 1) the AD976A receives the voltage signal from the filter and samples 1440 times per second for the frequency algorithm [9]. Specifically, the AD976A uses the R/C’ (Ready/Convert’) line as a trigger for conversions and it is driven by a PWM signal from MIOS port of the MPC555. There may be inaccuracies or drift in the MPC555’s oscillator output but this problem can be corrected by measuring the one pulse per second (PPS) signal from the GPS receiver using the MDASM subsystem [14]. The AD976A does require some extra circuitry to operate. Some resistors and capacitors are needed to connect to its input and reference pins. See Figure 7 [10] for a functional diagram and see Appendix C for schematic. The AD975A is parallel-interfaced to the MPC555 through the GPIO subsystem and there are two control signals, one is R/C’ that trigger for conversion and the other is a inverted BUSY’ signal that is generated from the MPC555 IRQ[1] pin.
3.3 Motorola M12+ Receiver

Time synchronization for FDRs is provided by the Motorola M12+ GPS receiver. The synchronized signal of GPS is the 1PPS pulse train where the rising edge of the pulse indicates the second change of UTC (Coordinated Universal Time) time and the pulses that triggers analog to digital conversion should be in phase with the 1PPS. The M12+ does not only provide the 1PPS but also a timestamp for each second of time. The GPS software will format each timestamp to be the current UTC, in seconds referenced to midnight on January 1, 1970. The timestamp is then provided to the TCP/IP communications interface where it is attached to each frequency estimate.

The M12+ GPS receiver uses a 12 channel parallel receiver design and operates on a +3Vdc power supply. See Figure 9 [39] for a functional diagram. In addition, the receiver uses 3V CMOS/TTL serial interface for communication to host equipment. Timing accuracy for M12+ [4] is list below:

200s TTFF (Time To First Fix)-cold (with current almanac, position, time and ephemeris
50s TTFF-warm (with current almanac, position and time)
25s TTFF-hot (no stored information)
< 1.0s internal reacquisition

Positioning accuracy is within 25 meters with Selective Availability (SA) disabled [4]. The M12+ also features straight 10-pin power/data header for low-profile flat mounting against host circuit board [41]. Finally, the optional on-board Lithium battery allows for saving setup information and increasing the speed of satellite acquisition when the receiver is powered up after a period of inactivity. Finally, Figure 8 [38] shows the bottom view of M12+ GPS receiver.

Figure 8 - Bottom view of M12+ receiver
3.4 Serial to Ethernet Converter

In order to transmit data to the central server, there is a need for a TCP/IP device. Since there is no Ethernet port on the CME555, there needs to be a serial-to-Ethernet converter. The device that is currently used for serial-to-Ethernet conversion is the Moxa DE-311. The DE-311 provides a DB9 serial interface to the CME555 board and an RJ-45 Ethernet interface to a TCP/IP network. Furthermore, DE-311 uses CTS/RTS handshaking to accommodate for the binary data sent to the server.
CHAPTER 4
FDR SOFTWARE

The FDR has 3 states of operation: acquisition, initialization, and collection. In the acquisition phase, the GPS receiver gets a hold of the GPS satellites and receipt of UTC time. Since all the calculations are synchronized to the UTC time, the FDR unit will not proceed until a good GPS signal is established. The threshold for determining if a GPS signal is established is the receipt of 4 one-second time pulses from the GPS receiver [14]. An internal timer within the MPC555 is used to verify the accuracy of the pulses. In the initialization phase, the GPS signal has already been received and is providing accurate UTC time. The FDR begins collecting signal measurements to seed frequency algorithm. This initialization process takes up 1 second so that synchronized frequency measurements can be made at the start of every second afterwards [14]. If the GPS signal is lost during this stage, the FDR will go back to the acquisition state and discard all measurements collected. This makes certain that the FDR is using only current data with known times. Once the frequency algorithm has started, the FDR is able to generate frequency estimates in a continuous fashion until there is a loss of power or GPS signal. If GPS signal is lost, the FDR will go back to the acquisition phase and discard the current phasor and frequency calculations. A flow chart of FDR initialization procedures as well as the main program is displayed in Figure 10 [14] and Figure 11 [14] respectively.
Figure 10 - Flow chart of algorithm for FDR initialization
Figure 11 - Flow chart of algorithm for main program
 CHAPTER 5
PAST AND RELATED EFFORTS

5.1 Protel

To start the PCB design, it is necessary to create the circuit schematic. Once the schematic is developed, the PCB layout can be established. With the aid of modern day design software, circuit design becomes trivial once the concept is recognized. There are many circuit design software in the market today and the one that is used very prominently for PCB design is Protel DXP. As a result, a brief overview of the Protel software is needed before starting the FDR PCB design.

Protel DXP from Altium is a single application that provides all the functionalities needed to take a design concept to a board level design. Therefore, Protel is used to generate all of the design schematics shown in this thesis. Many of the functionalities of Protel are provided specially for PCB design. However, for the purpose of this research, the schematic editor was used most extensively. At the preface of PCB design, the circuit schematic is the first step to the final PCB design. Included with Protel DXP is a comprehensive set of integrated component libraries that combine each schematic symbol with their respective PCB footprints [16]. So once the schematic of the PCB has been established, the board layout can be completed fairly quickly. Protel’s PCB editor allows for specifying the board requirements and a hierarchical design rules and DRC system allows the control of all aspects of board design [16], verifying that the design is correct from start to finish.

5.2 MPC555 Support Pins

Design for FDR PCB started with a preliminary study of the 272 pins on the MPC555 chip. In order to establish a circuitry for the PCB, it is crucial to know which signals from the MPC555 are needed for FDR operations. Once the essential signals are known, those signals that are not essential to the operation of FDR will also become obvious. In general, the beginning efforts were concentrated on separating the MPC555 signals that are needed and not needed. The essential MPC555 signals include those that are needed to support the operation of MPC555 chip as well as the FDR. However, it is obvious that the internal QADC and TouCAN subsystem are not needed for the FDR.
The rest of the MPC555 subsystems are needed for operation of FDR in some manner. Before delving into the external interface for the MPC555, it is important to go through a detailed description of the MPC555 support signals and how they should be setup for proper operation of the chip. Some of the pins may have to be pulled high or low with resistors to keep electromagnetic interference from interfering with operation of the chip.

The MPC555 support signals include the clock, RESET, RESET configuration word, power supply, JTAG/BDM(background debug monitor) and bus control. The RESET pins consist of PORESET, HRESET and SRESET. Power on Reset (PORESET) initializes everything and is often used as a reset input when power is lost to the chip [25]. PORESET pin must be driven by an external source and cannot be left floating [13]. Hard Reset (HRESET) initializes registers other than those kept alive by keep alive power [25] and is used by Macraigor Wiggler to reset the processor via the background debug monitor connector. HRESET pin must be pulled up to 3.3V with a 10K pull up resistor [13]. Soft Reset (SRESET) can be used as a CPU output to memory or/and peripherals reset input pin [25]. It is recommended for SRESET to be pulled up to 3.3V with a 10K pull up resistor [13].

There are three options to use RESET configuration words (RSTCONF) [29]. But the CME555 board is setup to read the reset configuration from the internal flash shadow array [1]. The new design will keep this characteristic and the RSTCONF pin is pulled up through a 10K resistor to 3.3V (CMFCFIG register bit HC is programmed to 0) [29].

There are 5 support signals for MPC555 clock. External clock input (EXTCLK) [26] pin should be connected directly to the ground since there is no need for external clock input [13]. Engineering clock output/Backup clock output (ENGCLCK/BUCLK) pin is the engineering clock output, and the backup clock is operated when the chip is in limp mode, which enables a less precise on-chip oscillator to allow the system to continue minimum functionality until the system clock is fixed [26]. ENGCLCK/BUCLK pin should be pulled low with a 10K resistor to ground to reduce electromagnetic interference (EMI) [13].

XTAL and EXTAL pins are used to connect to either a 20 MHz or 4 MHz external crystal for the internal oscillator circuitry [26]. The FDR circuitry will use a 4
MHz external crystal connected to XTAL and EXTAL pins to reduce electromagnetic interference emission from the oscillator [13].

The external filter capacitance (XFC) pin is an input line for an external capacitor filter for the PLL circuitry [26]. One terminal of the capacitor is to be connected to XFC and the other terminal is to be connected to VDDSYN. The capacitor value is calculated as [26]:

\[
0 < MF + 1 < 4 \quad \text{(680 x (MF + 1) – 120) pF} \quad \text{Equation 1}
\]

\[
MF + 1 \geq 4 \quad \text{1100 x (MF + 1) pF} \quad \text{Equation 2}
\]

Where MF is one less than the desired frequency multiplication factor

For a 4 MHz XTAL the multiplication factor would be 10 (40 MHz operation) and one less of multiplication factor MF is 9. According to Equation 2, capacitance yields 11000 pF or 11 nF, which results in a 10nF capacitor to be implemented. Finally, the Clock out (CLKOUT) pin represents the external bus clock [29], and it is used to provide clock pulses to the EPLD on the CME555 board.

Joint Test Action Group (JTAG) [31]/ Background Debug Monitor (BDM) is a functional group within MPC555 that allows for uploading and debugging of FDR software. To permit compatibility of the firmware for FDR and compiler, the JTAG/BDM subsystem has to remain intact in the FDR board. MPC555 allows for three possible pin outs of the BDM connector to meet the various needs for different applications. Option 1 allows for maximum debug capability, which is recommended by Motorola and used in the CME555 board [13]. Option 2 gives the maximum external bus capability, which is the default power on reset condition [13]. Option 3 maximizes the input/output configuration [13]. In the case of FDR development, option 1 would yield the most efficient solution since it allows for compatibility to the current debugger hardware that is used, or the Wiggler.

Bus arbitration on the MPC555 is handled by 3 pins, specifically Bus Grant (BG), Bus Request (BR) and Bus Busy (BB) [28]. Bus Grant indicates external data bus status; it is low when the bus is granted to the MPC555 [28]. BG pin should be pulled low if the board is to have only a single chip system [13]. Bus Request indicates that the data bus
has been requested for external cycle [28]. Bus Busy indicates that the master is using the bus [28]. BR and BB pin should be pulled up to 3.3V if the board is a single chip system and pulled up with 10K resistor to 3.3V otherwise [13].

Power supply pins makes up the most pin counts on the MPC555. Global power supplies provide various voltage levels to the MPC555 for internal logic operation. Also there is the CDR MoneT flash (CMF) [32] power supply for the flash memory operation [27]. Since these power supplies are crucial to the internal operation of MPC555 and they do not affect interfacing with other components, the power supply circuit should be kept similar to that of CME555. However, special attention should be focused on the CMF power supply since CME555 has a large number of switches for configuring the flash memory. The FDR board should keep number of configuration switches to the minimum. Specifically, the FDR board needs just one jumper for providing power supply to the internal CMFI Flash memory. Once installed, the jumper enables +5V DC for programming internal CMFI flash memory and it should be removed when not programming to protect the internal flash memory during power cycles [32].

At this point, the internal support signals of MPC555 have been covered. The rest of the signals deal with external interfacing to other devices. There are a total of 8 external interrupt service pins on the MPC555 [3]. The interrupt service pins can request service routine by internal interrupt controller. However, the FDR only uses one interrupt service signal for triggering the analog to digital converter [14]. All of the interrupt service pins should be pulled up to +5V with 10K resistors to allow for programming as IRQs [13].

For the rest of the external interfacing signals, the unused bi-directional IO pins for MDA, PWM, GPIO and TPU should be pulled up with a 10K resistor to 5V [13]. But if electromagnetic interference becomes a major consideration, then pulling the unused pins low with a 10K resistor will reduce emissions [13]. In the FDR design (Figure 1), there is only one pin needed from MDA, one pin from PWM, one pin from IRQ and 16 GPIO pins from MIOS subsystem.

Finally, to summarize the preliminary study of MPC555 support pins, a schematic was drawn in Protel to show the MPC555 connections based on the study of the support pins (see Figure 12). As seen in Figure 12, some of the MPC555 pins are not used for the
operation of FDR and thus were left floating. The preliminary study has suggested connections for some of these pins to reduce electromagnetic interference [13] or provide

<table>
<thead>
<tr>
<th>Pin</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORESET</td>
<td>Cannot left floating, needs to be driven by external source</td>
</tr>
<tr>
<td>HRESET, SRESET, RSTCONF</td>
<td>Pulled up to 3.3V with 10K resistor</td>
</tr>
<tr>
<td>EXTCLK</td>
<td>Connect to ground with pull down 4.7K resistor</td>
</tr>
<tr>
<td>ENGCLK/BUCLK</td>
<td>Pulled low with 10K resistor to ground</td>
</tr>
<tr>
<td>XTAL/EXTAL</td>
<td>4MHz crystal, circuit setup same as CME555 (see Appendix C)</td>
</tr>
<tr>
<td>XFC</td>
<td>Place 10 nF capacitor between XFC and VDDSYN 4700 pF capacitor for XFC? (larger capacitor result in stable clock, time to lock will be longer)</td>
</tr>
<tr>
<td>BDM/JTAG</td>
<td>SRESET, HRESET, VFLS0, DSCK, VFLS1, DSDI, DSDO connection according to CME555 schematic (see Appendix C)</td>
</tr>
<tr>
<td>IRQ[1:7]</td>
<td>Pulled up to +5V with 10K resistor</td>
</tr>
<tr>
<td>Unused MDA, PWM, GPIO and TPU</td>
<td>Unused bidirectional I/O pulled up with 10K resistor to 5V.</td>
</tr>
<tr>
<td>BG’</td>
<td>Pulled low</td>
</tr>
<tr>
<td>BR’, BB’</td>
<td>Pulled up to 3.3V without resistor to allow for single chip system</td>
</tr>
<tr>
<td>EPEE</td>
<td>Pulled up to 5V</td>
</tr>
<tr>
<td>VDDH[1:10]</td>
<td>Connect to 5V supply</td>
</tr>
<tr>
<td>VDDL[1:8], VDDI[1:4]</td>
<td>Connect to 3.3V supply</td>
</tr>
<tr>
<td>VPP</td>
<td>5V applied when programming flash, use jumper connection</td>
</tr>
<tr>
<td>VDDSRAM</td>
<td>Connect to 3.3V with diode</td>
</tr>
<tr>
<td>VDDSYN</td>
<td>Filter circuit to keep noise low, uncertain</td>
</tr>
<tr>
<td>VSSSYN</td>
<td>Use CME555 connection (see Appendix C)?</td>
</tr>
<tr>
<td>ETRIG[1:2]</td>
<td>Connect to ground</td>
</tr>
<tr>
<td>TOUCAN pins</td>
<td>Uncertain, floating?</td>
</tr>
<tr>
<td>VRH</td>
<td>+5V OR +3V supply</td>
</tr>
<tr>
<td>VRL, VSSA</td>
<td>Connect to ground</td>
</tr>
<tr>
<td>ADC pins</td>
<td>Uncertain, left floating?</td>
</tr>
<tr>
<td>AAN[0:59], BAN[0:59]</td>
<td>Uncertain, left floating?</td>
</tr>
<tr>
<td>Unused WE[0:3], CS[0:3]</td>
<td>Uncertain, left floating?</td>
</tr>
<tr>
<td>Bus control pins *</td>
<td>Uncertain, left floating?</td>
</tr>
<tr>
<td>VDDA</td>
<td>Uncertain, left floating?</td>
</tr>
<tr>
<td>ECK</td>
<td>Pulled low with 10K resistor</td>
</tr>
<tr>
<td>GND</td>
<td>Connect to ground</td>
</tr>
</tbody>
</table>

*Bus control pins include BURST’, TSIZ[0:1], AT[0:3], RSV, PTR, RETRY’, BDIP’, TS’, STS’, CR, KR’, TA’, TEA’, BI’
the support logic needed by MPC555. To conclude this study, Table 1 [13] is compiled for the possible connections of MPC555 support signals. A detailed schematic for MPC555 connection is shown in Figure 12. As seen in Figure 12, most of the unused pins are pulled up to 5V supply with 10K resistors. These include the unused data and address pins, unused MDA, PWM, QGPI and TPU pins. However, some unused pins are still left floating due to uncertainty. These include the pins that support the QADC subsystem, TouCAN subsystem, and most of the bus control pins. Since all of these pins are used for input/output operations, they are susceptible to EMI and it is uncertain whether they should be pulled high or low. Nevertheless, to speed up the process for developing a new PCB, the research work has turned to a new direction. Instead of dwelling on the study of MPC555 signals, it would be more effective to take a step forward in the PCB design and start the implementation.

Figure 12 - Preliminary MPC555 pin configuration for FDR board design
6.1 Prototype Proposal

Initial PCB implementation started with a simple development board from Axiom. This approach to PCB prototyping was aroused from the uncertainty with MPC555 pin connections. Instead of concentrating on the details of MPC555 pins and the external circuitry that is needed for operation, the research effort has been shifted to the use of a simpler development board for a PCB prototype. The idea for using a simpler development board is driven by the need for a prototype strictly for PCB development. The development board to be used for prototyping needs to have the MPC555 chip mounted and be simple in circuitry. With a simple MPC555-based development board, there would be no need for going through all the support pins for the chip and the design process would move forward to interfacing of FDR components.

One ideal candidate for this prototype development is the PB555 from Axiom. With the introduction of PB555, the concept for PCB design became clear. The proposed final PCB design has the functional diagram shown in Figure 13. Since the MP555 needs to be interfaced to several external devices, there will be some tasks to be completed in order to establish the final version of FDR PCB. The tasks are listed as follow but not necessarily in this order:

- Interface the PB555 board to LCD
- Interface the PB555 board to GPS
- Interface the PB555 board to ADC and signal conditioning unit

The important aspect of this design approach is by breaking down the problem into parts, the problem can be approached in a more efficient manner. Each of the tasks listed above needs to be done separately to avoid wiring complexity and when all of the tasks have been successfully implemented, a final PCB design can be established.
6.2 PB555 board

The Axiom PB555 is a Motorola MPC555 module with I/O pin headers, oscillator circuit, power supply and one RS232 port [2]. In addition, the PB555 has the same BDM real-time debugger as the CME555. The PB555 was chosen for the prototype design of PCB because of its similarities to the CME555 board.
Since one of the requirements of the new board design is to have software compatibility with the CME555, it is important for the new PCB to be as similar to the CME555 as possible. However, the PB555 has some drawbacks in terms of communication interfaces. The PB555 has only one RS232 port and no LCD port [2], which indicates that the GPS and the LCD interfaces needs to be created in some manner. The ADC and device server can be interfaced to the PB555 board just like the CME555 since there is no need for additional external circuit. As a result, most of the research work is concentrated on interfacing the LCD and GPS to the PB555 board.

At the same time, it is also important to take into account the software compatibility issues. Specifically, the memory map of PB555 board is a little different from the CME555 board. Figure 15 [2] shows the memory map for PB555 board, it is important to note that there is no external memory involved and the board monitor utilities are located in memory space 0x00000000 to 0x0000FFFF [2]. Therefore, care should be taken when writing to the on-chip Flash memory so that the monitor utilities would not be accidentally erased.

The most critical aspect of this design is moving the code from external memory to the valid memory space within the MPC555 chip. Consequently, the software that was used for the CME555 board needs to be modified so that it can operate within the on-chip memory.
<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000</td>
<td><strong>Monitor Utilities</strong> – CMFI flash memory Debug monitor and CMFI programming tools</td>
<td>64K</td>
</tr>
<tr>
<td>0001 0000</td>
<td><strong>User Program Memory</strong> – CMFI flash memory Can be used for non-volatile program and data storage</td>
<td>380K</td>
</tr>
<tr>
<td></td>
<td>The utilities also re-map interrupt vectors here starting at 0x10100 for the RESET vector and continue to 0x12000</td>
<td></td>
</tr>
<tr>
<td>0007 0000</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>002F C000</td>
<td>Control and Status Registers</td>
<td></td>
</tr>
<tr>
<td>0030 2000</td>
<td><strong>User RAM</strong> – DPTRAM Array</td>
<td>6K</td>
</tr>
<tr>
<td>0030 3800</td>
<td>Control and Status Registers</td>
<td></td>
</tr>
<tr>
<td>003F 9800</td>
<td><strong>Monitor Utilities RAM</strong> – SRAM Array Used by monitor and CMFI programming tools</td>
<td>2K</td>
</tr>
<tr>
<td>003F A000</td>
<td><strong>User RAM</strong> – SRAM Array</td>
<td>24K</td>
</tr>
<tr>
<td>0040 0000</td>
<td>Unused</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 15 - PB555 Memory map**

### 6.3 Prototype Development

Once the development board is specified, the next step is to select a method for connecting the board to external devices. Since the PB555 board does not have any connectors for the MPC555 signals, connecting the board to external devices becomes a challenge. The most frequently used method for electrical interconnection is soldering, which does require a certain degree of aptitude and a large amount of time to do well, thus the idea was abandoned quickly. Another solution was proposed instead and it was accepted as the best way to approach the prototype connections. Wire wrapping has been in existence for over fifty years [11] and it is used most significantly for interconnection on PCBs.
Wire wrapping was originally developed in the late 1940’s by Bell Labs in the United States. Up until that time, soldering was the commonly used method for cable interconnection. Most importantly, soldering presented many problems in electrical interconnections [11]. Solder joints can easily fail if the soldering process is not just right and it is quite messy to attempt rework of a defective one. Mechanically it is brittle, weakens with age and varies widely in strength from connection to connection [11].

With the introduction of wire wrapping into the electronics world, the problems presented by soldering were quickly diminished. Wire wrapping is a process for connecting stripped copper wire to a terminal to form an electrical circuit. The wires are tightly coiled around the terminal (see Figure 16 [11] and Figure 17 [11]) through the motion of a rotating equipment, called a bit. The bit is produced from solid steel rods and has a large central hole along its axis for slipping the tool over the terminal. The terminal will remain inside this cavity as the bit rotates. The top surface of the bit has a wire slot, which accepts the length of stripped wire that is to be wrapped. While the bit is in operation, it rotates and pulls the wire across the sharp corners of the terminal.

This process gives the wire a large amount of tension and compression, typically 100,000 PSI at the terminal’s edge [11]. The wrapping action causes both the wire and terminal to diffuse. This results in a connection of high mechanical strength with very low electrical resistance. It also allows a gas-tight connection with a design life of 40 years [11], yet the wires can be easily unwrapped to correct any wiring errors or to replace the equipment to which the wire is terminated. This high level of reliability and environmental resistance makes wire wrapping a preferred method for building the PB555 prototype.

![Figure 16 - Wire wrapping - View from the top of terminal](image-url)
6.4 LCD Interface

To determine how the LCD should be interfaced to the PB555, a study was conducted to see how this device was interfaced to the CME555. The study indicates that the LCD is interfaced by memory mapped I/O with the use of an EPLD (Electrically Programmable Logic Device) chip for address decoding. See Figure 18 [22] and Figure 19 [22] for LCD interfacing in CME555.

In memory-mapped I/O the LCD device has an address just like a memory location. The MPC555 has its lower 14 address lines going into the AT22LV10 for address decoding. When the processor needs to read or write to the LCD, it can select the memory location specifically for the LCD. According to Figure 18 [22] the EPLD AT22LV10 generates a LCD chip select signal that makes sure the LCD chip is selected when it is intended. Since the LCD chip will always be memory mapped to CS3 [1], CS3 should always be connected to AT22LV10. The CLKOUT line from MPC555 generates the clock pulses and the R/W* line provides the read/write control signals for the EPLD. As a result, the AT22LV10 generates two chip select signals for the LCD and one chip select for Keypad ports [1].
For the LCD display connection, there is a LCD port on the CME555 board, which is connected according to Figure 19 [22]. The LCD port on CME555 can be connected via either to the left or the right two columns of pins. Actual connection is only made for 14 pins since there is connection between signals on either side of the port. The
current FDR uses a ribbon cable for making the connection between the port and the display. To clarify the connections, a mapping of signals between the LCD port and the display is given in Table 2. Table 2 shows where the signals are generated for each

<table>
<thead>
<tr>
<th>CME555 LCD port (Pin number)</th>
<th>MPC555 signal (Pin number)</th>
<th>DMC20261NY-LY-AXE display (Pin number)</th>
<th>DMC20261NY-LY-AXE display (Pin name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>1</td>
<td>( V_{SS} )</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
<td>2</td>
<td>( V_{DD} )</td>
</tr>
<tr>
<td>4</td>
<td>**</td>
<td>3</td>
<td>( V_{EE} )</td>
</tr>
<tr>
<td>5</td>
<td>A31</td>
<td>4</td>
<td>RS (Register Select)</td>
</tr>
<tr>
<td>7</td>
<td>R/W*</td>
<td>5</td>
<td>R/W (Read/Write)</td>
</tr>
<tr>
<td>8</td>
<td>LCDCS1*</td>
<td>6</td>
<td>E (Enable)</td>
</tr>
<tr>
<td>10</td>
<td>D7</td>
<td>7</td>
<td>DB0</td>
</tr>
<tr>
<td>11</td>
<td>D6</td>
<td>8</td>
<td>DB1</td>
</tr>
<tr>
<td>13</td>
<td>D5</td>
<td>9</td>
<td>DB2</td>
</tr>
<tr>
<td>14</td>
<td>D4</td>
<td>10</td>
<td>DB3</td>
</tr>
<tr>
<td>16</td>
<td>D3</td>
<td>11</td>
<td>DB4</td>
</tr>
<tr>
<td>17</td>
<td>D2</td>
<td>12</td>
<td>DB5</td>
</tr>
<tr>
<td>19</td>
<td>D1</td>
<td>13</td>
<td>DB6</td>
</tr>
<tr>
<td>20</td>
<td>D0</td>
<td>14</td>
<td>DB7</td>
</tr>
</tbody>
</table>

* Generated from EPLD
** Generated from power supply circuit

pin the LCD. Note that the register select line is generated from the least significant address line of the processor, R/W signal is connected to the R/W* from MPC555 and Enable is generated from LCD chip select 1.

To better understand how the LCD operates, it is important to go through an analysis of the timing diagrams specified by the LCD. Figure 20 [5], Figure 21 [5] and Table 3 [5] shows the timing diagram and specific timing constraints for each signal entering the LCD. Figure 20 illustrates signal timing when there is a write operation on the LCD and Figure 21 shows what signal timing when there is a read operation on the LCD. In order to read or write to the LCD, the processor must provide stable data within the time limits shown relative to the clock.

Table 3 indicates that for the write data operation, minimum setup time is 80 ns. So the time period for when there is valid data on the data bus to when Enable signal starts to fall has to be at least 80ns. Likewise, Table 3 also indicates that the minimum time period for after the fall of Enable signal to the end of valid data has to be 10 ns.
Consequently, these two criteria need to be fulfilled when writing to the LCD. When reading data from the LCD, the maximum delay time or the maximum time it takes from the rise of Enable signal to when there is valid data on the data bus has to be 160 ns. Again, the minimum hold time for reading data is 5 ns. As a conclusion, there are a total of four timing criteria to be fulfilled when writing or reading to the LCD.

**Table 3 - DMC20261NY-LY-AXE timing constraints**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Cycle Time</td>
<td>$t_{CYC}$</td>
<td>500</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Enable Pulse Width</td>
<td>$PW_{EH}$</td>
<td>230</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Enable Rise/Fall Time</td>
<td>$t_{ER}, t_{Ef}$</td>
<td>-</td>
<td>20</td>
<td>ns</td>
</tr>
<tr>
<td>Address Setup Time</td>
<td>$t_{AS}$</td>
<td>40</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Address Hold Time</td>
<td>$t_{AH}$</td>
<td>10</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Write Data Setup Time</td>
<td>$t_{DSW}$</td>
<td>80</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Write Data Hold Time</td>
<td>$t_{DHW}$</td>
<td>10</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Read Data Delay Time</td>
<td>$t_{DDR}$</td>
<td>-</td>
<td>160</td>
<td>ns</td>
</tr>
<tr>
<td>Read Data Hold Time</td>
<td>$t_{DHR}$</td>
<td>5</td>
<td>-</td>
<td>ns</td>
</tr>
</tbody>
</table>
Figure 20 - Write Operation Timing for DMC20261NY-LY-AXE

Figure 21 - Read Operation Timing for DMC20261NY-LY-AXE
After the analysis of the timing criteria for reading/writing operation to LCD, it is necessary to isolate the AT22LV10 EPLD and study how it functions as an address decoder. Since the AT22LV10 was implemented on the Axiom CME555 board, the EPLD software is available to the public for development. See Appendix B for the EPLD code in PLD format. The EPLD codes basically setup the pins for input/output and implements the decoding logic using the R/W* strobe, address, and chip select signals.

The next step was to take action in the implementation of the PB555 prototype. As indicated earlier, the FDR software compatibility issue comes into play if the design is differentiated from that of CME555. To speed up the process of building the prototype and be able to reuse the same FDR software, it was decided the LCD interface that is to be implemented on the PB555 board to be the same as on the CME555 board. Moreover, the address-decoding scheme that was used in CME555 is implemented the same way in

![Figure 22 - Schematic of LCD interface for PB555 board](image)

the PB555 prototype. See Figure 22 for a schematic of LCD connections of the PB555 prototype. Figure 22 shows that the address decoding is implemented with Atmel ATF22LV10CZ, which is a new version of the AT22LV10 chip. The AT22LV10 chip
that was used on the CME555 board has become old and is not supported by most of the modern EPLD programmers. Therefore, this particular EPLD is to be replaced by the ATF22LV10CZ, which is fusemap and pin-to-pin compatible with the AT22LV10 [6]. According to Atmel, the AT22LV10CZ has pin-keeper circuits on all of its input and I/O pins while the AT22LV10 does not, so as long as the outputs are not considered when they are tri-stated, the pin-keeper should not affect the decoding logic. In effect, by using the same AT22LV10 software from Axiom (See Appendix B), the address decoding should work in the same way as it was on the CME555 board.

Since the proposed decoding algorithm is the same as that of CME555 board, the code in Appendix B is implemented in the ATF22LV10CZ chip. The implementation was done using an EPLD programmer, the EMP-11 from Needhams Electronics Inc. EMP-11 universal device programmer supports devices that include EPROM’s, EEPROM’s, FLASH, 8051 microcontrollers, Microchip PIC, PLDs and other specialty devices [20]. The main advantage of the EMP-11 is its compatibility with Windows operating system, which makes it user-friendly compared to other programmers. Also, the programmer communicates with the PC through a standard parallel port cable [20]. The programmer also comes with EMPWin software that is specifically for programming different devices from a PC. Within the EMPWin software, there is a main menu where the basic functions of a programmer can be accessed. The programmer includes functions such as erase, blank check, program, verify and read. To initiate connection to the device of interest, simply select the device in the device selection menu and all the functions will become accessible. The code in PLD format was uploaded to the ATF22LV10CZ chip successfully.

One of the concerns during the PCB design was whether there is a need for the use of a contrast circuit for the LCD. The CME555 board has a potentiometer for adjusting voltage levels supplied to the LCD for contrast adjustment. The DMC20261NY-LY-AXE manual recommends a potentiometer for the contrast adjust due to the tolerance of the driving voltage and its temperature dependence [5]. An example power supply circuit is shown in Figure 23 [5]. The PB555 prototype power supply is designed according this example circuit and the electrical specifications indicated in Table 4 [5]. The power supply for logic $V_{DD}$ can be generated by the +5V supply from
MPC555 but the power supply for LCD drive $V_{EE}$ needs to be negative as a bias source [5]. This bias voltage cannot be generated straight from MPC555 and extra circuitry is needed to generate this voltage. Two different approaches are available to resolve this problem and each one has its own trade-offs in terms of implementation or efficiency.

One approach would be switch to another LCD that has a voltage converter built inside to generate the bias voltage and the other approach is to get a separate voltage converter chip. By switching to another LCD, there wouldn’t be the need to have extra circuitry for power supply. But the LCD dimensions are constrained so that it can be able to fit inside the FDR case.

To save time and speed up the process for building the prototype, it was decided to get a separate voltage converter chip for generating the bias voltage. The voltage converter implemented in the prototype is ICL7660 from Intersil. This is a CMOS voltage converter that performs supply voltage conversions from positive to negative for an input range of $+1.5V$ to $+10.0V$ resulting in complementary output voltages of $-1.5V$ to $-10.0V$ [23]. The operation of the voltage converter requires that two external capacitors used, one for a charge pump function and one for a charge reservoir function [23] (See Figure 22 for schematic). With the ICL7660, a $+5V$ power supply from

![Figure 23 - Example power supply circuit for LCD module](image-url)
MPC555 can be translated to –5V for bias voltage of LCD and the power supply circuit for LCD is complete.

As it was discussed earlier, the prototype is implemented with wire wrapping. The idea is to get wire wrap sockets for the EPLD and the voltage converter and wire wrap the sockets to the PB555 board. The wire wrap socket is just like the regular DIP socket with special wire wrapping pins for connection. Once the sockets are wire wrapped to the PB555 board, the EPLD and voltage converter chips can be plugged into the sockets for testing. During the development phase, the sockets allows for easy removal of the chip for debugging or replacement. Likewise, the LCD DMC20261NY-LY-AXE has header connectors on the bottom, which allows for wire wrapping to be implemented. Once the LCD interfacing has been established physically, it is ready for a software driver to test it.

![Wire wrapped PB555 prototype for LCD interface](image)

**Figure 24 - Wire wrapped PB555 prototype for LCD interface**

6.5 LCD Test
The DMC20261NY-LY-AXE display uses Hitachi HCD44780 controller. See Figure 25 [5] for a functional diagram of the controller. HCD44780 has two 8-bit registers, an instruction register (IR) and a data register (DR) [35]. The instruction register stores instruction codes such as display clear and cursor shift, and address information for display data RAM (DD RAM) and character generator RAM (CG RAM) [8]. DD RAM stores display data represented in 8-bit character codes, its capacity is 80x8 bits, or 80 characters [8]. With displays using fewer than 80 characters, DD RAM can be used as a general data RAM. CG RAM is RAM with which the user can redefine character patterns in software [8]. The instruction register can be written from the processor but cannot read by the processor. The data register temporarily stores data to be written into the DD RAM or the CG RAM and data to be read out from the DD RAM or the CG RAM [8]. When address information is written into the instruction register, data is read into the data register from the DD RAM or the CG RAM by internal operation. Data transfer to the processor is then completed by reading data register. After the processor reads the data register, data in the DD RAM or CG RAM at the next address is sent to the data register for next read from the processor [8]. Register select (RS) signal indicates register selection between data and address. Table 5 [35] shows the operation of HCD44780 using register select, read/write and Enable line.
Writing the LCD driver software requires referencing to the instruction set that is available for the controller. See Table 6 [34] for instruction set and its respective operation code. To make sure the LCD displays characters correctly, there are several initialization steps it must go through after power-on (see Appendix B). The initialization must consist of at least a Function Set command, followed by Display Control, Clear Display and Entry Mode Set [36]. Issuing each of these commands ensures that the programmer knows

**Table 5 - HCD44780 controller register selection**

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>Enable</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>High, then High to Low</td>
<td>Write to Instruction Register</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>High</td>
<td>Read busy flag indicated by DB7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>High, then High to Low</td>
<td>Write data from processor to HD44780</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>High</td>
<td>Read data from HD44780 to processor</td>
</tr>
</tbody>
</table>

**Table 6 - Operation code for HCD44780 instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear Display</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Return Home</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>Entry Mode Set</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>I/D</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Display ON/OFF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>DOF</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>Cursor and Display Shift</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>S/C</td>
<td>R/L</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Function set</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>DL</td>
<td>N</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set CG RAM address</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>CG RAM address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set DD RAM address</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DD RAM address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read bus flag and address</td>
<td>0</td>
<td>1</td>
<td>BF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CG RAM/DD RAM address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write data to CG or DD RAM</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read data from CG or DD RAM</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Read data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* - Don’t cares (either 1 or 0)  
B - Blink
C - Cursor display
DOF - Display On/Off
I/D – Increment/decrement DD RAM address
BF - Busy Flag
DL - Data Length
N - Number of display lines
F - Character font
S/C and R/L - Cursor and display shift

the state of the display. After the initialization is finished, the HCD44780 is ready for read and write operation. The pseudo-code for read/write to LCD is shown below:

Reading from a register
1. Set Register Select line high or low to indicate which register to access
2. Set Read/Write line high to indicate a read
3. Set Data Port for the processor to input
4. Set Enable line high to begin read cycle
5. Set delay to allow LCD to fetch data
6. Read data from Data Port of the processor
7. Set Enable line low to finish read cycle

Writing to a register
1. Set Register Select line high or low to designate the register to access
2. Set Read/Write line low to indicate a write
3. Set Data Port for the processor to output
4. Write data to Data Port of processor
5. Set Enable line high to begin write cycle
6. Set delay to allow LCD to accept the data
7. Set Enable line low to finish read cycle

The actual LCD driver for testing is listed in Appendix B. This driver program only displays the initialization messages for the FDR. Since FDR firmware development was all done in Metrowerks Codewarrior and implemented using C++, the LCD test driver was also implemented in the same programming environment. However, ever since the beginning of FDR development, there have been numerous upgrades for
Metrowerks Codewarrior. In order to keep up with the latest programming environment, the LCD test driver was implemented in the newest version of Codewarrior Development Studio for MPC5xx - version 8.1. All the code development and debugging were done in Codewarrior Integrated Development Environment (IDE).

The approach to testing of the PB555 prototype begins by implementing the LCD test driver on the CME555 board. Once it is verified that the test driver is able to drive the LCD on CME555, the test driver can then be transferred to the PB555 prototype with some modifications in memory allocation and initialization settings. In Codewarrior IDE, memory allocation can be setup within EPPC Linker and linker command file [18]. The initialization settings for the board involves several files such as debug initialization and memory configuration files [19]. When it was verified that the code in Appendix B is able to drive the LCD successfully on the CME555 board, the same code can be used on PB555 prototype to test LCD interface.

Early research efforts were focused on the study of PB555 memory mapping. Since the software driver was verified to be bug-free, the rest of the work would be placing the code within the appropriate memory space in the PB555 board. However, Metrowerks Codewarrior does not support PB555 board, the board initialization and configuration files will have to be derived from that of CME555. See Appendix A for initialization and configuration files for the PB555 board modified from that of CME555. Generally, the goal was to move the code from CME555 external memory to the internal memory of MPC555. Table 6 shows the modifications that were implemented in Codewarrior settings, see Appendix B for details on coding.

<table>
<thead>
<tr>
<th>Table 7 - Memory initialization modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CME555 (beginning address)</td>
</tr>
<tr>
<td>Flash source</td>
</tr>
<tr>
<td>RAM</td>
</tr>
<tr>
<td>ROM</td>
</tr>
<tr>
<td>Stack</td>
</tr>
<tr>
<td>Code (Debug target)</td>
</tr>
<tr>
<td>Code (ROM target)</td>
</tr>
<tr>
<td>Code (Flash target)</td>
</tr>
<tr>
<td>RAM buffer (Flash)</td>
</tr>
<tr>
<td>ROM image (Flash)</td>
</tr>
</tbody>
</table>
From the configurations listed in Table 7, the project was compiled using debug target within Codewarrior. The following shows part of the resulting mapping file that was generated from the debug target:

Memory map:

<table>
<thead>
<tr>
<th>Starting Address</th>
<th>Size</th>
<th>File Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>.compress.init</td>
<td>003fa000</td>
<td>0000000000 00000180</td>
</tr>
<tr>
<td>.init</td>
<td>003fa000</td>
<td>00000378 00000180</td>
</tr>
<tr>
<td>.compress.text</td>
<td>003fa378</td>
<td>0000000000 000004f8</td>
</tr>
<tr>
<td>.text</td>
<td>003fa378</td>
<td>00000000bc 000004f8</td>
</tr>
<tr>
<td>.compress.fini</td>
<td>003fa834</td>
<td>0000000000 000009b4</td>
</tr>
<tr>
<td>.fini</td>
<td>003fa834</td>
<td>0000000000 000009b4</td>
</tr>
<tr>
<td>.rodata</td>
<td>003fa834</td>
<td>0000000000 000009b8</td>
</tr>
<tr>
<td>.ctors</td>
<td>003fa834</td>
<td>0000000008 000009b4</td>
</tr>
<tr>
<td>.dtors</td>
<td>003fa83c</td>
<td>0000000008 000009bc</td>
</tr>
<tr>
<td>extab</td>
<td>003fa844</td>
<td>0000000040 000009c4</td>
</tr>
<tr>
<td>extabindex</td>
<td>003fa884</td>
<td>0000000080 00000a04</td>
</tr>
<tr>
<td>.BINARY</td>
<td>003fa904</td>
<td>0000000000 00000a84</td>
</tr>
<tr>
<td>.data</td>
<td>003fa908</td>
<td>000000002c 00000a88</td>
</tr>
<tr>
<td>.got</td>
<td>003fa934</td>
<td>0000000000 00000ab4</td>
</tr>
<tr>
<td>.sdata</td>
<td>003fa938</td>
<td>0000000004 00000ab8</td>
</tr>
<tr>
<td>.sbss</td>
<td>003fa940</td>
<td>0000000010 00000ac0</td>
</tr>
<tr>
<td>.sdata2</td>
<td>003fa950</td>
<td>0000000000 00000ac0</td>
</tr>
<tr>
<td>.sbss2</td>
<td>003fa950</td>
<td>0000000000 00000ac0</td>
</tr>
<tr>
<td>.PPC.EMB.sdata0</td>
<td>fffffff8000</td>
<td>0000000000 00000ac0</td>
</tr>
<tr>
<td>.PPC.EMB.sbss0</td>
<td>fffffff8000</td>
<td>0000000000 00000ac0</td>
</tr>
<tr>
<td>.bss</td>
<td>003fa950</td>
<td>0000000110 00000ac0</td>
</tr>
<tr>
<td>.debug_abbrev</td>
<td>0000012b</td>
<td>00000ac0</td>
</tr>
<tr>
<td>.debug_info</td>
<td>000014a5</td>
<td>00000beb</td>
</tr>
<tr>
<td>.debug_line</td>
<td>00001f71</td>
<td>00002090</td>
</tr>
<tr>
<td>.debug_loc</td>
<td>0000068c</td>
<td>00004001</td>
</tr>
<tr>
<td>.debug_pubnames</td>
<td>0000034a</td>
<td>0000468d</td>
</tr>
</tbody>
</table>

Linker generated symbols:

- _f_compress_init 003fa000
- _f_compress_init_rom 003fa000
- _e_compress_init 003fa000
- _f_init 003fa000
- _f_init_rom 003fa000
- _e_init 003fa378
- _f_compress_text 003fa378
- _f_compress_text_rom 003fa378
- _e_compress_text 003fa378
- _f_text 003fa378
The mapping file verifies that the driver program has been uploaded to the MPC555 chip in the correct locations. However, some important observations can be made from the mapping file. The stack memory location indicates that there are 16Kbytes resorted to stack use and 4Kbytes resorted to heap use. The default configuration in Codewarrior allocates 16Kbytes for stack and heap, but this is not possible on the PB555 board since there is only 24 Kbytes of accessible RAM space. Therefore, care should be taken when uploading to RAM space since there is restricted amount of space in comparison to CME555. Future developments in FDR firmware may run out of RAM space on the PB555 board since the ROM and Flash target does require more on-board RAM space to operate. In particular, the ROM and AutoFlash target requires a RAM buffer space allocated for flashing the program to ROM.

AutoFlash target also requires some code that programs the flash memory, which takes up the RAM buffer space [18][19]. If RAM does become insufficient in the future for AutoFlash target, another option would be using the Codewarrior built-in Flash programmer. The built-in Flash programmer does not need the extra RAM space but it does take more time to program the flash memory than AutoFlash target [19]. Depending on the specific need for development, more RAM chips may be needed to interface to the board in the future.

Once the code has been placed in the appropriate RAM memory space, the LCD interface is ready to be tested. Codewarrior debugger indicates that during the initialization procedures for LCD, the instruction register of the LCD keeps returning a Busy Flag. In particular, the processor reads the instruction register and returns the value
‘0xFF’ repeatedly. As a result, the debugger suspends in checking the Busy Flag. To make sure that it was not the hardware that caused the problem, the same program (see Appendix 2) was tested on the CME555 board using the PB555 memory configuration. Debugger shows that the instruction register returns the same value ‘0xFF’ and suspends during code execution. Clearly, the processor is not reading the correct value from the instruction register. In view of the fact that the only change made was where the program is placed in memory, it is difficult to find the source of the problem by debugger. With the time limit imposed in this research effort, the LCD interface for PB555 prototype is currently still undergoing testing.

To conclude the LCD interface discussion, the root of the problem may lie in one of the following:
- To access the external LCD memory space, the code has to be placed within a certain memory space – CME555 has the code allocated in external Flash for debug target
- Problem with board initialization or configuration. Since Codewarrior cannot provide these overhead files for PB555, there is not enough information given to determine the source of the problem.

6.6 GPS Interface

As it was indicated earlier, the current FDR implements timing synchronization using the M12+ GPS receiver. However, Motorola will discontinue the production of M12+ at the end of 2005 [17]. Therefore new FDRs need to have a different GPS receiver to replace the M12+. At this point, several GPS have been explored and the most promising one being the M12M from Motorola. The M12M is a direct replacement to the current M12+ receiver. In addition the M12M has the same mechanical dimensions, same options, same communication commands and it has improved Time to First Fix by 50 seconds for a Cold Start [17]. At the same time, power consumption on the M12M has reduced to 150 mW whereas the M12+ consumes 195mW. Finally, the M12M will be launched in the December of 2005 [17].
Since the M12M seems to be a good replacement for M12+ and it is backwards compatible with M12+, the design for a new FDR PCB continued with M12+ as the timing synchronizer. M12+ receiver is interfaced to the main board via serial interface. As mentioned earlier, there is a serial port on M12+ for DB-9 cable connection to the CME555 board. The same implementation would not be possible for the PB555 prototype. PB555 has only one serial port available, and it must be used for connection to the TCP/IP device since the software driver requires RTS/CTS handshaking [14], where RTS is the request to send signal and CTS is the clear to send signal.

The GPS interface will be restricted to a connection setup without hardware handshaking. This interface can be implemented by wire wrapping the M12+ to the PB555 using null modem connection without hardware handshaking. Note that this would not be possible if the FDR software implements GPS interface using RTS/CTS handshaking. See Figure 26 for data communication connections. Flow control for this particular interface is done in software only; hardware flow control is not implemented.

However, there is one flaw with transmitting and receiving data via TxD and RxD, the M12+ serial ports are not 5V logic compliant [42]. MPC555 processor SCI transmits data using 5V CMOS logic [3], which may cause damage to GPS serial port even though their logic levels are compatible. The 3V low voltage TTL devices like the M12+ cannot withstand more than approximately 3.3V on their I/O pins. If an I/O pin of a 3V TTL device is driven above its 3.3V limit, the P Channel device in the output driver will conduct, causing current flow from the bus to the 3.3V input through the device. The resulting high current flow can cause destruction of device by latchup effects [15].

One possible solution to this is implementing another serial port by using a RS-232 driver/receiver IC such as Maxim’s MAX3232 [33]. But this solution would overcome the purpose of using the PB555 for prototyping since the proposed solution would require only one serial port for RTS/CTS handshaking. The implementation of another serial port would not be best choice for the PCB design.
Another solution involves getting a TTL converter chip that drops the 5V TTL to 3V TTL. One TTL converter that is used prominently in the industries is the QuickSwitch QS3861 5V to 3V converter chip. The QS3861 provides a set of ten high speed CMOS TTL compatible bus switches [15]. The $5\,\Omega$ ON resistance of the QS3861 allows inputs to be connected without adding propagation delay [15]. Figure 27 [15] shows a power supply circuit for the QS3861. By supplying 4.3V to the Vcc pin of QS3861, the driven output will be limited to 3.3V maximum [15] to avoid damaging the GPS serial port. A 4.3V power supply can be created by adding a diode between the 5V supply and the device (see Figure 27). The diode will provide approximately 0.70V drop, thereby creating the 4.3V supply [15]. The 10k$\,\Omega$ resistor is added between the diode’s cathode and ground to provide a current path for the diode [15].
10 pins on the M12+ receiver and the best way to make connections to these pins is having a port on the PCB to hold the GPS receiver. However, during the initial testing stages, the GPS can be wire wrapped to the PB555 board.

Before making connections to the GPS pins, a review of the pin functions is necessary. See Table 8 [41] for M12+ pin functions. Pins 1 through 5 are connected to the MPC555 like shown in Figure 28. Pin 6 uses optional backup battery, which is not a mandatory connection and can still have the same effect by applying an external backup voltage source. If without any backup power, the receiver must perform a “Cold Start”, where none of the setup information is available [40]. The Time to First Fix (TTFF) will be longer than if the information had been available [4]. For testing purposes, the optional backup battery will not be implemented in the prototype. However, it can be implemented in the final PCB design by simply adding an external voltage supply [40]. Pin 7 and pin 10 can be ignored since they are reserved and have no functionalities [41]. Pin 8, or the RTCM (Radio Technical Commission Marine) pin takes in SC-104 format for the reception of differential corrections [41]. RTCM input pin allows for Differential GPS (DGPS) correction message [43], which is transmitted through another serial port on the M12+. Since the current FDR does not use any DGPS correction messages, the pin can be left floating. Finally, the antenna bias voltage is applied to pin 9 and this pin can accept any voltage from +2.5 to 5.5 Vdc [44]. So either a +5V or +3.3V supply from the MPC555 can generate the voltage for this pin.

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Signal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TxD1</td>
<td>Transmit Data (3V logic)</td>
</tr>
<tr>
<td>2</td>
<td>RxD1</td>
<td>Receive Commands (3V logic)</td>
</tr>
<tr>
<td>3</td>
<td>+3V PWR</td>
<td>Regulated 3Vdc Input</td>
</tr>
<tr>
<td>4</td>
<td>1PPS</td>
<td>1 pulse-per-second output</td>
</tr>
<tr>
<td>5</td>
<td>Ground</td>
<td>Signal and Power common</td>
</tr>
<tr>
<td>6</td>
<td>Battery</td>
<td>Optional External Backup</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>Not currently used</td>
</tr>
<tr>
<td>8</td>
<td>RTCM In</td>
<td>RTCM correction input</td>
</tr>
<tr>
<td>9</td>
<td>Antenna Bias</td>
<td>3V-5V antenna bias input</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>Not currently used</td>
</tr>
</tbody>
</table>
As a conclusion, a PB555 prototype for GPS interface can be implemented using wire wrapping. But due to time constraints on this project, a prototype for GPS interface was not implemented. Nevertheless, once the proposed GPS interface is implemented on the PB555 and confirmed to be able to provide the 1PPS and timestamps, the final PCB can use the same circuit and have a 10 pins port to mount the M12+ or the M12M receiver.

### 6.7 Signal Conditioning Unit and ADC Interface

With the GPS and LCD interface established for the PB555 prototype, the next step would be integrating the signal conditioning circuit and ADC into the FDR board. Since the signal conditioning unit is only consist of resistors and capacitors (see Appendix C), it can be easily implemented in the prototype area [2] of the PB555 board for initial testing phase. However, when the final version of the PCB has been established, the circuit needs to be soldered to the board. On the other hand, the AD976A
analog to digital converter has a more complex circuitry. But for initial testing purposes, the ADC circuitry can be simply implemented with wire wrapping since the current ADC unit is on a PCB with the necessary circuits provided. See Appendix C for ADC board circuitry and Figure 29 for AD976A circuitry. Also, the ADC board has header pins on the bottom that allows for wire wrapping. Once again, when the ADC goes through the testing phase, the final FDR board will have a port specifically for mounting of the ADC board.

As a conclusion, the ADC and signal conditioning circuit can be interfaced to the PB555 in the same way as the current CME555 interface (see Appendix C). However, once the wire wrapped prototypes have gone through the testing phase, the final PCB will need to have an ADC port and the signal conditioning circuit soldered on board.
7.1 Conclusions

In this thesis, the need for a new FDR hardware design is discussed. Furthermore, the requirements for the new design are detailed and the approach to a final design has been proposed as well. The thesis goes through the early stages of PCB design starting from scratch and eventually proposes a more practical method of PCB design. The proposed method involves the use a simple MPC555-based development board, the Axiom PB555 for establishing the required circuitry for MPC555 operation. With the existence of PB555, there is no need to start the PCB design from the bottom level. Instead, the research work can be advanced to the interfacing of external devices. Specifically, the LCD and GPS interface needs to be established with additional circuitry but the ADC and signal conditioning unit can be interfaced to the board using the current design. In addition, the interface between the devices can be tested by wire wrapping a PB555 prototype. However, due to time constraints only the LCD interface has been implemented on the PB555 board and it is still going through testing stages. The idea for establishing the final PCB will be based on how the PB555 prototype is implemented. Once the interface has been verified in implementation, a final board design can be established. Appendix C shows schematics for potential final board design based on the PB555 board and Appendix E has the netlistings for the final board design based on the overall thesis work.

7.2 Future Work

This thesis has proposed a practical approach to the FDR PCB design. However, due to time constraints some of the interfaces did not get implemented in hardware. In order to confirm to a final PCB design, it is necessary to verify each of the interfaces by implementing a wire wrapped prototype. Currently, there is a prototype built for LCD interface and it is still undergoing testing stages. Once the LCD interface is confirmed, a prototype needs to be implemented for each of the interfacing involved, this include the GPS and the ADC and signal conditioning unit. A final design of the PCB will be determined by the results from the prototype implementations.
REFERENCES


MPC555UM October 15, 2000, pp. 2-12 to 2-28.


http://www.analog.com/UploadedFiles/Data_Sheets/61476620AD976_A_c.pdf
61476620AD976_A_c rev. C 1999

Report of OK Industries.


http://www.nabble.com/MPC555-HARDWARE-DESIGN-CHECKLIST-t531476.html


http://chipcatalog.com/Datasheet/F8B7C2C896160E42EE694AF61510A907.htm
F8B7C2C896160E42EE694AF61510A907 December, 2000.


MPC555UM October 15, 2000, pp. 7-7 to 7-8.
MPC555UM October 15, 2000, pp. 8-12.
MPC555UM October 15, 2000, pp. 22-1 to 22-3.
MPC555UM October 15, 2000, pp. 19-1 to 19-2.
[33] Maxim Integrated Products, Sunnyvale, California. “3.0V to 5.5V, Low-Power, up
to 1Mbps, True RS-232 Transceivers Using Four 0.1uF External Capacitors”,
[34] Hitachi, Tokyo, Japan. “HD44780U Dot Matrix Liquid Crystal Display
Controller/Driver”, http://www.electronic-
ing工程/microchip/datasheets/lcd/hd44780.pdf hd44780, pp. 25.
Controller/Driver”, http://www.electronic-
ing工程/microchip/datasheets/lcd/hd44780.pdf hd44780, pp. 10.
[36] Hitachi, Tokyo, Japan. “HD44780U Dot Matrix Liquid Crystal Display
Controller/Driver”, http://www.electronic-
ing工程/microchip/datasheets/lcd/hd44780.pdf hd44780, pp. 44.
[37] Hitachi, Tokyo, Japan. “HD44780U Dot Matrix Liquid Crystal Display
Controller/Driver”, http://www.electronic-
ing工程/microchip/datasheets/lcd/hd44780.pdf hd44780, pp. 46.


APPENDIX A
Initialization and configuration code for PB555

1. Linker Command File for CME555 (modified for PB555)

/* original declaration */
_flash_source = 0x00c10000;  // **NOTE: MUST match RAM buffer setting in linker preference panel */

/* modification for PB made by Lei */
_flash_source = 0x00010100;

MEMORY {
/* original declaration */
  ram : org = 0x00c02000
  rom : org = 0x00000000 // desired ROM address (boot address for 555)
/* modification for PB made by Lei */
  ram : org = 0x003FA000
  rom : org = 0x00010100
}

/* FORCEFILES are used so that the linker will not deadstrip the file reset.s. The function reset would be deadstripped since it is not ever called by anything */
FORCEACTIVE { gInterruptVectorTable, __reset }

SECTIONS {
  .reset : {} > rom
  .init : {} > rom
  GROUP : {
    .text (TEXT) ALIGN(0x1000) : {}
    .rodata (CONST) : {
      *(.rdata)
      *(.rodata)
    }
    .ctors : {}
    .dtors : {}
    extab : {}
    extabindex : {}
  } > rom  // for ROM images, this can be 'rom' if you want to execute in ROM
  // or 'code' if you want to execute in RAM
  GROUP : {
    .data : {}
    .sdata : {}
    .sbss : {}
    .sdata2 : {}
    .sbss2 : {}
    .bss : {}
    .PPC.EMB.sdata0 : {}
    .PPC.EMB.sbss0 : {}
  } > ram

60
// The dummy section is just a placeholder. The linker automatically
// generates an address for it in the ROM image, which tells us
// where the end of the ROM image is.
_dummy ALIGN(64): {}

_flash_dest = _f_reset; // true flash address starts w/.init

_flash_size = _f_dummy_rom - _flash_dest;
// The .fcopy section contains a small piece of code that copies the
// ROM image to flash. We don't copy the .fcopy section itself to flash
// because it could erase the flash if it were accidentally executed
// at a later time.

// Bind it to the address it will occupy in the RAM buffer so we can
// execute it directly from the RAM buffer.
.fcopy BIND(_flash_source + _flash_size) ALIGN(64): {
  *(.fcopy)
}

.fcopy_data : {}
// The internal flash algorithms provided by Motorola are
// packaged in a binary file. The linker includes the contents
// in the .BINARY section.
.BINARY : {}

2. Runtime initialization file (single chip operation)

/*
  56X_Chip_init.c

  This is the chip initialization file for the MPC56X series
*/
//file used for single chip operation – added by Lei

#pragma section code_type ".init"

#define SIUMCR 0x2fc000
#define SYPCR 0x2fc004
#define SIPEND 0x2fc010
#define SIMASK 0x2fc014
#define SIIEL 0x2fc018
#define SIVEC 0x2fc01c
#define PLPRCR 0x2fc284

//SPR defines
#define SPR_SRR1 27
#define SPR_ICTRL              158
#define SPR_IMMR    638
#define SPR_BBCMCR   560

#ifdef __cplusplus
extern "C" {
#endif
asm void __reset(void);
asm void usr_init();
extern void __start();

#ifdef __cplusplus
}
#endif
asm void __reset(void)
{
    //
    //   Enable machine check exceptions
    //
    lis r3, 0
    ori r3, r3, 0x1002
    mtmsr r3

    //
    // ROM is not relocated, so nothing left to do here.
    //

    b __start
}

asm void usr_init()
{
    nofralloc

    // Adr 0x002fc384 = 0x55ccaa33  PLPRCRK: open key
    lis r5, 0x55cc
    ori r5, r5, 0xaa33
    lis r4, 0x002fc384@ha
    stw r5, 0x002fc384@l(r4)

    // Adr 0x002fc284 = 0x00900000  PLPRCR
    lis r5, 0x0090
    ori r5, r5, 0x0000
    lis r4, 0x002fc284@ha
stw r5, 0x002fc284@l(r4)

// SPR MSR = 0x00003002 MSR
lis r5, 0x0000
ori r5, r5, 0x3002
mtmsr r5

// SPR 27 = 0x00003002 SSR1
lis r5, 0x0000
ori r5, r5, 0x3002
mtspr 27, r5

// SPR 638 = 0xffff0800 IMMR
lis r5, 0xffff0
ori r5, r5, 0x0800
mtspr 638, r5

// Adr 0x002fc000 = 0x00000000 SIUMCR
lis r5, 0x0000
ori r5, r5, 0x0000
lis r4, 0x002fc000@ha
stw r5, 0x002fc000@l(r4)

// Adr 0x002fc004 = 0x0000ff88 SYPCR
lis r5, 0x0000
ori r5, r5, 0xff88
lis r4, 0x002fc004@ha
stw r5, 0x002fc004@l(r4)

// SPR 158 = (SPR 158 & 0xffffffff) | 0x00000007 ICTRL: switch serialised mode off
lis r5, 0xffffffff
ori r5, r5, 0xffffffff
mfspr r4, 158
and r4, r4, r5
lis r5, 0x0000
ori r5, r5, 0x0007
or r4, r4, r5
mtspr 158, r4

// Adr 0x002fc380 = 0x55ccaa33 SCCRK: open key
lis r5, 0x55cc
ori r5, r5, 0xaaa33
lis r4, 0x002fc380@ha
stw r5, 0x002fc380@l(r4)

// Adr 0x002fc280 = 0x00010000 SCCR
lis       r5, 0x0001
ori       r5, r5, 0x0000
lis       r4, 0x002fc280@ha
stw       r5, 0x002fc280@l(r4)

// Adr 0x00305014 = 0x0000  PORTQS
li         r5, 0x0000
lis       r4, 0x00305014@ha
sth       r5, 0x00305014@l(r4)

// Adr 0x00305016 = 0x0000  PQSPAR/DDRQS
li         r5, 0x0000
lis       r4, 0x00305016@ha
sth       r5, 0x00305016@l(r4)

// Adr 0x00307f80 = 0x00000000  UMCR
lis       r5, 0x0000
ori       r5, r5, 0x0000
lis       r4, 0x00307f80@ha
stw       r5, 0x00307f80@l(r4)

3. Memory configuration file (single chip operation)

// 56x_chip.mem - memory config file for MPC56x
// The CodeWarrior debugger uses this file to allow the proper display of memory.
// This is for internal memory only. If using external memory,
// add the appropriate range command.
reservedchar 0xBA
// Flash area (UC3F flash array)
rangerange 0x00000000 0x000FFFFF 4 Read
reserved 0x00100000 0x002F7FFF
// DECRAM SRAM
range 0x002F8000 0x002F8FFF 4 ReadWrite
reserved 0x002F9000 0x002FA003
// BBC
range 0x002FA004 0x002FA00F 4 ReadWrite
range 0x002FA010 0x002FA013 2 ReadWrite
range 0x002FA014 0x002FA03F 4 ReadWrite
reserved 0x002FA040 0x002FBFFF
// USIU
range 0x002FC000 0x002FC007 4 ReadWrite
reserved 0x002FC008 0x002FC00D
range 0x002FC00E 0x002FC00F 2 Write
range 0x002FC010 0x002FC01B 4 ReadWrite
range 0x002FC01C 0x002FC01F 4 Read
range 0x002FC020 0x002FC033 4 ReadWrite
<table>
<thead>
<tr>
<th>Address Range</th>
<th>Size</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x002FC034-0x002FC03B</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC038-0x002FC057</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC100-0x002FC11F</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC120-0x002FC13F</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC140-0x002FC147</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC178-0x002FC179</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC17A-0x002FC1FF</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC200-0x002FC201</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC202-0x002FC203</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC204-0x002FC20B</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC20C-0x002FC21F</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC220-0x002FC221</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC222-0x002FC223</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC224-0x002FC22F</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC230-0x002FC23F</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC240-0x002FC241</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC242-0x002FC243</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC244-0x002FC247</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC248-0x002FC24B</td>
<td>4</td>
<td>Read</td>
</tr>
<tr>
<td>0x002FC24C-0x002FC27F</td>
<td></td>
<td>Read</td>
</tr>
<tr>
<td>0x002FC280-0x002FC287</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC288-0x002FC289</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC28A-0x002FC28B</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC28C-0x002FC28D</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC28E-0x002FC28F</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC290-0x002FC291</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC292-0x002FC2FF</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC300-0x002FC30F</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC310-0x002FC31F</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC320-0x002FC32F</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC330-0x002FC33F</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC340-0x002FC347</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC348-0x002FC37F</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC380-0x002FC38B</td>
<td>4</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x002FC38C-0x002FC7FF</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x00300000-0x0030000B</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x0030000C-0x0030003F</td>
<td></td>
<td>ReadWrite</td>
</tr>
<tr>
<td>0x00300040-0x0030004B</td>
<td>2</td>
<td>ReadWrite</td>
</tr>
</tbody>
</table>

// Flash Control Registers (UC3F_A)

// DPT RAM Control Registers

// AB

// C
reserved 0x0030004C 0x0030007F
// DLCM
range 0x00300080 0x00300081 2 ReadWrite
reserved 0x00300082 0x00300083
range 0x00300080 0x0030008F 2 ReadWrite
reserved 0x00300090 0x00300FFF
// DPT RAM Array
// NOTE: Access to DPT RAM Array is disabled once TPUMCR[EMU]=1
range 0x00301000 0x00303FFF 2 ReadWrite
// TPU3
// NOTE: Most TPU3 registers also allow 32 bit access
range 0x00304000 0x0030402F 2 ReadWrite
reserved 0x00304030 0x003040FF
range 0x003040100 0x003041FF 2 ReadWrite
reserved 0x00304100 0x003041FF 2 ReadWrite
reserved 0x00304200 0x003042FF
range 0x00304400 0x0030442F 2 ReadWrite
reserved 0x00304430 0x003044FF
range 0x00304500 0x003045FF 2 ReadWrite
reserved 0x00304600 0x003047FF
// QADC64
range 0x00304800 0x00304813 2 ReadWrite
reserved 0x00304813 0x003049FF
range 0x003049A00 0x00304C13 2 ReadWrite
reserved 0x00304C14 0x00304DFF
range 0x00305000 0x0030500F 2 ReadWrite
reserved 0x00305010 0x00305013
range 0x00305014 0x0030501D 2 ReadWrite
range 0x0030501E 0x0030501F 1 ReadWrite
range 0x00305020 0x0030506B 2 ReadWrite
reserved 0x0030506C 0x0030513F
range 0x00305140 0x003051DF 2 ReadWrite
reserved 0x003051F0 0x003053FF
// QSMCM_B
range 0x00305400 0x00305413 2 ReadWrite
reserved 0x00305414 0x0030541D 2 ReadWrite
range 0x0030541E 0x0030541F 1 ReadWrite
range 0x00305420 0x0030546B 2 ReadWrite
reserved 0x0030546C 0x0030553F
range 0x00305540 0x0030555F 2 ReadWrite
reserved 0x00305560 0x003055FF
// TPU3
range 0x00305C00 0x00305C2F 2 ReadWrite
reserved 0x00305C30 0x00305CFF
range 0x00305D00 0x00305DFF 2 ReadWrite
reserved 0x00305E00 0x00305FFF
// MIOS14
range  0x00306000 0x00306047 2 ReadWrite
reserved 0x00306048 0x0030604F
range  0x00306050 0x003060C7 2 ReadWrite
reserved 0x003060C8 0x003060D7
range  0x003060D8 0x00306103 2 ReadWrite
reserved 0x00306104 0x003067FF
range  0x00306800 0x00306807 2 ReadWrite
reserved 0x00306808 0x00306815
range  0x00306816 0x00306817 2 ReadWrite
reserved 0x00306818 0x00306BFF
range  0x00306C00 0x00306C01 2 ReadWrite
reserved 0x00306C02 0x00306C03
range  0x00306C04 0x00306C07 2 ReadWrite
reserved 0x00306C08 0x00306C2F
range  0x00306C30 0x00306C31 2 ReadWrite
reserved 0x00306C32 0x00306C3F
range  0x00306C40 0x00306C41 2 ReadWrite
reserved 0x00306C42 0x00306C43
range  0x00306C44 0x00306C47 2 ReadWrite
reserved 0x00306C48 0x00306C6F
range  0x00306C70 0x00306C71 2 ReadWrite
reserved 0x00306C72 0x0030707F
// TOUCAN
range  0x00307080 0x0030708B 2 ReadWrite
reserved 0x0030708C 0x0030708F
range  0x00307090 0x0030709B 2 ReadWrite
reserved 0x0030709C 0x0030709F
range  0x003070A0 0x003070A7 2 ReadWrite
reserved 0x003070A8 0x003070FF
range  0x00307100 0x003071FF 4 ReadWrite
reserved 0x00307200 0x0030747F
range  0x00307480 0x0030748B 2 ReadWrite
reserved 0x0030748C 0x0030748F
range  0x00307490 0x0030749B 2 ReadWrite
reserved 0x0030749C 0x0030749F
range  0x003074A0 0x003074A7 2 ReadWrite
reserved 0x003074A8 0x003074FF
range  0x00307500 0x003075FF 4 ReadWrite
reserved 0x00307600 0x0030787F
// TouCAN_C
range  0x00307880 0x0030788B 2 ReadWrite
reserved 0x0030788C 0x0030788F
range  0x00307890 0x0030789B 2 ReadWrite
reserved 0x0030789C 0x0030789F
APPENDIX B

LCD DRIVER CODE

1. LCD initialization steps

[Power ON]

[ Wait more than 15ms ]
[after Vdd rises to 4.5v]

RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * * Function set (8-bit interface)

[Wait more than 4.1ms]

RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * * Function set (8-bit interface)

[Wait more than 100us]

RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * * Function set (8-bit interface)

BF can be checked after the following instructions. When BF is not checked, the waiting time between instructions is longer than the execution time. (See Instruction set)

RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
2. Main program for LCD driver

```cpp
#include <iostream>
#include "FNET_LCD.h"

using namespace std;

void main()
{
    USIU.BR3.R = 0x01000403;
    LCD_InitDriver();
    LCD_WriteControl(OnOffCtrl);
    LCD_WriteControl(DisplayClear);
    LCD_CursorOff();
    LCD_WriteControl(E_M_RightIncrm);
    LCD_Clear();
    delay1ms(100);
    LCD_DisplayString(2,1,"Initializing...");
    LCD_DisplayString(2,16,"done!");
    LCD_Clear();
    LCD_DisplayString(1, 1, " Collecting Data...");
}

// Busy wait loop to generate a delay multiple of 1ms
void delay1ms(unsigned int msecs) { 
    unsigned int i,j;
    if (msecs > 0) { 
        for (j=0; j<=msecs; j++) 
            for (i=0; i<DELAY1MS; i++) 
                ;
    }
    return;
} 
```
void LCD_DisplayCharacter (unsigned char a_char) {
    delay1ms(10); //jcp
    LCD_WriteData (a_char);
}

/**
** LCD_DisplayString: Display a string at the specified row and column.
*/
// Note: this fn uses D0 as the start address of column 2 because
// using C0 doesn't work
void LCD_DisplayStringMoving (char row, char column, const char *string) {
    switch (row) {
        case 1:
            LCD_WriteControl (0x80 + column - 1);
            while (*string) {
                delay1ms(3000);
                LCD_DisplayCharacter (*string++);
            }
            break;

        case 2:
            LCD_WriteControl (0xD0 + column - 1);
            while (*string) {
                delay1ms(3000);
                LCD_DisplayCharacter (*string++);
            }
            default: break;
    }
}

void LCD_DisplayString (char row, char column, const char *string) {
    switch (row) {
        case 1:
LCD_WriteControl (0x80 + column - 1);
    while (*string)
        LCD_DisplayCharacter (*string++);
    break;

    case 2:
        LCD_WriteControl (0xC0 + column - 1);
        while (*string)
            LCD_DisplayCharacter (*string++);
    }
}

/*
** LCD_WipeOffLR: "Wipe" screen left-to-right.
*/
void LCD_WipeOffLR (void)
{
    // "wipe" off old screen (left to right)
    char i;
    for (i=0; i<20; i++) {
        LCD_WriteControl (0x80 + i - 1);
        LCD_DisplayCharacter(BLOCK);
        LCD_WriteControl (0xC0 + i - 1);
        LCD_DisplayCharacter(BLOCK);
        delay1ms(800);
    }
}

/*
** LCD_DisplayRow: Display a string at the specified row.
*/
void LCD_DisplayRow (int row, unsigned char *string)
{
    char i;
    LCD_Cursor (row, 1);
    for (i=0; i<20; i++)
        LCD_DisplayCharacter (*string++);
}

/*
** LCD_CursorLeft: Move the cursor left by one character.
*/
void LCD_ShiftLeft (void)
{
LCD_WriteControl (0x1c);
}

/*
** LCD_CursorRight: Move the cursor right by one character.
*/
void LCD_ShiftRight (void)
{
    LCD_WriteControl (0x18);
}

/*
** LCD_CursorOn: Turn the cursor on.
*/
void LCD_CursorOn (void)
{
    LCD_WriteControl (0x0e);
}

/*
** LCD_CursorOff: Turn the cursor off.
*/
void LCD_CursorOff (void)
{
    LCD_WriteControl (0x0c);
}

/*
** LCD_InitDriver: Initialize the LCD driver.
*/
void LCD_InitDriver (void)
{
    delay1ms(500);               // 10ms delay
    LCD_WriteControl(InitWord1); // 38h:8 bits,2 lines, 5*7 dots
    delay1ms(50);                // 10ms delay
    LCD_WriteControl(InitWord1);
    delay1ms(50);                // 10ms delay
    LCD_WriteControl(InitWord1);
    delay1ms(50);
    LCD_WriteControl(InitWord1); // 10ms delay
void LCDWaitForBusy()
{
    unsigned char lcdval;
    do{
        lcdval = get8(LCD_CMD);
    }while((lcdval & 0x80) != 0); // wait for busy bit to go low
}

/*
 ** LCD_WriteControl: Write a control instruction to the LCD
 */
void LCD_WriteControl (unsigned char cmdval)
{
    LCDWaitForBusy();
    put8(LCD_CMD,cmdval);
}

/*
 ** LCD_WriteData: Write one byte of data to the LCD
 */
void LCD_WriteData (unsigned char datval)
{
    LCDWaitForBusy();
    put8(LCD_DAT,datval);
}

/*
 ** LCD_Clear: Clear the LCD screen (also homes cursor).
 */
void LCD_Clear (void)
{
    LCD_WriteControl(0x01);
}

/*
 ** LCD_Home: Position the LCD cursor at row 1, col 1.
 */
void LCD_Home (void)
{
    LCD_WriteControl(0x02);
}

3. LCD driver header file
#ifndef FNET_LCD_H
#define FNET_LCD_H

#include "mpc555.h"

/*----------
    CONSTANTS
----------*/

typedef unsigned char UNS8;  
typedef unsigned short UNS16;    
typedef unsigned long UNS32;   
typedef unsigned char BOOL;    /* unsigned 8 bits */
typedef int SINT32;      /* signed int 32 bits */
typedef unsigned short UINT16; /* unsigned int 16 bits */
typedef short SINT16;      /* signed int 16 bits */
typedef unsigned char UINT8;   /* unsigned int 8 bits */
typedef char SINT8;            /* signed int 8 bits */
typedef volatile unsigned char VUNS8; 
typedef volatile unsigned short VUNS16; 
typedef volatile unsigned long VUNS32; 

typedef unsigned char* UNS8P; 

#define TRUE 1
#define FALSE 0

#define put8(p, x)   (*((volatile UNS8*)(p))) = (x))
#define put16(p, x)  (*((volatile UNS16*)(p))) = (x))
#define put32(p, x)  (*((volatile UNS32*)(p))) = (x))
#define get8(p)      (*((volatile UNS8*)(p)))
#define get16(p)     (*((volatile UNS16*)(p)))
#define get32(p)     (*((volatile UNS32*)(p)))

// begin lcd stuff

#define _IO_BASE 0x01000000
#define LCD_CMD 0x01000000
#define LCD_DAT 0x01000001
#define KEY_DAT 0x01000004

#define LINE_1 0x80    // beginning position of LCD line 1
```c
#define LINE_2 0xC0  // beginning position of LCD line 2
#define LINE_3 0x14  // beginning position of LCD line 3
#define LINE_4 0x54  // beginning position of LCD line 4
#define WaitTime 15
#define DELAY1MS 73  // number of loops = 1ms

// I/O Port Addresses
#define CS3_Base 0x0100
#define LCD_offset 0x0000
#define Keypad_offset 0x0004
#define InitWord1 0x38  // 8 bits, 2 lines, 5*7 dots
#define InitWord2 0x30  // 8 bits 1 line and 5*7 dots
#define InitWord3 0x3c  // 8 bits, 2 lines, 5*10 dots
#define OnOffCtrl 0xe
#define DisplayClear 01
#define E_M_ShiftLeft 07
#define E_M_RightIncrm 06
#define NormalWrite 0x14
#define Init_ScrClear 01  // Clear Display, Cursor to Home  1.65ms
#define CrstoHome 02    // Cursor to Home $02 1.65ms
#define CrsrDcrmnt_Shftoff 04 // Cursor decrement and shift off 40us
#define CrsrDcrmnt_Shfton 05  // Cursor Increment and shift on 40 ns
#define CrsrIncrmnt_Shftoff 06 // Cursor Increment and shift off 40us
#define CrsrIncrmnt_Shfton 07  // Cursor Increment and shift on 40us
#define LineLength 20
#define BLOCK 0xFF
#define Resolution 8  // significants number of the converting double,

void DisplayScreen (unsigned char *ptr);
void DisplayScreenWipeOld (unsigned char *ptr);

/*
** Local (Module-Level) Function Prototypes
*/

// INITIALIZE HARDWARE REGISTERS
// This routine is called once following RESET
void inithw(void);
void InitLCD(void);
void LCDWaitForBusy();
```
void LCD_InitDriver (void);
void LCD_WriteControl (unsigned char cmdval);
void LCD_WriteData (unsigned char datval);
void LCDWaitForBusy();

void LCD_Init (void);
void LCD_DefineChar (unsigned char address, const unsigned char *pattern);
void LCD_DisplayOff (void);
void LCD_DisplayOn (void);
void LCD_Clear (void);
void LCD_Home (void);
void LCD_Cursor (char row, char column);
void LCD_ShiftLeft (void);
void LCD_ShiftRight (void);
void LCD_CursorOn (void);
void LCD_CursorOff (void);
void LCD_DisplayCharacter (unsigned char a_char);
void LCD_DisplayString (char row, char column, const char *string);
void LCD_DisplayStringMoving (char row, char column, const char *string);
void LCD_DisplayStringCentered (char row, char *string);
void LCD_DisplayScreen (unsigned char *ptr);
void LCD_DisplayRow (int row, unsigned char *string);
void LCD_DisplayRowShift (int row, unsigned char *string);
void LCD_WipeOnLR (char *ptr);
void LCD_WipeOnRL (char *ptr);
void LCD_WipeOffLR (void);
void LCD_WipeOffRL (void);
char *__gcvt(double value, int digits, char *buffer);

4. EPLD code
Name            AXM0054;
Partno          CME0555;
Revision        01;
Date            01/04/99;
Designer        Forrester;
Company         Axiom Manufacturing;
Assembly        CME0555 lcd decode;
Location        U21;
Device          g22V10;

/**********************************************************************
/* This device generates the LCD and Key chip select signals          */
/* outputs = LCDCS1/2, KEYEN                                         */
/**********************************************************************
*/
/** Inputs **/
pin 1 = Clk ; /* CPU clock input */
pin 2 = !CS_in ; /* Chip select input */
pin 3 = R_W ; /* Read / Write strobe in */
pin [4..11,13..18] = [A14..A7,A6..A1] ; /* system addresses a2 - a15 */
/** Outputs **/
pin [19..20] = [Q0..Q1] ; /* Clock counter */
pin 21 = L2CS ; /* LCD 2 CS */
pin 22 = L1CS ; /* LCD 1 CS */
pin 23 = KEY ; /* Key CS */
/** Declarations and Intermediate Variable Definitions **/
key_eqn = [A14..A1] : 0004 ; /* keypad address */
lcd1_eqn = [A14..A1] : [0000..0001] ; /* LCD 1 address */
lcd2_eqn = [A14..A1] : [0002..0003] ; /* LCD 2 address */
field count = [Q1..0] ; /* declare counter bit field */
$define S0 'b'00 /* define counter states */
$define S1 'b'01
$define S2 'b'10
$define S3 'b'11
clear = !CS_in ; /* define count clear mode */
up = CS_in ; /* define count direction */
/* LCD wait state generator */
Sequenced count {
present S0 if up next S1;
if clear next S0;
present S1 if up next S2;
if clear next S0;
present S2 if up next S3;
if clear next S0;
present S3 if up next S0;
if clear next S0;
}
L1CS.d = lcd1_eqn & ((Q0 & Q1) # L1CS) ; /* define lcd 1 enable */
L1CS.ar = !CS_in ; /* define async reset */
L2CS.d = lcd2_eqn & ((Q0 & Q1) # L2CS) ; /* define lcd 2 enable */
L2CS.ar = !CS_in ; /* define async resets */
Q0.ar = !CS_in ;
Q1.ar = !CS_in ;
!KEY = key_eqn & R_W & CS_in ; /* Keypad select */

APPENDIX C
SCHEMATICS

1. CME555 schematic
LCD/Keypad interface and RS232 port
2. PB555 schematic with external interface

MPC555
Signal Conditioning Unit and ADC

[Diagram of signal conditioning unit and ADC circuitry, including various components like resistors, capacitors, ICs, and signals like VIN, GND, and VCC.]
GPS and LCD interface
PB555 header pins
**APPENDIX D**

**CME555 JUMPER AND SWITCH SETTING FOR FDR**

### Config Switch

<table>
<thead>
<tr>
<th>Position</th>
<th>Reset Function</th>
<th>Default Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MODCK1 – clock select</td>
<td>Off</td>
</tr>
<tr>
<td>2</td>
<td>MODCK2 – clock select = 4MHz Oscillator</td>
<td>On</td>
</tr>
<tr>
<td>3</td>
<td>MODCK3 – clock select</td>
<td>Off</td>
</tr>
<tr>
<td>4</td>
<td>External Hard Reset Word enable</td>
<td>On</td>
</tr>
<tr>
<td>5</td>
<td>EPEE signal High – on-chip program enable</td>
<td>On</td>
</tr>
<tr>
<td>6</td>
<td>VPP power enable</td>
<td>On</td>
</tr>
</tbody>
</table>

### Mode Switch 1

<table>
<thead>
<tr>
<th>Position</th>
<th>Reset Function</th>
<th>Default Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial interrupt prefix (IP), On = 1</td>
<td>Off</td>
</tr>
<tr>
<td>2</td>
<td>External Boot (BDIS), Disabled = 1</td>
<td>Off</td>
</tr>
<tr>
<td>3</td>
<td>Boot Port size (BPS0), on = 16 bit boot port, off = 32 bit</td>
<td>Off</td>
</tr>
<tr>
<td>4</td>
<td>Debug pins configuration (DBGC0)</td>
<td>Off</td>
</tr>
<tr>
<td>5</td>
<td>External Bus division factor (EBDF1), on = clock/2</td>
<td>Off</td>
</tr>
<tr>
<td>6*</td>
<td>Single Chip Mode Select (SC0:1)</td>
<td>Off</td>
</tr>
<tr>
<td>7*</td>
<td>Single Chip Mode Select (SC0:1)</td>
<td>Off</td>
</tr>
<tr>
<td>8</td>
<td>Exception table relocation enable (ETRE), on = relocated</td>
<td>Off</td>
</tr>
</tbody>
</table>

*Note:*

<table>
<thead>
<tr>
<th>Position 6</th>
<th>Position 7</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Off</td>
<td>Extended 32 bit data bus</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>Extended 16 bit data bus</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>Single chip with show cycles</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>Single chip with no external bus</td>
</tr>
</tbody>
</table>

### Mode Switch 2

<table>
<thead>
<tr>
<th>Position</th>
<th>Reset Function</th>
<th>Default Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Internal Flash enable (FLEN), Off = internal flash removed from memory map</td>
<td>On</td>
</tr>
<tr>
<td>2</td>
<td>Enable Compression (EN_COMP)</td>
<td>Off</td>
</tr>
<tr>
<td>3</td>
<td>Exception Compression Enable (EN_COMP)</td>
<td>Off</td>
</tr>
<tr>
<td>4</td>
<td>Reserved, do not enable</td>
<td>Off</td>
</tr>
<tr>
<td>5</td>
<td>Internal register space select (ISB)</td>
<td>Off</td>
</tr>
<tr>
<td>6*</td>
<td>off:off:off = 0:0:0, register space at 0x00000000</td>
<td>Off</td>
</tr>
<tr>
<td>7*</td>
<td></td>
<td>Off</td>
</tr>
<tr>
<td>8</td>
<td>Dual mapping enable (DME), Off = dual mapping disabled</td>
<td>Off</td>
</tr>
</tbody>
</table>
RAM-SEL JUMPER

1 2 3

FLSH-SEL JUMPER

1 2 3

M-SEL JUMPER

1 2 3

Note: Above jumpers are set to CME555 default settings for FDR

Position 1 = CS0
Position 2 = CS1
Position 3 = CS2

MEM_OPT JUMPERS

1 2 3 4 5 6 7 8

Note:
Position 5 = A16 to U2 pin 29
Position 7 = A16 to U3 pin 29

MEM_VOLT OPTION JUMPER

+5V

1

+3.3V

or

1

+3.3V

Option = +5V

JUMPER 10, JUMPER 11 and JUMPER 12 are installed
JUMPER 6, 8, and 9 are installed
APPENDIX E
NETLISTING FOR FDR PCB DESIGN

1. Netlist for MPC555 connection
(edif Design_FRU_board_simplifiednetlist_SCHDOC
(edifVersion 2 0 0)
(edifLevel 0)
(keywordMap
(keywordLevel 0)
)
(status
(written
(timeStamp 2006 1 11 20 18 22)
(program "Design Explorer DXP - EDIF For PCB"
(version "1.0.0")
)
(author "EDIF For PCB")
)
)

(library COMPONENT_LIB
(edifLevel 0)
(technology
(numberDefinition
(scale 1 1 (unit distance))
)
)
(cell Cap
(cellType GENERIC)
(view netListView
(viewType NETLIST)
(interface
(port (rename &1 "1") (direction INOUT))
(port (rename &2 "2") (direction INOUT))
)
)
)
(cell GND
(cellType GENERIC)
(view netListView
(viewType NETLIST)
(interface
(port Y (direction OUTPUT))
)
)
(cell (rename Header_2 "Header 2")
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INOUT))
      (port (rename &2 "2") (direction INOUT))
    )
  )
)

(cell (rename Header_5X2 "Header 5X2")
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INOUT))
      (port (rename &2 "2") (direction INOUT))
      (port (rename &3 "3") (direction INOUT))
      (port (rename &4 "4") (direction INOUT))
      (port (rename &5 "5") (direction INOUT))
      (port (rename &6 "6") (direction INOUT))
      (port (rename &7 "7") (direction INOUT))
      (port (rename &8 "8") (direction INOUT))
      (port (rename &9 "9") (direction INOUT))
      (port (rename &10 "10") (direction INOUT))
    )
  )
)

(cell Inductor
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INOUT))
      (port (rename &2 "2") (direction INOUT))
    )
  )
)

(cell MPC555
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INOUT))
      (port (rename &2 "2") (direction INOUT))
    )
  )
)
(port (rename &3 "3") (direction INOUT))
(port (rename &4 "4") (direction INOUT))
(port (rename &5 "5") (direction INOUT))
(port (rename &6 "6") (direction INOUT))
(port (rename &7 "7") (direction INOUT))
(port (rename &8 "8") (direction INOUT))
(port (rename &9 "9") (direction INOUT))
(port (rename &10 "10") (direction INOUT))
(port (rename &11 "11") (direction INOUT))
(port (rename &12 "12") (direction INOUT))
(port (rename &13 "13") (direction INOUT))
(port (rename &14 "14") (direction INOUT))
(port (rename &15 "15") (direction INOUT))
(port (rename &16 "16") (direction INOUT))
(port (rename &17 "17") (direction INOUT))
(port (rename &18 "18") (direction INOUT))
(port (rename &19 "19") (direction INOUT))
(port (rename &20 "20") (direction INOUT))
(port (rename &21 "21") (direction INOUT))
(port (rename &22 "22") (direction INOUT))
(port (rename &23 "23") (direction INOUT))
(port (rename &24 "24") (direction INOUT))
(port (rename &25 "25") (direction INOUT))
(port (rename &26 "26") (direction INOUT))
(port (rename &27 "27") (direction INOUT))
(port (rename &28 "28") (direction INOUT))
(port (rename &29 "29") (direction INOUT))
(port (rename &30 "30") (direction INOUT))
(port (rename &31 "31") (direction INOUT))
(port (rename &32 "32") (direction INOUT))
(port (rename &33 "33") (direction INOUT))
(port (rename &34 "34") (direction INOUT))
(port (rename &35 "35") (direction INOUT))
(port (rename &36 "36") (direction INOUT))
(port (rename &37 "37") (direction INOUT))
(port (rename &38 "38") (direction INOUT))
(port (rename &39 "39") (direction INOUT))
(port (rename &40 "40") (direction INOUT))
(port (rename &41 "41") (direction INOUT))
(port (rename &42 "42") (direction INOUT))
(port (rename &43 "43") (direction INOUT))
(port (rename &44 "44") (direction INOUT))
(port (rename &45 "45") (direction INOUT))
(port (rename &46 "46") (direction INOUT))
(port (rename &47 "47") (direction INOUT))
(port (rename &48 "48") (direction INOUT))
(port (rename &49 "49") (direction INOUT))
(port (rename &50 "50") (direction INOUT))
(port (rename &51 "51") (direction INOUT))
(port (rename &52 "52") (direction INOUT))
(port (rename &53 "53") (direction INOUT))
(port (rename &54 "54") (direction INOUT))
(port (rename &55 "55") (direction INOUT))
(port (rename &56 "56") (direction INOUT))
(port (rename &57 "57") (direction INOUT))
(port (rename &58 "58") (direction INOUT))
(port (rename &59 "59") (direction INOUT))
(port (rename &60 "60") (direction INOUT))
(port (rename &61 "61") (direction INOUT))
(port (rename &62 "62") (direction INOUT))
(port (rename &63 "63") (direction INOUT))
(port (rename &64 "64") (direction INOUT))
(port (rename &65 "65") (direction INOUT))
(port (rename &66 "66") (direction INOUT))
(port (rename &67 "67") (direction INOUT))
(port (rename &68 "68") (direction INOUT))
(port (rename &69 "69") (direction INOUT))
(port (rename &70 "70") (direction INOUT))
(port (rename &71 "71") (direction INOUT))
(port (rename &72 "72") (direction INOUT))
(port (rename &73 "73") (direction INOUT))
(port (rename &74 "74") (direction INOUT))
(port (rename &75 "75") (direction INOUT))
(port (rename &76 "76") (direction INOUT))
(port (rename &77 "77") (direction INOUT))
(port (rename &78 "78") (direction INOUT))
(port (rename &79 "79") (direction INOUT))
(port (rename &80 "80") (direction INOUT))
(port (rename &81 "81") (direction INOUT))
(port (rename &82 "82") (direction INOUT))
(port (rename &83 "83") (direction INOUT))
(port (rename &84 "84") (direction INOUT))
(port (rename &85 "85") (direction INOUT))
(port (rename &86 "86") (direction INOUT))
(port (rename &87 "87") (direction INOUT))
(port (rename &88 "88") (direction INOUT))
(port (rename &89 "89") (direction INOUT))
(port (rename &90 "90") (direction INOUT))
(port (rename &91 "91") (direction INOUT))
(port (rename &92 "92") (direction INOUT))
(port (rename &93 "93") (direction INOUT))
(port (rename &94 "94") (direction INOUT))
(port (rename &141 "141") (direction INOUT))
(port (rename &142 "142") (direction INOUT))
(port (rename &143 "143") (direction INOUT))
(port (rename &144 "144") (direction INOUT))
(port (rename &145 "145") (direction INOUT))
(port (rename &146 "146") (direction INOUT))
(port (rename &147 "147") (direction INOUT))
(port (rename &148 "148") (direction INOUT))
(port (rename &149 "149") (direction INOUT))
(port (rename &150 "150") (direction INOUT))
(port (rename &151 "151") (direction INOUT))
(port (rename &152 "152") (direction INOUT))
(port (rename &153 "153") (direction INOUT))
(port (rename &154 "154") (direction INOUT))
(port (rename &155 "155") (direction INOUT))
(port (rename &156 "156") (direction INOUT))
(port (rename &157 "157") (direction INOUT))
(port (rename &158 "158") (direction INOUT))
(port (rename &159 "159") (direction INOUT))
(port (rename &160 "160") (direction INOUT))
(port (rename &161 "161") (direction INOUT))
(port (rename &162 "162") (direction INOUT))
(port (rename &163 "163") (direction INOUT))
(port (rename &164 "164") (direction INOUT))
(port (rename &165 "165") (direction INOUT))
(port (rename &166 "166") (direction INOUT))
(port (rename &167 "167") (direction INOUT))
(port (rename &168 "168") (direction INOUT))
(port (rename &169 "169") (direction INOUT))
(port (rename &170 "170") (direction INOUT))
(port (rename &171 "171") (direction INOUT))
(port (rename &172 "172") (direction INOUT))
(port (rename &173 "173") (direction INOUT))
(port (rename &174 "174") (direction INOUT))
(port (rename &175 "175") (direction INOUT))
(port (rename &176 "176") (direction INOUT))
(port (rename &177 "177") (direction INOUT))
(port (rename &178 "178") (direction INOUT))
(port (rename &179 "179") (direction INOUT))
(port (rename &180 "180") (direction INOUT))
(port (rename &181 "181") (direction INOUT))
(port (rename &182 "182") (direction INOUT))
(port (rename &183 "183") (direction INOUT))
(port (rename &184 "184") (direction INOUT))
(port (rename &185 "185") (direction INOUT))
(port (rename &186 "186") (direction INOUT))
(viewType NETLIST)
(interface
  (port (rename &1 "1") (direction INOUT))
  (port (rename &2 "2") (direction INOUT))
)
)
)
(library SHEET_LIB
  (edfLevel 0)
  (technology
    (numberDefinition
      (scale 1 1 (unit distance))
    )
  )
)
)
cell FRU_board_simplifiednetlist_SCHDOC
  (cellType generic)
  (view netListView
    (viewType netlist)
    (interface)
    (contents
      (Instance (rename UNDEFINED "")
        (viewRef NetlistView
          (cellRef GND
            (LibraryRef COMPONENT_LIB)
          )
        )
      )
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property Class (String "General" ))
      (Property (rename Class_1 "Class I") (String "Passive" ))
      (Property (rename Class_1 "Class I") (String "Passive" ))
      (Property (rename Class_1 "Class I") (String "Passive" ))
    )
  )
)
(Instance Cmpc2
 (viewRef NetlistView
  (cellRef Cap
   (LibraryRef COMPONENT_LIB)
  ))
)

(Property (rename Class_I "Class I") (String "Passive" ))
(Property (rename Class_II "Class II") (String "Capacitor" ))
(Property Comment (String "Cmpc2" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "RAD-0.3" ))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "Cap" ))
(Property Manufacturer (String "Generic Components" ))
(Property Published (String "8-Jun-2000" ))
(Property Publisher (String "Altium Limited" ))
(Property Revision (String "July-2002: Re-released for DXP Platform." ))
(Property (rename Signal_Integrity "Signal Integrity") (String "Cap" ))
(Property Simulation (String "CAP" ))
(Property Value (String "10 pF" ))
(Property Footprint (String "RAD-0.3" ))
(Property Description (String "Capacitor" ))
(Property UniqueId (String "$$$JFJOSIFA" ))
)

(Instance Cmpc3
 (viewRef NetlistView
  (cellRef Cap
   (LibraryRef COMPONENT_LIB)
  ))
)

(Property (rename Class_I "Class I") (String "Passive" ))
(Property (rename Class_II "Class II") (String "Capacitor" ))
(Property Comment (String "Cmpc3" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "RAD-0.3" ))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "Cap" ))
(Property Manufacturer (String "Generic Components" ))
(Property Published (String "8-Jun-2000" ))
(Property Publisher (String "Altium Limited" ))
(Property Revision (String "July-2002: Re-released for DXP Platform." ))
)
(Property (rename Signal_Integrity "Signal Integrity") (String "Cap"))
(Property Simulation (String "CAP"))
(Property Value (String "10uF"))
(Property Footprint (String "RAD-0.3"))
(Property Description (String "Capacitor"))
(Property UniqueId (String "$$$\VEUQFLOU"))
)

(Instance Cmpc4
(viewRef NetlistView
  (cellRef Cap
   (LibraryRef COMPONENT_LIB)
  )
)
(Property (rename Class_I "Class I") (String "Passive"))
(Property (rename Class_II "Class II") (String "Capacitor"))
(Property Comment (String "Cmpc4"))
(Property (rename Component_Kind "Component Kind") (String "Standard"))
(Property Description (String "Capacitor"))
(Property Footprint (String "RAD-0.3"))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib"))
(Property (rename Library_Reference "Library Reference") (String "Cap"))
(Property Manufacturer (String "Generic Components"))
(Property Published (String "8-Jun-2000"))
(Property Publisher (String "Altium Limited"))
(Property Revision (String "July-2002: Re-released for DXP Platform."))
(Property (rename Signal_Integrity "Signal Integrity") (String "Cap"))
(Property Simulation (String "CAP"))
(Property Value (String "0.01uF"))
(Property Footprint (String "RAD-0.3"))
(Property Description (String "Capacitor"))
(Property UniqueId (String "$$$\JTAMHQGF"))
)

(Instance (rename L_ "L?"
(viewRef NetlistView
  (cellRef Inductor
   (LibraryRef COMPONENT_LIB)
  )
)
(Property (rename Class_I "Class I") (String "Passive"))
(Property (rename Class_II "Class II") (String "Inductor"))
(Property Comment (String ""))
(Property (rename Component_Kind "Component Kind") (String "Standard"))
(Property Description (String "Inductor"))
(Property Footprint (String "C1005-0402"))
(Property (rename Signal_Integrity "Signal Integrity") (String "Res1" ))
(Property (rename Signal_Integrity "Signal Integrity") (String "Res1" ))
(Property Simulation (String "RESISTOR" ))
(Property Simulation (String "RESISTOR" ))
(Property Value (String "10K" ))
(Property Footprint (String "AXIAL-0.3" ))
(Property Description (String "Resistor" ))
(Property Uniqueld (String "$$$\YNDVQOOM" ))
)

(Instance (rename VPP_EN_Jumper "VPP_EN Jumper")
  (viewRef NetlistView
    (cellRef Header_2
     (LibraryRef COMPONENT_LIB)
    )
  )
)

(Property (rename Class_I "Class I") (String "Connector" ))
(Property Comment (String "VPP_EN Jumper" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Header, 2-Pin" ))
(Property Footprint (String "HDR1X2" ))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Connectors.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "Header 2" ))
(Property Manufacturer (String "Generic Components" ))
(Property Publisher (String "Altium Limited" ))
(Property Revision (String "July-2002: Re-released for DXP Platform." ))
(Property (rename Signal_Integrity "Signal Integrity") (String "Connector" ))
(Property Footprint (String "HDR1X2" ))
(Property Description (String "Header, 2-Pin" ))
(Property UniqueId (String "$$$\AHHLJKGV" ))
)

(Net (rename PLUS3_3_V "+3.3 V")
  (Joined
    (PortRef &1)
    (PortRef &1)
    (PortRef &1)
  )
)

(Net (rename PLUS5_V "+5 V")
  (Joined
    (PortRef &1)
    (PortRef &1)
    (PortRef &1)
    (PortRef &1)
    (PortRef &1)
  )
)
(Net (rename PLUS5V "+5V")
  (Joined
    (PortRef &2 (InstanceRef PROG_EN Jumper))
    (PortRef &2 (InstanceRef Rmpc1))
    (PortRef &2 (InstanceRef Rmpc2))
    (PortRef &1 (InstanceRef Rmpc4))
    (PortRef &1 (InstanceRef VPP_EN Jumper))
  )
)

(Net (rename &10K "10K")
  (Joined
  )
)

(Net (rename ADDRESS_BUS "ADDRESS BUS")
  (Joined
  )
)

(Net (rename CONTROL_SIGNALS "CONTROL SIGNALS")
  (Joined
  )
)

(Net D0
  (Joined
    (PortRef &209 (InstanceRef MPC555))
  )
)

(Net D1
  (Joined
    (PortRef &210 (InstanceRef MPC555))
  )
)

(Net D2
  (Joined
    (PortRef &211 (InstanceRef MPC555))
  )
)

(Net D3
  (Joined
    (PortRef &212 (InstanceRef MPC555))
  )
)

(Net D4
(Joined
  (PortRef &213 (InstanceRef MPC555))
)
)
(Net D5
 (Joined
   (PortRef &214 (InstanceRef MPC555))
  )
)
(Net D6
 (Joined
   (PortRef &215 (InstanceRef MPC555))
  )
)
(Net D7
 (Joined
   (PortRef &216 (InstanceRef MPC555))
  )
)
(Net (rename DATA_BUS "DATA BUS")
 (Joined
  )
)
(Net DSCK
 (Joined
   (PortRef &4 (InstanceRef BDM-PORT))
  )
)
(Net DSDI
 (Joined
   (PortRef &8 (InstanceRef BDM-PORT))
   (PortRef &2 (InstanceRef Rmpc6))
  )
)
(Net DSDO
 (Joined
   (PortRef &10 (InstanceRef BDM-PORT))
  )
)
(Net ENGCLK
 (Joined
  )
)
(Net EPEE
 (Joined
   (PortRef &1 (InstanceRef PROG_EN Jumper))
  )
)
(PortRef &1 (InstanceRef Rmpc5))
)
)
(Net EXTCLK
 (Joined
   (PortRef &1 (InstanceRef Rmpc7))
 )
)
(Net (rename HRESET_ "HRESET*"))
 (Joined
   (PortRef &7 (InstanceRef BDM-PORT))
 )
)
(Net KAPWR
 (Joined
   (PortRef &2)
   (PortRef &2)
   (PortRef &2)
   (PortRef &2)
   (PortRef &9 (InstanceRef BDM-PORT))
 )
)
(Net MDA11
 (Joined
   (PortRef &1 (InstanceRef Rmpc1))
 )
)
(Net (rename MDA11___MDA31 "MDA11...MDA31")
 (Joined
 )
)
(Net MDA27
 (Joined
   (PortRef &1 (InstanceRef Rmpc2))
 )
)
(Net (rename MGPIO5___MGPIO19 "MGPIO5...MGPIO19")
 (Joined
 )
)
(Net (rename MPWM0___MPWM19 "MPWM0...MPWM19")
 (Joined
 )
)
(Net Net_1
 (Joined
)
(PortRef &1)
  (PortRef &1 (InstanceRef Cmpc1))
  (PortRef &1 (InstanceRef Rmpc9))
)
)
(Net Net_2
  (Joined
    (PortRef &2)
    (PortRef &1 (InstanceRef Cmpc2))
    (PortRef &2 (InstanceRef Rmpc9))
  )
)
)
(Net Net_Y
  (Joined
    (PortRef Y)
    (PortRef &2 (InstanceRef L?))
  )
)
)
(Net Net_Y
  (Joined
    (PortRef Y)
    (PortRef &2 (InstanceRef Rmpc5))
  )
)
)
(Net Net_Y
  (Joined
    (PortRef Y)
    (PortRef &3 (InstanceRef BDM-PORT))
    (PortRef &5 (InstanceRef BDM-PORT))
  )
)
)
(Net Net_Y
  (Joined
    (PortRef Y)
    (PortRef &2 (InstanceRef Rmpc8))
  )
)
)
(Net Net_Y
  (Joined
    (PortRef Y)
    (PortRef &2 (InstanceRef Rmpc7))
  )
)
)
(Net Net_Y
  (Joined
    (PortRef Y)
  )
2. Netlist for ADC and signal conditioning connection
(edif Design_ADCnetlist_SCHDOC
 (edifVersion 2 0 0)
 (edifLevel 0)
 (keywordMap
   (keywordLevel 0)
 )
 (status
   (written
     (timeStamp 2006 1 11 20 18 0)
     (program "Design Explorer DXP - EDIF For PCB"
      (version "1.0.0")
     )
   )
   (author "EDIF For PCB")
 )
)
(library COMPONENT_LIB
 (edifLevel 0)
 (technology
   (numberDefinition
     (scale 1 1 (unit distance))
   )
 )
 (cell (rename &8_HEADER "8 HEADER")
   (cellType GENERIC)
   (view netListView
(viewType NETLIST)
(interface
  (port (rename &1 "1") (direction INOUT))
  (port (rename &2 "2") (direction INOUT))
  (port (rename &3 "3") (direction INOUT))
  (port (rename &4 "4") (direction INOUT))
  (port (rename &5 "5") (direction INOUT))
  (port (rename &6 "6") (direction INOUT))
  (port (rename &7 "7") (direction INOUT))
  (port (rename &8 "8") (direction INOUT))
)
)
)

(cell (rename &74HC240 "74HC240")
(cellType GENERIC)
(view netListView
  (viewType NETLIST)
  (interface
    (port (rename &1 "1") (direction INPUT))
    (port (rename &2 "2") (direction INPUT))
    (port (rename &3 "3") (direction INOUT))
    (port (rename &4 "4") (direction INPUT))
    (port (rename &5 "5") (direction INOUT))
    (port (rename &6 "6") (direction INPUT))
    (port (rename &7 "7") (direction INOUT))
    (port (rename &8 "8") (direction INPUT))
    (port (rename &9 "9") (direction INOUT))
    (port (rename &10 "10") (direction INOUT))
    (port (rename &11 "11") (direction INPUT))
    (port (rename &12 "12") (direction INOUT))
    (port (rename &13 "13") (direction INPUT))
    (port (rename &14 "14") (direction INOUT))
    (port (rename &15 "15") (direction INPUT))
    (port (rename &16 "16") (direction INOUT))
    (port (rename &17 "17") (direction INPUT))
    (port (rename &18 "18") (direction INOUT))
    (port (rename &19 "19") (direction INPUT))
    (port (rename &20 "20") (direction INOUT))
  )
)
)
)

(cell (rename &74HC574 "74HC574")
(cellType GENERIC)
(view netListView
  (viewType NETLIST)
  (interface
(view netListView
  (viewType NETLIST)
  (interface
    (port (rename &1 "1") (direction INOUT))
    (port (rename &2 "2") (direction INOUT))
  )
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
(port (rename &24 "24") (direction INOUT))
(port (rename &25 "25") (direction INOUT))
(port (rename &26 "26") (direction INOUT))
(port (rename &27 "27") (direction INOUT))
(port (rename &28 "28") (direction INOUT))
(port (rename &29 "29") (direction INOUT))
(port (rename &30 "30") (direction INOUT))
(port (rename &31 "31") (direction INOUT))
(port (rename &32 "32") (direction INOUT))
(port (rename &33 "33") (direction INOUT))
(port (rename &34 "34") (direction INOUT))
)
)
)
cell MC1439
(cellType GENERIC)
(view netListView
 (viewType NETLIST)
 (interface
   (port (rename &1 "1") (direction INOUT))
   (port (rename &2 "2") (direction INOUT))
   (port (rename &3 "3") (direction INOUT))
   (port (rename &4 "4") (direction INOUT))
   (port (rename &5 "5") (direction INOUT))
   (port (rename &6 "6") (direction INOUT))
   (port (rename &7 "7") (direction INOUT))
   (port (rename &8 "8") (direction INOUT))
 )
)
)
cell RES1
(cellType GENERIC)
(view netListView
 (viewType NETLIST)
 (interface
   (port (rename &1 "1") (direction INOUT))
   (port (rename &2 "2") (direction INOUT))
 )
)
)
cell Res1
(cellType GENERIC)
(view netListView
 (viewType NETLIST)
 (interface
   (port (rename &1 "1") (direction INOUT))
 )
)
(port (rename &2 "2") (direction INOUT))
)
)
)
)
)
)
(library SHEET_LIB
  (edifLevel 0)
  (technology
    (numberDefinition
      (scale 1 1 (unit distance))
    )
  )
  )
)
(cell ADCnetlist_SCHDOC
  (cellType generic)
  (view netListView
    (viewType netlist)
    (interface
      )
    )
  )
  )
  (contents
    (Instance C1
      (viewRef NetlistView
        (cellRef CAP
          (LibraryRef COMPONENT_LIB)
        )
      )
    )
    (Property Comment (String "2nF" ))
    (Property (rename Component.Kind "Component Kind") (String "Standard" ))
    (Property Description (String "Capacitor" ))
    (Property Footprint (String "RAD0.1" ))
    (Property (rename Library_Name "Library Name") (String "" ))
    (Property (rename Library_Reference "Library Reference") (String "CAP" ))
    (Property Description (String "Capacitor" ))
    (Property Footprint (String "RAD0.1" ))
    )
    )
  )
  )
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "CAP" ))
(Property Footprint (String "RAD0.1" ))
(Property Description (String "Capacitor" ))
(Property UniqueId (String "$$\backslash{7902D3200392751720655237591@8T7EU" ")
)
)
Instance C3
(viewRef NetlistView
  (cellRef CAP
    (LibraryRef COMPONENT_LIB)
  )
)
(Property Comment (String "0.1uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "RAD0.1" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "CAP" ))
(Property Footprint (String "RAD0.1" ))
(Property Description (String "Capacitor" ))
(Property UniqueId (String "$$\backslash{7902D3200392751720655237591@8T7EU" ")
)
)
Instance C4
(viewRef NetlistView
  (cellRef CAP
    (LibraryRef COMPONENT_LIB)
  )
)
(Property Comment (String "0.1uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "RAD0.1" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "CAP" ))
(Property Footprint (String "RAD0.1" ))
(Property Description (String "Capacitor" ))
(Property UniqueId (String "$$\backslash{7902D3200392751720655237591@8T7EU" ")
)
)
Instance C5
(viewRef NetlistView
  (cellRef CAP
    (LibraryRef COMPONENT_LIB)
  )
)
(Property Comment (String "0.1uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Comment (String "0.1uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "RAD0.1" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "CAP" ))
(Property UniqueId (String "$$$\0HY\$NF200392751720655237591NK9A#0" ))
)

(Instance C11
 (viewRef NetlistView
  (cellRef CAP
   (LibraryRef COMPONENT_LIB)
  )
 )
)

(Property Comment (String "0.1uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "RAD0.1" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "CAP" ))
(Property UniqueId (String "$$$\0HY\$NF200392751720655237591QHXPYP" ))
)

(Instance C12
 (viewRef NetlistView
  )
)
(cellRef CAP
   (LibraryRef COMPONENT_LIB)
)

(Property Comment (String "0.1uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "RAD0.1" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "CAP" ))
(Property Footprint (String "RAD0.1" ))
(Property Description (String "Capacitor" ))
(Property UniqueId (String "$$\LXZ6FQ200392751720655237591R02NO8" ))
)

(Instance C13
 (viewRef NetlistView
  (cellRef ELECTRO1
   (LibraryRef COMPONENT_LIB)
  )
)

(Property Comment (String "10uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Electrolytic Capacitor" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "ELECTRO1" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property Description (String "Electrolytic Capacitor" ))
(Property UniqueId (String "$$\9XWI20200392751720655237591$%ZFID" ))
)

(Instance C14
 (viewRef NetlistView
  (cellRef ELECTRO1
   (LibraryRef COMPONENT_LIB)
  )
)

(Property Comment (String "10uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Electrolytic Capacitor" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "ELECTRO1" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property Description (String "Electrolytic Capacitor" ))

132
(Property UniqueId (String "$$\backslash BBX\backslash NC200392751720655237591QQ5AFJ"
))

) (Instance C15
(viewRef NetlistView
  (cellRef ELECTRO1
    (LibraryRef COMPONENT_LIB)
  )
)
)
(Property Comment (String "10uF")
(Property (rename Component_Kind "Component Kind") (String "Standard")
(Property Description (String "Electrolytic Capacitor")
(Property Footprint (String "CRB-.2/.4")
(Property (rename Library_Name "Library Name") (String ""))
(Property (rename Library_Reference "Library Reference") (String "ELECTRO1")
)
(Property Footprint (String "CRB-.2/.4")
(Property Description (String "Electrolytic Capacitor")
(Property UniqueId (String "$$\backslash T609\backslash Z200392751720655237591YO3M7J"))
)
) (Instance C16
(viewRef NetlistView
  (cellRef CAP
    (LibraryRef COMPONENT_LIB)
  )
)
)
(Property Comment (String "CAP")
(Property (rename Component_Kind "Component Kind") (String "Standard")
(Property Description (String "Capacitor")
(Property Footprint (String "1206")
(Property (rename Library_Name "Library Name") (String ""))
(Property (rename Library_Reference "Library Reference") (String "CAP")
(Property Footprint (String "1206")
(Property Description (String "Capacitor")
(Property UniqueId (String "$$\backslash 3FZ\backslash Q2003101854237347965R@@TV1"))
)
) (Instance C17
(viewRef NetlistView
  (cellRef ELECTRO1
    (LibraryRef COMPONENT_LIB)
  )
)
)
(Property Comment (String "10uF")
(Property (rename Component_Kind "Component Kind") (String "Standard")
(Property Description (String "Electrolytic Capacitor")
(Property Footprint (String "CRB-.2/.4"))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "ELECTRO1" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property Description (String "Electrolytic Capacitor" ))
(Property UniqueId (String "$$\backslash\text{ANZND4200392751720655237591LR@40\%}""))
)
)
)
(Instance C18
(viewRef NetlistView
 (cellRef ELECTRO1
   (LibraryRef COMPONENT_LIB)
 )
 )
)
(Property Comment (String "10uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Electrolytic Capacitor" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "ELECTRO1" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property Description (String "Electrolytic Capacitor" ))
(Property UniqueId (String "$$\backslash\text{4T8TDU200392751720655237591ALM3CO}" ))
)
)
(Instance C19
(viewRef NetlistView
 (cellRef CAPACITOR
   (LibraryRef COMPONENT_LIB)
 )
 )
)
(Property Comment (String "2.2uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Capacitor" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property (rename Library_Name "Library Name") (String "" ))
(Property (rename Library_Reference "Library Reference") (String "CAPACITOR" ))
(Property Footprint (String "CRB-.2/.4" ))
(Property Description (String "Capacitor" ))
(Property UniqueId (String "$$\backslash\text{1U11DF200392751720655237591BMB5DU}" ))
)
)
(Instance C20
(viewRef NetlistView
 (cellRef CAPACITOR
   (LibraryRef COMPONENT_LIB)
 )
)
(Net D3
  (Joined
    (PortRef &5 (InstanceRef U2))
    (PortRef &19 (InstanceRef U5))
  )
)
(Net D4
  (Joined
    (PortRef &6 (InstanceRef U2))
    (PortRef &18 (InstanceRef U5))
  )
)
(Net D5
  (Joined
    (PortRef &7 (InstanceRef U2))
    (PortRef &17 (InstanceRef U5))
  )
)
(Net D6
  (Joined
    (PortRef &8 (InstanceRef U2))
    (PortRef &16 (InstanceRef U5))
  )
)
(Net D7
  (Joined
    (PortRef &9 (InstanceRef U2))
    (PortRef &15 (InstanceRef U5))
  )
)
(Net D8
  (Joined
    (PortRef &2 (InstanceRef U3))
    (PortRef &13 (InstanceRef U5))
  )
)
(Net D9
  (Joined
    (PortRef &3 (InstanceRef U3))
    (PortRef &12 (InstanceRef U5))
  )
)
(Net D10
  (Joined
    (PortRef &4 (InstanceRef U3))
    (PortRef &11 (InstanceRef U5))
  )
)
(Net D11
  (Joined
    (PortRef &5 (InstanceRef U3))
    (PortRef &10 (InstanceRef U5))
  )
)
(Net D12
  (Joined
    (PortRef &6 (InstanceRef U3))
    (PortRef &9 (InstanceRef U5))
  )
)
(Net D13
  (Joined
    (PortRef &7 (InstanceRef U3))
    (PortRef &8 (InstanceRef U5))
  )
)
(Net D14
  (Joined
    (PortRef &8 (InstanceRef U3))
    (PortRef &7 (InstanceRef U5))
  )
)
(Net D15
  (Joined
    (PortRef &9 (InstanceRef U3))
    (PortRef &6 (InstanceRef U5))
  )
)
(Net GND
  (Joined
    (PortRef &2 (InstanceRef C2))
    (PortRef &2 (InstanceRef C3))
    (PortRef &2 (InstanceRef C4))
    (PortRef &2 (InstanceRef C5))
    (PortRef &2 (InstanceRef C9))
    (PortRef &2 (InstanceRef C12))
    (PortRef &2 (InstanceRef C13))
    (PortRef &2 (InstanceRef C16))
    (PortRef &2 (InstanceRef C18))
    (PortRef &2 (InstanceRef C21))
    (PortRef &2 (InstanceRef C22))
    (PortRef &2 (InstanceRef C23))
  )
)
(Net PPS
  (Joined
    (PortRef &5 (InstanceRef JP1))
    (PortRef &8 (InstanceRef JP2))
  )
)

(Net Q0
  (Joined
    (PortRef &19 (InstanceRef JP1))
    (PortRef &19 (InstanceRef U2))
  )
)

(Net Q1
  (Joined
    (PortRef &20 (InstanceRef JP1))
    (PortRef &18 (InstanceRef U2))
  )
)

(Net Q2
  (Joined
    (PortRef &21 (InstanceRef JP1))
    (PortRef &17 (InstanceRef U2))
  )
)

(Net Q3
  (Joined
    (PortRef &22 (InstanceRef JP1))
    (PortRef &16 (InstanceRef U2))
  )
)

(Net Q4
  (Joined
    (PortRef &23 (InstanceRef JP1))
    (PortRef &15 (InstanceRef U2))
  )
)

(Net Q5
  (Joined
    (PortRef &24 (InstanceRef JP1))
    (PortRef &14 (InstanceRef U2))
  )
)

(Net Q6
  (Joined

(PortRef &25 (InstanceRef JP1))
(PortRef &13 (InstanceRef U2))
)
)
(Net Q7
(Joined
 (PortRef &26 (InstanceRef JP1))
(PortRef &12 (InstanceRef U2))
)
)
)
(Net Q8
(Joined
 (PortRef &27 (InstanceRef JP1))
(PortRef &19 (InstanceRef U3))
)
)
)
(Net Q9
(Joined
 (PortRef &28 (InstanceRef JP1))
(PortRef &18 (InstanceRef U3))
)
)
)
(Net Q10
(Joined
 (PortRef &29 (InstanceRef JP1))
(PortRef &17 (InstanceRef U3))
)
)
)
(Net Q11
(Joined
 (PortRef &30 (InstanceRef JP1))
(PortRef &16 (InstanceRef U3))
)
)
)
(Net Q12
(Joined
 (PortRef &31 (InstanceRef JP1))
(PortRef &15 (InstanceRef U3))
)
)
)
(Net Q13
(Joined
 (PortRef &32 (InstanceRef JP1))
(PortRef &14 (InstanceRef U3))
)
)
)
(Net Q14
  (Joined
   (PortRef &33 (InstanceRef JP1))
   (PortRef &13 (InstanceRef U3))
  )
)

(Net Q15
  (Joined
   (PortRef &34 (InstanceRef JP1))
   (PortRef &12 (InstanceRef U3))
  )
)

(Net RC
  (Joined
   (PortRef &11 (InstanceRef JP1))
   (PortRef &24 (InstanceRef U5))
  )
)

(Net VANA
  (Joined
   (PortRef &1 (InstanceRef C10))
   (PortRef &1 (InstanceRef C17))
   (PortRef &2 (InstanceRef R4))
   (PortRef &27 (InstanceRef U5))
  )
)

(Net VCC
  (Joined
   (PortRef &1 (InstanceRef C2))
   (PortRef &1 (InstanceRef C3))
   (PortRef &1 (InstanceRef C4))
   (PortRef &1 (InstanceRef C5))
   (PortRef &1 (InstanceRef C9))
   (PortRef &1 (InstanceRef C12))
   (PortRef &1 (InstanceRef C13))
   (PortRef &1 (InstanceRef C16))
   (PortRef &1 (InstanceRef C18))
   (PortRef &1 (InstanceRef C21))
   (PortRef &1 (InstanceRef C22))
   (PortRef &1 (InstanceRef C23))
   (PortRef &1 (InstanceRef C24))
   (PortRef &3 (InstanceRef JP2))
   (PortRef &20 (InstanceRef U2))
   (PortRef &20 (InstanceRef U3))
   (PortRef &28 (InstanceRef U5))
   (PortRef &20 (InstanceRef U6))
  )
)
3. **Netlist for LCD and GPS connection**

```
(design Design_ADCnetlist_SCHDOC
  (cellRef ADCnetlist_SCHDOC
    (libraryRef SHEET_LIB)
  )
)
```

---

152
(library COMPONENT_LIB
  (edifLevel 0)
  (technology
    (numberDefinition
      (scale 1 1 (unit distance))
    )
  )
)

(cell (rename ATF22LV10CZ_25PI "ATF22LV10CZ-25PI")
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INPUT))
      (port (rename &2 "2") (direction INPUT))
      (port (rename &3 "3") (direction INPUT))
      (port (rename &4 "4") (direction INPUT))
      (port (rename &5 "5") (direction INPUT))
      (port (rename &6 "6") (direction INPUT))
      (port (rename &7 "7") (direction INPUT))
      (port (rename &8 "8") (direction INPUT))
      (port (rename &9 "9") (direction INPUT))
      (port (rename &10 "10") (direction INPUT))
      (port (rename &11 "11") (direction INPUT))
      (port (rename &12 "12") (direction INOUT))
      (port (rename &13 "13") (direction INPUT))
      (port (rename &14 "14") (direction INOUT))
      (port (rename &15 "15") (direction INOUT))
      (port (rename &16 "16") (direction INOUT))
      (port (rename &17 "17") (direction INOUT))
      (port (rename &18 "18") (direction INOUT))
      (port (rename &19 "19") (direction INOUT))
      (port (rename &20 "20") (direction INOUT))
      (port (rename &21 "21") (direction INOUT))
      (port (rename &22 "22") (direction INOUT))
      (port (rename &23 "23") (direction INOUT))
      (port (rename &24 "24") (direction INOUT))
    )
  )
)

(cell Cap
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INOUT))
      (port (rename &2 "2") (direction INOUT))
    )
  )
)
(cell (rename Header_7X2 "Header 7X2")
  (cellType GENERIC)
  (view netListView
   (viewType NETLIST)
   (interface
    (port (rename &1 "1") (direction INOUT))
    (port (rename &2 "2") (direction INOUT))
    (port (rename &3 "3") (direction INOUT))
    (port (rename &4 "4") (direction INOUT))
    (port (rename &5 "5") (direction INOUT))
    (port (rename &6 "6") (direction INOUT))
    (port (rename &7 "7") (direction INOUT))
    (port (rename &8 "8") (direction INOUT))
    (port (rename &9 "9") (direction INOUT))
    (port (rename &10 "10") (direction INOUT))
    (port (rename &11 "11") (direction INOUT))
    (port (rename &12 "12") (direction INOUT))
    (port (rename &13 "13") (direction INOUT))
    (port (rename &14 "14") (direction INOUT))
   )
  )
)
(cell (rename Header_12X2A "Header 12X2A")
  (cellType GENERIC)
  (view netListView
   (viewType NETLIST)
   (interface
    (port (rename &1 "1") (direction INOUT))
    (port (rename &2 "2") (direction INOUT))
    (port (rename &3 "3") (direction INOUT))
    (port (rename &4 "4") (direction INOUT))
    (port (rename &5 "5") (direction INOUT))
    (port (rename &6 "6") (direction INOUT))
    (port (rename &7 "7") (direction INOUT))
    (port (rename &8 "8") (direction INOUT))
    (port (rename &9 "9") (direction INOUT))
   )
  )
)
(port (rename &10 "10") (direction INOUT))
(port (rename &11 "11") (direction INOUT))
(port (rename &12 "12") (direction INOUT))
(port (rename &13 "13") (direction INOUT))
(port (rename &14 "14") (direction INOUT))
(port (rename &15 "15") (direction INOUT))
(port (rename &16 "16") (direction INOUT))
(port (rename &17 "17") (direction INOUT))
(port (rename &18 "18") (direction INOUT))
(port (rename &19 "19") (direction INOUT))
(port (rename &20 "20") (direction INOUT))
(port (rename &21 "21") (direction INOUT))
(port (rename &22 "22") (direction INOUT))
(port (rename &23 "23") (direction INOUT))
(port (rename &24 "24") (direction INOUT))
)
)
)
)
(cell RPot1
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INOUT))
      (port (rename &2 "2") (direction INOUT))
      (port (rename &3 "3") (direction INOUT))
    )
  )
)
)
)
(cell Res1
  (cellType GENERIC)
  (view netListView
    (viewType NETLIST)
    (interface
      (port (rename &1 "1") (direction INOUT))
      (port (rename &2 "2") (direction INOUT))
    )
  )
)
)
)
(library SHEET_LIB
  (editLevel 0)
  (technology
    (numberDefinition
      (scale 1 1 (unit distance))
    )
  )
)
(cell Sheet3 new_SCHDOC
  (cellType generic)
  (view netListView
   (viewType netlist)
   (interface
    )
  )
  (contents
   (Instance (rename UNDEFINED "")
    (viewRef NetlistView
     (cellRef GND
      (LibraryRef COMPONENT_LIB)
     )
    )
   )
   (Property Class (String "General")
   (Property Class (String "General")
   (Property Class (String "General")
   (Property Class (String "General")
   (Property Class (String "General")
   (Property Class (String "General")
   (Property Class (String "General")
   (Property Class (String "General")
   (Property (rename Class_1 "Class I") (String "Discrete")
   (Property (rename Class_2 "Class II") (String "Diode - Switching")
   (Property Comment (String "")
   (Property Comment (String "")
   (Property Comment (String "1N4148")
   (Property Comment (String "")
   (Property Comment (String "")
   (Property Comment (String "")
   (Property Comment (String "")
   (Property Comment (String "")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property (rename Component_Kind "Component Kind") (String "Standard")
   (Property Description (String "Ground")
   (Property Description (String "Ground")
   (Property Description (String "Default Diode")
   (Property Description (String "Ground"))
)
(Property Description (String "Ground" ))
(Property Description (String "Ground" ))
(Property Description (String "Ground" ))
(Property Description (String "Ground" ))
(Property Description (String "Ground" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property FPGAVendor (String "Actel" ))
(Property Footprint (String "DSO-C2/X3.3" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Name "Library Name") (String "Actel 3200DX FPGA.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "GND" ))
(Property (rename Library_Reference "Library Reference") (String "GND" ))
(Property (rename Library_Reference "Library Reference") (String "Diode" ))
(Property (rename Library_Reference "Library Reference") (String "GND" ))
(Property (rename Library_Reference "Library Reference") (String "GND" ))
(Property (rename Library_Reference "Library Reference") (String "GND" ))
(Property (rename Library_Reference "Library Reference") (String "GND" ))
(Property (rename Library_Reference "Library Reference") (String "GND" ))
(Property Manufacturer (String "Generic Components" ))
(Property Published (String "4/17/2002 3:26:26 PM" ))
(Property Published (String "4/17/2002 3:26:26 PM" ))
(Property Published (String "8-Jun-2000" ))
(Property Published (String "4/17/2002 3:26:26 PM" ))
(Instance Cpot1
  (viewRef NetlistView
   (cellRef Cap2
    (LibraryRef COMPONENT_LIB)
  )
)

(Property (rename Class_I "Class I") (String "Passive" ))
(Property (rename Class_II "Class II") (String "Capacitor") )
(Property Comment (String "10uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard") )
(Property Description (String "Capacitor") )
(Property Footprint (String "CAPR5-4X5" ))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "Cap2") )
(Property Manufacturer (String "Generic Components") )
(Property Published (String "23-Sep-2002") )
(Property Publisher (String "Altium Limited") )
(Property (rename Signal_Integrity "Signal Integrity") (String "Cap") )
(Property Simulation (String "CAP") )
(Property Value (String "16V") )
(Property Footprint (String "CAPR5-4X5") )
(Property Description (String "Capacitor") )
(Property Uniqueld (String "$$$\LRLWEVNV" ))
)

(Instance Cpot2
  (viewRef NetlistView
   (cellRef Cap
    (LibraryRef COMPONENT_LIB)
  )
)

(Property (rename Class_I "Class I") (String "Passive" ))
(Property (rename Class_II "Class II") (String "Capacitor") )
(Property Comment (String "10uF" ))
(Property (rename Component_Kind "Component Kind") (String "Standard") )
(Property Description (String "Capacitor") )
(Property Footprint (String "VR5") )
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "Cap") )
(Property Manufacturer (String "Generic Components") )
(Property Published (String "8-Jun-2000") )
(Property Publisher (String "Altium Limited") )
(Property Revision (String "July-2002: Re-released for DXP Platform." )
(Property (rename Signal_Integrity "Signal Integrity") (String "Cap") )

160
(LibraryRef COMPONENT_LIB)
)
)

(Property (rename Class_I "Class I") (String "Passive" ))
(Property (rename Class_II "Class II") (String "Resistor" ))
(Property Comment (String "" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Resistor" ))
(Property Footprint (String "DSO-C2/X3.3" ))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "Res1" ))
(Property Manufacturer (String "Generic Components" ))
(Property Published (String "8-Jun-2000" ))
(Property Publisher (String "Altium Limited" ))
(Property Revision (String "July-2002: Re-released for DXP Platform." ))
(Property (rename Signal_Integrity "Signal Integrity") (String "Res1" ))
(Property Simulation (String "RESISTOR" ))
(Property Value (String "10K" ))
(Property Footprint (String "DSO-C2/X3.3" ))
(Property Description (String "Resistor" ))
(Property Uniqueld (String "$\\TEGNYTOE" ))
)

(Instance Repld
  (viewRef NetlistView
    (cellRef Res1
      (LibraryRef COMPONENT_LIB)
    )
  )
)

(Property (rename Class_I "Class I") (String "Passive" ))
(Property (rename Class_II "Class II") (String "Resistor" ))
(Property Comment (String "10K" ))
(Property (rename Component_Kind "Component Kind") (String "Standard" ))
(Property Description (String "Resistor" ))
(Property Footprint (String "DIP-24/X1.5" ))
(Property (rename Library_Name "Library Name") (String "Miscellaneous Devices.IntLib" ))
(Property (rename Library_Reference "Library Reference") (String "Res1" ))
(Property Manufacturer (String "Generic Components" ))
(Property Published (String "8-Jun-2000" ))
(Property Publisher (String "Altium Limited" ))
(Property Revision (String "July-2002: Re-released for DXP Platform." ))
(Property (rename Signal_Integrity "Signal Integrity") (String "Res1" ))
(Property Simulation (String "RESISTOR" ))
(Property Value (String "" ))
(Property Footprint (String "DIP-24/X1.5" ))

(Net A28
  (Joined
    (PortRef &16 (InstanceRef ATF22LV10CZ-25PI))
  )
)
(Net A29
  (Joined
    (PortRef &17 (InstanceRef ATF22LV10CZ-25PI))
  )
)
(Net A30
  (Joined
    (PortRef &18 (InstanceRef ATF22LV10CZ-25PI))
  )
)
(Net (rename CS3_ "CS3*")
  (Joined
    (PortRef &2 (InstanceRef ATF22LV10CZ-25PI))
  )
)
(Net D7
  (Joined
    (PortRef &7 (InstanceRef LCD DISPLAY))
  )
)
(Net LCDCS1
  (Joined
    (PortRef &22 (InstanceRef ATF22LV10CZ-25PI))
    (PortRef &6 (InstanceRef LCD DISPLAY))
  )
)
(Net LCDCS2
  (Joined
    (PortRef &21 (InstanceRef ATF22LV10CZ-25PI))
  )
  (Property SuppressERC (String "True" ))
)
(Net (rename MDA11___MDA31 "MDA11...MDA31")
  (Joined
  )
)
(Net MDA15
  (Joined
    (PortRef &4 (InstanceRef GPS CONNECTOR))
  )
)
(Net (rename MGPI05___MGPI019 "MGPI05...MGPI019")
 (Joined
 )
)

(Net (rename MPWM0___MPWM19 "MPWM0...MPWM19")
 (Joined
 )
)

(Net (rename NetATF22LV10CZ_25PI_23 "NetATF22LV10CZ-25PI_23")
 (Joined
   (PortRef &23 (InstanceRef ATF22LV10CZ-25PI))
   (PortRef &1 (InstanceRef Repld))
 )
)

(Net NetCpot1_1
 (Joined
   (PortRef &1 (InstanceRef Cpot1))
   (PortRef &2 (InstanceRef ICL7660))
 )
)

(Net NetCpot1_2
 (Joined
   (PortRef &2 (InstanceRef Cpot1))
   (PortRef &4 (InstanceRef ICL7660))
 )
)

(Net NetCpot2_2
 (Joined
   (PortRef &2 (InstanceRef Cpot2))
   (PortRef &5 (InstanceRef ICL7660))
   (PortRef &2 (InstanceRef TRIMPOT))
 )
)

(Net (rename NetLCD_DISPLAY_3 "NetLCD DISPLAY_3")
 (Joined
   (PortRef &3 (InstanceRef LCD DISPLAY))
   (PortRef &3 (InstanceRef TRIMPOT))
 )
)

(Net Net_2
 (Joined
   (PortRef &2)
   (PortRef Y)
   (PortRef &24 (InstanceRef QS3861))
 )
)
(PortRef &1 (InstanceRef Rconv))
)
)
(Net Net_Y
  (Joined
   (PortRef Y)
   (PortRef &12 (InstanceRef QS3861))
  )
)
)
(Net Net_Y
  (Joined
   (PortRef Y)
   (PortRef &5 (InstanceRef GPS CONNECTOR))
  )
)
)
(Net Net_Y
  (Joined
   (PortRef Y)
   (PortRef &12 (InstanceRef ATF22LV10CZ-25PI))
  )
)
)
(Net Net_Y
  (Joined
   (PortRef Y)
   (PortRef &1 (InstanceRef LCD DISPLAY))
  )
)
)
(Net Net_Y
  (Joined
   (PortRef Y)
   (PortRef &1 (InstanceRef Cpot2))
  )
)
)
(Net Net_Y
  (Joined
   (PortRef Y)
   (PortRef &1 (InstanceRef ICL7660))
  )
)
)
(Net (rename R_W_ "R/W*")
  (Joined
   (PortRef &3 (InstanceRef ATF22LV10CZ-25PI))
  )
)
)
(Net RXD2
  (Joined